Impact Oriented Programming Prototype

Students: Meesha Chauhan, Gerard O'Rourke, Samhita Vempatti, Palash Oswal, Yu-Hsin Wang
Faculty advisor: Gregory Kesden
Sponsor: Adam Cecchetti, Staris Labs

Project Overview

Background

Debugging helps identify and address vulnerabilities in code. Programmers inefficiently debug their own code by using print statements and debuggers. The lost time can be significantly reduced if the programmers can see the impact of their code in real time. Our team worked with Staris Labs to deliver a proof of concept to show the impact with various techniques, such as fuzzing and static analysis. We were able to verify the presence of known vulnerabilities in code.

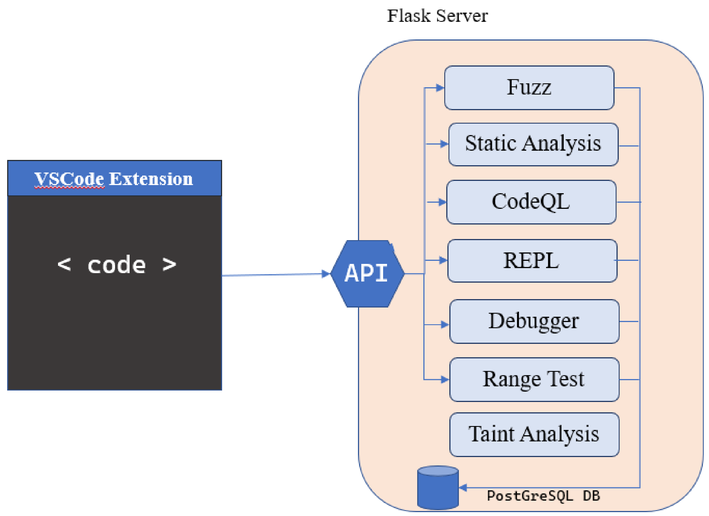
Project Goal

Develop a prototype that showcases how a programmer can utilize the tools such as fuzzing and static analysis to see the impacts.

Tools Used

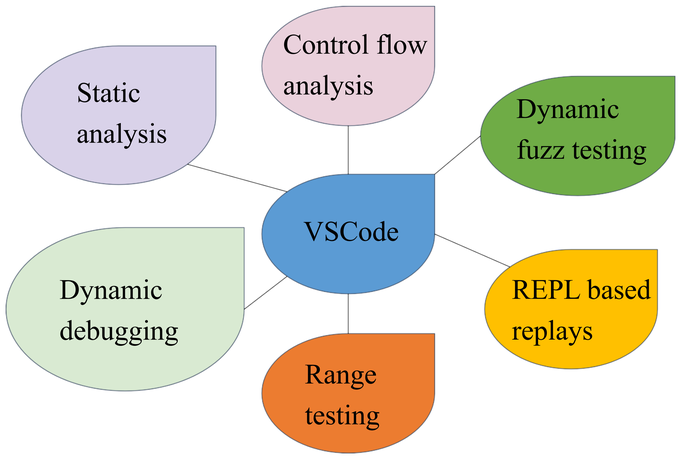
AFL++, CodeChecker, ,CodeQL, Docker, Flask, GraphViz, PostgreSQL, Rust, TreeSitter, VSCode. GDB, Clang/LLVM

Architecture



Capabilities

Our program can do the following for any C function within VSCode:



Solution

We developed a VSCode extension that the user runs while developing C code. They select any C function (MUSL libc) and send it to a remote backend. The backend, a Flask server, is where the function is fuzzed and statically analyzed using various tools. The abstracted results are available to the user. The results help the user automate testing and debugging and better understand the impact of the code they write on the machine.

Scan QR code to view demo

Implementation

- **VSCode Extension**
  Customizable extension that supports C language semantics and web based result explorer.
- **HTTP API Backend with Tools**
  Portable Python based backend that abstracts the complex tooling infrastructure.
- **C++ Harness Generators**
  Executable Binaries produced for REPL and Fuzzing purposes. These have support for complex types.
- **Custom CodeQL Analyses**
  A full suite of static analysis of the code using the power of custom CodeQL queries.
- **Automated Line by Line Debugging with Concolic Execution**
  Stepping through the function code line by line and demonstrating state of variables for a given parameter value.

Results & Future Work

Identification Of Known Vulnerabilities

We were able to verify the presence of known vulnerabilities in code, notably CVE-2020-28928 was observable in the Static Analysis view.

User Feedback

We have obtained feedback from several INI students with programming backgrounds and a professor for our product. User responses have been very positive.

Future Work

Our project has shown that Impact Oriented Programming (IoP) is possible from a technical perspective. The next stage is to see if there is a market for IoP as a product. Currently our prototype only supports C and has been tested with MUSL libc. Expanding Impact Oriented Programming to work with other languages is a potential area of future work.

Carnegie Mellon University
Information Networking Institute