

Project: The Great Admin Heist – CTF Forensic Analysis (Cyber Range)

Target System: Acme Corp Endpoint

Device Name: anthony-001

Threat Actor: The Phantom Hackers (Simulated APT Group)

Analyst: Venkata Bharath Devulapalli

Platform Used: Microsoft Defender for Endpoint (MDE)

Date Completed: 05/19/2025

Capture The Flag (CTF) Project Report: Malware-Based Incident on Acme Corp Endpoint (anthony-001)

1. Project Overview

Project Title: Multi-Stage Malware Detection and Attribution: The Phantom Hackers Case

Objective: To investigate a simulated advanced persistent threat (APT) attack executed by a fictional group known as "The Phantom Hackers" against Acme Corp. The focus of this threat hunt is to identify the origin, behavior, and persistence mechanisms of the fake antivirus software (BitSentinelCore.exe) planted on the host device, anthony-001.

2. Scenario Description:

An insider-themed narrative reveals that an eccentric IT admin unknowingly became the target of a stealthy multi-stage attack. The attackers used deception, fileless malware, and persistence techniques to silently infiltrate Acme Corp's systems. As part of a cybersecurity exercise, the task was to detect and document each stage of the attack using telemetry data from Microsoft Defender for Endpoint (MDE) and investigate how attackers achieved stealthy long-term access.

Flag 1 – Suspicious Antivirus Discovery

During the initial phase of the investigation, a suspicious executable masquerading as antivirus software was observed on the system. Its filename resembled a legitimate tool, which helped it blend into the environment. Analysts had to identify this root artifact to begin unraveling the attack chain.

Flag 2 – Malicious File Dropped

After recognizing the fake antivirus, attention turned to how it got there. The team investigated the method of delivery and discovered that the malware had not been downloaded from the internet — it had been compiled locally using a legitimate Microsoft tool, showing how trusted components can be abused for malicious purposes.

Flag 3 – Execution Confirmation

Next, it was important to verify whether the fake antivirus was ever actually run. By analyzing execution logs, it was confirmed that a user — presumably Bubba — had manually launched the malicious executable, triggering the beginning of its operations on the host.

Flag 4 – Keylogger Artifact

As analysts dug deeper, they uncovered evidence that the malware dropped a hidden file associated with keylogging activity. This file was placed in a well-known Startup location, likely to log credentials or keystrokes over time. Its name was crafted to look benign and avoid detection.

Flag 5 – Registry-Based Persistence

The malware didn't stop at a single execution. It ensured longevity by modifying the Windows Registry, embedding itself in startup keys to relaunch on every reboot or user login. This registry entry became a critical indicator of compromise and long-term persistence.

Flag 6 – Scheduled Task Persistence

To further solidify its presence, the malware created a scheduled task that executed daily. Its name was crafted to mimic health telemetry, hinting at legitimacy, while silently ensuring that malicious payloads would continue executing without user awareness.

Flag 7 – Process Spawn Chain

The execution path revealed a deeper level of sophistication. Analysts traced the full parent-child-grandchild process chain showing how the malware leveraged trusted system binaries like cmd.exe and schtasks.exe to achieve its goals, obscuring its behavior within legitimate OS processes.

Flag 8 – Root Cause Timeline

Finally, by correlating all logs and timestamps — from file creation and registry changes to scheduled task execution — the team pinpointed the earliest confirmed action that triggered the entire attack sequence. This became the forensic anchor that validated the timeline and confirmed the source of compromise.

3. Threat Hunt Methodology

- **Data Collection:** Logs from DeviceProcessEvents, DeviceFileEvents, and DeviceRegistryEvents were reviewed.
- **Initial Hypothesis:** Root cause was linked to a fake antivirus execution that enabled staged execution and stealthy persistence.
- **Timeline Creation:** Events were chronologically sorted to determine the order of infection and persistence.
- **KQL Analysis:** Queries were executed using Microsoft Defender for Endpoint's advanced hunting features.
- **MITRE ATT&CK Mapping:** Each observed action was mapped to ATT&CK tactics and techniques.
- **Artifact Correlation:** Each malicious outcome (e.g., persistence, keylogging, compilation) was tied back to the initiating malware.

