MASTER PRAKTIKUM SS18

PROF. DR. ALEXANDER PRETSCHNER
SAAHIL OGNAWALA

HACKING FUZZERS FOR VULNERABILITY SCANNING

WHO WE ARE

Prof. Dr. Alexander Pretschner

Chair IV – Software and Systems Engineering

Saahil Ognawala

Office: MI - 01.11.041

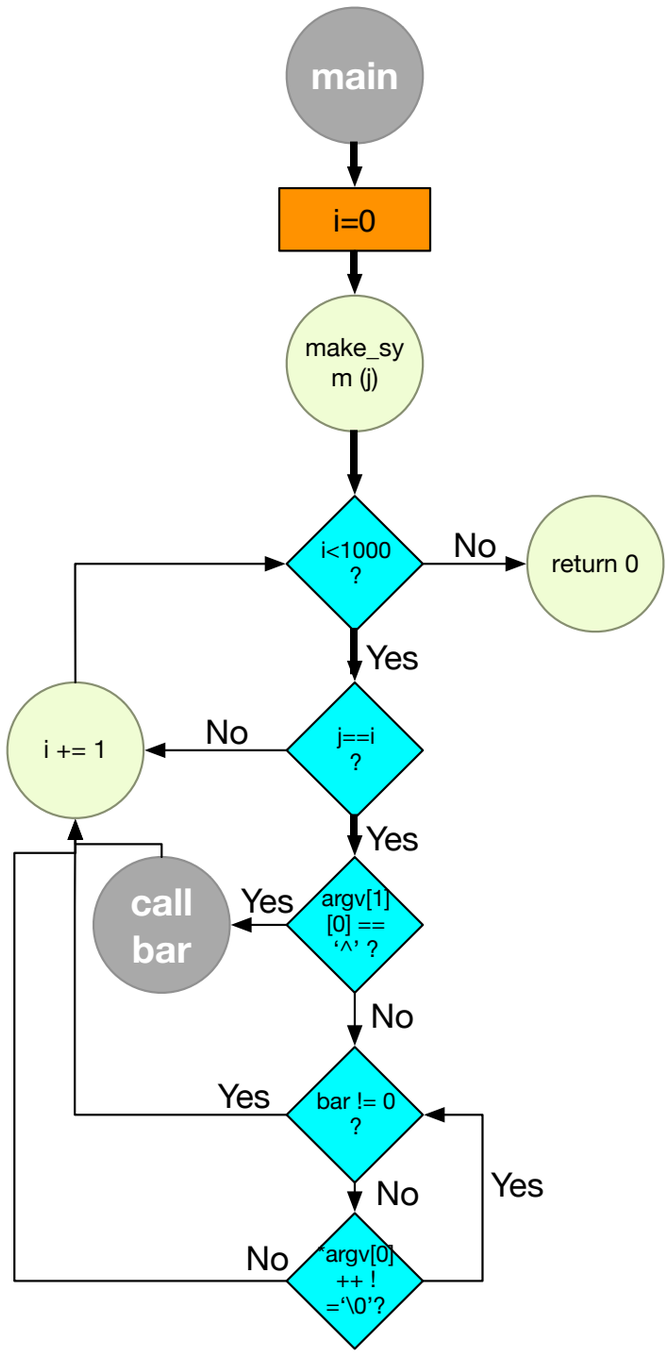
Email: ognawala@in.tum.de

Our website: https://www4.in.tum.de/

LEARNING GOALS

- State-of-the-art in automated vulnerability scanning
- Key challenges of whitebox and blackbox fuzzing
- Domain-specific solutions (*hacks!*)
- The *coverage* question
- Dealing with compositionality
- Severity assessment

