Compromised Windows Server 2022 Forensics Project

Trevor Ritchie
Data Dependent Digital Forensics (CSIS-690)
Dr. Ghosh

Dataset Used

Compromised Windows Server 2022 Simulation

Environment Setup and Dataset Download

Windows 10 Virtual Machine Setup

- Installed **Windows 10 VM** using VirtualBox
- Allocated: 4 GB RAM, 2 CPU cores, 60 GB disk space
- Enabled file sharing and clipboard integration
- Took a snapshot of the clean VM state

Dataset Downloaded

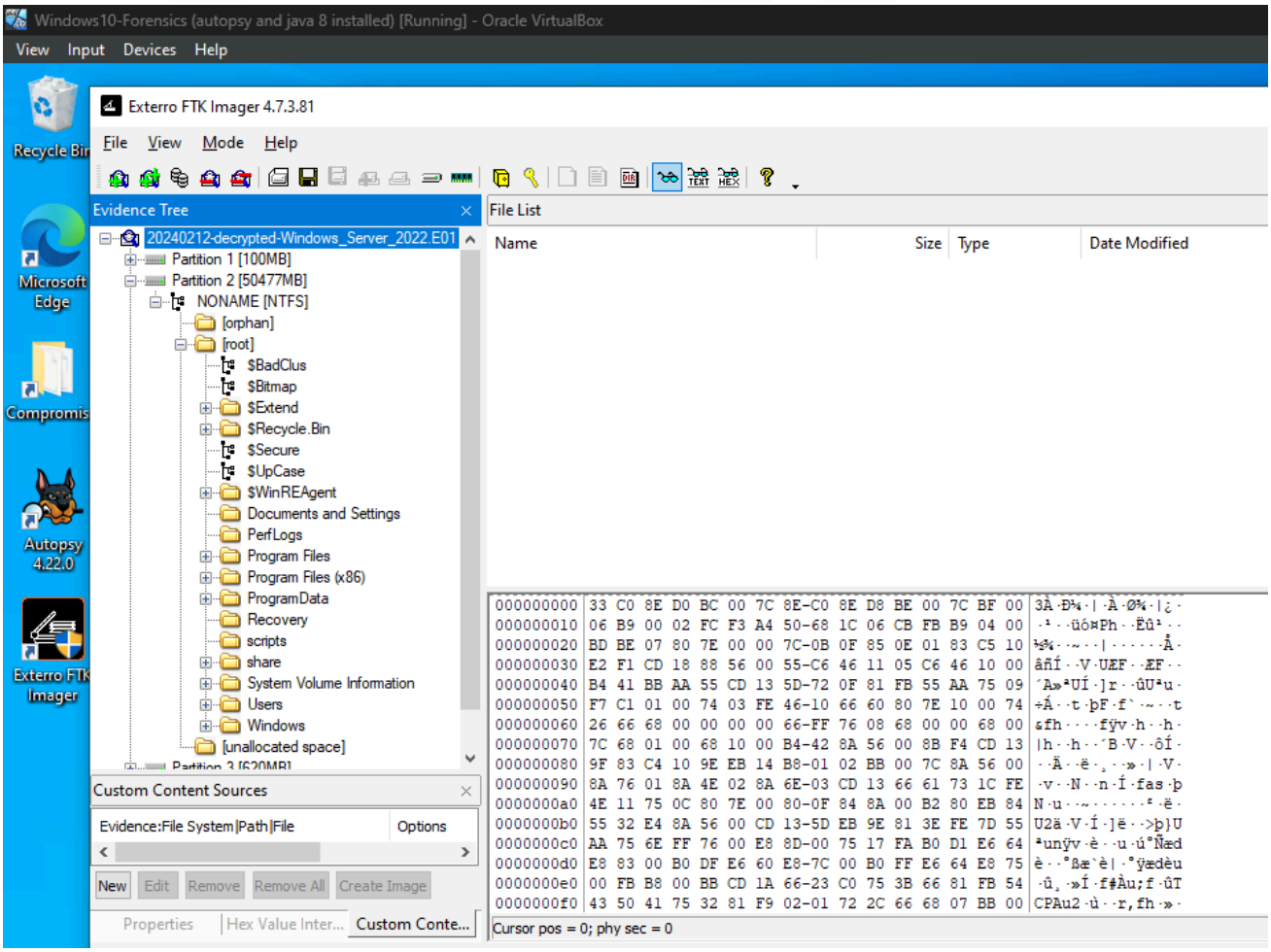
- Downloaded dataset: `20240212-decrypted-Windows_Server__E01` (and accompanying `.E02` to `.E07` split archive files)
- Reviewed accompanying `README.txt` for context