4. Tables Used to Detect IoCs (Indicators of Compromise):

Parameter	Name	Info	Purpose
DeviceProcessEvents	Process Execution Logs	Microsoft Docs – DeviceProcessEvents	Used to trace malware execution, parent-child chains, scheduled task creation, and persistence.
DeviceFileEvents	File Creation Logs	Microsoft Docs – DeviceFileEvents	Used to detect initial malware drop (BitSentinelCore.exe), keylogger files, and .lnk artifacts.

DeviceRegistryEvents	Registry Persistence	Microsoft Docs – DeviceRegistryEvents	Used to identify malicious registry keys (e.g., Run, RunOnce) that enable startup persistence.
DeviceNetworkEvents	Network Connections	Microsoft Docs – DeviceNetworkEvents	(Optional) Used to monitor command-and-control connections if the malware had outbound behavior.
DeviceImageLoadEvents	DLL Injection Tracing	Microsoft Docs – DeviceImageLoadEvents	Helped validate stealthy DLL injections (e.g., rundll32 with PcaSvc.dll) tied to memory execution.

4.1 Tools Used

- Microsoft Defender for Endpoint (MDE)
- KQL (Kusto Query Language)

5. Investigative KQL Queries and Observations














Step 1: Detect Execution of Fake Antivirus (BitSentinelCore.exe) Flag 1 – Fake Antivirus Execution:

Objective: We needed to confirm whether the suspicious executable FakeAntiVirus.exe was actually run on the endpoint and who initiated it. This helps confirm attacker interaction or user deception.

KQL Query:

```
DeviceProcessEvents
| where Timestamp > ago(30d)
| where DeviceName == "anthony-001"
| where InitiatingProcessAccountName == '4nth0ny!'
| where FileName endswith ".exe" or ProcessCommandLine has ".exe"
| where FileName startswith 'a' or FileName startswith 'b' or FileName startswith 'c'
| project Timestamp, DeviceName, FileName, FolderPath, ProcessCommandLine,
InitiatingProcessAccountName, InitiatingProcessFileName
```

Timestamp	DeviceName	FileName	FolderPath	ProcessCommandLine	InitiatingProc	InitiatingProcessFileName
5/6/2025 19:59	anthony-001	cvtres.exe	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\cvtres.exe	cvtres.exe /NOLOGO /READONLY /MA	4nth0ny!	csc.exe
5/6/2025 20:24	anthony-001	conhost.exe	C:\Windows\System32\conhost.exe	conhost.exe 0xffffffff -ForceV1	4nth0ny!	powershell.exe
5/6/2025 20:57	anthony-001	conhost.exe	C:\Windows\System32\conhost.exe	conhost.exe 0xffffffff -ForceV1	4nth0ny!	powershell.exe
5/6/2025 21:00	anthony-001	csc.exe	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe	"csc.exe" /noconfig /fullpaths @"C:\U	4nth0ny!	powershell.exe
5/6/2025 21:00	anthony-001	cvtres.exe	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\cvtres.exe	cvtres.exe /NOLOGO /READONLY /MA	4nth0ny!	csc.exe
5/6/2025 21:02	anthony-001	BitSentinelCore.exe	C:\ProgramData\BitSentinelCore.exe	BitSentinelCore.exe	4nth0ny!	explorer.exe
5/6/2025 21:02	anthony-001	conhost.exe	C:\Windows\System32\conhost.exe	conhost.exe 0xffffffff -ForceV1	4nth0ny!	bitsentinelcore.exe
5/6/2025 21:02	anthony-001	cmd.exe	C:\Windows\System32\cmd.exe	"cmd.exe" /c schtasks /Create /SC DA	4nth0ny!	bitsentinelcore.exe
5/6/2025 21:02	anthony-001	conhost.exe	C:\Windows\System32\conhost.exe	conhost.exe 0xffffffff -ForceV1	4nth0ny!	cmd.exe
5/6/2025 21:03	anthony-001	BitSentinelCore.exe	C:\ProgramData\BitSentinelCore.exe	BitSentinelCore.exe	4nth0ny!	explorer.exe
5/6/2025 21:03	anthony-001	conhost.exe	C:\Windows\System32\conhost.exe	conhost.exe 0xffffffff -ForceV1	4nth0ny!	bitsentinelcore.exe
5/6/2025 21:03	anthony-001	cmd.exe	C:\Windows\System32\cmd.exe	"cmd.exe" /c schtasks /Create /SC DA	4nth0ny!	bitsentinelcore.exe
5/6/2025 21:03	anthony-001	conhost.exe	C:\Windows\System32\conhost.exe	conhost.exe 0xffffffff -ForceV1	4nth0ny!	cmd.exe
5/6/2025 21:03	anthony-001	BitSentinelCore.exe	C:\ProgramData\BitSentinelCore.exe	BitSentinelCore.exe	4nth0ny!	explorer.exe
5/6/2025 21:03	anthony-001	conhost.exe	C:\Windows\System32\conhost.exe	conhost.exe 0xffffffff -ForceV1	4nth0ny!	bitsentinelcore.exe
5/6/2025 21:03	anthony-001	cmd.exe	C:\Windows\System32\cmd.exe	"cmd.exe" /c schtasks /Create /SC DA	4nth0ny!	bitsentinelcore.exe
5/6/2025 21:03	anthony-001	conhost.exe	C:\Windows\System32\conhost.exe	conhost.exe 0xffffffff -ForceV1	4nth0ny!	cmd.exe

