Blue Trace User Manual

WHITE HAT WES CYBERSECURITY
WESLEY WIDNER

VERSION 2.0.0

Table of Contents

Introduction

Blue Trace is a powerful, analyst-driven Windows artifact collection tool designed to streamline digital forensics, incident response, system health monitoring, and compliance verification. Built for speed and clarity, Blue Trace empowers security professionals, IT administrators, and forensic analysts to quickly gather, analyze, and report on critical Windows system artifacts—all with a single click.

Primary Purpose

The primary purpose of Blue Trace is to automate the collection and organization of vital user, system, network, and security artifacts from Windows devices. Whether responding to a security incident, monitoring system health, or ensuring compliance, Blue Trace provides a modular and customizable scanning experience, complete with real-time dashboards, historical scan tracking, and support for a wide range of export formats.

With Blue Trace, users can:

Conduct thorough investigations by collecting a comprehensive set of forensic artifacts.

Visualize device health and security status through an interactive dashboard.

Customize or use preconfigured scan profiles tailored to specific needs such as incident response, networking, system health, or compliance.

Export results in multiple formats for analysis, reporting, and record-keeping.

Blue Trace is engineered by analysts for analysts, combining reliability, speed, and flexibility to meet the evolving needs of modern digital investigations.

Scan Options

Account Lockouts

Information Pulled:

- TimeCreated: The timestamp when each account lockout event was generated (formatted as yyyy-MM-dd HH:mm:ss)
- User: The username associated with the lockout event (extracted from the event message if available)
- Message: The full event log message text (condensed for readability)
- Section: Static identifier labeling the data as "AccountLockouts"

Purpose & Usefulness:

- This function retrieves and parses recent account lockout events (Event ID 4740) from the Windows Security Event Log.
- TimeCreated and User allow for identification of when and for which account lockouts occurred, aiding in timeline and user-specific analysis.
- Message provides detailed context for each lockout event, supporting forensic investigations and detection of brute-force attacks, misconfigurations, or suspicious account activity.
- Collecting account lockout data is important for security monitoring, incident response, and compliance, as repeated or unexplained lockouts can be an indicator of attempted unauthorized access or account misuse.

Alternate Data Streams

Information Pulled:

- File: The full file system path to each file found on the Desktop (and subfolders)
- Stream: The name of the alternate data stream (ADS) associated with each file (excluding the default data stream)
- Length: The size (in bytes) of each alternate data stream
- SHA256: The SHA-256 cryptographic hash of the main file's contents
- Section: Static identifier labeling the data as "AlternateDataStreams"

Purpose & Usefulness:

- This function scans all files on the Desktop (including subfolders) for alternate data streams (ADS), which are hidden streams of data that can be attached to files on NTFS file systems.
- File and Stream allow for identification of where non-standard data is stored, which is significant because ADS are often used to hide malicious code, scripts, or data from standard file listings.
- Length provides the size of each ADS, helping assess their potential impact or contents.
- SHA256 serves as a unique identifier for the main file, supporting further integrity or threat analysis.

- Collecting ADS information is important for security auditing, incident response, and digital forensics, as attackers and malware commonly leverage ADS for stealthy persistence or data concealment.

Amcache Check

Information Pulled:

- File: The name of the file checked ("Amcache.hve")
- Path: The full file path to Amcache.hve
- Exists: Indicates whether the file was found ("Yes" or "No")
- Note: Description indicating the file is a binary registry hive and suggesting use of an external parser (such as AmcacheParser)
- Section: Static identifier labeling the data as "AmcacheCheck"

Purpose & Usefulness:

- This function checks for the presence of the Amcache.hve registry hive, which contains historical data about executables, installers, and device drivers that have run on the system.
- Amcache.hve is a valuable forensic artifact that records program execution and can be parsed to build a timeline of software usage and installation.
- Path and Exists help determine if this artifact is available for deeper forensic analysis.
- Note guides the analyst to appropriate tools for parsing, since the file is in binary format.
- Collecting Amcache information is important for digital forensics, incident response, and threat hunting, as it often preserves evidence of execution even after files or logs have been deleted.

App Compat Cache

Information Pulled:

- Section: Static identifier labeling the data as "AppCompatCache"
- Note: Text indicating whether the AppCompatCache registry value exists, or if it is not available/inaccessible

Purpose & Usefulness:

- This function checks for the presence of the AppCompatCache registry key, which stores Windows Application Compatibility Cache (ShimCache) data.
- The AppCompatCache is a valuable forensic artifact that contains execution evidence of programs on the system, even if the binaries no longer exist.
- The function does not parse the binary data itself, but notes the presence or absence of the cache for follow-up analysis (typically with specialized tools like Volatility or ShimCacheParser).

- Collecting this information is important for digital forensics, incident response, and timeline reconstruction, as AppCompatCache can provide evidence of historical program execution that might not be recorded elsewhere.

Application Event Log

Information Pulled:

- TimeCreated: The timestamp when each Application event log entry was generated (formatted as yyyy-MM-dd HH:mm:ss)
- EventID: The numeric ID of the event (e.g., 1000 for application error, 11707 for application installed)
- EventIDMeaning: A friendly description of the event type (e.g., "Application Error / Crash", "Application Installed", or "Unknown")
- Level: The severity or level of the event (e.g., Information, Warning, Error)
- Message: The event log message text (condensed for readability)
- Section: Static identifier labeling the data as "ApplicationEventLog"

Purpose & Usefulness:

- This function retrieves and summarizes recent entries from the Windows Application Event Log.
- TimeCreated and Level support quick filtering and timeline analysis of application-related events.
- EventID and EventIDMeaning help analysts rapidly understand the relevance and type of each event for efficient triage and troubleshooting.
- Message supplies detailed context for application errors, installation events, runtime issues, and software licensing actions, supporting incident response and compliance audits.
- Collecting and interpreting Application event logs is vital for detecting software failures, installation issues, and .NET/runtime errors on Windows systems.

BamDam Activity

Information Pulled:

- UserSID: The Security Identifier (SID) of the user profile associated with each activity entry
- Executable: The name (usually path) of the executable that was monitored
- LastUsed: The timestamp when the executable was last used, formatted as yyyy-MM-dd HH:mm:ss (or "Unreadable Timestamp" if conversion fails)
- Section: Static identifier labeling the data as "BamDamActivity"

Purpose & Usefulness:

- This function parses the Background Activity Moderator (BAM) registry data, which tracks per-user executable usage and last access times.
- UserSID and Executable provide a mapping of which user ran which executables, and when.

- LastUsed supplies precise timing of process execution, which is useful for building user activity timelines and identifying potentially suspicious or rare executable launches.
- Collecting BAM/DAM activity is valuable for digital forensics, incident response, and security auditing, as it may reveal hidden or short-lived process executions that are not captured by normal logging mechanisms.

BitLocker Status

Information Pulled:

- MountPoint: The drive letter or mount point for each volume
- VolumeType: The type of volume (e.g., Fixed, Removable)
- ProtectionStatus: Indicates if BitLocker protection is on, off, or suspended
- EncryptionMethod: The algorithm/method used for encryption
- EncryptionPercentage: How much of the volume is currently encrypted
- KeyProtector: The types of BitLocker key protectors present (e.g., password, TPM, recovery key)
- Section: Static identifier labeling the data as "BitLockerStatus"

Purpose & Usefulness:

- This function collects the encryption status for all detected disk volumes using BitLocker.
- Details such as ProtectionStatus, EncryptionMethod, and EncryptionPercentage are vital for determining if data on the system is protected at rest.
- KeyProtector types indicate how encryption keys are managed, which impacts recoverability and attack surface.
- This information is critical for compliance validation, digital forensics, and incident response, providing insight into whether sensitive data may be protected or exposed if drives are removed or stolen.

Clipboard History

Information Pulled:

- Name: The name of each file or item found in the Windows Clipboard history storage directory
- FullPath: The full file system path to each clipboard history item
- Size: The size (in bytes) of each clipboard history file
- Modified: The last modified timestamp of each clipboard history file (formatted as yyyy-MM-dd HH:mm:ss)
- Section: Static identifier labeling the data as "ClipboardHistory"

Purpose & Usefulness:

- This function enumerates the contents of the Windows Clipboard history storage directory, collecting metadata for each item.

- Name and FullPath provide precise identification and location of clipboard history items, supporting targeted analysis or retrieval.
- Size and Modified offer insights into the recency and potential relevance of clipboard usage, aiding timeline construction for user activity.
- Collecting clipboard history metadata is valuable for digital forensics, user activity monitoring, and incident response, as clipboard data may include sensitive information (such as passwords, copied files, or confidential data) that was recently accessed or transferred.

COM Hijacking Entries

Information Pulled:

- CLSIDPath: The registry path for each COM objectâ€™s CLSID key
- DLLPath: The path to the DLL file registered for that CLSIDâ€™s InprocServer32 (the file loaded by the COM object)
- SHA256: The SHA-256 hash of the DLL file (if found and accessible)
- IsSigned: Boolean indicating if the DLL file is digitally signed and valid
- SignatureStatus: The status of the DLLâ€™s digital signature (e.g., Valid, NotSigned, CheckFailed)
- Publisher: The publisher or subject of the DLLâ€™s signing certificate (if available)
- Suspicious: Boolean flag indicating if the DLL path is potentially suspicious (e.g., located in AppData, Temp, Roaming, Downloads, or is unsigned)
- Section: Static identifier labeling the data as "COMHijackingEntries"

Purpose & Usefulness:

- This function scans CLSID entries in the registry for both the current user and the system, extracting DLL paths and related metadata for each registered COM object.
- CLSIDPath and DLLPath help identify which DLLs are set to load for COM objects, which is crucial for detecting COM hijacking attacks (a stealthy persistence technique).
- SHA256, IsSigned, SignatureStatus, and Publisher enable integrity checks and help distinguish trusted system DLLs from potentially malicious ones.
- Suspicious flags quickly highlight risky or non-standard DLL locations and unsigned files, which may indicate unauthorized persistence or code injection.
- Collecting this data is important for digital forensics, incident response, and security audits, as malicious actors may register their own DLLs for auto-loading via COM object hijacking.

