Human-Controlled Fuzzing with AFL

Maxim Grishin

Igor Korkin

2022

WHO WE ARE

Maxim Grishin

- Bachelor of Information Security
- National Research Nuclear University MEPhI
- Cryptology and Cybersecurity Department

Igor Korkin, PhD

- Independent Security Researcher
- Speaker at CDFSL, BlackHat, HITB, SADFE
- sites.google.com/site/igorkorkin

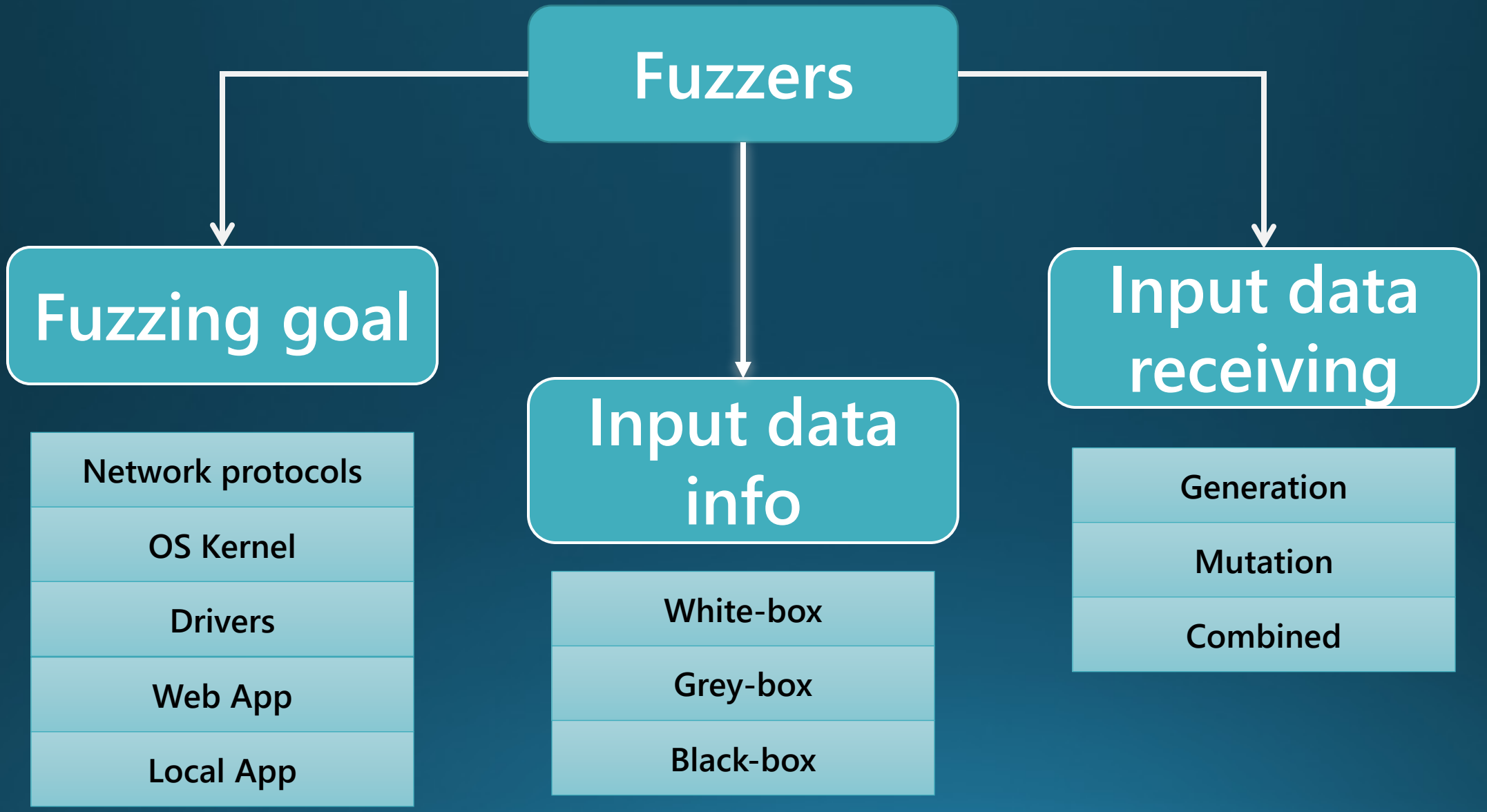
AGENDA

- Fuzzing and classification of fuzzers
- The main fuzzing strategy
- Fuzzing with AFL. The problem of analyzing identical code sections
- The proposed tool. The algorithm and details of the work
- Execution traces registration
- Test results
- Future plans

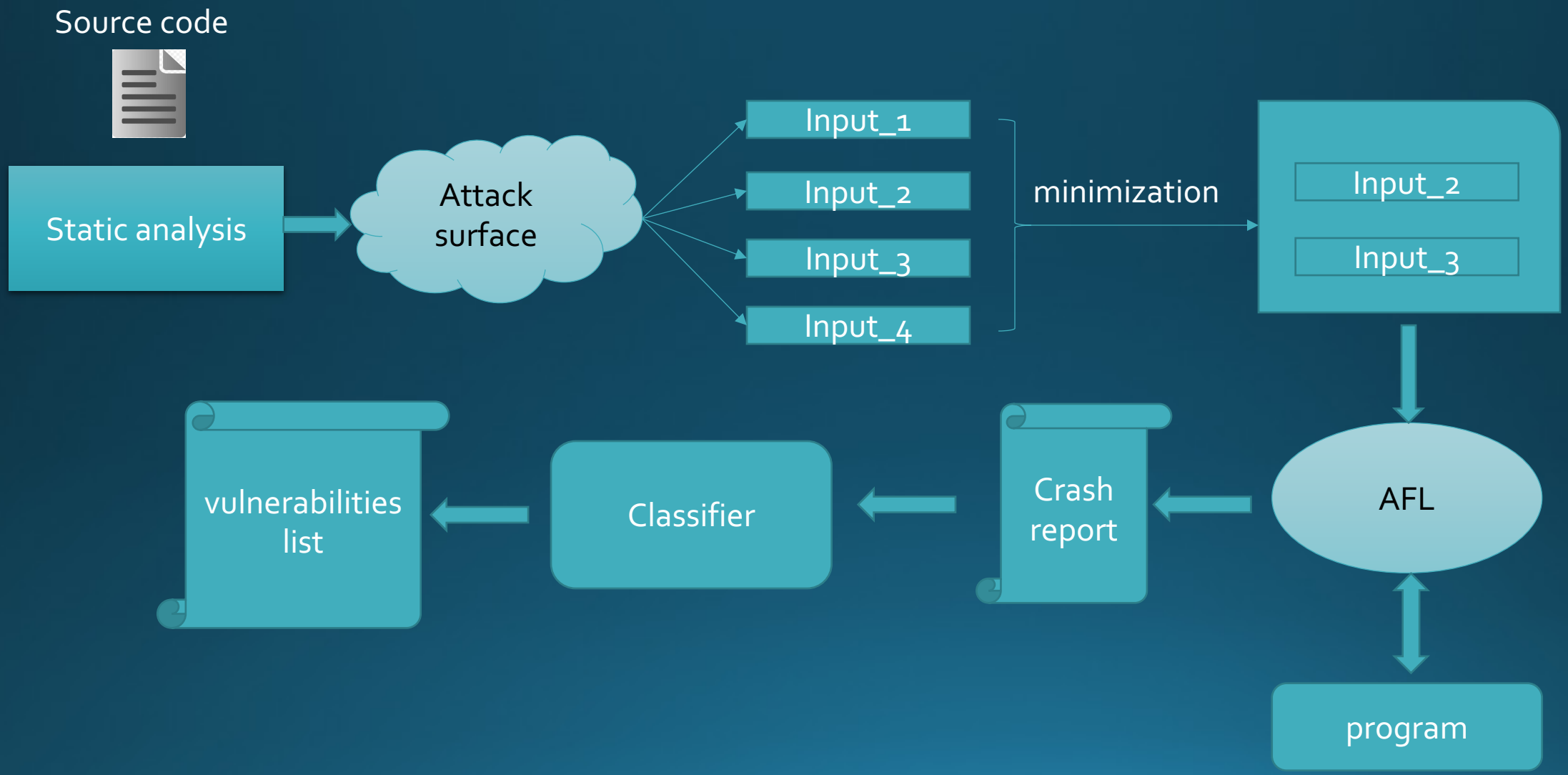
WHAT IS FUZZING?

- Fuzzing is a technique of automated testing when a program receives specially modified, incorrect data.

