REV 2 — 2026
LARS BLOMGAARD
www.defencia.dk

Steps for Analysing Malware

Digital Forensics & Incident Response — Malware Analysis Procedure

! GOLDEN RULE ALWAYS work on a copy — NEVER on the original sample. Verify hash before and after every transfer. Never upload samples to external services without explicit written authorisation.

This action card covers the four phases of malware analysis:

01 Case Preparation & Sample Documentation

02 Static Analysis

03 Dynamic Analysis

04 Reverse Engineering Analysis

1 — Case Preparation & Sample Documentation

1.1 Sample Metadata

Document ALL of the following before any analysis begins:

Sample origin	Where was the sample collected? (Time, date, system name, user, department / division)
Hash values	MD5, SHA-1, SHA-256 — calculate on the ORIGINAL before any processing. SHA-256 is the primary value.
File metadata	Filename, file size, file type (magic bytes), timestamps (Created / Modified / Accessed)
Analyst responsible	Full name, role, contact. Who carried out the analysis?
Analysis constraints	Legal restrictions, time limitations, classification level, scope boundaries
Case reference	Ticket number, case ID, or IR reference for traceability

! DO NOT UPLOAD Do NOT upload samples to VirusTotal, ANY.RUN, Hybrid Analysis, or any external service UNLESS explicitly approved in writing by the case owner. Samples may contain proprietary data or trigger adversary notifications.

1.2 Malware Lab / Sandbox Setup

Document the full environment used for analysis:

- Network setup
  - Default: ALWAYS offline unless controlled network is required
  - If online: use isolated VLAN or dedicated hardware (e.g. GL-iNet router with VPN)
  - Were VMs configured to communicate with each other for traffic capture (INetSim, FakeNet-NG)?
  - Document all network rules and capture setup (Wireshark / tcpdump)

- Virtualisation environment
  - Platform and version: VMware Workstation / Fusion, VirtualBox, Proxmox, ESXi
  - Snapshot taken BEFORE execution — revert after each run
  - Anti-VM evasion: consider bare-metal analysis if VM evasion is suspected

- Virtual machines used
  - Windows: FlareVM or Remnux-on-Windows for Windows malware analysis
  - Linux: REMnux for Linux malware, script analysis, and network simulation
  - macOS: macOS VM for Apple-targeted malware (if applicable)
  - Other: document OS version, patch level, and installed software

- Search methods for the sample
  - IOC match (IP, domain, file hash, registry key)
  - YARA rules — note rule name and source
  - Vendor file signature / AV detection name
  - Hash search (VirusTotal, MalwareBazaar, MWDB)

- Online cloud sandboxes (if approved)
  - Service used and link to result (e.g. app.any.run/tasks/..., hybrid-analysis.com/...)
  - What services / detections were provided?
  - Constraints: was the sample run for the full duration (5–60 min)?
  - Was the decision to use cloud sandbox actively and explicitly approved?
  - Was the sandbox configured to simulate user interaction?

2 — Static Analysis

Static analysis examines the file without executing it. Always perform static analysis before dynamic analysis.

2.1 File Identification & Fingerprinting

- File type identification (magic bytes, not just extension)
  - Use: file, TrID, DIE (Detect It Easy)
  - Hash fingerprinting: MD5, SHA-1, SHA-256, SSDeep (fuzzy hash), TLSH
  - IMPHASH, AUTHENTIHASH (PE files)
  - Search hashes against: VirusTotal, MalwareBazaar, MWDB, AlienVault OTX

2.2 Antivirus & Signature Scanning

- Run local AV scan (Windows Defender, ClamAV, YARA)
- Document detection name, engine version, and signature date
- Note: AV detection alone is NOT sufficient analysis
- Check for known packer signatures (UPX, MPRESS, custom packers)

2.3 String Extraction & Artefact Analysis

- Extract strings: floss, strings, FLARE-FLOSS for obfuscated strings
- Look for: URLs, IPs, domain names, registry keys, file paths, commands, credentials
- Identify compiler artefacts, debug paths, PDB paths
- Extract embedded resources (icons, manifests, version info, certificates)

