arm

Fuzzing TF-RMM with AFL++

2026 update

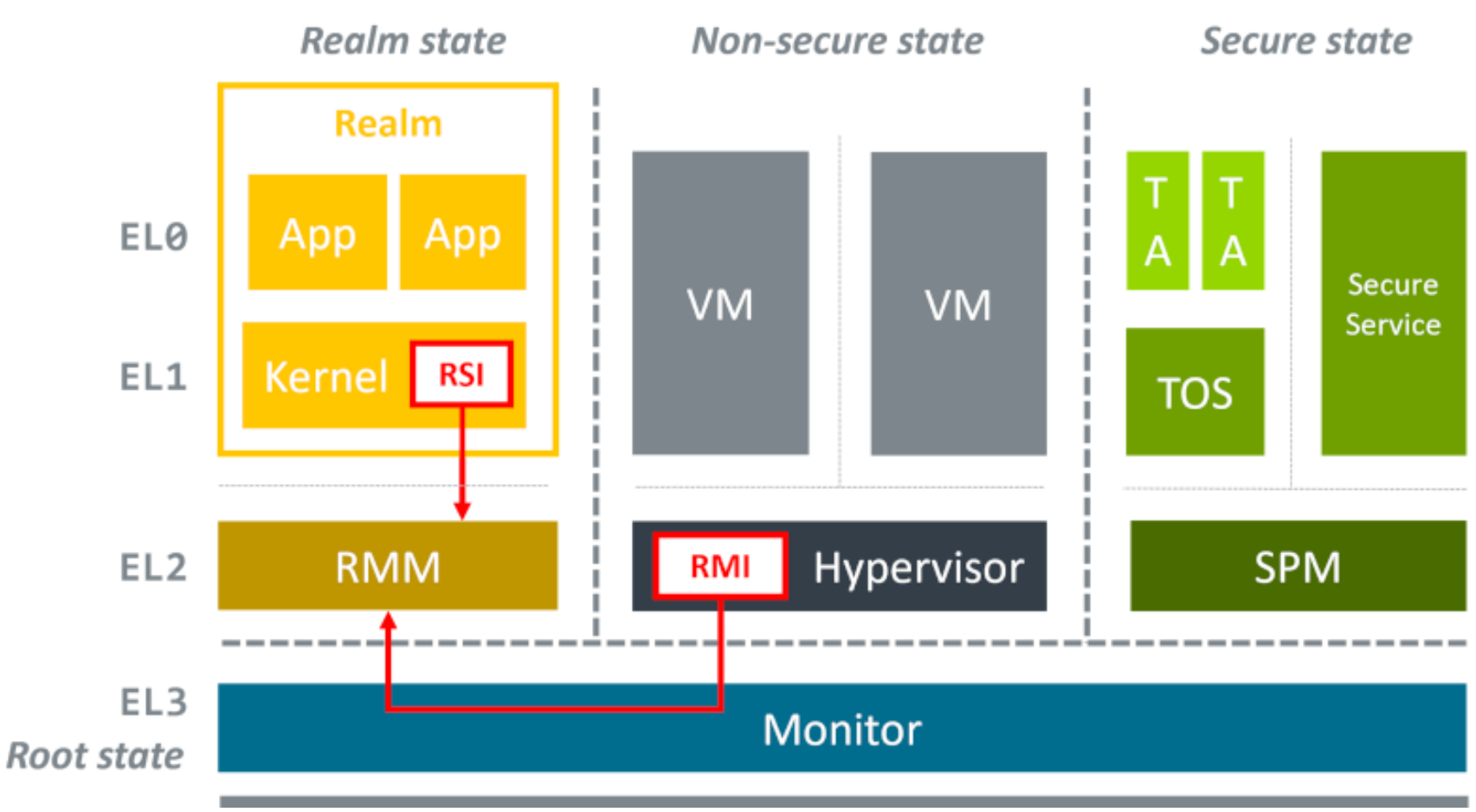
Rustam Ismayilov

May 2026

TF-RMM

Fuzzed interfaces

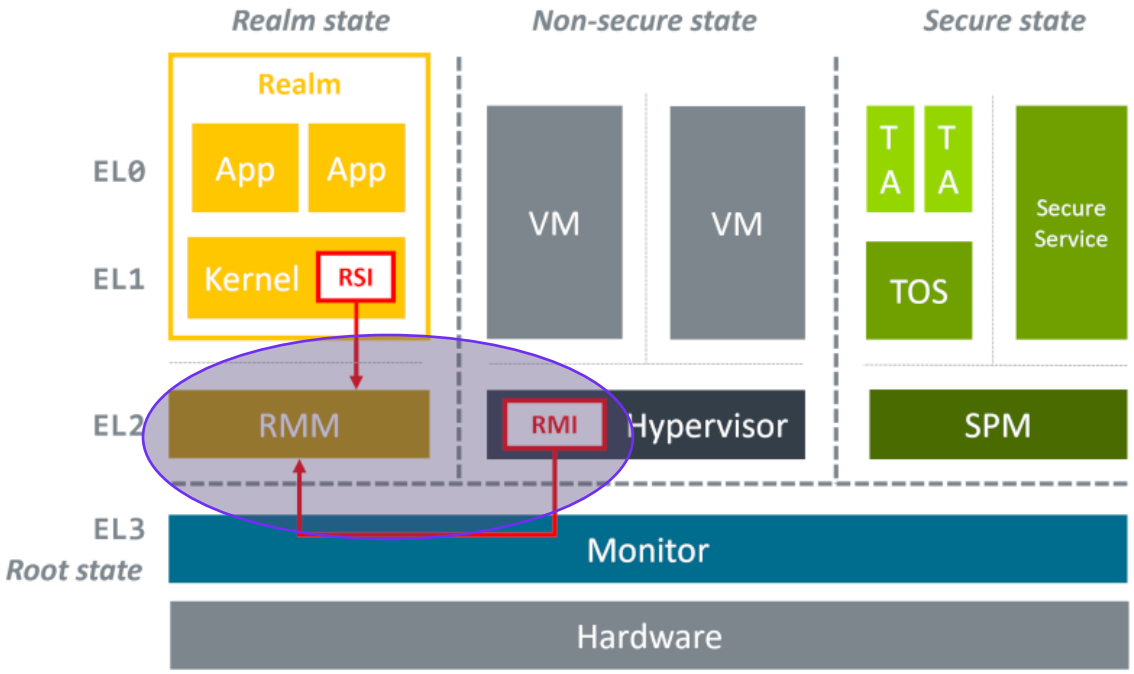
- RMI
- RSI (WIP)
- PSCI (WIP)



TF-RMM

RMI - Realm Management Interface

- RMM manages Realms
- Host calls RMM through RMI



FID	Command
0xC4000150	RMI_VERSION
0xC4000151	RMI_GRANULE_DELEGATE
0xC4000152	RMI_GRANULE_UNDELEGATE
0xC4000153	RMI_DATA_CREATE
0xC4000154	RMI_DATA_CREATE_UNKNOWN
0xC4000155	RMI_DATA_DESTROY
...	
0xC4000157	RMI_REALM_ACTIVATE
0xC4000158	RMI_REALM_CREATE
0xC4000159	RMI_REALM_DESTROY
0xC400015A	RMI_REC_CREATE
0xC400015B	RMI_REC_DESTROY
0xC400015C	RMI_REC_ENTER
0xC400015D	RMI_RTT_CREATE
0xC400015E	RMI_RTT_DESTROY
0xC400015F	RMI_RTT_MAP_UNPROTECTED
...	

TF-RMM

RMI - Realm Management Interface

- RMM behaviour depends on previous states
- Fuzzing is useful because these state combinations are hard to cover manually.

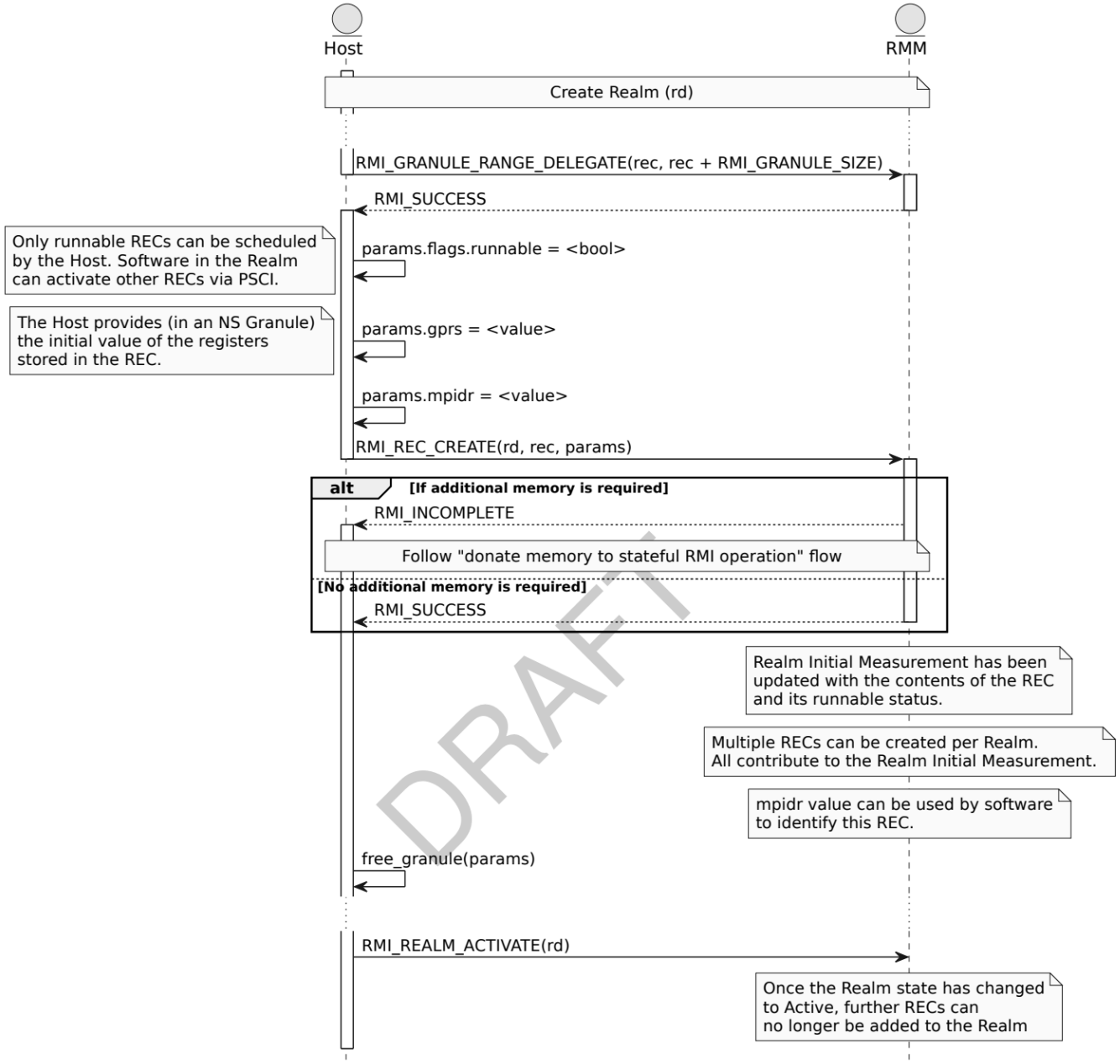


Figure D1.7: REC creation flow

RMI – command flow

Non-secure Host

Test driver

SMC exception

RMI wrapper

```
void handle_ns_smc(unsigned int function_id,
                   unsigned long arg0,
                   unsigned long arg1,
                   unsigned long arg2,
                   unsigned long arg3,
                   unsigned long arg4,
                   unsigned long arg5,
                   struct smc_result *res){...}
```

```
HANDLER(VERSION,                1, 2, smc_version,
HANDLER(GRANULE_DELEGATE,       1, 0, smc_granule_delegate,
HANDLER(GRANULE_UNDELEGATE,     1, 0, smc_granule_undelegate,
HANDLER(DATA_CREATE,            5, 0, smc_data_create,
HANDLER(DATA_CREATE_UNKNOWN,    3, 0, smc_data_create_unknown,
HANDLER(DATA_DESTROY,           2, 2, smc_data_destroy,
HANDLER(PDEV_AUX_COUNT,         0, 0, NULL,
HANDLER(REALM_ACTIVATE,         1, 0, smc_realm_activate,
HANDLER(REALM_CREATE,           2, 0, smc_realm_create,
HANDLER(REALM_DESTROY,          1, 0, smc_realm_destroy,
HANDLER(REC_CREATE,             3, 0, smc_rec_create,
HANDLER(REC_DESTROY,            1, 0, smc_rec_destroy,
HANDLER(REC_ENTER,              2, 0, smc_rec_enter,
HANDLER(RTT_CREATE,             4, 0, smc_rtt_create,
HANDLER(RTT_DESTROY,            3, 2, smc_rtt_destroy,
```