OVERVIEW OF WHITEBOX FUZZING



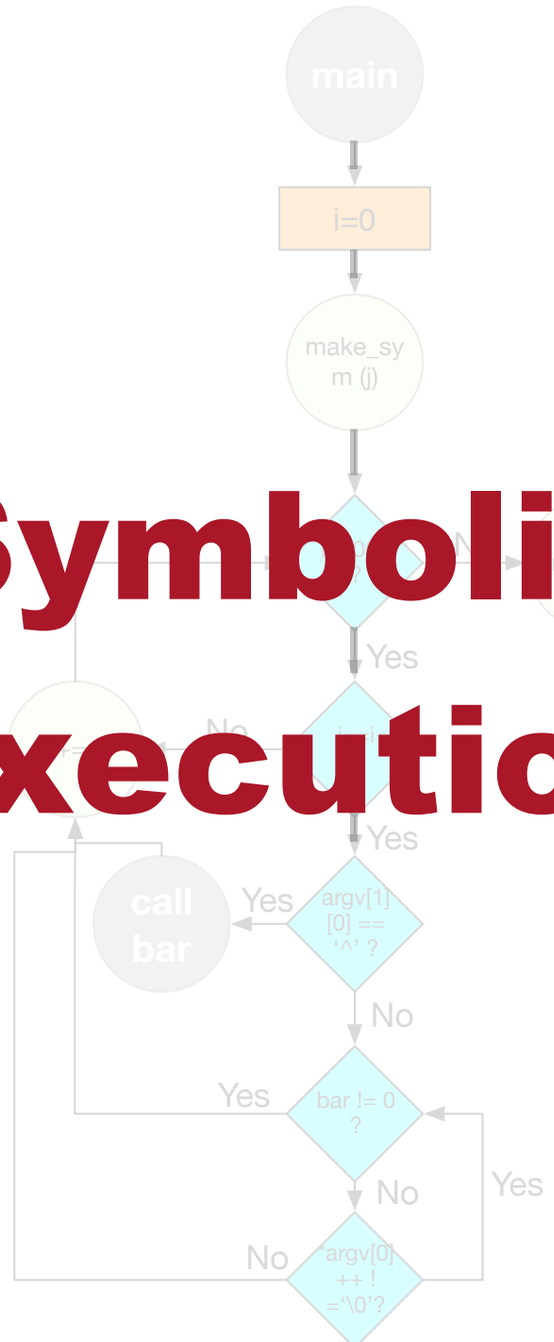
(i<1000) && (j!=i)

Solution!
i = 0; j = 100

At at the same time collect
(i<1000) && (j==i)

Solution!
i = 0; j = 0

OVERVIEW OF WHITEBOX FUZZING



Symbolic execution

(i<1000) && (j!=i)