Credential Manager Items

Information Pulled:

- Target: The identifier or resource name for the credential (e.g., server, share, or application name)
- Username: The username associated with the saved credential

- Retrieved: The date and time the credential entry was retrieved (formatted as yyyy-MM-dd HH:mm:ss)
- Section: Static identifier labeling the data as "CredentialManager"

Purpose & Usefulness:

- This function retrieves entries from the Windows Credential Manager using the `cmdkey` command, collecting relevant details about saved credentials.
- Target and Username reveal what resources (network shares, remote systems, applications) the user has stored credentials for, which is useful for mapping access relationships and potential attack paths.
- Retrieved provides a timestamp for when the data was collected, supporting timeline and audit requirements.
- Collecting Credential Manager entries is important for digital forensics, security assessments, and incident response, as saved credentials can reveal persistence mechanisms, third-party access, or targets for lateral movement.

Desktop File Timestamps

Information Pulled:

- Name: The name of each file on the user's Desktop
- FullPath: The full file system path to each Desktop file
- LastModified: The last modified timestamp for each file (formatted as yyyy-MM-dd HH:mm:ss)
- SHA256: The SHA-256 cryptographic hash of each file's contents
- Section: Static identifier labeling the data as "DesktopFileTimestamps"

Purpose & Usefulness:

- This function gathers detailed information about all files located on the user's Desktop.
- Name and FullPath identify where files are located and what they are called, which can be important in tracking user activity and understanding file organization.
- LastModified records when files were last changed, which is valuable for forensic analysis, incident response, or detecting recent suspicious activity.
- SHA256 provides a unique hash for each file, enabling reliable file identification, detecting tampering, or checking for known malicious files by hash comparison.
- Collecting this information helps monitor critical user-facing files, supports investigations into data exfiltration or modification, and assists in compliance with security policies regarding sensitive files.

DLL Search Order Hijacks

Information Pulled:

- DLLName: The file name of each DLL found in target directories

- Path: The full file system path to each DLL
- LastWritten: The last modified timestamp of the DLL (formatted as yyyy-MM-dd HH:mm:ss)
- SHA256: The SHA-256 hash of the DLL file
- IsSigned: Boolean indicating if the DLL is digitally signed and valid
- SignatureStatus: The status of the digital signature (e.g., Valid, NotSigned, CheckFailed)
- Publisher: The publisher or subject of the signing certificate (if available)
- Suspicious: Boolean flag indicating if the DLL is potentially suspicious (e.g., found in risky folders or unsigned)
- Section: Static identifier labeling the data as "DLLSearchOrderHijacks"

Purpose & Usefulness:

- This function scans common and high-risk directories for DLL files, collecting metadata that helps detect DLL search order hijacking risks.
- DLLName and Path identify DLL files and their locations, which is important because DLL search order hijacks occur when malicious DLLs are placed in locations loaded preferentially by Windows or specific applications.
- LastWritten provides file timeline information, aiding in forensic analysis and detecting recent unauthorized changes.
- SHA256, IsSigned, SignatureStatus, and Publisher enable file integrity checks and help distinguish trusted DLLs from potentially malicious ones.
- Suspicious flags highlight unsigned DLLs or those found in user-writable or high-risk directories, supporting prioritization in investigations.
- Collecting this data is crucial for security audits, incident response, digital forensics, and threat hunting, as DLL search order hijacking is a common persistence and code injection technique used by attackers.

DNS Cache

Information Pulled:

- RecordName: The domain name queried and cached by the system DNS resolver
- RecordType: The DNS record type (e.g., "A (Host Address)", "CNAME (Canonical Name)", "AAAA (IPv6 Address)", or "Other")
- TTL: The remaining Time To Live (in seconds) for the cache entry
- Data: The record data (such as IP address for an A record, or target hostname for a CNAME)
- Section: Static identifier labeling the data as "DNSCache"

Purpose & Usefulness:

- This function parses the system's DNS resolver cache, displaying all cached DNS records with their type, data, and expiry.

- RecordName, RecordType, and Data provide visibility into recent and current domain resolutions by the system, supporting investigation of user and process network activity.
- TTL helps analysts understand when the cached entry is set to expire, useful for timeline analysis.
- Collecting DNS cache information is valuable for digital forensics, incident response, and network monitoring, as it can reveal evidence of connections to suspicious, malicious, or noteworthy domainsâ€”even if browser or application history has been cleared.

Downloads Folder

Information Pulled:

- Name: The name of each file found in the Downloads folder
- Path: The full path to each file
- SizeKB: The size of each file in kilobytes
- Modified: The last modified timestamp of each file
- SHA256: The SHA-256 hash of each file (or an error message if hash generation fails)
- Section: Static identifier labeling the data as "DownloadsFolder"

Purpose & Usefulness:

- This function recursively scans the user's Downloads folder, collecting metadata and hashes for all files found.
- Name, Path, SizeKB, and Modified provide critical triage and evidence information for each download, helping identify when files were received and their details.
- SHA256 ensures files can be uniquely identified, correlated with threat intelligence, and checked for tampering or malicious content.
- Collecting Downloads folder data is important for digital forensics, incident response, and malware investigations, as this location is a common source for new, suspicious, or harmful files downloaded by the user or attacker.

Environment Variables

Information Pulled:

- Name: The name of each environment variable
- Value: The value assigned to each environment variable
- Scope: Specifies whether the variable is set at the "User" (current user only) or "System" (all users on the machine) level
- Section: Static identifier labeling the data as "EnvironmentVariables"

Purpose & Usefulness:

- This function gathers all environment variables defined for both the user and the system.
- Name and Value provide details about the runtime configuration, paths, and settings that can influence how applications behave on the system.

- Scope indicates whether each variable affects just the current user or all users, which is important for understanding the variableâ€™s impact and for troubleshooting.
- Collecting environment variables is valuable for incident response, troubleshooting software issues, detecting malicious persistence methods, and auditing system configurations. Unusual or suspicious variables may indicate compromise or misconfiguration.

File Metadata

Information Pulled:

- Name: The name of each file in the user's profile directory (recursively)
- FullPath: The complete file system path to each file
- CreationTime: The timestamp when each file was created (formatted as yyyy-MM-dd HH:mm:ss)
- LastModified: The timestamp when each file was last modified (formatted as yyyy-MM-dd HH:mm:ss)
- LastAccessed: The timestamp when each file was last accessed (formatted as yyyy-MM-dd HH:mm:ss)
- SHA256: The SHA-256 cryptographic hash of each file's contents
- Section: Static identifier labeling the data as "FileMetadata"

Purpose & Usefulness:

- This function collects comprehensive metadata for all files within the user's profile directory, including subdirectories.
- Name and FullPath allow precise identification and location of files, which is vital for investigations and audits.
- CreationTime, LastModified, and LastAccessed provide a complete activity timeline for each file, useful for forensic analysis, incident response, and detecting unusual or suspicious behavior (such as unauthorized access or changes).
- SHA256 supplies a unique cryptographic fingerprint for each file, enabling file integrity checks, tampering detection, and correlation with known malicious files or threat intelligence databases.
- This detailed file inventory supports data loss prevention, compliance efforts, and investigations into user activity or potential security incidents.

Files Accessed Last 14 Days

Information Pulled:

- FullName: The full file system path to each file accessed within the last 14 days on the C: drive
- LastAccessTime: The timestamp when each file was last accessed
- SizeKB: The size of each file in kilobytes (rounded to two decimals)
- LastWriteTime: The timestamp when each file was last modified

- Created: The creation timestamp of each file
- SHA256: The SHA-256 cryptographic hash of each file's contents
- Section: Static identifier labeling the data as "FilesAccessedLast14Days"

Purpose & Usefulness:

- This function collects metadata about every file on the C: drive that has been accessed within the last 14 days.
- FullName provides the precise file locations, enabling investigators to see exactly which files have been recently interacted with.
- LastAccessTime, LastWriteTime, and Created timestamps give a comprehensive view of file activity and lifecycle, aiding in establishing timelines or detecting unusual access patterns.
- SizeKB helps quickly assess file relevance based on size.
- SHA256 supplies a cryptographic fingerprint for each file, which is useful for checking integrity, tracking changes, or matching files against known threat intelligence databases.
- This information is especially valuable for incident response, data loss prevention, and forensic investigations, as it highlights potentially sensitive or suspicious files that have been recently opened or used.

Firewall Rules

Information Pulled:

- Name: The display name of each Windows Firewall rule
- Enabled: Whether the rule is enabled ("True") or disabled ("False")
- Direction: The rule's direction ("Inbound", "Outbound", or "Unknown")
- Action: The rule's action ("Allow", "Block", or "Unknown")
- Profile: The firewall profile(s) the rule applies to ("Domain", "Private", "Public", "All", or combinations)
- Section: Static identifier labeling the data as "FirewallRules"

Purpose & Usefulness:

- This function enumerates all configured Windows Firewall rules on the system and collects their key properties.
- Name, Direction, and Action give a clear description of what traffic is being allowed or blocked and in which direction.
- Enabled indicates which rules are actively enforced, assisting with compliance and security validation.
- Profile reveals the network environment (Domain, Private, Public) where each rule applies, allowing targeted policy analysis.
- Collecting firewall rule information is essential for security auditing, troubleshooting network issues, verifying compliance, and identifying misconfigurations or unauthorized changes that may impact system security.

