Automated Binary Morphing and Evasion Analytics

Marc Rivero | @seifreed

https://github.com/seifreed/r2morph

WHO AM I

Marc Rivero López
Cybersecurity Researcher
@seifreed

- 🧠 I've spent more hours inside aaa than in actual sleep cycles.
- 🧩 I believe s + pdf@entry0 is a lifestyle, not a command.
- 💀 I've learned that "just one more afl" can easily become 3 AM.
- 🕵️ I trust Radare2's izz output more than some threat intel feeds.

The problem

The problem

- 🧱 **Binary analysis is mature** — but binary transformation isn't.

  We can analyze any executable… yet safely morphing one is still a pain.

- 🧩 **Manual mutation =** inconsistent and error-prone.

  Hand-crafted patching breaks imports, offsets, and sometimes even logic flow.

- ⚙️ **Lack of automation.**

  There's no standardized way to generate metamorphic or evasive binaries for research, testing, or training.

- 🧠 **Missing reproducibility.**

  The same sample, transformed twice, often produces wildly different — and non-traceable — results.

We needed a framework to morph binaries safely, reproducibly, and at scale — built on the power of Radare2.

What is r2morph?

What is r2morph?

- 🧬 **A metamorphic binary transformation engine** built on **Radare2 and r2pipe.**
- 🎯 **Purpose:** apply code-level mutations that preserve functionality while changing structure.
- ⚙️ **Core idea:** transform executables deterministically — instruction substitutions, block shuffling, opaque predicates, NOP injection, etc.
- 🧠 **Research-oriented:** designed for malware analysts, red-teamers, and binary security researchers.
- 🧩 **Multi-architecture support:** PE, ELF, Mach-O — works wherever Radare2 does.
- 📃 **Output-first design:** generates both the mutated binary and a JSON trace describing every change.

r2morph turns Radare2 from an analysis framework into a transformation framework.

What is r2morph?

Key Features

- **Deep Binary Analysis**: Leverage radare2's powerful analysis engine
- **Metamorphic Transformations**: Apply semantic-preserving code mutations
- **Modular Architecture**: Extensible pipeline-based design
- **Multi-Architecture**: Support for PE/ELF/Mach-O on x86/x64/ARM
- **Plugin System**: Easy-to-create custom mutation passes
- **Rich Analytics**: Detailed statistics and reporting
- **CLI + Python API**: Powerful command-line and programmatic interfaces
- **Validation & Testing**: Automated validation, fuzzing, regression tests
- **Relocation Management**: Code cave finding, reference updates
- **Anti-Detection Analysis**: Evasion scoring, entropy analysis, similarity hashing
- **ARM64/ARM32 Support**: 180+ patterns for ARM architectures
- **Advanced Mutations**: Opaque predicates, dead code, control flow flattening
- **Platform Support**: Code signing for macOS/Windows, format-specific handlers
- **Profile-Guided**: Hot path detection, execution profiling
- **Session Management**: Checkpoints, rollback, versioning
- **Memory-Efficient Mode**: Automatic OOM prevention for large binaries (>50MB)

```
❯ r2morph --help

Usage: r2morph [OPTIONS] [INPUT_FILE] [OUTPUT_FILE] COMMAND [ARGS]...

A metamorphic binary transformation engine based on r2pipe and radare2

╭─ Arguments ──────────────────────────────────────────────────────────────────╮
│   input_file       [INPUT_FILE]   Input binary file                          │
│   output_file      [OUTPUT_FILE]  Output binary file (optional)              │
╰──────────────────────────────────────────────────────────────────────────────╯
╭─ Options ────────────────────────────────────────────────────────────────────╮
│ --input       -i      PATH  Input binary file (alternative style)            │
│ --output      -o      PATH  Output binary file (alternative style)           │
│ --aggressive  -a            Aggressive mode: more mutations, higher probability │
│ --force       -f            Force mutations to be different from original  │
│ --verbose     -v            Enable verbose output                          │
│ --debug       -d            Enable debug output                            │
│ --help                      Show this message and exit.                    │
╰──────────────────────────────────────────────────────────────────────────────╯
╭─ Commands ───────────────────────────────────────────────────────────────────╮
│ analyze     Analyze a binary and display statistics.                         │
│ functions   List functions in a binary.                                      │
│ morph       Apply metamorphic transformations to a binary.                   │
│ version     Display version information.                                     │
╰──────────────────────────────────────────────────────────────────────────────╯
```

