

Incident Response Simulation with



Agent Tesla Behavioral Detection

By: Yaaseen Sheriff



DISCLAIMER

This project involves executing malware for the purpose of **learning cybersecurity detection and response techniques**. Please **DO NOT attempt this on your regular PC or work machine**.

This activity must be conducted strictly in a sandboxed or virtualized environment (e.g., VirtualBox or VMware with host-only networking and no internet access). Executing malware like Agent Tesla outside of a properly isolated lab environment can lead to serious consequences, including but not limited to:

- Compromising your host operating system
- Data leakage or credential theft
- Unintentional propagation of the malware
- Network infection or irreversible system damage

In professional environments, malware analysis is performed in **isolated lab setups** to ensure safety. **Always follow the same best practices at home.**



Never run malware on your local operating system or connected to your personal network. The risks are real — and so are the consequences.

By following proper isolation methods, you not only protect your system but also train under conditions that mirror real-world incident response workflows.

Table of Contents

| | |
|--|-----------|
| 1. Introduction | 1 |
| 1.1. Purpose of the Project | |
| 2. Prerequisites | 4 |
| 2.1. System Requirements | |
| 2.2. Tools Used | |
| 2.3. Skills Needed (including encouragement for beginners and the importance of curiosity) | |
| 3. Environment Setup | 6 |
| 3.1. Windows 10 VM (Host-Only Configuration) | |
| 3.2. Snapshot Strategy (Pre & Post Execution) | |
| 3.3. Obtaining Agent Tesla Sample (with safe-use hyperlink) | |
| 3.4. Splunk Installation (Local) | |
| 4. Malware Execution & Log Generation | 10 |
| 4.1. Executing Agent Tesla in the VM | |
| 4.2. Verifying Post-Execution Activity | |
| 4.3. Snapshot Management for Safe Rollback | |
| 5. Log Collection | 12 |
| 5.1. Exporting Windows Event Logs (.EVTX) | |
| 5.2. Types of Logs Collected: Application, Security, Setup, System | |
| 6. Splunk Configuration & Data Ingestion | 15 |
| 6.1. Creating a Custom Index (agent_tesla_lab) | |
| 6.2. Uploading EVTX Files to Splunk | |
| 6.3. Parsing and Verifying Log Data | |
| 7. Detection & Investigation | 20 |
| 7.1. Key EventCodes Queried (e.g., 4688) | |
| 7.2. Keyword & Pattern-Based Searches | |
| 7.3. Working with Raw Logs | |
| 7.4. Sample Query Results | |
| 8. Response Recommendations | 25 |
| 8.1. Suggested Containment Actions | |
| 8.2. Lessons Learned | |
| 9. Challenges & Workarounds (Recap) | 27 |
| 9.1. Missing Field Values | |
| 9.2. Snapshot Confusion (Pre vs Post Execution) | |
| 9.3. Search Query Limitations | |
| 9.4. Manual Pattern Matching | |
| 10. Conclusion | 28 |
| 10.1. Summary of Outcomes | |
| 10.2. Next Steps for Improvement | |
| 11. Appendix | 29 |
| 11.1. Screenshots | |
| 11.2. Queries Used | |

1. Introduction

This project simulates a real-world incident response workflow using Splunk and log data generated from executing Agent Tesla, a known information-stealing malware, in a controlled Windows 10 virtual machine. The goal is to practice detection, triage, and analysis of malicious activity using .evtx logs and to gain familiarity with core processes that Security Operations Center (SOC) analysts and incident responders follow in live environments.

Who This Project is For

This exercise is intended to benefit:

- Aspiring cybersecurity professionals seeking hands-on experience with SIEM tools like Splunk.
- Learners trying to bridge the gap between theoretical knowledge and practical incident response workflows.
- Anyone looking to understand how raw Windows logs can be used to uncover suspicious behavior — especially when that behavior doesn't come with a flashing red warning label.
- And yes, anyone who wants to legitimately and confidently include “Splunk” on their resume — this one's for you.

Why This Project Matters

The project highlights key elements of process monitoring, malware behavior detection, and log correlation. By analyzing Windows Event Logs (especially those related to process creation like EventCode=4688), this simulation reinforces the concepts of host-based forensics and showcases how Splunk can be used to:

- Parse and analyze system activity,
- Identify Indicators of Compromise (IOCs), and

- Document findings in a clear, reportable format.

Even though it doesn't replicate a full-scale enterprise attack, this project captures an essential slice of the incident response cycle: detection → investigation → recommendation.

What I Learned & Where I Struggled

While the objective was clear at the start, the journey wasn't entirely smooth. Some of the key challenges I faced included:

- Snapshot issues – I initially used a VM snapshot from *before* the malware was executed, so the logs were practically useless. I had to learn the hard way to validate timing and create post-execution snapshots.
- Parsing limitations in Splunk – Some queries didn't return structured fields like `ParentImage` or `CommandLine`. I had to adapt by using broader keyword-based searches on the raw logs.
- Trial-and-error with log ingestion – I re-exported and re-uploaded `.evtx` files a couple of times before getting usable post-execution data.
- The "invisible malware" problem – Agent Tesla didn't make itself obvious. There was no `agent_tesla.exe` in the logs, no giant red flag. This project reminded me that in real-life incident response, you won't always see malware by name. Instead, you need to infer its presence through behaviors — suspicious `.exe` launches, odd directories like `AppData`, or anomalous process trees.

Note: Agent Tesla isn't going to tell you it's Agent Tesla — and that's the point. This is exactly how real-world malware hides, and it's what makes this exercise so valuable.

These obstacles pushed me to think like an analyst, work through data limitations, and gain insight into how nuanced real threat detection can be.

Value to Others

If you're pursuing a role in incident response, SOC analysis, or any blue team position, this project will:

- Build your familiarity with Splunk search queries, syntax, and workflow.
- Show you how to work with incomplete or imperfect data — a reality in most investigations.
- Teach you to document your findings clearly, a key skill in any professional setting.

Finally, this project proves that you don't need a full enterprise environment to simulate meaningful detection scenarios. Just a VM, a bit of malware, and a tool like Splunk — and you're already learning the mindset and skill set of a blue teamer.


2. Prerequisites & Environment Setup

Before diving into the investigation, make sure you have the right tools, environment, and mindset. This section outlines what you need and how to set it up.