Group Policy Results

Information Pulled:

- Section: Distinguishes between "GroupPolicyResults_Computer" and "GroupPolicyResults_User"
- Scope: Indicates if the output is for Computer or User scope
- Output: Full raw text output from `gpresult /scope computer /r` and `gpresult /scope user /r`, showing applied Group Policy Objects (GPOs), settings, and policy inheritance details

Purpose & Usefulness:

- This function gathers the results of applied Group Policy Objects (GPOs) for both the computer and the current user.
- The Output includes security policies, login scripts, software restrictions, auditing configurations, and other enforced settings.
- Provides visibility into which policies are in effect, their sources, and any potential conflicts or overrides.
- Useful for auditing, security reviews, troubleshooting group policy issues, and verifying compliance with organizational baselines.

Hidden Files On C

Information Pulled:

- FullName: The full file system path to each hidden file on the C: drive
- SizeKB: The size of each hidden file in kilobytes (rounded to two decimals)
- Created: The creation timestamp of each hidden file (formatted as yyyy-MM-dd HH:mm:ss)
- Modified: The last modified timestamp of each hidden file (formatted as yyyy-MM-dd HH:mm:ss)
- Accessed: The last accessed timestamp of each hidden file (formatted as yyyy-MM-dd HH:mm:ss)
- SHA256: The SHA-256 cryptographic hash of each file's contents
- Section: Static identifier labeling the data as "HiddenFilesOnC"

Purpose & Usefulness:

- This function searches for and collects metadata about all hidden files on the system's C: drive.
- FullName provides exact locations of hidden files, which are often overlooked and may contain sensitive or suspicious data.
- SizeKB, Created, Modified, and Accessed timestamps allow investigators to analyze the activity, creation, and potential relevance of these hidden files, aiding in forensic timelines and root cause analysis.

- SHA256 enables reliable file identification and integrity checking, supporting detection of known malicious files or tracking unauthorized changes.
- Collecting and analyzing hidden file information is important for security audits, malware investigations, and monitoring for signs of compromise, as hidden files are commonly used by attackers to conceal their presence or data.

Host File

Information Pulled:

- IPAddress: The IP address specified in each non-comment line of the hosts file
- Hostname: The hostname mapped to each IP address entry
- SHA256: The SHA-256 hash of the entire hosts file (or error status)
- Section: Static identifier labeling the data as "HostsFile"

Purpose & Usefulness:

- This function parses the Windows hosts file and collects a mapping of static hostname-to-IP assignments along with a hash of the entire file.
- IPAddress and Hostname reveal custom DNS overrides or redirects, which are critical for identifying potential malware, adware, or manual configuration that could affect network behavior or security.
- SHA256 enables integrity checking, ensuring the hosts file has not been tampered with or altered unexpectedly.
- Collecting hosts file information is important for digital forensics, security auditing, troubleshooting DNS issues, and detecting unauthorized or malicious changes to system name resolution.

Image-Video Inventory

Information Pulled:

- Section: "WERCrashDumps"
- FullName: Full path of the crash dump file found in C:\ProgramData\Microsoft\Windows\WER\ReportQueue
- LastWriteTime: Last modified date/time of the dump file
- Length: Size of the dump file in bytes

Purpose & Usefulness:

- This function collects metadata about crash dump files generated by Windows Error Reporting (WER).
- Helps identify application or system crashes that have occurred recently.
- Provides forensic and troubleshooting value: paths can be provided to analysts for deep-dive investigation.
- Useful for system health monitoring, incident response, and root cause analysis.

Installed Programs

Information Pulled:

- DisplayName: The name of each installed program
- DisplayVersion: The version number of each installed program
- Publisher: The software publisher or vendor
- InstallDate: The date when the program was installed (formatted as yyyy-MM-dd if possible)
- Section: Static identifier labeling the data as "InstalledPrograms"

Purpose & Usefulness:

- This function collects details about all programs installed on the system by querying relevant registry paths for both 32-bit and 64-bit applications, as well as per-user installations.
- DisplayName and DisplayVersion help identify exactly what software is present and which versions are in use, which is essential for vulnerability management, software inventory, compliance audits, or incident response.
- Publisher provides information about the vendor, which can help distinguish legitimate software from potentially unwanted or suspicious programs.
- InstallDate is useful for tracking recent software installations that could be linked to security incidents or unauthorized changes.
- Aggregating this information gives a clear picture of the system's software footprint, supporting system health assessments and security investigations.

IPconfig Display DNS

Information Pulled:

- RecordName: The domain name associated with each DNS cache entry
- RecordType: The type of DNS record (e.g., 1 for A record, 5 for CNAME, 28 for AAAA)
- TTL: The remaining Time To Live for each cache entry (in seconds)
- Data: The record data (e.g., IP address, canonical name)
- Section: Static identifier labeling the data as "Ipconfig/displaydns"

Purpose & Usefulness:

- This function parses the output of `ipconfig /displaydns`, collecting all currently cached DNS records from the Windows DNS resolver.
- RecordName, RecordType, and Data provide visibility into what domains and addresses have been recently resolved and accessed by the system.
- TTL shows how long each record will remain in cache, aiding in timeline analysis.
- Collecting DNS cache entries is important for digital forensics, incident response, and network security monitoring, as it reveals recent domain lookups even if browser or application history has been deleted, supporting investigations into suspicious network activity.

Jump Lists

Information Pulled:

- FileName: The name of each Jump List file found in the AutomaticDestinations folder
- FullPath: The full file system path to each Jump List file
- Modified: The last modified timestamp of each Jump List file (formatted as yyyy-MM-dd HH:mm:ss)
- Section: Static identifier labeling the data as "JumpLists"

Purpose & Usefulness:

- This function enumerates Jump List files from the user's AutomaticDestinations folder, which Windows uses to track recently and frequently accessed files and destinations for applications pinned to the taskbar or Start menu.
- FileName and FullPath allow for precise identification and further analysis of each Jump List, which may contain references to user activity and file access history.
- Modified gives a timeline for when the Jump List was last updated, supporting investigations into recent user actions.
- Collecting Jump List metadata is important for digital forensics, user activity monitoring, and incident response, as these files can reveal valuable information about what documents and applications the user has interacted with, even if other traces have been deleted.

Loaded DLLs

Information Pulled:

- ProcessName: The name of the process in which each DLL/module is loaded
- PID: The process ID for each process
- ModuleName: The name of the loaded DLL or module
- FilePath: The file system path to the DLL or module file
- SHA256: The SHA-256 cryptographic hash of the DLL file (or error indicator)
- Signature: The Authenticode digital signature status of the DLL file (e.g., Valid, Invalid, or error indicator)
- CompanyName: The company name from the DLL's version info (if available)
- SuspiciousPath: Boolean indicating whether the DLL is loaded from potentially suspicious locations (such as AppData, Temp, Roaming, or Downloads)
- Section: Static identifier labeling the data as "LoadedDLLs"

Purpose & Usefulness:

- This function enumerates all DLLs (modules) loaded into running processes, along with relevant metadata for each.
- ProcessName, PID, ModuleName, and FilePath provide precise context on which processes are using which DLLs and where those DLLs reside on disk.

- SHA256 uniquely identifies the DLL for integrity checks, threat intelligence matching, or further forensic analysis.
- Signature and CompanyName assist in distinguishing trusted, signed modules from unsigned or suspicious third-party code.
- SuspiciousPath flags modules loaded from unusual or high-risk directories, which are often used by malware or attackers for persistence or injection.
- Collecting this data is vital for security monitoring, digital forensics, malware detection, and understanding the operational context of system and third-party code in memory.

Logon Events

Information Pulled:

- TimeCreated: The timestamp when each logon event was generated (formatted as yyyy-MM-dd HH:mm:ss)
- EventID: The event log ID (should be 4624, indicating a successful logon)
- Account: The username/account name that logged in (extracted from the event message)
- LogonType: The numeric logon type (e.g., 2 for interactive, 3 for network, etc.)
- LogonID: The unique logon session ID (from the event)
- Message: The full event log message (condensed for readability)
- Section: Static identifier labeling the data as "LogonEvents"

Purpose & Usefulness:

- This function retrieves recent successful logon events (Event ID 4624) from the Security Event Log and extracts key fields for each.
- TimeCreated, Account, LogonType, and LogonID allow you to reconstruct user authentication activity, track specific sessions, and identify logon methods.
- Message provides the raw context for deeper forensic review or correlation.
- Collecting this data is critical for digital forensics, security monitoring, and incident response, as it helps identify legitimate or suspicious logons, detect lateral movement, and support investigations into unauthorized access.

MAC Addresses

Information Pulled:

- InterfaceAlias: The name or alias of each physical network adapter that is currently up
- MACAddress: The MAC (Media Access Control) address of each network adapter
- LinkSpeed: The connection speed of each network adapter
- Section: Static identifier labeling the data as "MACAddresses"

Purpose & Usefulness:

- This function enumerates all physical network adapters that are currently active (status "Up") and collects key information for each.

- InterfaceAlias and MACAddress uniquely identify each physical network interface, supporting asset tracking, network monitoring, and forensic investigations.
- LinkSpeed helps assess network capability and detect unusual network configurations.
- Collecting MAC address and interface data is important for digital forensics, network troubleshooting, inventory management, and security monitoring, as it provides a basis for correlating network activity with specific hardware on the system.

MUI Cache

Information Pulled:

- Executable: The path or name of each executable file recorded in the MUICache
- Description: The descriptive text (application name or title) associated with each executable
- Section: Static identifier labeling the data as "MUICache"

Purpose & Usefulness:

- This function parses the MUICache registry key, which tracks recently executed applications and their user interface display names on a per-user basis.
- Executable provides evidence of what programs have been run by the user, even if the binaries have been deleted or moved.
- Description helps identify each executable by its user-facing application name or title.
- Collecting MUICache entries is important for digital forensics, incident response, and activity reconstruction, as it gives a historical record of program execution and can reveal evidence of suspicious or rare applications having been launched on the system.