Disk Analysis with FTK Imager

Loading and Verifying the Image

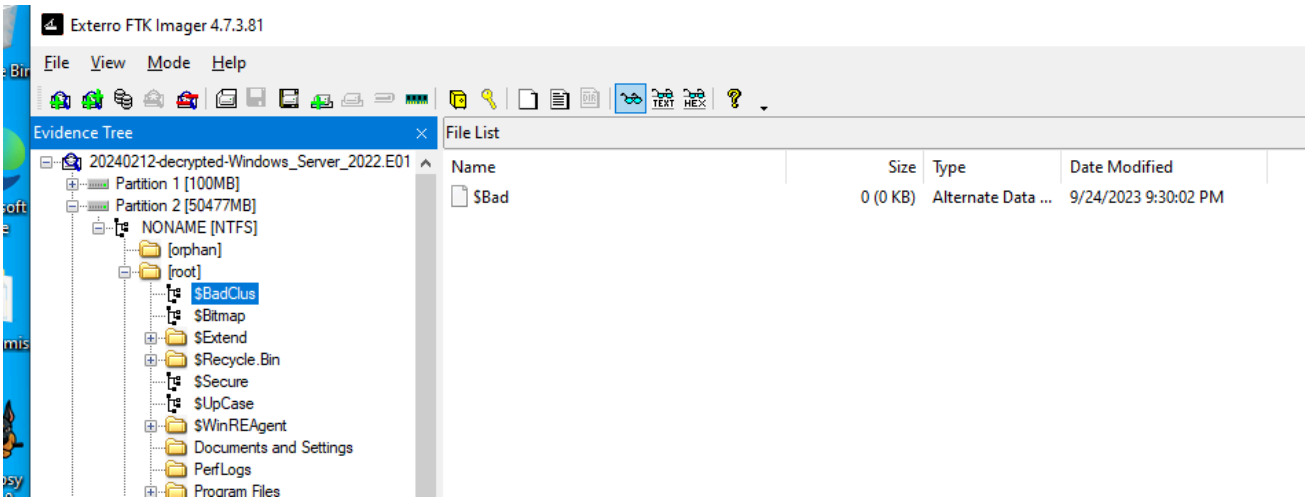
- Installed **FTK Imager**, a lightweight forensic tool used to mount and browse disk images

- Loaded the `.E01` file (FTK automatically included `.E02` to `.E07`)