2.1. System & Virtual Lab Requirements

To replicate this project safely, you'll need:

- A host machine with at least **8 GB of RAM** and **50 GB of free space**
- **Oracle VirtualBox** or **VMware** for virtualization
- A **Windows 10 virtual machine** configured with a **host-only network adapter**
 - This ensures the VM is isolated from the internet, preventing malware from reaching beyond the sandbox
- **Splunk Enterprise (Free Trial)** installed on the host system or a second VM

 Never run malware on your host machine. Always use a virtual, isolated environment.

2.2. Tools Used

This project relies on the following tools:

- **Windows 10 VM** – controlled environment to execute malware safely
 - **Splunk Enterprise** – to ingest .evtx logs and perform threat detection queries
 - **Windows Event Viewer** – used to export Security, Application, Setup, and System logs
 - **Agent Tesla sample** – an infostealer malware used for simulation
-

2.3. Skills Needed (and Not Needed!)

You'll get the most out of this project if you're comfortable with:

- Basic understanding of **Windows Event Logs** (especially EventCode 4688)
- Familiarity with virtual machines and navigating a Windows OS
- Curiosity and a detective mindset — seriously, a curious mind goes a long way!

Don't know Splunk SPL (Search Processing Language)? No worries!

This project stays mostly beginner-friendly. While it introduces some SPL basics like filtering, keyword searches, and field extractions, it avoids overly complex syntax.

If you're familiar with SQL, you'll feel right at home — SPL uses a similar structure (think: SELECT, WHERE, GROUP BY, etc.), just tailored to log data instead of tables.

Even when raw logs are involved, you'll be guided through simple rex (regular expression) commands to extract useful fields. Think of it as a **soft landing into the world of SIEM queries** — just enough to get started with detection and investigation.

3. Environment Setup

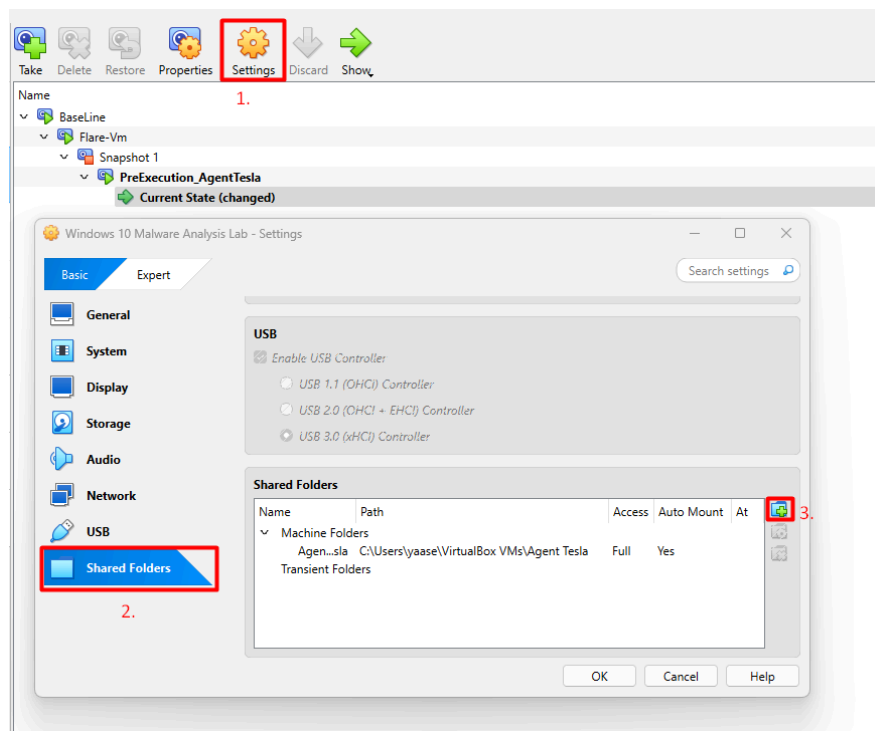
3.1. Windows 10 VM (Host-Only Configuration with Shared Folder)

To isolate malware safely, we use a **Windows 10 Virtual Machine** with a **Host-Only Adapter**:

- This setup prevents the VM from accessing the internet, cutting off communication with potential command-and-control (C2) servers.
- A **shared folder** is also configured to transfer files (like ZIPs) between host and guest systems without using USB or the internet.

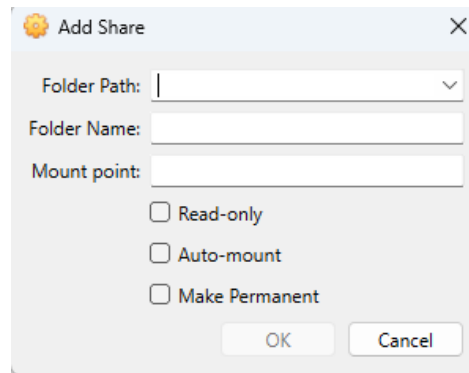
Steps to Set Up Shared Folder in VirtualBox:

1. Open VirtualBox → Select your VM → Click **Settings(1)**
2. Go to **Shared Folders(2)** tab → Click the + icon(3)

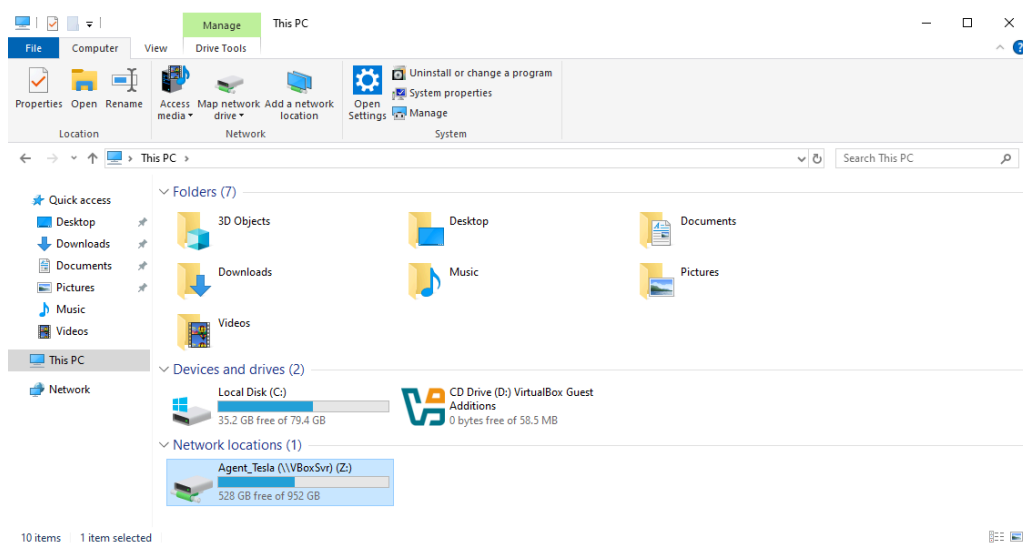


3. Choose a folder path (e.g., C:\VM_Share) and Enable:

- ☒ “Auto-mount”
- ☒ “Make Permanent”
- ☐

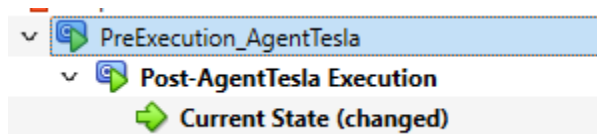


4. Inside the VM, the shared folder will appear as a drive (usually under \\VBOXSVR\ or Z:)



5. You can now drop password-protected ZIPs from your host machine into this folder for extraction within the VM

3.2. Snapshot Strategy (Pre & Post Execution)



Snapshots serve as restore points that let you roll back your virtual machine to a previous clean or known-good state. This is essential for safely analyzing malware like Agent Tesla without needing to rebuild your environment every time.

● Pre-Execution Snapshot

This snapshot is taken **after the VM is fully configured** (tools installed, shared folder working, networking set to host-only) but **before executing the malware**.

Purpose:

- Ensures a clean, repeatable baseline
- Lets you re-run or debug the malware execution without having to reinstall or reconfigure anything
- Provides a forensic-safe point of reference for comparing pre- and post-infection states

● Post-Execution Snapshot

This snapshot is taken **immediately after the malware has been executed** and before you begin exporting logs or performing forensic investigation.

Purpose:

- Captures the system's state after infection
- Prevents accidental re-execution
- Provides a frozen point in time for log review, memory analysis, or testing detection tools

Note: You'll be told exactly when to take this post-execution snapshot in the next lesson:

4. Malware Execution & Log Generation.


3.3. Obtaining Agent Tesla Sample (Generic Guidance)

Agent Tesla samples can be obtained from trusted malware research repositories like:

- <https://bazaar.abuse.ch/> (MalwareBazaar)

Search using:

tag:agenttesla

 Always download **password-protected ZIPs** only, and **extract them inside the VM** (password: infected). Never extract or run samples on your host machine.

3.4. Splunk Installation (Local)

Splunk will be used to analyze logs exported from the VM. Install it on your **host machine** or on a separate VM:

- Download from: https://www.splunk.com/en_us/download.html
- Choose the **Free Enterprise Trial**
- Allocate at least **8 GB RAM** for smooth operation
- No persistent internet access required after installation

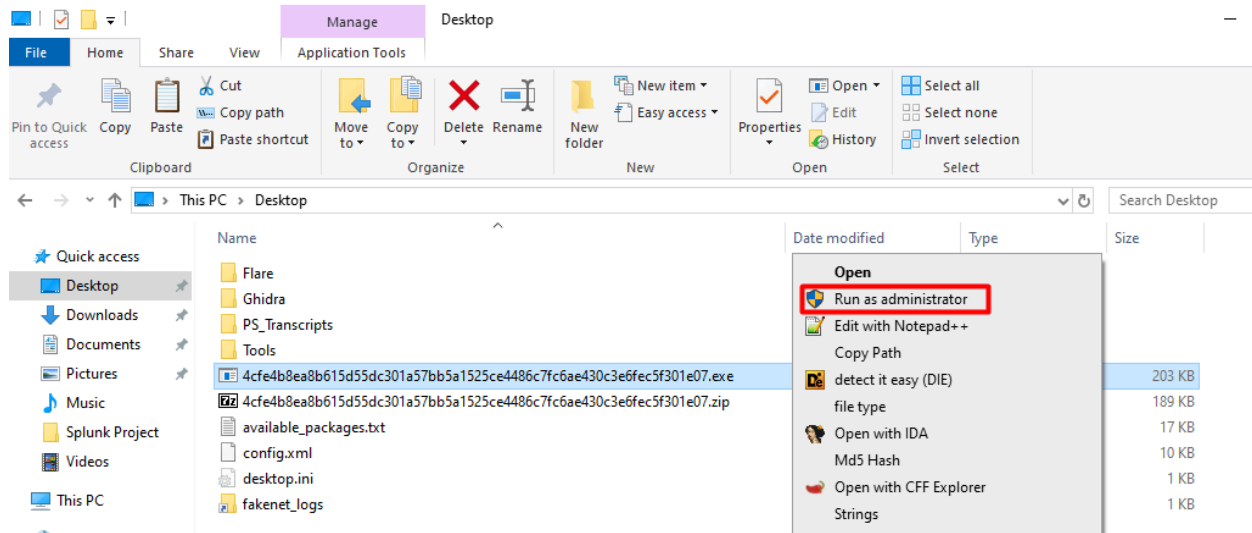
4. Malware Execution & Log Generation

4.1. Executing Agent Tesla in the VM

With your pre-execution snapshot in place and the malware sample safely extracted inside your VM, you're ready to execute the sample.

Steps:

1. Navigate to the extracted .exe file (from the shared folder or desktop).



2. Right-click → **Run as administrator**.
3. The malware may appear to do nothing — this is expected. Agent Tesla typically runs silently in the background, so wait for a few minutes.

💡 *Do not interact with the file multiple times. One execution is enough to trigger process events and logging.*

4.2. Verifying Post-Execution Activity

After executing the sample, verify that something happened by checking for signs of activity:

- Open **Event Viewer** and go to:
 - Windows Logs > Security
 - Windows Logs > Application

- Look for new entries with recent timestamps, especially under **Event ID 4688** (process creation).

This confirms that logs were generated and the malware executed successfully.

4.3. Snapshot Management for Safe Rollback

Now that the malware has been executed and activity has been confirmed, it's time to take the **Post-Execution Snapshot**.


