Reverse Engineering — Toolset Reference

Zakariyae Belagraa

November 13, 2025

Contents

1 Introduction

This document lists tools commonly used across reverse engineering domains: binary analysis (ELF/PE/Mach-O), dynamic debugging, firmware and embedded reversing, mobile apps (Android/iOS), cryptanalysis, and supporting utilities. Each listing gives a short description and typical use-cases.

2 Essentials (OS, virtualization, general)

- **Linux (Ubuntu, Kali)** — primary host for toolchains and scripting.
- **Windows** — target platform for many PE binaries and Windows-specific debuggers.
- **VirtualBox / VMware / QEMU** — snapshots and isolated analysis environments.
- **Docker** — reproducible analysis environments and build isolation.
- **Snapshots / Checkpointing** — essential for safe dynamic analysis.

3 Static analysis: Disassemblers and decompilers

- **Ghidra** — open-source disassembler + decompiler. Multi-architecture support and scripting (Java/Python).
- **IDA Pro (Hex-Rays)** — industry standard disassembler and decompiler (commercial).
- **Binary Ninja** — interactive GUI, API for plugins, good scripting support.
- **radare2 / Cutter** — powerful CLI framework (radare2) and GUI (Cutter).
- **Hopper** — lightweight disassembler/decompiler (macOS/Linux/Windows).
- **JADX** — decompiler for Android APK (Java/Kotlin) to Java source.
- **JEB** — commercial Android/iOS decompiler (if available).

4 Command-line ELF/PE/Mach-O utilities

- **objdump, readelf, nm, strings, file** — basic binary introspection (ELF/PE/Mach-O where supported).
- **ldd, ltrace, strace** — runtime dependency and syscall tracing on Unix.
- **dumpbin / dependency walker** — Windows binary inspection.
- **PE-bear / CFF Explorer / PE-sieve** — PE format inspection and lightweight analysis.
- **otool, class-dump** — Mach-O and Objective-C introspection on macOS/iOS.

5 Dynamic analysis: Debuggers and runtime tooling

- **GDB / GEF / pwndbg / peda** — core Unix debugger with powerful plugins for exploit-oriented reversing.
- **x64dbg** — modern Windows debugger with GUI and scripting support.
- **WinDbg / WinDbg Preview** — low-level Windows kernel and usermode debugging.
- **LLDB** — debugger for macOS/iOS and LLVM-based toolchains.
- **Frida** — dynamic instrumentation toolkit (scripting with JavaScript/Python) for hooking functions at runtime (Windows/Linux/macOS/Android/iOS).
- **Pin / DynamoRIO** — binary instrumentation frameworks for dynamic analysis and tracing.
- **Strace / LTrace** — syscall and library call tracing on Linux.

6 Emulation and sandboxing

- **Unicorn Engine** — lightweight CPU emulator library (useful for scripted emulation of small code snippets).
- **QEMU-user / QEMU-system** — full-system or user-mode emulation, helpful for foreign-arch binaries.
- **Bochs** — x86 emulator for low-level inspection.
- **Cuckoo Sandbox / in-house sandboxes** — automated dynamic malware behavior analysis.

7 Binary instrumentation, fuzzing, and symbolic execution

- **AFL++ / libFuzzer** — coverage-guided fuzzing for discovering crashes and behavior.
- **S2E / Triton / Angr** — symbolic execution frameworks for automated path exploration and constraint solving.
- **Frida / Pin / DynamoRIO** — runtime instrumentation for tracing and modifying program behavior.

8 Scripting, APIs and libraries

- **Python 3** — primary scripting language for tool integration and automation.
- **pwntools** — exploitation and binary interaction helper library (CTF-focused).
- **capstone (disassembly), keystone (assembly), unicorn (emulation)** — low-level binary tooling libraries.
- **r2pipe, ghidra-python, ida-python** — script interfaces to radare2, Ghidra, and IDA.
- **binwalk, pyelftools, pefile** — file format parsing and firmware helpers.

9 Mobile reversing

Android

- **APKTool** — decode resources and rebuild APKs (smali level).
- **JADX / CFR / FernFlower** — Java/Kotlin decompilers to recover source.
- **adb (Android Debug Bridge)** — install, forward ports, and remote-debug devices/emulators.
- **Android Studio** — emulator, platform SDK, and debugging tools.
- **Frida / Objection** — runtime instrumentation and tampering of Android apps.

iOS

- **class-dump, otool** — Objective-C runtime inspection.
- **Hopper / IDA / Ghidra** — disassembly and decompilation for Mach-O binaries.
- **Frida / cycript** — runtime introspection and hooking on jailbroken devices.