<input type="checkbox"/>	Timestamp	DeviceName	FileName	FolderPath	ProcessCommandLine	InitiatingProcessAccountNa...	InitiatingProcessFileName	
<input type="checkbox"/>	>	May 6, 2025 7:59:0...	 anthony-001	cvtres.exe	C:\Windows\Microsoft.N...	cvtres.exe /NOLOGO /R...	4nth0ny!	csc.exe
<input type="checkbox"/>	>	May 6, 2025 8:24:0...	 anthony-001	conhost.exe	C:\Windows\System32\c...	conhost.exe 0xffffffff -Fo...	4nth0ny!	powershell.exe
<input type="checkbox"/>	>	May 6, 2025 8:57:3...	 anthony-001	conhost.exe	C:\Windows\System32\c...	conhost.exe 0xffffffff -Fo...	4nth0ny!	powershell.exe
<input type="checkbox"/>	>	May 6, 2025 9:00:3...	 anthony-001	csc.exe	C:\Windows\Microsoft.N...	"csc.exe" /noconfig /full...	4nth0ny!	powershell.exe
<input type="checkbox"/>	>	May 6, 2025 9:00:3...	 anthony-001	cvtres.exe	C:\Windows\Microsoft.N...	cvtres.exe /NOLOGO /R...	4nth0ny!	csc.exe
<input type="checkbox"/>	>	May 6, 2025 9:02:1...	 anthony-001	BitSentinelCore.exe	C:\ProgramData\BitSenti...	BitSentinelCore.exe	4nth0ny!	explorer.exe
<input type="checkbox"/>	>	May 6, 2025 9:02:1...	 anthony-001	conhost.exe	C:\Windows\System32\c...	conhost.exe 0xffffffff -Fo...	4nth0ny!	bitsentinelcore.exe
<input type="checkbox"/>	>	May 6, 2025 9:02:1...	 anthony-001	cmd.exe	C:\Windows\System32\c...	"cmd.exe" /c schtasks /C...	4nth0ny!	bitsentinelcore.exe
<input type="checkbox"/>	>	May 6, 2025 9:02:1...	 anthony-001	conhost.exe	C:\Windows\System32\c...	conhost.exe 0xffffffff -Fo...	4nth0ny!	cmd.exe
<input type="checkbox"/>	>	May 6, 2025 9:03:1...	 anthony-001	BitSentinelCore.exe	C:\ProgramData\BitSenti...	BitSentinelCore.exe	4nth0ny!	explorer.exe
<input type="checkbox"/>	>	May 6, 2025 9:03:1...	 anthony-001	conhost.exe	C:\Windows\System32\c...	conhost.exe 0xffffffff -Fo...	4nth0ny!	bitsentinelcore.exe
<input type="checkbox"/>	>	May 6, 2025 9:03:1...	 anthony-001	cmd.exe	C:\Windows\System32\c...	"cmd.exe" /c schtasks /C...	4nth0ny!	bitsentinelcore.exe
<input type="checkbox"/>	>	May 6, 2025 9:03:1...	 anthony-001	conhost.exe	C:\Windows\System32\c...	conhost.exe 0xffffffff -Fo...	4nth0ny!	cmd.exe
<input type="checkbox"/>	>	May 6, 2025 9:03:2...	anthony-001	BitSentinelCore.exe	C:\ProgramData\BitSenti...	BitSentinelCore.exe	4nth0ny!	explorer.exe
<input type="checkbox"/>	>	May 6, 2025 9:03:2...	anthony-001	conhost.exe	C:\Windows\System32\c...	conhost.exe 0xffffffff -Fo...	4nth0ny!	bitsentinelcore.exe
<input type="checkbox"/>	>	May 6, 2025 9:03:2...	anthony-001	cmd.exe	C:\Windows\System32\c...	"cmd.exe" /c schtasks /C...	4nth0ny!	bitsentinelcore.exe
<input type="checkbox"/>	>	May 6, 2025 9:03:2...	anthony-001	conhost.exe	C:\Windows\System32\c...	conhost.exe 0xffffffff -Fo...	4nth0ny!	cmd.exe