Why Now?

- This snapshot preserves the state of the system immediately after compromise.
- It allows for:
 - Consistent forensic analysis
 - Memory analysis
 - Comparing system behavior over time
 - Safe rollback if anything breaks during log collection

Quick Steps to Take a Snapshot in VirtualBox

1. Open **VirtualBox** and make sure your **Windows 10 VM is running**.
2. In the left panel, **select your running VM**.
3. On the top right, click the **“Take”** button (camera icon).
4. (Optional) Add a short description like "After Agent Tesla execution".

Reminder:

 *This is the “post-execution snapshot” mentioned in [3.2 Snapshot Strategy].*

Snapshot Name Tip: Use something like

PostExecute_AgentTesla_YYYYMMDD

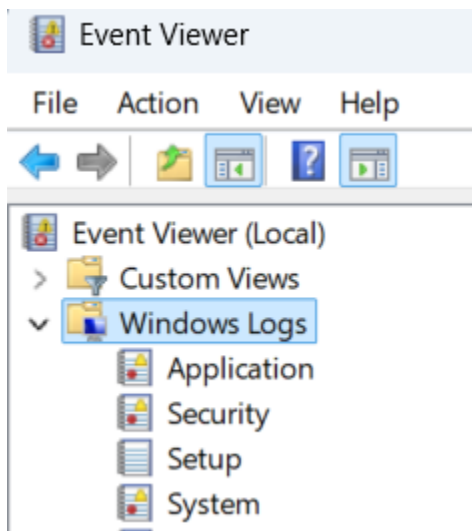
5. Log Collection

5.1. Exporting Windows Event Logs (.EVTX)

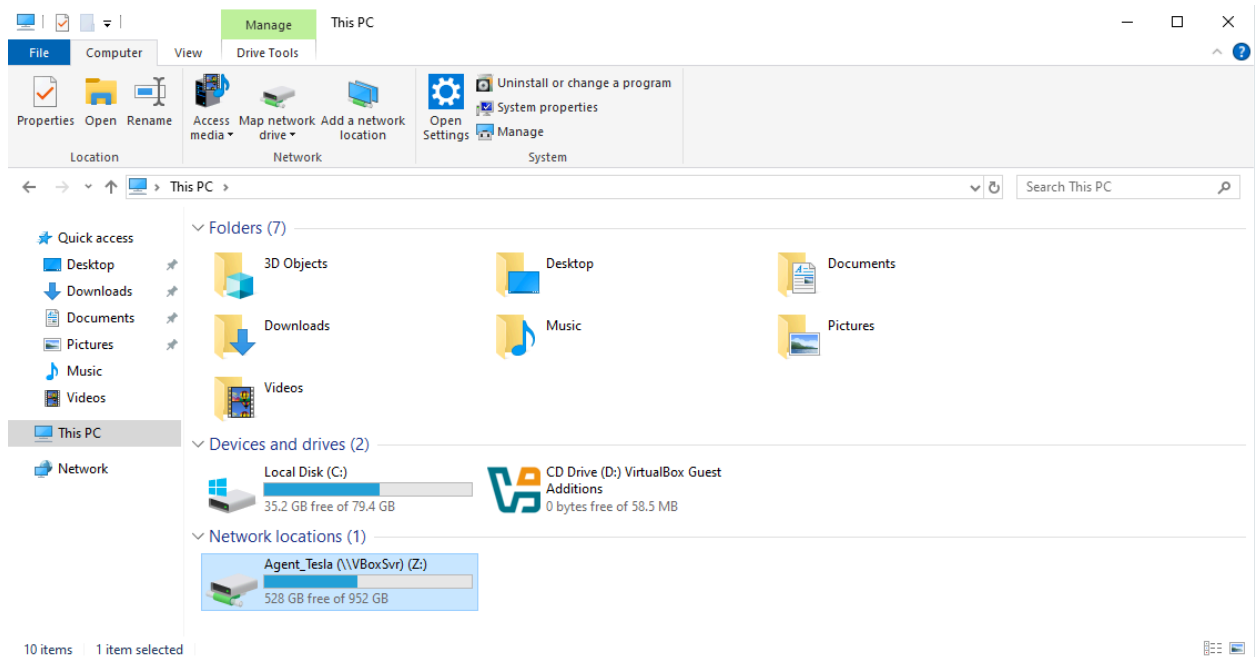
To analyze post-execution behavior in Splunk, it's crucial to extract relevant log files from the Windows 10 virtual machine. These logs are in .evtx format and contain rich details about process execution, system behavior, and application events.

Steps:

1. **Boot into the post-execution snapshot** of your Windows 10 VM (the one after Agent Tesla has been executed).
2. Open **Event Viewer** by typing eventvwr.msc in the Start Menu or Run dialog.
3. In the left-hand navigation pane, expand **Windows Logs**.




4. For each of the following logs — *Application*, *Security*, *Setup*, and *System* — do the following:
 - Right-click the log name.
 - Select **Save All Events As...**
 - Choose .evtx as the file format.
 - Name each file clearly (e.g., application.evtx, security.evtx, etc.).
 - Navigate to and **save directly to the shared folder:**
Agent_Tesla (\\VBoxSvr) (Z:)



Once saved inside the VM, the logs will be accessible from your **local machine** via the shared folder path (Z: drive). Here's how:

5. On your **local machine**, open File Explorer.
6. Navigate to the shared folder — wherever you mapped it locally (e.g., Z:\, D:\Agent_Tesla, or another custom path).
7. **Copy the .evtx files** to a convenient location on your local system for later use in Splunk.

 **Tip:** Always double-check that you are in the *post-execution snapshot*, or else the logs may not reflect the malware behavior.


5.2. Types of Logs: Application, Security, Setup, System

Each of these log types serves a unique purpose in the investigation:

- **Application.evtx:**
Captures errors and events from installed software. Sometimes malware might trigger logs related to application crashes or DLL loads.
- **Security.evtx:**
One of the most valuable logs for detecting unauthorized activity. Contains authentication

attempts, privilege escalations, and especially **Event ID 4688**, which logs process creations — key for tracking the malware's execution chain.

- **Setup.evtx:**
Contains system-level setup events, such as changes to roles or features. Less commonly used in malware analysis but included for completeness.
- **System.evtx:**
Logs Windows system events like driver failures, shutdowns, or services starting/stopping — useful for understanding changes triggered by malware at the system level.

