

AI-Assisted Software Reverse Engineering

Behavioral Analysis of a Benign File-Manifest Utility | Independent Cybersecurity Lab

Executive Summary

This report documents a repeatable workflow for investigating an unfamiliar utility as a black box. The analyst runs controlled probes, records observable evidence, uses AI to suggest hypotheses and test ideas, and independently verifies claims before documenting them. The target is a small educational Python script that emits a JSON manifest of files under a supplied directory.

Scope and Safety

The work is limited to a benign local utility and included test fixtures. It performs no exploitation, persistence, evasion, or network communication. The project is a self-contained educational simulation, not an assessment of a third-party commercial binary.

Methodology

1. Define scope and establish a controlled sample directory.
2. Probe normal input, missing arguments, a nonexistent path, nested files, hidden and temporary files, and content changes.
3. Capture stdout, stderr, return codes, parsed output paths, and digest comparisons.
4. Use an LLM to generate competing explanations and discriminating tests, labeling model statements as hypotheses.
5. Independently execute the tests and promote only supported claims to findings.
6. Encode the verified behavior in regression tests and a behavioral specification.

Key Findings

ID	Finding	Evidence / status
F-01	Valid directory input emits JSON and returns 0.	Probe + unit test
F-02	Output contains root, file_count, and files; file records include path, size, sha256	Output schema
F-03	Nested sample file is included; tested hidden and .tmp fixtures are excluded	Observed paths + unit test
F-04	Digest changes when file bytes change while filename stays constant	Content-sensitivity probe
F-05	Missing argument returns 2 with usage on stderr; missing directory returns 1	Error-path probes + tests

AI-Assisted Analysis: Guardrails

AI output is used to prioritize experiments, not as proof. Prompts request multiple explanations, confidence labels, and a test that could disprove each hypothesis. Analysts should avoid submitting secrets, personal information, or proprietary code to a hosted model without authorization.

Security Interpretation

A file-hash manifest can support integrity comparison when a trusted baseline is available. It does not prove that a file is benign or authentic. If both the files and baseline can be replaced, matching hashes do not establish trust. Operational use requires protecting the baseline separately.

Limitations

The probe set does not exhaustively test permissions, symlinks, special files, large files, race conditions, Unicode normalization, or every operating system. Findings apply to the supplied educational target and tested fixtures only. Further work could add hidden-directory probes, uppercase suffix tests, empty directories, unreadable files, and known SHA-256 test vectors.

Reproduction

```
python mystery_manifest.py sample_data
python probe_target.py
python -m unittest -v
```

Prepared as an independent portfolio lab. The accompanying repository contains the target, probe harness, evidence JSON, AI workflow prompts, findings, and tests.