

Lab Guide

Four exercises: sandbox report review, hash / reputation analysis, strings triage, and YARA rule authoring — classify suspicious files with AI assistance.

Authorized Use Only

All samples are **synthetic reports, not real malware**. Never download or execute live samples outside a dedicated isolated sandbox.

Prerequisites

- Completed the [Theory](#) reading
- Access to an AI assistant (Claude, ChatGPT, or local LLM)
- Downloaded all four files into a folder named `malware-lab/`
- Familiarity with: PE/ELF basics, common malware families, YARA syntax

Lab Scenario

A user reported a suspicious attachment. The SOC pushed the file through the sandbox, extracted strings, calculated hashes, and flagged some rules for review. Your job: confirm malicious intent, classify the family, and produce a YARA signature that would detect the next variant.

Download Lab Files

Save all 4 files into a folder named `malware-lab/` before starting the exercises.

↓ [file-hashes.txt](#) ~5 KB

↓ [sandbox-report.txt](#) ~12 KB

↓ [strings-output.txt](#) ~6 KB

↓ [yara-challenge.txt](#) ~9 KB

Exercise 1 — Sandbox Report Review

Open [sandbox-report.txt](#). Cuckoo / ANY.RUN / Hybrid-Analysis reports are dense — you need structure.

Task 1.1 — Triage the behavior

Below is a sandbox detonation report. Extract:

1. File metadata: type, size, compiler, signer, imphash.
2. Static indicators: suspicious imports (CreateRemoteThread, VirtualAllocEx), high entropy sections, packer signatures.
3. Dynamic behavior:
 - File system changes
 - Registry persistence
 - Process injection
 - Network connections (DNS / HTTP / TLS)
 - Scheduled tasks or services
4. MITRE ATT&CK technique mapping for each behavior.
5. Likely family and verdict (clean / PUA / malware).
6. Confidence level and 3 key evidence lines to support it.

[paste sandbox-report.txt]

✓ **Self-check: You should end with a family guess and a list of at least 3 MITRE techniques.**

Exercise 2 — Hash / Reputation Analysis

Open `file-hashes.txt`. Hashes are cheap lookup keys — but also misleading if taken alone.

Task 2.1 — Rank the hashes

Below is a list of file hashes (MD5/SHA1/SHA256) with their observed filenames and paths. For each:

1. Rate reputation risk using heuristics you can see:
 - Path legitimacy (C:\Windows vs. %TEMP% vs. user download)
 - Name/extension mismatch (invoice.pdf.exe)
 - Known-good vs. known-bad filename patterns
 - Imphash duplication suggesting code reuse
2. Which hash would you submit to VirusTotal first?
3. Identify pivot opportunities: shared imphash, ssdeep neighbors, filename reuse across hosts.
4. Build a triage queue: [immediate / suspicious / likely benign].

[paste file-hashes.txt]

✓ **Self-check: You should identify at least one file that requires immediate attention based on path + name alone.**

Exercise 3 — Strings Intelligence

Open `strings-output.txt`. Strings reveal intent faster than disassembly.

Task 3.1 — Pull the IOCs

Below is strings output. Extract:

1. URLs, IPs, domains (potential C2).
2. Registry keys (persistence).
3. File paths (drops, staging).
4. Command-line fragments (cmd.exe, powershell, schtasks).
5. Error messages / log strings (hint at the language / framework).
6. Compiler / build banners (attribution / family fingerprint).
7. Any base64 or hex-encoded blobs — decode and re-analyze.

8. Imported library names and suspicious API combinations.

Output: an IOC list ready to paste into a threat-intel platform (hash, domain, IP, path, rule).

[paste strings-output.txt]

✓ Self-check: Every IOC should have a type label and a confidence note.

Exercise 4 — YARA Rule Authoring

Open `yara-challenge.txt`. A YARA rule is the crystallized form of your analysis.

Task 4.1 — Build a high-fidelity YARA rule

Below are notes on a malware sample (strings, imports, behavior).

Write a YARA rule that detects this family with:

1. Meta section: author, date, description, hash, reference.
2. Strings: 3-5 unique byte sequences or text that are unlikely to appear in benign binaries.
3. Condition: filesize bounds, PE header check, string counts, entropy hints.
4. A companion "loose" rule for variants (hunting rule).
5. Performance: no unanchored PCRE that would scan slow.
6. Test: verify it does not match common Windows binaries by inspection (e.g., explorer.exe signatures in your mental model).

Output: the two YARA rules and a short paragraph explaining why each string was chosen.

[paste yara-challenge.txt]

✓ Self-check: Your rule should have at least one string that is clearly attacker-specific — not a common library function name.

Final Exercise — Classification Report

Task 5.1 — Single-file verdict

Combine the four analyses (sandbox, hashes, strings, YARA) into a one-page classification report:

1. Verdict: malicious / suspicious / benign (with confidence %)
2. Family (or "unknown, similarities to X")
3. Capabilities observed
4. Kill-chain stage coverage
5. IOCs (hashes, domains, IPs, paths)
6. Detection: the YARA rule + Sigma rule
7. Recommended response: quarantine / scan fleet / reimage

Format as markdown.

✓ **Self-check: A SOC manager should be able to action your report without reading the source artifacts.**

What's Next

Lesson 04 moves from reactive analysis to **proactive hardening** — SSH, Linux, Windows, and Docker baselines.