Observation:

We are looking for some file name that should look like an antivirus file name. Confirmed that BitSentinelCore.exe was launched by explorer.exe, strongly indicating user interaction, simulating a user click (e.g., double-clicked), which aligns with the CTF clue that “Bubba clicked it.”

Step 2: Find How BitSentinelCore.exe Arrived on Disk (Flag 2 – Malicious File Dropped to Disk)

Objective: To determine the delivery mechanism of the malware — whether it was dropped, downloaded, or compiled — and what program introduced it into the system.

KQL Query:

DeviceFileEvents

| where Timestamp > ago(30d)

| where DeviceName == "anthony-001"

| where FileName == "BitSentinelCore.exe"

| project Timestamp, FileName, FolderPath, InitiatingProcessFileName, InitiatingProcessCommandLine, ReportId

<input type="checkbox"/>	Timestamp	FileName	FolderPath	InitiatingProcessFileName	InitiatingProcessCommandL...	ReportId
<input type="checkbox"/>	May 6, 2025 9:00:3...	BitSentinelCore.exe	C:\ProgramData\BitSenti...	csc.exe	"csc.exe" /noconfig /full...	2663
	Timestamp	May 6, 2025 9:00:36 PM				
	FileName	BitSentinelCore.exe				
	FolderPath	C:\ProgramData\BitSentinelCore.exe				
	InitiatingProcessFileName	csc.exe				
	InitiatingProcessComma...	"csc.exe" /noconfig /fullpaths @"C:\Users\4nth0ny!\AppData\Local\Temp\c5gy0jzg\c5gy0jzg.cmdline"				
	ReportId	2663				

Observation:

The file was written by csc.exe, proving that it was compiled on the machine from source using the .NET compiler, a stealth technique often used in fileless or staged malware.

Step 3: *Execution of the Program* (Flag 3- Execution of Malware)

Objective: Confirm that the dropped file was executed. Verifying execution ties dropped binaries to active threat behavior.

KQL Query:

[DeviceProcessEvents](#)

```
| where DeviceName == "anthony-001"
| where Timestamp > ago(30d)
| where FileName == "BitSentinelCore.exe"
| project Timestamp, FileName, ProcessCommandLine, InitiatingProcessFileName,
InitiatingProcessCommandLine, AccountName
```

<input type="checkbox"/>	Timestamp	FileName	ProcessCommandLine	InitiatingProcessFileName	InitiatingProcessCommandL...	AccountName
<input type="checkbox"/>	> May 6, 2025 9:02:14 PM	BitSentinelCore.exe	BitSentinelCore.exe	explorer.exe	Explorer.EXE	4nth0ny!
<input type="checkbox"/>	> May 6, 2025 9:03:16 PM	BitSentinelCore.exe	BitSentinelCore.exe	explorer.exe	Explorer.EXE	4nth0ny!
<input type="checkbox"/>	> May 6, 2025 9:03:20 PM	BitSentinelCore.exe	BitSentinelCore.exe	explorer.exe	Explorer.EXE	4nth0ny!

