

Behavioral Malware Analysis: Investigating a Multi-Stage Malware Sample Inside an Isolated Lab

 infosecwriteups.com/behavioral-malware-analysis-investigating-a-multi-stage-malware-sample-inside-an-isolated-lab-fba44d152d5b

Allen Ace

August 4, 2026

How I investigated persistence, C2 communications, process injection, and data exfiltration using FLARE-VM, REMnux, Sysmon, Procmon, Wireshark, and MITRE ATT&CK.

Press enter or click to view image in full size



Static analysis provides a valuable starting point, but it rarely tells the complete story of what a piece of malware is truly capable of.

Modern malware is designed to reveal its most dangerous behaviors only at runtime, establishing persistence, communicating with attacker-controlled infrastructure, fetching additional payloads, evading defensive controls, and exfiltrating sensitive data. Capturing and understanding these behaviors requires observing the sample as it executes within a controlled and monitored environment.

To replicate the conditions of a real-world investigation, I constructed an isolated behavioral analysis laboratory using **FLARE-VM** and **REMnux**, and subjected a simulated multi-stage malware sample — `walkthrough.exe` — to a structured dynamic analysis workflow.

Rather than treating this as a tool demonstration, the investigation follows the same evidence-driven methodology used by malware analysts, DFIR investigators, and SOC teams in operational settings: collecting telemetry across multiple sources, correlating findings, extracting Indicators of Compromise (IOCs), mapping observed behaviors to the **MITRE ATT&CK** framework, and translating those observations into actionable detection opportunities.

This article documents the investigation from initial execution through to final detection engineering.

Why Behavioral Analysis Matters

Static analysis answers questions like:

- What functions exist?
- Which strings are embedded?
- Is the binary packed?

Behavioral analysis answers a different set of questions:

- How does the malware execute?
- Does it establish persistence?
- Does it communicate externally?
- Does it retrieve additional payloads?
- Does it inject into legitimate processes?
- What indicators can defenders use to detect it?

These questions are often the difference between identifying a malicious file and understanding an active compromise.

Building the Investigation Lab

Before executing any malware, a safe and isolated environment is essential.

The laboratory consisted of two virtual machines:

Press enter or click to view image in full size

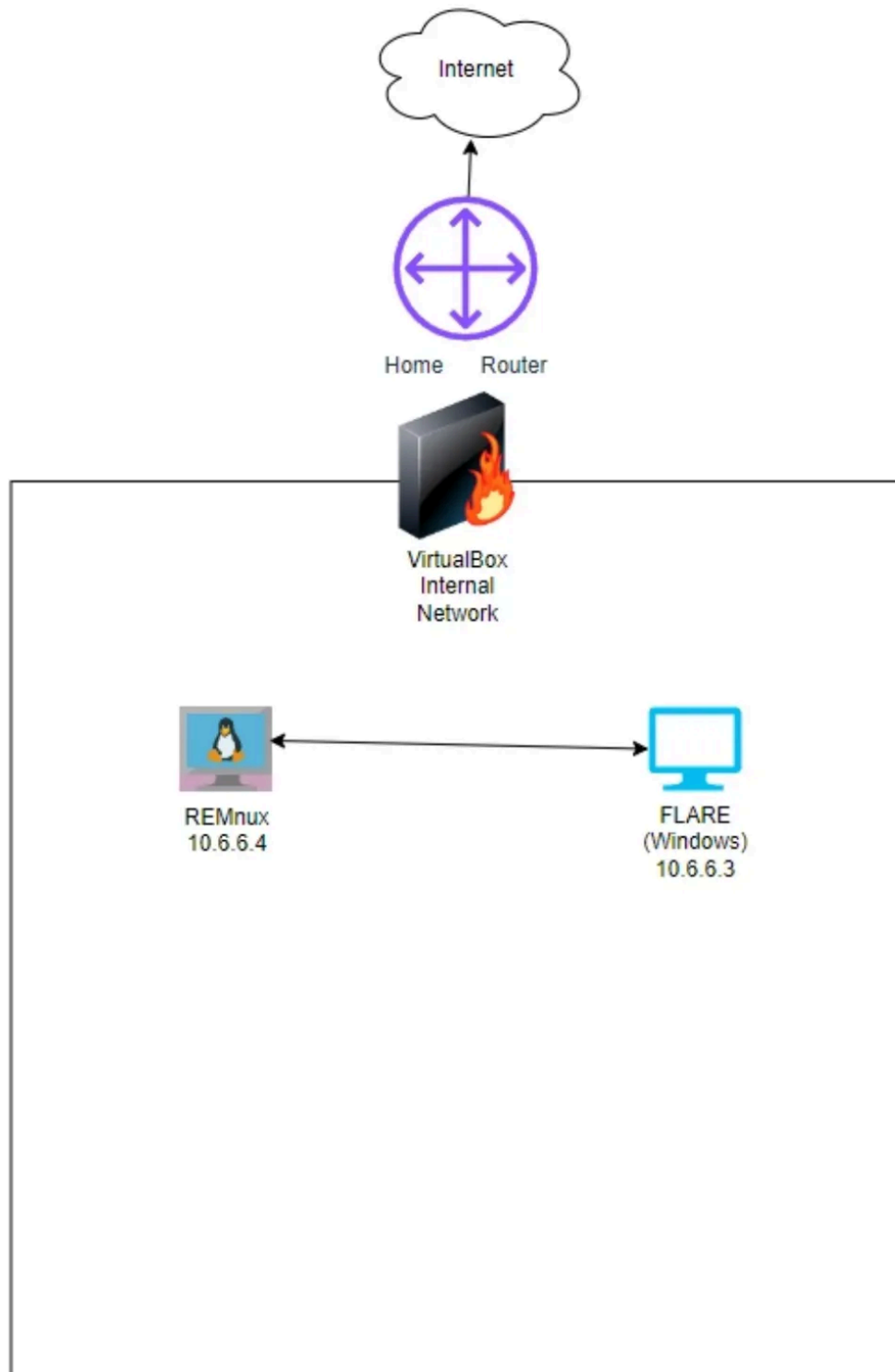


Figure 1. Malware analysis laboratory consisting of FLARE-VM and REMnux connected through a Host-Only network.