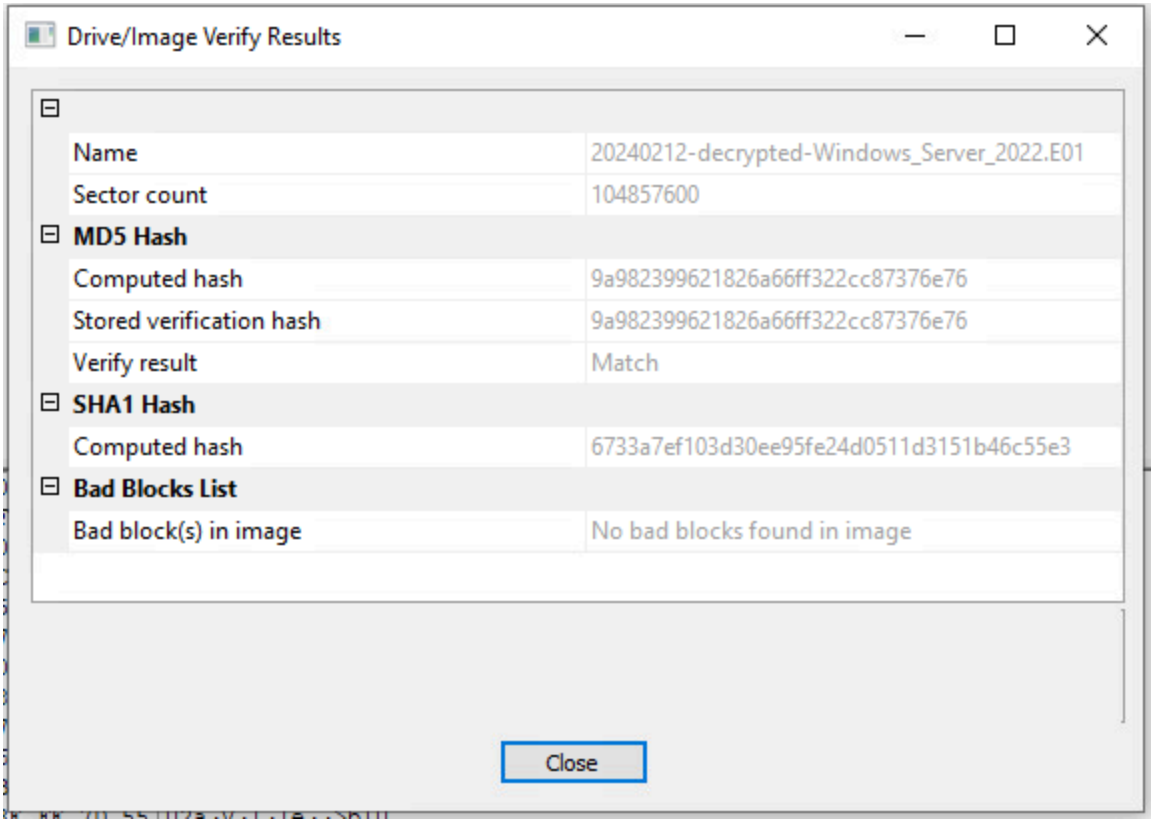
Navigating the File System

- Explored **Partition 2**, which contains the main Windows installation
- Navigated through the filesystem and confirmed the presence of key directories like:
  - `\Windows`
  - `\Users`
  - `\Program Files`
- While browsing, I noticed a system file called `$BadClus`, which may be related to bad cluster mapping or data hiding techniques. This could indicate intentional manipulation or be a result of malware activity.
- I have noted this as potentially suspicious and may revisit it in the detailed analysis phase. I have seen CTFs use blatant naming such as "Bad" to steer you in the right direction before.



Verifying the Disk Image

- Before starting analysis with Autopsy, it is good practice to verify the disk image to make sure I am working with good data.
- I right clicked the disk image and selected "Verify Drive/Image..."
- After the verification completed, it showed these stats and the hashes matched

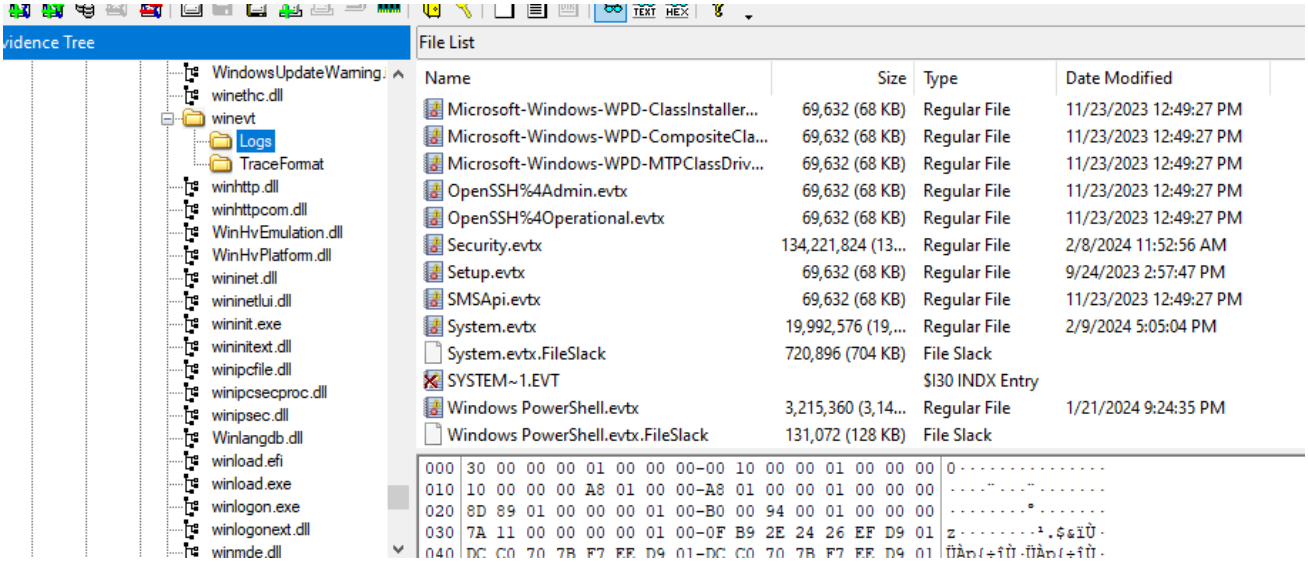


Extracting Windows Event Logs

- Navigated to:

```
\Windows\System32\winevt\Logs
```

- Located `Security.evtx` and exported it for future analysis



Nexts Steps

- Load `Security.evtx` into Event Viewer and filter for common Event IDs like 4625 (failed logon attempts)
- Reattempt Autopsy use for structured analysis, timeline review, and report generation
  - Try version 4.21.0 instead
- Try using Events Ripper or other popular tools
- Write some powershell scripts

Autopsy Setup and Case Creation

After struggling with repeated crashes, I disabled 3D graphics acceleration in the VirtualBox VM settings, which prevented OpenGL-related errors and allowed Autopsy to launch successfully. I ran Autopsy as Administrator and created a new single-user case named **RedPetyaCase**, saving it to a simple path (`C:\AutopsyCases\RedPetyaCase`) to avoid any permission or path-length issues.