Observation: Shows user 4nth0ny! manually executed the compiled binary — a key indicator of compromise.

Step 4: *Keylogger Artifact Written* (Flag 4 – Keylogger Artifact Dropped)

Objective: We suspected a keylogger was part of the attack. This query searches for files typically associated with logging inputs or persistence, including .lnk, .log, or suspicious .exe in startup paths. We have to detect any file linked to keylogging behavior. Attackers often drop .lnk, .log, or .txt files for stealth keylogging.

KQL Query (lnk and related):

```
DeviceFileEvents
| where DeviceName == "anthony-001"
| where Timestamp > ago(30d)
| where FileName has_any("key", "log", "input", "lnk")
| where InitiatingProcessFileName contains "explorer.exe"
| project Timestamp, FileName, FolderPath, InitiatingProcessFileName, ActionType
```

Additional Query (extensions-based):

```
DeviceFileEvents
| where DeviceName == "anthony-001"
| where Timestamp > ago(30d)
| where FileName endswith ".key" or FileName endswith ".log" or FileName endswith ".txt"
| project Timestamp, FileName, FolderPath, InitiatingProcessFileName, ActionType
```

<input type="checkbox"/>	Timestamp	FileName	FolderPath	InitiatingProcessFileName	ActionType
<input type="checkbox"/>	> May 6, 2025 8:30:30 PM	Document3.lnk	C:\Users\4nth0ny!\AppD...	explorer.exe	FileCreated
<input type="checkbox"/>	> May 6, 2025 8:30:30 PM	FinanceStatements.lnk	C:\Users\4nth0ny!\AppD...	explorer.exe	FileCreated
<input type="checkbox"/>	> May 6, 2025 9:06:51 PM	systemreport.lnk	C:\Users\4nth0ny!\AppD...	explorer.exe	FileCreated
<input type="checkbox"/>	> May 6, 2025 9:06:51 PM	ThreatMetrics.lnk	C:\Users\4nth0ny!\AppD...	explorer.exe	FileCreated

Observation: Detected a file named systemreport.lnk created in the Startup folder, and dropped by explorer.exe. This aligns with the clue about an efficient process and news-sounding names. The file was likely tied to a keylogger like AutoHotkeyU32.exe.

Step 5: Investigate Registry-Based Persistence (Flag 5)

Objective: Registry keys under Run or RunOnce are often used for stealthy startup execution. We needed to confirm whether the malware used such a tactic.

KQL Query:

```
DeviceRegistryEvents
| where DeviceName == "anthony-001"
| where Timestamp > ago(30d)
| where RegistryKey has_any("Run", "RunOnce")
| project Timestamp, RegistryKey, RegistryValueName, RegistryValueData, InitiatingProcessFileName
```

<input type="checkbox"/>	Timestamp	RegistryKey	RegistryValueName	RegistryValueData	InitiatingProcessFileName
<input type="checkbox"/>	May 6, 2025 9:02:14 PM	HKEY_CURRENT_USER\S...	BitSecSvc	"C:\ProgramData\BitSentinelCo...	bitsentinelcore.exe
	Timestamp	May 6, 2025 9:02:14 PM			
	RegistryKey	HKEY_CURRENT_USER\S-1-5-21-2009930472-1356288797-1940124928-500\SOFTWARE\Microsoft\Windows\CurrentVersion\Run			
	RegistryValueName	BitSecSvc			
	RegistryValueData	"C:\ProgramData\BitSentinelCore.exe"			
	InitiatingProcessFileName	bitsentinelcore.exe			

Observation:

The malware added a persistence entry under HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run pointing to systemreport.lnk. This ensures the fake AV or keylogger re-launches every time the user logs in.

