APPLIED INCIDENT RESPONSE

Applied Incident Response Scripts Usage Instructions

Usage Instructions

Version 20240415

Executive Summary

These scripts solve the problem of using free and open-source tools for incident response in a scalable, repeatable manner that protects the privileged credentials needed to collect evidence. This is accomplished by using a temporary server to provide trusted copies of collection executables and receive collected evidence. PowerShell remoting is used to run the tools with the necessary permissions, while not exposing the credentials on compromised hosts. Parsing of the collected artifacts is then automated to improve consistency and efficiency of analysis.

Technical Details

The free and open-source software community offers many amazing ways to simplify incident response. Examples include:

> KAPE (Kroll Artifact Parser and Extractor) is a powerful and efficient tool designed to quickly extract forensic artifacts from Windows computers. Its configuration allows for rapid deployment and customizable outputs, making it a favorite among forensic analysts for initial triage and subsequent deep-dive investigations.
>
> Eric Zimmerman's tools provide a comprehensive suite for detailed forensic analysis. These tools enable investigators to uncover detailed insights into file system activities, registry changes, and much more. The suite includes utilities like AmcacheParser, AppCompatCacheParser, MFTECmd, PECmd, and EvtxECmd, each targeting specific artifacts for extraction and analysis.
>
> Hayabusa excels in the rapid analysis of Windows event logs, aiding responders in quickly understanding the scope and impact of an incident. Using its own rules and open-source Sigma rules to identify events of interest, Hayabusa can expedite log analysis and facilitate triage of possibly compromised systems.
>
> Elastic Stack provides a powerful platform for searching, analyzing, and visualizing log data. Its ability to scale and process vast amounts of data makes it an excellent choice for incident detection, response, and proactive threat hunting.

The tools listed above are free for most use cases but should be downloaded from their respective websites to comply with their licensing rules.

The Applied Incident Response PowerShell scripts assist with the deployment and operation of these tools, enhancing their efficiency and integration into incident response workflows. These scripts enable responders to rapidly deploy these tools across affected systems, standardizing the collection and analysis of forensic data, and ensuring consistent, thorough investigative processes.

The Applied Incident Response scripts rely upon KAPE to collect evidence from target systems. KAPE is deployed through PowerShell remoting, which is assumed to be available in the target environment. PowerShell remoting allows for KAPE to be run with the required administrative privileges, without exposing the associated credential on the potentially compromised endpoints.

Prepare-AutoKapeServer.ps1 helps set up a collection server, designed to receive forensic evidence from networked systems. It configures the necessary network and file sharing settings, turning the server into a centralized point for collecting digital artifacts and allowing for a streamlined evidence gathering process across multiple systems. The evidence files are stored in virtual hard disk (VHDX) containers, each named for the target system from which they originated.

Run-AutoKape.ps1 enables incident responders to collect forensic artifacts from large numbers of systems. Operating from a secure administrative workstation, this script initiates PowerShell remoting connections to target systems, executes KAPE to capture relevant artifacts, and facilitates the transmission of this evidence back to the KAPE server.

Settings.psd1 acts as the configuration file for the collection server and the connections from target computers back to it.

Restore-AutoKapeServer.ps1 is designed for post-collection cleanup, focusing on reverting the server back to its pre-collection posture. By disabling the file sharing and network configurations enabled by Prepare-AutoKapeServer.ps1, it allows the temporary file sharing adjustments made to facilitate evidence collection to be reverted when no longer needed.

Mount-TriageVhdx.ps1 automates the process of mounting VHDX files to specifically named folders, reflecting the origin's system name. This script streamlines the setup for forensic analysis by organizing evidence in an easily navigable structure, enhancing efficiency and accuracy in reviewing collected data.

Dismount-TriageVhdx.ps1 is responsible for orderly cleanup after analysis, systematically dismounting VHDX volumes and removing the temporary directories used during the investigation.

Once evidence is collected, additional scripts can help streamline the analysis process.

Run-ZimmermanTools.ps1 streamlines the forensic analysis by applying many of Eric Zimmerman's artifact processing tools to the data collected by KAPE. This script automates the parsing process, allowing for a faster review and extraction of information from forensic artifacts. The EvtxECmd utility is used to parse event logs, with options to reduce the data to only specific event IDs of common interest and combine logs from all systems to streamline analysis. AmcacheParser, AppCompatCacheParser, MFTECmd, and PECmd are also run by this script.