EXE

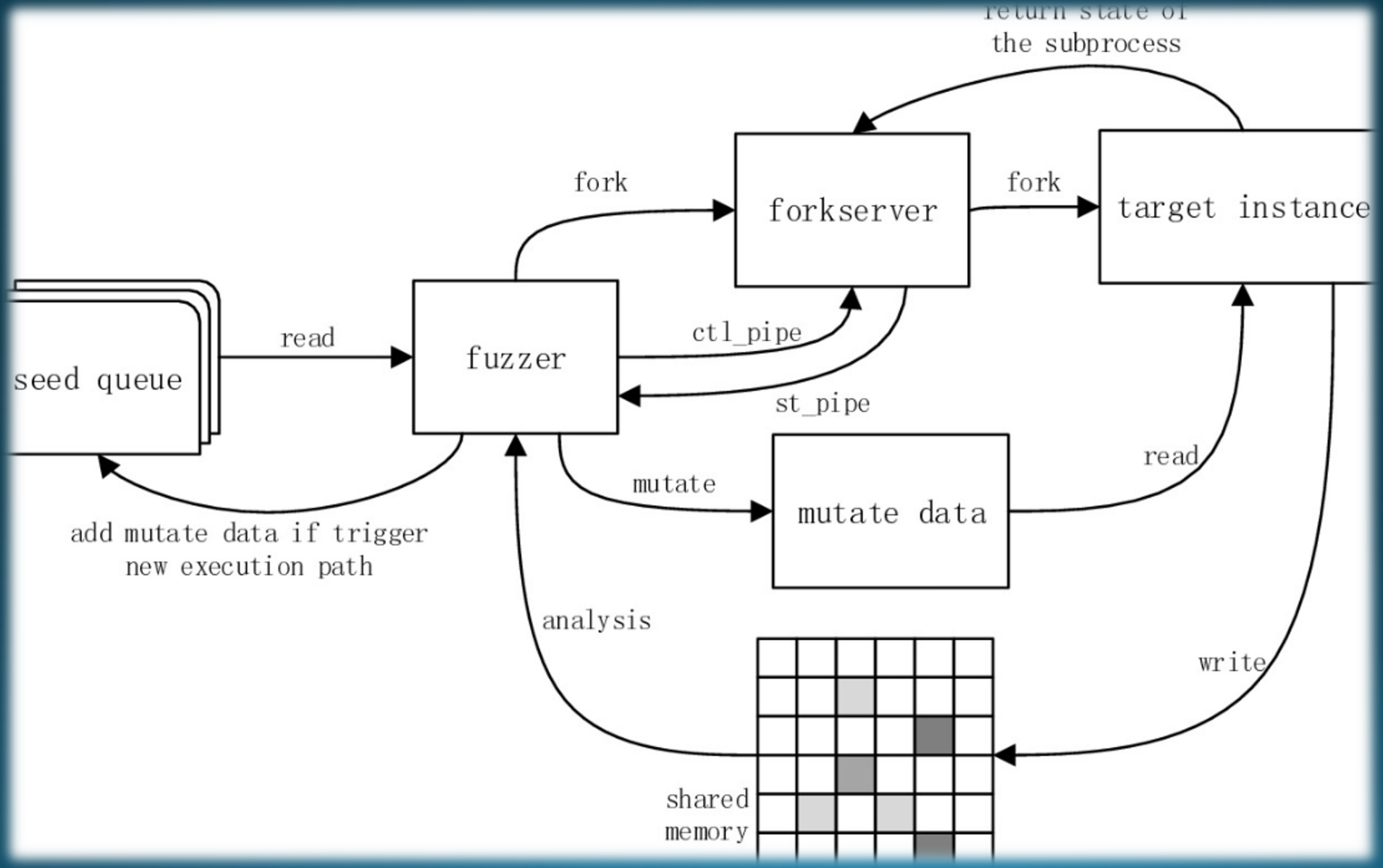

Fuzzers
Fuzzing goal
Input data info
Input data receiving
Network protocols
OS Kernel
Drivers
Web App
Local App
White-box
Grey-box
Black-box
Generation
Mutation
Combined