Step 6: Discover Persistence via Scheduled Task (Flag 6 – Scheduled Task Persistence)

Objective: Persistence mechanisms are crucial in APTs. We aimed to find if the attacker registered any recurring scheduled task to maintain access after reboots. We have to detect scheduled tasks configured for long-term access. To find if scheduled tasks were used to maintain ongoing access without user action.

KQL Query:

```
DeviceProcessEvents
| where DeviceName == "anthony-001"
| where Timestamp > ago(30d)
| where ProcessCommandLine contains "schtasks" or ProcessCommandLine contains "Schedule.Service"
| project Timestamp, InitiatingProcessFileName, ProcessCommandLine
| sort by Timestamp desc
```

<input type="checkbox"/>	Timestamp	InitiatingProcessFileName	ProcessCommandLine
<input type="checkbox"/>	> May 6, 2025 9:03:20 PM	cmd.exe	schtasks /Create /SC DAILY /TN "UpdateHealthTelemetry" /TR "C:\ProgramData\BitSentinelCore.exe" /ST 14:00
<input type="checkbox"/>	> May 6, 2025 9:03:20 PM	bitsentinelcore.exe	"cmd.exe" /c schtasks /Create /SC DAILY /TN "UpdateHealthTelemetry" /TR "C:\ProgramData\BitSentinelCore.exe" /ST 14:00
<input type="checkbox"/>	> May 6, 2025 9:03:16 PM	cmd.exe	schtasks /Create /SC DAILY /TN "UpdateHealthTelemetry" /TR "C:\ProgramData\BitSentinelCore.exe" /ST 14:00
<input type="checkbox"/>	> May 6, 2025 9:03:16 PM	bitsentinelcore.exe	"cmd.exe" /c schtasks /Create /SC DAILY /TN "UpdateHealthTelemetry" /TR "C:\ProgramData\BitSentinelCore.exe" /ST 14:00
<input type="checkbox"/>	> May 6, 2025 9:02:15 PM	cmd.exe	schtasks /Create /SC DAILY /TN "UpdateHealthTelemetry" /TR "C:\ProgramData\BitSentinelCore.exe" /ST 14:00
<input type="checkbox"/>	> May 6, 2025 9:02:14 PM	bitsentinelcore.exe	"cmd.exe" /c schtasks /Create /SC DAILY /TN "UpdateHealthTelemetry" /TR "C:\ProgramData\BitSentinelCore.exe" /ST 14:00

Observation:

A task named UpdateHealthTelemetry was created using schtasks.exe, launched from cmd.exe, which was initiated by BitSentinelCore.exe. This confirms the malware established long-term persistence via task scheduling.

Step 7: Trace Process Chain to Task Creation (Flag 7-Process Spawn Chain)

To confirm the execution chain that led from the initial dropper to the persistence setup. This helps visualize the attacker’s multi-stage delivery. We have to trace parent-child-grandchild relationships for the process chain. To show the hierarchy from the initial binary to the persistence command.

KQL Query:

DeviceProcessEvents
| where DeviceName == "anthony-001"
| where Timestamp > ago(30d)
| where ProcessCommandLine has "schtasks" or ProcessCommandLine has "/Create"
| project Timestamp, FileName, ProcessCommandLine, InitiatingProcessFileName,
InitiatingProcessCommandLine
| order by Timestamp asc

