

# Endpoint Memory Forensic Investigation

FOR502

Uncover in-memory threats, fileless malware, and advanced attacker activity that never touches the disk. Capture, analyze, and extract critical evidence from volatile memory across Windows and Linux using Volatility 3, MemProcFS, and WinDbg.

|                     |                                 |
|---------------------|---------------------------------|
| <b>Modules</b>      | 18                              |
| <b>Lessons</b>      | 114                             |
| <b>Phases</b>       | 4                               |
| <b>Free preview</b> | Course Orientation (11 lessons) |

## The method

Memory holds what disk never recorded, and it holds it for as long as the machine stays up. This course runs the same loop for every technique it investigates.

1. Acquire before you decide it matters. Memory is the most volatile evidence you can take and the decision to capture it is made under time pressure with incomplete information. Capture, then triage.
2. Establish what normal looks like on this system. Process trees, loaded modules, network connections. Anomaly detection in memory is comparison, and without a baseline every unfamiliar service looks suspicious.
3. Read the artifact the technique cannot avoid. Injection has to allocate and write. Credential theft has to read a specific process. Reason from what the technique requires rather than from what the tool is named.
4. Corroborate against disk and log. A memory finding on its own is a strong lead. The same activity in a log or on disk, with times that agree, is a conclusion.
5. Build the timeline across sources. Memory tells you what was true at capture. Disk and logs tell you how it got that way, and the intrusion is the sequence rather than the snapshot.

## What this course assumes

No minimum experience and no prerequisite course. Process structures, virtual memory and the acquisition model are explained from the ground up.

What makes it go faster: a machine you can capture memory from and enough disk to hold the image. Not required, and every analysis in the course is worked against captures the course supplies.

What this course does not cover: reverse engineering, disk forensics in depth, and network investigation. Those are separate courses, and memory analysis is strongest when it corroborates them rather than replaces them.

# Full syllabus

## Phase 1: Foundations

### Course Orientation [FREE PREVIEW](#)

What Endpoint Memory Forensic Investigation teaches: capture and analyze volatile memory across Windows and Linux to find the fileless malware, injected code, and stolen credentials that never touch disk.

1. MF0.1 Why Memory Forensics Matters
2. MF0.2 What Lives in Memory That Isn't on Disk
3. MF0.3 The Memory Forensics Workflow and Documentation Standard
4. MF0.4 Physical Memory, Virtual Memory, and Why It Matters for Forensics
5. MF0.5 Volatility 3 and MemProcFS, the Analysis Framework
6. MF0.6 WinDbg and the Kernel Debugger Perspective
7. MF0.7 Evidence Reliability and Confidence Assessment
8. MF0.8 Legal Context for Memory Evidence
9. MF0.9 Lab Environment Setup
10. MF0.10 Volatility 3 and MemProcFS Orientation
11. MF0.11 Interactive: Triage a Suspect Capture

### Memory Acquisition

Acquisition as a standard IR step, not a specialist exception.

1. MF1.1 The Acquisition Problem
2. MF1.2 Windows Acquisition with WinPmem
3. MF1.3 Linux Acquisition with LiME and AVML
4. MF1.4 Hypervisor-Based Acquisition
5. MF1.5 Pagefile and Swap as Memory-Adjacent Evidence
6. MF1.6 Acquisition Verification and Integrity
7. MF1.7 Smear Detection and Acquisition Quality
8. MF1.8 Anti-Acquisition Techniques
9. MF1.9 Interactive: Decide the Acquisition

## Phase 2: Windows Memory Analysis

### Process Injection

The first transient-artifact category in depth.

1. MF2.1 The Process Abstraction in Memory
2. MF2.2 Virtual Memory and Page Tables
3. MF2.3 VAD Tree Fundamentals
4. MF2.4 Executing the Attack
5. MF2.5 Acquiring the Compromised Capture
6. MF2.6 VAD Analysis of the Injected Region
7. MF2.7 Process Hollowing and Doppelgänger
8. MF2.8 Extracting and Analyzing Injected Code
9. MF2.9 Interactive: Hunt the Injections

## Credential Theft in Memory

Detecting and proving credential theft from a captured image.

1. MF3.1 LSASS and the Authentication Architecture
2. MF3.2 Executing the Attack
3. MF3.3 Capturing Post-Credential-Theft Memory
4. MF3.4 Extracting NTLM Hashes from LSASS
5. MF3.5 Kerberos Tickets in LSASS
6. MF3.6 Credential Guard and Its Forensic Implications
7. MF3.7 Memory Signatures of Credential-Theft Tools
8. MF3.8 Non-LSASS Credential Sources
9. MF3.9 Interactive: The Credential Hunt

## Fileless Malware

Investigating malware that lives only in memory.

1. MF4.1 The Fileless Malware Problem
2. MF4.2 PowerShell as an Attack Vehicle
3. MF4.3 Executing the Attack
4. MF4.4 Capturing Post-Attack Memory
5. MF4.5 PowerShell Process Memory Analysis
6. MF4.6 Extracting the In-Memory Payload
7. MF4.7 C2 Configuration Extraction
8. MF4.8 PowerShell Event Log Correlation
9. MF4.9 Interactive: The Fileless Hunt

## Persistence Mechanisms

Finding the mechanisms attackers install to survive a reboot.