FUZZING STRATEGY

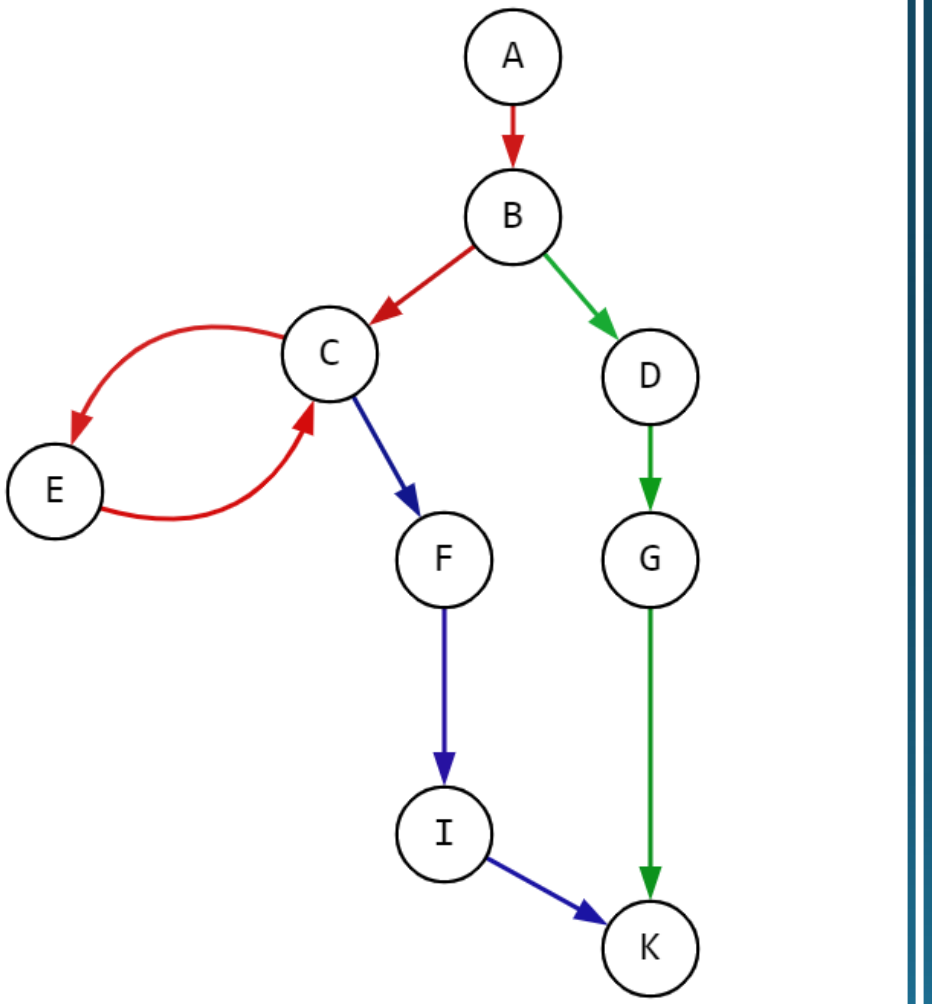


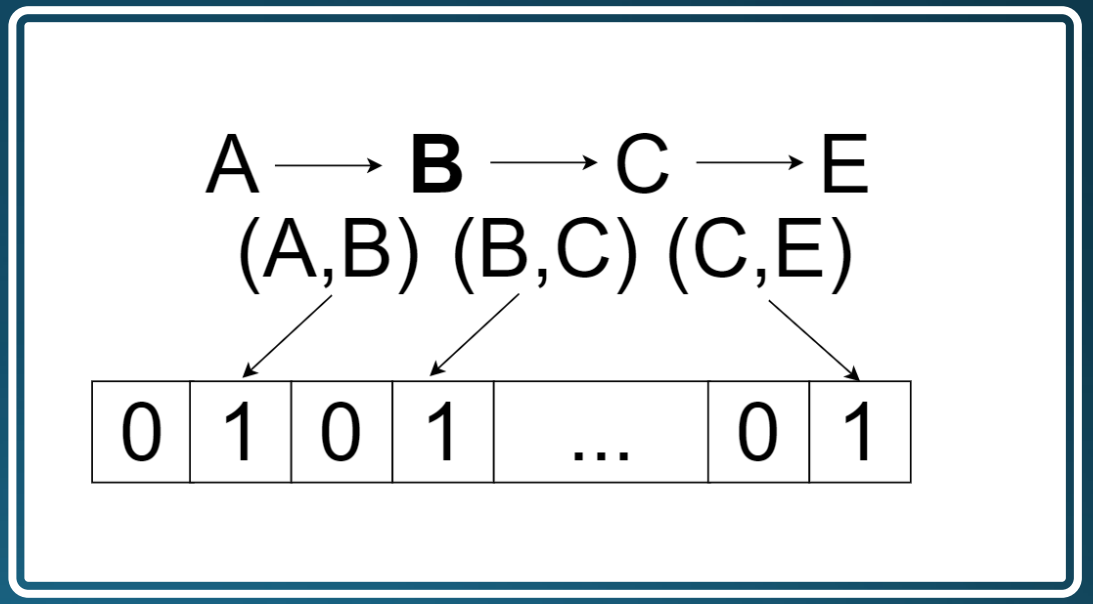
American Fuzzy Lop

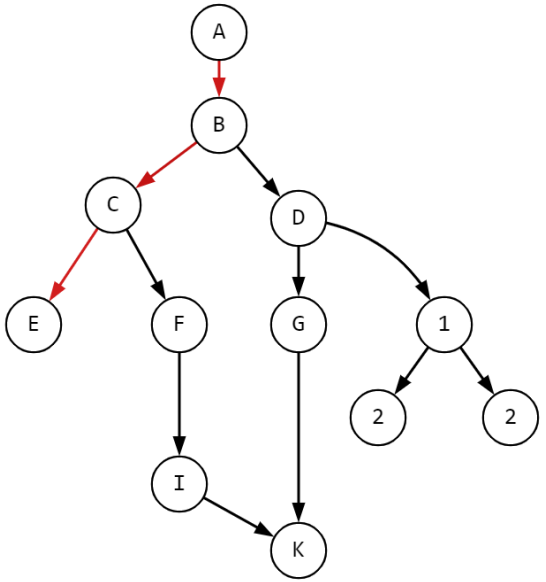
AFL – PROCESS STANDARD CYCLE



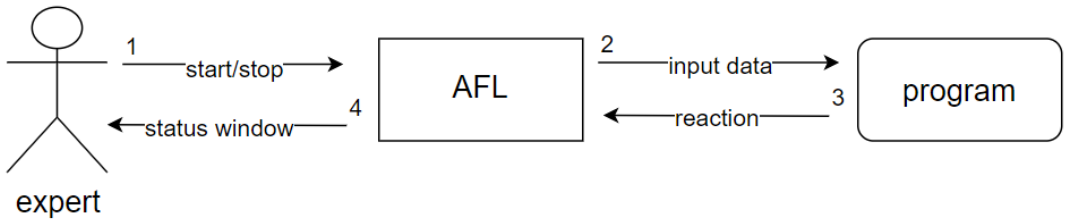
EXECUTION TRACE



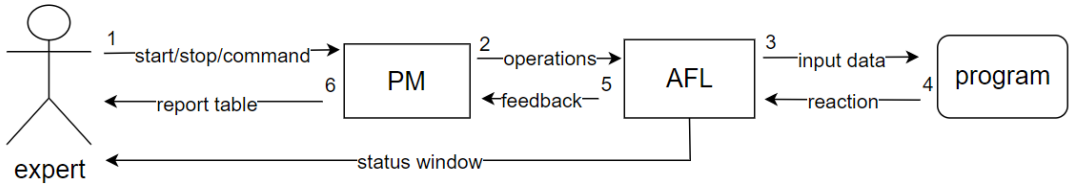




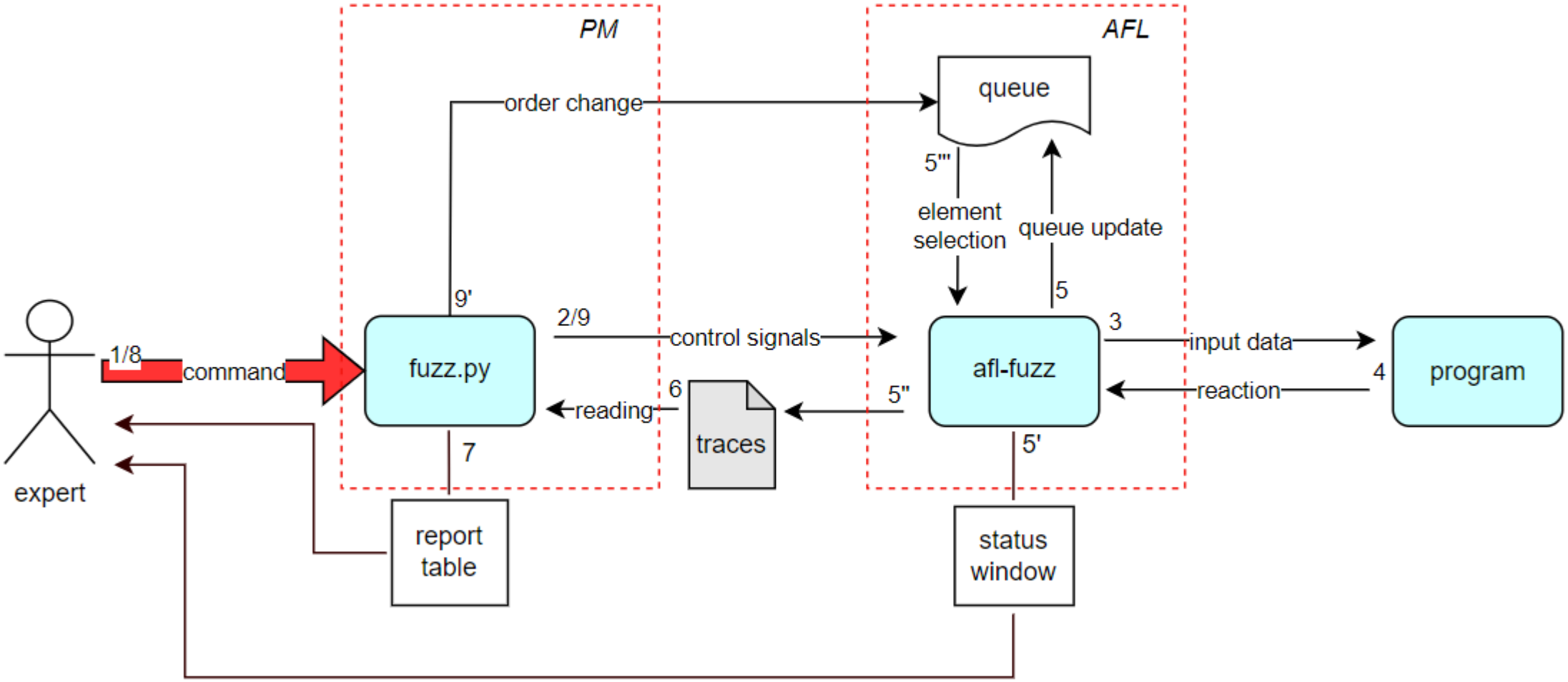
(A,B) (B,C) (C,E)
(A,B) (B,C) (C,E)
...
(A,B) (B,C) (C,E)