Timestamp	FileName	ProcessCommandLine	InitiatingProcess	InitiatingProcessCommandLine
5/6/2025 21:02	cmd.exe	"cmd.exe" /c schtasks /Create /SC DAILY /TN "UpdateHealthTelemetry" /TR "C:\ProgramData\BitSent	bitsentinelcore.exe	BitSentinelCore.exe
5/6/2025 21:02	schtasks.exe	schtasks /Create /SC DAILY /TN "UpdateHealthTelemetry" /TR "C:\ProgramData\BitSentinelCore.exe	cmd.exe	"cmd.exe" /c schtasks /Create /SC DAILY /TN "UpdateHealthTelemetry" /TR
5/6/2025 21:03	cmd.exe	"cmd.exe" /c schtasks /Create /SC DAILY /TN "UpdateHealthTelemetry" /TR "C:\ProgramData\BitSent	bitsentinelcore.exe	BitSentinelCore.exe
5/6/2025 21:03	schtasks.exe	schtasks /Create /SC DAILY /TN "UpdateHealthTelemetry" /TR "C:\ProgramData\BitSentinelCore.exe	cmd.exe	"cmd.exe" /c schtasks /Create /SC DAILY /TN "UpdateHealthTelemetry" /TR
5/6/2025 21:03	cmd.exe	"cmd.exe" /c schtasks /Create /SC DAILY /TN "UpdateHealthTelemetry" /TR "C:\ProgramData\BitSent	bitsentinelcore.exe	BitSentinelCore.exe
5/6/2025 21:03	schtasks.exe	schtasks /Create /SC DAILY /TN "UpdateHealthTelemetry" /TR "C:\ProgramData\BitSentinelCore.exe	cmd.exe	"cmd.exe" /c schtasks /Create /SC DAILY /TN "UpdateHealthTelemetry" /TR

<input type="checkbox"/>	Timestamp	FileName	ProcessCommandLine	InitiatingProcessFileName	InitiatingProcessCommandL...	
<input type="checkbox"/>	>	May 6, 2025 9:02:14 PM	cmd.exe	"cmd.exe" /c schtasks /Create /SC DAILY ...	bitsentinelcore.exe	BitSentinelCore.exe
<input type="checkbox"/>	>	May 6, 2025 9:02:15 PM	schtasks.exe	schtasks /Create /SC DAILY /TN "Update...	cmd.exe	"cmd.exe" /c schtasks /C...
<input type="checkbox"/>	>	May 6, 2025 9:03:16 PM	cmd.exe	"cmd.exe" /c schtasks /Create /SC DAILY ...	bitsentinelcore.exe	BitSentinelCore.exe
<input type="checkbox"/>	>	May 6, 2025 9:03:16 PM	schtasks.exe	schtasks /Create /SC DAILY /TN "Update...	cmd.exe	"cmd.exe" /c schtasks /C...
<input type="checkbox"/>	>	May 6, 2025 9:03:20 PM	cmd.exe	"cmd.exe" /c schtasks /Create /SC DAILY ...	bitsentinelcore.exe	BitSentinelCore.exe
<input type="checkbox"/>	>	May 6, 2025 9:03:20 PM	schtasks.exe	schtasks /Create /SC DAILY /TN "Update...	cmd.exe	"cmd.exe" /c schtasks /C...

Context:
This query was used to trace the hierarchy of processes involved in creating persistence through a scheduled task. By inspecting DeviceProcessEvents filtered for "schtasks" or "/Create", we can reveal not only that the scheduled task was created, but also which processes were responsible for launching each step.

- What the Output Shows:
- At May 6, 2025 9:02:14 PM, BitSentinelCore.exe launches cmd.exe with the command to create a scheduled task.
 - One second later, at 9:02:15 PM, that same cmd.exe process launches schtasks.exe to execute the task creation.
 - This pattern repeats at 9:03:16 PM and 9:03:20 PM, confirming multiple attempts or retries to ensure task creation.

Observation:
Chain confirmed as:
gc_worker.exe → BitSentinelCore.exe → cmd.exe → schtasks.exe.
(GrandParent.exe → Parent.exe → Child.exe → GrandChild.exe.)
Though several variants exist, the official CTF answer required a specific chain matching the challenge’s narrative format.

Step 8: Root Cause Timestamp (Flag 8-Timestamp Correlation)


Objective: Identify the timestamp of the earliest root action that triggered the attack. Correlates event timing to prove cause-and-effect chain tied to fake antivirus.