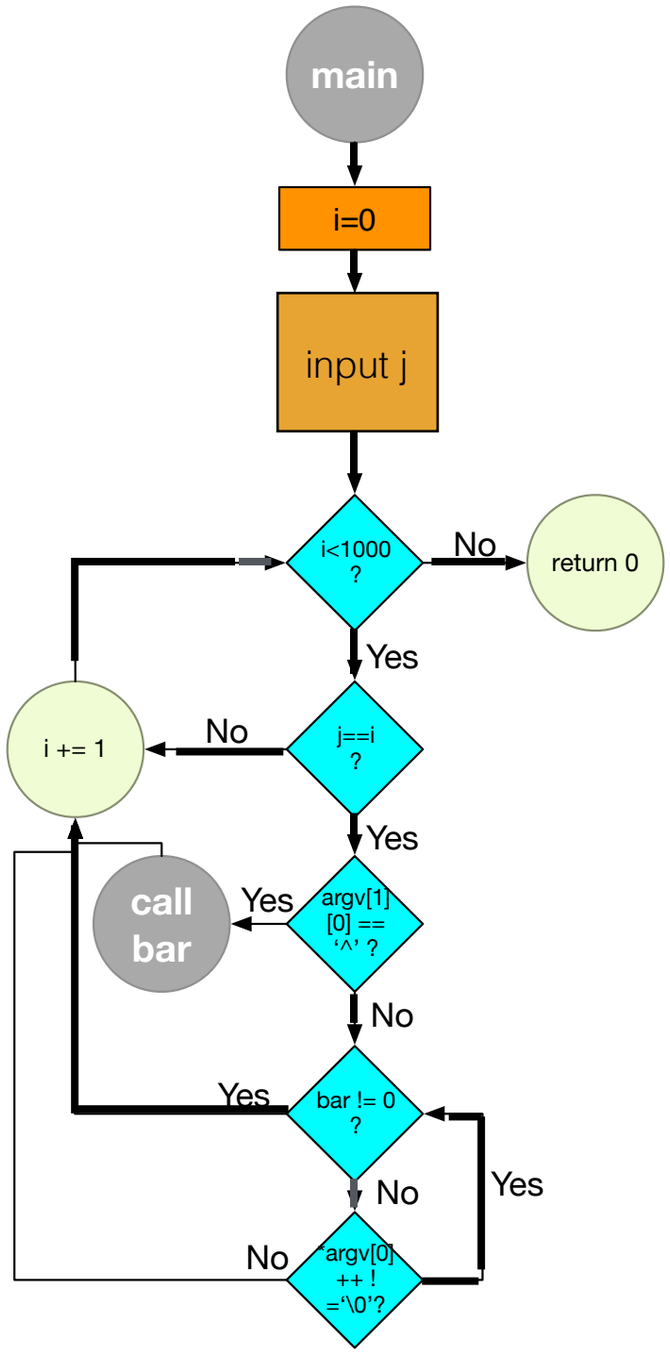
Solution!
i = 0; j = 100

At at the same time collect

(i<1000) && (j==i)

Solution!
i = 0; j = 0

OVERVIEW OF BLACKBOX FUZZING

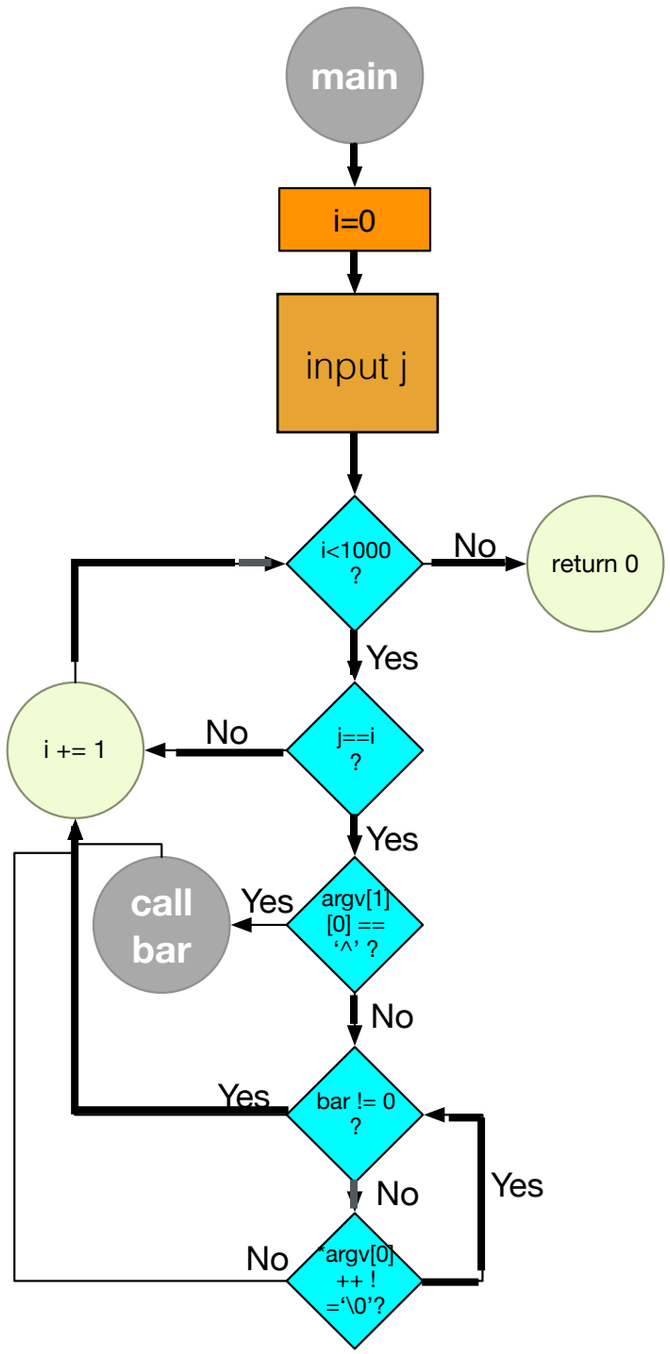


Seed input

"el", "Hello", j=12

Mutate this!