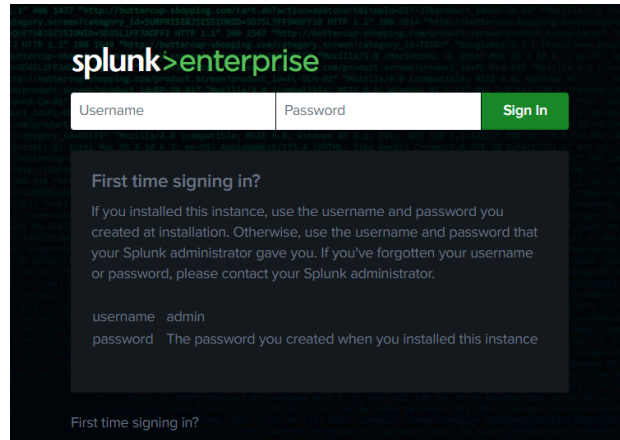
 **Best Practice:** Always collect these four log types when investigating endpoint activity. Together, they offer a complete picture of what happened before, during, and after the suspected compromise.

6. Splunk Configuration & Data Ingestion

6.1. Launching Splunk

Steps:

1. Launch **Splunk Web** (typically at <http://localhost:8000>).
2. Log in with your credentials.

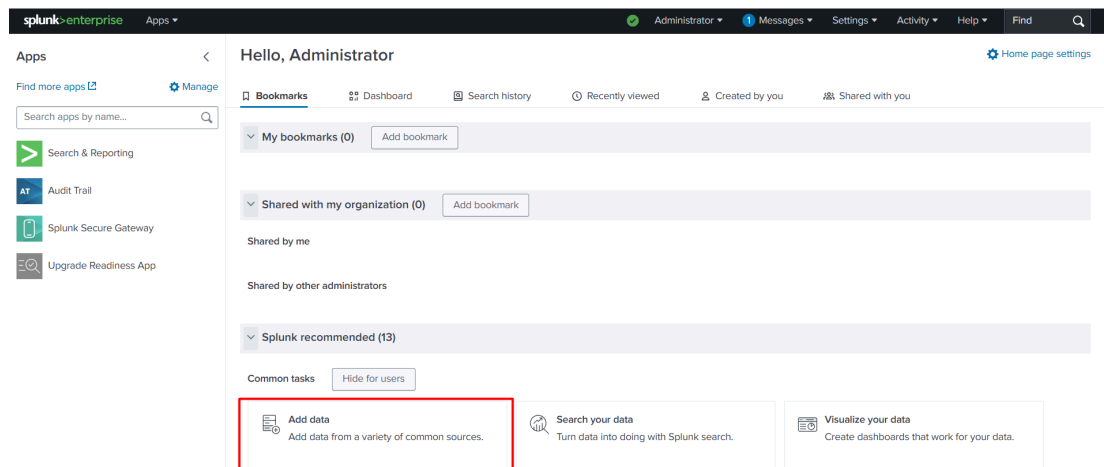


6.2. Uploading EVTX Files to Splunk

Upload the .evtx files collected from your VM into Splunk.

Steps:

1. In the homepage, click **Add Data**.



2. Choose **Upload** and select the .evtx files (e.g., application.evtx, security.evtx).

splunk-enterprise

Apps

Administrator

Messages

Settings

Activity

Help

Find

Add Data

Select Source

Set Source Type

Input Settings

Review

Done

< Back

Next >

Select Source

Choose a file to upload to the Splunk platform, either by browsing your computer or by dropping a file into the target box below. [Learn More](#)

Selected File: **Application.evtx**

Select File

Drop your data file here

The maximum file upload size is 500 Mb

3. Click **Next** when prompted to preview data.

4. At **Set Source Type**, do the following:

[illegible]

- Click **Select** next to preprocess-winevt (or wineventlog if available).
- This helps Splunk understand the structure of Windows event logs.

5. On the **Input Settings** screen:

- Set host as Constant value

Add Data

Select Source Set Source Type **Input Settings** Review Done

< Back Review >

Host

When the Splunk platform indexes data, each event receives a "host" value. The host value should be the name of the machine from which the event originates. The type of input you choose determines the available configuration options. [Learn More](#)

☒ Constant value
☐ Regular expression on path
☐ Segment in path

Host field value: DESKTOP-HLULV7B

Index

The Splunk platform stores incoming data as events in the selected index. Consider using a "sandbox" index as a destination if you have problems determining a source type for your data. A sandbox index lets you troubleshoot your configuration without impacting production indexes. You can always change this setting later. [Learn More](#)

Index: agent_tesla_lab Create a new index

- Click on **Create a new index** and then just give it the name of the lab and then click **Save**

New Index

General Settings

Index Name: agent_tesla_lab
Set index name (e.g., INDEX_NAME). Search using index=INDEX_NAME.

Index Data Type: Events Metrics
The type of data to store (event-based or metrics).

Home Path: optional
Hot/warm db path. Leave blank for default (\$SPLUNK_DB/INDEX_NAME/db).

Cold Path: optional
Cold db path. Leave blank for default (\$SPLUNK_DB/INDEX_NAME/colddb).

Thawed Path: optional
Thawed/resurrected db path. Leave blank for default (\$SPLUNK_DB/INDEX_NAME/thaweddb).

Data Integrity Check: Enable Disable
Enable this if you want Splunk to compute hashes on every slice of your data for the purpose of data integrity.

Save Cancel

6. Click **Review** then **Submit**.

The screenshot shows the 'Review' step of a five-step workflow. The steps are: Select Source, Set Source Type, Input Settings, Review, and Done. The 'Review' step is currently active, indicated by a green dot. Below the progress bar, the following information is displayed:

| | |
|-------------------|-------------------|
| Input Type | Uploaded File |
| File Name | Application.evtx |
| Source Type | preprocess-winevt |
| Host | DESKTOP-HLULV7B |
| Index | agent_tesla_lab |

At the top right, there are two buttons: '< Back' and 'Submit >'.

