ECE 49595 OSS Final Proposal

Team: Yasmin Shahed, Brandon Coleman, Rohan Sagar

Repository URL:

Cutter: https://github.com/Rohan-Sagar/ECE49595-OSS
Rizin: https://github.com/yasmiins/rizin

Introduction

For our ECE49595O project, we will be contributing to Cutter, an open source reverse engineering platform powered by Rizin (core engine), a reverse engineering framework. Cutter is natively integrated with Ghidra's decompiler, ready out-of-the-box. Additionally, it includes powerful features such as a graph view, dynamic analysis debugger, disassembly view, hex editor, emulation, plugins, and more.

There are various different use cases for Cutter, such as Malware Analysis, Binary Exploitation, Firmware Analysis, and Software Auditing. Through these use cases, Cutter is able to be a vital tool in contributing to the improvement of Software security and anti-malware development.

Being an open source software, there are plenty of issues to contribute to. For this project, our main goals consist of developing the solution for these three issues within Cutter.

1. Allow adding new flags from Hexdump
2. Multi-Line operations / Multi-line Selection
3. Semantically-aware syntax highlighting

Background

1. Allow adding new flags from Hexdump
   Issue Link: https://github.com/rizinorg/cutter/issues/2932

   This issue asks for the ability to add flags from the hexdump widget. In base cutter you can add flags from the disassembly widget from the right click menu. However, currently that isn't possible from the hexdump widget. This will allow users more customizability for their output files in the Hexdump widget and make for a more seamless experience between the two widgets

2. Multi-line operations / Multi-line Selection
   Issue Link: https://github.com/rizinorg/cutter/issues/2601

   This issue asks for a couple of new features for the user interface:
   “Allow selection of multiple lines by drawing a rectangle with the mouse (click-hold-move-release left mouse button). Allow adding additional individual lines by shortcut like Shift+Left Click.” This will allow users the flag or change multiple lines at a time

3. Semantically-aware syntax highlighting
   Issue Link: https://github.com/rizinorg/cutter/issues/3098

   Currently, the Cutter program is unable to highlight the syntax of the output file depending on semantics that the user can change. There are lines that should be highlighted that will no longer be highlighted based on a small, semantic change. This will allow Cutter to have a deeper semantic analysis of output files. This will allow for a more personalized experience with cutter, and ensure consistency between semantic changes in the files or programs, allowing for an improved user experience and a more correct output when it comes to syntax.

Requirements

1. Allow adding new flags from hexdump
    a. Adding flag via hexdump widget right click menu
        i. Users should be able to to add a flag through the hexdump widget right click menu
    b. Adding flag via keyboard shortcut
        i. As more of a subtask to the overall requirement, there should also be a keyboard shortcut to add a flag through the hexdump widget
2. Multi Line Operations / Multi Line Selection
    a. Allow Selection of Multiple Lines by drawing a rectangle with the mouse
        i. Users should be able to draw a rectangle with their mouse on the output to select multiple lines.
    b. Allow adding additional individual lines by shortcut like Shift + Left Click
        i. Users should be able to add individual lines to the selected lines via a keyboard shortcut such as “Shift + Left click”
3. Semantically-aware syntax highlighting
    a. Ensure Rizin has the proper programming to handle semantically-aware syntax analysis
        i. Develop a semantic analysis module for Rizin
        ii. Extend existing data structures to include semantic data

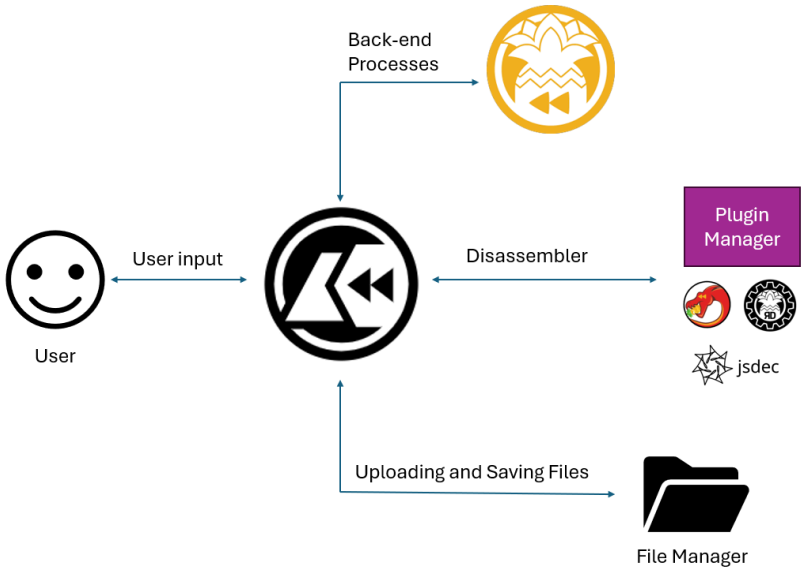
iii. Modify Rizin API to allow Cutter to request semantic changes

b. Enhance Cutter's syntax highlighting features
   i. This will be done by creating a Cutter plugin to integrate our semantic analysis backend changes for Rizin, or integrating our changes directly