OVERVIEW OF BLACKBOX FUZZING



Seed input
"^o", "Hello", j=12

And so on…

OVERVIEW OF BLACKBOX FUZZING

- Variant of random testing
  - Input mutation, instead of random sampling.
    - Basic fuzzers mutate inputs randomly.
- Automation is the key!
  - *"~~Move~~ Mutate fast, break things"*
  - Dependant only on (input,output)

(SOME) IDEAS FOR GREYBOX FUZZING

Idea 1

1. First fuzzing
    a. Collect branch coverage statistics
2. Symbolic execution of the program
    a. Search strategy - Prefer branch closest to *uncovered branch*

Advantage

- "Easy" (commonly hit) branches covered with fuzzing fast
- Less reliance on constraint solver bottleneck

(SOME) IDEAS FOR GREYBOX FUZZING

Idea 2

1. Start concolic execution
2. Threshold path constraint size
3. Solve constraint at cut-off point
4. Generate counter-example
5. Use counter-example as seed input for fuzzer

Advantage

- Guided method for generating “good” seed inputs
- No need to solve large constraints

ANALYSES TARGETS

Programs to be analyzed -

- C language *(maybe C++)*
- Grep, Flex, Bzip2, OpenSSL, etc.

Vulnerability database example -

- National Vulnerability Database (NVD) - https://nvd.nist.gov/

TASKS OVERVIEW

- Team formation (teams of 2)
- Usage of existing tech
- Engineering your own (tailor-made) solution
- Reporting vulnerabilities
  - Existing vulnerabilities
  - Zero-day vulnerabilities
- Generating exploits
- Justifying solutions
  - Program types
  - Specifics of the solution
  - Does it generalize?

ROUGH SCHEDULE

Phase 1

- Introduction to techniques and tools
  - Symbolic execution + KLEE
  - Fuzzing + AFL
- Comparative analysis of existing techniques
- Phase 1 reporting
  - Results of comparative analysis
  - Lessons learnt
  - Planned engineering solution

Phase 2

- Implementation of planned solution
- Preliminary evaluation of solution
- Comparison with phase 1 results
- Phase 2 reporting
  - Results of preliminary analysis
  - Lessons learnt
  - Improvements planned

Phase 3

- Second implementation phase
- Comparison with phase 2 and phase 1 results
- Final reporting

ROUGH SCHEDULE

Phase 1

- Introduction to techniques and tools
  - Symbolic execution + KLEE
  - Fuzzing + AFL
- Comparative analysis of existing techniques
- Phase 1 reporting
  - Results of comparative analysis
  - Lessons learnt
  - Planned engineering solution

Phase 2

- Implementation of planned solution
- Preliminary evaluation of solution
- Comparison with phase 1 results
- Phase 2 reporting
  - Results of preliminary analysis
  - Lessons learnt
  - Improvements planned

Phase 3

- Second implementation phase
- Comparison with phase 2 and phase 1 results
- Final reporting