RMI – command flow

RMI_REC_CREATE

Non-secure Host

Test driver

SMC exception

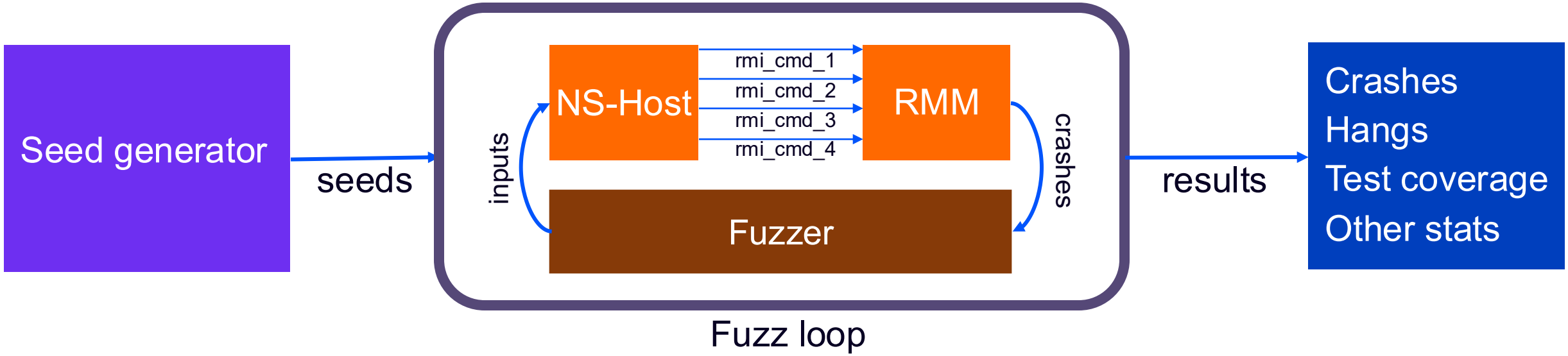
RMI wrapper

```
handle_ns_smc( function_id: SMC_RMI_REC_CREATE,
               arg0: (uintptr_t)rd,
               arg1: (uintptr_t)rec,
               arg2: (uintptr_t)params_ptr,
               arg3: 0,  arg4: 0,  arg5: 0,
               res);
```

```
HANDLER(VERSION,                1, 2, smc_version,
HANDLER(GRANULE_DELEGATE,       1, 0, smc_granule_delegate,
HANDLER(GRANULE_UNDELEGATE,     1, 0, smc_granule_undelegate,
HANDLER(DATA_CREATE,            5, 0, smc_data_create,
HANDLER(DATA_CREATE_UNKNOWN,    3, 0, smc_data_create_unknown,
HANDLER(DATA_DESTROY,           2, 2, smc_data_destroy,
HANDLER(PDEV_AUX_COUNT,         0, 0, NULL,
HANDLER(REALM_ACTIVATE,         1, 0, smc_realm_activate,
HANDLER(REALM_CREATE,           2, 0, smc_realm_create,
HANDLER(REALM_DESTROY,          1, 0, smc_realm_destroy,
HANDLER(REC_CREATE,             3, 0, smc_rec_create,
HANDLER(REC_DESTROY,            1, 0, smc_rec_destroy,
HANDLER(REC_ENTER,              2, 0, smc_rec_enter,
HANDLER(RTT_CREATE,             4, 0, smc_rtt_create,
HANDLER(RTT_DESTROY,            3, 2, smc_rtt_destroy,
```

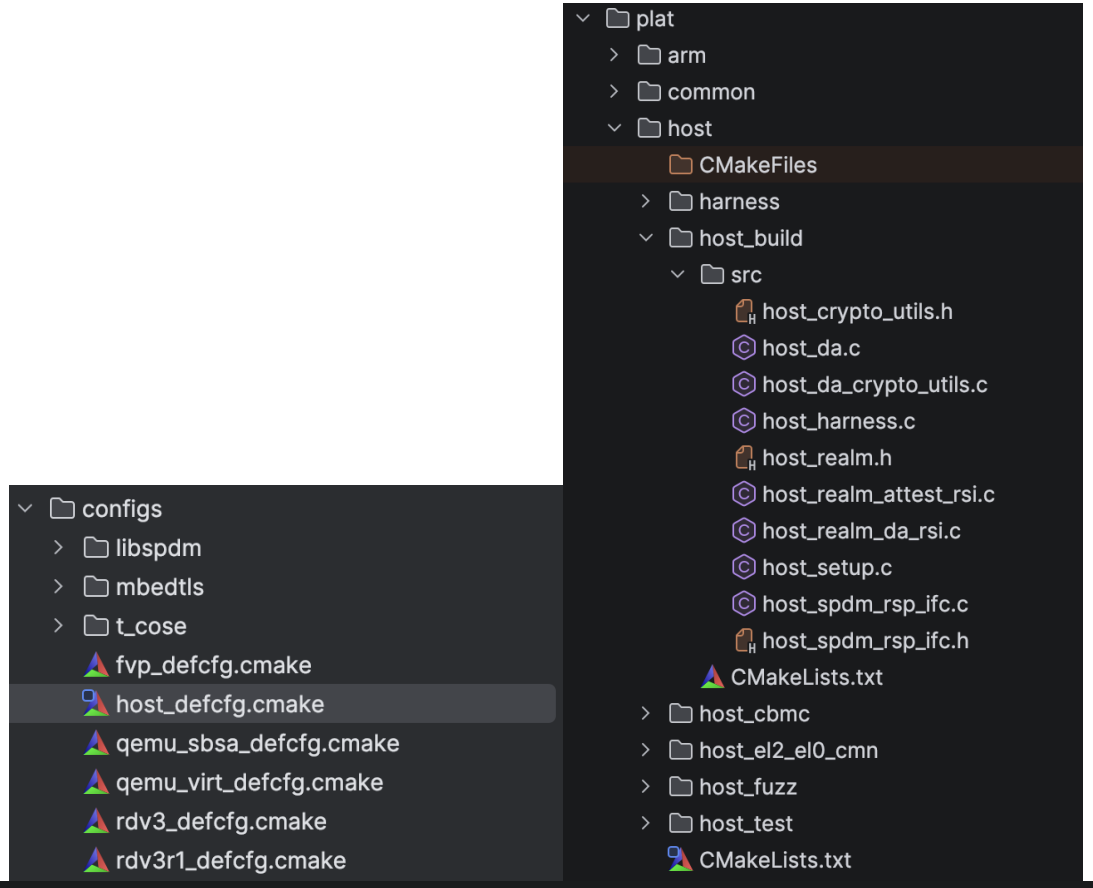
```
unsigned long smc_rec_create(unsigned long rd_addr,
                             unsigned long rec_addr,
                             unsigned long rec_params_addr)
```

Fuzz overview



Fake Host

- Emulates enough of the Arm platform and firmware context for RMM code
- Runs as a native process
- Quick functional test environment
- Not an accurate hardware representation



18. RMM Fake Host Build

RMM also provides a `fake_host` target architecture which allows the code to be built natively on the host using the host toolchain. To build for `fake_host` architecture, set RMM_CONFIG=host_defcfg during the configuration stage.

Fake Host

fake_host for fuzzing?

- Faster than full platform simulation
- Easier to instrument with AFL++
- Easier crash reproduction
- Supports coverage reporting
- Allows RMM logic to be fuzzed in-process

```
f host_util_set_sysreg_cb(char* name, rd_cb_t rd_cb, wr_cb
f host_util_set_default_sysreg_cb(char* name, u_register_t
f host_util_take_sysreg_snapshot(void) -> void
f host_util_restore_sysreg_snapshot(void) -> void
f host_util_zero_sysregs_and_cbs(void) -> void
f host_util_get_granule_base(void) -> unsigned long
f host_util_get_dev_granule_base(void) -> unsigned long
f host_util_set_cpuid(unsigned int cpuid) -> void
f host_util_get_el3_rmm_shared_buffer(void) -> unsigned c
f host_util_setup_sysreg_and_boot_manifest(void) -> void
f host_util_rec_run(unsigned long* regs) -> int
T realm_entrypoint_t -> int(*)(unsigned long*)
f host_util_rsi_helper(realm_entrypoint_t ep) -> int
f process_command_line_arguments(int argc, char* argv[])
```

```
void plat_setup(uint64_t x0, uint64_t x1,
                uint64_t x2, uint64_t x3)
{
        (void)host_csl_init();

        register_host_monitor_functions(
                monitor_call: host_monitor_call,
                host_monitor_call_with_res);

        /* Initialize the RMM-EL3 interface */
        if (plat_cmn_init_el3_ifc(x0, x1, x2, x3) != 0) {
                panic();
        }

        /* Carry on with the rest of the system setup */
        if (plat_cmn_setup(NULL, nregions: 0) != 0) {
                panic();
        }

        plat_warmboot_setup(x0, x1, x2, x3);
}
```

```
#include <smc-rmi.h>

void host_rmi_version(unsigned long rmi_version, struct smc_result *res);
void host_rmi_granule_delegate(void *granule_address, struct smc_result *res);
void host_rmi_granule_undelegate(void *granule_address, struct smc_result *res);
void host_rmi_data_create(void *rd, void *data, uintptr_t ipa, void *src,
                uint64_t flags, struct smc_result *res);
void host_rmi_data_create_unknown(void *rd, uintptr_t data, uintptr_t ipa, struct
void host_rmi_data_destroy(void *rd, uintptr_t ipa, struct smc_result *res);

void host_rmi_realm_activate(void *rd, struct smc_result *res);
void host_rmi_realm_create(void *rd, void *params_ptr, struct smc_result *res);
void host_rmi_realm_destroy(void *rd, struct smc_result *res);
void host_rmi_rec_create(void *rd, void *rec, void *params_ptr, struct smc_result
void host_rmi_rec_destroy(void *rec, struct smc_result *res);
void host_rmi_rec_enter(void *rec, void *run_ptr, struct smc_result *res);
void host_rmi_rtt_create(void *rd, void *rtt, void *ipa,
```

RMI – host_fuzz flow

RMI_REC_CREATE

Non-secure Host

Test driver

SMC exception

RMI wrapper

```
void host_rmi_rec_create(void *rd, void *rec, void *params_ptr, struct smc_result *res)
{
        handle_ns_smc( function_id: SMC_RMI_REC_CREATE,
                       arg0: (uintptr_t)rd,
                       arg1: (uintptr_t)rec,
                       arg2: (uintptr_t)params_ptr,
                       arg3: 0,  arg4: 0,  arg5: 0,
                       res);
}
```

```
HANDLER(VERSION,                1, 2, smc_version,
HANDLER(GRANULE_DELEGATE,       1, 0, smc_granule_delegate,
HANDLER(GRANULE_UNDELEGATE,     1, 0, smc_granule_undelegate,
HANDLER(DATA_CREATE,            5, 0, smc_data_create,
HANDLER(DATA_CREATE_UNKNOWN,    3, 0, smc_data_create_unknown,
HANDLER(DATA_DESTROY,           2, 2, smc_data_destroy,
HANDLER(PDEV_AUX_COUNT,         0, 0, NULL,
HANDLER(REALM_ACTIVATE,         1, 0, smc_realm_activate,
HANDLER(REALM_CREATE,           2, 0, smc_realm_create,
HANDLER(REALM_DESTROY,          1, 0, smc_realm_destroy,
HANDLER(REC_CREATE,             3, 0, smc_rec_create,
HANDLER(REC_DESTROY,            1, 0, smc_rec_destroy,
HANDLER(REC_ENTER,              2, 0, smc_rec_enter,
HANDLER(RTT_CREATE,             4, 0, smc_rtt_create,
HANDLER(RTT_DESTROY,            3, 2, smc_rtt_destroy,
```

```
unsigned long smc_rec_create(unsigned long rd_addr,
                             unsigned long rec_addr,
                             unsigned long rec_params_addr)
```

