AI-Assisted Malware Analysis for Defenders

FOR404

Malware analysis for defenders with an assistant connected to the tools: delivery chains, loaders, infostealers, session theft, RAT configuration, fileless execution, malicious packages and model files, and samples written to steer the assistant reading them. Every claim is checked against tool output on inert specimens, confirmed in code where a verdict rests on it, and turned into detections measured against a clean corpus.

Modules	21
Lessons	149
Phases	5
Free preview	Malware Analysis Foundations (10 lessons)

The method

Every lesson works a specimen built for it: inert, with a known answer, and a clean look-alike beside it in the corpus. The loop is the same in every module.

1. Run the tools on the specimen. Every output on the page is a real run in REMnux, and you can reproduce it from the lab pack.

2. Ask the assistant, then check it. Each lesson shows the prompt that gets a checkable answer, the session in which the assistant proposes each step, and where it needed steering. Thirteen lessons carry real captured sessions beside the prepared ones.

3. Confirm what a verdict rests on, in code. One claim, the least code that settles it, through Ghidra over MCP or by hand.

4. Respond to what was taken. Which stolen sessions and secrets still work, and in what order to shut them off.

5. Detect it, and measure the detection. Rules from structure rather than strings alone, scored against the specimens and the clean corpus, then hunted across a fleet.

The capstone works one case from the alert to a graded report, and the exam is a queue of files no lesson uses.

What this course assumes

Working knowledge of Windows internals, the command line and one log query language. The foundation module covers how malware reaches a host now, packing, configurations and how an assistant reaches the tools before any module relies on them.

An assistant is optional. Any assistant that can reach a shell works, free or paid, and every lesson can be worked by hand with the same commands.

What this course does not cover: full reverse engineering and debugging (unpacking by hand, getting past anti-debugging), malware development, running live samples, memory forensics beyond what a verdict needs, which is FOR502 Memory Forensics, and mobile malware. The specimens are simpler than most

real samples, and lessons 2.1 to 2.3 say what a real one adds, from published research.

Full syllabus

Phase 1: Foundations, Delivery and Loaders

Malware Analysis Foundations FREE PREVIEW

Module 0 of AI-Assisted Malware Analysis for Defenders: how malware reaches a host now, loaders and packing, what infostealers take and the sessions they open, command and control, execution without a file, the software and model supply chain, analysis with an assistant connected to the tools...

1. 0.1 How Malware Reaches a Host Now
2. 0.2 Loaders, Packers and Staged Payloads
3. 0.3 What an Infostealer Takes
4. 0.4 What Stolen Sessions and Tokens Still Open
5. 0.5 Command and Control, and the Configuration That Drives It
6. 0.6 Execution Without a File
7. 0.7 The Software and Model Supply Chain
8. 0.8 Analysis With an Assistant Connected to the Tools
9. 0.9 Reading Code to Answer One Question
10. 0.10 From a Sample to a Detection

Delivery Chains

Module 1 of AI-Assisted Malware Analysis for Defenders: HTML and SVG smuggling, shortcut files, disk image containers, MSI and MSIX installers, paste-to-run lures, nested and password-protected archives, OneNote and other containers outside Office, and a chain mapped to the controls that break it.

1. 1.1 HTML Smuggling and the Blob Download
2. 1.2 LNK Files: Arguments, Targets and Icon Tricks
3. 1.3 ISO, IMG and VHD Containers and Mark-of-the-Web
4. 1.4 MSIX and MSI Installers as a First Stage
5. 1.5 Paste-to-Run Lures and Fake CAPTCHA Pages
6. 1.6 Nested, Password-Protected and Polyglot Archives
7. 1.7 OneNote, SVG and Other Containers Outside Office
8. 1.8 Mapping a Chain to the Controls That Break It

Loaders and Staged Payloads

Module 2 of AI-Assisted Malware Analysis for Defenders: entropy and imports read honestly, .NET, Go and Rust loaders, DLL sideloading, staged downloads under a real cipher, shellcode loaders, and what links one loader to another.

1. 2.1 Packed or Encrypted: Reading Entropy, Sections and Imports Honestly
2. 2.2 .NET Loaders: Resources, Reflection and Assembly.Load
3. 2.3 Go Loaders: Recovering Symbols and Strings
4. 2.4 Rust Loaders and What the Toolchain Leaves Behind
5. 2.5 DLL Sideloading and Abused Signed Hosts
6. 2.6 Staged Downloads: URLs, Keys and the Second Stage
7. 2.7 Shellcode Loaders and the In-Memory Handoff
8. 2.8 Loader Turnover and What Attribution Can Bear

Phase 2: What Malware Takes

Infostealers

Module 3 of AI-Assisted Malware Analysis for Defenders: what infostealers collect, browser credential and cookie stores, App-Bound Encryption, tokens and wallets, exfiltration channels, configurations, host evidence, and scoping what one infection took.