Net Local Group

Information Pulled:

- GroupName: The name of each local group found on the system
- Collected: The date and time the group name was collected (formatted as yyyy-MM-dd HH:mm:ss)
- Section: Static identifier labeling the data as "NetLocalGroup"

Purpose & Usefulness:

- This function runs the `net localgroup` command and parses its output to enumerate all local groups on the Windows system.
- GroupName provides a list of local groups, supporting group inventory, privilege review, and potential misuse investigation.
- Collected supplies a timestamp for when the group enumeration occurred, aiding in audit trails and repeatability.
- Collecting local group information is important for digital forensics, security auditing, and incident response, as it helps identify unauthorized, overly privileged, or suspicious groups that may affect the systemâ€™s security posture.

Net User

Information Pulled:

- Username: The name of each local user account found on the system
- Collected: The date and time the username was collected (formatted as yyyy-MM-dd HH:mm:ss)
- Section: Static identifier labeling the data as "NetUser"

Purpose & Usefulness:

- This function runs the `net user` command and parses its output to enumerate all local user accounts on the Windows system.
- Username provides a list of accounts that exist locally, supporting account inventory, privilege review, and potential misuse investigation.
- Collected supplies a timestamp for when the enumeration occurred, aiding in audit trails and repeatability.
- Collecting local user account information is crucial for digital forensics, security auditing, and incident response, as it helps identify unauthorized, dormant, or suspicious user accounts that may present security risks.

Netstat Output

Information Pulled:

- Protocol: The network protocol (TCP or UDP) for the connection
- LocalAddress: The local address and port number used by the connection
- ForeignAddress: The remote (foreign) address and port number to which the connection is established or listening
- State: The state of the connection (e.g., LISTENING, ESTABLISHED, TIME_WAIT), or "N/A" if not applicable (such as with UDP)
- PID: The Process ID associated with the network connection
- Section: Static identifier labeling the data as "NetstatOutput"

Purpose & Usefulness:

- This function collects and parses the output of the `netstat -ano` command, which provides a snapshot of all current TCP and UDP network connections and listening ports, along with their associated process IDs.
- Protocol, LocalAddress, and ForeignAddress provide a full view of current network activity and communication endpoints.
- State helps determine the status of each connection, which is crucial for identifying active, listening, or terminated sessions.
- PID links each network connection to a specific process on the system, supporting further investigation of potentially suspicious or unauthorized network activity.

- Collecting this data is important for network forensics, security auditing, threat hunting, and incident response, as it helps detect malware communications, lateral movement, unauthorized listeners, and exfiltration channels.

Network Interfaces

Information Pulled:

- Name: The name or alias of each network adapter on the system
- Status: The operational status of the adapter (e.g., Up, Down, Disabled)
- MACAddress: The MAC address of the network adapter
- LinkSpeed: The connection speed of the network interface
- InterfaceID: The index or identifier for the network interface
- Section: Static identifier labeling the data as "Network Interfaces"

Purpose & Usefulness:

- This function enumerates all network interfaces present on the system and collects key metadata for each.
- Name, MACAddress, and InterfaceID uniquely identify each network adapter, supporting network inventory, device tracking, and troubleshooting.
- Status and LinkSpeed provide information on which adapters are active and at what speed, which is useful for diagnosing connectivity issues or understanding available network capabilities.
- Collecting network interface data is important for asset management, digital forensics, incident response, and network troubleshooting, as it gives a comprehensive overview of the network hardware available on the system.

NTUSER Dat Check

Information Pulled:

- Username: The user account associated with each NTUSER.DAT file found under C:\Users
- NTUSERPath: The full file system path to the NTUSER.DAT registry hive file
- LastModified: The last modified timestamp of the NTUSER.DAT file (formatted as yyyy-MM-dd HH:mm:ss)
- SizeKB: The size of the NTUSER.DAT file in kilobytes (rounded to two decimals)
- SHA256: The SHA-256 cryptographic hash of the NTUSER.DAT file
- Section: Static identifier labeling the data as "NTUSERDat"

Purpose & Usefulness:

- This function enumerates all NTUSER.DAT files for user profiles on the system, collecting key metadata for each.

- Username and NTUSERPath provide attribution and location of the per-user registry hive, which stores personal settings, user-specific configuration, and sometimes persistence mechanisms.
- LastModified and SizeKB help identify recent changes or abnormal file sizes, which may indicate malware or unauthorized modification of user registry data.
- SHA256 supplies a unique identifier for the file, supporting file integrity checks and forensic comparisons across systems or over time.
- Collecting this information is valuable for digital forensics, incident response, and monitoring for suspicious changes to user-specific registry hives that may impact security or system stability.

Parent Child Process Tree

Information Pulled:

- ProcessName: The name of each process
- ProcessId: The unique process ID (PID) for each process
- ParentProcessId: The process ID of the parent (which process launched this one)
- ExecutablePath: The file system path to the process's executable (if available)
- CommandLine: The command line string used to launch the process (if available)
- CreationDate: The timestamp when the process was started (formatted as yyyy-MM-dd HH:mm:ss or "Unavailable")
- Section: Static identifier labeling the data as "ParentChildProcessTree"

Purpose & Usefulness:

- This function enumerates all processes and reconstructs their parent-child relationships, providing a detailed process tree for the system.
- ProcessName, ProcessId, and ParentProcessId reveal how processes are spawned and linked, which is critical for identifying suspicious or malicious process chains.
- ExecutablePath and CommandLine provide insight into how and from where each process was launched, aiding in security investigations and incident response.
- CreationDate helps build a timeline of process activity, supporting event correlation and anomaly detection.
- Collecting parent-child process data is essential for digital forensics, security monitoring, and troubleshooting, as it exposes potentially unwanted or attacker-controlled process hierarchies and execution paths.

PowerShell History

Information Pulled:

- Timestamp: The last modified time of the PowerShell history file containing the command
- LineNumber: The line number of the command in the history file
- Command: The exact PowerShell command or line that was entered

- Section: Static identifier labeling the data as "PowerShellHistory"

Purpose & Usefulness:

- This function collects and parses PowerShell command history from standard transcript and history files found in the user's profile.
- Timestamp shows when the history file was last updated, helping establish timelines for script or command execution.
- LineNumber helps reference and review specific commands within the context of their sequence.
- Command provides full visibility into user and script activity within the PowerShell environment, which is invaluable for incident response, security auditing, and troubleshooting.
- Collecting PowerShell command history can reveal attempted or successful privilege escalation, persistence, data exfiltration, or misconfiguration efforts performed via PowerShell.

PowerShell Operational Log

Information Pulled:

- TimeCreated: The timestamp when each PowerShell Operational event log entry was generated (formatted as yyyy-MM-dd HH:mm:ss)
- EventID: The numeric ID of the event (e.g., 4100 for engine lifecycle error, 4104 for scriptblock logging)
- EventIDMeaning: A friendly description of the event type (e.g., "PowerShell Engine Lifecycle Error", "ScriptBlock Logging (Executed Code)", or "Other/Unknown")
- Level: The severity or level of the event (e.g., Information, Warning, Error)
- Message: The event log message text (condensed for readability)
- Section: Static identifier labeling the data as "PowerShellOperationalLog"

Purpose & Usefulness:

- This function retrieves and summarizes recent entries from the Microsoft-Windows-PowerShell/Operational event log.
- TimeCreated and Level support timeline analysis and prioritization of notable PowerShell events.
- EventID and EventIDMeaning help quickly identify significant PowerShell activity, such as executed scripts, errors, or pipeline execution.
- Message provides detailed context on what PowerShell code was run or what issues occurred, aiding in digital forensics, incident response, and troubleshooting.
- Collecting and reviewing PowerShell Operational event logs is critical for detecting malicious PowerShell usage, script execution, and monitoring automation activities for security and compliance on Windows systems.

Prefetch Files

Information Pulled:

- Name: The name of each Prefetch file (.pf)
- Path: The full file system path to the Prefetch file
- SizeKB: The size of the Prefetch file in kilobytes
- Modified: The last modified timestamp of the Prefetch file
- Section: Static identifier labeling the data as "PrefetchFiles"

Purpose & Usefulness:

- This function checks for the existence of the Windows Prefetch folder and collects metadata on all available Prefetch files.
- Prefetch files (.pf) are created by Windows to optimize application startup and track program execution history.
- Name, Path, SizeKB, and Modified provide information for triage and further offline analysis (using forensic tools like PECmd or WinPrefetchView).
- Collecting Prefetch metadata is crucial for digital forensics, incident response, and activity reconstruction, as these files reveal when and how often programs were run, last execution time, and sometimes the path to the executed binary, even if the program or logs have been deleted.

Process Tree WMI

Information Pulled:

- Name: The name of each process
- PID: The process ID (unique identifier) for each process
- ParentPID: The parent process ID (which process launched this process)
- CommandLine: The command line string used to launch the process (if available)
- StartTime: The process creation time (if available and convertible)
- User: The user account under which the process is running (in DOMAIN\User format if available)
- Section: Static identifier labeling the data as "ProcessTreeWMI"

Purpose & Usefulness:

- This function builds a process tree using WMI, providing detailed relationships and context for every process on the system.
- Name, PID, and ParentPID reveal process hierarchies, which is essential for detecting suspicious process spawning, understanding parent-child relationships, and correlating process activity in incident response.
- CommandLine shows the exact execution string, aiding in the identification of malicious scripts, hidden arguments, or command-line-based attacks.