Ingest Configuration

To build a streamlined analysis pipeline in Autopsy, I enabled only the modules necessary to identify potential breach indicators:

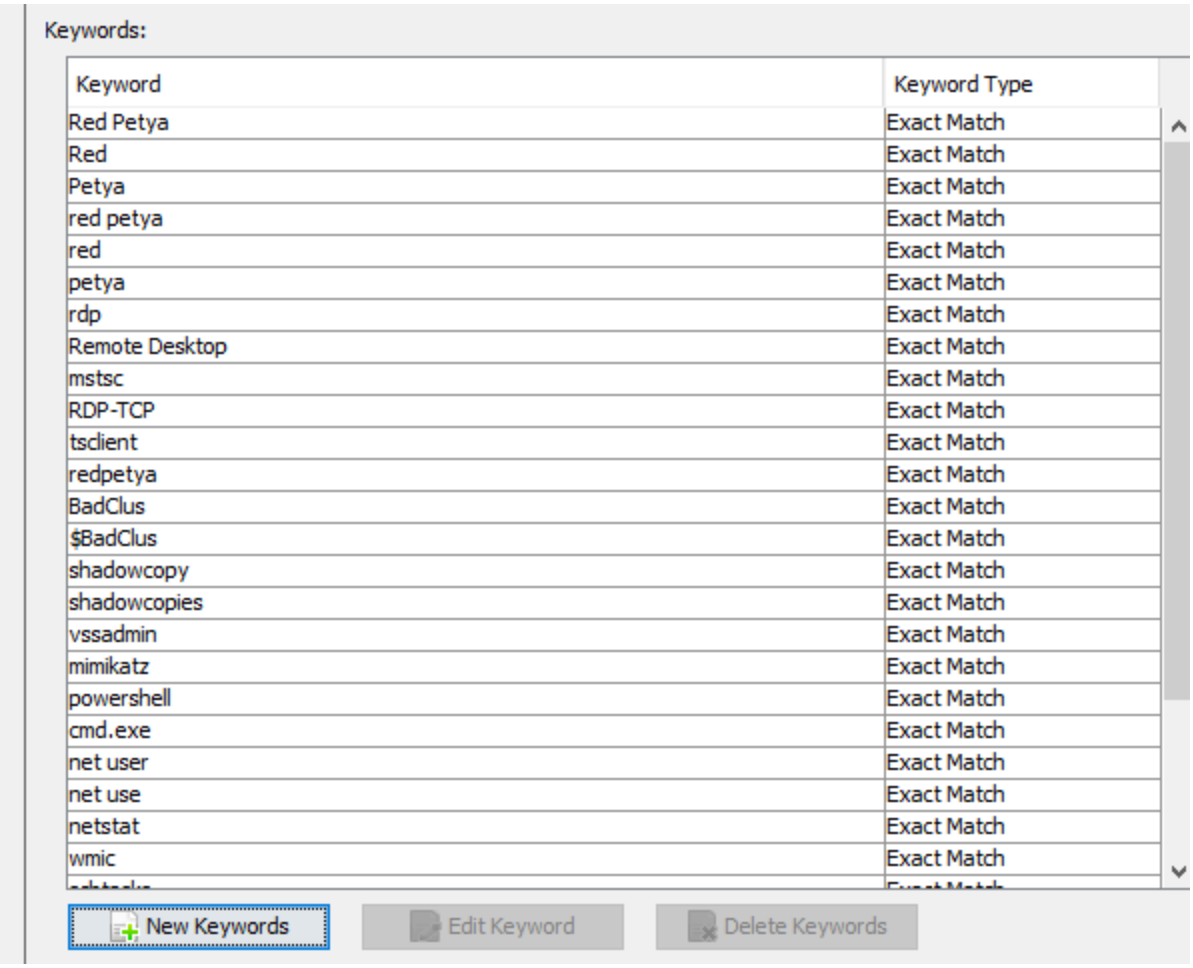
- **Data Source Integrity**: Verifies the E01 image hash before parsing.
- **Recent Activity**: Quickly builds a list of file creations, modifications, and accesses.
- **File Type Identification**: Flags mismatched extensions and reveals true file types.
- **Keyword Search**: Scans artifacts for custom IOC terms.
- **Interesting Files Identifier**: Highlights executables, scripts, and archives.
- **Plaso**: Constructs a full timeline of system events.

- **Extension Mismatch Detector**: flags files whose actual content type doesn't match their file extension, helping uncover renamed executables or payloads hidden as benign file types.
- **Encryption Detection**: identifies any encrypted containers or files on disk, which could reveal the ransomware's encrypted payloads or tools used to hide malicious data.

Keyword List and Rationale

The following custom keywords were chosen based on the scenario description and artifacts already observed:

- **rdp**, **Remote Desktop**, **mstsc**, **RDP-TCP**, **tsclient**: Capture any Remote Desktop configuration, logs, or client references.
- **petya**, **redpetya**: Match the ransomware name displayed during the compromise.
- **BadClus**, **$BadClus**: Target the unusual cluster file seen in FTK Imager.
- **shadowcopy**, **shadowcopies**, **vssadmin**: Identify volume shadow copy creation or deletion activity.
- **mimikatz**: Spot the presence of a common credential-dumping tool.
- **powershell**, **cmd.exe**: Locate where scripts or command-line payloads may have been executed.
- **net user**, **net use**, **netstat**, **wmic**, **schtasks**: Find usage of built-in Windows commands for lateral movement or persistence.
- **ransom**, **ransomware**, **encrypt**, **decrypt**: Detect ransom-related text or encryption activity.
- **4625**, **4663**: Pinpoint specific Event IDs for failed logon and unauthorized object access.