Communication occurred exclusively over a Host-Only VirtualBox network, ensuring the malware could not reach the public Internet.

Instead of allowing outbound connections, INetSim simulated Internet services while FakeDNS redirected all DNS requests to REMnux.

This configuration allowed the malware to behave naturally without exposing external systems.

Investigation Methodology

Rather than focusing on individual tools, I treated this analysis as a complete investigation.

Each observation had to answer an investigative question:

1. What happened?
2. Which tool observed it?
3. What evidence supports the finding?
4. What does it mean?
5. Can defenders detect it?

This methodology reduced assumptions and ensured every conclusion was supported by evidence collected during execution.

Stage 1: Initial Execution

The sample executed successfully inside FLARE-VM.

Immediately after execution, new child processes appeared, indicating that the malware had begun interacting with the operating system.

Process Explorer provided the first view of the execution chain.

Press enter or click to view image in full size

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
svchost.exe		1,624 K	7,428 K	2184	Host Process for Windows S...	Microsoft Corporation
svchost.exe		17,520 K	33,448 K	2284	Host Process for Windows S...	Microsoft Corporation
OfficeClickToRun.exe		33,480 K	59,964 K	2320	Microsoft Office Click-to-Run...	Microsoft Corporation
Sysmon.exe	1.01	16,120 K	24,576 K	2420		
unsecapp.exe		1,300 K	6,892 K	2632	Sink to receive asynchronou...	Microsoft Corporation
sihost.exe		5,412 K	25,280 K	3280	Shell Infrastructure Host	Microsoft Corporation
svchost.exe		10,716 K	41,244 K	3292	Host Process for Windows S...	Microsoft Corporation
MicrosoftEdgeUpdate.exe		2,616 K	3,568 K	3340	Microsoft Edge Update	Microsoft Corporation
taskhostw.exe		5,884 K	16,728 K	3408	Host Process for Windows T...	Microsoft Corporation
ctfmon.exe		4,148 K	20,756 K	3604	CTF Loader	Microsoft Corporation
explorer.exe	0.50	61,044 K	157,924 K	4012	Windows Explorer	Microsoft Corporation
SecurityHealthSystray.exe		1,656 K	9,108 K	5700	Windows Security notificatio...	Microsoft Corporation
VBoxTray.exe	< 0.01	2,592 K	11,088 K	5756	VirtualBox Guest Additions Tr...	Oracle and/or its affiliates
ZoomIt64.exe		1,892 K	8,576 K	5980	Sysinternals Screen Magnifier	Sysinternals - www.sysinter...
procexp64.exe	2.01	28,848 K	54,712 K	3664	Sysinternals Process Explorer	Sysinternals - www.sysinter...
walkthrough.exe	< 0.01	1,388 K	7,340 K	2968		
walkthrough.exe		28,236 K	34,732 K	5000		
dllhost.exe		3,176 K	11,292 K	1924	COM Surrogate	Microsoft Corporation
svchost.exe		1,888 K	8,740 K	4152	Host Process for Windows S...	Microsoft Corporation
SearchIndexer.exe		24,924 K	34,736 K	4184	Microsoft Windows Search I...	Microsoft Corporation
svchost.exe	< 0.01	3,704 K	21,580 K	4292	Host Process for Windows S...	Microsoft Corporation
StartMenuExperienceHost.exe		21,048 K	65,796 K	2148		
RuntimeBroker.exe		5,604 K	23,844 K	2892	Runtime Broker	Microsoft Corporation
SearchApp.exe	Susp...	102,472 K	178,104 K	3036	Search application	Microsoft Corporation
RuntimeBroker.exe		18,352 K	52,060 K	2716	Runtime Broker	Microsoft Corporation
Regshot-x64-Unicode.exe		234,084 K	258,604 K	888	Regshot 1.9.1 x64 Unicode (...)	Regshot Team
LockApp.exe	Susp...	10,684 K	46,000 K	4624	LockApp.exe	Microsoft Corporation
RuntimeBroker.exe		7,848 K	31,844 K	1236	Runtime Broker	Microsoft Corporation
RuntimeBroker.exe		2,488 K	13,676 K	5460	Runtime Broker	Microsoft Corporation
SecurityHealthService.exe		4,088 K	15,604 K	5732		
svchost.exe		1,872 K	7,664 K	1244	Host Process for Windows S...	Microsoft Corporation
SgmBroker.exe		3,228 K	6,772 K	6088		
svchost.exe		2,916 K	10,672 K	2996		
ShellExperienceHost.exe	Susp...	11,508 K	49,912 K	6096	Windows Shell Experience H...	Microsoft Corporation
RuntimeBroker.exe		2,604 K	17,020 K	2756	Runtime Broker	Microsoft Corporation
cmd.exe		2,036 K	1,980 K	4264	Windows Command Processor	Microsoft Corporation
conhost.exe	< 0.01	6,668 K	1,624 K	4800	Console Window Host	Microsoft Corporation

CPU Usage: 5.03% Commit Charge: 27.98% Processes: 74 Physical Usage: 52.43%

Figure 2. Process tree captured immediately after execution.

Stage 2: Registry Persistence

The next question was whether the malware could survive a reboot.