RMI – host_fuzz flow

RMI_REC_CREATE

Fuzz harness:

```
int main(int argc, char *argv[])
{
 size_t read_res = readCorpus(argc, argv, buffer);

 execute(buffer, read_res);
```

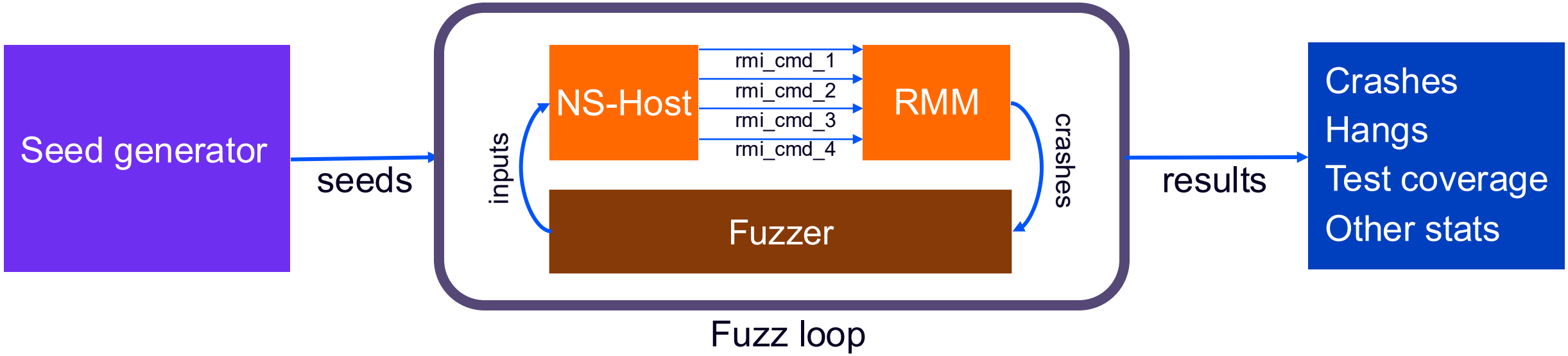
```
int execute(unsigned char *buffer, size_t read_res)
        while (1) {
                switch (command) {

                case COMMAND_DATA_DESTROY:{...}

                case COMMAND_REALM_ACTIVATE:{...}

                case COMMAND_REALM_CREATE:{...}

                case COMMAND_REALM_DESTROY:{...}

                case COMMAND_REC_CREATE:{...}

                case COMMAND_REC_DESTROY:{...}
```

```
unsigned long smc_rec_create(unsigned long rd_addr,
                             unsigned long rec_addr,
                             unsigned long rec_params_addr)
```

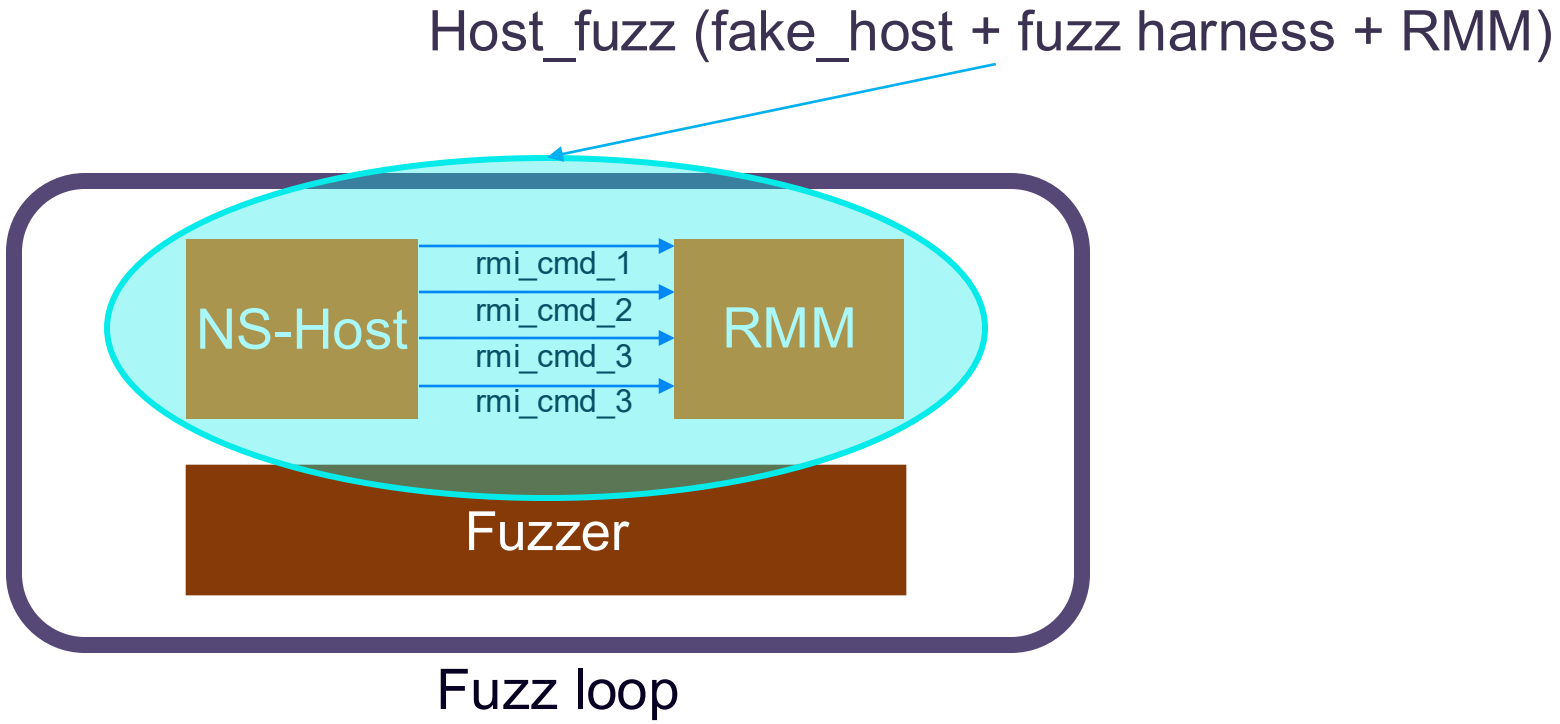
```
void host_rmi_rec_create(void *rd, void *rec, void *params_ptr, struct smc_result *res)
{
        handle_ns_smc( function_id: SMC_RMI_REC_CREATE,
                       arg0: (uintptr_t)rd,
                       arg1: (uintptr_t)rec,
                       arg2: (uintptr_t)params_ptr,
                       arg3: 0,  arg4: 0,  arg5: 0,
                       res);
}
```

```
HANDLER(VERSION,                1, 2, smc_version,
HANDLER(GRANULE_DELEGATE,       1, 0, smc_granule_delegate,
HANDLER(GRANULE_UNDELEGATE,     1, 0, smc_granule_undelegate,
HANDLER(DATA_CREATE,            5, 0, smc_data_create,
HANDLER(DATA_CREATE_UNKNOWN,    3, 0, smc_data_create_unknown,
HANDLER(DATA_DESTROY,           2, 2, smc_data_destroy,
HANDLER(PDEV_AUX_COUNT,         0, 0, NULL,
HANDLER(REALM_ACTIVATE,         1, 0, smc_realm_activate,
HANDLER(REALM_CREATE,           2, 0, smc_realm_create,
HANDLER(REALM_DESTROY,          1, 0, smc_realm_destroy,
HANDLER(REC_CREATE,             3, 0, smc_rec_create,
HANDLER(REC_DESTROY,            1, 0, smc_rec_destroy,
HANDLER(REC_ENTER,              2, 0, smc_rec_enter,
HANDLER(RTT_CREATE,             4, 0, smc_rtt_create,
HANDLER(RTT_DESTROY,            3, 2, smc_rtt_destroy,
```

