Software Testing

Hands on…

Evgenii Vinarskii <vinarskii.evgenii@telecom-sudparis.eu>
Jorge López <jorge.lopez@telecom-sudparis.eu>

What is this Lab about ?

1. Consider two test generation strategies:
   - Test Generation based on the Code coverage
   - Model Based Test Generation methods
2. Compare the fault coverage of test suites generated based on the Code coverage and using Model Based Test Generation

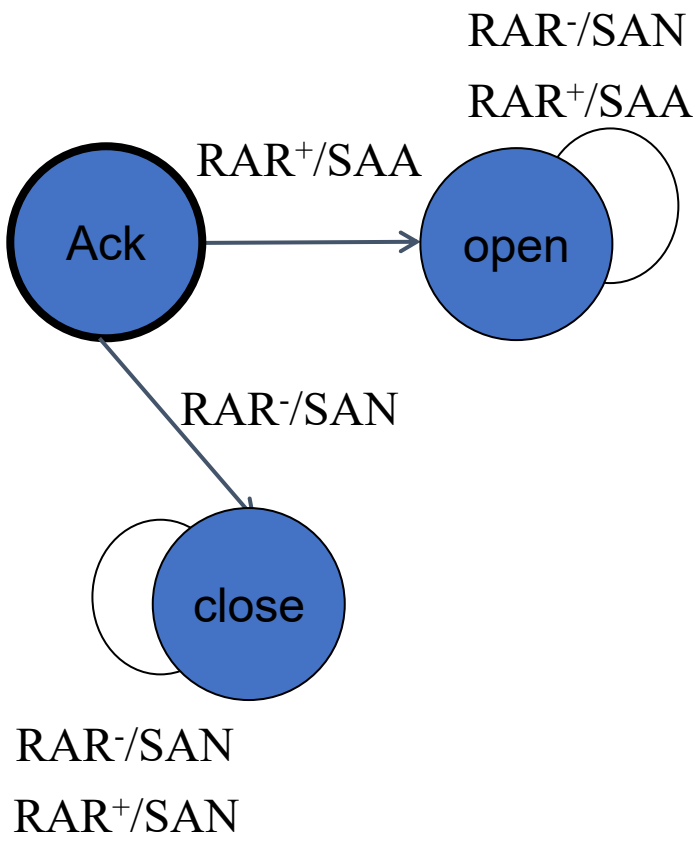
Test Generation based on the Code coverage

- Tools are available!!!
  - Sometimes you need to modify your code as they won't recognize all "versions" of the language…
  - E.g., http://pathcrawler-online.com:8080

- Test Generation depends only on the given code, independently from the code semantics