KQL Query:

```
DeviceFileEvents
| where DeviceName == "anthony-001"
| where Timestamp > ago(30d)
| where FileName == "BitSentinelCore.exe"
| order by Timestamp asc
| project Timestamp, DeviceName, ActionType, FileName
```

Timestamp	DeviceName	ActionType	FileName
5/6/2025 21:00	anthony-001	FileCreated	BitSentinelCore.exe


☐ TimestampDeviceNameActionTypeFileName

☐ ▾ May 6, 2025 9:00:36...  anthony-001FileCreatedBitSentinelCore.exe

Timestamp

May 6, 2025 9:00:36 PM

DeviceName

 anthony-001

ActionType

FileCreated

FileName

BitSentinelCore.exe

Observation: The correlated timestamp was 2025-05-07T02:00:36.794406Z (5/6/2025 21:00), marking when BitSentinelCore.exe first appeared on disk.

6. Final Observations

- Attack began **before** user execution of BitSentinelCore.exe.
- The fake AV ([BitSentinelCore.exe](#)) was likely a second-stage payload, with previous actions handled by PowerShell and senseir.exe.
- The CTF expected **2025-05-07T02:00:36.794406Z** as the official "root timestamp" even though earlier events like rundll32 injection happened.

7. Timeline of Events

Time (UTC)	Event	Description
2025-05-06T22:01:28Z	PowerShell execution	Initiated by senseir.exe, used to load further stages
2025-05-06T22:01:58Z	csc.exe execution	PowerShell called .NET compiler to build payload

Time (UTC)	Event	Description
2025-05-06T22:02:25Z	gc_worker.exe	Credential harvesting using reversible encryption
2025-05-06T22:03:16Z	schtasks.exe	Scheduled task "UpdateHealthTelemetry" created
2025-05-06T22:06:51Z	systemreport.lnk written	Keylogger shortcut dropped into Startup folder
2025-05-06T20:23:40Z	rundll32.exe (earliest DLL injection)	Indicates pre-positioned code injection
2025-05-07T02:00:36.794406Z	BitSentinelCore.exe file creation	The officially accepted CTF timestamp for Flag 8

8. MITRE ATT&CK Mapping

Tactic	Technique	ID	Observed Action
Execution	User Execution	T1204	senseir.exe launching PowerShell
Defense Evasion	Obfuscated Files	T1027	PowerShell + EncodedCommand usage
Persistence	Scheduled Task/Startup	T1053.005 / T1547.001	UpdateHealthTelemetry + systemreport.lnk
Credential Access	OS Credential Dumping	T1003	gc_worker.exe using StorePasswords flags
Collection	Input Capture	T1056	systemreport.lnk (AutoHotkey keylogger)

9. Incident Response and Recommendations

Incident Containment Actions:

- Isolated the endpoint anthony-001 from the network to prevent further spread.
- Disabled the created scheduled task UpdateHealthTelemetry.
- Deleted the file BitSentinelCore.exe and removed associated .lnk files from startup.
- Removed the malicious registry key under HKCU\Software\Microsoft\Windows\CurrentVersion\Run.

Forensic Preservation:

- Exported all relevant logs from Defender for Endpoint.
 - Captured memory images for advanced investigation.
 - Saved file hashes and paths of artifacts such as BitSentinelCore.exe, AutoHotkeyU32.exe, and systemreport.lnk.
-

10. Recommendations to Harden Systems:

- **Application Control:** Implement AppLocker or WDAC to block execution of unauthorized executables like csc.exe and PowerShell scripts.
 - **PowerShell Logging:** Enable enhanced PowerShell transcription and command line logging via Group Policy.
 - **Startup Folder Monitoring:** Set real-time alerts on writes to %AppData%\Microsoft\Windows\Start Menu\Programs\Startup.
 - **Registry Monitoring:** Monitor and alert on new Run/RunOnce keys in user and system hives.
 - **Scheduled Task Review:** Periodically audit scheduled tasks across endpoints for unusual task names and creators.
 - **User Awareness Training:** Educate staff on identifying phishing and suspicious software prompts.
 - **Endpoint Detection & Response (EDR):** Ensure Defender EDR or equivalent is actively scanning and configured with behavioral rules.
 - **Patch Management:** Regularly update .NET and PowerShell versions to close vulnerabilities exploited by LOLBins like csc.exe.
-

End of Report