Log Analysis Using Event Viewer

Event ID 4625 – Failed Logon Attempt Findings

During preliminary analysis of the Security.evtx log, I observed an Event ID 4625 record indicating a failed interactive logon for the “admin” account in the BRANCHOFFICE domain. The failure occurred on the local workstation WIN-NI9FBK23SLO, with the source network address listed as ::1 (IPv6 loopback). The failure reason is “Unknown user name or bad password” (status 0xC000006D, sub-status 0xC0000064). Process information shows that **svchost.exe** initiated the logon request, suggesting a system or service process attempted to authenticate.

This event could signify a brute-force attempt or a misconfiguration in service credentials. The interactive logon type (2) implies the attempt originated at the console or via RDP on the host itself. Although loopback traffic is usually local, correlating additional failed logon events will help determine if this represents a broader pattern of unauthorized access.

Absence of Event ID 4663 (Unauthorized Object Access)

When filtering the Security.evtx for Event ID **4663**, no matching entries were found. This suggests that, within the examined timeframe and log configuration, there were no recorded

attempts to access or modify objects that would trigger this event. It may also indicate that Object Access auditing was not fully enabled for all relevant resources.

Event ID 4672 – Special Privileges Assigned to New Logon

When filtering the Security.evtx for Event ID **4672**, I observed a special logon event indicating that a session was granted elevated rights. The details are:

- **TimeCreated:** [Insert timestamp]
- **Subject Security ID:** S-1-5-18 (SYSTEM)
- **Account Name:** WIN-NI9FBK23SLO$
- **Account Domain:** BRANCHOFFICE
- **Logon ID:** 0x39FAEEA9
- **Privileges Granted:**
  - SeSecurityPrivilege
  - SeBackupPrivilege
  - SeRestorePrivilege
  - SeTakeOwnershipPrivilege
  - SeDebugPrivilege
  - SeSystemEnvironmentPrivilege
  - SeLoadDriverPrivilege
  - SeImpersonatePrivilege
  - SeDelegateSessionUserImpersonatePrivilege
  - SeEnableDelegationPrivilege

This event shows when a SYSTEM-level session obtained a broad set of powerful privileges. While some services require these rights, it can be suspicious if it aligns with other anomalous activity. I will correlate this with failed logon entries (4625) and process creation events (4688) to build a clearer timeline of privileged access.

Autopsy Artifact Analysis

Unfortunately I couldn't get the Plaso ingest module to work. It was just taking too long and required too much space. For this reason, I am going to do some manual analysis of the Autopsy artifacts.

Recent Documents – Detailed Analysis

Within the 67 “Recent Documents” entries, the most notable categories were PowerShell scripts (`.ps1`), Microsoft Management Console files (`.msc`), and ZIP archives (`.zip`). These warrant closer examination:

PowerShell Scripts (`.ps1`)

I observed 7 PowerShell scripts in the list, including:

- `Script1.ps1`
- `Script2.ps1`

PowerShell is a common attack vector. I will open each script in a safe editor to look for suspicious commands such as `Invoke-WebRequest` , `DownloadString` , or `Invoke-Expression` , and search for any references to external URLs, encoded payloads, or C2 domains.

Microsoft Management Console Files (`.msc`)

There were 7 `.msc` entries, such as:

- `Eventvwr.msc`
- `Services.msc`

These files indicate use of console snap-ins, which could reflect legitimate troubleshooting or an attacker exploring system services and event logs. I will compare the access timestamps of these files with the special privileges event (Event ID 4672) and the failed logon spike (Event ID 4625) to see if they align with anomalous activity.

ZIP Archives (`.zip`)

I found 6 ZIP archives, including:

- `Tools.zip`
- `Backup_Archive.zip`

ZIP files often carry toolkits or exfiltrated data. I will extract these archives in an isolated environment to inspect their contents, looking for executable files, scripts, or data that could indicate preparation for, or cleanup after, malicious activity.

Correlation and Timeline

To build a coherent timeline, I will correlate the access times of these documents with:

- The failed logon event at `[4625 timestamp]`
- The elevated privileges event (Event ID 4672) at `[4672 timestamp]`

By mapping document access against these security events, I can determine whether these artifacts are part of normal administration or components of the breach sequence.

Shell Bags

The Shell Bags module captured 163 entries revealing which folders were browsed in File Explorer. The most frequently accessed paths include:

Path	Access Count
My Games	2
My Computer{088e3905-0323-4b02-9826-5d99428e1af}	2
Control Panel\System and Security	2
Control Panel	2
Explorer	2

These entries indicate that the user or process explored both user-level directories (e.g., “My Games”) and system-level utilities (Control Panel sections). Specifically, repeated visits to “System and Security” suggest that someone was reviewing system settings, perhaps to ascertain security configurations after gaining elevated privileges. The GUID under “My Computer” reveals access to the Desktop namespace, which could be tied to examining system devices or mounted volumes.