Using Procmon, registry modifications were filtered to display only **RegSetValue** operations.

This quickly revealed creation of a Run key:

HKCU\Software\Microsoft\Windows\CurrentVersion\Run\FakeMalware

Rather than relying on one tool, the finding was validated using:

- Procmon

- Sysmon Event ID 13
- Autoruns
- Regshot

Four independent sources confirmed the persistence mechanism.

Join Medium for free to get updates from this writer.

Remember me for faster sign in

This illustrates why evidence correlation is more valuable than trusting a single artifact.

Press enter or click to view image in full size

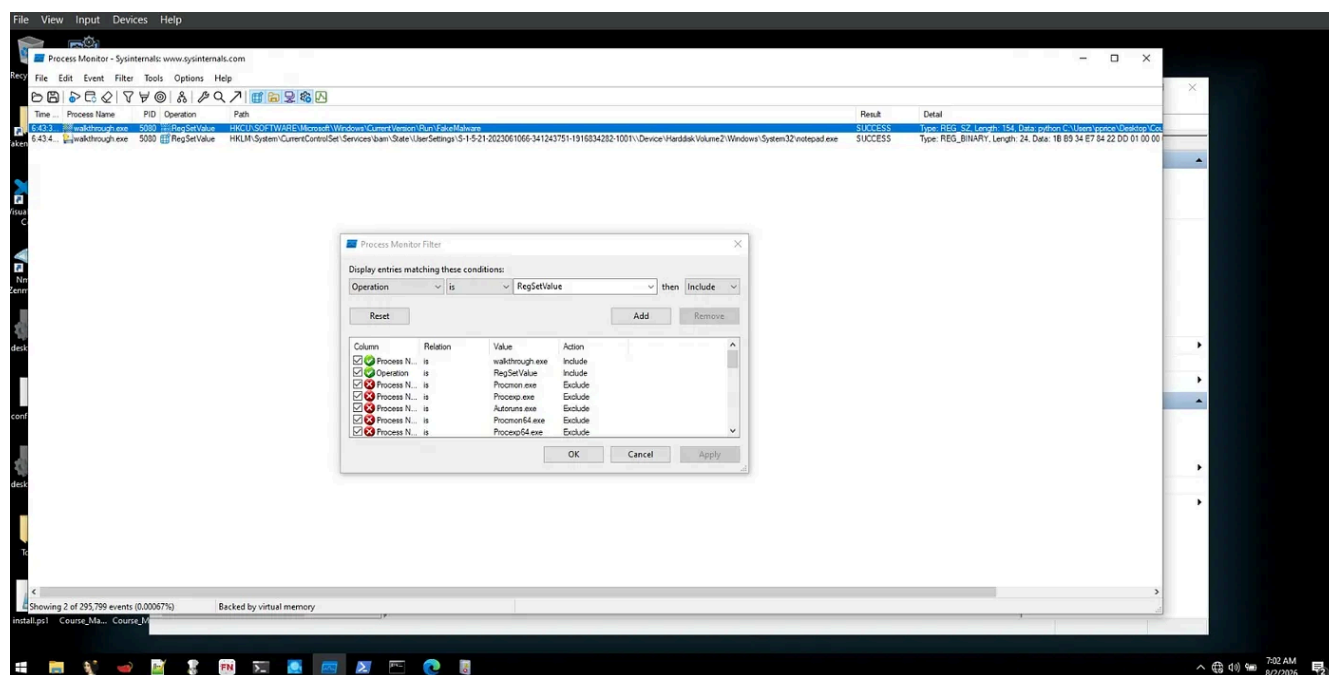


Figure 3. Registry value creation observed in Procmon.