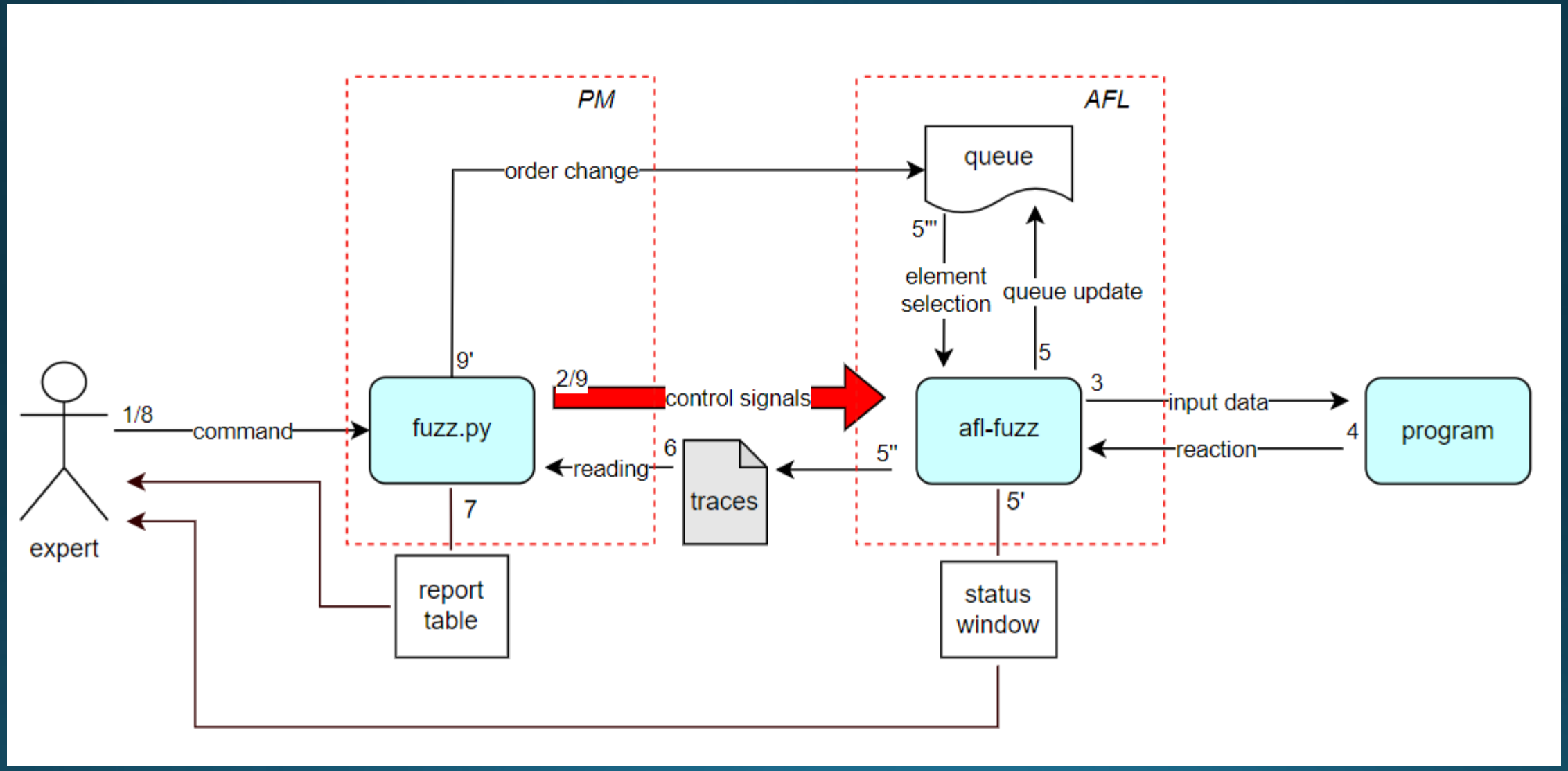


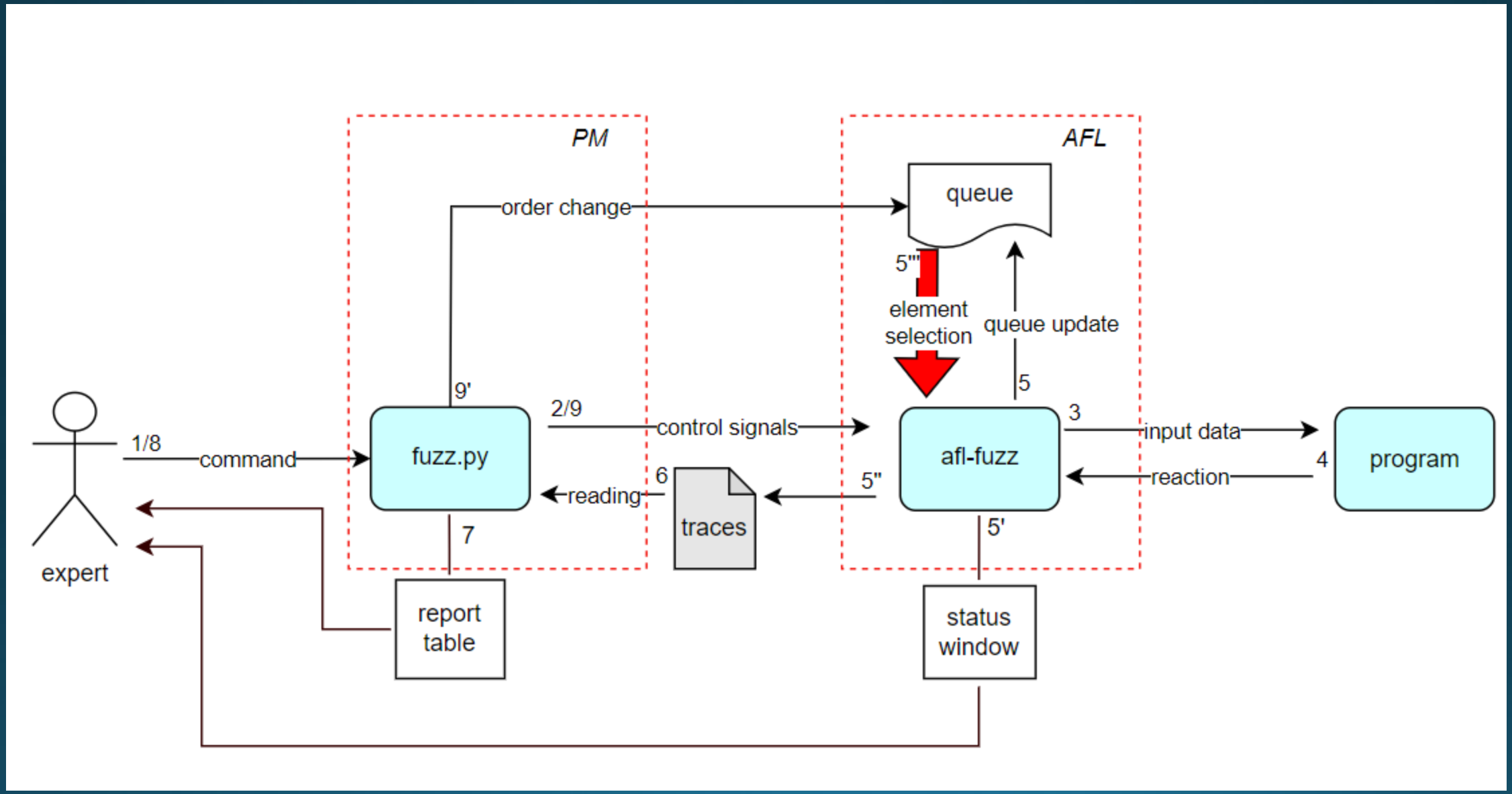
Classic AFL scheme

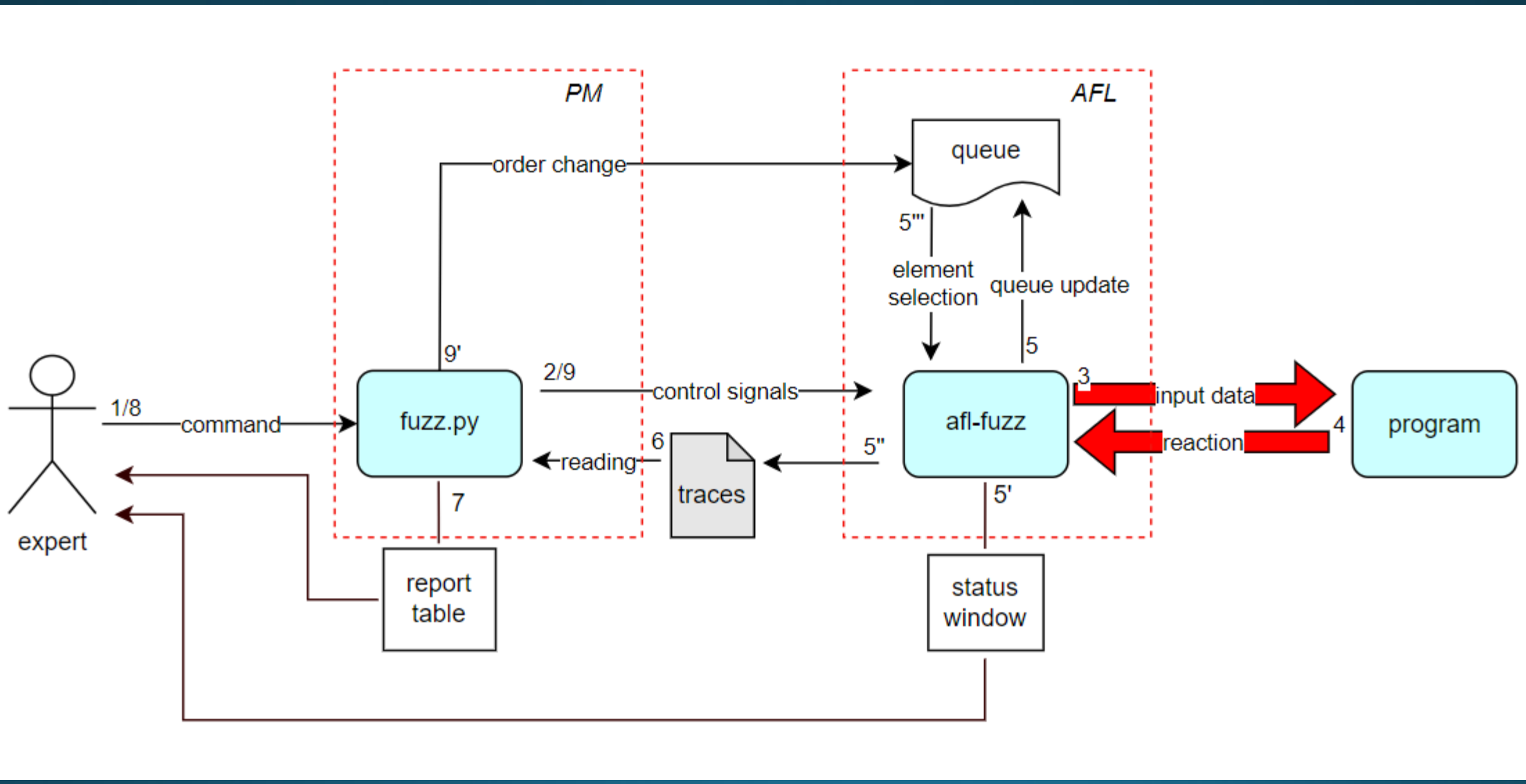


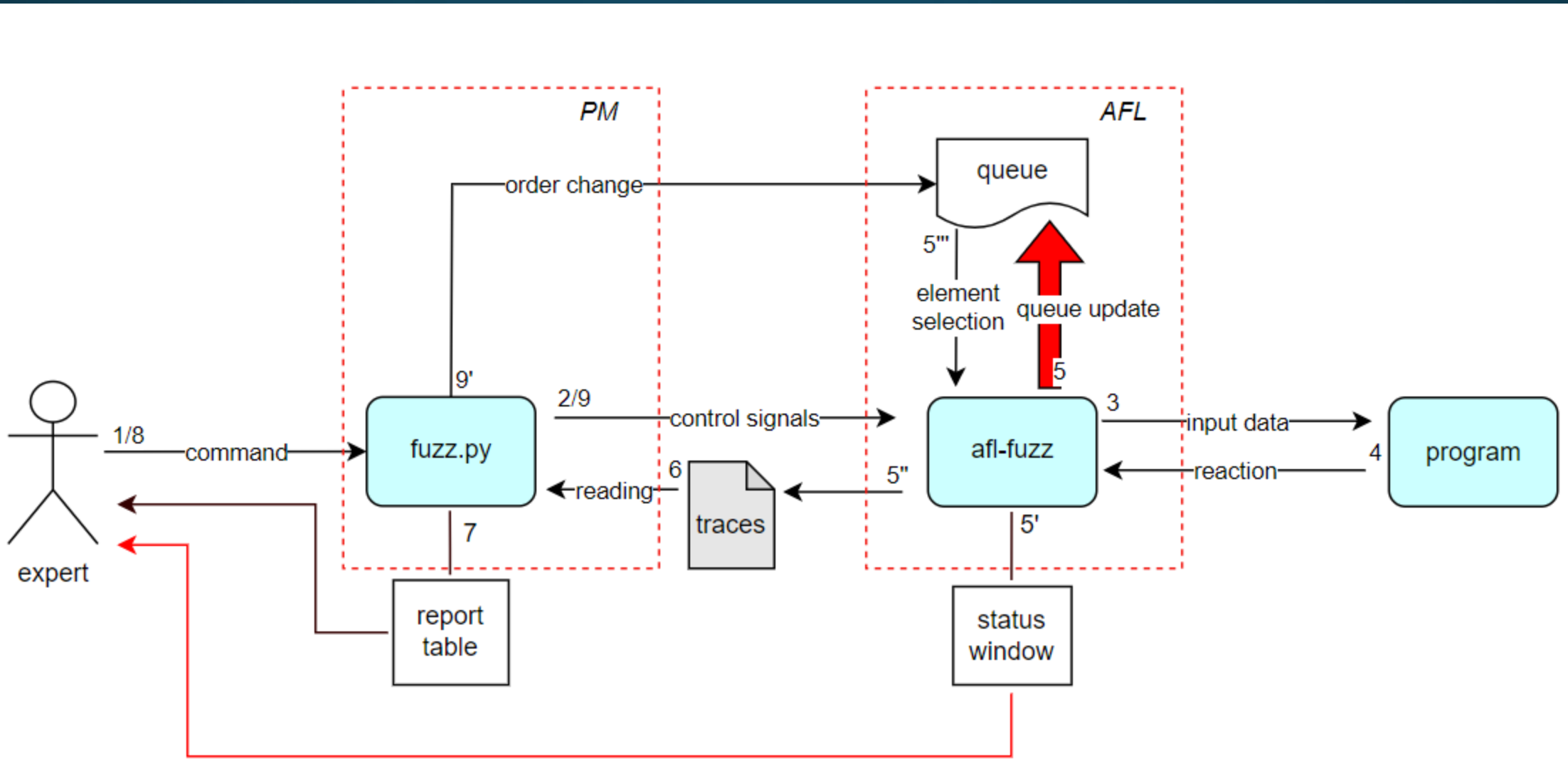
Proposed fuzzing system scheme

