



# ACF: Infrastructure for Debugging Long-Running Systems with Tracing and Unit Test Execution

**Roman Vašut, Pavel Parízek**

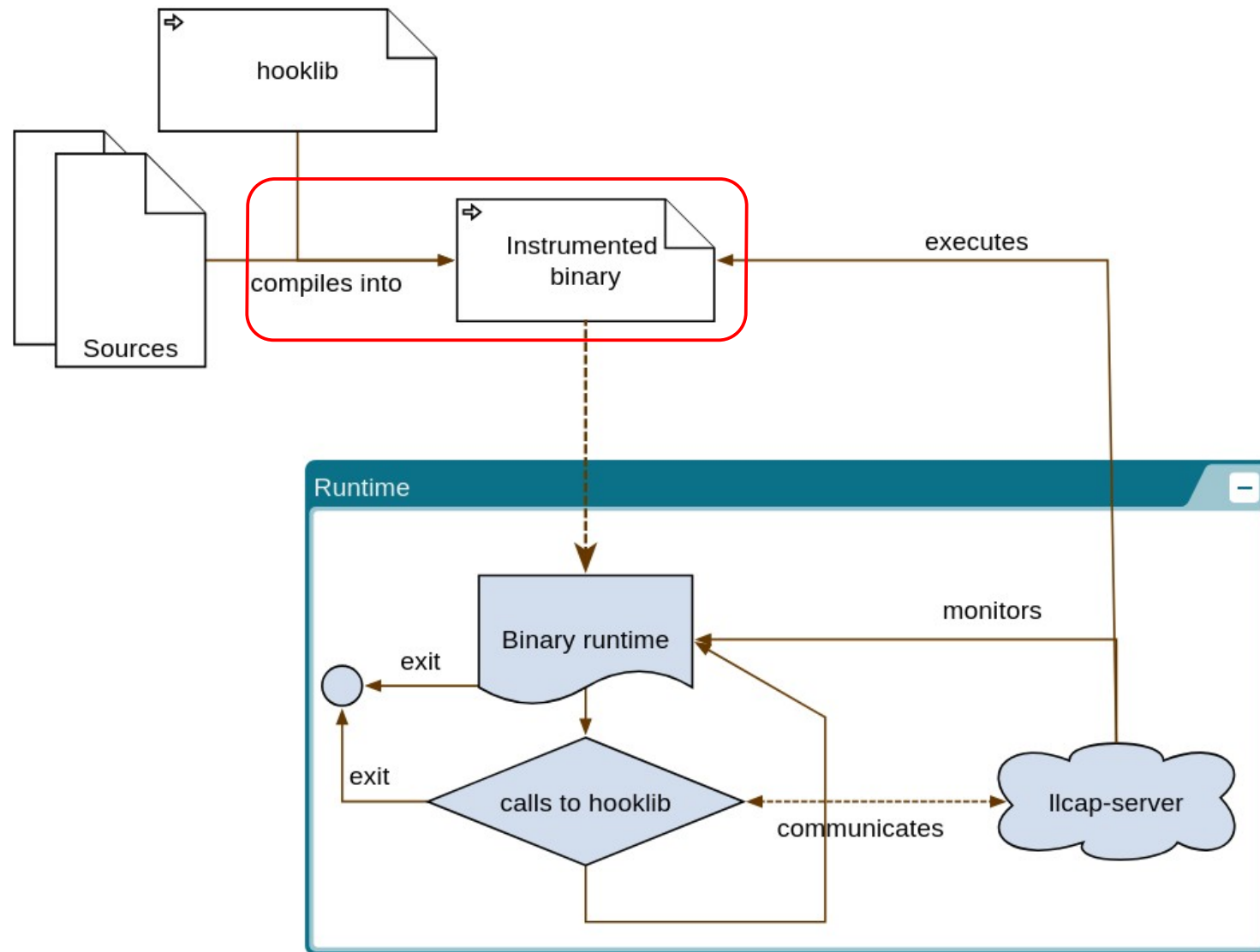
# Motivating example

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

```
1 typedef struct{
2     size_t length;
3     char payload[4096];
4 } hb_message_t;
5
6 // returns the actual length of payload
7 int recv_hb(int sock, hb_message_t* msg);
8
9 int send_hb(int sock, char*buf, size_t len);
10
11 int main(int argc, char* argv[])
12 {
13     hb_message_t hb_msg;
14
15     int len1 = recv_hb(cli_sock, hb_msg);
16     int resp1 = len1;
17     send_hb(cli_sock, hb_msg.payload, resp1);
18     // ...
19     int len2 = recv_hb(cli_sock, hb_msg);
20     // ... // possibly lot of code in between
21     int resp2 = hb_msg.length;
22     send_hb(cli_sock, hb_msg.payload, resp2);
23 }
```

# ACF

- Three-phase workflow
    - Record-and-replay style
    - 2 compilations, 3 program runs
      - 1 compilation, 2 runs possible
- 1) Call tracing (1<sup>st</sup> compilation)
  - 2) **Argument capture** (2<sup>nd</sup> compilation)
    - Capture arguments of selected calls
  - 3) **Testing**
    - Replace arguments with recorded values in different execution context – result interpretation up to the user