Press enter or click to view image in full size

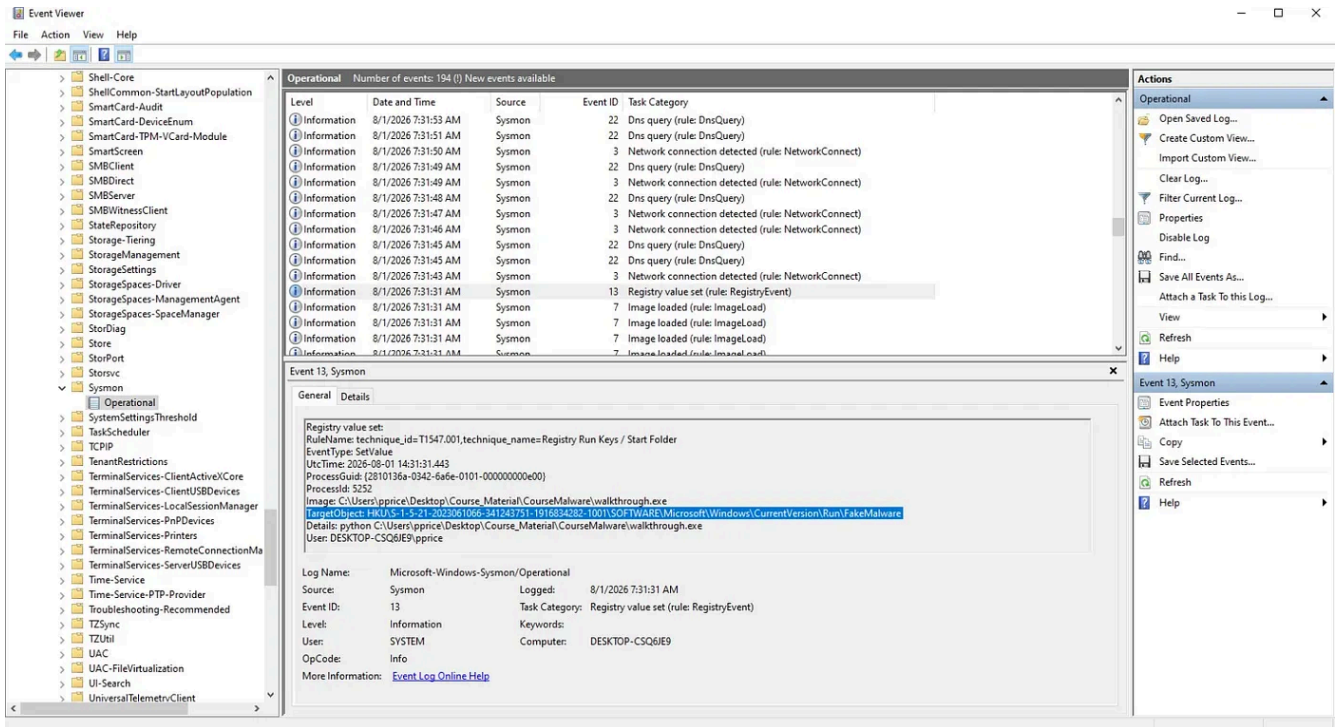


Figure 4. Sysmon confirming registry modification.

Press enter or click to view image in full size

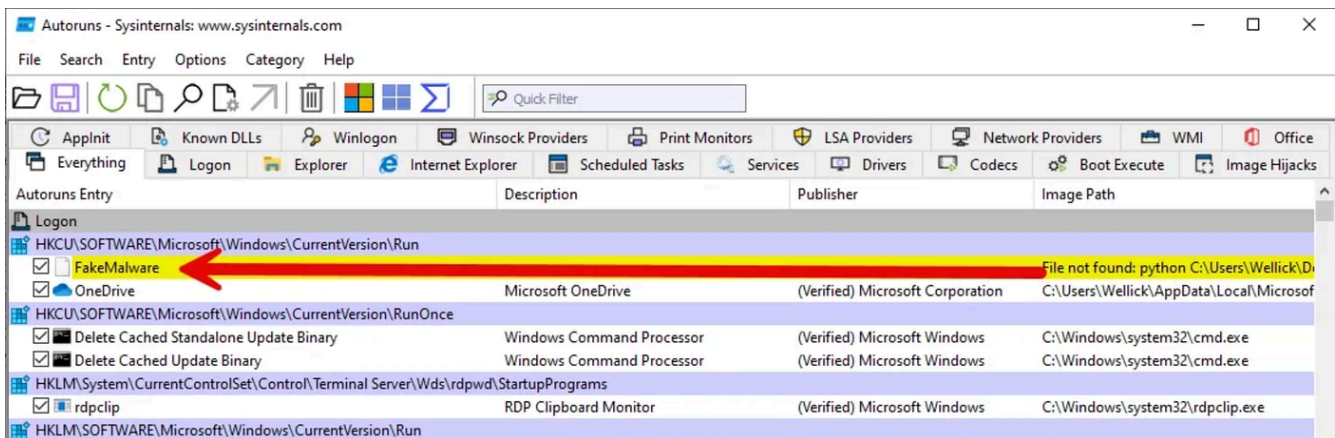


Figure 5. Autoruns identifying the startup entry.

Press enter or click to view image in full size

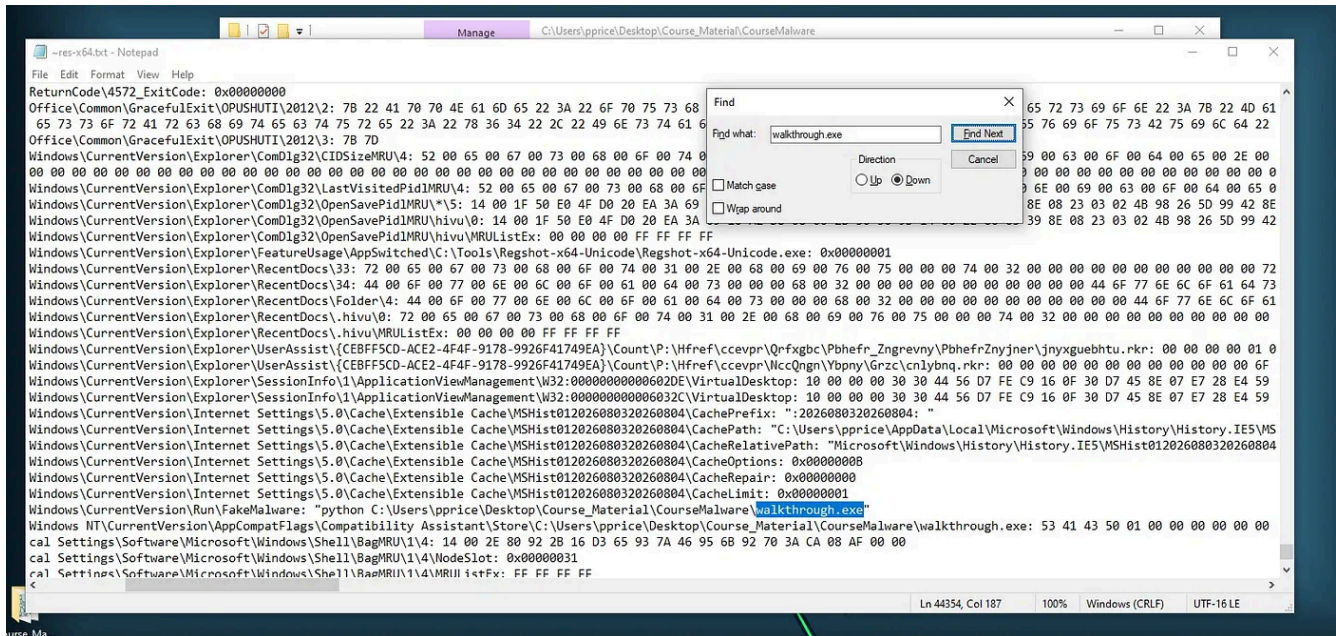


Figure 6. Regshot highlighting registry differences before and after execution.