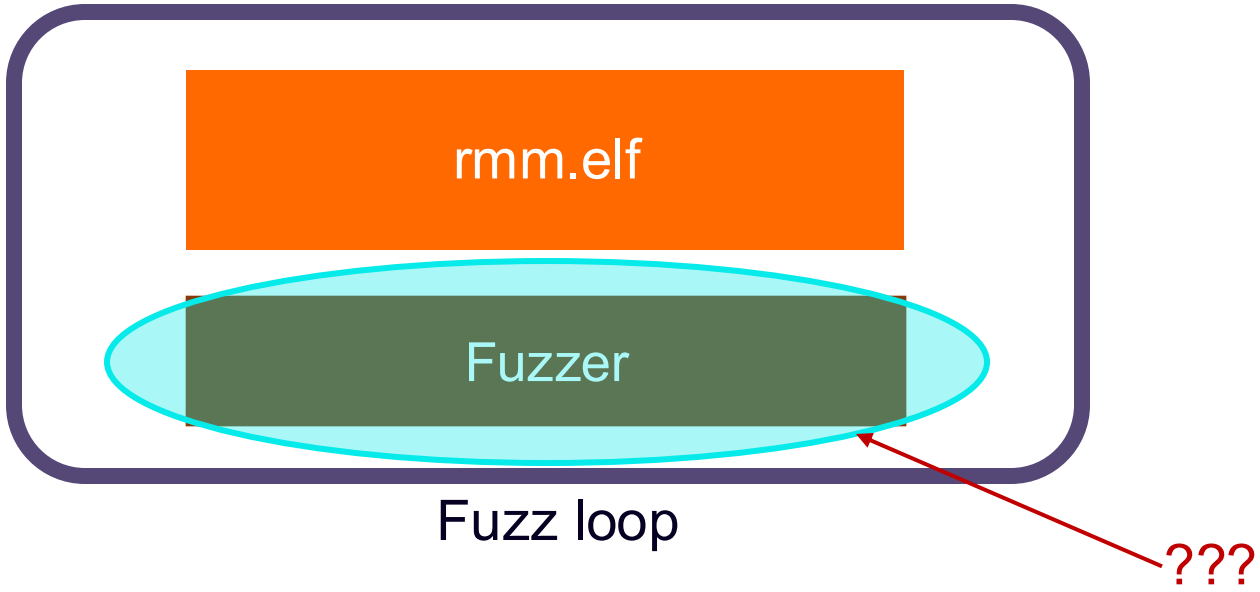
Target



Target



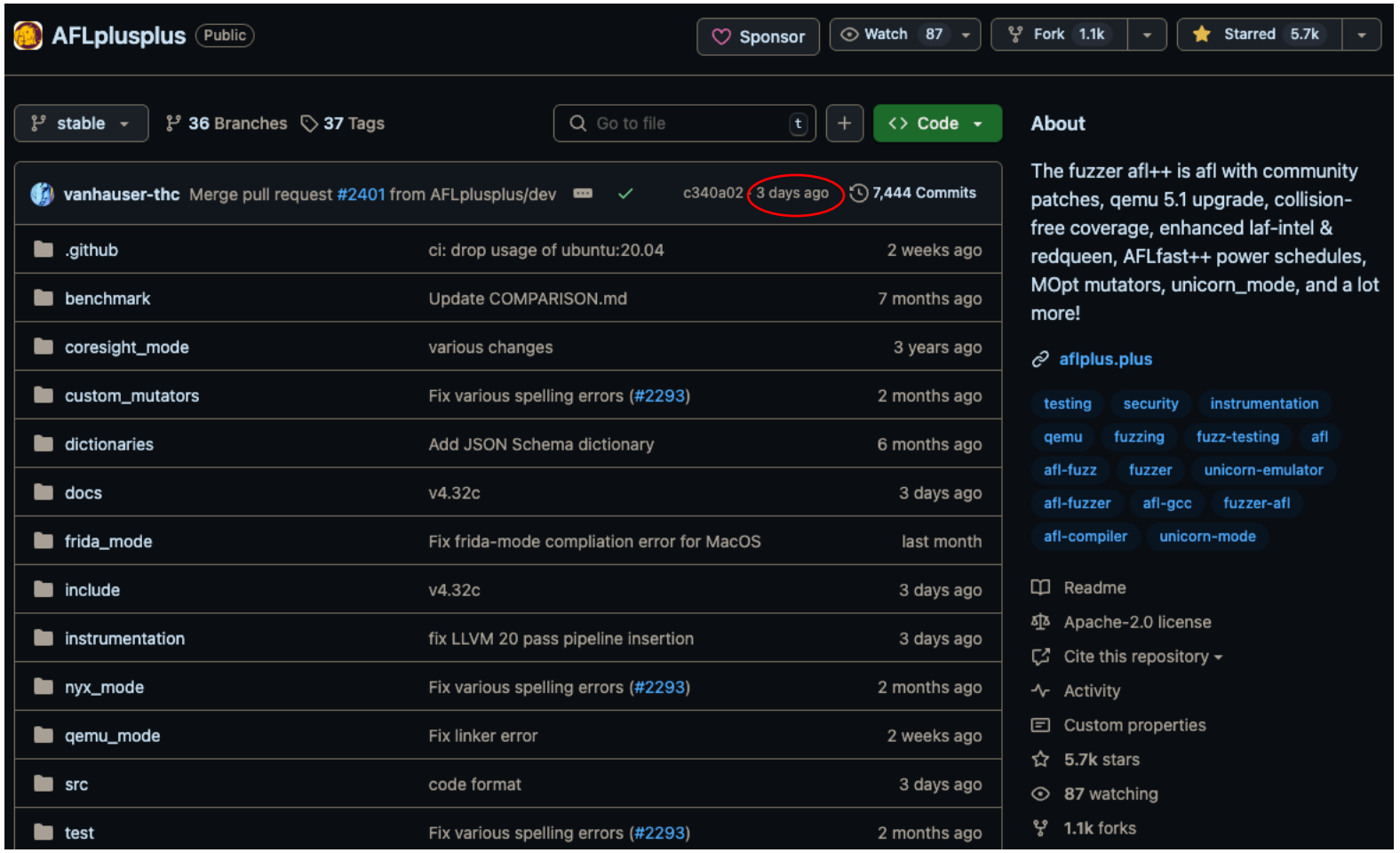
Fuzzer?



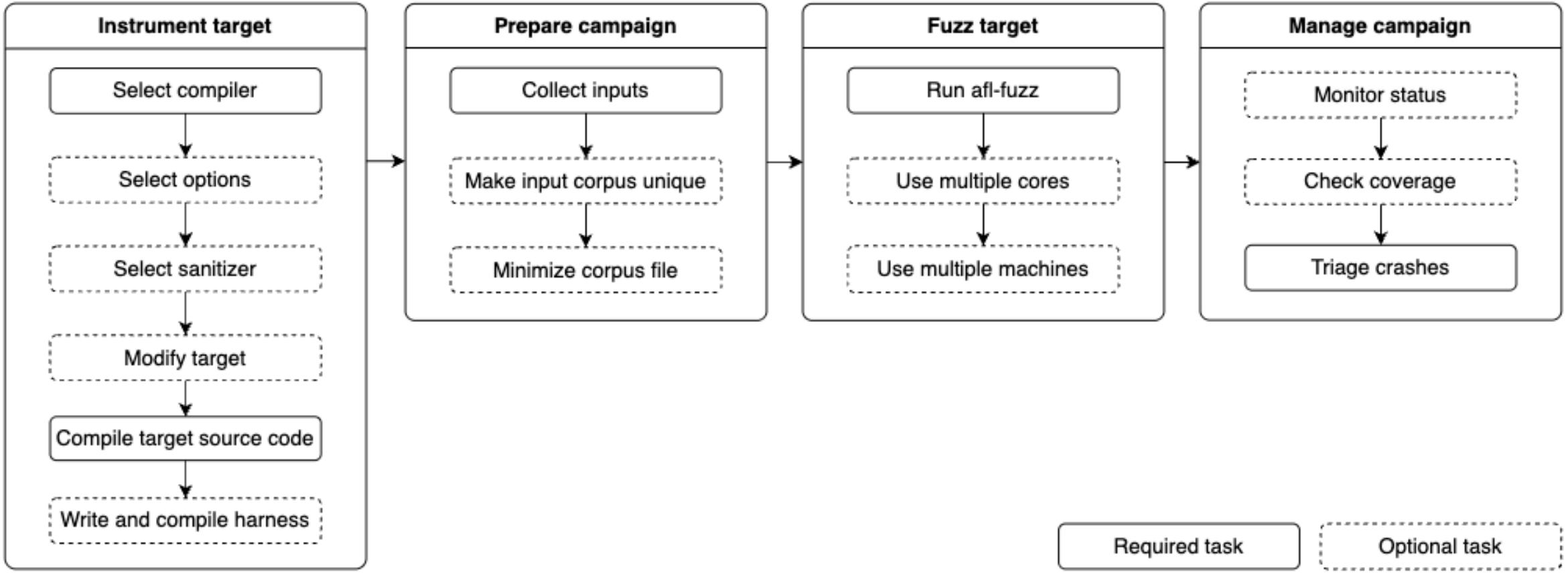
AFL++

"… a superior fork to Google's AFL - more speed, more and better mutations, more and better instrumentation, custom module support, etc."

- Actively maintained
- Parallelizable
- Persistent mode*

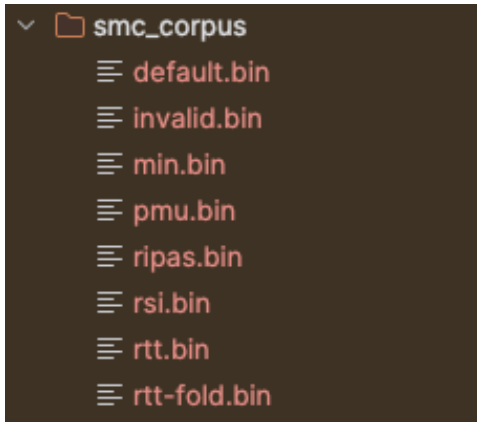


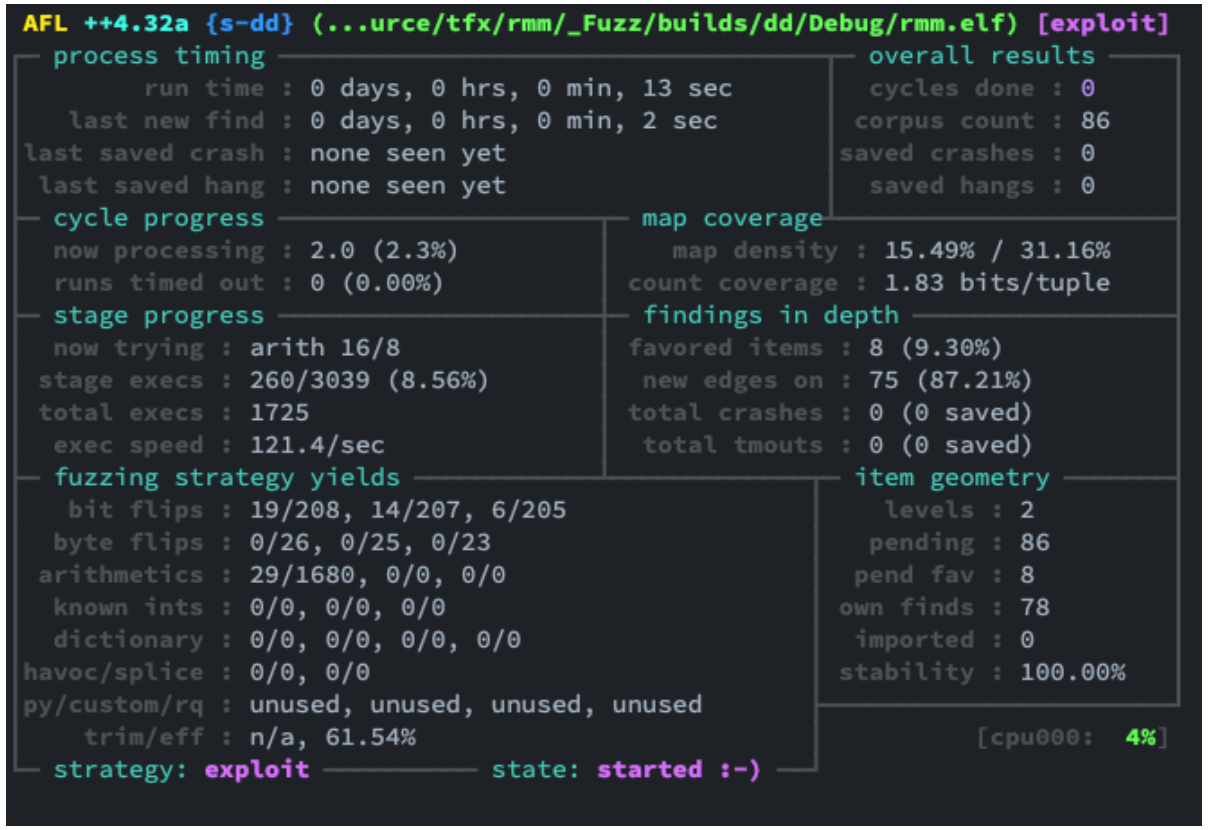
Fuzzing steps



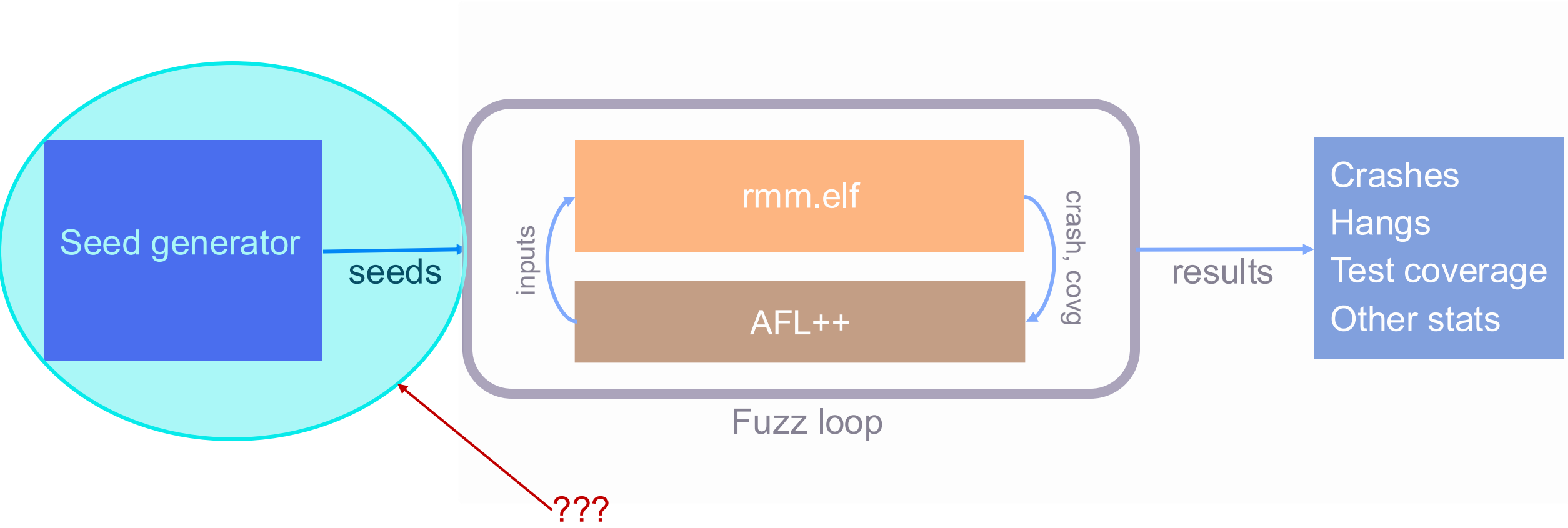
Instrumentation, Inputs, Fuzz

```
find_program(CMAKE_C_COMPILER
        NAMES "afl-clang-lto"
#        NAMES "afl-clang-fast"
        DOC "Path to AFL clang."
        REQUIRED)

find_program(CMAKE_CXX_COMPILER
        NAMES "afl-clang-lto++"
#        NAMES "afl-clang-fast++"
        DOC "Path to AFL clang++."
        REQUIRED)
```