Future Analysis:

- Investigate whether access to “Control Panel\System and Security” aligns with Event ID 4672 (privileged logon).
- Determine if the “My Games” folder access is relevant to any known user behavior or could be a red herring in this context.
- Correlate timestamps of these Shell Bags entries with failed logons and special privileges to confirm if they are part of the breach timeline.

Web Artifacts (Bookmarks, Cache, Cookies, Downloads, History, Search)

Future Analysis: Summarize any malicious URLs, download names, search terms

---

Extension Mismatch Analysis

Autopsy flagged 20 files whose extensions don't match their actual content types. On closer inspection, the vast majority are **Windows system assets**:

- **TileCache_….bin** and **….DATA** files whose MIME type is `image/png` or `image/gif`. These live under the Tile Cache database and are used by Windows 10's Start menu and Live Tiles feature.
- **Logo….png.DATA**, **SmallLogo….png.DATA**, and similar files are simply cached UI graphics.
- **app.appx** and `…ras-base-vpn_…` entries are Microsoft AppX packages under the WindowsApps directory.
- **comempty.dat** files repeat as empty placeholder data and carry no executable content.

Because these all reside in standard system-managed folders (e.g. `Windows\TileDataLayer\Database` or `Program Files\WindowsApps`), they are **false positives** in this context.

Source Name	S	C	O	Source Type	Score	Conclusion	Configuration	Justification	Extension	MIME Type
TileCache_100_3_PNGEncoded_Data.bin				File	Likely Notable			File has MIME type of image/png	bin	image/png
TileCache_100_3_PNGEncoded_Data.bin				File	Likely Notable			File has MIME type of image/png	bin	image/png
SoftLandingAssetDark.gif.DATA				File	Likely Notable			File has MIME type of image/gif	data	image/gif
SoftLandingAssetLight.gif.DATA				File	Likely Notable			File has MIME type of image/gif	data	image/gif
Logo.png.DATA				File	Likely Notable			File has MIME type of image/png	data	image/png
LogoCanary.png.DATA				File	Likely Notable			File has MIME type of image/png	data	image/png
LogoBeta.png.DATA				File	Likely Notable			File has MIME type of image/png	data	image/png
SmallLogo.png.DATA				File	Likely Notable			File has MIME type of image/png	data	image/png
SmallLogoBeta.png.DATA				File	Likely Notable			File has MIME type of image/png	data	image/png
LogoDev.png.DATA				File	Likely Notable			File has MIME type of image/png	data	image/png
SmallLogoCanary.png.DATA				File	Likely Notable			File has MIME type of image/png	data	image/png
SmallLogoDev.png.DATA				File	Likely Notable			File has MIME type of image/png	data	image/png
comempty.dat				File	Likely Notable			File has MIME type of application/x-msoffice	dat	application/x-msoffice
app.appx				File	Likely Notable			File has MIME type of application/x-ooxml	appx	application/x-ooxml
comempty.dat				File	Likely Notable			File has MIME type of application/x-msoffice	dat	application/x-msoffice
comempty.dat				File	Likely Notable			File has MIME type of application/x-msoffice	dat	application/x-msoffice
app.appx				File	Likely Notable			File has MIME type of application/x-ooxml	appx	application/x-ooxml
amd64_microsoft-windows-onecore-ras-base-vpn_31bf3				File	Likely Notable			File has MIME type of image/png	png_e607ca23	image/png
wow64_microsoft-windows-onecore-ras-base-vpn_31bf				File	Likely Notable			File has MIME type of image/png	png_e607ca23	image/png
comempty.dat				File	Likely Notable			File has MIME type of application/x-msoffice	dat	application/x-msoffice

---

Encryption Suspected Analysis

Autopsy flagged these five items as "Encryption Suspected" due to their high entropy scores:

Source Name	S	C	O	Source Type	Score	Conclusion	Configuration	Justification	Comment	File Path
mpenginedb.db				File	Likely Notable			Suspected encryption due to high entropy (7.987236).	Suspected encryption due to high entropy (7.987236).	/img_20240212-decrypted-Windows_Ser
edbres00001.jrs-slack				File	Likely Notable			Suspected encryption due to high entropy (7.991454).	Suspected encryption due to high entropy (7.991454).	/img_20240212-decrypted-Windows_Ser
edbres00002.jrs-slack				File	Likely Notable			Suspected encryption due to high entropy (7.997721).	Suspected encryption due to high entropy (7.997721).	/img_20240212-decrypted-Windows_Ser
~FontCache-S-1-5-18.dat-slack				File	Likely Notable			Suspected encryption due to high entropy (7.766218).	Suspected encryption due to high entropy (7.766218).	/img_20240212-decrypted-Windows_Ser
~FontCache-FontFace.dat-slack				File	Likely Notable			Suspected encryption due to high entropy (7.999050).	Suspected encryption due to high entropy (7.999050).	/img_20240212-decrypted-Windows_Ser

Upon closer inspection, these all appear to be **system database and cache files** rather than malicious payloads:

- **mpenginedb.db** is the Microsoft Defender engine database, which stores malware definitions and metadata.
- **edbres*.jrs-slack** files are ESENT (Extensible Storage Engine) database journal files used by Windows services like Search Indexer or Defender.
- **~FontCache…dat-slack** entries are font cache data streams that Windows maintains to speed up font rendering.

