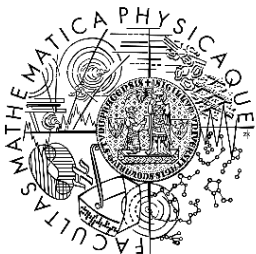


ACF: DEBUGGING LONG-RUNNING SYSTEMS WITH TRACING AND UNIT TEST EXECUTION

<https://d3s.mff.cuni.cz/>

Roman Vašut
Pavel Parízek



FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

Department of
Distributed and
Dependable
Systems



MOTIVATION EXAMPLE

- Assume anomalous network traffic sample
 - How to trace an anomaly to the bug?
- Test generation/execution from runtime data to help root cause localization

```
typedef struct {
    size_t length;
    char payload[4096];
} hb_message_t;

// returns the actual length of payload
int recv_hb(int sock, hb_message_t* msg);

int send_hb(int sock, char* buf, size_t len);

int main(int argc, char* argv[]) {
    // initializing sockets and connection

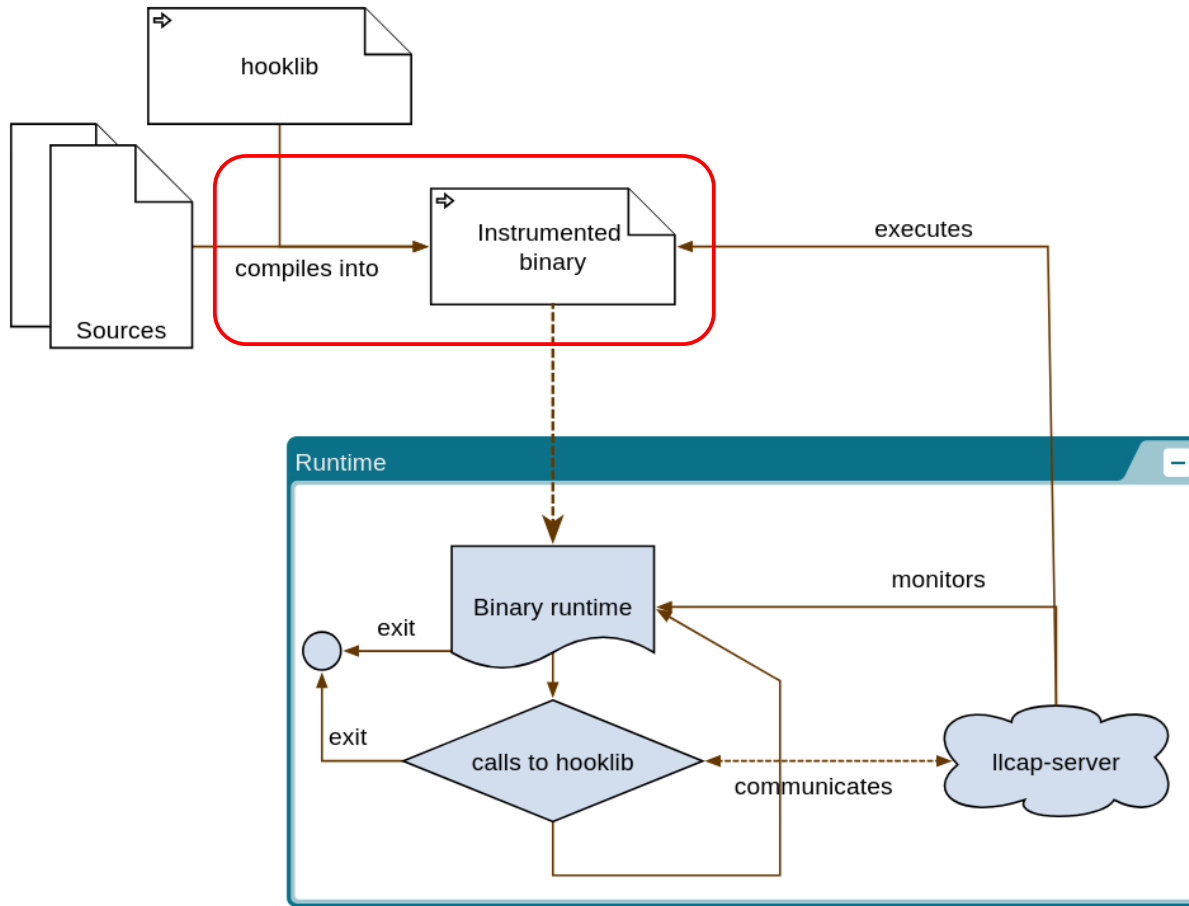
    hb_message_t hb_msg;

    int len1 = recv_hb(cli_sock, hb_msg);
    int resp1 = len1;
    send_hb(cli_sock, hb_msg.payload, resp1);
    ...
    int len2 = recv_hb(cli_sock, hb_msg);
    ... // possibly lot of code in between
    int resp2 = hb_msg.length;
    send_hb(cli_sock, hb_msg.payload, resp2);
}
```