- StartTime and User help establish process activity timelines and attribute actions to specific users or accounts.
- Collecting this information is crucial for digital forensics, threat hunting, security monitoring, and troubleshooting, as it allows investigators to reconstruct how programs and scripts were launched and under what security context.

RDP Logon Events

Information Pulled:

- TimeCreated: The timestamp when each RDP logon event occurred (formatted as yyyy-MM-dd HH:mm:ss)
- EventID: The event ID (should be 1149 for successful Remote Desktop connection attempts)
- Message: The full event log message text (condensed for readability)
- Section: Static identifier labeling the data as "RDP Logon Events"

Purpose & Usefulness:

- This function retrieves and summarizes recent Remote Desktop Protocol (RDP) logon events (Event ID 1149) from the Microsoft-Windows-TerminalServices-RemoteConnectionManager/Operational event log.
- TimeCreated and EventID provide a timeline of RDP connection attempts to the system, helping to identify both legitimate and potentially unauthorized access.
- Message contains details such as user, source IP, and session information, which are crucial for forensic investigations, incident response, and security monitoring.
- Collecting RDP logon event data is essential for tracking remote access activity, detecting brute-force or lateral movement attempts, and maintaining audit trails for compliance or review.

Recent Apps

Information Pulled:

- AppID: The unique identifier for each recently used application, as stored in the registry
- Executable: The path or name of the application's executable file
- LastAccessTime: The last time the application was accessed or used (raw registry value, may need conversion)
- Section: Static identifier labeling the data as "RecentApps"

Purpose & Usefulness:

- This function parses the RecentApps registry key, which tracks applications that have been recently accessed or launched by the user through Windows Search.
- AppID and Executable provide a mapping of recent program usage, allowing investigators to identify what apps were recently run.

- LastAccessTime can help build a timeline of user activity, which is critical for digital forensics, incident response, and behavioral analysis.
- Collecting recent application usage data supports detection of suspicious or rare activity, reconstruction of user actions, and validation of incident timelines.

Recent Docs

Information Pulled:

- SubKey: The subkey under RecentDocs (typically representing file type or grouping)
- ValueName: The name of the registry value storing the recent document reference
- ValueType: The data type of the registry value
- RawValue: The raw value stored in the registry (often a file path or filename)
- SHA256: The SHA-256 hash of the referenced file (if accessible and not a folder), or "N/A"/error indicator
- Section: Static identifier labeling the data as "RecentDocs"

Purpose & Usefulness:

- This function retrieves and parses the Windows RecentDocs registry key, collecting references to files that have been recently accessed by the current user.
- SubKey, ValueName, and RawValue reveal the types of documents accessed and their actual paths or identifiers, helping to reconstruct user activity and document usage.
- ValueType provides insight into how the recent document reference is stored, which can aid in further analysis.
- SHA256 enables integrity checking and identification of specific files accessed, which is valuable for digital forensics, user activity monitoring, and incident response.
- Collecting this data is important for tracking user actions, understanding recent file access patterns, and identifying potentially sensitive or exfiltrated documents.

Recent File Cache

Information Pulled:

- Path: The file path to RecentFileCache.bcf (or "N/A" if not found)
- Note: Description indicating file existence and recommendation to use external tools for parsing (PECmd or AppCompatParser), or stating that the file was not found
- SHA256: The SHA-256 hash of the RecentFileCache.bcf file (or error status)
- Section: Static identifier labeling the data as "RecentFileCache"

Purpose & Usefulness:

- This function checks for the existence of the RecentFileCache.bcf file, which contains binary data about recently accessed or executed programs.
- Path and SHA256 are used for integrity validation and further analysis with specialized tools.

- Note provides context for analysts, clarifying that while the file is present, its binary format requires external utilities for meaningful interpretation.
- Collecting this artifact is important for digital forensics, incident response, and timeline reconstruction, as it can contain evidence of program execution not found in other logs or artifacts.

Recycle Bin Contents

Information Pulled:

- OriginalPath: The original file system path of each item before it was deleted and moved to the Recycle Bin
- DeletedDate: The date and time the item was deleted (as displayed in the Recycle Bin)
- SizeBytes: The size of each deleted item in bytes
- Name: The name of each deleted file or folder
- SHA256: The SHA-256 cryptographic hash of the original file's contents (if still accessible)
- Section: Static identifier labeling the data as "RecycleBinContents"

Purpose & Usefulness:

- This function collects information about all items currently in the user's Recycle Bin.
- OriginalPath and Name help identify what was deleted and from where, supporting recovery operations or investigations into potential data removal.
- DeletedDate allows for establishing a timeline of file deletion activity, which can be useful for forensic investigations or compliance tracking.
- SizeBytes helps assess the storage impact of deleted files and prioritize recovery or investigation efforts.
- SHA256 provides a unique cryptographic identifier for each file (when available), supporting file integrity checks, detection of sensitive data, or correlation with known threat databases.
- Gathering this information is valuable for incident response, digital forensics, and monitoring for suspicious or unauthorized deletion of important files.

Registry Run Keys

Information Pulled:

- RegistryPath: The full registry path of the Run or RunOnce key where the entry is found
- Name: The name of the registry value (the autostart entry label)
- Value: The full command line or value stored in the registry for autostart
- Executable: The path to the executable or script (extracted from the command line where possible)
- SHA256: The SHA-256 hash of the executable file (if found and accessible)
- Suspicious: Boolean flag indicating if the entry is potentially suspicious (e.g., launches from risky folders or runs scripts, batch files, or rundll32)
- Section: Static identifier labeling the data as "RegistryRunKeys"

Purpose & Usefulness:

- This function enumerates all autorun entries from common "Run" and "RunOnce" registry keys for both the current user and the system.
- RegistryPath, Name, and Value reveal what is set to launch automatically on startup or user login, which is critical for finding persistence mechanisms.
- Executable and SHA256 help identify the actual file being run and enable integrity checks or threat intelligence correlation.
- Suspicious highlights risky or non-standard entries (such as scripts or items launched via rundll32 or from temp/user folders) for further investigation.
- Collecting this data is essential for digital forensics, security auditing, and incident response, as malware and attackers often use these registry keys to maintain persistence.

Run MRU

Information Pulled:

- EntryKey: The name of the registry value (usually a single letter or number referencing the order in which the command was run)
- Command: The actual command entered by the user in the Windows Run dialog
- Section: Static identifier labeling the data as "RunMRU"

Purpose & Usefulness:

- This function collects entries from the RunMRU registry key, which records commands and file paths run by the user through the Windows Run dialog.
- EntryKey and Command reveal direct evidence of user-launched programs, scripts, or paths, which may indicate system configuration, application use, or attempted access to sensitive utilities.
- Collecting this data is valuable for digital forensics, user activity reconstruction, and incident response, as it helps track explicit actions and intentions of users or attackers interacting with the system.

Running Processes

Information Pulled:

- Name: The name of each running process
- Id: The process ID (PID) of each running process
- Path: The file system path to the executable for each process (if accessible; otherwise "ACCESS_DENIED" or "N/A")
- StartTime: The timestamp when each process was started (if accessible; otherwise "ACCESS_DENIED")
- Section: Static identifier labeling the data as "RunningProcesses"

Purpose & Usefulness:

- This function collects information about all currently running processes on the system.
- Name and Id uniquely identify each process, which is useful for system monitoring, troubleshooting, and correlation with security events.
- Path shows the location of the executable file for each process, helping identify legitimate versus suspicious or potentially malicious processes.
- StartTime indicates when the process began, which aids in timeline reconstruction for incident response and in identifying long-running or recently started processes.
- Collecting this data supports system health checks, digital forensics, security monitoring, and can help detect unauthorized or anomalous processes running on the system.

SAM Hive Check

Information Pulled:

- Hive: The name of the registry hive ("SAM")
- Path: The file path to the SAM hive on disk
- Exists: Indicates whether the file was found ("Yes" or "No")
- Note: Description indicating that the SAM hive is in binary format and requires external tools (such as `reg save` or `secretsdump.py`) for analysis
- Section: Static identifier labeling the data as "SAM Hive"

Purpose & Usefulness:

- This function checks for the presence of the Security Account Manager (SAM) registry hive, which stores local user account information, password hashes, and other sensitive security data.
- Path and Exists help analysts confirm whether this critical security artifact is present and available for forensic extraction.
- Note guides further analysis by referencing common tools used to process or extract data from the binary hive.
- Collecting SAM hive information is essential for digital forensics, credential auditing, incident response, and post-compromise investigations, as it is often targeted by attackers and can provide evidence of user accounts, group memberships, and potential password compromise.

Scheduled Tasks

Information Pulled:

- TaskName: The name of each scheduled task
- TaskPath: The path of each scheduled task in Task Scheduler
- State: The current status of the task (e.g., Ready, Running, Disabled)

- LastRunTime: The last time the task was run (formatted as yyyy-MM-dd HH:mm:ss or "Never")
- NextRunTime: The next scheduled run time for the task (if available)
- LastResult: The last result or exit code of the task's last run
- Hidden: Boolean indicating if the task is hidden from standard Task Scheduler views
- Principal: The user account under which the task runs
- Action: The command(s) or executable(s) the task runs
- Trigger: The trigger(s) for the task (e.g., schedule, at logon)
- SHA256: The SHA-256 hash of the executable/action for the task (if resolvable)
- SuspiciousPath: Boolean flag indicating if the task's action points to suspicious locations (e.g., AppData, Temp, Downloads, or user profile directories)
- SuspiciousTrigger: Boolean flag indicating if the task uses suspicious or high-risk triggers (e.g., running late at night, at logon, or with repetition)
- Section: Static identifier labeling the data as "ScheduledTasks"

Purpose & Usefulness:

- This function enumerates all scheduled tasks on the system and gathers detailed metadata about each one.
- TaskName, TaskPath, State, Action, and Trigger allow for identification and review of what each scheduled task does and when it executes.
- Principal identifies the user context, which is crucial for assessing privilege levels and risk.
- LastRunTime, NextRunTime, and LastResult provide execution history for auditing, troubleshooting, and incident investigation.
- Hidden and SuspiciousPath/SuspiciousTrigger flags help quickly spot stealthy or potentially malicious tasks that could be used for persistence, privilege escalation, or lateral movement.
- SHA256 supports file integrity checks and threat hunting by providing a unique hash for each executable or script run by a task.
- Collecting this information is essential for security monitoring, compliance auditing, digital forensics, and detecting unauthorized or malicious persistence mechanisms.