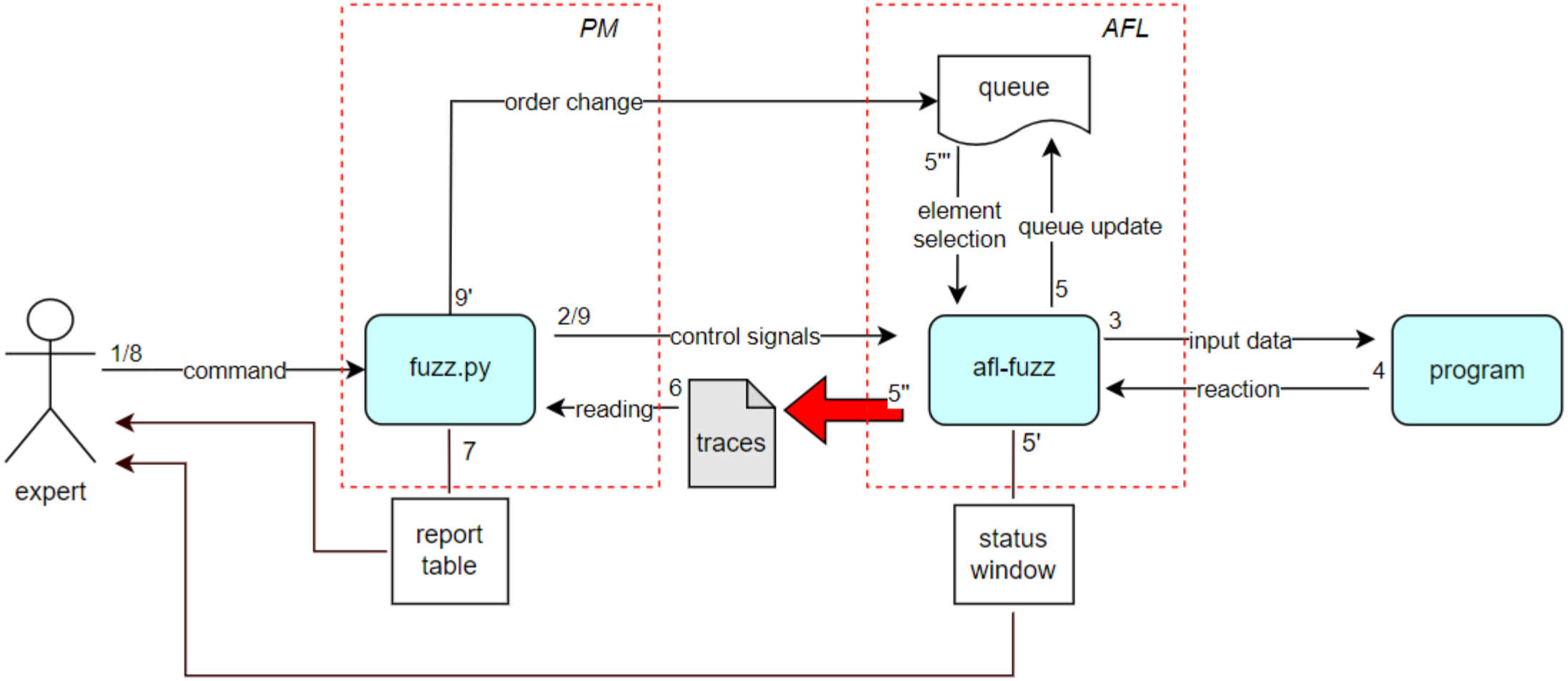

PM
AFL
queue
order change
5'''
element selection
queue update
5
9'
2/9
control signals
3
input data
1/8
command
fuzz.py
afl-fuzz
program
4
reaction
6
reading
5''
traces
7
5'
expert
report table
status window