Run-ZTSrumECmd.ps1 is designed to use Zimmerman's SrumECmd utility on the evidence files. SrumECmd analyzes System Resource Usage Monitor (SRUM) data, offering insights into application and network usage. Due to the large number of output files created by this tool, it is placed in a separate script.

Run-Hayabusa.ps1 parses event logs collected from systems to identify matches against Sigma rules and Hayabusa's own detection rules. By examining folders within the mounted VHDX artifacts, the script enhances the analysis of logs gathered by KAPE, focusing on streamlining the identification of potential security incidents.

Run-Winlogbeat.ps1 serves to ingest event log data into an Elastic Stack instance for more in-depth analysis and visualization. To utilize this script, specifics of your Elastic Stack setup must be configured within the **winlogbeat.yml** file. This optional step enriches the investigative process by leveraging Elastic Stack's powerful data processing and analysis capabilities. You can find instructions for using Docker containers to run Elastic Stack here. Winlogbeat itself can be downloaded from here.

A generalized workflow with these scripts might include:

- Connect a collection laptop, virtual machine, or other system to the target network, ensuring that IP connectivity and firewall rules exist to allow for data to be sent from target systems to SMB (port 445) on the collection server.
- Use **Prepare-AutoKapeServer.ps1** and the associated **Settings.psd1** file to configure the connected system to receive data.
- Logon to a secure administrative workstation (SAW), or similar system as available within the environment, with a privileged credential that has admin rights on the target systems. Ensure that the workstation chosen for this task is secure, as the act of interactively logging on with a privileged credential can expose that credential on the local system.
- On the SAW, use the Run-AutoKape.ps1 script, in conjunction with the associated Settings.psd1 file, to remotely run KAPE on the target systems and send the results to the KAPE Server. Note that KAPE is run remotely from the server without the need to copy it to each target system, thereby reducing the forensic impact on the target systems. Furthermore, the use of PowerShell remoting protects the admin credentials needed to run KAPE from exposure on the potentially compromised target systems.
- Once the data is collected and stored on the KAPE server, the associated VHDX files can be copied to any analysis platform desired. If desired, the KAPE server itself can also be used for analysis, but analysis should be conducted on copies with the original evidence files preserved.
- On the analysis platform, use Mount-TriageVhdx.ps1 to mount copies of each of the VHDX files collected by KAPE to a folder, named for the originating system.
- Run each of the parsing scripts, as desired. Alternatively, you can use KAPE Modules to process the data collected.
- Analyze the results.

The usage details of each script, including the valid parameters and their function, are included as comment-based help within the scripts. Simply run Get-Help followed by the name of the script for detailed usage information. Additional details of the key scripts are also found below.

Prepare-AutoKapeServer.ps1

Synopsis

Configures a computer to serve as a Kape collection server, adjusting network settings, creating a temporary user, sharing necessary folders, and configuring inbound connection rules.

Description

This script prepares a system to act as a KAPE collection server. It automates network configuration, user account creation, folder sharing for KAPE executables and results, and enables port 445 for inbound connections. Configuration settings are managed through a **Settings.psd1** file to maintain consistency across server setup and collection scripts.

Parameters

- **SettingsFile**: Path to the **Settings.psd1** file. If not specified, the script searches for the file in the script's directory.

Example Usage

1. **Default Settings File**

```
.\Prepare-AutoKapeServer.ps1
```

Uses **Settings.psd1** located in the same directory as the script for configuration.

2. **Specified Settings File**

```
.\Prepare-AutoKapeServer.ps1 -SettingsFile C:\Collection\Settings.psd1
```

Configures the server using settings specified in **C:\Collection\Settings.psd1**

Notes

- Script written by Steve Anson.
- Distributed under the GNU General Public License v3.0.
- Ensure the script is run with Administrative privileges.
- Requires a **Settings.psd1** file for configuration details.

Operational Steps

1. **Run as Administrator**: The script checks if it's run with administrative privileges and exits if not.
2. **Settings File**: Imports configuration from **Settings.psd1**. If the file is not found, the script exits.
3. **Transcript Logging**: Starts a transcript log to record actions taken by the script.
4. **Environment Verification**: Confirms the KAPE executable path and results destination path exist and are properly configured.
5. **Network Configuration**: Sets static IP, subnet mask, DNS servers, and gateway based on **Settings.psd1**.
6. **Power Options**: Configures the system to prevent sleeping during the collection process.
7. **User and Share Setup**: Creates a temporary user and shares for KAPE executables and result folders with appropriate NTFS permissions.
8. **Firewall Configuration**: Opens port 445 for inbound connections from target computers specified in the settings file.
9. **Cleanup Reminder**: Recommends using **Restore-AutoKapeServer.ps1** post-collection to revert changes.