Repeat this process for each .evtx file, After reaching the **"Done"** screen (as shown in your screenshot), clicking **"Add More Data"** will take users back into the **data upload workflow**, allowing them to:

The screenshot shows the 'Done' step of the workflow. The progress bar now has a green checkmark at the 'Done' step. The main content area displays a success message: 'File has been uploaded successfully.' Below this, there is a link to 'Configure your inputs by going to Settings > Data Inputs'. A 'Start Searching' button is prominently displayed. Below it, there are five more buttons, each with a description and a link to learn more:

- Extract Fields**: Create search-time field extractions. [Learn more about fields.](#)
- Add More Data**: Add more data inputs now or see [examples and tutorials.](#)
- Download Apps**: Apps help you do more with your data. [Learn more.](#)
- Build Dashboards**: Visualize your searches. [Learn more.](#)

At the top right, there are two buttons: '< Back' and 'Next >'.

- Select additional .evtx or other log files
- Set a new or existing source type
- Use the same or a different index (like agent_tesla_lab)
- Upload and ingest those logs seamlessly

⚠ Note: If wineventlog isn't available, preprocess-winevt should work. In some environments, one might yield better parsing results than the other.

6.3. Parsing and Verifying Log Data

Once all EVTX logs are uploaded into Splunk under your custom index, it's time to verify that the data has been successfully ingested and is searchable.

Steps to Verify

1. Run this to check if your index is returning any results at all:

The screenshot shows the Splunk Enterprise web interface. At the top, there's a navigation bar with 'splunk>enterprise' and various menu items like 'Apps', 'Administrator', 'Messages', 'Settings', 'Activity', 'Help', and 'Find'. Below this is a 'Search & Reporting' section with a 'New Search' button. The search bar contains the query 'index=agent_tesla_lab'. Below the search bar, it shows '6,494 events (before 6/24/25 4:59:25.000 PM)' and 'No Event Sampling'. The interface includes a timeline visualization and a table of event details. The table has columns for 'Time' and 'Event'. The first event is from 6/24/25 11:10:55.000 AM, with LogName=Application, EventCode=16384, EventType=4, and ComputerName=DESKTOP-D6KMFNCN. The second event is from 6/24/25 11:10:29.000 AM, with LogName=Application, EventCode=1001, EventType=4, and ComputerName=DESKTOP-D6KMFNCN.

index=agent_tesla_lab

If you see events populating below the search bar, your ingestion was successful. If not, revisit your upload process or check the time range filter (try setting it to “All Time”).

2. This confirms which types of logs (e.g., WinEventLog:Security, WinEventLog:Application) were ingested and categorized by Splunk:

index=agent_tesla_lab | stats count by sourcetype

3. Event ID 4688 logs process creation. If Agent Tesla executed successfully and the snapshot was post-execution, you should see entries like GASHGHAHS.exe or other suspicious processes:

index=agent_tesla_lab EventCode=4688

-
4. This displays the raw log entries along with timestamps. It's especially useful if structured fields like ParentImage, Image, or CommandLine aren't parsed:

```
index=agent_tesla_lab | table _time, _raw
```


Observation Tip

Some fields might not automatically parse depending on the source type or how the logs were formatted. If structured columns show up blank, rely on the `_raw` output to manually spot:

- Suspicious .exe launches
- Unexpected directory paths (e.g., AppData\Roaming)
- Patterns like script abuse or encoded commands

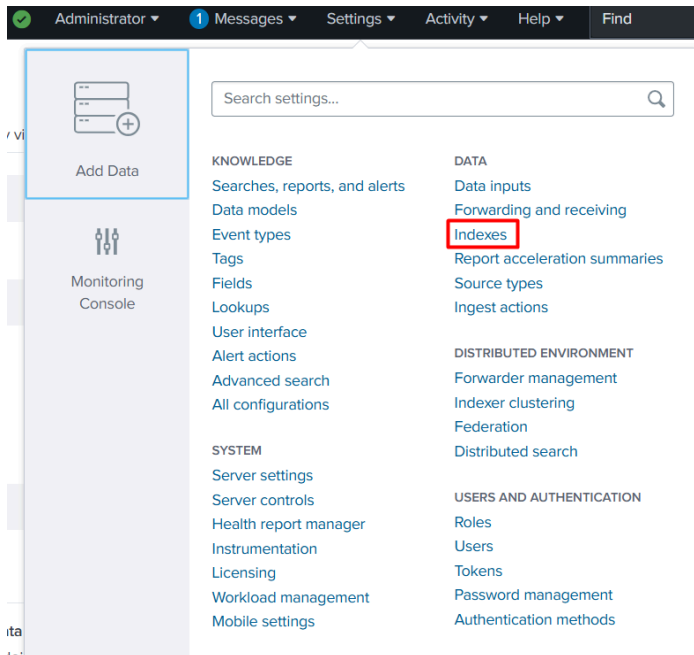
6.4. If You Messed Up the Index – Don't Worry, Do This

Sometimes you might assign logs to the wrong index, use the wrong source type, or simply want to start over with a clean slate. Don't worry — here's how to delete your existing index and reconfigure things from scratch.

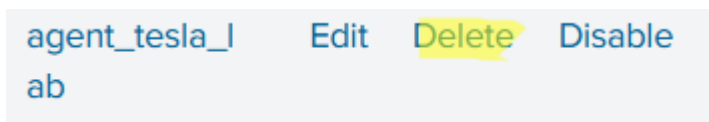
 **Important Note:** Deleting an index will permanently remove any ingested data associated with it. Only do this if you're certain you want to reset the process.

Steps to Reset Your Index in Splunk:

1. Go to the Splunk main menu (top left corner).
2. Click "Settings" → "Indexes".




3. Find the index you want to delete (e.g., agent_tesla_lab).
4. Click **"Delete"** next to the index name.



5. Confirm deletion when prompted.

Once deleted, you can re-upload your EVTX logs and reassign them to a new or re-created custom index — this time using the correct source type or other settings.

 **Tip:** It's often a good idea to create a new index with a slightly different name (e.g., agent_tesla_lab_v2) just to avoid confusion and keep track of revisions.

7. Detection & Investigation

7.1. Key EventCodes Queried (e.g., 4688)

The primary Event ID used in this project was **4688**, which logs process creation events. It is especially valuable when investigating malware like Agent Tesla, which spawns new processes during execution.