1. 3.1 What a Stealer Collects, Store by Store
2. 3.2 Browser Credential and Cookie Stores
3. 3.3 App-Bound Encryption and How Stealers Get Around It
4. 3.4 Session Tokens, Wallets and Desktop App Secrets
5. 3.5 Staging and Exfiltration: Archives, Telegram, HTTP and Paste Sites
6. 3.6 Reading a Stealer's Configuration and Target List
7. 3.7 Host Evidence of a Stealer Run
8. 3.8 Scoping What One Infection Took

Session-Theft Malware

Module 4 of AI-Assisted Malware Analysis for Defenders: malicious browser extensions and their scripts, .NET token grabbers, cookie dumpers that drive the browser, adversary-in-the-middle phishing kits, re-theft and persistence, indicator extraction and rules for the family.

1. 4.1 Malicious Browser Extensions: Manifests and Permissions
2. 4.2 Deobfuscating an Extension's JavaScript
3. 4.3 Token Grabbers in .NET
4. 4.4 Cookie Dumpers That Drive the Browser
5. 4.5 Adversary-in-the-Middle Phishing Kits
6. 4.6 Re-Theft and Persistence
7. 4.7 Pulling the Operator's Indicators Out
8. 4.8 Rules for a Session-Theft Family

RAT Configuration and C2

Module 5 of AI-Assisted Malware Analysis for Defenders: where remote access tools keep their configuration, the public parsers that read it and where they stop, finding a decryption routine with MCP, command channels in captures, build artifacts, indicators and Suricata rules, forks, and...

1. 5.1 How a RAT Stores Its Configuration
2. 5.2 Extracting Configuration With CAPE-Parsers and configextractor
3. 5.3 When No Parser Exists: Finding the Decryption With MCP
4. 5.4 Command Channels: HTTP, Raw TCP, WebSockets and Legitimate Services
5. 5.5 Mutexes, Campaign IDs and Build Artifacts
6. 5.6 Turning a Configuration Into Network and Host Indicators
7. 5.7 Open-Source RAT Forks and Why Family Names Mislead
8. 5.8 Checking an Assistant's Extracted Configuration

Phase 3: Evasive and New Formats

Fileless Execution and Living Off the Land

Module 6 of AI-Assisted Malware Analysis for Defenders: layered PowerShell peeled safely, script block logs as evidence, .NET loaded from memory, LOLBin chains, persistence that stores code, script-host droppers, one fileless run rebuilt from its logs, and rules measured on all of it.

1. 6.1 PowerShell Obfuscation, Layer by Layer
2. 6.2 AMSI and Script Block Logging as Deobfuscation Evidence
3. 6.3 In-Memory .NET Execution
4. 6.4 LOLBins in a Malware Chain: Download, Execute, Proxy
5. 6.5 Persistence Mechanisms That Carry Encoded Code
6. 6.6 JavaScript, VBScript and HTA Through Script Hosts
7. 6.7 Reconstructing a Fileless Sample's Execution From Its Traces
8. 6.8 What a Fileless Chain Leaves for Detection

Malicious Packages and CI Compromise

Module 7 of AI-Assisted Malware Analysis for Defenders: how npm and PyPI packages run code, the names that get them installed, triage with GuardDog and by hand, worms and stolen publishing tokens, CI pipelines as the target, scoping a bad version across an organization, and the controls measured.

1. 7.1 How Packages Execute: Install Scripts, Imports and Runtime Triggers
2. 7.2 Typosquats, Dependency Confusion and Abandoned Names
3. 7.3 Triaging an npm Package With GuardDog and by Hand
4. 7.4 Triaging a PyPI Package and Its Wheel
5. 7.5 Package Worms and Stolen Publishing Tokens
6. 7.6 CI Pipelines as the Target: Secrets, Runners and Actions
7. 7.7 Scoping a Compromised Dependency Across an Organization
8. 7.8 Controls That Stop the Next One

Model Files and AI Artifacts

Module 8 of AI-Assisted Malware Analysis for Defenders: why loading a model can run code, reading pickle opcodes, what ModelScan, Fickling and picklescan miss, verifying safetensors, malicious models on public hubs, extensions and MCP servers as delivery, notebooks and agent files carrying code...

1. 8.1 Why Loading a Model Can Run Code
2. 8.2 Reading Pickle Opcodes
3. 8.3 ModelScan, Fickling and picklescan, and What They Miss
4. 8.4 Safe Formats and How to Verify One
5. 8.5 Malicious Models on Public Hubs
6. 8.6 AI Tool Packages, Extensions and MCP Servers as Delivery
7. 8.7 Notebooks and Agent Configurations Carrying Code
8. 8.8 A Loading Policy for Model Files

Malware That Targets the Analyst

Module 9 of AI-Assisted Malware Analysis for Defenders: how an assistant reads a sample and where it can be steered, instructions hidden in strings and resources, misleading metadata and copied signatures, anti-analysis aimed at sandboxes and tools, measuring how often an assistant is misled...