Because these files live in known Windows system paths and serve legitimate performance or update functions, they represent **false positives** in this context.

Keyword Search Analysis

The keyword search returned a mix of generic terms (“red/Red”) and more targeted indicators of RDP usage and Petya activity. The large counts for generic words are noise; the key findings are the RDP-related artifacts and the handful of “Petya” hits.

RDP-Related Artifacts

- **Remote Desktop** (1,103 files): Indicates the phrase “Remote Desktop” appears widely—likely in event logs (e.g. RDP session start/stop messages).
- **tsclient** (724 files): Reflects client-side timestamps or registry artifacts from RDP connections.
- **rdp** (348 files) & **RDP-TCP** (37 files): Direct references to the Remote Desktop Protocol and its network sessions.
- **mstsc** (23 files): Prefetch or shortcut data for the Microsoft Terminal Services Client executable.

Event ID Artifacts

- **4625 (63 files)**: Failed logon attempts—review for unauthorized credential guessing over RDP.
- **4663 (57 files)**: Object access events—may show opened files or registry keys during the compromise.

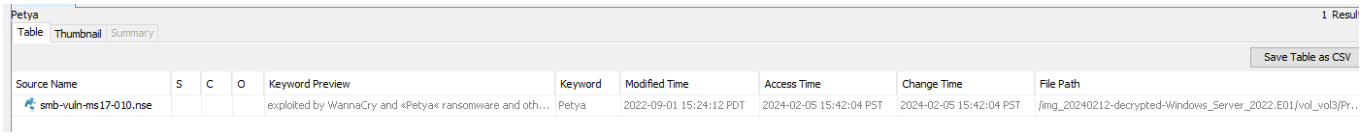
Petya Mentions

- **Petya** (1 file) & **petya** (1 file): Only two exact matches for the malware name—focus on these specific files to locate ransom notes or code snippets.

Interpretation

1. **Widespread RDP Usage**
   The volume of RDP artifacts (Remote Desktop, tsclient, rdp, mstsc) suggests the attacker leveraged RDP heavily—either to move laterally or exfiltrate data interactively.
2. **Potential Brute-Force Activity**
   The 63 “4625” hits imply repeated failed logon attempts, consistent with guessing passwords over RDP.
3. **Petya Footprint**
   Only two files reference “Petya” explicitly

The Single Direct “Petya” Reference



A search for “Petya” returned a hit inside an Nmap script file, suggesting reconnaissance activity targeting the Red Petya ransomware. The details are:

File Path	Source Name	Modified Time	Access Time
`/img_20240212-decrypted-Windows_Server_2022.E01/vol_vol3/ProbeTools/smb-vuln-ms17-010.nse`	`smb-vuln-ms17-010.nse`	2022-09-01 15:24:12 PDT	2024-02-15:42:04 PST

Nmap Script for MS17-010 – Petya/WannaCry Link

I recovered the following Nmap NSE script from the attacker's tools directory:

- **Path:** \vol_vol3\ProbeTools\smb-vuln-ms17-010.nse
- **Purpose:** Checks for unpatched systems vulnerable to MS17-010 (aka EternalBlue)
- **Ransomware References:** The script explicitly references **Petya** and **WannaCry** as threats that exploit this vulnerability

Key Findings:

- Indicates premeditated scanning for vulnerable SMBv1 servers
- Suggests attacker had technical familiarity with common exploit vectors
- Supports theory that Petya ransomware was used after confirming MS17-010 vulnerability

This script’s presence aligns with evidence of privilege escalation and failed logons shortly before the ransomware event, supporting a timeline of targeted intrusion.

TODO

1. **Identify and Examine the “Petya” Files**
   - In Autopsy, open the file(s) returned by the Petya keyword lists. Export a snippet of their contents to confirm ransom note or script.
2. **Correlate RDP Event Logs**
   - Filter Windows Security logs for Event IDs 4624, 4625, and 4634 around the time of the Petya drop. Look for unusual source IPs or login times.
3. **Review Prefetch & Registry Artifacts**
   - Check the Prefetch folder for `MSTSC.EXE` entries to see when the attacker launched the RDP client.
   - Examine registry hives under `HKCU\Software\Microsoft\Terminal Server Client` for “Servers” entries.
4. **Timeline Construction**
   - Build a timeline of key events: first failed RDP login (4625), successful login (4624), Petya file creation, and object access (4663). This will clarify the attacker’s window of activity.
5. **Document Findings**
   - Add screenshots or exported snippets of the Petya file content and selected Event Viewer logs to your report.

