SLAIT

Secure Language Assembly Inspection Tool

Dr. Marius Silaghi
Michael Bratcher
Maria Linkins-Nielsen

Goal and Motivation

"Develop a secure, streamlined way to inspect snippets of code at a low level of abstraction"

- Anyone programming or observing their high level program in low level languages (MASM, Bytecode) put their devices at risk
- Devices require you lower its security levels to run these programs

- We wish to provide an environment that:
  - Eliminates risk to host machines
  - Allows users to inspect hardware changes made by their programs
  - Allows users to compare previous iterations and other programs' effects on the hardware

Approach (Key Features)

Secure Inspection

- Submit x86 or Java bytecode snippets
- Analyze safely without exposing your system

Targeted Tracking

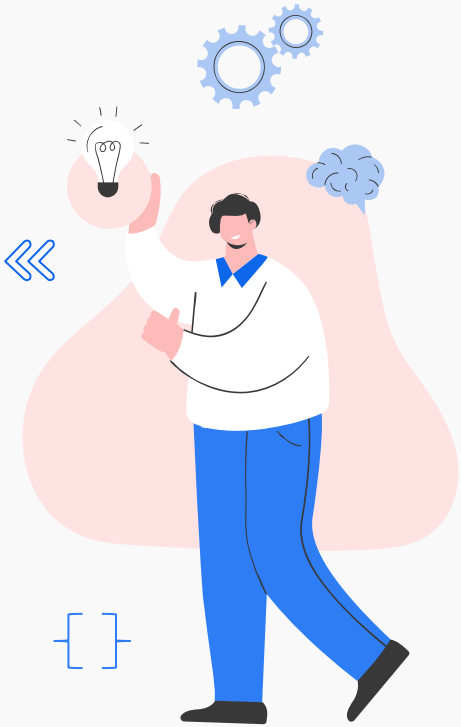
- Mark key points in code to observe registers/flags
- Capture states before/during/after execution

Isolated Execution

- Run your code in a Docker-based sandbox
- Prevent damage to your personal machine

Clear Visualization

- See an organized timeline of state changes
- Compare runs with varied inputs



Novel Features & Functionalities

- Secure, on-demand, & low-level code inspection
- Docker sandbox keeps execution isolated
- Track registers & flags at key points
- Compare runs with clear visualizations

Tools

Docker

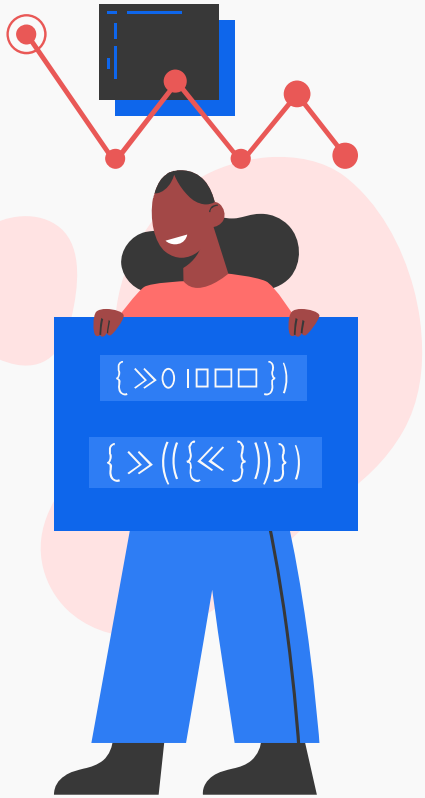
- ★ Containerized Secure Environment
- ★ Consistent between separate instances

Angular

- ★ Widget based UI for front-end
- ★ Typescript based library

Parsing Libraries

- ★ Identifying where user actions are placed post compilation



Challenges

Docker and Angular

- Specifically Requested
- Little Experience with Docker (OS 161)
  - No experience with API or CLI interactions
- No Experience with Angular
  - No Experience with TypeScript

Potential Solution:
Research and Develop and Procedure Documentation
OR
Find a justifiable alternative

Challenges

Backend Setup and Communication

- No Tangible Experience with Backend Server OS
- Ubuntu Server
- Interactions with Docker within the environment

Potential Solution:
Research and Develop and Procedure Documentation

Challenges

Parsing Actions Between High and Low Level

- Mandatory for basic functions
- How do we pass off information of actions without affecting any potential runtime? (or as minimally as possible)
- Bad solutions are easy to implement but will affect the runtime of the program

Potential Solution:
Parsing Library Investigation and choosing a good language for the backend

Milestone 1 Goals

Tool Selection

- Choose frontend tools (Angular, visualization libraries)
- Choose backend tools (Docker setup, server framework)
- Select code parsing/compilation support - x86 & Java bytecode

Initial Demos

- Initial testing with x86 code in Docker
- Display sample register/flag data in Angular
- Test basic frontend to backend communication

cont.

Investigation of Technical Challenges

- Configuring docker and sandbox permissions
- Ensuring flagged code sections translate correctly after compilation
- Establishing angular build/test pipeline
- Configuring backend compilation for one language

Documentation

- Create Requirements Document
- Create Design Document
- Create Test Plan

Milestone 2 Goals

Implement, test, and demo

- User input of code snippets
- Run of x86 & Java bytecode in Docker with flag/register tracking
- Angular visualization of registers/flags
- Error handling

- Implement basic Java bytecode support
  - Test frontend parsing between different languages
- Implement Docker security (restricting system calls)
- Update Requirement and Design Documents

Milestone 3 Goals

Implement, test, and demo:

- Marking registers/flags at code locations and comparing state changes
- Timeline view of register/flag states
- Comparative run mode (different initialization states)
- Enhanced error reporting and flag tracking

- Polish Angular frontend visualization widgets
- Implement frontend comparison of multiple executions using visual graphs
- Continue frontend–backend integration (end-to-end flow)
- Prepare semester 1 progress presentation and documentation

Milestone 1 Task Matrix

Task	Michael	Maria
Compare and select tools	Frontend tools	Backend(Docker, server)
“Hello World” demos	Angular component with sample data	Testing minimal code within Docker
Resolve technical issues	Setup angular build/test	Configure docker sandbox & compile x86 test
Frontend-Backend Demo	Connect angular API	Establish backend API
Requirement Document	Write 50%	Write 50%
Design Document	Write 50%	Write 50%
Test Plan	Write 50%	Write 50%

The end.

Questions?