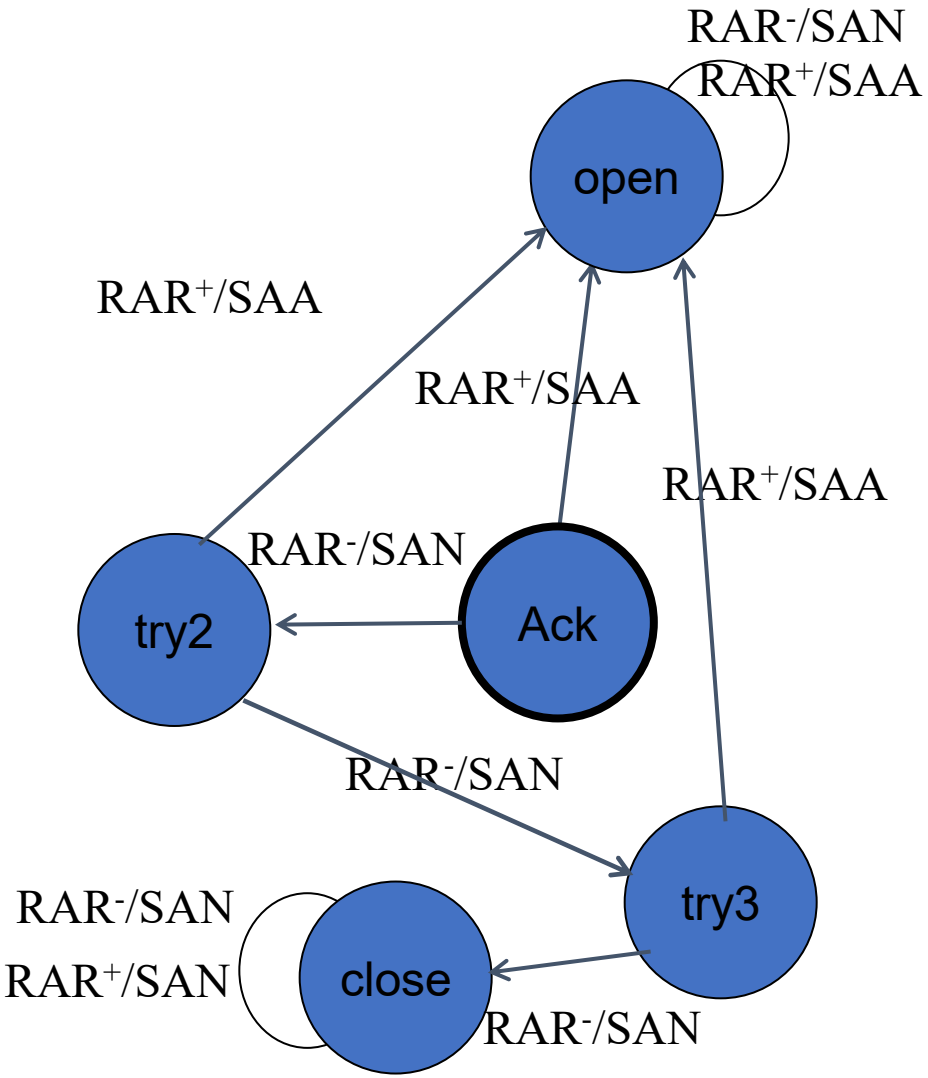
! It can be hard to test a code with non-primitive data types and macros

