Lab Security in Distributed Systems

MA-INF 3320

Application

In order to get a topic, you will need to submit your expose until 07.10.2020. Please send it to the respective supervisors. You will be informed whether you were chosen/not chosen for your favourite lab topic until 14.10.2020. Multiple applications are allowed for up to three topics.

Deadline for expose: 07.10.2020

Deadline for registration in BASIS: 15.11.2020

Fuzzing

Supervisor

Klaus Tulbure
tulburek@iai.uni-bonn.de
Mischa Meier
meierm@cs.uni-bonn.de

Fuzzing

Fuzzing is a state-of-the-art analysis for finding bugs and vulnerabilities in software. It can be described as an "intelligent" brute-force analysis that penetrates a program with input to, hopefully, cause a crash. In their cores, most fuzzers use an evolutionary algorithm and combine dozens of different approaches for input generation, selection, mutation, fitness function, etc. They also leverage a broad variety of techniques for the implementation, such as static analyzes, machine learning, hardware acceleration, etc.

Your task is to develop a new technique for a fuzzer. You are free to chose, whether you want to improve an existing technique, port it from one fuzzer to another, or even come up with your approach. You can suggest your own idea or be provided with one by us.

Requirements are advanced programming skills in C/C++ or Java.

Literature https://www.cs.umd.edu/~mwh/papers/fuzzeval.pdf

Anti- and Anti-Anti-Fuzzing Techniques

Supervisor

Klaus Tulbure
tulburek@iai.uni-bonn.de

Anti- and Anti-Anti-Fuzzing Techniques

Some developers might not want their software to be fuzzable, e.g. to protect it against adversaries. Hence, they implement **anti-fuzzing techniques** that slow down the fuzzing process. However, that's just security through obscurity (i.e. not good) and on the long term an attacker will overcome the obstacle. Hence, one wants to break these anti-fuzzing techniques and find potentially hidden bugs before a malevolent attacker does.

Your task will be to develop such an anti-fuzzing technique,
or to take an existing technique and improve a fuzzer to overcome it.

Requirements are advanced programming skills in C/C++ or Java.

Literature https://www.usenix.org/system/files/sec19fall_jung_prepub.pdf

Own ideas?

Supervisors

all of us

send your ideas to
Klaus Tulbure
tulburek@iai.uni-bonn.de

Own ideas?

You've got some **own ideas** for a project in Usable Security and Privacy?

GREAT!

We'd love to work with you on that!