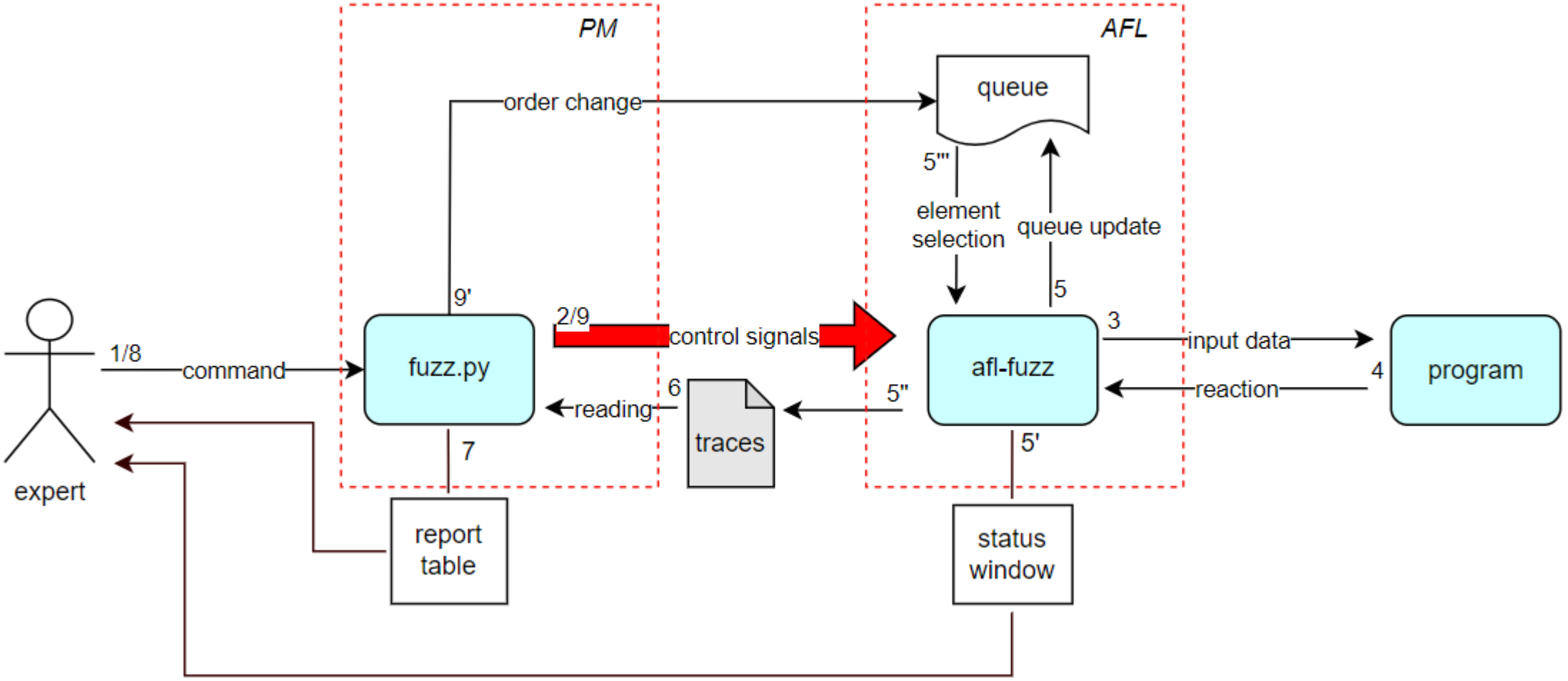

PM
AFL
queue
order change
5'''
element selection
queue update
5
9'
2/9
control signals
3
input data
4
program
1/8
command
fuzz.py
afl-fuzz
reaction
6
reading
traces
5''
7
5'
expert
report table
status window

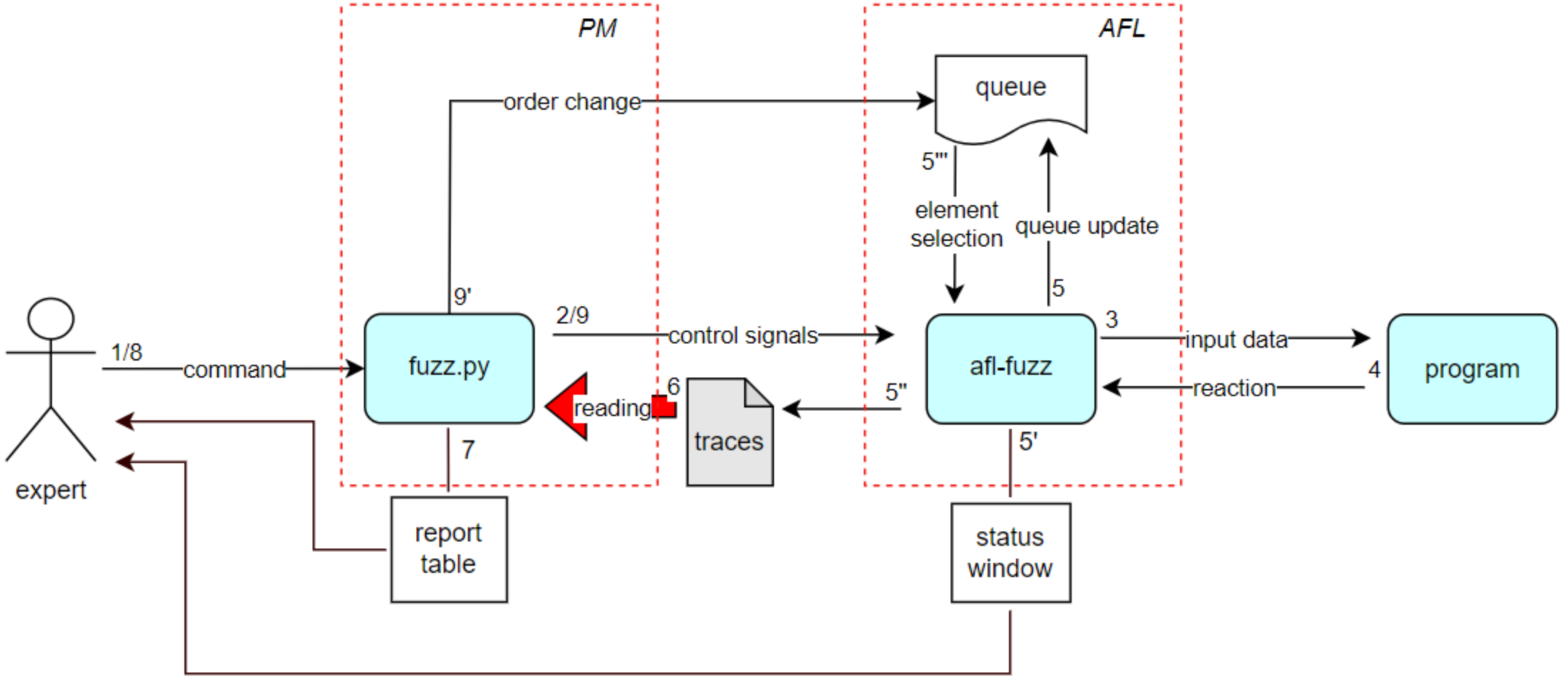

PM
AFL
queue
order change
5'''
element selection
queue update
5
9'
2/9
control signals
3
input data
4
program
1/8
command
fuzz.py
afl-fuzz
6
reading
traces
5"
reaction
7
5'
expert
report table
status window

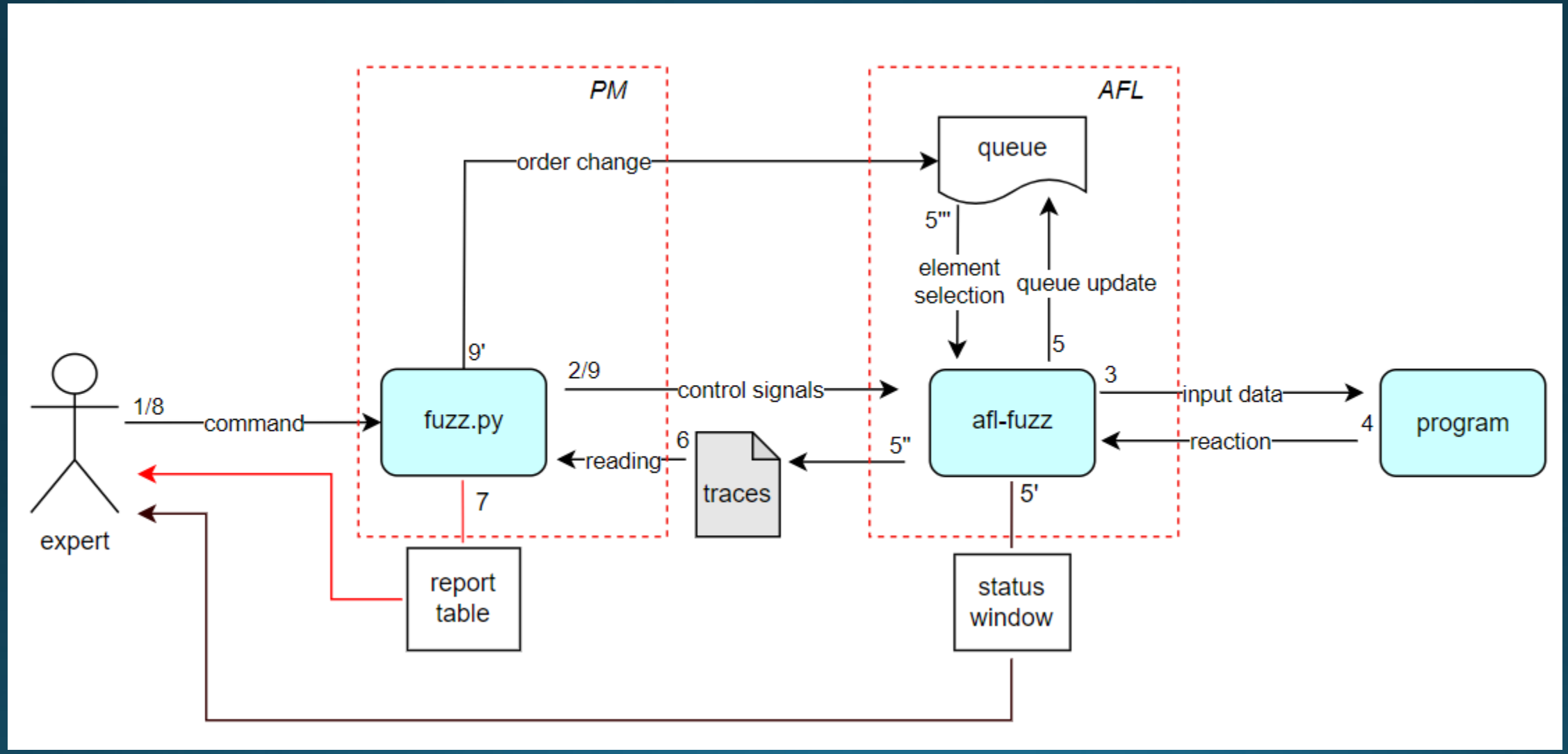

PM
AFL
queue
order change
5'''
element selection
queue update
5
9'
2/9
control signals
3
input data
4
program
1/8
command
fuzz.py
afl-fuzz
6
reading
traces
5''
reaction
7
5'
expert
report table
status window

