undo
making software reliable

Datasheet: Live Recorder

Accelerate software defect resolution by eliminating the guesswork in software failure diagnosis

Key fact

91% of software developers admit to having unresolved defects because they cannot reproduce them.* *Reproducibility is* the fundamental problem in software defect resolution.

The challenge

- Engineering teams regularly hit intermittent failing tests which they cannot reproduce (and therefore cannot fix) - every deployment is full of ticking time bombs.
- Troubleshooting software defects is hard and time-consuming because it is impossible to recreate all of the conditions under which the software was running.
- Traditional debugging techniques (e.g. printf, logging, etc) are based on guesswork - wasting valuable engineering time, blocking development pipelines and making customers wait longer for a fix.

What does Live Recorder do?

- Captures exact recordings of program execution down to instruction level – turning sporadic failures into 100% reproducible events.
- Data capture is optimized so it can be stored in memory (configurable size) and the entire program's memory and register state can be recreated on demand with minimal overhead.
- Comes with an integrated reversible debugger so the recording can be debugged by replaying it backwards and forwards (replay on any machine).

Key features

Supports C/C++, Go, Rust, Ada applications on Linux x86 and x86_64. No kernel changes required. Compatible with all mainstream Linux distributions. Java support coming soon.

Faster than you think. Expect a 2x to 5x slowdown. Only record failed processes (not all of them) and record them only once to get all the data you need to debug.

Seamless integration into your Linux program & development workflow via command-line recording, easy-to-use API control, IDE integrations with Eclipse, Clion, Goland & Emacs.

Handles multi-threaded programs and those that use shared memory & asynchronous I/O.

Multi-process correlation identifies the processes and times at which data structures of interest are accessed in applications that share data structures in shared memory.

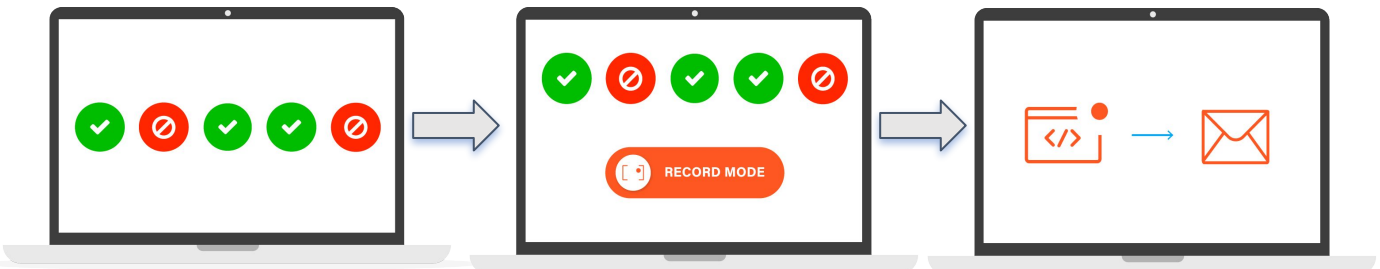
Thread-fuzzing randomizes thread execution to reveal race conditions and other multi-threading defects.

undo
making software reliable

Datasheet: Live Recorder

Get full visibility into what your software *really* did when it failed

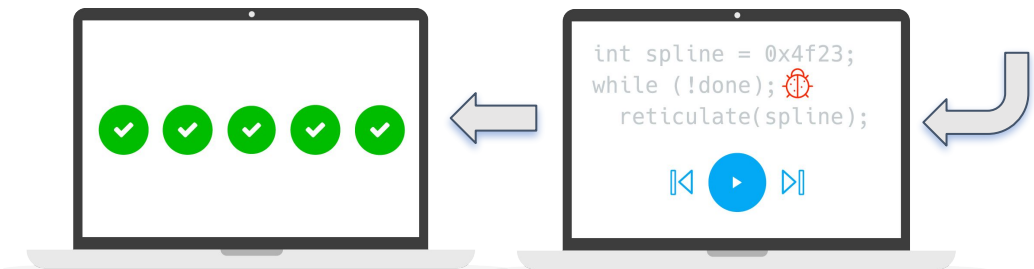
How it works in development and test



Your test system reveals a failed test. Live Recorder integrates with all major test systems.

Re-run failed tests under recording; or record failed tests automatically using Live Recorder APIs.

The recording is self-contained and made available as an artifact of your test system. Share the recording file with anyone in Engineering



Debug the recording using Live Recorder's integrated reversible debugger. Step backwards & forwards in the recording to diagnose the bug. The recording is an exact replica of the failing run; no need to try to *reproduce* the defect.

Fix the defect and commit the fix to your build system. Re-run tests, pass the tests and move on!

How it works in production

1. A customer reports a failure with your live application. But it's impossible to reproduce what the customer was doing

2. Activate Live Recorder (which is embedded in your product when you deploy) and capture the failure in a recording file. The recording serves as a standalone reproducible test case.

3. FAEs or Core Engineers can debug the recording offline with Live Recorder's integrated reversible debugger. Once the root cause is identified, a fix can be implemented & committed to the build.

4. Once your tests turn green, deploy your changes into production

Trusted by leaders

Undo helps enterprise software vendors make their mission-critical software reliable

cadence SAP Mentor A Siemens Business MICRO FOCUS

Request a demo at https://undo.io/