Additional EventCodes of interest:

- **4624** – Successful Logon
- **4656** – Handle to an Object Requested (useful for tracking file or registry access)
- **4670** – Permissions on an object were changed
- **7045** – A new service was installed
- **4104** – PowerShell script block logging (useful for script-based payloads)
- **5156** – Windows Filtering Platform has permitted a connection



Note: Not all of these events were present or populated in this specific analysis. However, they represent crucial log entries often investigated during real-world incidents and should be kept in mind for future projects or more advanced log sets.

7.2. Keyword & Pattern-Based Searches

Due to limitations in how Splunk parsed the EVTX files, many structured fields such as ParentImage or CommandLine were unavailable. As a result, keyword-based searches became the primary investigative technique.

Some useful patterns included:

- `index=agent_tesla_lab "*.exe"` — to identify executables
- `index=agent_tesla_lab "**AppData**"` — to locate processes or paths executing from AppData
- `index=agent_tesla_lab "**temp**"` — for potentially suspicious temp folder execution

These helped highlight anomalies such as:

- EXEs running from %AppData% or other non-standard directories

- Obfuscated or generic filenames that did not align with typical software installations


7.3. Working with Raw Logs

Due to Splunk's incomplete parsing of .evtx logs, many structured fields (such as NewProcessName, ParentImage, and CommandLine) were not available by default. To overcome this, manual field extraction using rex commands became necessary.

One particularly effective query:

```
index=agent_tesla_lab EventCode=4688
| rex field=_raw "NewProcessName:\s+(?<NewProcessName>[^\r\n]+)"
| rex field=_raw "ParentImage:\s+(?<ParentImage>[^\r\n]+)"
| rex field=_raw "CommandLine:\s+(?<CommandLine>[^\r\n]+)"
| table _time, NewProcessName, ParentImage, CommandLine
```

This allowed visibility into key process relationships and command-line arguments, even when Splunk failed to parse them automatically.

 **Takeaway:** This is a powerful example of how raw log forensics and manual field extraction are vital skills in incident response. When SIEM tools don't do the heavy lifting, analysts must rely on regex and intuition to reconstruct the attack timeline.

7.4. Sample Query Results

Some queries that helped guide the investigation:

```
index=agent_tesla_lab EventCode=4688
index=agent_tesla_lab "*.exe"
index=agent_tesla_lab "**AppData*"
index=agent_tesla_lab | stats count by sourcetype
index=agent_tesla_lab | table _time, _raw
```

And of course, the detailed rex-based extraction query (from Section 7.3) proved invaluable in surfacing behavioral indicators.

This combination of default and manual queries helped build a narrative around what the malware was doing — despite its efforts to stay hidden.

7.5. Indicators of Compromise (IOCs)

Although the string "Agent Tesla" never explicitly appeared in the logs, the following IOCs were identified through behavioral patterns and process traces:

- **Suspicious executable locations** (e.g., executables running from AppData)
- **Unusual process spawns** (e.g., processes launched without an expected parent process)
- **Obfuscated or generic EXE names** not matching system or user-installed software
- **Repeated process creation attempts** around the time Agent Tesla was executed


This absence of a clear signature forced a behavior-based approach—something that’s extremely common in real-world SOC environments. In other words, **malware doesn’t politely announce itself**. This made the exercise more realistic and emphasized why defenders must rely on context, timing, and anomaly detection.

8. Response Recommendations

8.1. Suggested Containment Actions

Based on the activity observed from the Agent Tesla execution, the following actions are recommended to contain and mitigate the threat:

- **Immediate Host Isolation:** Remove the infected host from the network to prevent further data exfiltration or lateral movement.
- **Kill Malicious Processes:** Terminate processes associated with suspicious executables (e.g., *.exe running from unusual directories like %AppData% or %Temp%).
- **Remove Persistence Mechanisms:** Investigate and delete any autorun registry keys or scheduled tasks that may ensure Agent Tesla runs at startup.
- **Credential Reset:** Promptly reset passwords for accounts that may have been compromised during data-stealing activity.
- **Conduct Full Antivirus & EDR Scans:** Perform a deep scan using updated threat signatures to ensure complete eradication.
- **Log Retention:** Retain relevant log data for forensic analysis and possible legal review.


 *Although this was done in a lab, these recommendations reflect real-world blue team procedures following initial detection of commodity malware.*

8.2. Lessons Learned

This project highlighted a number of key takeaways relevant to real-world incident response:

- **Malware isn't always obvious:** Agent Tesla did not name itself in logs. Detection relied on suspicious behavior like .exe launches from non-standard directories.
- **Snapshots matter:** Using a pre-execution snapshot caused misleading results and wasted analysis time. Always validate the environment state before extracting logs.
- **Log data is imperfect:** Field extractions were inconsistent, forcing reliance on raw log inspection and keyword searches — a common reality in incident response.

- **Splunk is powerful, but manual analysis is crucial:** You don't always get clean, structured results — human interpretation of anomalies is just as important as query syntax.

 *This lab reinforced the importance of patience, attention to detail, and flexible thinking when responding to real-world security events.*

9. Challenges & Workarounds (Recap)

While these challenges were touched on earlier in the report, the following recap highlights key roadblocks faced during the project and how they were overcome. These notes serve as a quick guide for others who may follow a similar process.

9.1. Missing Field Values

Splunk didn't extract structured fields like ParentImage or CommandLine from EVTX logs consistently. Workaround: fallback to raw log searches and keyword-based detection.

9.2. Snapshot Confusion (Pre vs Post Execution)

Initial analysis used a snapshot taken **before** executing the malware, which led to a lack of expected log entries. Workaround: reran Agent Tesla from the correct snapshot and re-exported the logs.

9.3. Search Query Limitations

Certain queries using specific fields returned no results due to inconsistent parsing. Workaround: broadened searches using index=... `"*tesla*" OR "*.exe"` style queries and raw log inspection.

9.4. Manual Pattern Matching

Agent Tesla's name didn't appear explicitly in logs, simulating a real-world challenge where indicators are subtle or obfuscated. Solution: relied on suspicious file paths, timestamps, and process activity to infer compromise.

10. Conclusion

10.1. Summary of Outcomes

This project offered hands-on experience with Splunk as a detection and investigation tool, simulating a real-world incident response scenario involving Agent Tesla malware. While the malware name itself didn't explicitly appear in the logs, behavioral indicators such as suspicious .exe launches from user directories and unusual process activity provided enough insight to detect compromise. The project also reinforced the importance of snapshots, structured logging, and iterative query building in threat hunting.