Section

- # Requires: Microsoft Word installed on the system
- $inputFolder = "C:\Users\whisk\OneDrive\Desktop\Sections"
- $outputDocx = "C:\Users\whisk\OneDrive\Desktop\WhiteHatWes\Blue Trace Documentation\Sections.docx"
- # Create Word COM object
- $word = New-Object -ComObject Word.Application
- $word.Visible = $false
- $doc = $word.Documents.Add()
- # Set default font
- $word.Selection.Font.Name = "Times New Roman"

- $word.Selection.Font.Size = 12
- # Process each file in the folder
- Get-ChildItem -Path $inputFolder -File | Sort-Object Name | ForEach-Object {
- $file = $_
- $header = [System.IO.Path]::GetFileNameWithoutExtension($file.Name)
- # Add Header (bold, centered)
- $para = $doc.Paragraphs.Add()
- $para.Range.Text = $header
- $para.Range.ParagraphFormat.Alignment = 1 # Centered
- $para.Range.Font.Bold = $true
- $para.Range.Font.Name = "Times New Roman"
- $para.Range.Font.Size = 12
- $para.Range.InsertParagraphAfter()
- # Add bullets for each line (default bullet, left-aligned)
- $lines = Get-Content $file.FullName | Where-Object { $_.Trim() -ne "" }
- foreach ($line in $lines) {
- $para = $doc.Paragraphs.Add()
- $range = $para.Range
- $range.Text = $line.Trim()
- $range.Font.Name = "Times New Roman"
- $range.Font.Size = 12
- $range.ParagraphFormat.Alignment = 0 # Left
- $range.ListFormat.ApplyBulletDefault()
- $range.InsertParagraphAfter()
- }
- }
- # Save and close
- $doc.SaveAs([ref]$outputDocx)
- $doc.Close()
- $word.Quit()
- Write-Output "Document created at: $outputDocx"

Security Event Log

Information Pulled:

- TimeCreated: The timestamp when each Security event log entry was generated (formatted as yyyy-MM-dd HH:mm:ss)
- EventID: The numeric ID of the event (e.g., 4624 for logon success, 4688 for process creation)
- EventIDMeaning: A friendly description of the event type (e.g., "Account Logon Success", "New Process Created", or "Other / Unknown")

- Level: The severity or level of the event (e.g., Information, Warning, Error)
- Message: The event log message text (condensed for readability)
- Section: Static identifier labeling the data as "SecurityEventLog"

Purpose & Usefulness:

- This function retrieves and parses recent entries from the Windows Security Event Log, providing both raw data and summarized descriptions.
- TimeCreated and Level allow for quick filtering and timeline analysis of security-relevant activity.
- EventID and EventIDMeaning help analysts understand the type and context of each event, supporting incident triage and security monitoring.
- Message provides the detailed event context needed for forensic investigations, threat hunting, and audit compliance.
- Collecting and interpreting Security event logs is crucial for detecting suspicious authentication, privilege escalation, process creation, and other potentially malicious activity.

Security Hive Check

Information Pulled:

- Hive: The name of the registry hive ("SECURITY")
- Path: The file path to the SECURITY hive on disk
- Exists: Indicates whether the file was found ("Yes" or "No")
- Note: Description explaining that this is a binary hive file and should be parsed offline with specialized tools for detailed analysis
- Section: Static identifier labeling the data as "SECURITY Hive"

Purpose & Usefulness:

- This function checks for the presence of the SECURITY registry hive, which contains sensitive local security policy information, user rights assignments, and security identifiers.
- Path and Exists allow analysts to confirm the presence and availability of this critical system hive for forensic imaging or offline analysis.
- Note provides guidance for proper handling and analysis, since the hive is in binary format and not human-readable.
- Collecting the SECURITY hive is important for digital forensics, security auditing, and incident response, as it contains historical and current security policy configurations, audit policy, and account rights information that can be used to track changes, detect unauthorized privilege escalation, or support investigations of malicious activity.

Service Information

Information Pulled:

- Name: The service's short (system) name

- DisplayName: The display name of the service
- State: The current state of the service (e.g., Running, Stopped)
- StartMode: The startup type for the service (e.g., Auto, Manual, Disabled)
- PathName: The full command line or path used to launch the service
- Executable: The extracted executable file path for the service (if available)
- SHA256: The SHA-256 hash of the service's executable (if available)
- IsSigned: Boolean indicating if the service's executable is digitally signed and valid
- SignatureStatus: The status of the executable's digital signature (e.g., Valid, NotSigned, CheckFailed)
- Publisher: The publisher or subject of the signing certificate (if available)
- Suspicious: Boolean indicating if the service is suspicious (e.g., unsigned, script-based, or running from risky folders)
- Description: The service description, if available
- Section: Static identifier labeling the data as "ServiceInformation"

Purpose & Usefulness:

- This function enumerates all Windows services and collects detailed metadata about each one.
- Name, DisplayName, State, and StartMode provide an overview of all services and their operational status, which is essential for system monitoring and auditing.
- PathName and Executable identify what is actually run when the service starts, supporting detection of potentially unwanted or malicious services.
- SHA256 and digital signature fields (IsSigned, SignatureStatus, Publisher) enable integrity checks, threat hunting, and identification of trusted vs. untrusted code.
- Suspicious flags help focus attention on services that are unsigned, use scripts, or reside in non-standard or high-risk locations.
- Description supplies additional context to aid in the identification and validation of each service's purpose.
- Collecting this information is crucial for digital forensics, security auditing, incident response, and system hardening, as malicious actors often use services for persistence or privilege escalation.

Set Up Event Log

Information Pulled:

- TimeCreated: The timestamp when each Setup event log entry was generated (formatted as yyyy-MM-dd HH:mm:ss)
- EventID: The numeric ID of the event (e.g., 1 for setup started, 30103 for feature update completed)
- EventIDMeaning: A friendly description of the event type (e.g., "Setup started", "Feature Update Completed", or "Unknown")

- KB: The KB (Knowledge Base) identifier extracted from the event message, if available
- Level: The severity or level of the event (e.g., Information, Warning, Error)
- Message: The event log message text (condensed for readability)
- Section: Static identifier labeling the data as "SetupEventLog"

Purpose & Usefulness:

- This function retrieves and summarizes recent entries from the Windows Setup Event Log.
- TimeCreated and Level help build a timeline of setup, update, and restoration events, supporting troubleshooting and historical analysis.
- EventID and EventIDMeaning provide quick insight into the type and status of each setup or update event.
- KB identifies specific Microsoft updates, patches, or rollups, supporting patch management, vulnerability tracking, and compliance verification.
- Message supplies detailed context for installation and update events, aiding in root cause analysis for setup or update failures.
- Collecting and interpreting Setup event logs is crucial for identifying upgrade history, troubleshooting failed installations, verifying system updates, and supporting digital forensics on Windows systems.

Shell Bags

Information Pulled:

- Path: The registry path checked for ShellBag information (e.g., BagMRU, Bags under HKCU)
- Exists: Boolean indicating whether the registry key exists on the system
- LastWrite: The last modified timestamp of the registry key (formatted as yyyy-MM-dd HH:mm:ss or "N/A")
- Section: Static identifier labeling the data as "ShellBags"

Purpose & Usefulness:

- This function checks for the existence and last modification time of key ShellBag registry entries, which Windows uses to store folder view settings and track folder access/navigation history.
- Path and Exists indicate which ShellBag structures are present for the user, supporting validation and discovery during forensic analysis.
- LastWrite helps establish a timeline for when a user last interacted with or modified folder views, which can correlate with user activity or the presence of evidence in certain folders.
- Collecting ShellBag information is valuable for digital forensics and incident response, as ShellBags can reveal folder traversal and access even if files or shortcuts have been deleted or cleared.

Software Hive Check

Information Pulled:

- Hive: The name of the registry hive ("SOFTWARE")
- Path: The full file system path to the SOFTWARE hive
- Exists: Indicates whether the SOFTWARE hive file is present ("Yes" or "No")
- Note: Guidance on usage (recommending offline registry parsing tools if found)
- Section: Static identifier labeling the data as "SoftwareHive"

Purpose & Usefulness:

- This function checks for the presence of the SOFTWARE registry hive file, which stores information about installed applications, OS components, and configuration settings.
- Path and Exists show whether the hive can be collected for forensic or offline analysis.
- The Note field provides guidance, as SOFTWARE is a binary file requiring special registry tools for detailed examination.
- The SOFTWARE hive is a core forensic artifact for investigations and incident response, enabling analysts to recover details about software installation, updates, application usage, and malware persistence mechanisms.

SRUM Data

Information Pulled:

- SRUDB_Path: The file path to SRUDB.dat (the System Resource Usage Monitor database), or "Not Found" if the file does not exist
- Note: A short message indicating whether the database exists and, if so, that external tools like SRUM-Dump or SQLite tools are needed for analysis
- Section: Static identifier labeling the data as "SRUM"

Purpose & Usefulness:

- This function checks for the existence of the SRUM database, which contains detailed system resource usage information, including app/network usage, energy consumption, and some user activity history.
- SRUDB_Path and Note inform an analyst of the presence of this key forensic artifact and provide next steps for deeper analysis.
- The SRUM database is important for digital forensics, incident response, and activity reconstruction, as it can reveal patterns of application/network usage, logon activity, and system events even when traditional event logs have been cleared.

