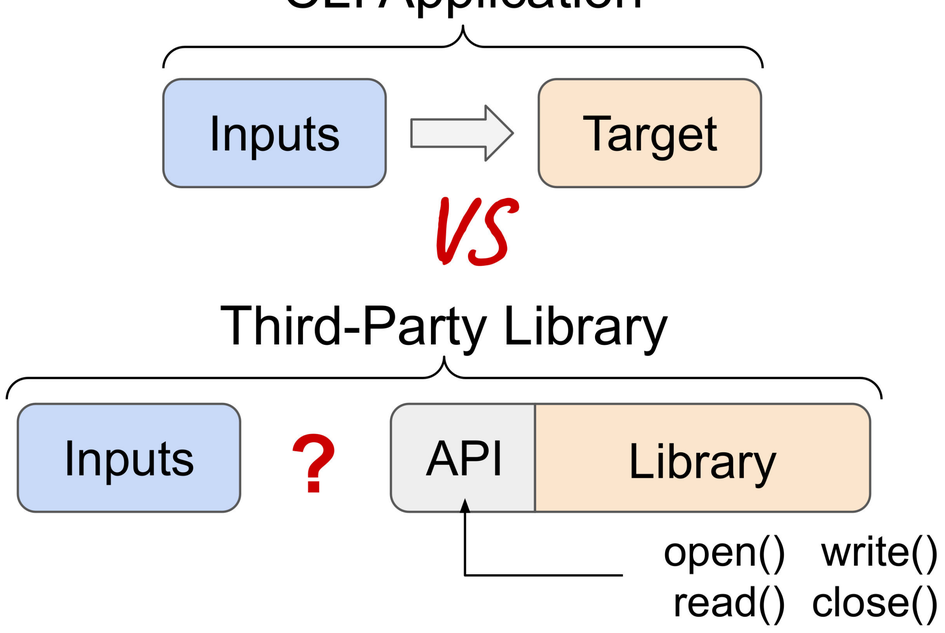
No App, No Problem: Learning and Fuzzing Library Usage Automatically

Flavio Toffalini, **Nicolas Badoux**, Zurab Tsinadze, Mathias Payer
EPFL, RUB EPFL EPFL EPFL

Libraries cannot be directly executed

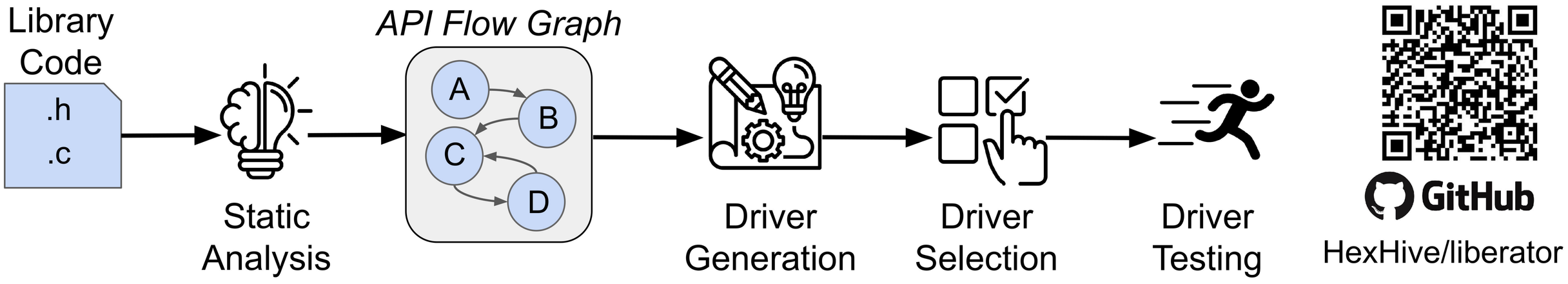


Learning API usage is complex

- Testing deepcode require state buildup
- Inter-API dependencies
  - write() comes before close()
- API arguments dependencies
  - Buffer and size for example
- How many drivers do we need?

Fuzzing drivers are still mostly manually written

libErator Architecture



Data acquisition through Static Analysis

- Populate the *API Flow Graph* (AFG)
- Interactions with arguments define API role
  - Writing to a field ➡ setup
  - free ➡ cleanup
- Infer intra-API dependencies from usage
- Type-based heuristics for initialization

Driver Generation and Selection

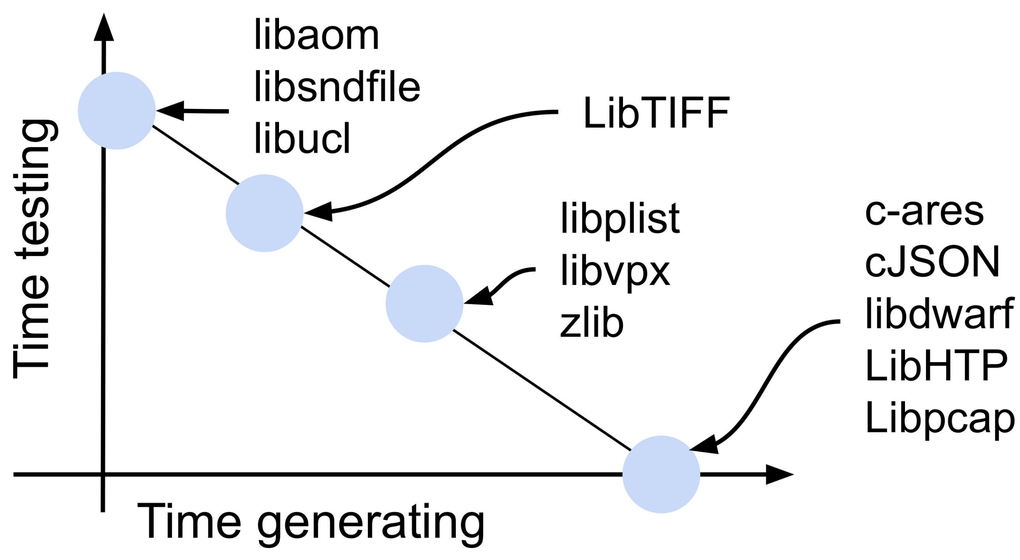
- Driver generation through AFG traversal
  - Grow driver iteratively
- Learns from seed feedback in quick testing
  - Discard unsuccessful chains
- Maximize drivers diversity through clustering

Evaluation

- 24 unique bugs, including a CVE in Libpcap
- 25% true positive crashes (vs 0.7% for SotA)
- False positives caused by:
  - Incoherent arguments (e.g., 2D array)
  - Incorrect memory tracking (e.g., UAF)
- Bugs were fixed, test cases contributed

Consumer-aware and manually written drivers are exhausted.

- First to be configurable with driver generation vs testing time which is different for each library



F. Toffalini, N. Badoux, Z. Tsinadze & M. Payer, "Liberating Libraries through Automated Fuzz Driver Generation" in Proceedings of the ACM on Software Engineering, 2 (FSE), p. 2123-2145. DOI:10.1145/3729365.