KEY FEATURES

KEY FEATURES

- 🧩 **Modular mutation engine** — each transformation is implemented as an independent plugin (e.g., NOP injector, block shuffler, register renamer).
- 🧠 **Semantics-preserving mutations** — changes structure, not behavior.
- ⚙️ **Instruction-level control** — substitute, insert, or reorder assembly instructions using Radare2 analysis.
- 🗒️ **Traceable transformations** — every mutation logged in a structured JSON diff.
- 🧬 **Multi-architecture support** — works with x86, x64, and ARM binaries.
- 🧰 **CLI & API modes** — command-line usage or programmatic integration with r2pipe.
- 🔄 **Deterministic workflows** — reproducible morphing sequences with seed-based randomness.
- 🚀 **Integration-ready** — compatible with Radare2, CI/CD setups, and malware research pipelines.

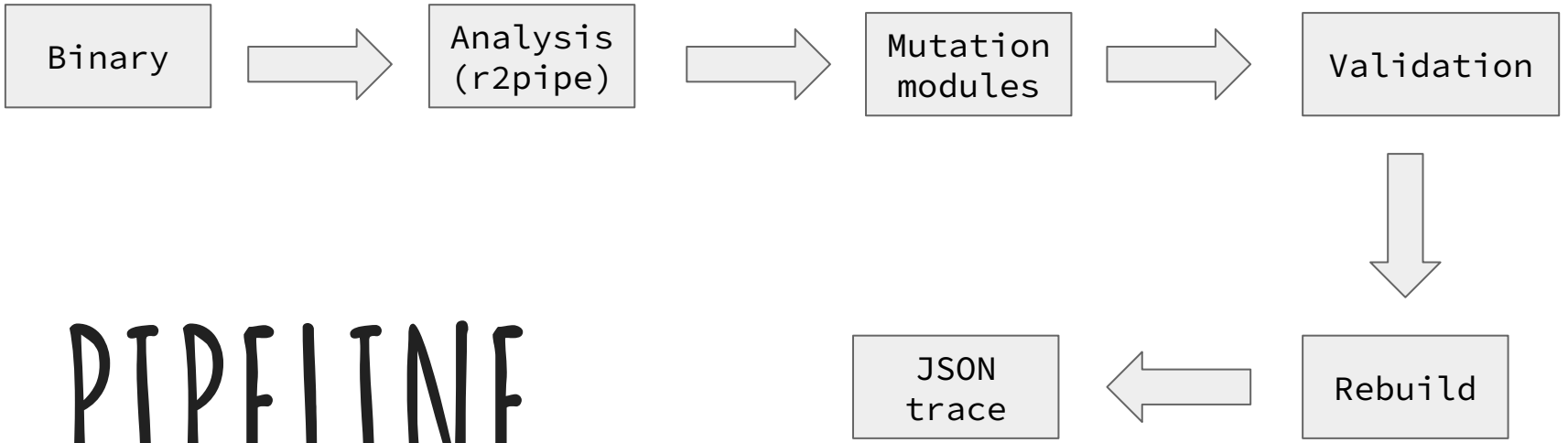
Transform binaries safely — with full visibility and reproducibility.

How does it work internally?

How does it work internally?