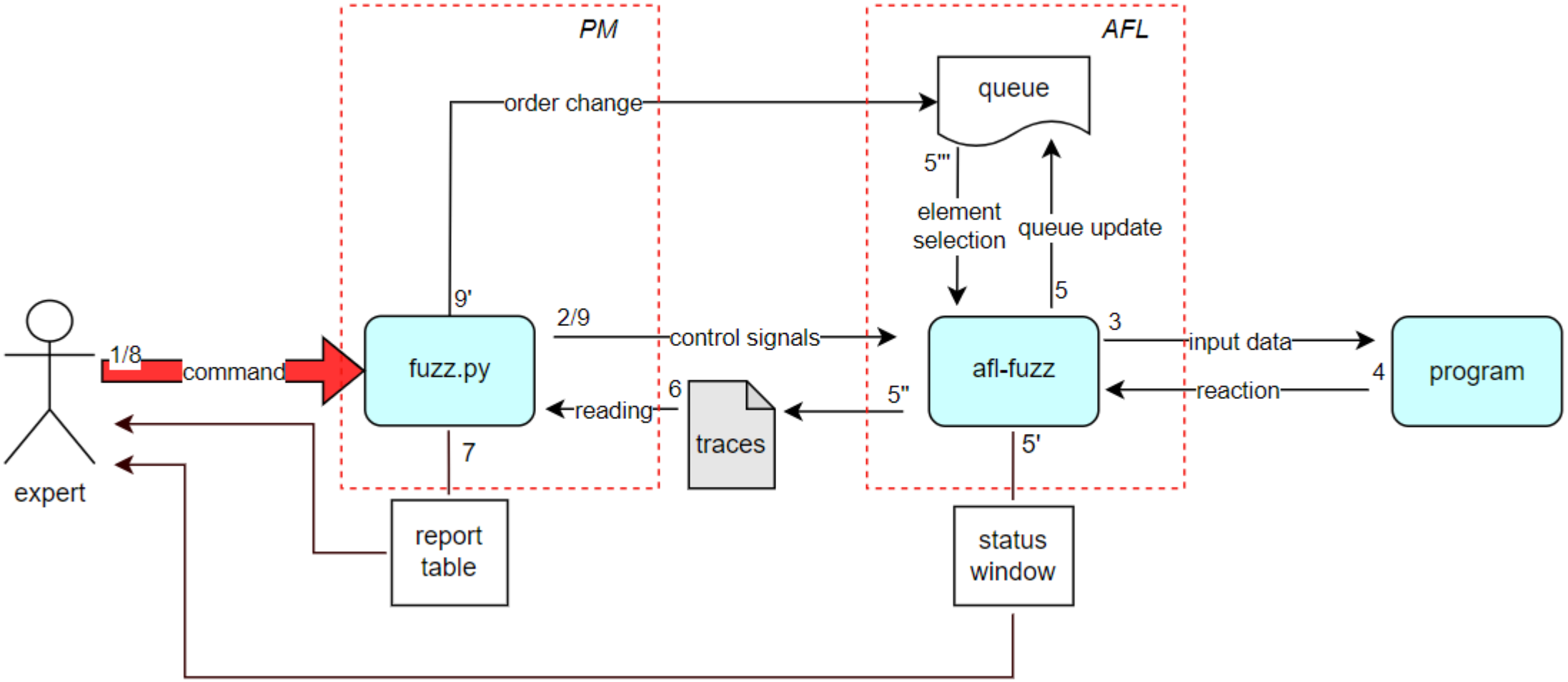

PM
AFL
queue
order change
5'''
element selection
queue update
5
9'
2/9
control signals
3
input data
4
program
1/8
command
fuzz.py
afl-fuzz
reaction
6
reading
traces
5"
7
5'
expert
report table
status window

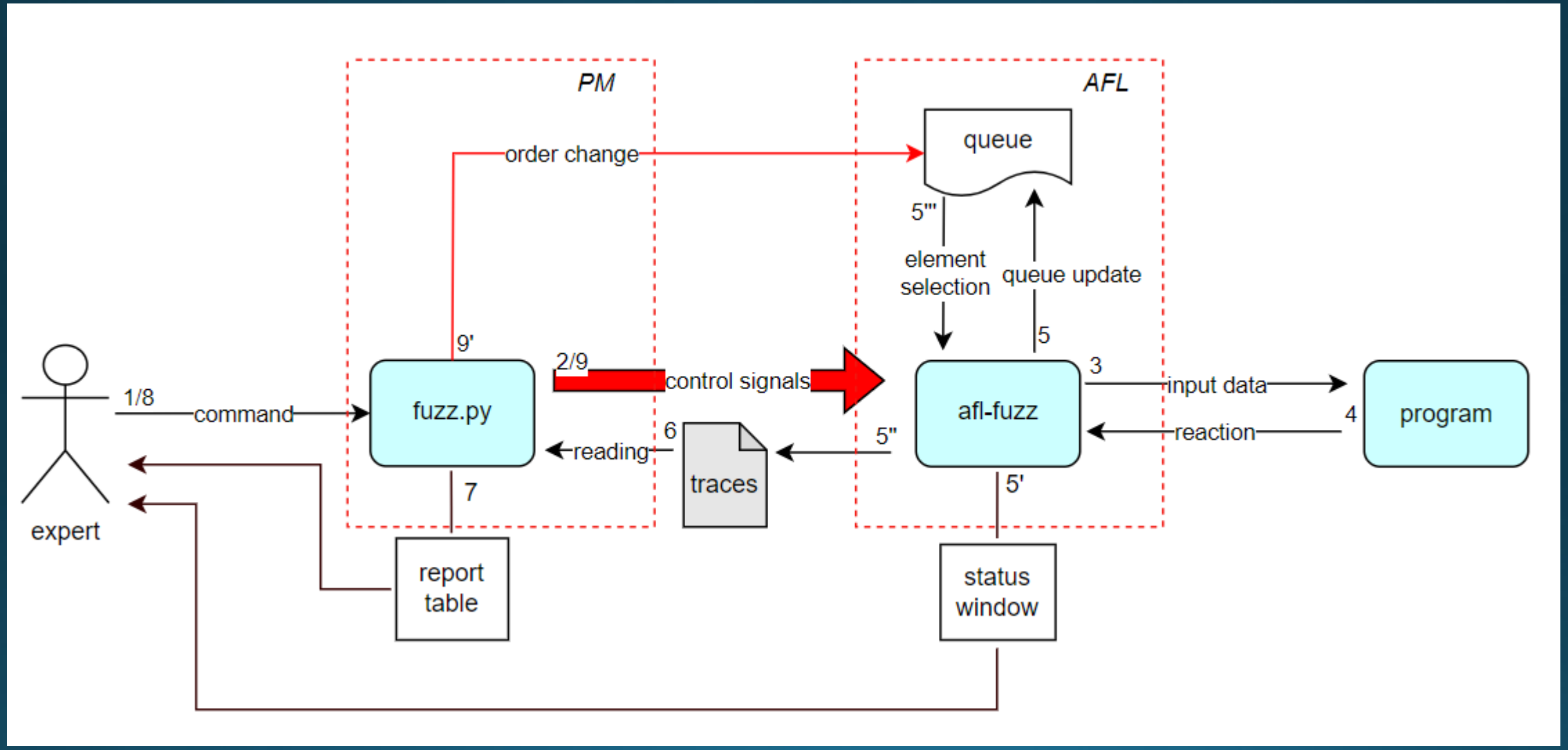

PM
AFL
queue
order change
5'''
element selection
queue update
5
9'
2/9
control signals
3
input data
1/8
command
fuzz.py
afl-fuzz
program
4
reaction
6
reading
traces
5"
7
5'
expert
report table
status window


PM
AFL
queue
order change
5'''
element selection
queue update
5
9'
2/9
control signals
3
input data
1/8
command
fuzz.py
afl-fuzz
program
4
reaction
6
reading
traces
5''
7
5'
expert
report table
status window