2.4 File Format & Structure Analysis

- PE header analysis (Windows executables)
  - Tools: PEview, PE-bear, CFF Explorer, pestudio
  - Sections: .text, .data, .rsrc — note anomalies (high entropy = packed/encrypted)
  - Import Address Table (IAT): suspicious API calls (VirtualAlloc, CreateRemoteThread, etc.)
- Check code signing certificate (valid? revoked? stolen?)
- For scripts / documents: olevba, oledump, mraptor (Office), pdfid / pdf-parser (PDF)

2.5 Packer & Obfuscation Detection

- Tools: DIE (Detect It Easy), ExeinfoPE, Unpacker plugins
- Note: unpacking may be required before further static analysis
- Document entropy per section (> 7.0 suggests packing or encryption)
- Check for anti-analysis techniques: anti-debug, anti-VM, sleep timers

2.6 Disassembly (Light)

- Quick disassembly review without full RE: IDA Free, Binary Ninja, Cutter (Rizin)
- Identify entry point and main function
- Note obvious suspicious patterns: shellcode, self-modification, reflective loading
- Document tools used and their versions

2.7 Static Analysis Toolbox

PEStudio All-in-one PE static analyser	**DIE** Packer / compiler identification	**FLARE-FLOSS** Obfuscated string extraction
PE-bear PE structure deep dive	**CFF Explorer** PE header and resource editor	**olevba / oledump** Office document macro analysis
pdfid / pdf-parser PDF structure and object analysis	**Cutter / Rizin** Open-source disassembly & RE	**YARA** Pattern matching and rule scanning

3 — Dynamic Analysis

Dynamic analysis executes the sample in a controlled environment. The approach varies — pre-define exactly what actions will be taken before execution.

BEFORE EXEC	Take a clean snapshot before every execution run. Revert after each run. Never analyse on a production system or a system connected to production networks.

3.1 Analysis Environment

Operating system	Which OS version? (e.g. Windows 10 22H2, Windows 11 23H2, Ubuntu 22.04)
Configuration	How was it configured? Patched or intentionally unpatched? AV enabled or disabled?
Network setup	Offline / INetSim / FakeNet-NG / live internet (if approved)
Virtual or hardware	Note: some malware detects VMs and alters behaviour. Consider bare-metal if evasion suspected.
Lab type	Freshly installed (clean) vs. legacy lab (simulated user activity: docs, browser history, media files, recent files)

3.2 Runtime Behaviour Monitoring

- Process monitoring
  - Tools: Process Hacker 2 / System Informer, Process Monitor (ProcMon), Process Explorer
  - What processes were spawned? Parent/child relationships?
  - What files were created, modified, or deleted?
  - What registry keys were written or read?

- Network activity
  - Tools: Wireshark, tcpdump, FakeNet-NG, INetSim, NetworkMiner
  - DNS queries made? What domains / IPs were contacted?
  - C2 communication patterns: beaconing interval, protocol (HTTP/S, DNS, custom)?
  - Data exfiltration indicators: volume, destination, encoding
  - TLS certificate details of contacted servers

- Registry monitoring
  - Tools: Regshot (before/after snapshot diff), ProcMon
  - Run/RunOnce keys — persistence via autorun
  - Service installation keys
  - WMI subscriptions (check: wmic subscription list)

3.3 Persistence Mechanisms

- Scheduled tasks: schtasks, Task Scheduler XML
- Service installation: new service entries, driver loading
- Registry autorun keys: HKCU/HKLM Run, RunOnce, AppInit_DLLs
- Hidden sectors / MBR / bootkit infection
- COM hijacking, DLL sideloading, DLL search order hijacking
- WMI event subscriptions
- Startup folder and shortcut manipulation
- Contact to Command and Control (C2): document all external IPs and domains

3.4 Dynamic Analysis Toolbox

FlareVM Windows malware analysis distro	**REMnux** Linux malware analysis distro	**Process Hacker 2** Deep process & memory inspection
ProcMon File, registry, process events	**Regshot** Registry diff (before/after)	**FakeNet-NG** Network simulation & capture
INetSim Internet service simulation	**Wireshark** Full packet capture & analysis	**x64dbg / x32dbg** Interactive debugger (Windows)

4 — Reverse Engineering Analysis

RE is performed based on findings from static and dynamic analysis. Scope and depth depend on the purpose of the analysis — define this explicitly before starting.