Additional Information

- For further details, refer to the GNU General Public License.
- Download KAPE from Kroll's official website.

Run-AutoKape.ps1

Synopsis

Executes KAPE on a list of target computers using PowerShell remoting, saving the output to a network share.

Description

The script connects to target computers defined in a **Settings.psd1** file or specified through parameters when running KAPE with configured arguments. Output is directed to a network share. It prompts for credentials with administrative access to both the target computers and the server hosting the share, which should be set up in advance with **Prepare-AutoKapeServer.ps1**.

Parameters

- **SettingsFile**: Path to the **Settings.psd1** file for configuration. Defaults to looking in the script's directory if not specified.

Example Usage

1. **Default Settings File**

```
.\Run-AutoKape.ps1
```

This looks for **Settings.psd1** in the same directory as the script and runs Kape on the target computers with the arguments specified in the settings file.

2. **Specified Settings File**

```
.\Run-AutoKape.ps1 -SettingsFile C:\Users\user\Desktop\Settings.psd1
```

Executes KAPE using settings from the specified **Settings.psd1** file.

Notes

- Script written by Steve Anson.
- Released under GNU General Public License v3.0.
- Requires administrative credentials for both target computers and the share server.

Operational Steps

- Prompts for administrative credentials for target computers and the server.
- Validates the presence of the **Settings.psd1** file and imports its configurations.
- Confirms target computers are listed and accessible.
- Starts a transcript log for recording actions taken by the script.
- Establishes PowerShell sessions to each target computer.
- Verifies credentials provide administrative access.
- Executes KAPE with configured arguments on target computers.
- Outputs are saved to the specified network share.
- Cleans up by closing PowerShell sessions and removing temporary drives.

Additional Information

- Ensure KAPE is downloaded and accessible as per configurations. Download link: KAPE.

Mount-TriageVhdx.ps1

Synopsis

Mounts VHD files to directories for easier content viewing, supporting both .vhd and .vhdx formats. It assumes specific naming conventions for the files and requires administrative permissions to run.

Description

Designed for incident response, this script mounts .vhd or .vhdx files, collected using Kape, to specified directories. The script supports filenames that start with a timestamp in a specific format and can strip this timestamp if desired. It necessitates running PowerShell as an Administrator.

Parameters

- **VhdDirectory**: The directory containing .vhd or .vhdx files to be mounted. The script does not check subdirectories.
- **MountDirectory**: The target directory where mount points will be created, each named after the source computer of the VHDX image.
- **StripDateTime**: Optional. When used, removes a datetime prefix from the file names when determining computer names.

Example Usage

1. **Without Stripping DateTime**

```
.\Mount-TriageVhdx.ps1 -VhdDirectory "D:\VHDXTriageImagesFolder\" -MountDirectory
"D:\MountedImagesFolder\"
```

This example mounts disk images located in D:\VHDXTriageImagesFolder\ to D:\MountedImagesFolder\.

2. **With Stripping DateTime**

```
.\Mount-TriageVhdx.ps1 -VhdDirectory "D:\VHDXTriageImagesFolder\" -MountDirectory
"D:\MountedImagesFolder\" -StripDateTime
```

This example mounts disk images located in D:\VHDXTriageImagesFolder\ to D:\MountedImagesFolder\ removing the date and time stamp from the beginning of the file name before determining the computer name.

Notes

- Developed by Steve Anson and Dr. Sorot Panichprecha.
- Distributed under the GNU General Public License v3.0.
- The script warns against using it on original evidence and advises on ensuring disk images are initialized.

Operational Steps

- Validates administrative privileges.
- Confirms the presence and accessibility of specified directories.
- Warns the user about the intended use on copies of evidence and the potential change in hash values for uninitialized disk images.
- Optionally strips datetime prefixes from filenames before processing.
- Mounts each VHD or VHDX to a newly created directory within the specified mount directory, named after the source computer.

Important Considerations

- Only .vhd and .vhdx files are supported; other files are ignored.
- The script emphasizes the necessity of administrative rights and careful handling to preserve evidence integrity.

Run-ZimmermanTools.ps1

Synopsis

Automatically runs a selection of Eric Zimmerman's forensic tools on mounted disk images for comprehensive analysis.