Stage 3: Command-and-Control Communication

Once persistence was established, attention shifted to network behavior.

FakeDNS recorded multiple DNS requests generated by the sample, while Wireshark captured outbound HTTP traffic redirected to REMnux.

Although the infrastructure was simulated, the observed behavior closely resembled real command-and-control beaconing.

The malware attempted to:

- Resolve dynamically generated domains
- Establish outbound HTTP sessions
- Retrieve additional resources
- Transmit collected information

Press enter or click to view image in full size

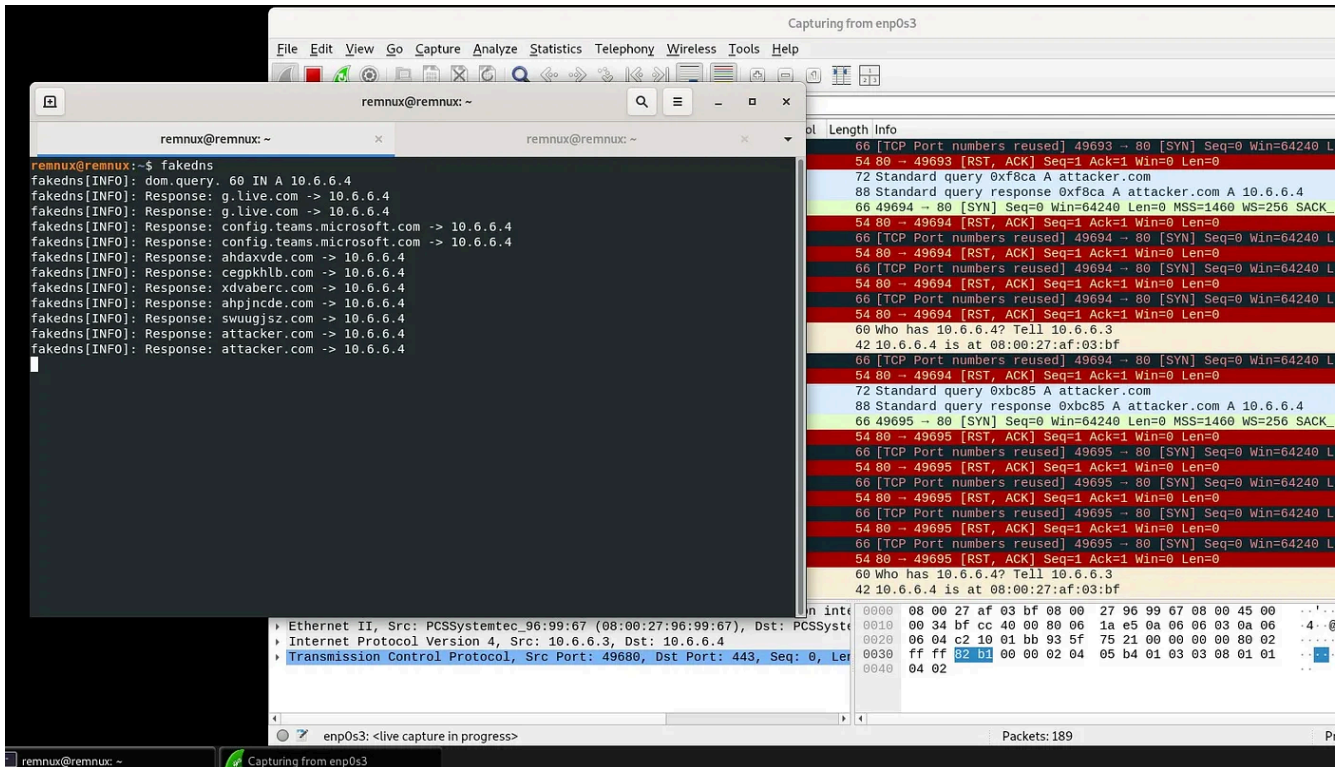


Figure 7. DNS requests redirected to REMnux.