Start Up Folder Items

Information Pulled:

- User: The user account associated with each Startup folder item

- ItemName: The name of the file (shortcut, script, or executable) found in a Startup folder
- FullPath: The complete file system path to the Startup item
- LastModified: The last modified timestamp of the Startup item
- SHA256: The SHA-256 cryptographic hash of the Startup item (for integrity and threat hunting)
- Suspicious: Boolean flag indicating if the item is potentially suspicious (e.g., located in risky folders or is a script or shortcut file)
- Section: Static identifier labeling the data as "StartupFolderItems"

Purpose & Usefulness:

- This function enumerates all files in the Startup folders for the current user, all users, and each profile on the system, collecting detailed metadata for each item.
- User, ItemName, and FullPath allow clear attribution and location of auto-starting programs, which is essential for investigating persistence mechanisms.
- LastModified shows when each item was last changed, supporting timeline analysis and detecting recent additions or modifications.
- SHA256 uniquely identifies each file for integrity checks and correlation with threat intelligence or known malware.
- Suspicious flags quickly highlight items that might represent malicious persistence (such as scripts, links, or items in risky locations).
- Collecting this information is vital for digital forensics, incident response, security auditing, and detection of unauthorized or malicious programs set to run at login or system start.

Symbolic Links and Junctions

Information Pulled:

- FullName: The full file system path to each symbolic link or junction found on the C: drive
- Attributes: The file system attributes of each reparse point (as a string)
- LinkType: The type of reparse point detected ("SymbolicLink" for files or "Junction" for folders)
- Section: Static identifier labeling the data as "SymbolicLinksAndJunctions"

Purpose & Usefulness:

- This function enumerates all symbolic links and junctions (reparse points) on the C: drive.
- FullName allows identification and location of each link or junction, which is critical for understanding file system structure and navigation.
- Attributes provide additional context about the item, including its reparse point status and other file properties.
- LinkType distinguishes between file-based symbolic links and folder-based junction points, which may behave differently.

- Collecting information on symbolic links and junctions is valuable for security audits, digital forensics, and troubleshooting, as these features can be abused for persistence, evasion, or redirection by attackers, or can complicate file system analysis and data recovery.

System Event Log

Information Pulled:

- TimeCreated: The timestamp when each System event log entry was generated (formatted as yyyy-MM-dd HH:mm:ss)
- EventID: The numeric ID of the event (e.g., 6008 for unexpected shutdown, 7045 for service installation)
- EventIDMeaning: A friendly description of the event type (e.g., "Unexpected Shutdown", "Service Installed", or "Unknown")
- Level: The severity or level of the event (e.g., Information, Warning, Error)
- Message: The event log message text (condensed for readability)
- Section: Static identifier labeling the data as "SystemEventLog"

Purpose & Usefulness:

- This function retrieves and summarizes recent entries from the Windows System Event Log.
- TimeCreated and Level allow for quick filtering and timeline analysis of system-related activity and incidents.
- EventID and EventIDMeaning help analysts understand the type and relevance of each event, aiding in event triage and system health monitoring.
- Message provides detailed context for system events, supporting troubleshooting, digital forensics, and compliance auditing.
- Collecting and interpreting System event logs is critical for detecting service issues, boot problems, hardware failures, and other significant operational or security events on a Windows system.

System Hive Check

Information Pulled:

- Hive: The name of the registry hive ("SYSTEM")
- Path: The full file system path to the SYSTEM hive
- Exists: Indicates whether the SYSTEM hive file is present ("Yes" or "No")
- Note: Guidance on usage (recommending offline registry parsing tools if found)
- Section: Static identifier labeling the data as "SystemHive"

Purpose & Usefulness:

- This function checks for the presence of the SYSTEM registry hive file, which contains key configuration data for the operating system, hardware, and system services.

- Path and Exists indicate whether the hive can be collected for forensic imaging or offline analysis.
- The Note field provides next steps, since SYSTEM is a binary file requiring specialized parsing tools.
- SYSTEM hive data is crucial for digital forensics, malware analysis, and incident response, as it reveals boot configuration, device drivers, service settings, and other OS-level details that can be used to investigate persistence, attacks, or system changes.

System Information

Information Pulled:

- Hostname: The computerâ€™s network name (`$env:COMPUTERNAME`)
- UserDomain: The domain name of the logged-in user (`$env:USERDOMAIN`)
- Username: The current username (`$env:USERNAME`)
- SID: The Security Identifier for the current user
- IsAdmin: Whether the script is running with administrator privileges
- SystemTimeLocal: The local system date and time
- SystemTimeUTC: The system date and time in UTC
- TimeZone: The current system time zone ID
- BootTime: The timestamp of the last system boot

Purpose & Usefulness:

- This function gathers fundamental system and user session details necessary for incident response, auditing, or troubleshooting.
- Hostname, UserDomain, and Username uniquely identify the machine and current user contextâ€”useful for correlating logs or tracking user activity.
- SID is critical for uniquely identifying users, especially in environments with identical usernames across domains.
- IsAdmin reveals whether the script has elevated privileges, affecting the ability to perform certain actions or access protected data.
- SystemTimeLocal and SystemTimeUTC provide precise timing for events, supporting log correlation and forensic timelines.
- TimeZone helps interpret timestamps in the context of the machineâ€™s location.
- BootTime establishes how long the system has been running, which can help detect recent reboots (possibly related to incidents or updates).

Temp Folder Contents

Information Pulled:

- FullName: The complete file system path to each file in the user's TEMP directory (recursively)
- Length: The size of each temporary file in bytes

- LastWriteTime: The last modified timestamp of each file (formatted as yyyy-MM-dd HH:mm:ss)
- SHA256: The SHA-256 cryptographic hash of each file's contents
- Section: Static identifier labeling the data as "TempFolderContents"

Purpose & Usefulness:

- This function collects metadata for all files found in the user's TEMP directory and its subfolders.
- FullName identifies where each temporary file is stored, which is helpful in locating suspicious or potentially sensitive files that could be left behind by applications or malware.
- Length gives the size of each file, which helps in prioritizing analysis or cleanup.
- LastWriteTime shows when the file was last modified, which can help correlate temporary file creation or changes with system events or possible compromise.
- SHA256 provides a unique identifier for each file's contents, useful for integrity checking and for matching files against known malicious or sensitive file hashes.
- Collecting this data supports digital forensics, incident response, and system maintenance by highlighting transient files that may hold evidence of user or attacker activity.

Typed Paths

Information Pulled:

- EntryName: The name of the registry value (typically in the format url1, url2, etc.)
- PathValue: The actual path or URL typed by the user into Windows Explorer's address bar
- Section: Static identifier labeling the data as "TypedPaths"

Purpose & Usefulness:

- This function retrieves data from the TypedPaths registry key, which tracks file system paths and URLs entered by the user into the Windows Explorer address bar.
- EntryName and PathValue reveal direct user navigation, showing what locations have been accessed or browsed, regardless of whether the user used links or shortcuts.
- Collecting this data is useful for digital forensics, user activity monitoring, and incident response, as it provides insight into user intent and access historyâ€”potentially exposing attempted access to sensitive, remote, or otherwise noteworthy folders or shares.

UAC Settings

Information Pulled:

- ConsentPromptBehaviorAdmin: How User Account Control (UAC) prompts administrators (various prompt levels or silent elevation)
- ConsentPromptBehaviorUser: How UAC prompts standard users when elevation is requested
- EnableLUA: Whether UAC is enabled or disabled on the system

- PromptOnSecureDesktop: Whether UAC prompts are shown on a secure desktop (protects against spoofing)
- EnableVirtualization: Whether UAC file and registry virtualization is enabled (helps with legacy applications)
- ValidateAdminCodeSignatures: Whether admin code signatures are validated before running
- FilterAdministratorToken: Whether administrator accounts use a split or full security token
- Section: Static identifier labeling the data as "UACSettings"

Purpose & Usefulness:

- This function collects the key UAC (User Account Control) configuration settings from the system registry.
- These settings determine how Windows manages privilege elevation and how users are prompted for administrative actions.
- Reviewing these values is crucial for understanding the system's security posture, since relaxed UAC settings can increase the risk of privilege escalation and malware execution.
- Secure desktop and code signature validation help mitigate spoofing and ensure only trusted code is run with elevated permissions.
- Virtualization settings impact compatibility and containment of legacy applications.
- This data supports security audits, incident response, and compliance by showing how strictly Windows enforces elevation and admin actions.

USB History

Information Pulled:

- Device: The registry subkey name for each detected USB storage device
- FriendlyName: The user-friendly name of the device (if available)
- Service: The driver/service used by the device (if available)
- ContainerID: The unique container ID assigned to the device (if available)
- PSPath: The full registry path to the device entry
- Section: Static identifier labeling the data as "USBHistory"

Purpose & Usefulness:

- This function enumerates the USBSTOR registry key, collecting information about all USB storage devices (such as flash drives and external hard drives) ever connected to the system.
- Device, FriendlyName, Service, and ContainerID provide device-specific details useful for tracking usage, identifying devices, and correlating with user or incident activity.
- PSPath offers precise registry location for reference or deeper manual analysis.
- Collecting USB history is essential in digital forensics, insider threat investigations, and data exfiltration cases, as it helps identify removable devices that may have been used to copy or transfer sensitive data.