Description

This script executes AmcacheParser, AppCompatCacheParser, MFTECmd, PECmd, and optionally EvtxECmd against mounted disk images. It's intended for incident response workflows where disk images are mounted for analysis. The Zimmerman Tools must be present locally.

Parameters

- **MountDirectory**: Specifies the directory containing mounted disk images.
- **ZimmermanToolsDirectory**: Directory where Zimmerman Tools are located.
- **OutDirectory**: Destination for the output CSV files.
- **IncludeEvtxECmd**: Whether to include EvtxECmd in the analysis (default: true).
- **StartDateTime** and **EndDateTime**: Optional parameters to filter logs by time.
- **ReduceEvtxECmdOutput**: Reduces output from EvtxECmd to specified event IDs.
- **Confirm**: Prompts user for confirmation before proceeding.
- **CombineEvtx**: Combines EvtxECmd output into a single CSV (with certain conditions).

Example Usage

1. Minimal parameters, including processing with EvtxECmd:

```
.\Run-ZimmermanTools.ps1 -MountDirectory "E:\MountDir\" -ZimmermanToolsDirectory
"C:\Tools\ZimmermanTools\net6\" -OutDirectory "F:\OutDir"
```

2. Excluding EvtxECmd from processing:

```
.\Run-ZimmermanTools.ps1 -MountDirectory "E:\MountDir\" -ZimmermanToolsDirectory
"C:\Tools\ZimmermanTools\net6\" -OutDirectory "F:\OutDir" -IncludeEvtxECmd $false
```

Notes

- Developed by Steve Anson and Dr. Sorot Panichprecha.
- Released under GNU General Public License v3.0.
- Requires the Zimmerman Tools to be pre-downloaded and extracted locally.

Operational Steps

- Validates the presence of the necessary directories and parameters.
- Confirms user intent if the **-Confirm** switch is used.
- Searches for Zimmerman tool executables within the specified directory.
- Processes each mounted disk image with selected tools based on provided parameters.
- Optionally reduces EvtxECmd output and combines it into a single file.

Important Considerations

- Intended for use with copies of uncompressed evidence files, not original evidence.
- Users are advised to confirm that Zimmerman's tools are up-to-date and downloaded from https://ericzimmerman.github.io/ or other trusted sources.

Run-ZTSrumECmd.ps1

Synopsis

Executes SRUM (System Resource Usage Monitor) analysis on mounted disk images using Eric Zimmerman's SRUMCmd tool, enhancing forensic investigations with detailed resource usage data.

Description

This script is tailored for forensic analysts who require an automated approach to analyze SRUM data from disk images mounted through **Mount-TriageVhdx.ps1**. Utilizing Eric Zimmerman's **SrumECmd**, it processes SRUM data to output detailed CSV reports for each mounted image. It supports optional repair of corrupted **srudb.dat** files, ensuring comprehensive analysis despite potential data integrity issues.

Parameters

- **MountDirectory**: The directory containing mounted disk images, with each image mounted to its subfolder.
- **ZimmermanToolsDirectory**: The directory where Zimmerman's SRUMCmd tool is located. This script requires **SrumECmd.exe** to function.
- **OutDirectory**: The destination folder for the generated CSV files from the SRUM analysis.
- **RepairSrum**: An optional switch that attempts to repair and process a copy of **srudb.dat** if the original file is found to be corrupted.

Example Usage

1. Basic Command:

```
.\Run-ZTSrumECmd.ps1 -MountDirectory "D:\MountedImagesFolder\" -ZimmermanToolsDirectory
"D:\ZimmermanTools" -OutDirectory "F:\OutDir"
```

This command processes all SRUM data from disk images mounted in **D:\MountedImagesFolder\**, using **SrumECmd.exe** located in **D:\ZimmermanTools**, and outputs CSV files to **F:\OutDir**.

2. With SRUM Repair:

```
.\Run-ZTSrumECmd.ps1 -MountDirectory "D:\MountedImagesFolder\" -ZimmermanToolsDirectory
"D:\ZimmermanTools" -OutDirectory "F:\OutDir" -RepairSrum
```

This command includes an attempt to repair corrupted **srudb.dat** files before processing.

Notes

- Developed by Steve Anson.
- Released under the GNU General Public License v3.0.
- Ensure that **SrumECmd.exe** is updated and sourced from a trusted provider.
- Intended for use with uncompressed evidence files in a forensic investigation.

Operational Steps

1. Verifies the existence of the specified directories and **SrumECmd.exe**.
2. Processes each mounted disk image to analyze SRUM data, outputting results to the designated output directory.
3. If enabled, attempts to repair corrupted **srudb.dat** files before analysis, enhancing data recovery efforts.