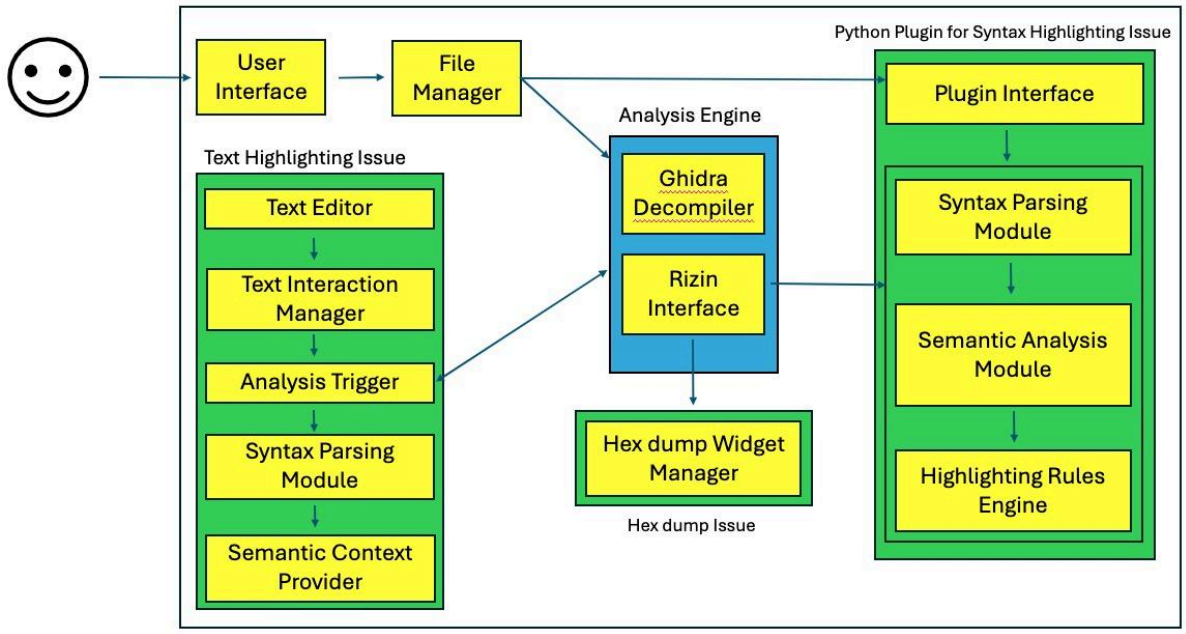
Context Diagram

This context diagram describes the interactions between cutter and 4 other external parties.

The user uploads a file from the file system to Cutter to be reverse engineered. To do the reverse engineering, Rizin runs as the back end support to the Cutter program and various plugins such as the Ghidra compiler also work to support the reverse engineering process of Cutter



System Diagram



Green - Issues we are fixing/adding
Blue - Main component of cutter (Analysis Engine)

- **User Interface** - This is where the user interacts with the system
- **File Manager** - Manages importing and exporting files, interfacing with the file system (first point of contact for user commands related to file operations)
- **Analysis Engine** - Main components of Cutter
    - **Ghidra Decompiler** - Integrated Ghidra's decompilation capabilities, translating disassembled binary into higher-level representations
    - **Rizin Interface** - Core engine, handling disassembly, analysis and manipulation of binaries
- **Hexdump Widgit Manager** - Manages the Hexdump Widget and Hexdump output
- **Text Editor** - Allows editing the text of the file that has been inputted
- **Text Interaction Manager** - Handles user input, like selecting text
- **Analysis Trigger** - Determines which text or snippet to Rizin or Ghidra for disassembly or decompilation analysis
- **Syntax Parsing Module** - Interprets the text structure and prepares it for highlighting or selection

- **Semantic Context Provider** - If the selection requires semantic information like a jump instruction, it would fetch results from Ghidra/Rizin.
- **Plugin Interface** - Interface where plugins are managed, connector between external plugins and the Cutter program itself
- **Semantic Analysis Module** - Uses predefined rules to analyze the parsed syntax to understand its semantic context
- **Highlighting Rules Engine** - Based on the semantic analysis, this engine would determine the highlighting rules to apply such as color or styles for different syntax elements

References

https://cutter.re/
https://github.com/rizinorg/cutter
https://rizin.re/
https://ghidra-sre.org/

Libraries

Library	Component	Description
Rizin	API	Modify the API to provide enhanced semantic data from the backend to the frontend
Qt Framework	UI/Context Menu UI development	We will implement UI changes using tools that integrate Qt for the respective IDEs we will each be using (VSCode, CLion)
CMake	Building	Using CMake to build our changes during development
git-clang-format 16	Code formatter for C/C++	We will use this clang-format library to ensure our commits align with the code guidelines

		of the Cutter/Rizin GitHub repositories.
Confluence, Discord Server with GitHub webhook	Project Management	● We will use Confluence for project management and to track progress. ● We will also use a Discord server for discussion and to receive notifications of new commits, pull requests, new deployments, etc. to the GitHub repositories.