## 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

## 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;
```

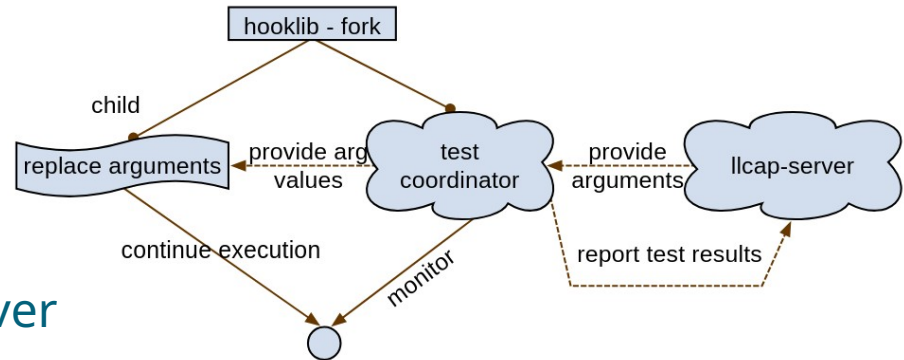


# C++ (Linux) exception support

- 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
  - 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**

# 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

- Limited testing (proof-of-concept)
- 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
  - Targeting specific behavior/sequences of threads

# Future work

- **Thread interleaving modes**
  - Current: 1 trivial mode
  - Up-to fully user-defined strategy
- **Automating interpretation:** hooks and predicates asserting input argument values vs. return value
  - Testing hypotheses when looking for a cause
  - Sanity checks
- Configuration streamlining, IR improvements, ...
- Fuzzing opportunities?

# Summary & Discussion

- Technical proof-of-concept automating the testing loop (user interpreting the result)
  - Evaluated on 3 programs (KeepassXC, GNU find, yaga SMT solver)