ACF: AUTODEBUGGER C/C++ FRAMEWORK

- Three-phase workflow (record-and-replay)
 - two compilations, three program runs
 - one compilation, two runs possible
- 1) **Call tracing** (1st compilation and run)
- 2) **Argument capture** (2nd compilation and run)
 - Capture arguments of selected calls
- 3) **Testing** (3rd run)
 - Replace arguments with recorded values in different execution context
 - Result interpretation up to the user

ACF: ARCHITECTURE



Hooklib

- Runtime library cooperating with llcap-server

llcap-server

- Central dispatcher and manager of the workflow/data storage

ACF OUTPUT

```
P | [main] -----
P | [main] Test results (7):
P | [main] Module ID | Function ID | Call | Packet | Result
P | [main] F3199851 | 01000000 | 1 | 0 | Pass
P | [main] F3199851 | 01000000 | 1 | 1 | Pass
P | [main] F3199851 | 01000000 | 1 | 2 | Pass
P | [main] F3199851 | 01000000 | 1 | 3 | Pass
P | [main] F3199851 | 01000000 | 1 | 4 | Pass
P | [main] F3199851 | 01000000 | 1 | 5 | Pass
P | [main] F3199851 | 01000000 | 1 | 6 | Pass
P | [main] -----
P | [main] Exiting...
```

Function returned



Exit / killed with signal
before the function returned



P		[main]		553E8EC8		01000000		4		2		Pass
P		[main]		553E8EC8		01000000		4		3		Pass
P		[main]		553E8EC8		01000000		4		4		Exit(0)
P		[main]		553E8EC8		01000000		5		0		Pass
P		[main]		553E8EC8		01000000		5		1		Pass
P		[main]		553E8EC8		01000000		5		2		Pass
P		[main]		553E8EC8		01000000		5		3		Pass
P		[main]		553E8EC8		01000000		5		4		Signal(6)

INSTRUMENTATION

- Standard interplay of instrumentation and a runtime library

AST-level and LLVM IR-level plugins (C++)

- Inspect targeted functions, insert hooklib calls
- **LLVM patch** facilitates metadata transfer between the AST and LLVM IR plugins
 - Properties such as whether a given function (IR) came from a system header file or not

hooklib (C++)

- Cooperates with llcap-server (Rust)

CUSTOM TYPES

- Adjust and link necessary files with hooklib
- Protocol buffers allow recursive structures
 - Support code in hooklib for implementors
 - Pointer indirections not supported

```
message Vector {  
  uint64 capacity = 10;  
  repeated SingleArgVariant values = 12;  
}
```

```
message SingleArgVariant {  
  oneof value {  
    float flt = 10;  
    double dbl = 12;  
    sint32 i32 = 14;  
    sint64 i64 = 16;  
    uint32 u32 = 18;  
    uint64 u64 = 20;  
    bytes arbitrary = 22;  
    StringWrap str = 24;  
    Vector vec = 26;  
  }  
}
```