1. MF5.1 Persistence as an Attacker Investment
2. MF5.2 Scheduled Tasks in Memory
3. MF5.3 WMI Event Subscriptions in Memory
4. MF5.4 Executing the Attack
5. MF5.5 Capturing Post-Persistence Memory
6. MF5.6 Analyzing Scheduled Task Persistence
7. MF5.7 Analyzing WMI Event Subscriptions
8. MF5.8 Registry in Memory
9. MF5.9 Interactive: The Persistence Hunt

## Kernel-Level Compromise

When the attacker stops being a tenant of the system and becomes a co-owner of the kernel.

1. MF6.1 Crossing into Kernel Space
2. MF6.2 The Driver List, and What It Hides
3. MF6.3 Executing the Attack
4. MF6.4 Capturing the Compromised Host
5. MF6.5 Finding the Hidden Driver
6. MF6.6 Callback Analysis
7. MF6.7 System Call Table Hooks
8. MF6.8 DKOM and the Detection Floor
9. MF6.9 Interactive: The Kernel-Rootkit Hunt

## Phase 3: Linux Memory Analysis

## Linux Attack and Memory Analysis: SSH Compromise and Kernel Rootkit

Work a two-stage Linux intrusion in memory on SRV-NGE-MCR-APP01: an SSH brute force into a sudo misconfiguration to root and a credential harvest, then an LKM rootkit that hooks getdents64 to hide a process.

1. MF7.1 Linux Memory Architecture
2. MF7.2 Executing the User-Space Attack
3. MF7.3 Capturing the Compromised Host
4. MF7.4 User-Space Analysis
5. MF7.5 Credential Extraction from Linux Memory
6. MF7.6 The Kernel Rootkit: Loading and Live Capture
7. MF7.7 LKM Rootkit Detection
8. MF7.8 The eBPF Variant
9. MF7.9 Interactive: The Linux Compromise Hunt

## Phase 4: Integration & Capstone

### Memory Timeline Construction and Multi-Source Correlation

Build forensic timelines from your MF2–MF7 memory captures.

1. MF8.1 Memory Timestamps and Reliability
2. MF8.2 Building a Memory-Only Timeline
3. MF8.3 Memory-to-Event-Log Correlation
4. MF8.4 Memory-to-Disk Correlation
5. MF8.5 Memory-to-Network Correlation
6. MF8.6 Multi-Host Correlation
7. MF8.7 The Unified DFIR Timeline
8. MF8.8 Anti-Forensic Timeline Manipulation
9. MF8.9 Interactive: The Timeline Correlation Hunt

### Capstone, Full Attack Chain Investigation

Execute a multi-stage attack chain from the attacker host against NE-FIN-014: initial access via reflective payload, privilege escalation, credential theft, persistence via WMI subscription, and data staging.

1. MF9.1 The Attack Chain Brief
2. MF9.2 Executing the Full Chain
3. MF9.3 Multi-Checkpoint Acquisition
4. MF9.4 Initial Access Analysis
5. MF9.5 Privilege Escalation and Credential Theft Analysis
6. MF9.6 Persistence and Staging Analysis
7. MF9.7 Timeline Reconstruction and Cross-Checkpoint Correlation
8. MF9.8 Report Production
9. MF9.9 Report Critique
10. MF9.10 Interactive: The Independent Investigation

## Phase 0: Course Resources

### Cheatsheets

The lookup layer: every command the course teaches, grouped by what you are trying to establish.

1. Cheatsheet

### Cookbooks

Procedures: what you do, in what order, and what each step must produce.

1. Cookbook

### Lab Setup

Build the environment that produces the evidence this course analyzes.

1. Lab Setup

## Walkthroughs

Complete investigations worked end to end, including the wrong turns.

1. Injected Code in a Trusted Process
2. Credential Theft With No Dump File
3. A PowerShell Command With No File On Disk
4. Persistence That Survived A Reboot
5. A Driver That Is Not In The Driver List
6. Root On A Linux Web Server
7. Building The Timeline That Ties It Together
8. Triage: A Capture You Know Nothing About
9. The Capture That Went Wrong
10. The Negative Result

## Playbooks

Detonation procedures. Each produces the artifact a module teaches you to find, so you analyze evidence whose ground truth you already know.

1. Reflective DLL Injection via Meterpreter
2. Mimikatz Credential Theft
3. PowerShell Reflective Loader
4. Scheduled Task and WMI Event Subscription
5. Kernel Driver with Process-Creation Callback
6. SSH Compromise and LKM Rootkit
7. The Full Attack Chain

## Playground

Resources for practicing memory forensics and using it at work: an open-source DFIR toolkit, the forensic lab, the detection library, response playbooks, and guided investigations.

## References

External sources this course draws on: tooling documentation, research, standards and community memory forensics material.

## Course Completion

### Course Exam

Endpoint Memory Forensic Investigation end-of-course exam: a three-phase simulation testing whether you can apply the method to a case the course did not walk through.

1. Course Completion. Endpoint Memory Forensic Investigation

Generated 2026-09-01 from the published course. The current syllabus is always at [ridgelinecyber.com/training/courses/applied-memory-forensics/](https://ridgelinecyber.com/training/courses/applied-memory-forensics/)