Press enter or click to view image in full size

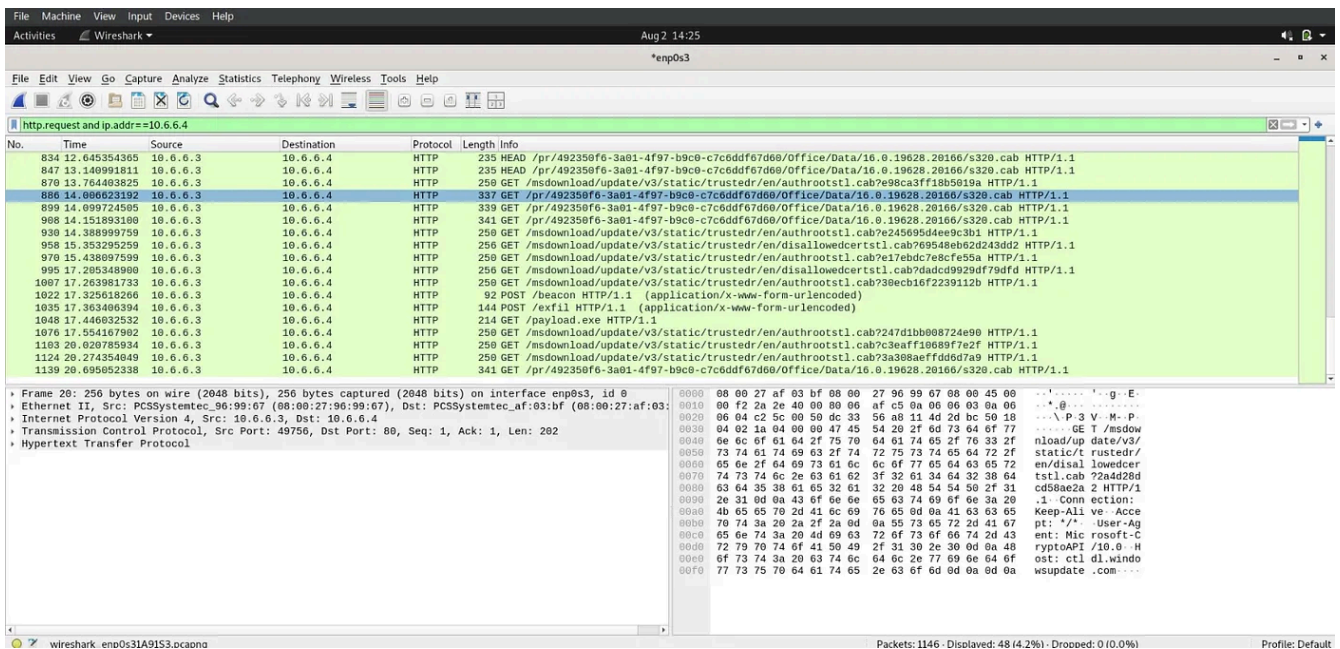


Figure 8. HTTP beaconing observed during execution.

Stage 4: Secondary Payload Retrieval

One of the most interesting observations occurred when the malware requested an encrypted payload.

Wireshark showed:

GET /payload.exe

The payload was extracted directly from the packet capture.

Although simulated, this accurately represents how analysts recover downloaded malware for additional analysis.

Press enter or click to view image in full size

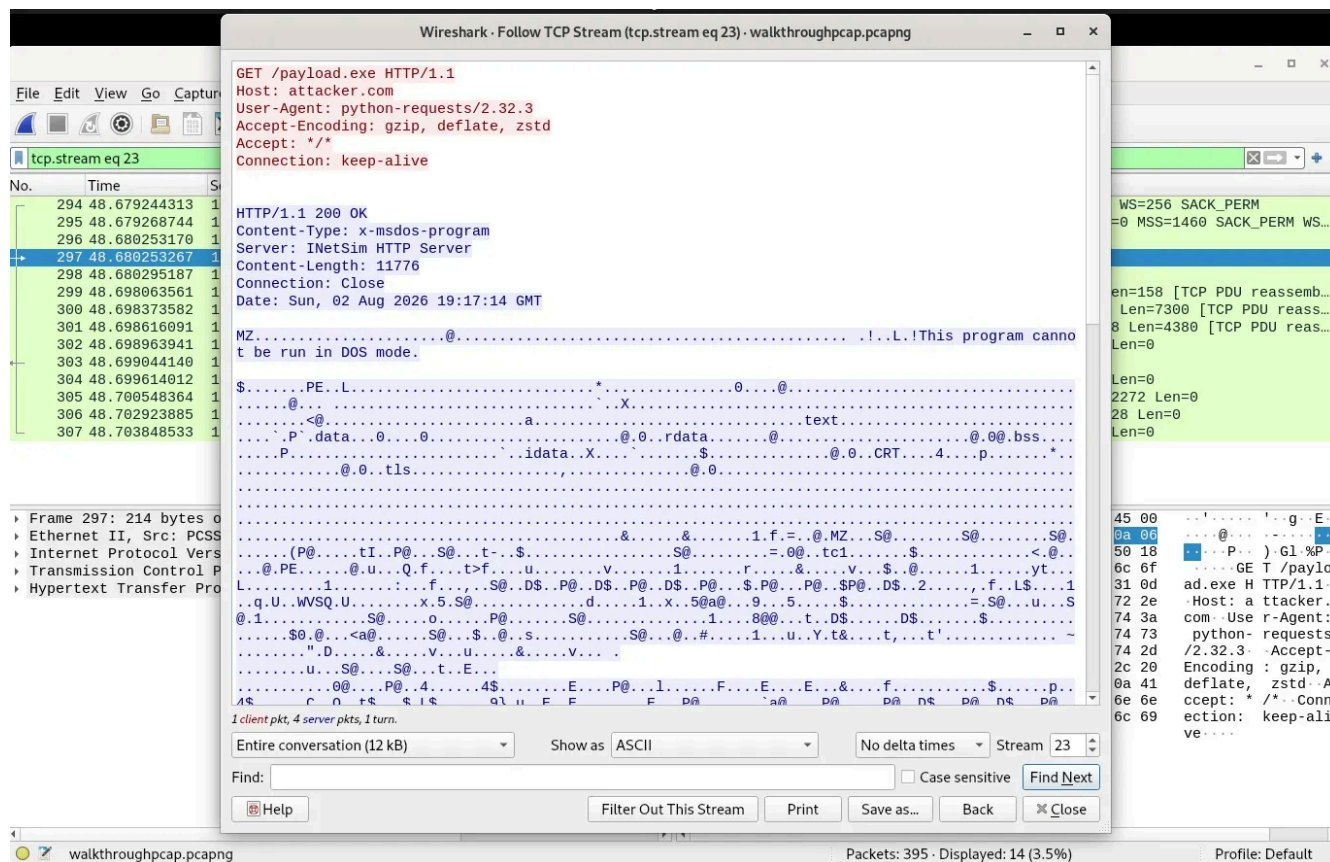


Figure 9. Retrieval of an encrypted second-stage payload.