Seeds



Input binaries

scapy

```
class RecCreate(Packet):
    name = "Rec create"
    fields_desc = [
        ByteField( name: "rd_index",  default: 0),
        ByteField( name: "rec_index",  default: 0),
        ByteField( name: "param_index",  default: 0),
        LELongField( name: "flags",  default: 0),
        LELongField( name: "mpidr",  default: 0),
        # LELongField("pc", 0),
        LELongField( name: "num_aux",  default: 0),
        ByteField( name: "aux_index0",  default: 0),
        ByteField( name: "aux_index1",  default: 0),
        ByteField( name: "aux_index2",  default: 0),
        ByteField( name: "aux_index3",  default: 0),
        ByteField( name: "aux_index4",  default: 0),
        ByteField( name: "aux_index5",  default: 0),
        ByteField( name: "aux_index6",  default: 0),
        ByteField( name: "aux_index7",  default: 0),
        ByteField( name: "aux_index8",  default: 0),
```

```
bind_layers(RMI, AllocateGranule, command=99)
bind_layers(RMI, InitRsi, command=98)

bind_layers(RMI, Version, command=0)
bind_layers(RMI, GranuleDelegate, command=1)
bind_layers(RMI, GranuleUndelegate, command=2)
bind_layers(RMI, DataCreate, command=3)
bind_layers(RMI, DataCreateUnknown, command=4)
bind_layers(RMI, DataDestroy, command=5)
# < reserved command 6>
bind_layers(RMI, RealmActivate, command=7)
bind_layers(RMI, RealmCreate, command=8)
bind_layers(RMI, RealmDestroy, command=9)
bind_layers(RMI, RecCreate, command=10)
bind_layers(RMI, RecDestroy, command=11)
bind_layers(RMI, RecEnter, command=12)
bind_layers(RMI, RTTCreate, command=13)
bind_layers(RMI, RTTDestroy, command=14)
bind_layers(RMI, RTTMapUnprotected, command=15)
```

```
if __name__ == "__main__":
    packets = []

    packets.append(Version(req=0x00010000))
    packets.append(Features(index=1))

    rd = 0
    rec = 1

    packets.append(AllocateGranule(index=rd))
    packets.append(AllocateGranule(index=rec))

    packets.append(GranuleDelegate(index=rd))
    packets.append(GranuleDelegate(index=rec))

    packets.append(GranuleUndelegate(index=rd))
    packets.append(GranuleUndelegate(index=rec))

    import os
    import sys

    os.makedirs(os.path.dirname(sys.argv[1]), exist_ok=True)
    with open(sys.argv[1], "wb") as f:
        for p in packets:
            rmi_packet = RMI() / p
            f.write(raw(rmi_packet))
```

```
00000000: 0000 0001 0000 0000 0015 0100 0000 6300  ..............c.
00000010: 6301 0100 0101 0200 0201                 c.........
```

Input binaries

Stateful fuzzing

- RMM behavior depends on previous calls
- Calls may require objects to already exist
- Object states:
  - delegated vs undelegated granules
  - active vs inactive realm
  - mapped vs unmapped RTT entries
  - REC created vs destroyed
- Valid seeds help AFL reach deeper states

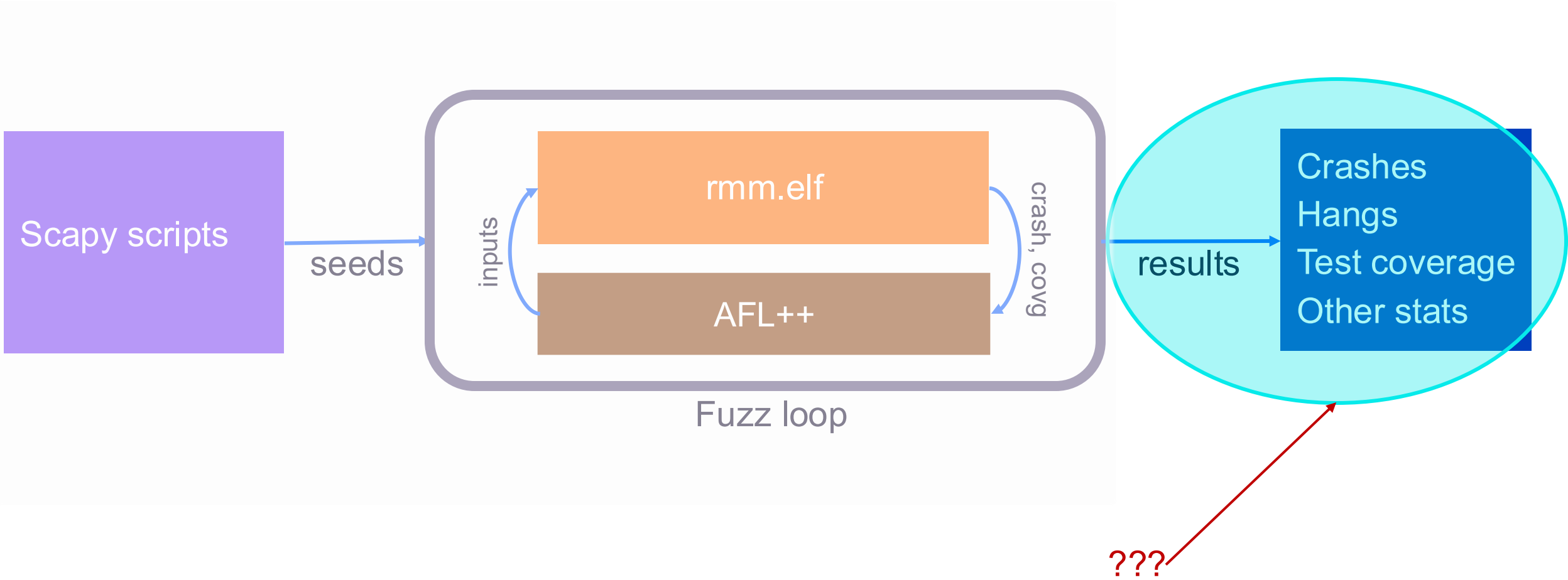
```
# Allocate source granules (NS - used to seed data map)
packets.append(AllocateGranule(index=src))
packets.append(AllocateGranule(index=src2))

# --- Realm create ---
# s2sz=0x30 (48-bit PA), rtt_level_start=0 means 4-level walk starting
# from L0; rtt_num_start=1 means one L0 RTT entry (rtt[0]).
packets.append(
    RealmCreate(rd_index=rd, param_index=realm_params,
                s2sz=0x30, num_bps=1, num_wps=1,
                rtt_base_index=rtt_base, rtt_level_start=0, rtt_num_start=1))

# --- Create RTT sub-tables for protected IPA 0 (levels 1, 2, 3) ---
for level in range(1, 4):
    packets.append(RTTCreate(rd_index=rd, rtt_index=rtt_base + level,
                             ipa=0, level=level))

# --- Create RTT sub-tables for unprotected IPA 2**47 (levels 1, 2, 3) ---
for level in range(1, 4):
    packets.append(RTTCreate(rd_index=rd, rtt_index=rtt_base + 3 + level,
                             ipa=UNPROTECTED_IPA, level=level))

# --- Map data granules into the protected IPA space ---
packets.append(RTTReadEntry(rd_index=rd, ipa=0, level=1))
packets.append(RttInitRipas(rd_index=rd,
                            base=REALM_DATA_IPA,
                            top=REALM_DATA_IPA + 2 * GRANULE_SIZE))
packets.append(RttDataMapInit(rd_index=rd, data_index=realm_data,
                              ipa=REALM_DATA_IPA, src=src))
packets.append(RttDataMapInit(rd_index=rd, data_index=realm_data2,
                              ipa=REALM_DATA_IPA + GRANULE_SIZE, src=src2))

# --- Create REC then drive SRO protocol to donate aux granules ---
packets.append(RecCreate(rd_index=rd, rec_index=rec,
                         param_index=rec_params, flags=0x1))
packets.append(SroDonate(count=0))
packets.append(SroContinue(flags=0))

# --- Activate realm ---
packets.append(RealmActivate(rd_index=rd))

# --- Map an NS (unprotected) page for the realm to use ---
packets.append(RTTMapUnprotected(rd_index=rd, ipa=UNPROTECTED_IPA,
                                 level=3, desc=0b1100))

# --- Enter realm ---
packets.append(RecEnter(rec_index=rec, run_index=rec_run))

# --- Teardown ---
# Unmap unprotected entry
packets.append(RTTUnmapUnprotected(rd_index=rd, ipa=UNPROTECTED_IPA, level=3))
```

Run RMM binary with input binary

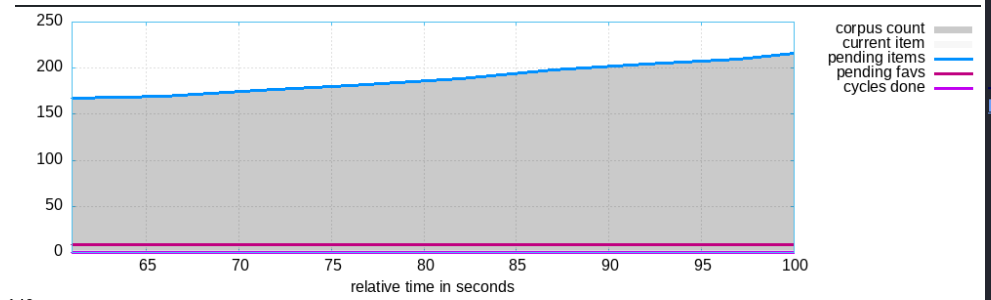
Typical host_fuzz cli use

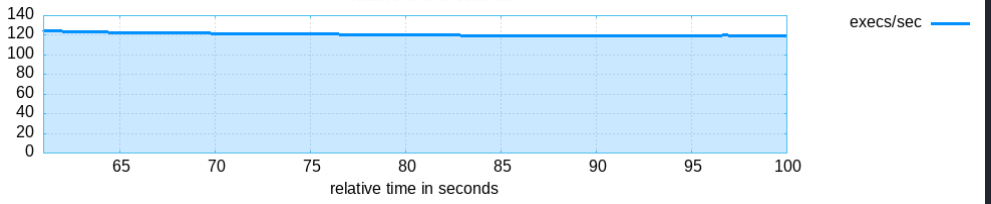
```
(venv) rusism01@e129829:/data/rusism01/workspace/source/tfx/rmm$ ./_Fuzz/builds/dd/Debug/rmm.elf _Fuzz/builds/dd/smc_corpus/min.bin
Booting RMM v.0.6.0(debug) tf-rmm-v0.6.0-106-g7a12bf0-dirty Built: Apr 28 2025 18:27:55 with Clang 14.0.0
RMM-EL3 Interface v.0.4
Boot Manifest Interface v.0.3
RMI ABI revision v1.0
RSI ABI revision v1.0
_FH: Version(req = 0x10000),
SMC_RMI_VERSION                    10000 > RMI_SUCCESS 10000 10000
RMI interface version: Low: 1.0 High: 1.0
_FH: Features(index = 0x1),
SMC_RMI_FEATURES                   1 > RMI_SUCCESS 0
Feature register value: 0x00000000
_FH: AllocateGranule(index = 0x0),
_FH: AllocateGranule(index = 0x1),
_FH: GranuleDelegate(index = 0x0),
_FH: GranuleDelegate(index = 0x1),
_FH: GranuleUndelegate(index = 0x0),
_FH: GranuleUndelegate(index = 0x1),
```

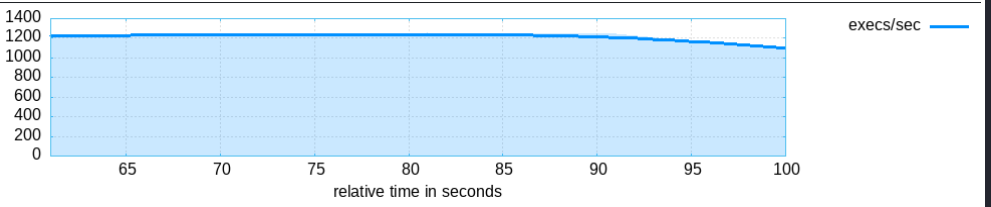

Outputs



Coverage and stats







GCC Code Coverage Report

Directory: ./
Date: 2026-05-13 18:12:47
Coverage: low: ≥ 0% medium: ≥ 75.0% high: ≥ 90.0%

List of functions

	Coverage	Exec	Excl	Total
Lines:	44.4%	7198	0	16209
Functions:	53.1%	751	0	1413
Branches:	30.2%	2417	0	8016