4.1 Purpose & Scope Definition

Purpose of analysis	Insider threat, legal case, nation-state actor, espionage, ransomware, fraud, other?
Time constraints	Max hours allocated for RE. Document decision to stop and reasoning.
Deliverables expected	Technical report, IOCs, YARA rules, MITRE ATT&CK mapping, court-ready evidence?
Tools & versions	IDA Pro, IDA Free, Ghidra, Binary Ninja, Cutter (Rizin). Document version exactly.

4.2 Code Analysis

- Function mapping: identify main(), WinMain(), DllMain() and key functions
- Call graph analysis: what does the malware call and in what order?
- Return value analysis: what does the code return and how are errors handled?
- Identify cryptographic routines: custom XOR loops, known cipher constants (AES S-box, RC4)
- Identify network communication functions: socket, WSAStartup, connect, send, recv
- Identify anti-analysis techniques
  - IsDebuggerPresent, CheckRemoteDebuggerPresent, RDTSC timing checks
  - CPUID checks for VM detection
  - Sleep / NtDelayExecution for sandbox timeout evasion

4.3 Memory Analysis

- Memory calls: VirtualAlloc, VirtualProtect, WriteProcessMemory, NtAllocateVirtualMemory
- Process injection techniques: classic injection, process hollowing, reflective DLL, APC injection
- Heap spray indicators
- Identify possible memory tampering or hooking (SSDT hooks, IAT hooks, inline hooks)
- Use Volatility 3 or Rekall if memory dump is available

4.4 Language, Compiler & Attribution Artefacts

- Programming language identified (C/C++, Delphi, Go, Rust, .NET, Python compiled, AutoIt)
- Compiler version and build environment (MSVC, MinGW, GCC)
- PDB path present? (may reveal developer path structure or username)
- Comments or strings in non-English languages? (possible attribution indicator)
- Compile timestamp (note: can be forged)
- Code reuse: match against known malware families (use BinDiff, Lumina, JARVIS)

4.5 Obfuscation & Evasion Analysis

- String obfuscation: XOR encoding, base64, custom encoding routines
- Control flow obfuscation: opaque predicates, junk code, dispatcher patterns
- Just-In-Time (JIT) compilation or dynamic code generation
- Encryption / decryption routines: document key material and algorithm
- Living-off-the-Land (LotL): abuse of legitimate binaries (LOLBins)
- Fileless techniques: script-based execution, in-memory only execution

4.6 RE Toolbox

Ghidra NSA open-source RE framework (free)	**IDA Pro / Free** Industry standard disassembler & RE	**Binary Ninja** Modern RE platform with API
x64dbg Windows debugger for RE	**dnSpy / ILSpy** .NET assembly decompiler	**de4dot** .NET obfuscation remover
BinDiff Binary diffing for code reuse detection	**FLARE-FLOSS** Obfuscated string extraction	**Cutter (Rizin)** Open-source cross-platform RE

5 — Reporting & MITRE ATT&CK Mapping

5.1 IOC Extraction

- File hashes (MD5, SHA-1, SHA-256, SSDeep)
- IP addresses and domains (C2, drop zones, update servers)
- URLs and URI patterns
- Registry keys created or modified
- File paths and filenames
- Mutex names
- Network signatures (JA3, JA3S, JARM hashes for TLS fingerprinting)
- YARA rules: write and test rules based on unique strings or code patterns

5.2 MITRE ATT&CK Mapping

Map all observed behaviours to MITRE ATT&CK Enterprise (https://attack.mitre.org). Document Tactic, Technique, and Sub-technique IDs.

Tactic	e.g. Execution, Persistence, Defence Evasion, C2, Exfiltration
Technique ID	e.g. T1059.001 — Command and Scripting Interpreter: PowerShell
Evidence	Screenshot or log reference supporting the mapping
Confidence	High / Medium / Low — how certain are we of this mapping?

5.3 Report Checklist

- Executive summary (non-technical, 1 page max)
- Sample metadata and hash values
- Analysis environment description
- Static analysis findings
- Dynamic analysis findings
- RE findings (if performed)
- Complete IOC list (in machine-readable format: JSON, CSV, STIX/TAXII)
- MITRE ATT&CK mapping table
- Recommendations for remediation and detection
- Chain of custody documentation
- All evidence hashed and stored — original sample untouched

Document every step. If it is not documented, it did not happen. Malware analysis without documentation is not forensics — it is guesswork.