Press enter or click to view image in full size

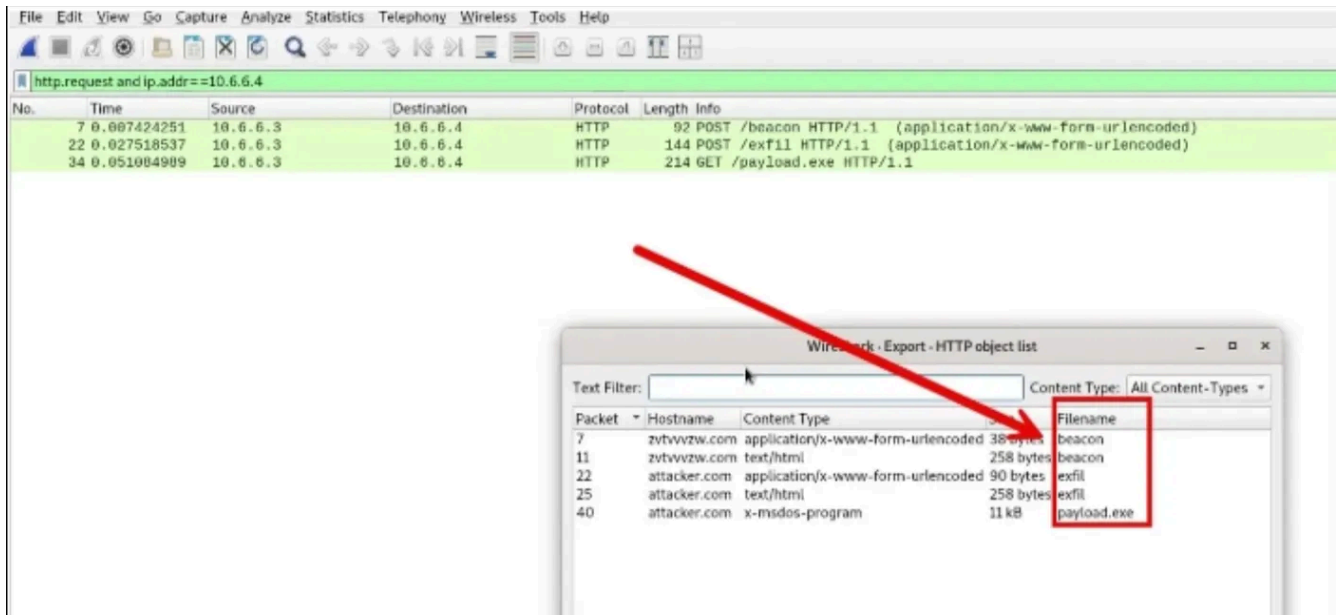


Figure 10. Retrieval of a second-stage payload.

Stage 5: Process Injection

Many modern malware families inject into legitimate processes to evade detection.

API Monitor identified API activity associated with process injection, while Process Explorer showed the target process.

The malware injected into **Notepad.exe**, allowing malicious code to execute under the context of a trusted Windows process.

Press enter or click to view image in full size

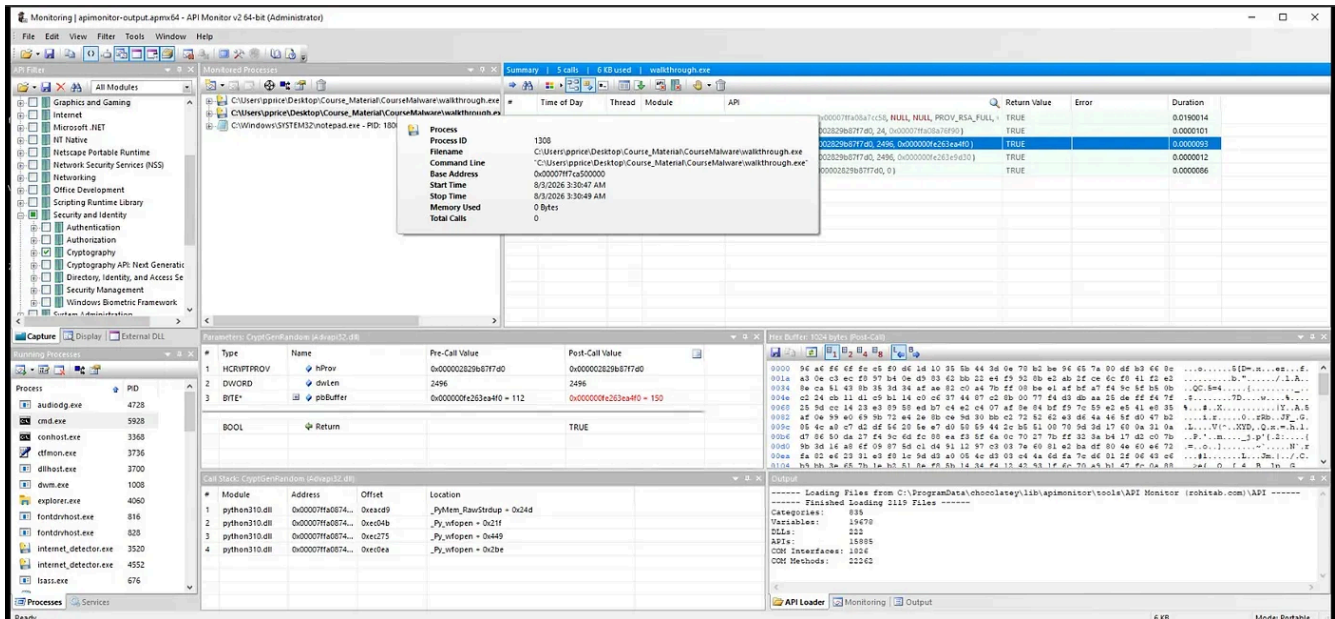


Figure 11. API activity associated with process injection.