File	Lines		Functions		Branches	
app/common/el0_app/src/el0_app_helpers.c	0.0%	0 / 0 / 23	0.0%	0 / 0 / 4	0.0%	0 / 0 / 4
app/common/el0_app/src/fake_host/el0_app_arch.c	0.0%	0 / 0 / 224	0.0%	0 / 0 / 14	0.0%	0 / 0 / 218
app/common/framework/src/fake_host/app.c	77.6%	204 / 0 / 263	76.5%	13 / 0 / 17	49.5%	101 / 0 / 204
app/common/framework/src/fake_host/app_header.c	88.9%	32 / 0 / 36	100.0%	6 / 0 / 6	62.5%	5 / 0 / 8
app/common/rmm_svc/src/app_services.c	30.4%	84 / 0 / 276	30.8%	4 / 0 / 13	15.1%	19 / 0 / 126
app/device_assignment/el0_app/spdm_requester/rmm_libspdm_config.h	0.0%	0 / 0 / 5	0.0%	0 / 0 / 1	-%	0 / 0 / 0
app/device_assignment/el0_app/src/dev_assign_cmds.c	0.0%	0 / 0 / 136	0.0%	0 / 0 / 7	0.0%	0 / 0 / 38
app/device_assignment/el0_app/src/dev_assign_el0_app.c	0.0%	0 / 0 / 826	0.0%	0 / 0 / 34	0.0%	0 / 0 / 315
app/device_assignment/el0_app/src/dev_assign_helper.c	0.0%	0 / 0 / 34	0.0%	0 / 0 / 1	0.0%	0 / 0 / 18
app/device_assignment/el0_app/src/dev_assign_ide_cmds.c	0.0%	0 / 0 / 290	0.0%	0 / 0 / 13	0.0%	0 / 0 / 90
app/device_assignment/el0_app/src/dev_tdisp_cmds.c	0.0%	0 / 0 / 150	0.0%	0 / 0 / 4	0.0%	0 / 0 / 48
app/device_assignment/el0_app/src/fake_host/dev_assign_app_host.c	0.0%	0 / 0 / 3	0.0%	0 / 0 / 1	-%	0 / 0 / 0
app/device_assignment/el0_app/src/rme_dvsec.c	0.0%	0 / 0 / 139	0.0%	0 / 0 / 10	0.0%	0 / 0 / 80
app/device_assignment/rmm_stub/src/dev_assign_app_stub.c	0.0%	0 / 0 / 105	0.0%	0 / 0 / 6	0.0%	0 / 0 / 62
app/random/el0_app/src/fake_host/random_app_host.c	0.0%	0 / 0 / 3	0.0%	0 / 0 / 1	-%	0 / 0 / 0
app/random/el0_app/src/random_app.c	0.0%	0 / 0 / 52	0.0%	0 / 0 / 4	0.0%	0 / 0 / 17
app/random/rmm_stub/src/app_support/random_app.c	85.9%	79 / 0 / 92	87.5%	7 / 0 / 8	56.8%	25 / 0 / 44
lib/addr_list/include/addr_list.h	100.0%	4 / 0 / 4	100.0%	1 / 0 / 1	50.0%	1 / 0 / 2
lib/addr_list/src/addr_list.c	83.1%	162 / 0 / 195	100.0%	13 / 0 / 13	48.3%	72 / 0 / 149
lib/addr_list/src/addr_list_prv.h	100.0%	57 / 0 / 57	100.0%	15 / 0 / 15	50.0%	4 / 0 / 8
lib/arch/include/arch_features.h	69.0%	80 / 0 / 116	77.8%	28 / 0 / 36	35.4%	17 / 0 / 48
lib/arch/include/arch_helpers.h	41.9%	98 / 0 / 234	38.2%	149 / 0 / 390	-%	0 / 0 / 0
lib/arch/include/esr.h	9.1%	3 / 0 / 33	14.3%	1 / 0 / 7	0.0%	0 / 0 / 12
lib/arch/include/fake_host/atomics.h	71.9%	41 / 0 / 57	70.0%	7 / 0 / 10	-%	0 / 0 / 0
lib/arch/include/fake_host/cpuid.h	100.0%	3 / 0 / 3	100.0%	1 / 0 / 1	-%	0 / 0 / 0
lib/arch/include/fake_host/entropy.h	100.0%	4 / 0 / 4	100.0%	1 / 0 / 1	-%	0 / 0 / 0
lib/arch/include/fake_host/memory.h	-%	0 / 0 / 0	-%	0 / 0 / 0	-%	0 / 0 / 0
lib/arch/include/fake_host/spinlock.h	100.0%	8 / 0 / 8	100.0%	2 / 0 / 2	-%	0 / 0 / 0
lib/arch/include/simd.h	0.0%	0 / 0 / 8	0.0%	0 / 0 / 2	-%	0 / 0 / 0
lib/arch/src/arch_features.c	80.5%	136 / 0 / 169	100.0%	25 / 0 / 25	45.2%	28 / 0 / 62
lib/arch/src/fake_host/cache_wrappers.c	66.7%	10 / 0 / 15	66.7%	2 / 0 / 3	-%	0 / 0 / 0
lib/arch/src/fake_host/instr_helpers.c	100.0%	9 / 0 / 9	100.0%	2 / 0 / 2	-%	0 / 0 / 0
lib/arch/src/fake_host/simd_helpers.c	0.0%	0 / 0 / 59	0.0%	0 / 0 / 9	0.0%	0 / 0 / 14
lib/arch/src/include/simd_private.h	0.0%	0 / 0 / 6	0.0%	0 / 0 / 2	-%	0 / 0 / 0
lib/arch/src/pauth.c	72.3%	34 / 0 / 47	75.0%	3 / 0 / 4	0.0%	0 / 0 / 4
lib/arch/src/pmu.c	0.0%	0 / 0 / 104	0.0%	0 / 0 / 3	0.0%	0 / 0 / 80
lib/arch/src/simd.c	20.2%	45 / 0 / 223	42.9%	6 / 0 / 14	15.9%	21 / 0 / 132
lib/arch/src/vmid.c	81.7%	49 / 0 / 60	100.0%	5 / 0 / 5	40.7%	11 / 0 / 27
lib/common/include/bitmap.h	77.8%	7 / 0 / 9	100.0%	1 / 0 / 1	50.0%	2 / 0 / 4
lib/common/include/status.h	100.0%	14 / 0 / 14	100.0%	4 / 0 / 4	50.0%	1 / 0 / 2
lib/console/src/console.c	31.8%	7 / 0 / 22	33.3%	1 / 0 / 3	14.3%	2 / 0 / 14
lib/debug/src/backtrace.c	0.0%	0 / 0 / 4	0.0%	0 / 0 / 1	-%	0 / 0 / 0
lib/gic/include/gic.h	0.0%	0 / 0 / 3	0.0%	0 / 0 / 1	-%	0 / 0 / 0
lib/gic/src/gic.c	29.6%	50 / 0 / 169	33.3%	5 / 0 / 15	9.2%	7 / 0 / 76
lib/glob_data/src/glob_data.c	67.7%	113 / 0 / 167	90.0%	9 / 0 / 10	46.7%	28 / 0 / 60
lib/granule/include/dev_granule.h	0.0%	0 / 0 / 31	0.0%	0 / 0 / 5	0.0%	0 / 0 / 6
lib/granule/include/fake_host/granule_lock.h	50.0%	9 / 0 / 18	50.0%	2 / 0 / 4	25.0%	1 / 0 / 4

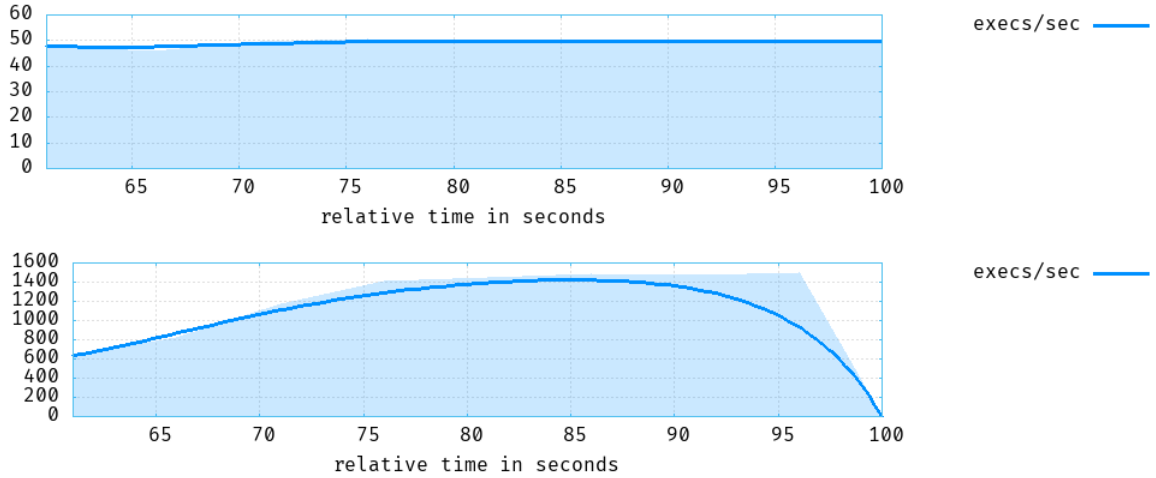
Persistent mode

“In persistent mode, AFL++ fuzzes a target multiple times in a single forked process, instead of forking a new process for each fuzz execution. This is the most effective way to fuzz,
as the speed can easily be x10 or x20 times faster without any disadvantages. All professional fuzzing uses this mode”

```
int main(int argc, char *argv[])
{
        VERBOSE("RMM: Beginning of Fake Host execution\n");
        init();
#ifdef PERSISTENT_MODE
        (void)argc;
        (void)argv;

        __AFL_INIT();

        unsigned char *buffer = __AFL_FUZZ_TESTCASE_BUF;

        /* Starting AFL_LOOP */
        while (__AFL_LOOP(10000)) {
                size_t len = __AFL_FUZZ_TESTCASE_LEN;
                /* INFO("Loop: %d Len: %lu\n", loop++, len); */

                if (len < MINIMUM_LENGTH_FOR_FUZZING) {
                        continue;
                }
                execute(buffer, len);
        }
#else #ifdef PERSISTENT_MODE
        unsigned char buffer[4096] = { 0 };
        size_t read_res = readCorpus(argc, argv, buffer);

        execute(buffer, read_res);
#endif #ifdef PERSISTENT_MODE #else

        VERBOSE("RMM: Fake Host execution completed\n");

        return 0;
}
```

Persistent mode stats

Faster exec = more exec in same duration =
more test inputs = better probability of new edges




```
AFL ++4.32c {s-demo3} (...fx/rmm/_Fuzz/builds/demo3/Debug/rmm.elf) [exploit]
┌─ process timing ──────────────────────┐┌─ overall results ──────┐
│        run time : 0 days, 0 hrs, 0 min, 7 sec ││  cycles done : 0       │
│   last new find : 0 days, 0 hrs, 0 min, 0 sec ││ corpus count : 248     │
│last saved crash : none seen yet       ││saved crashes : 0       │
│ last saved hang : none seen yet       ││  saved hangs : 0       │
├─ cycle progress ──────────────────────┤├─ map coverage ─────────┤
│  now processing : 7.0 (2.8%)          ││   map density : 2.95% / 9.08%
│  runs timed out : 0 (0.00%)           ││count coverage : 2.54 bits/tuple
├─ stage progress ──────────────────────┤├─ findings in depth ────┤
│  now trying : user extras (over)      ││favored items : 5 (2.02%)
│ stage execs : 3111/22.3k (13.95%)     ││ new edges on : 157 (63.31%)
│total execs : 17.4k                    ││total crashes : 0 (0 saved)
│  exec speed : 2584/sec                ││ total tmouts : 0 (0 saved)
├─ fuzzing strategy yields ─────────────┤└─ item geometry ─────────┤
│   bit flips : 59/1416, 31/1415, 16/1413 │  levels : 2            │
│  byte flips : 2/177, 1/176, 0/174     │  pending : 245         │
│ arithmetics : 65/11.6k, 2/20.2k, 0/18.4k │ pend fav : 4         │
│  known ints : 0/1294, 0/5590, 4/8552  │own finds : 240         │
│  dictionary : 0/0, 0/0, 0/0, 0/0      │  imported : 0          │
│havoc/splice : 0/100, 0/0              │stability : 99.56%      │
│py/custom/rq : unused, unused, unused, unused │                  │
│    trim/eff : 10.84%/83, 69.49%       │                  [cpu000:  4%]
└─ strategy: exploit ──────── state: started :-) ─┘
```