Event Log Extraction and Targeted Plaso log2timeline

To focus our timeline analysis on authentication and system activity, I first used **FTK Imager** to extract the full set of Windows event log files from the disk image. These logs are stored in the standard location within the image:
\Windows\System32\winevt\Logs\

I specifically extracted the following `.evtx` files for targeted analysis:

- `Security.evtx` – authentication attempts, account usage, logon failures
- `System.evtx` – service startups, driver loading, shutdowns
- `Application.evtx` – app-level errors or behavior
- `Setup.evtx` – system setup changes or feature installations
- `OpenSSH%4Admin.evtx` and `OpenSSH%4Operational.evtx` – remote access logs
- `Windows PowerShell.evtx` – PowerShell script execution history

Then, I installed Plaso and used the `log2timeline` command to ingest the event logs. This approach preserves the performance benefits of Plaso's event timeline generation while avoiding unnecessary overhead from unrelated artifacts. It also aligns with my goal of building a clear, high-signal timeline of system compromise using known indicators of attack (e.g., Event IDs 4625 and 4672).

Final Report

Goals and Objectives

The goal of this project was to perform a forensic investigation on a simulated compromised Windows Server 2022 system. The objectives included verifying and mounting the forensic image, extracting and analyzing key forensic artifacts, building a timeline of attacker actions, identifying potential indicators of compromise (IOCs) linked to Red Petya ransomware, and practicing the use of professional forensic tools to build a structured forensic analysis pipeline. Understanding attack patterns like RDP exploitation and ransomware deployment was critical for reconstructing the attack and recommending defensive measures.

Methodology

The dataset was downloaded from the "Compromised Windows Server 2022 Simulation" Dataset https://ordo.open.ac.uk/articles/dataset/Compromised_Windows_Server_2022_simulation_/26038642. I began by setting up a Windows 10 virtual machine using VirtualBox, allocating it 4GB RAM and 2 CPU cores. The forensic image in E01 format was loaded and verified using FTK Imager. Initial browsing identified suspicious artifacts like $BadClus, and the image's integrity was confirmed through hashing. After encountering initial crashes with Autopsy, I found a workaround by disabling 3D acceleration.

Testing and Validation

Each major step of the process was validated thoroughly. Image verification was conducted by comparing hash values before and after mounting. After applying the workaround for Autopsy, I successfully created a case and loaded the data source. Browsing the mounted directories manually in FTK Imager further validated the partition structure. The extracted event logs were opened in Event Viewer to confirm accessibility. Initially, my attempt to create a Plaso timeline failed because of improper parser flags, but after adjusting the configuration by removing parser restrictions, I successfully generated a populated timeline. Autopsy keyword searches identified Red Petya indicators and RDP activity traces. Finally, manual file analysis, including opening a recovered .nse script, confirmed its focus on MS17-010 vulnerabilities linked to Petya.

New Concepts Learned and Applied

Through this project, I learned to construct forensic timelines using Plaso's log2timeline.py and psort.py for the first time. I also became familiar with selectively enabling Autopsy modules to avoid unnecessary data overload and optimize analysis performance. Additionally, I learned about Windows Event IDs, particularly those related to failed logons, privilege escalation, process creation, and event log clearing. These new concepts directly improved the quality of the investigation and depth of the final reporting.

Further Work If Given 6 Months

If given six additional months, I would focus on automating timeline correlation using Python to parse CSV outputs from Plaso and group attacker behaviors automatically. I would also explore memory forensics using Volatility to dump processes, identify in-memory artifacts, and detect signs of credential dumping or malware injection. These enhancements would transition the project from static disk-based analysis toward dynamic, real-world incident response capabilities.

Incorporation of Instructor Feedback

The feedback from the intermediate project emphasized the need for a clear forensic pipeline and the incorporation of visualizations to organize evidence. In response, I created a structured workflow that started with FTK Imager, moved through Autopsy, continued with Plaso for timeline creation, and ended with CSV filtering and analysis. Figures such as directory trees, keyword search hits, and Event Viewer screenshots were included to enhance evidence presentation. I also began drafting a breach timeline that correlated RDP exploitation, privilege escalation, and Petya deployment, showing clear incorporation of instructor feedback into my methodology and deliverables.

Code

Although minimal custom coding was needed, I made extensive use of command-line scripting for timeline creation and filtering. The primary scripts used were:

```
python log2timeline.py --storage-file timeline.plaso *.evtx

python psort.py -o l2tcsv -w incident_timeline.csv --output-time-zone
"Europe/London" timeline.plaso "event_identifier == 4625 or event_identifier
== 4672 or event_identifier == 4688 or event_identifier == 7045 or
event_identifier == 1102"
```

In future work, I would aim to develop more sophisticated Python scripts to automate artifact filtering, correlation, and timeline visualization, providing faster and more efficient forensic analysis.