10 Firmware and embedded

- **binwalk** — firmware extraction, carving, and basic analysis.
- **firmware-mod-kit** — utilities for common firmware tasks.
- **radare2 / Ghidra** — multi-architecture support for embedded binaries.
- **OpenOCD, JTAG adapters (Segger J-Link, FTDI)** — hardware debugging and flashing.
- **Qiling framework** — emulation framework for firmware analysis.

11 Cryptanalysis and encoding helpers

- **CyberChef** — quick transforms, encodings, and small crypto utilities.
- **Hashcat / john** — password cracking and hash analysis.
- **sage / python (cryptography libraries)** — custom crypto analysis and prototyping.

12 Binary diffing and patching

- **Ghidra / IDA / Binary Ninja** — built-in or plugin-based function diffing and patch workflows.
- **diaphora, BinDiff** — binary diffing tools (Diaphora works with IDA/Ghidra; BinDiff is commercial).
- **radare2 patching / rizin** — patching and rewriting binaries from CLI.
- **Hex editors (HxD, bless, 010 Editor)** — raw bytes editing and templates.

13 Network and protocol analysis

- **Wireshark / tcpdump** — packet capture and protocol analysis.
- **mitmproxy / Burp Suite** — HTTP/HTTPS interception and modification for application-layer reversing.
- **scapy** — scripted packet crafting for protocol fuzzing or testing.

14 Packing detection and challenge hardening

Detecting packing, obfuscation and protection

- **Detect It Easy (DIE)** — fast packer/packer-signature detector (supports many formats). Good first-pass to identify common packers and protectors.
- **PEiD / Exeinfo PE / Exeinfo** — legacy packer/signature scanners useful for older Windows binaries.
- **TrID** — file type identification based on signatures; useful when headers are missing or modified.

- **pefile + pe-sieve** — programmatic inspection of PE anomalies and runtime modifications; pe-sieve can detect injected/unpacked code at runtime.
- **strings / rabin2 / binwalk** — quick heuristics: compressed/encrypted sections, odd section names, or high entropy indicate packing.
- **Entropy analysis tools (e.g., binwalk –entropy)** — detect compressed/encrypted segments and packed sections.
- **Unpackers / packer-specific tools (UPX)** — some packers have known removal tools (e.g., `upx -d`).
- **Dynamic unpacking with debuggers or emulators** — GDB/x64dbg + memory dumps to recover in-memory unpacked code for analysis.

15 Recommended minimal toolset (quick start)

1. Linux host (Ubuntu/Kali) + VirtualBox or QEMU.
2. Ghidra and radare2 (Cutter) for static analysis.
3. GDB with pwndbg (or x64dbg for Windows) for dynamic debugging.
4. Frida for runtime hooking and instrumentation.
5. Python 3 + capstone + unicorn + pwntools + r2pipe for scripting.
6. binwalk and Qiling for firmware work; APKTool and JADX for Android.

16 Example workflows

Simple native binary:

1. Enumerate with `file`, `strings`, `readelf/objdump`.
2. Open in Ghidra / IDA for a first pass decompilation.
3. Run under GDB with breakpoints on main and suspicious functions; use pwndbg for convenience.
4. If needed, emulate small routines with Unicorn or instrument with Frida.

Android APK:

1. Unpack with APKTool, inspect resources and manifest.
2. Decompile classes.dex with JADX to understand logic.
3. Run app in emulator or device; use Frida/Objection for function hooking and bypassing checks.

Firmware image:

1. Use binwalk to extract filesystem and components.
2. Identify CPU architecture and load binaries into Ghidra/r2 for analysis.
3. Use Qiling or QEMU-user to emulate components where possible.

17 Further reading and learning resources

- Classic books: "Practical Malware Analysis", "Reversing: Secrets of Reverse Engineering", "The IDA Pro Book".
- Online: project pages and official docs for each tool (Ghidra docs, radare2 book, Frida docs, binwalk README).
- CTF practice: pwnable challenges, reversing categories on popular CTF platforms.

This document is intended as a starting checklist. Customize the toolset to match the target platform (Windows, Linux, Android, embedded) and the goals of your reversing tasks.