100 second fuzzing
non-persistent vs persistent:

	Coverage	Exec	Excl	Total
Lines:	41.4%	6781	0	16381
Functions:	51.1%	726	0	1422
Branches:	25.9%	2100	0	8106

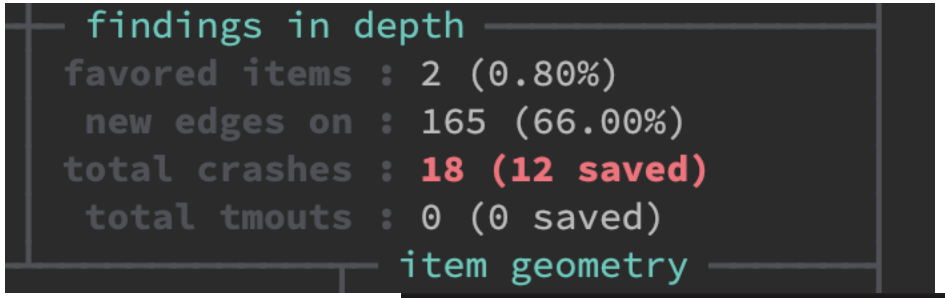
	Coverage	Exec	Excl	Total
Lines:	45.0%	7375	0	16381
Functions:	53.3%	758	0	1422
Branches:	30.6%	2481	0	8106

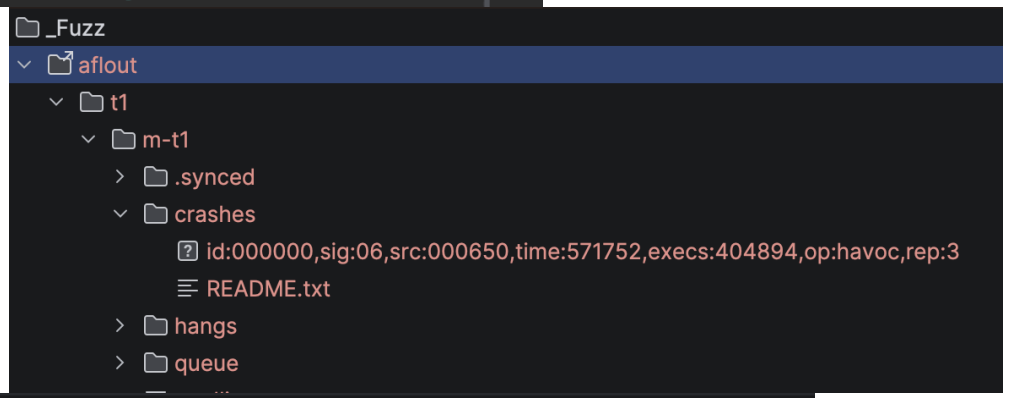
Interpreting Crashes

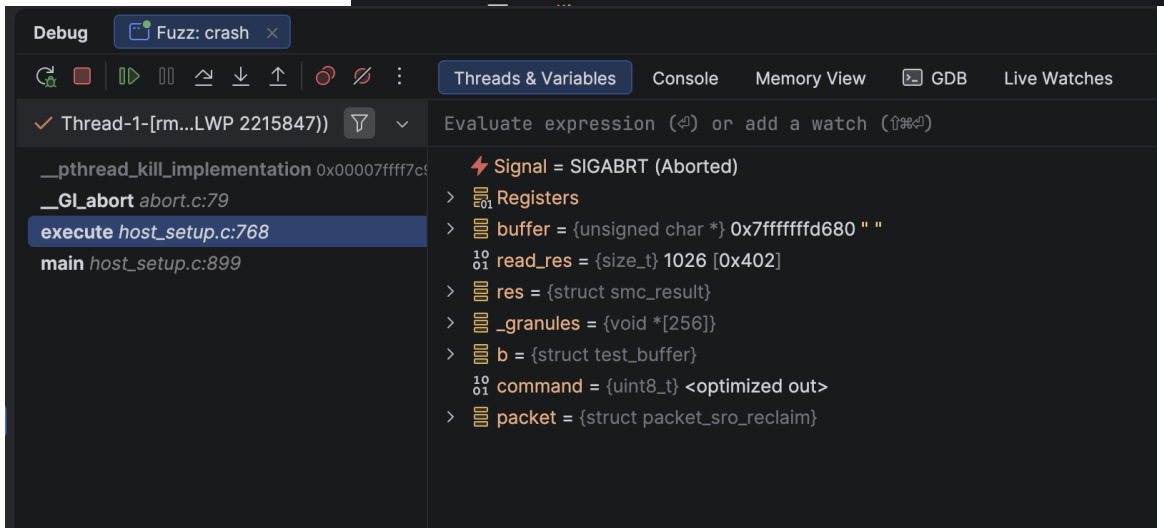
- Reproduce crash
  1. Compile non-persistent rmm.elf
  2. Feed corpus to rmm.elf
  3. Observe logs
  4. Debug

- If can’t reproduce, probably leaking states
  - Enhance reset logic

- Is it in RMM logic or fake_host harness code?





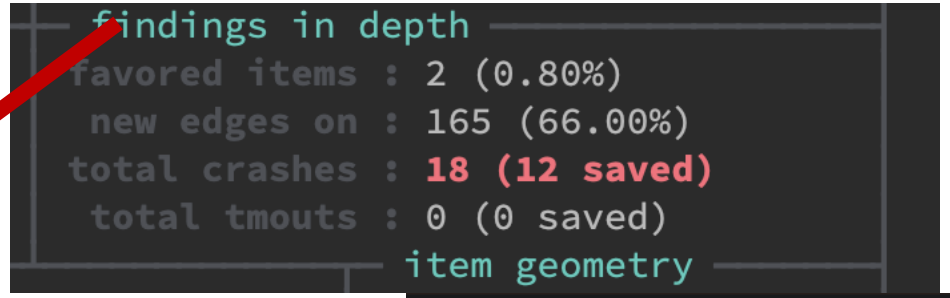


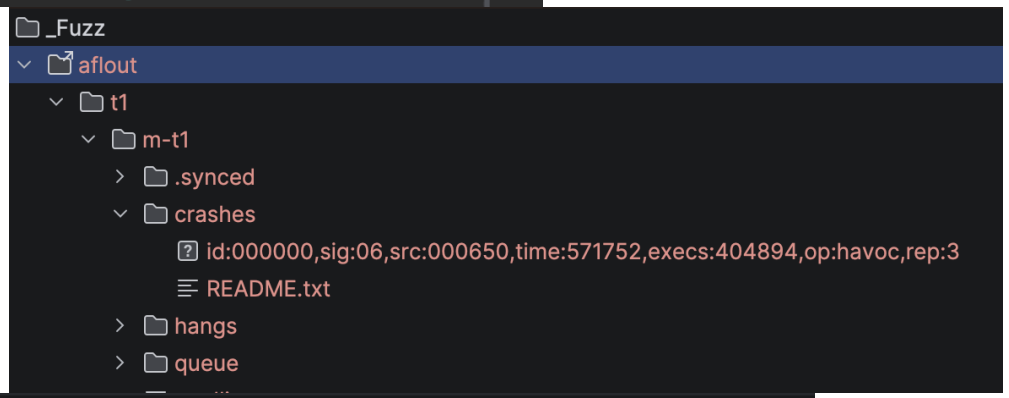
Interpreting Crashes

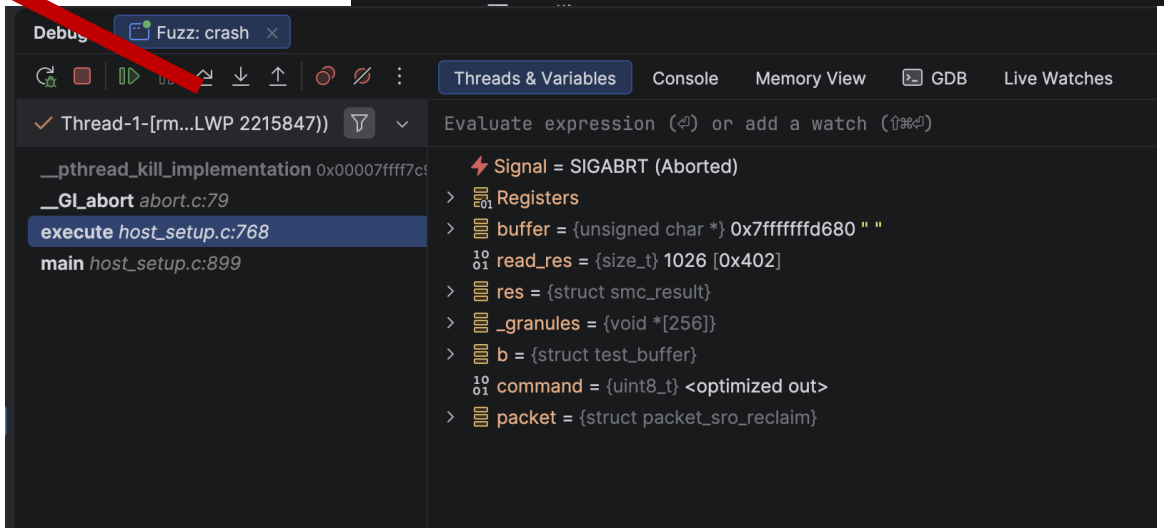
- Reproduce crash
  1. Compile non-persistent rmm.elf
  2. Feed corpus to rmm.elf
  3. Observe logs
  4. Debug

- If can’t reproduce, probably leaking states
  - Enhance reset logic

- Is it in RMM logic or fake_host harness code?