C++ (LINUX) EXCEPTION SUPPORT

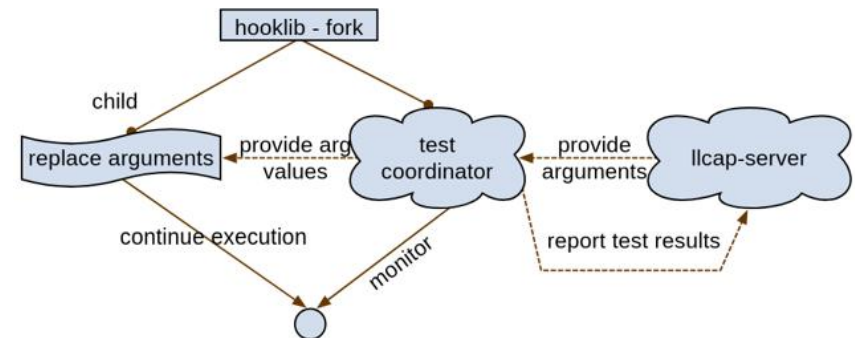
- SSA form easy to manipulate
- LLVM IR restructuring
- "try-catch wrapping" of all calls within the function

MULTI-THREADED PROGRAM SUPPORT

- Fork approach in testing (original solution)
 - Different argument set (argument values of a given call) => new forked child
 - Fast, cheap **X** fork in a MT process **kills all other threads**

- *Workaround:* swap the fork child and parent

- Child now communicates with llcap-server
- Parent fully multi-threaded **X one entire execution per one test case**
- Collected runtime argument values associated with thread ID



● CRIU: CHECKPOINT & RESTORE IN USERSPACE

- Checkpointing at right times just before the function starts (even the hooks)
- Restoring checkpoints and receiving argument values simulates forking
 - Except with threads staying alive
- Speed gains trivial and large
 - Even with one test case, wind up to n-th call

SCALING TO LARGE PROGRAMS

- Forking approach rather swift/cheap
- CRIU's footprint based on more factors
 - Resident set size, storage medium for CRIU dumps, available RAM (offloading)
- Testing loop itself
 - Influencing which argument sets are used (order)
 - Targeting specific behavior (sequences of threads)

FUTURE WORK

- **Automating interpretation:** hooks and predicates asserting input argument values vs. return value
 - Testing hypotheses when looking for a root cause
- Thread interleaving exploration (modes)
 - Current: rather trivial mode
 - Up-to fully user-defined strategy
- Configuration streamlining, IR improvements, ...
- Fuzzing opportunities ?

- Interactive GUI tool
 - Controlling test execution and exploring results

INTEGRATION WITH OTHER TOOLS

- CPAchecker
 - Analyze behavior of individual procedures
 - Driven by the small focused unit tests
 - Symbolic execution of directed small tests
 - Representing unspecified values and system inputs
 - Validation (including witness checker)
 - Q: API for invoking CPAchecker from another program and getting outputs ?

SUMMARY & DISCUSSION

- Technical proof-of-concept
 - Automating the testing loop (user interpreting the result)
 - Evaluated on three programs: KeepassXC, GNU find, yaga SMT solver

System	Target procedure	Command-line	Calls	Tests	Running time
KeepassXC	PasswordGenerator::setLength(int)	keepassxc (the GUI variant), keepassxc-cli generate -length 30	211	23	18 minutes
find	inside_dir(int fd)	find /etc/systemd -name '*.conf'	265	70225	65 minutes
YAGA	yaga::Model_base::is_defined(int)	<yaga-build-dir>/smt input.smt	360	112559	102 minutes

- Future work focused on main usability drawbacks
 - Interpretation of results, thread interleaving
- Unsolvable in current model: inherent nondeterminism
- <https://github.com/d3sformal/debug-infra-gentests>
- R. Vašut and P. Parízek. *ACF: Infrastructure for Debugging Long-Running Systems with Tracing and Unit Test Execution*. PLOS 2026, ACM.

