EDB: Debugger for Ethereum's Programming Languages

Report #2: Justification & Feasibility
Advised by Dr. Jackowitz
University of Scranton

Andrew Plaza

September 21, 2018

Contents

0.1 Justification

0.1.1 Project Justification

The goal of this project is to help fill in the gaps of the current experience developing applications on the Ethereum Blockchain. This project specifically addresses the lack of a robust, full-featured and un-opinionated debugger for the most popular Ethereum Programming languages Solidity and Vyper. Current debugger implementations force a user to debug through a specialized web application, only debug one of Ethereum's languages, or force the use a very particular Ethereum Test JavaScript Object Notation Remote Procedure Call[1] (TestRPC/JSON-RPC) implementation. For many developers, this is not ideal, and requires extra work to integrate with these solutions. In addition, some of these solutions prevent users from integrating with existing popular workflows such as Visual Studio Code or IntelliJ IDE's completely.

There are currently 1,861 Ethereum Decentralized Applications[2] (Dapps) available for anyone to use, with that number growing daily. This is impressive considering Ethereum has only begun development four years ago, while gaining mainstream media attention and acknowledgment as a result of it's price only approximately one year ago. This rapid development of Dapps has resulted in a sub par development experience, since the speed at which tools and development for the core Ethereum protocol occur at a slower pace. This is analogous to the development of Smart Phones and apps. There are simply less people working on the tools and software that allow these apps to run than the developers building upon those platforms. Often, Dapp-developers struggle with a variety of issues, namely: security, quality assurance, and testing. This is mostly due to the permanent nature of Blockchain. Once a developer commits their code to the official Ethereum Blockchain, it cannot be modified or changed so any vulnerabilities or bugs that exist will continue to exist for as long as the Ethereum Blockchain stays online. Therefore, it is my belief a debugger that integrates with the most popular developer workflows, while covering most use cases will make it simpler for developers to test, secure, and ensure code that is free of more bugs.

0.1.2 Language Justification

Past work on this project, in 2017, was done using the NodeJS runtime using Javascript. This project will not use NodeJS. Rather, parts of previous work will be rewritten to use Rust. The choice of Rust for this project, instead of Javascript was out of concern for performance, maintainability, and usability. Javascript Node Package Manager (NPM) dependencies grew quickly, which proved difficult to maintain in the first version. Dependencies grew out-of-date, or were unmaintained by the project owner. Additionally, Javascript is

[1] RPC Specification *https://tools.ietf.org/html/rfc5531, https://www.jsonrpc.org/specification*
[2] StateOfDapps *https://www.stateofthedapps.com/*

loosely-typed, which proved difficult to document, since additional documentation on the exact types was required. In addition, this added to the difficulty of maintenance. Conversely, NodeJS runtime performance was sub-par, and the debugger would benefit greatly from a language which took performance into greater consideration.

On the contrary to Javascript, Rust's performance is comparable to that of C and C++ while also including modern-day programming language paradigms. Among these include functional features, a form of object-orientation, and uniquely includes memory-safety. Secondly, I'd like to stress that many popular Ethereum frameworks and protocols have implementations in Rust. This makes it significantly easier to integrate into the Ethereum ecosystem, as well as simplify the usage of any libraries that may be important to use.

0.2 Feasibility

The completion of this project requires development of a few main components:

1. A Core Open-Source (MIT Licensed) Library to provide the main functionality
   (a) An interface over an Ethereum Virtual Machine to provide debugger functionality (Step Back, Step Forward, Step Into, Next, Info, Backtrace)
   (b) An module to provide Source Mapping (Source Code $\rightarrow$ Bytecode) for Solidity and Vyper
2. A module exposing debug functions (as in item '1.a') over JSON-RPC
3. A Client which can interact with the RPC, and provide a pleasant user interface

Much intensive and important work for this debugger has already been done in my first version of this project, ethdbg[3]. This includes work on Source-Mapping, and Debug Functions. Source Mapping was completed, as well as step forward, and continue debug functions. Thus, I already own past work on this project that contains significant progress, and I have accumulated a fair amount of information concerning implementation and development detail.

In consideration of the timeframe, I fully expect to complete the core library with at least support for Solidity, a simple JSON-RPC interface, and a simple CLI client to demonstrate the usage of the debugger. In addition, a Visual Studio Code plugin and support for Vyper and LLL will be a stretch-goal. However, I expect to be able to complete this as well considering I am working from the perspective of the work I have done for a debugger written in NodeJS.

[3]ETHDBG *https://github.com/ethdbg/ethdbg*