1. 9.1 How an Assistant Reads a Sample, and Where It Can Be Steered
2. 9.2 Instructions Hidden in Strings, Resources and Comments
3. 9.3 Misleading Metadata, Fake Signatures and Decoy Code
4. 9.4 Anti-Analysis Aimed at Sandboxes and Tools
5. 9.5 Measuring How Often Your Assistant Is Misled
6. 9.6 Keeping Tool Output and Instructions Apart
7. 9.7 Hosted AI Verdicts as Claims to Check
8. 9.8 A Verification Discipline That Survives Adversarial Samples

Phase 4: Proof and Detection

Confirming Capability in Code

Module 10 of AI-Assisted Malware Analysis for Defenders: choosing the one claim worth confirming, Ghidra through MCP and what the assistant can see, finding the function behind a behavior, confirming a decryption key, a configuration parser and a capa capability, catching the decompiler being...

1. 10.1 Choosing the One Claim Worth Confirming
2. 10.2 Ghidra Through MCP: What the Assistant Can and Cannot See
3. 10.3 Finding the Function Behind a Behavior
4. 10.4 Confirming a Decryption Routine and Its Key
5. 10.5 Confirming a Configuration Parser
6. 10.6 Confirming a Capability capa Reported
7. 10.7 When the Decompiler Is Wrong
8. 10.8 Stopping: How Much Code a Verdict Needs

Detection and Hunting From Analysis

Module 11 of AI-Assisted Malware Analysis for Defenders: choosing indicators that last, YARA-X rules from structure, measuring rules against a clean corpus and for speed, comparing with YARA-Forge, behavior detections with Sigma, hunting a family across a fleet with Velociraptor, and maintaining...

1. 11.1 Choosing What to Detect: How Long Each Indicator Lasts
2. 11.2 YARA-X Rules From Structure, Not Strings Alone
3. 11.3 Measuring Rules Against a Clean Corpus
4. 11.4 YARA-X Performance at Fleet Scale
5. 11.5 Comparing With YARA-Forge Rules
6. 11.6 Behavior Detections From a Sample's Analysis
7. 11.7 Hunting a Sample's Family Across a Fleet With Velociraptor
8. 11.8 Retiring and Updating Rules as Families Change

Phase 5: Capstone

Capstone

Module 12 of AI-Assisted Malware Analysis for Defenders: one investigation from an endpoint alert to the report, through the delivery chain, the loader, the stealer and its haul, the sessions and token it exposed, the package published with that token, detections, a fleet hunt and a graded...

1. 12.1 The Alert and the File
2. 12.2 The Delivery Chain
3. 12.3 The Loader and Its Payload
4. 12.4 What Was Stolen
5. 12.5 The Sessions Still Open
6. 12.6 The Package Behind It
7. 12.7 Detections and the Fleet Hunt
8. 12.8 The Report

Phase 0: Course Resources

Cheatsheets

One sheet per subject: the question, the command run on the lab pack, what to read and what it does not prove.

1. First Look at an Unknown File
2. Delivery Containers and Lures
3. Loaders and Staged Payloads
4. Decoding and Decrypting
5. Infostealers
6. Session Theft
7. RAT Configuration and C2
8. PowerShell, Script Hosts and LOLBins
9. Packages and CI
10. Model Files and AI Tooling
11. Samples Aimed at the Analyst
12. Confirming in Code
13. YARA-X Rules and Hunting
14. Lookup Values

Cookbooks
One procedure per kind of work, in runbook form: the steps, the reason for each, what it produces and where it branches.

1. First Look at an Unknown File
2. From a Delivery File to Its Payload
3. Peeling a Staged Loader
4. Extracting a Configuration
5. Scoping a Stealer Infection
6. Closing Stolen Sessions in Order
7. Triaging a Suspect Package
8. From a Sample to a Fleet Hunt
9. Handing a Sample to an Assistant

Lab Setup
Building the REMnux VM, installing and checking the lab pack, connecting an assistant and verifying the lab.

1. The Lab and What It Builds
2. Building the REMnux VM
3. Installing the Lab Pack
4. Connecting an Assistant
5. Settings and Verification

Walkthroughs
Four cases worked end to end on files built for them, including the wrong turn each one invites.

1. A Verdict the Sample Wrote
2. Nothing Happened: a Paste-to-Run Command
3. A Dependency Update That Runs on Require
4. A Fleet Rule That Matched the Vendor

Playbooks
One playbook per thing that fires: the warning, a pre-flight, the steps with their commands, three outcomes and the record.

1. Encoded PowerShell From a Shortcut or Script Host
2. A Fleet YARA Hit
3. A Confirmed Stealer Infection
4. A Malicious Dependency Advisory
5. A Model File Flagged or About to Load
6. An Unfamiliar Signed Binary
7. An Extension Installed by Policy
8. An Outside Verdict on a File

Practice Playground
The practice scenarios by module, and the lab pack as free practice.

1. Practice Scenarios and the Lab Pack

References and Reading
Every source the lessons cite, by module.

1. References by Module

Course Completion

Course Exam
AI-Assisted Malware Analysis for Defenders end-of-course exam: twelve tasks, one per module, each answered from a queue of files in the lab pack that no lesson uses.

1. Course Completion. AI-Assisted Malware Analysis for Defenders

Generated 2026-11-24 from the published course. The current syllabus is always at ridgelinecyber.com/training/courses/ai-assisted-malware-analysis/