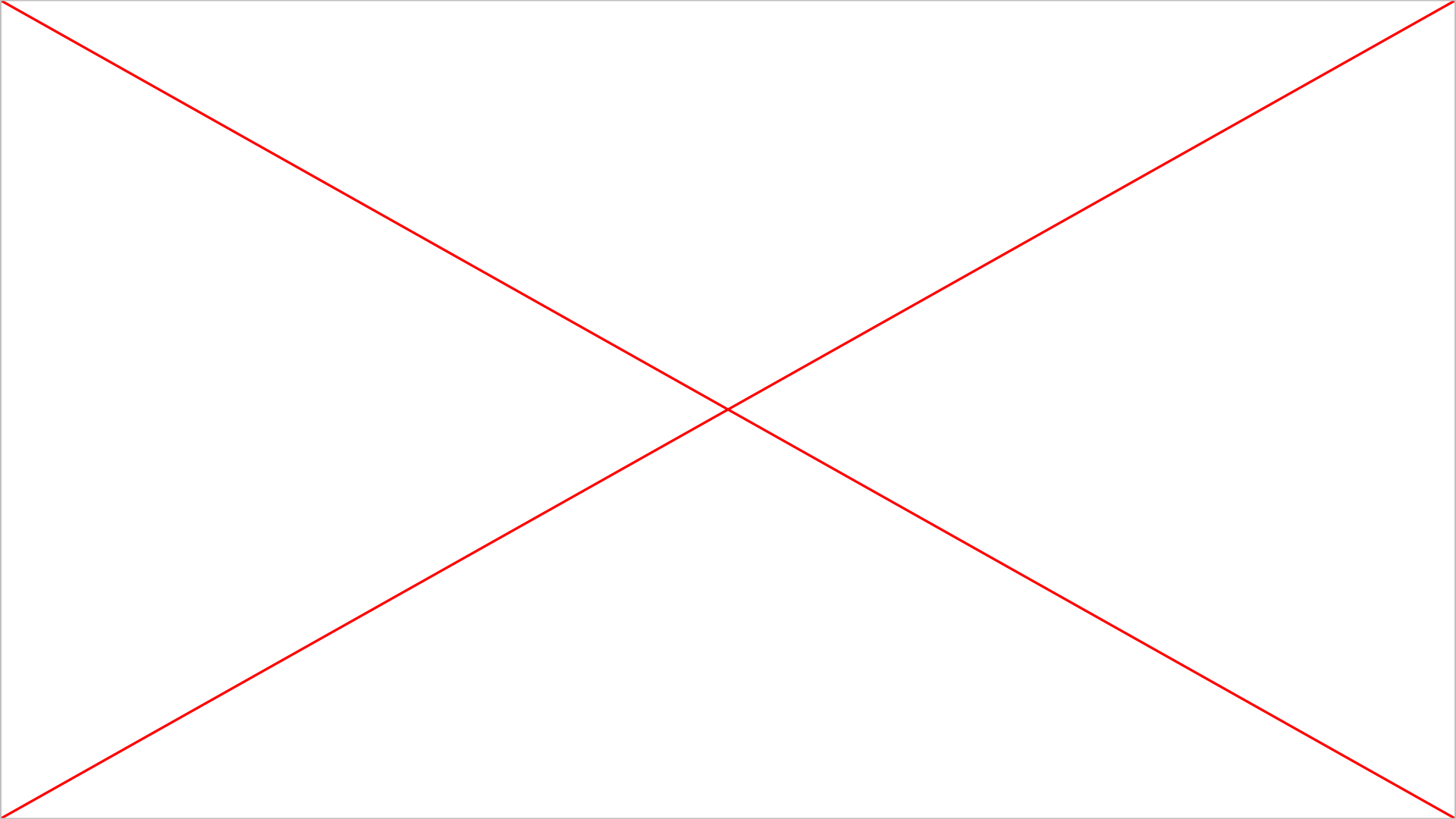
- 🧩 **Load & analyze the binary**
  a. Radare2 opens the file via r2pipe and performs lightweight analysis (aa, afl, pdf).
  b. Metadata, functions, and instruction streams are extracted.
- ⚙️ **Mutation pipeline**
  a. Selected mutation modules (NOP injection, substitution, register swap, etc.) are applied sequentially.
  b. Each module receives an abstracted representation of code blocks and modifies them safely.
- 🧬 **Verification & integrity checks**
  a. Re-analyzes mutated code to ensure instruction validity and control-flow consistency.
  b. Detects broken jumps, bad relocations, or alignment issues before rebuild.
- 🧱 **Rebuild & export**
  a. Generates a new binary (PE, ELF, Mach-O) preserving entry point and original behavior.
  b. Logs every applied transformation in a structured JSON trace for reproducibility.
- 🔁 **Optional scoring**
  a. Compare original vs. morphed sample: entropy delta, instruction diversity, opcode distribution, etc.