User Assist

Information Pulled:

- SubKey: The registry subkey under UserAssist (typically a GUID, often associated with application usage tracking)
- ValueName: The name of the registry value (often an encoded or application-specific identifier)
- ValueType: The data type of the value (e.g., String, Int, DateTime)
- RawValue: The raw data stored for the UserAssist entry (may require decoding for full analysis)
- Section: Static identifier labeling the data as "UserAssist"

Purpose & Usefulness:

- This function retrieves and parses data from the UserAssist registry keys, which Windows uses to track programs and files recently executed or accessed by the user.
- SubKey, ValueName, and RawValue reveal detailed information about user activity, application usage, and file execution on the system, supporting timeline and behavior analysis.
- ValueType informs analysts of how the data is stored, which may be helpful for decoding or deeper investigation.
- Collecting this data is vital for digital forensics, user activity monitoring, and incident response, as it provides insights into what applications and files have been used, regardless of whether shortcuts remain or standard logs are cleared.

User Class Check

Information Pulled:

- File: The name of the file checked ("UsrClass.dat")
- Path: The full path to the UsrClass.dat file for the current user
- Exists: Indicates whether the file was found ("Yes" or "No")
- Note: Description noting that this file contains ShellBag and Jump List data, and should be parsed with Registry Explorer or a ShellBag parser
- Section: Static identifier labeling the data as "USRCLASS.DAT"

Purpose & Usefulness:

- This function checks for the presence of the UsrClass.dat registry hive, which stores user-specific Windows Explorer settings, ShellBag information (folder view history), and Jump List data (recent app/file activity).
- Path and Exists help forensic analysts confirm whether this key user registry hive is available for further analysis.
- Note offers guidance for deeper investigation using common forensic tools.

- UsrClass.dat is a critical artifact for digital forensics, incident response, and user activity reconstruction, as it can reveal historical folder access, desktop layout, and recent file/app usage even when other evidence has been cleared.

Volume Shadow Copies

Information Pulled:

- ID: The unique identifier for each volume shadow copy (snapshot)
- VolumeName: The original volume name or drive letter associated with the shadow copy
- InstallDate: The creation timestamp of each shadow copy (InstallDate or Creation Time)
- Section: Static identifier labeling the data as "VolumeShadowCopies"

Purpose & Usefulness:

- This function enumerates all Volume Shadow Copies (snapshots) present on the system, using both WMI and a command-line fallback for completeness.
- ID provides a unique handle for each shadow copy, useful for referencing or managing specific snapshots.
- VolumeName shows which disk volume or partition the shadow copy belongs to, supporting targeted restoration or analysis.
- InstallDate establishes when each shadow copy was created, which helps in constructing backup, recovery, or compromise timelines.
- Collecting this information is important for data recovery planning, forensic investigations (since deleted or modified data may be recoverable from snapshots), and for identifying unauthorized or suspicious shadow copy creation that may indicate ransomware or attacker activity.

WER Crash Dumps

Information Pulled:

- Section: "WERCrashDumps"
- FullName: Full path of the crash dump file found in C:\ProgramData\Microsoft\Windows\WER\ReportQueue
- LastWriteTime: Last modified date/time of the dump file
- Length: Size of the dump file in bytes

Purpose & Usefulness:

- This function collects metadata about crash dump files generated by Windows Error Reporting (WER).
- Helps identify application or system crashes that have occurred recently.
- Provides forensic and troubleshooting value: paths can be provided to analysts or developers for deeper crash/memory analysis.
- Useful for system health monitoring, incident response, and root cause analysis.

Whomami Groups

Information Pulled:

- GroupName: The name of each group the current user is a member of
- SID: The Security Identifier associated with each group
- Attributes: Group membership attributes (e.g., Enabled, Mandatory, Default Owner) or "None" if not present
- Section: Static identifier labeling the data as "WhoamiGroups"

Purpose & Usefulness:

- This function runs `whoami /groups` and parses its output to enumerate all security groups the current user is a member of, including group SIDs and any relevant attributes.
- GroupName and SID provide a comprehensive mapping of the user's group memberships, which is vital for privilege and access reviews.
- Attributes indicate special group properties that may affect security context or privileges (such as Enabled or Default Owner).
- Collecting this data is important for digital forensics, incident response, and privilege escalation investigations, as it helps determine the effective permissions and roles assigned to the current user context.

Windows Defender Status

Information Pulled:

- Section: "WindowsDefenderStatus"
- AM Service Enabled: Indicates if Windows Defender Antispyware service is running
- Real-Time Protection Enabled: Indicates if real-time protection is active
- Behavior Monitoring Enabled: Indicates if behavior monitoring is enabled
- Signature Version: Shows the version of the Defender antivirus signatures
- Last Signature Update Time: Timestamp of the last signature update

Purpose & Usefulness:

- This function collects core status details about Windows Defender, including protection status and signature currency.
- Provides confirmation that Defender is running and real-time/behavior monitoring protections are active.
- Shows signature version and update time, indicating whether malware definitions are current.
- Useful for verifying endpoint security, detecting misconfiguration or outdated protections, and for compliance checks.

Windows PowerShell Log

Information Pulled:

- TimeCreated: The timestamp when each Windows PowerShell event log entry was generated (formatted as yyyy-MM-dd HH:mm:ss)
- EventID: The numeric ID of the event (e.g., 400 for engine lifecycle, 403 for command started)
- EventIDMeaning: A friendly description of the event type (e.g., "Engine Lifecycle State Changed", "Command Started", or "Other/Unknown")
- Level: The severity or level of the event (e.g., Information, Warning, Error)
- Message: The event log message text (condensed for readability)
- Section: Static identifier labeling the data as "WindowsPowerShellLog"

Purpose & Usefulness:

- This function retrieves and summarizes recent entries from the Windows PowerShell event log.
- TimeCreated and Level help build a timeline and filter events based on severity for efficient analysis.
- EventID and EventIDMeaning provide quick context about the nature of each PowerShell event, supporting triage and incident response.
- Message offers detailed information about PowerShell execution, module loads, and errors, which is essential for digital forensics, monitoring for unauthorized PowerShell activity, and troubleshooting script or automation issues.
- Collecting and reviewing Windows PowerShell event logs is vital for detecting potential abuse of PowerShell by attackers, understanding automation activity, and maintaining security on Windows systems.

Windows Version Information

Information Pulled:

- Caption: The friendly name or edition of the installed Windows operating system
- Version: The OS version number (major.minor.build)
- BuildNumber: The specific Windows build number
- Architecture: The operating system architecture (e.g., 64-bit, 32-bit)
- InstallDate: The date and time the operating system was installed (formatted for readability)
- LastBootUpTime: The date and time of the last system startup (formatted for readability)
- Section: Static identifier labeling the data as "WindowsVersionInfo"

Purpose & Usefulness:

- This function collects essential information about the Windows operating system version and its installation history.

- Caption, Version, and BuildNumber help identify exactly which version and build of Windows is running, which is important for assessing compatibility, patch levels, and vulnerability exposure.
- Architecture clarifies if the OS is 32-bit or 64-bit, which affects software compatibility and some security controls.
- InstallDate can provide insights into the systemâ€™s lifecycle and help in investigations involving OS upgrades or reinstalls.
- LastBootUpTime is useful for uptime calculations and can help correlate system events with reboots, updates, or possible incident timelines.
- Collecting this information supports troubleshooting, asset management, incident response, and security compliance efforts.

WMI Activity Logs

Information Pulled:

- TimeCreated: The timestamp when each WMI event was generated (formatted as yyyy-MM-dd HH:mm:ss)
- EventID: The event ID associated with each WMI activity event
- EventType: A descriptive label for the type of WMI activity (e.g., Query Execution Started, Query Execution Completed, Error, etc.)
- ProcessID: The process ID that initiated the WMI operation (if available)
- User: The user account that performed the WMI operation (if available)
- Operation: The specific operation being performed (e.g., WMI query, method call)
- Namespace: The WMI namespace in which the operation occurred
- Query: The actual WMI query or operation details (if available)
- Section: Static identifier labeling the data as "WmiActivityLogs"

Purpose & Usefulness:

- This function collects recent events from the Windows WMI Activity Operational log, which records detailed activity for all WMI (Windows Management Instrumentation) operations.
- TimeCreated, EventID, and EventType provide essential timeline and classification data for each event.
- ProcessID and User identify who initiated the activity, supporting attribution and investigation of potentially malicious or unauthorized activity.
- Operation, Namespace, and Query provide context on exactly what was being accessed or executed, helping in threat hunting, security auditing, and troubleshooting.
- Collecting and analyzing WMI activity is critical for incident response and digital forensics, as attackers often abuse WMI for persistence, lateral movement, and execution of malicious code.

WMI Event Consumers

Information Pulled:

- Name: The name of the WMI event consumer
- ConsumerType: The class/type of the WMI event consumer (e.g., CommandLineEventConsumer, ActiveScriptEventConsumer)
- CommandLine: The command line to be executed by the consumer (if available)
- Executable: The extracted path to the executable file (if available)
- Suspicious: Boolean flag indicating if the command line is potentially suspicious (e.g., involves scripts, PowerShell, or runs from risky folders)
- Section: Static identifier labeling the data as "WMIEventConsumers"

Purpose & Usefulness:

- This function enumerates all WMI event consumers in the system's `root\subscription` namespace and collects metadata about each one.
- Name and ConsumerType help identify the type and purpose of each event consumer, which are used by the WMI system to execute actions in response to certain triggers or events.
- CommandLine and Executable indicate what action or file will be executed when the event consumer is triggered, which is vital for detecting persistence mechanisms or potential abuse by attackers.
- Suspicious flags highlight consumers that run from non-standard or high-risk locations, or that use interpreters or scripts commonly abused by malware.
- Collecting this data is essential for security monitoring, digital forensics, and incident response, as malicious actors often use WMI event consumers for stealthy, persistent code execution.