- Future work focused on main usability drawbacks
  - Interpretation of results, thread interleaving
- Unsolvable in current model: inherent nondeterminism



[github.com/d3sformal/debug-infra-gentests](https://github.com/d3sformal/debug-infra-gentests)

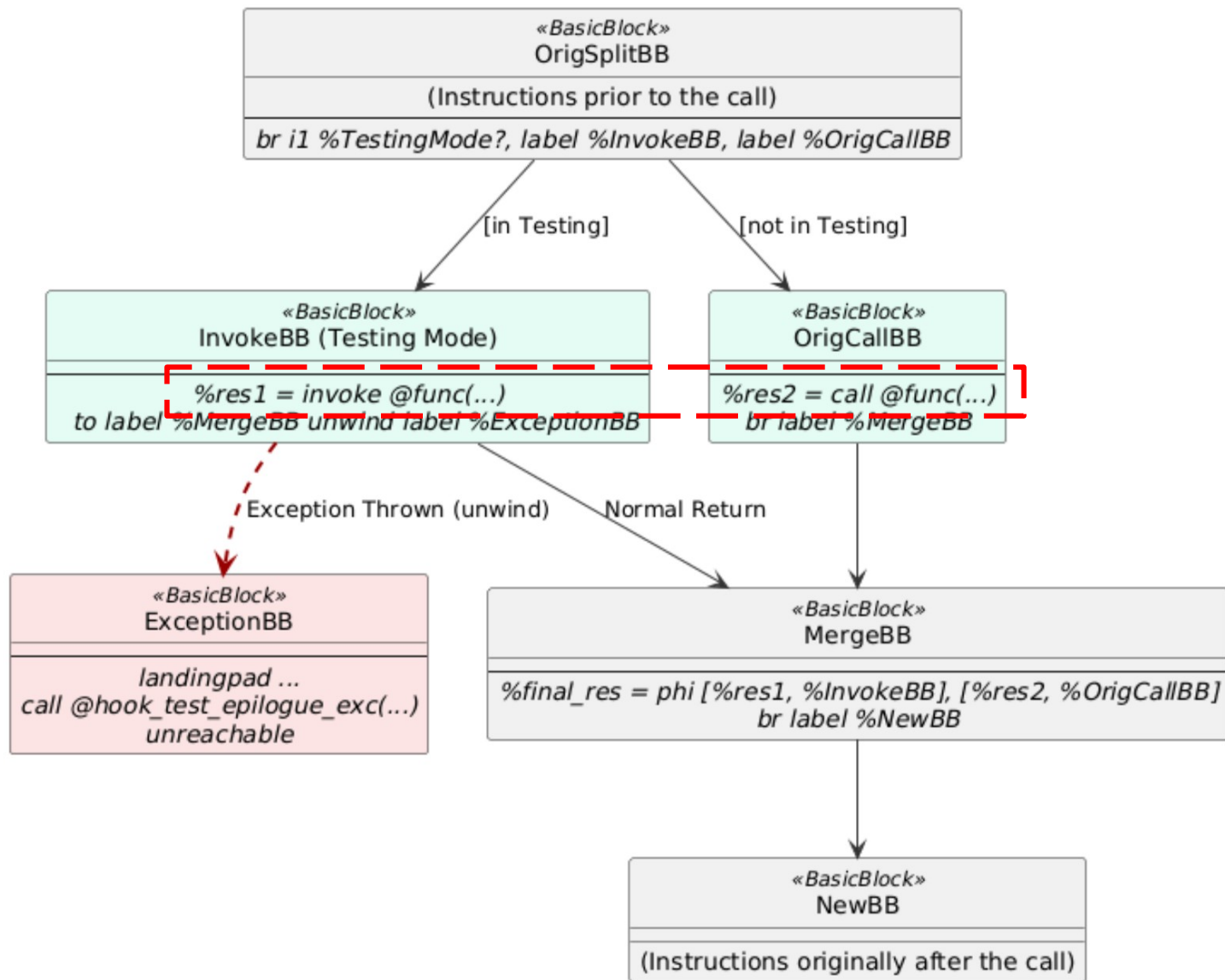


[github.com/BohdanQQ/debug-infra-gentests/tree/thesis/main](https://github.com/BohdanQQ/debug-infra-gentests/tree/thesis/main)

Extra slides

# Thread identification

- Thread logical ID (LID) – simple first-come-first-serve auto-incrementing integer
- Relation to testing: test only a predetermined LID (bind tests to a thread)
  - Calls from other threads (different LID) possible, even concurrently, but not intercepted (!)
- Open-ended (extensions planned)



# Pros & Cons thus far

- (+) Automated execution and monitoring
- (-) Interpretation of outcomes required
- (-) Shared state can tamper with reproduction of bugs/environment
- (-) Inherent nondeterminism of multi-threaded execution

Which ones can we realistically try to tackle?