10.2. Next Steps for Improvement

- Incorporate **Sysmon** for more granular visibility (e.g., Image, CommandLine, ParentImage fields).
- Automate data ingestion and alert creation using custom Splunk correlation rules.
- Reproduce this workflow using **Elastic Stack** to diversify tool familiarity.
- Perform multi-host analysis to simulate lateral movement and privilege escalation.

11. Appendix

11.1. Screenshots

Relevant screenshots throughout the document showcase:

The screenshot shows the Splunk Enterprise interface. At the top, there's a navigation bar with 'splunk>enterprise' and various menu items like 'Apps', 'Administrator', 'Messages', 'Settings', 'Activity', 'Help', and 'Find'. Below this is a 'Search & Reporting' section with a 'New Search' header. The search query is 'index=agent_tesla_lab EventCode=4688 | table _time, ParentImage, NewProcessName, CommandLine | sort _time'. The results show 367 events. The table has columns for '_time', 'ParentImage', 'NewProcessName', and 'CommandLine'. The first few rows show timestamps from 2025-01-28 11:18:46 to 2025-01-28 11:19:00.

| _time | ParentImage | NewProcessName | CommandLine |
|---------------------|-------------|----------------|-------------|
| 2025-01-28 11:18:46 | | | |
| 2025-01-28 11:18:46 | | | |
| 2025-01-28 11:18:50 | | | |
| 2025-01-28 11:18:59 | | | |
| 2025-01-28 11:19:00 | | | |
| 2025-01-28 11:19:00 | | | |
| 2025-01-28 11:19:00 | | | |
| 2025-01-28 11:19:00 | | | |
| 2025-01-28 11:19:00 | | | |
| 2025-01-28 11:19:00 | | | |

The screenshot shows the Splunk Enterprise interface. At the top, there's a navigation bar with 'splunk>enterprise' and various menu items like 'Apps', 'Administrator', 'Messages', 'Settings', 'Activity', 'Help', and 'Find'. Below this is a 'Search & Reporting' section with a 'New Search' header. The search query is 'index=agent_tesla_lab | stats count by EventCode | sort -count'. The results show 22,958 events. The table has columns for 'EventCode' and 'count'. The first few rows show EventCode values like 5379, 1001, 4624, 4672, 4799, 4688, 6, 16, 16394, 16384, 1003, 1, 4798, 24, 1004, 1034, 10000, 4616, 4648, 10001.

| EventCode | count |
|-----------|-------|
| 5379 | 4782 |
| 1001 | 4769 |
| 4624 | 3848 |
| 4672 | 2921 |
| 4799 | 511 |
| 4688 | 367 |
| 6 | 323 |
| 16 | 293 |
| 16394 | 249 |
| 16384 | 241 |
| 1003 | 230 |
| 1 | 219 |
| 4798 | 190 |
| 24 | 170 |
| 1004 | 161 |
| 1034 | 154 |
| 10000 | 139 |
| 4616 | 138 |
| 4648 | 135 |
| 10001 | 128 |

New SearchSave AsCreate Table ViewClose

```

index=agent_tesla_lab EventCode=4688
| rex field=_raw "New Process Name:\s+(?<NewProcessName>[^\r\n]+)"
| rex field=_raw "Creator Process Name:\s+(?<ParentImage>[^\r\n]+)"
| rex field=_raw "Process Command Line:\s+(?<CommandLine>[^\r\n]+)"
| table _time, ParentImage, NewProcessName, CommandLine
| sort _time

```

All time

✓ 367 events (before 6/25/25 5:54:13.000 PM) No Event Sampling

Job

Smart Mode

Events Patterns **Statistics (367)** Visualization

Show: 20 Per Page

Format

Preview: On

< Prev 1 2 3 4 5 6 7 8 ... Next >

| _time | ParentImage | NewProcessName | CommandLine |
|---------------------|---------------------------------|---------------------------------|---|
| 2025-01-28 11:18:46 | Process Command Line: | C:\Windows\System32\smss.exe | Token Elevation Type indicates the type of token that was assigned to the new process in accordance with User Account Control policy. |
| 2025-01-28 11:18:46 | Process Command Line: | Registry | Token Elevation Type indicates the type of token that was assigned to the new process in accordance with User Account Control policy. |
| 2025-01-28 11:18:50 | C:\Windows\System32\smss.exe | C:\Windows\System32\autochk.exe | Token Elevation Type indicates the type of token that was assigned to the new process in accordance with User Account Control policy. |
| 2025-01-28 11:18:59 | C:\Windows\System32\smss.exe | C:\Windows\System32\smss.exe | Token Elevation Type indicates the type of token that was assigned to the new process in accordance with User Account Control policy. |
| 2025-01-28 11:19:00 | C:\Windows\System32\wininit.exe | C:\Windows\System32\lsass.exe | Token Elevation Type indicates the type of token that was assigned to the new process in accordance with User Account Control policy. |

11.2. Queries Used

A sample of working queries used during this project:

Basic Validation & Log Inspection

```
index=agent_tesla_lab
```

```
index=agent_tesla_lab | stats count by sourcetype
```

```
index=agent_tesla_lab | table _time, host, source, sourcetype, _raw
```

Behavior-Based Keyword Searches

```
index=agent_tesla_lab "*.exe"
```

```
index=agent_tesla_lab "**AppData**"
```

```
index=agent_tesla_lab "*.vbs" OR "*.bat" OR "*.ps1"
```

Event ID Specific Queries

index=agent_tesla_lab EventCode=4688


index=agent_tesla_lab EventCode=4624

index=agent_tesla_lab EventCode=4656

(Useful for: Logon activity, registry/file access, and process creation)

Manual Field Extraction via Raw Log Parsing

```
index=agent_tesla_lab EventCode=4688  
| rex field=_raw "NewProcessName:\s+(?<NewProcessName>[^\r\n]+)"  
| rex field=_raw "ParentImage:\s+(?<ParentImage>[^\r\n]+)"  
| rex field=_raw "CommandLine:\s+(?<CommandLine>[^\r\n]+)"  
| table _time, NewProcessName, ParentImage, CommandLine
```

 *Note: If queries using structured fields like ParentImage or NewProcessName return blank results, rely on raw log inspection for context.*