

Project Title: SLAIT – Secure Language Assembly Inspector Tool

Team Members:

- Maria Linkins-Nielsen – mlinkinsniel2022@my.fit.edu
- Michael Bratcher – mbratcher2021@my.fit.edu

Faculty Advisor: Dr. Marius Silaghi – msilaghi@fit.edu

Client: Dr. Marius Silaghi – CSE Professor & Faculty Advisor

Progress of Current Milestone:

Task	Completion%	Michael	Maria	Reasoning/To-Do
1. Establishing backend access	75%	50%	50%	<p>Frontend - backend access at a development level has been established, but the actual front-to-back data flow has not been established. File transfer over HTTP has been set up within the frontend, but the backend server does not communicate with the frontend at the moment.</p> <p>Backend - actively setting up backend environment, including automated dockerized execution and register parsing logic for inspection results.</p>
2. Frontend Visualization of returned information	25%	100%	0%	<p>Visualization libraries have been chosen, but development efforts mostly focused on further specified GUI requirements and the HTTP implementation</p>

3. Finalize container environment	100%	0%	100%	Docker environment finalized with NASM, GCC, GDB, and supporting tools installed. User permissions, testing tools, and safety configurations implemented.
4. Enhance Frontend GUI	100%	100%	0%	After meetings with Dr. Silaghi, further frontend additions along with NASM oriented changes were made.
5. Confirm container compatibility with API and other tools	100%	0%	100%	Container tested and confirmed compatible with NASM compilation, GDB register tracing, and script automation. Backend is now ready to integrate with HTTP services and frontend API routes.

Discussion of Milestone Tasks:

1.

“Establishing backend access” had multiple independent goals. Primarily, the goal was to ensure we had access to the development environment and could set up the environment for the code execution layer of the project. Once established, the frontend began development on the HTTP service that would send the .asm file and the instruction file to the backend. Maria worked on developing and testing the backend container to execute NASM code safely and reproducibly in a Linux environment. This included:

- Creating a Dockerfile with the required packages and user permissions.
- Implementing scripts to assemble, link, and execute .asm files.
- Adding structured stdout and file logging to capture program outputs.
- Developing a parse_registers.py utility that uses GDB to set breakpoints and capture register states.

2.

Frontend Visualization development was mostly stalled as we still have not defined the format for how information will be returned to the frontend before its to be displayed. Delaying this did give more time to focus on working towards connectivity and updating the initial GUI, while also seeming like a bit of an out of place task after the fact. Regardless of the actual developmental progress, the library Chart.js was selected after a bit of research for its versatile library of visualization tools that work seamlessly with Angular projects.

3.

The container is now fully operational, equipped with NASM, GCC, Python, and GDB. Testing confirmed successful compilation and execution of NASM assembly files within the container. It includes user sandboxing, inline debugging, and output redirection, ensuring safe execution and proper file handling.

4.

“Enhance Frontend GUI” included a very wide range of changes and modifications to the frontend, and is where most of my (Michael) time was spent. The largest change users will notice is the addition of a file creator, which allows users to create and edit a new file within the webpage itself, modify the name of the file, and then finalize it for inspections. A significant amount of the time spent on this task was debugging the logic flow when it comes to the specific states buttons and information displayed should be in at specific times. This included the file name display within the inspection manager, disabling the “add inspection button” while a file is being edited, wiping all current inspections when a new file is being edited or a file is swapped, and changing the “test” button to show “uploading” while the application waits for a response from the server. Finally, MASM to NASM changes were made, which includes an update to the template that was originally in the text editor (is now a NASM hello world) and flags were also updated accordingly.

5.

The Docker container was confirmed to correctly handle inputs, generate logs, and produce register dump outputs in formats that will be parsed and returned via API.

Contribution to Milestone Tasks:

Michael:

My contributions included confirming backend server access, all enhancements to the Frontend GUI, Frontend visualization library selection, and HTTP service development.

Maria:

My contributions focused on backend development and environment setup.

I created the Dockerized NASM execution environment, including configuration for compilation, linking, debugging, and register extraction using GDB. I also developed scripts (run_asm.sh, parse_registers.py) to automate assembly execution, log stdout output, and extract register states based on user-specified breakpoints.

1. Plan for the next Milestone (task matrix) or [skip if this is for Milestone 6]

Task	Michael	Maria
Initial HTTP Test	50%	50%
Update File Manager Widget Visuals	100%	0%
Create Widget Component for Visualization	100%	0%
Finalize backend file upload and execution API	0%	100%

Implement Multi-Line Register Tracking in Backend	0%	100%
---	----	------

2. Discussion of each planned task for the next Milestone

- Task 1: The Initial HTTP Test involves an initial test to verify the method of file transfer to the backend, which involves developing the service to receive the information via HTTP, then performing testing to verify the sent information.
- Task 2: This involves modernizing the widget design away from plain basic white boxes with basic button functionality into more consistent, modular design with clear user feedback on actions.
- Task 3: This involves the creation of the widget that will hold the return of the backend execution information once it has been developed. This is meant to be proactive and will begin the final step for complete main functionality of the frontend.
- Task 4: Finalize Backend API for receiving files, executing NASM code, and returning parsed results in JSON format to the frontend.
- Task 5: Expand parse_registers.py to support multiple breakpoints and line-based register inspection, matching frontend user inputs.

3. Date(s) of meeting(s) with Client during the current milestone: n/a

4. Client feedback on the current milestone

- see Faculty Advisor Feedback below


5. Date(s) of meeting(s) with Faculty Advisor during the current milestone:

- Friday, October 10th
- Friday, October 17th

6. Faculty Advisor feedback on each task for the current Milestone

- Task 1:
- Task 2: Good job so far for the frontend.
- Task 3: Potentially a drop-down is needed to select the language/OS and a flag to tell if it is a snippet
- Task 4:
- Task 5:

7. Faculty Advisor Signature: _____ Date: _____



8. Evaluation by Faculty Advisor

- Faculty Advisor: detach and return this page to Dr. Chan (HC 209) or email the scores to pkc@cs.fit.edu
- Score (0-10) for each member: circle a score (or circle two adjacent scores for .25 or write down a real number between 0 and 10)

Michael	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Maria	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10

○ Faculty Advisor Signature: _____ Date: _____