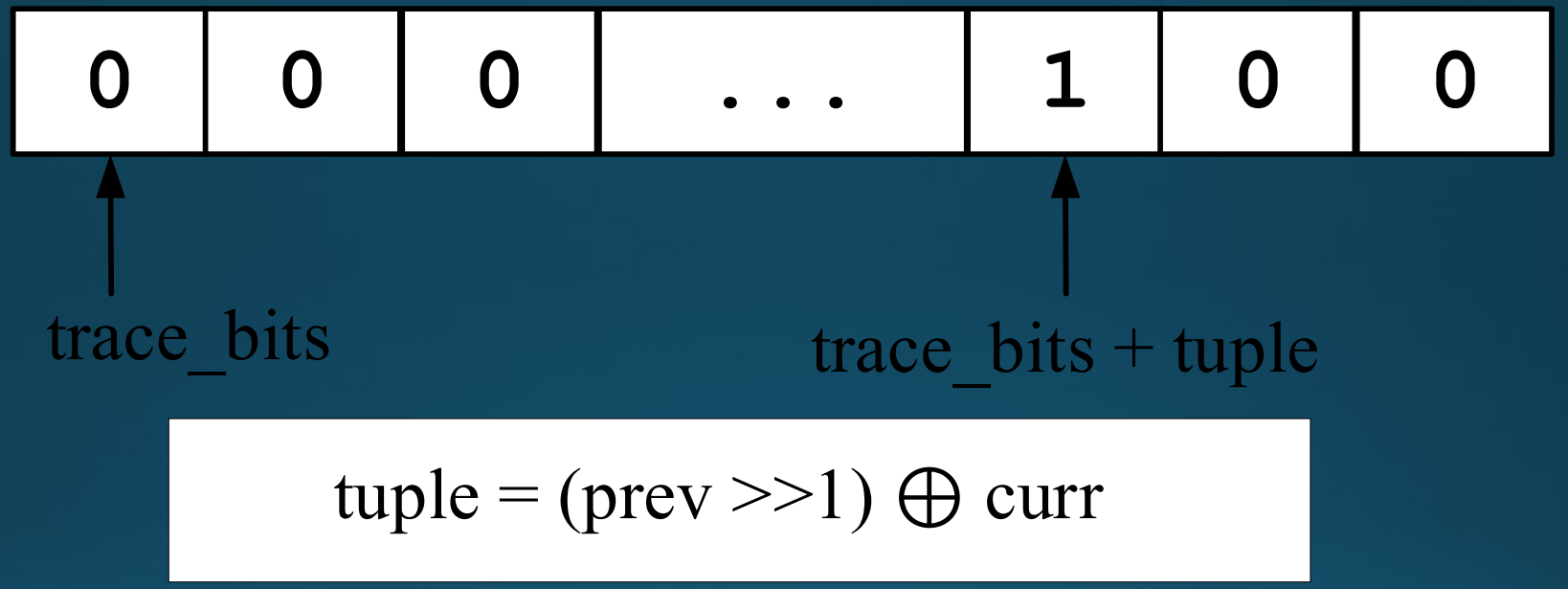

PM
AFL
queue
order change
5'''
element selection
queue update
5
9'
2/9
control signals
3
input data
1/8
command
fuzz.py
afl-fuzz
program
4
6
reading
traces
5''
reaction
7
5'
expert
report table
status window


PM
AFL
queue
order change
5'''
element selection
queue update
5
9'
2/9
control signals
3
input data
1/8
command
fuzz.py
afl-fuzz
program
4
reading
6
traces
5''
reaction
7
5'
expert
report table
status window


PM
AFL
queue
order change
5'''
element selection
queue update
5
9'
2/9
control signals
3
input data
1/8
command
fuzz.py
afl-fuzz
program
4
6
reading
5''
traces
reaction
7
5'
expert
report table
status window


PM
AFL
queue
order change
5'''
element selection
queue update
5
9'
2/9
control signals
3
input data
1/8
command
fuzz.py
afl-fuzz
program
4
reaction
6
reading
traces
5''
7
5'
expert
report table
status window

EXECUTION TRACE DETECTION



Prev – previous base block label;
Cur – current base block label.

EXECUTION TRACE DETECTION

```
#endif /* ^WORD_SIZE_64 */

      }

      *virgin &= ~*current;

    }

    current++;
    virgin++;

  }

  s32 fd;
  fd = open(out_file, OWRONLY | O_CREAT | O_EXCL, 0600);

  FILE *f = fdopen(fd, "w");

  for (u32 i = 0; i < MAP_SIZE; i++){
        if (trace_bits[i] != 0) fprintf(f,"%06u(%u) ",i,trace_bits[i]);
  }

  fclose(f);
```

REPORT TABLE

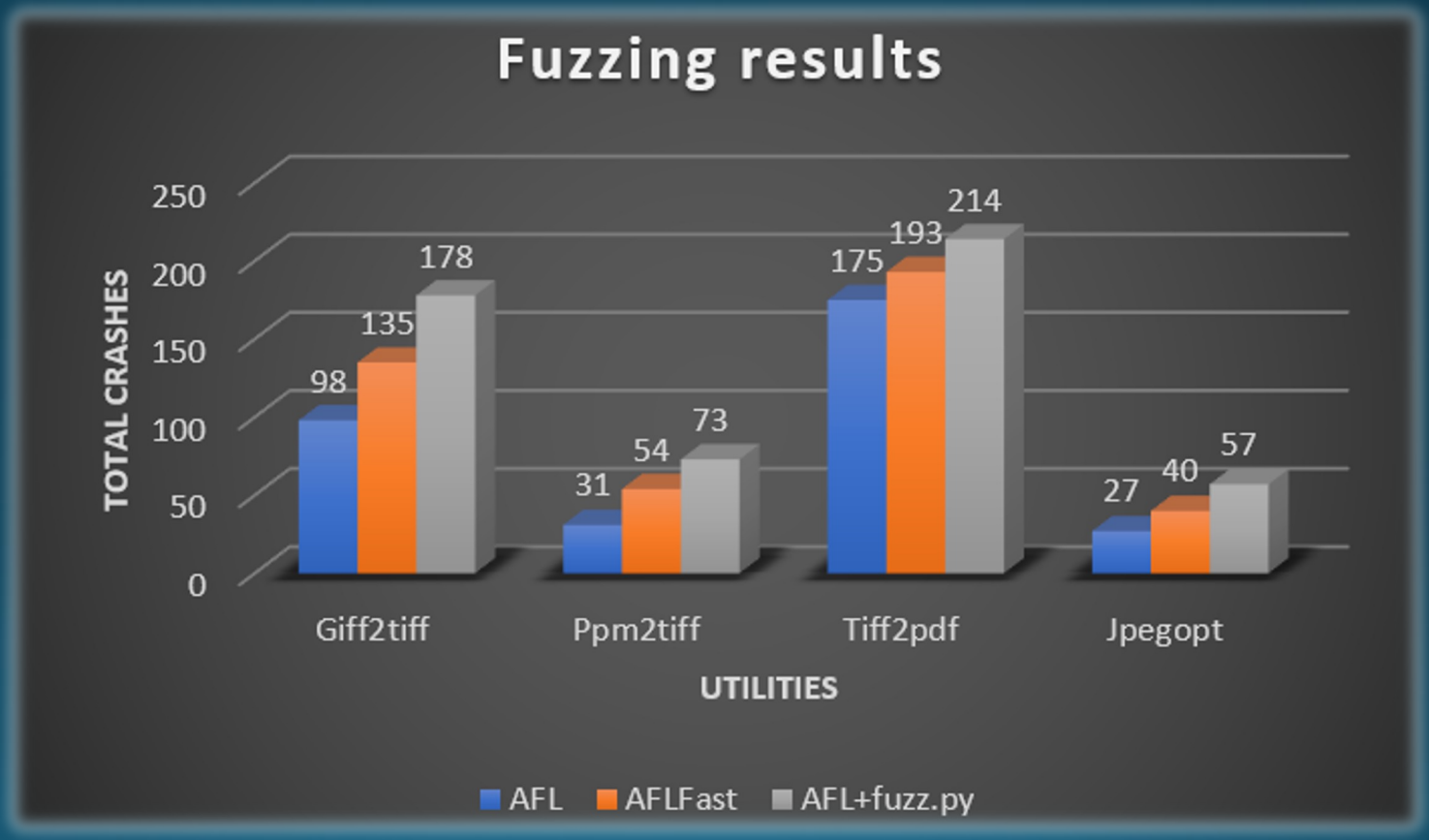
```
+-----------+-------+------+------------+----------+----------+--------+
|   object  |  qrun | qcov | part_execs | part_cov | now_exec |  src   |
+-----------+-------+------+------------+----------+----------+--------+
| id:000002 | 30266 |  1   |     56     |    0     |    +     |  orig  |
| id:000004 |   1   |  1   |     0      |    0     |    -     | 000000 |
| id:000005 |   1   |  1   |     0      |    0     |    -     | 000001 |
+-----------+-------+------+------------+----------+----------+--------+

0 --- Exit
1 --- Continue
2 --- Change flow in format: "2->id"

Enter command > 2->000004
<flow changed> | now_exec -> id:000004
```

FUZZING RESULTS

$T_{RT} = 30\ min$
$T_f = 12\ h$



FUTURE PLANS

- **To consider the scheme of using the proposed solution with static analyzers**
- To add visualization module
- Machine learning elements usage
- To consider the approach in schemes based on other fuzzing tools

THANK YOU FOR YOUR ATTENTION