# **SPLab** Research for Safe and Effective Software

**S**ystem **P**rogramming **Lab**oratory
Yerevan State University

Sevak Sargsyan
sevaksargsyn@ispras.ru

# SPLab Team

Founded by **Victor Ivannikov** in 2009.

International team of bright researchers:

Members from Armenia

1. Yerevan State University
2. National Polytechnic University
3. Russian-Armenian University

Members from Russia, ISP RAS

1. Moscow State University
2. Moscow Institute of Physics and Technology

## SPLab Team

Summer courses for new members selection (2, 3, 4 grade):

1. Compilers: Design and Implementation
2. Software Security
3. Advanced C++ and Algorithms

**Victor Ivannikov** nominal scholarship for 10 months.
Graduate work.

# SPLab Projects

1. Compiler optimizations
2. Code Obfuscation
3. Source code clone detection
4. Code static analysis
5. Code dynamic analysis

## Compiler Optimizations

1. GCC – Optimal code generation for ARM architecture (patches accepted by community)
2. LLVM – Vectorization, instruction scheduling (Intel, ARM)
3. V8 – Register allocation
4. V8 – «Hot» functions profiling
5. V8 – Register rematerialization
6. V8 – LLVM as backend
7. Webkit – Register allocation
8. LLVM as backend for PostgreSQL (github *https://github.com/ispras/postgres*)

## Code Obfuscation

Code obfuscation is used for:

1. Security Improvement
2. Protection from reverse engineering

LLVM based source code obfuscator (data and control flaw obfuscation):

1. Functions merge
2. Local variables reordering on stack
3. Redundant calculation
4. Branching
5. Extra functions call
6. …..

# Source Code Clones Detection

1. Code clones detection based on program dependence graph (supported for: C/C++/Objective-C, JavaScript, Ruby, Python, Haskell, Java, PHP, Pure, Lua, LLVM bitcode)
2. Scalable (million lines of source code: Android, Linux kernel)
3. Accurate (> 90%)
4. Extendable for new language (based on LLVM bitcode or PDG)
5. Cross-Language (can detect rewritten code fragments from one language to another)
6. Copy-paste error detection

## Code Static Analysis

1. Binary code clone detection (viruses detection, etc.)
2. Old/buggy software components/library detection
3. Buffer overflows detection
4. Format string detection (C/C++ printf)
5. Use after free detection (C/C++, new/delete)

## Code Dynamic Analysis

1. BNF grammar fuzzing (compiler, interpreter)
2. Directed fuzzing
3. Network fuzzing
4. STDIN, ARGV, ENVIRONMENT fuzzing

# Our Research Results

All instruments are comparable or exceed best analogs:

1. More than 30 publications and conferences
2. Three PhD candidate works
3. More than 20 graduate works

# Thank You!