How does it work internally?



PIPELINE

DEMO



USE CASES

Use cases

🧬 1. Malware Variant Generation

- Create functionally identical but structurally unique binaries using instruction substitution, NOP insertion, and block reordering.
    a. → One sample → hundreds of morphs, all still functional.

🧩 2. Red Team / AV Evasion Testing

- Morph known payloads with control-flow flattening and opaque predicates to test signature-based detection resilience.
- → Measure how detection drops — safely, reproducibly.

🤖 3. Dataset Augmentation for ML Detectors

- Generate large, diverse binary sets for training machine learning models, without changing program semantics.
    - → More data, same logic — better generalization.

🧱 4. Defensive Research & Similarity Testing

- Evaluate how hash- or feature-based tools (Imphash, TLSH, SSDeep) react to structural code mutations.
    - → Understand what really breaks similarity.

PERFORMANCE & LIMITS

PERFORMANCE & LIMITS

- ⚡ **Optimized for medium-size binaries** — best results under ~20 MB; larger files supported with reduced depth.
- 🧠 **Memory-aware execution** — avoids full disassembly of huge binaries; processes one region or function at a time.
- 🧩 **Selective mutation** — you can target specific functions, sections, or architectures for faster runs.
- 🔄 **Deterministic randomness** — seed-based mutation ensures reproducibility in experiments.
- 🔬 **Integrity validation overhead** — verification and rebuild steps add minor processing time (~5–10 %).
- 🧾 **Supported formats: PE / ELF / Mach-O (limited testing on stripped or packed samples).**
- ⚙️ **Known limitations:**
    a. Deeply obfuscated binaries may lose entry-point alignment.
    b. No support (yet) for self-modifying or compressed executables.
    c. Some relocations may require manual patching on exotic formats.

Goal: keep transformations safe, measurable, and reproducible — not "fastest," but trustworthy.

ROADMAP

PERFORMANCE & LIMITS

- 🧩 **Plugin system for mutation modules** — automatic discovery and loading of new morphers without editing the core.
- 🧠 **Mutation graph intelligence** — understand dependencies between mutations and optimize transformation plans.
- 🧾 **Full transformation trace** — per-instruction diff (before/after) with rule metadata for reproducibility and auditing.
- 🧪 **Semantic validation & benchmark suite** — ensure mutated binaries preserve behavior through automated tests.
- 🌐 **REST API / Service mode** — expose r2morph as a remote transformation engine for automation and scaling.
- ⚙️ **Parallel & distributed execution** — morph large binary sets with isolated processes and memory controls.
- 📈 **Advanced evasion scoring** — enhance the scoring model using code similarity, entropy delta, and opcode diversity.
- 🔍 **Visualization toolkit** — generate before/after CFG graphs and mutation density maps for analysis and presentations.
- 🤝 **Ecosystem integration** — seamless workflow with r2inspect, mwemu, and YARA-AST for full pipeline analysis.

CONCLUSIONS

PERFORMANCE & LIMITS

- 🧠 **r2morph bridges analysis and transformation** — moving from understanding code to shaping it with precision.
- ⚙️ **Built on Radare2's foundation**, it proves that safe, reproducible binary morphing is achievable at scale.
- 🧩 **For researchers and defenders alike:** enables controlled experimentation, evasion testing, and dataset generation.
- 📊 **Transparency first:** every mutation is logged, measurable, and verifiable — reproducibility is the core principle.
- 🌍 **Open, extensible, and community-driven** — ready to evolve with new mutation logic, plugins, and integrations.

From static analysis to controlled chaos — r2morph helps us understand transformation itself.

THANKS

https://github.com/seifreed/r2morph