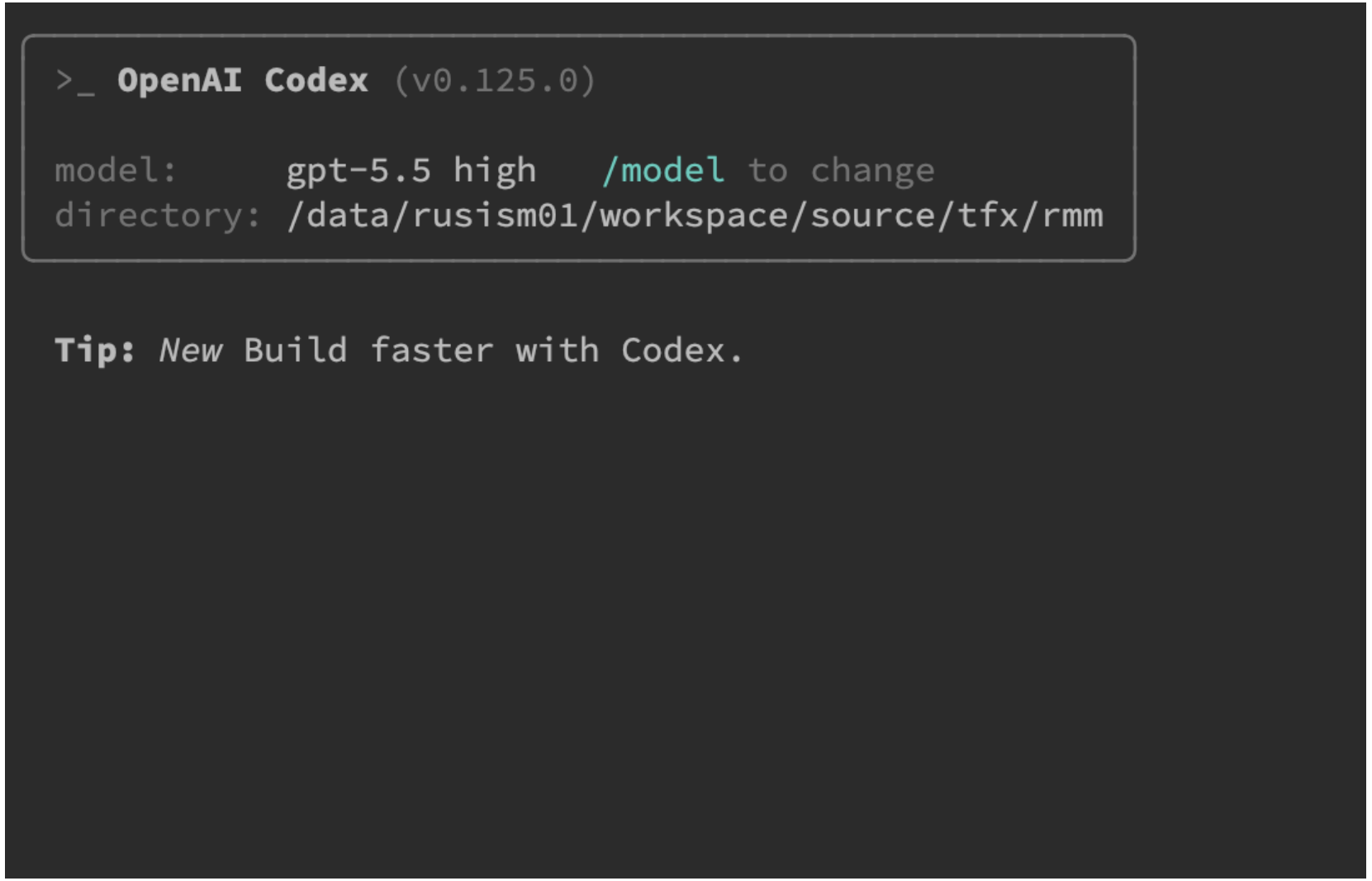






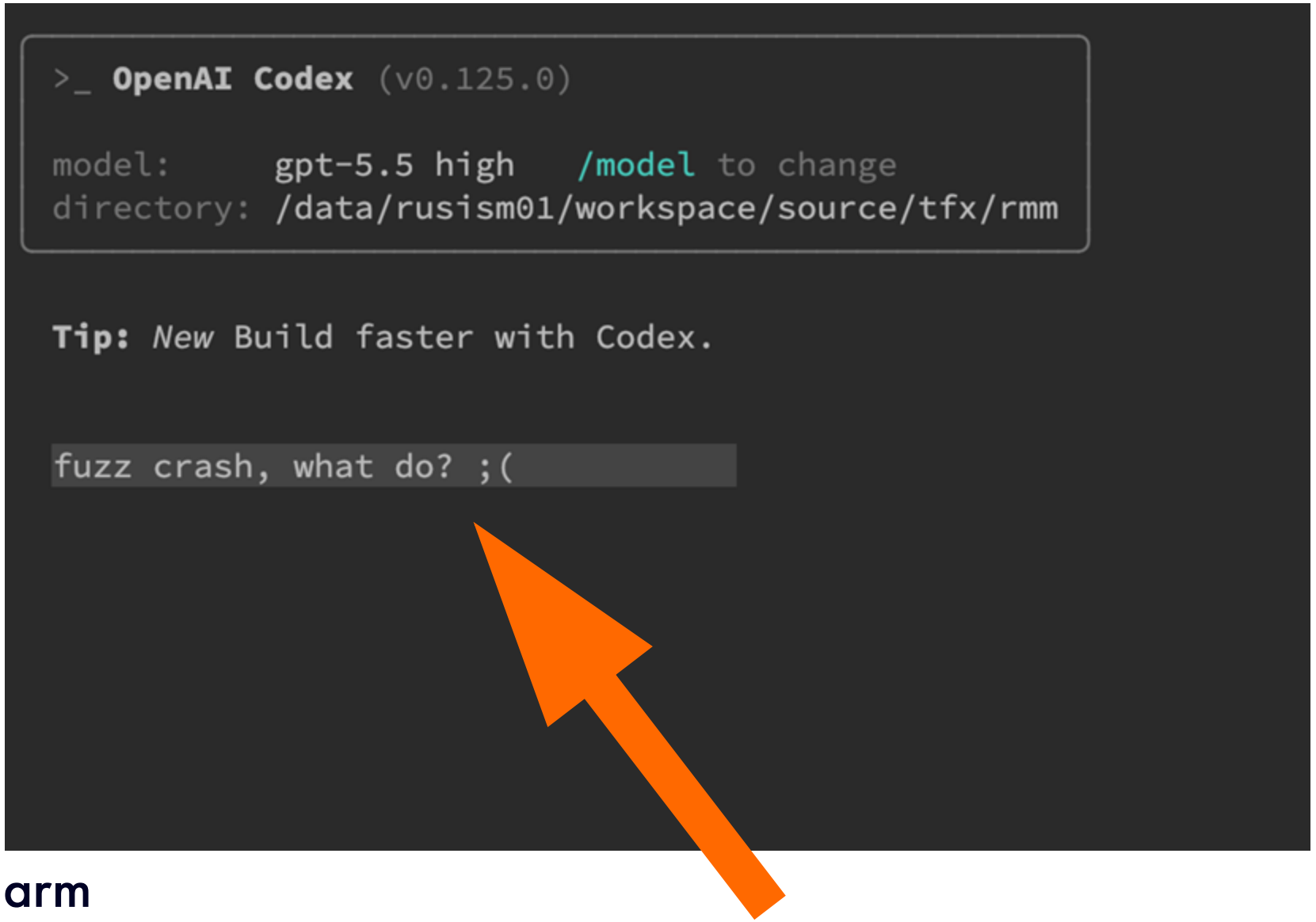
Interpreting Crashes

Alternative approach



Interpreting Crashes

Alternative approach



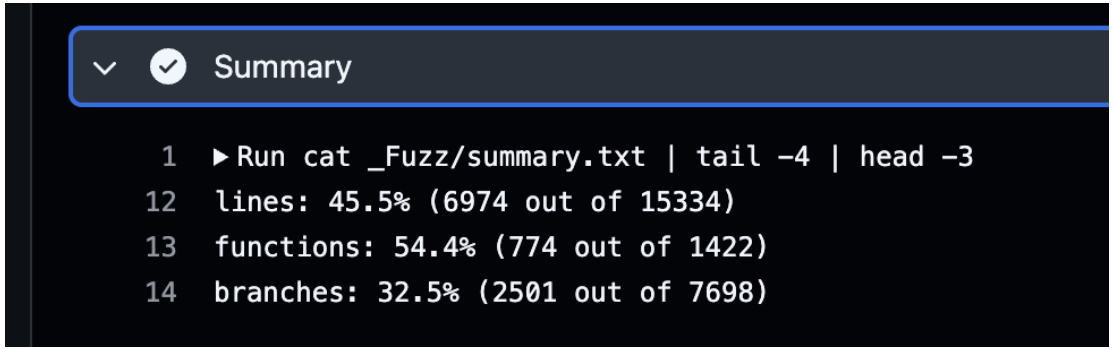
Interpreting Crashes

Explanation of recently observed crash from codex

```
• Reproduced in non-persistent mode with:

  ./build-f-np/Debug/rmm.elf _Fuzz/aflout/t1/m-t1/crashes/id:000000,sig:06,src:000650
,time:571752,execs:404894,op:havoc,rep:3

  The crash is a fake-host slot mapping bug exposed by this corpus, not caused by the
 earlier invalid RMI_VERSION calls.

  The important sequence is:

  REC_CREATE -> RMI_INCOMPLETE, asks for REC aux donation
  SroDonate(count = 0xa) -> donates 10 aux granules, still incomplete
  AllocateGranule(index = 0xc) -> consumes one granule in between
  SroDonate(count = 0x0) -> donates remaining 6 aux granules
  SroContinue(flags = 0) -> REC_CREATE continues and crashes

  REC_CREATE explicitly does not require contiguous donated aux memory: see runtime/r
mi/rec.c:823. But fake host maps buffers by returning the physical host address
  directly: plat/host/host_fuzz/src/host_harness.c:12.

  That works only when all aux granules are physically contiguous. Here they are not:
 the extra AllocateGranule(index=0xc) between the two donate calls creates a hole.
  The first 10 donated aux pages are contiguous, one unrelated page is allocated, the
n the final 6 donated aux pages continue after the hole.
```


CI Integration

- Existing workflow: .github/workflows/fuzz_rmm.yml
  - Builds fuzz target in AFL++ container
  - Runs parallel fuzz workers for 5min
  - Generates coverage
  - Archives fuzzing artifacts



- TF-CI job (WIP):
  - Run for longer? (15-30 mins)
  - Fail if coverage drop below XY%
  - Fail if crashes observed

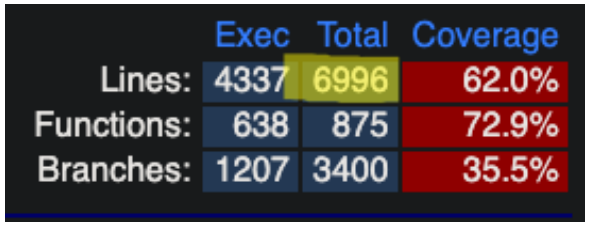
Limitations

- fake_host is not hardware-accurate
- Some failures may be harness artifacts
- Coverage is not proof of correctness
- Spec changes break fuzzing
- Deep protocol states still need good seed design

todo;

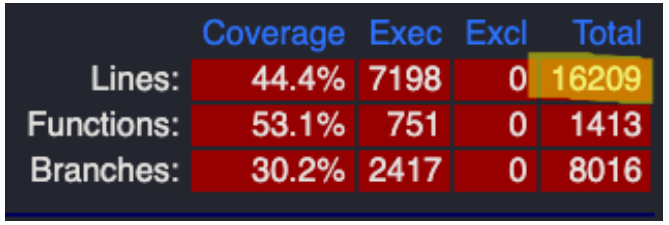
- Enable RSI fuzzing
- Enable PSCI fuzzing
- Check seeds for output
- Debug reported instability during calibration
- Slim the reset phase
- Refine crash triage process
  - Deduplicate
  - Minimize
  - Reproduce in CI(?)
- Custom mutator
- …
- …

- Consider FVP fuzzing?



	Exec	Total	Coverage
Lines:	4337	6996	62.0%
Functions:	638	875	72.9%
Branches:	1207	3400	35.5%

2025 (RMI+RSI+PSCI)



	Coverage	Exec	Excl	Total
Lines:	44.4%	7198	0	16209
Functions:	53.1%	751	0	1413
Branches:	30.2%	2417	0	8016

2026 (RMI Only)

Contributors

- Armando Miraglia <armax@google.com>
- Rustam Ismayilov <rustam.ismayilov@arm.com>
- Soby Mathew <soby.mathew@arm.com>

Previous work:

- Imre Kis <imre.kis@arm.com>
- Shale Xiong <shale.xiong@arm.com>

arm

Merci
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Thank You
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה
ధన్యవాదములు
Köszönöm

arm

EXTRAS

Scripts

Fuzz in FVP?

fuzz.env and multirun.py

- tools/fuzz/fuzz.env
  - EZ build
  - Example use: fuzzbuild 1 demo
  - Suggests commands after build

- multirun.py
  - Script for multicore fuzzing
  - Mix various scenarios
  - Used in GitHub Actions

```
[100%] Generating rmm.elf
[100%] Built target rmm

Built with -DPERSISTENT_MODE
Suggested AFL commands:
afl-fuzz -i "_Fuzz/builds/demo/smc_corpus" -o "_Fuzz/aflout/demo/" -M "m-demo" -- "./_Fu
afl-fuzz -i "_Fuzz/builds/demo/smc_corpus" -o "_Fuzz/aflout/demo/" -S "s-demo" -- "./_Fu

Fuzz for 1000 seconds:
cmake --build "_Fuzz/builds/demo" -- run-fuzzer
Benchmark for 100 seconds and generate reports:
cmake --build "_Fuzz/builds/demo" -- quick-bench

Execute with default corpus (Only for non-persistent mode):
./_Fuzz/builds/demo/Debug/rmm.elf "_Fuzz/builds/demo/smc_corpus/default.bin"

Generate coverage report:
cmake --build "_Fuzz/builds/demo" -- run-coverage
```

```
Usage: multirun.py [OPTIONS] COMMAND [ARGS]...

Options:
  -h, --help  Show this message and exit.

Commands:
  bench-corpora  Benchmark corpora
  bench-sched    Benchmark schedules
  get-coverage   Get coverage report for specified build variants
  get-plots      Get afl-plot for specified build variants
  multibuild     Build multiple targets (consecutively)
  spawn          Spawn multiple fuzzers
```