Stage 6: Data Exfiltration

Finally, Wireshark captured outbound HTTP POST requests consistent with simulated data exfiltration.

Rather than viewing this as an isolated network event, the evidence was correlated with the earlier execution chain to reconstruct the complete attack lifecycle:

Execution

↓

Persistence

↓

Command-and-Control

↓

Payload Retrieval

↓

Process Injection

↓

Data Exfiltration

This chronological reconstruction mirrors how malware investigations are conducted in enterprise DFIR engagements.

Press enter or click to view image in full size

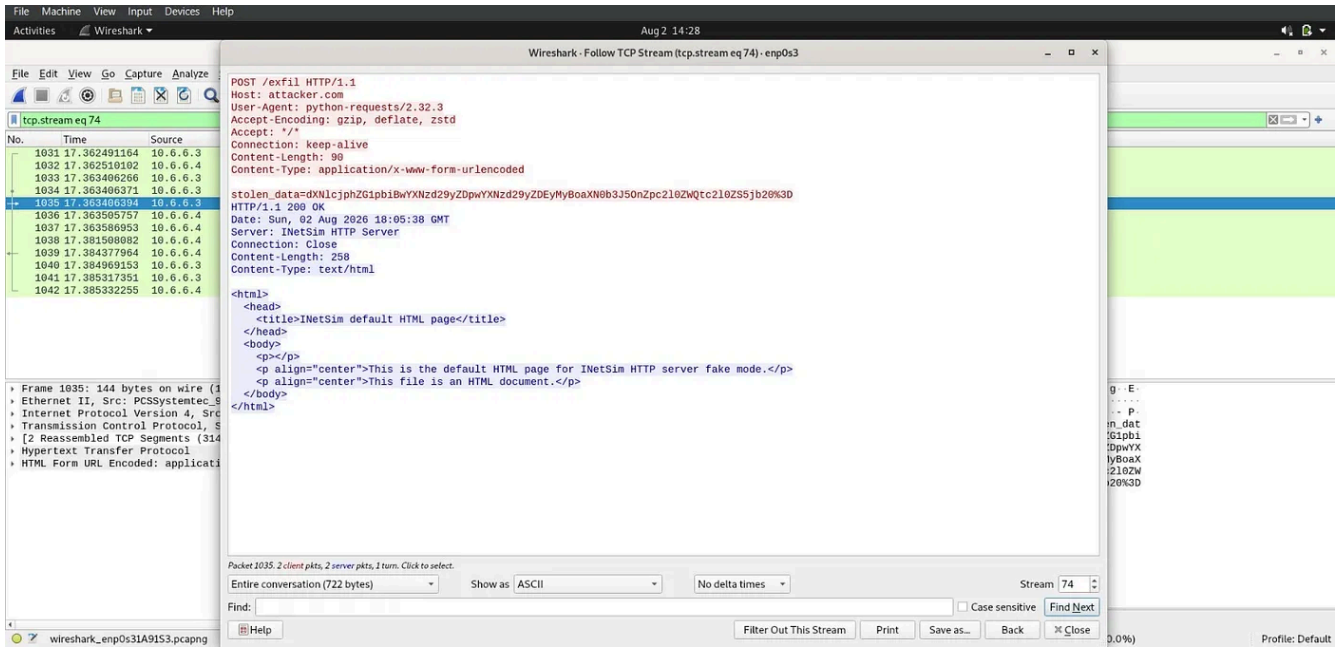


Figure 12. Simulated outbound data exfiltration.

Mapping the Investigation to MITRE ATT&CK

One of the goals of behavioral analysis is translating observations into standardized adversary techniques.

Press enter or click to view image in full size

ATT&CK Technique	Description
T1547.001	Registry Run Keys
T1071.001	Web Protocols
T1071.004	DNS
T1105	Ingress Tool Transfer
T1055	Process Injection
T1041	Exfiltration Over C2 Channel

Figure13. Mapping the Investigation to MITRE ATT&CK

Using MITRE ATT&CK helps defenders communicate findings consistently across SOC, DFIR, and threat hunting teams.

From Investigation to Detection Engineering

An investigation has limited value if the findings cannot be operationalized.

Using the collected evidence, I developed detection content including:

- Sigma rules
- Splunk SPL searches
- Microsoft Sentinel KQL queries
- YARA rules

These detections enable defenders to identify similar behaviors in production environments rather than relying on file hashes alone.

Conclusion

Behavioral malware analysis extends far beyond observing a malicious executable. It is a disciplined investigative process that combines host telemetry, network captures, forensic artifacts, and threat intelligence to reconstruct an attacker's actions with confidence.

In this investigation, **walkthrough.exe** demonstrated a complete multi-stage execution chain involving registry-based persistence, command-and-control communication, secondary payload retrieval, process injection, and simulated data exfiltration. By correlating evidence across Procmon, Sysmon, Wireshark, Autoruns, Regshot, Process Explorer, and API Monitor, the investigation produced validated indicators of compromise, MITRE ATT&CK mappings, and practical detection opportunities.

This project reflects the investigative workflow used by modern SOC, DFIR, and malware analysis teams, where the goal is not only to understand malicious behavior but also to convert those findings into actionable detections that strengthen defensive capabilities.

GitHub Repository

The complete investigation, evidence, detection rules, IOC datasets, and documentation are available on GitHub.

★ If you find the project useful, consider starring the repository and sharing your feedback.

| Repository: [GitHub](#)

Contact: [Portfolio](#)