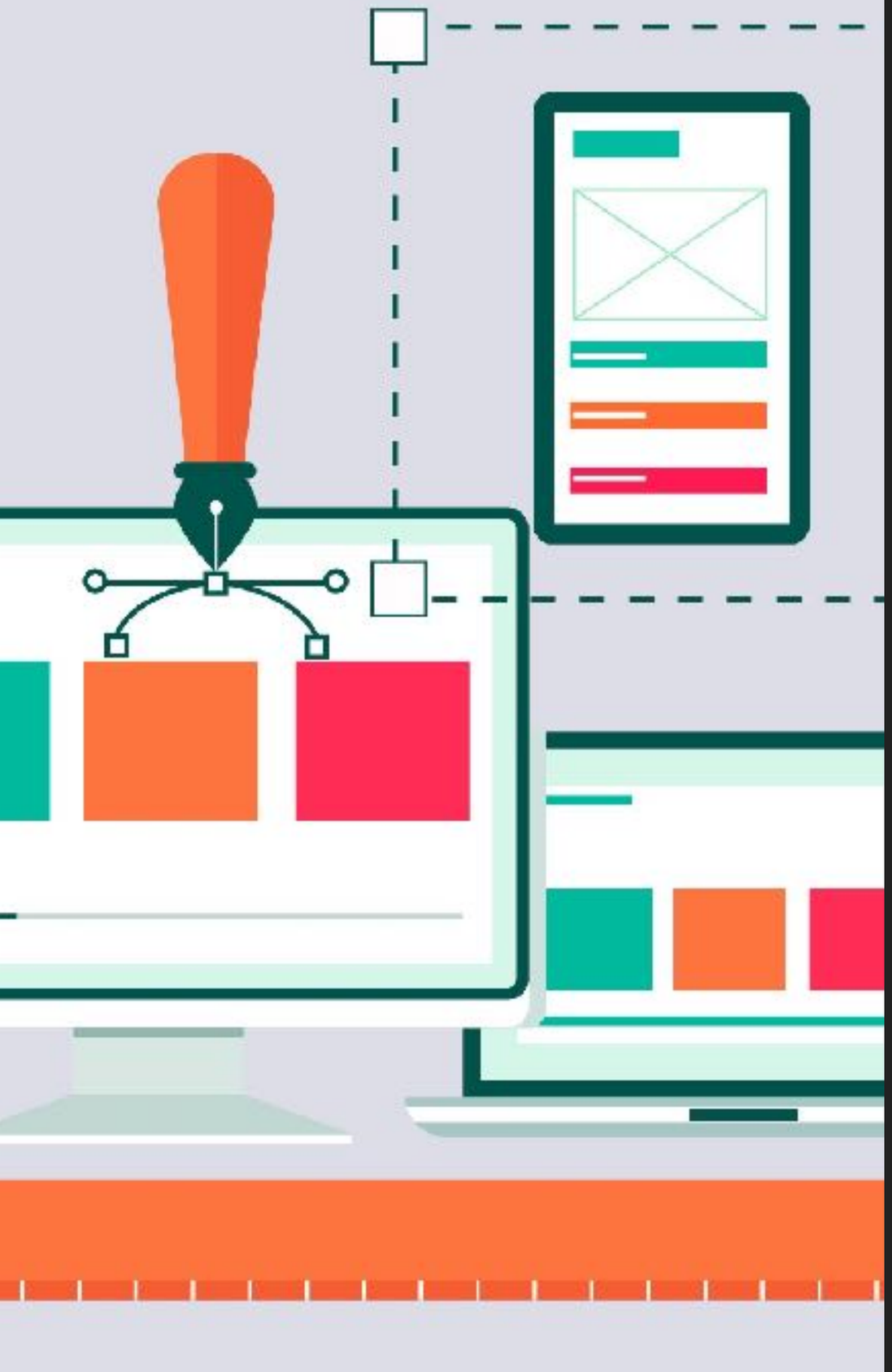
- **Everything in teams!**
- **List of recommended tasks to be published on Moodle**
- **Upcoming lectures/q&a-sessions to be announced on Moodle**

FINDING LITERATURE

- TUM Library
  - Informatik
  - Others…
- Online portals
  - Springer (www.springerlink.com/)
  - ACM (dl.acm.org/)
  - IEEE (ieeexplore.ieee.org/Xplore/guesthome.jsp)
  - Google Scholar (scholar.google.com)
  - Scopus (scopus.com)

REGISTRATION

- Matching system: http://docmatching.in.tum.de/
- What are your chief skills? (programming and others)
  - Mail ognawala@in.tum.de latest by Thursday, 15th February, 2018
    - Include - Full name, IMAT number, TUM email ID
- Receive a confirmation email after matching is complete.
  - Includes schedule



ognawala@in.tum.de

THANK YOU