Important Considerations

- Designed for analysis of post-mounted disk images using **Mount-TriageVhdx.ps1**.
- Best practice involves running this script on copies of evidence to preserve data integrity.

Run-Hayabusa.ps1

Synopsis

Leverages Hayabusa to parse Windows event logs from mounted disk images, generating CSV outputs that highlight detections with a focus on high and critical findings.

Description

The script facilitates the analysis of Windows event logs from disk images that have been prepared with **Mount-TriageVhdx.ps1**, using Hayabusa to parse the logs. It produces individual CSV files for each analyzed system and offers an option to create separate summaries for high and critical detections.

Parameters

- **MountDirectory**: Specifies the directory containing the mounted disk images.
- **HayabusaDirectory**: Location of the Hayabusa executable.
- **OutDirectory**: Destination directory for the CSV output files.
- **HighCrit**: Optional. Determines whether to generate separate CSVs for high and critical detections. Defaults to true.
- **StartDateTime** and **EndDateTime**: Optional. Filters logs based on specified time frames.
- **Confirm**: Optional. Enables a prompt for user confirmation before proceeding.

Example Usage

1. **Default Processing**:

```
.\Run-Hayabusa.ps1 -MountDirectory "D:\MountedImagesFolder\" -HayabusaDirectory
"D:\Hayabusa" -OutDirectory "D:\HayabusaOutput\"
```

This will run the script with only required parameters, and default to also generating a summary of high and critical detections.

2. **Without Additional High and Critical Summaries**:

```
.\Run-Hayabusa.ps1 -MountDirectory "D:\MountedImagesFolder\" -HayabusaDirectory
"D:\Hayabusa" -OutDirectory "D:\HayabusaOutput\" -HighCrit $false
```

This will run the script with the required parameters, but not generate a summary of high and critical detections.

3. **With Time Frame Filtering**:

```
.\Run-Hayabusa.ps1 -MountDirectory "D:\MountedImagesFolder\" -HayabusaDirectory
"D:\Hayabusa" -OutDirectory "D:\HayabusaOutput\" -StartDateTime "2022-01-01 09:00:00
+03:00" -EndDateTime "2022-01-01 09:00:00 +03:00"
```

Filters events between specified start and end times.

Notes

- Co-authored by Steve Anson and Dr. Sorot Panichprecha.
- Released under GNU General Public License v3.0.
- Requires pre-existing disk images mounted via **Mount-TriageVhdx.ps1** and Hayabusa installed in the specified directory.

Operational Steps

- Validates the existence of specified directories and checks for Hayabusa's executable.
- Optionally confirms user intent to proceed.
- Processes each mounted image directory, utilizing Hayabusa for log analysis.
- Generates individual CSV outputs for each system, with optional high and critical detections summaries.
- Offers filtering based on specified time frames to narrow down the analysis scope.

Important Considerations

- Intended for analyzing copies of disk images, not the original evidence.
- Users are encouraged to ensure Hayabusa is updated and obtained from a trusted source, such as https://github.com/Yamato-Security/hayabusa

Run-Winlogbeat.ps1

Synopsis

Facilitates the ingestion of Windows event logs from disk images into Elasticsearch for centralized analysis.

Description

This script leverages Winlogbeat to parse and ingest Windows event logs found in disk images prepared with **Mount-TriageVhdx.ps1**. It requires the configuration of Winlogbeat and Elasticsearch details within a YAML file to successfully send log data to an Elasticsearch instance.

Parameters

- **MountDirectory**: Directory where disk images are mounted, each in its subfolder.
- **WinlogbeatDirectory**: Location of Winlogbeat executable and the YAML configuration file.
- **YamlFile**: The Winlogbeat YAML configuration file (default is **winlogbeat.yml**).

Example Usage

1. **Default Configuration**:

```
.\Run-Winlogbeat.ps1 -MountDirectory "D:\MountedImagesFolder\" -WinlogbeatDirectory
"D:\Winlogbeat"
```

Ingests logs using the default **winlogbeat.yml** configuration file.

2. **Custom YAML Configuration**:

```
.\Run-Winlogbeat.ps1 -MountDirectory "D:\MountedImagesFolder\" -WinlogbeatDirectory
"D:\Winlogbeat" -YamlFile "winlogbeat-forensics.yml"
```

Uses a custom YAML file name for Winlogbeat settings.