PathCrawler: PAP with one attempt to authenticate



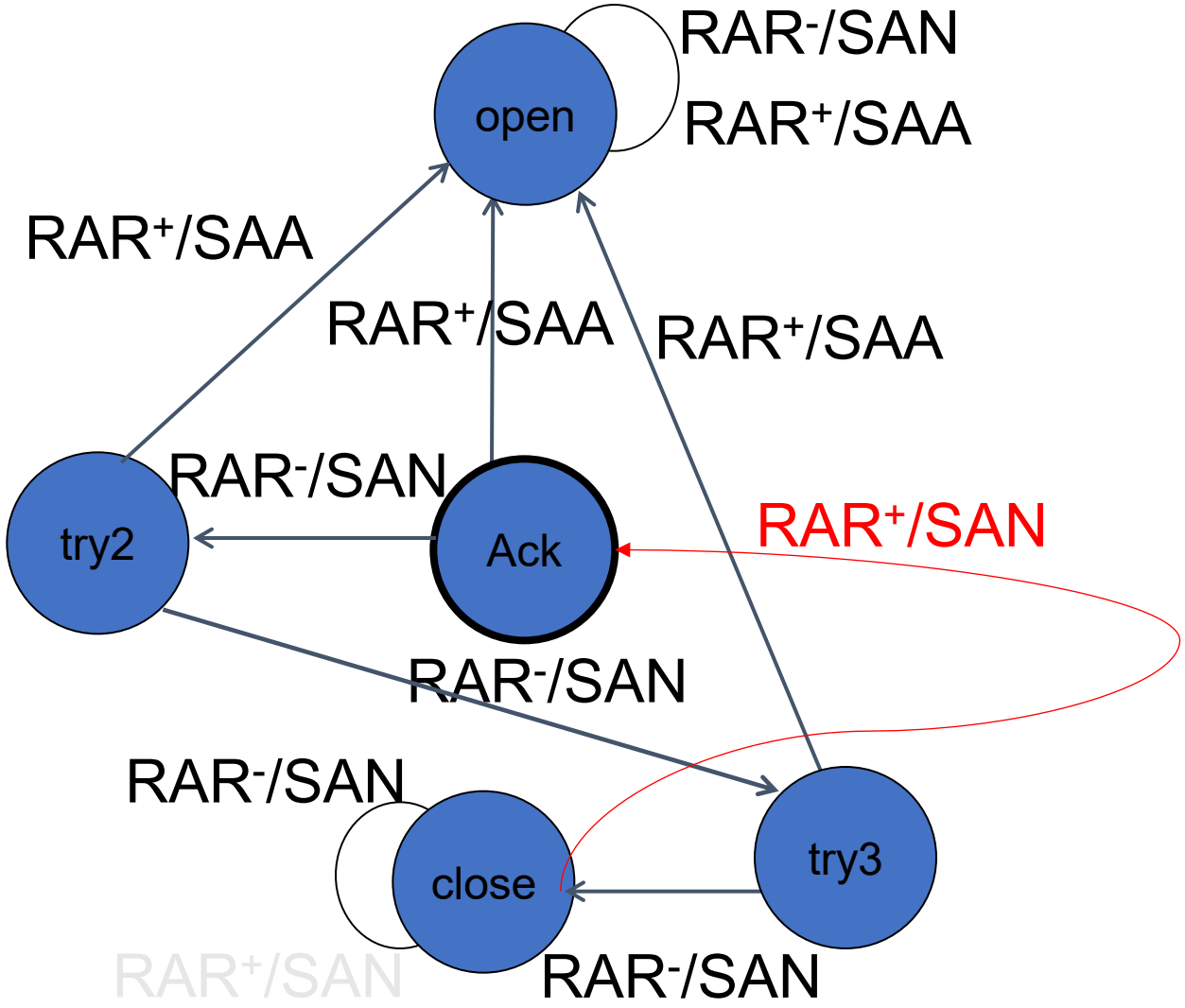

```c
int pap_fsm_next_state(int state, int input) {
    int next_state;

    if(state == ACK && input == RAR_PLUS)
    {
        printf("\ni=%s\to=SAA\n\n", (input==RAR_PLUS)?"RAR+":"RAR-"); //OUTPUT
        state = OPEN;
    }
    else if(state == ACK && input == RAR_MINUS)
    {
        printf("\ni=%s\to=SAN\n\n", (input==RAR_PLUS)?"RAR+":"RAR-"); //OUTPUT
        state = TRY2;
    }

    else if(state == CLOSE && input == RAR_PLUS)
    {
        printf("\ni=%s\to=SAN\n\n", (input==RAR_PLUS)?"RAR+":"RAR-"); //OUTPUT
        state = ACK;
    }
    else if(state == CLOSE && input == RAR_MINUS)
    {
        printf("\ni=%s\to=SAN\n\n", (input==RAR_PLUS)?"RAR+":"RAR-"); //OUTPUT
        state = CLOSE;
    }

    else if(state == OPEN && input == RAR_PLUS)
    {
        printf("\ni=%s\to=SAA\n\n", (input==RAR_PLUS)?"RAR+":"RAR-"); //OUTPUT
        state = OPEN;
    }
    else if(state == OPEN && input == RAR_MINUS)
    {
        printf("\ni=%s\to=SAN\n\n", (input==RAR_PLUS)?"RAR+":"RAR-"); //OUTPUT
        state = OPEN;
    }
    return state;
}
```

PAP with three attempts to authenticate



- A naive/simulation of an implementation can be found on: https://github.com/jorgelopezcoronado/PAPLab
- The Code can be compiled in a *-nix system with *make* utility and then executed with *./pap* (Mac OS X executable included)
- Valid user/password combinations may be found in the file users.db, a single user/password entry per line: user and password separated by a single space

Testing the PAP implementation



- *How can we test it using PathCrawler?*

- *The implementation provided has the fault!!!*
  - *What is the test suite derived using the code coverage strategy**?** Does it detect the fault**?***
  - *What is the test suite derived using the transition tour method **?** Does it detect the fault**?***
  - *What is the test suite derived using the W-method**?** Does it detect the fault**?***

Your tasks

1. Study the PathCrawler and derive a test suite based on the Code coverage for the function *pap_fsm_next_state* in file *pap.c*
2. Derive a test suite to detect for the FSM mutant of PAP from the previous slide
3. Apply the test suites derived at steps 1 and 2

Thank you for your attention
Questions ?