Notes

- Developed by Dr. Sorot Panichprecha and Steve Anson.
- Distributed under GNU General Public License v3.0.
- Elasticsearch must have Winlogbeat components installed for successful data ingestion.

Operational Steps

- Validates the existence of specified directories and the Winlogbeat executable.
- Processes each mounted disk image directory, extracting Windows event logs and ingesting them into Elasticsearch.
- Utilizes the YAML configuration file for details about the Elasticsearch setup.

Important Considerations

- The script assumes Winlogbeat and Elasticsearch are correctly configured and operational.
- It is crucial to modify the YAML file with accurate details of the Elasticsearch implementation before running the script.
- Designed to streamline the analysis of event logs from multiple sources, enhancing the incident response process.

Restore-AutoKapeServer.ps1

Synopsis

Reverts most changes made during the setup of an AutoKape server, including the deletion of temporary user accounts, SMB shares, and associated NTFS permissions.

Description

This PowerShell script simplifies the cleanup process after using KAPE for evidence collection on a server by removing temporary modifications. It focuses on user accounts, SMB shares, and NTFS permissions, but does not adjust network configurations or the contents of specific directories used during the collection process.

Parameters

- **SettingsFile**: Path to the Settings.psd1 configuration file. The same Settings.psd1 file must be used with the same values that were used to configure the server with Prepare-AutoKapeServer.ps1.

Example Usage

1. **Default Settings File**:

```
.\Restore-AutoKapeServer.ps1
```

Uses **Settings.psd1** in the current directory to revert changes.

2. **Custom Settings File**:

```
.\Restore-AutoKapeServer.ps1 -SettingsFile C:\Users\user\Desktop\Settings.psd1
```

Reverts server settings using a specified Settings.psd1 file.

Important Considerations

- Requires administrative privileges to execute.
- It does not revert network settings or modify the contents of the KAPE executable or results folders.
- Users are prompted to confirm before the script executes the reversion tasks.

Operational Steps

- Validates administrative rights and the existence of the specified Settings.psd1 file.
- Reads configurations from the Settings.psd1 file.
- Deletes the specified temporary user account and associated SMB shares.
- Removes NTFS permissions granted to the temporary user on designated folders.
- Offers a review of the current network settings post-cleanup as an optional step.

Notes

- Authored by Steve Anson.
- Distributed under the GNU General Public License v3.0.

Dismount-TriageVhdx.ps1

Synopsis

Automates the dismounting of VHD/VHDX images and cleanup of their mount point directories, reversing the actions performed by the **Mount-TriageVhdx.ps1** script.

Description

This PowerShell script dismounts disk images (VHD/VHDX) that were previously mounted to a specified directory for forensic analysis. It cleans up by removing the directories created as mount points. This process requires administrative privileges due to the nature of disk image management and directory removal.

Parameters

- **VhdDirectory**: The directory containing the disk images (VHD/VHDX) to be dismounted.
- **MountDirectory**: The directory where the disk images were mounted, with each image having its subdirectory as a mount point.
- **StripDateTime**: An optional switch to handle file names with a datetime prefix. This helps in identifying and removing the correct mount point directories.

Example Usage

1. **Without DateTime Stripping**:

```
.\Dismount-TriageVhdx.ps1 -VhdDirectory "D:\VHDXTriageImagesFolder\" -MountDirectory
"D:\MountedImagesFolder\"
```

This example dismounts disk images located in **D:\VHDXTriageImagesFolder\** and removes the associated mount points in **D:\MountedImagesFolder\**.

2. **With DateTime Stripping**:

```
.\Dismount-TriageVhdx.ps1 -VhdDirectory "D:\VHDXTriageImagesFolder\" -MountDirectory
"D:\MountedImagesFolder\" -StripDateTime
```

This variant includes the **-StripDateTime** switch to handle file names prefixed with a datetime string, adjusting the removal process of mount point directories accordingly.

Operational Steps

- Validates that the PowerShell session is running with administrative privileges.
- Checks the existence of the specified **VhdDirectory** and **MountDirectory**.
- Iterates over each disk image in the **VhdDirectory**, attempting to dismount it and remove its associated mount point directory from **MountDirectory**.
- If **-StripDateTime** is specified, the script adjusts to process file names with datetime prefixes accordingly.

Important Considerations

- Requires administrative rights to execute properly.

Notes

- Authored by Dr. Sorot Panichprecha and Steve Anson.
- Distributed under the GNU General Public License v3.0, ensuring its free redistribution and modification under the same license terms.