Gera Jahja

Class of 2022

IN3007 Project Report

Decompiling Ethereum EVM Bytecode for Static Analysis.

Decompiling Ethereum EVM Bytecode for Static Analysis.

* Prepared by: Gera Jahja, Computer Science BSc
* Prepared for: City UOL, Department of Computer Science
* Email: gera.jahja@city.ac.uk / g\_jahja31@outlook.com
* Consultant: Martin Nyx Brain
* Academic Client: Martin Nyx Brain and Michał Król
* Version Control:
  + V1.0:16th Feb 2022: Introduction and Project Definition sections added (pages 1-7, Word Count: 1293/15,000)

This project is an academic proposal supervised by Martin Nyx Brain and Michał Król. The project will be to write a decompiler so that it can convert EVM byte code into C that can be handled by the CPROVER tools.

Table of contents

[Introduction 2](#_Toc95864683)

[Project Objectives: 2](#_Toc95864684)

[Project Plan 3](#_Toc95864685)

[Project Structure: 3](#_Toc95864686)

[Iterative Development: 3](#_Toc95864687)

[Project Beneficiaries: 4](#_Toc95864688)

[Risks of the project: 5](#_Toc95864689)

[Project Management: 5](#_Toc95864690)

[Technical: 5](#_Toc95864691)

[Personal: 5](#_Toc95864692)

[Risk assessment: 6](#_Toc95864693)

[References: 7](#_Toc95864694)

## Introduction

Ethereum is one of the most exciting block chain technologies as it is not just a cryptocurrency but also supports smart contracts. These are programs that are written in a variety of programming languages and compiled to EVM, a byte-code format like JVM, before they are run on the block chain. The security of these contracts is vital as they can control significant amounts of cryptocurrency.

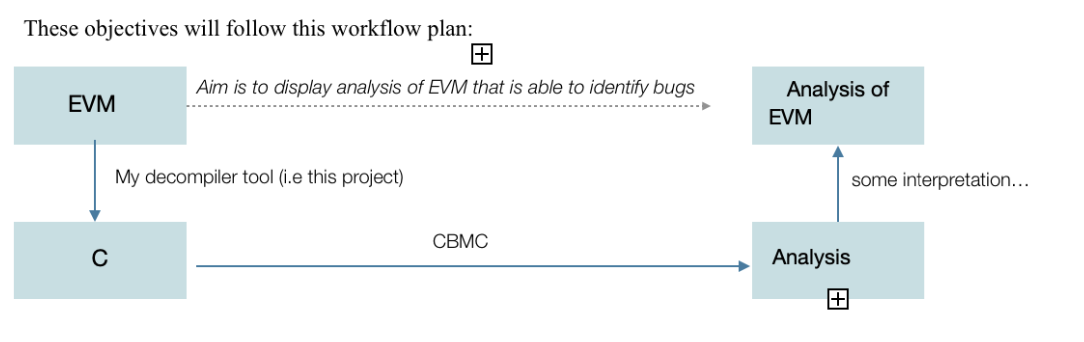
This project requires me to translate EVM byte code to C code. The purpose of this translation is to see whether we can detect bugs using CPROVER tools. If we can successfully use tools used to detect C code bugs on EVM this means we can add verification to EVM (as well as aiding our understanding of the behaviour of the smart contracts) and ensure that the Ethereum currency that is associated with the byte code is protected and less viable to hacking. Decompiling is a part of reverse engineering, I will be using this approach to convert EVM byte code to op code, and then generating C code from this.

When looking to verify whether the software works, I will be specifically be looking at:

- Will the program crash? Can it be hacked? Can we do it without running the program? (Static Analysis, i.e., using CPROVER)

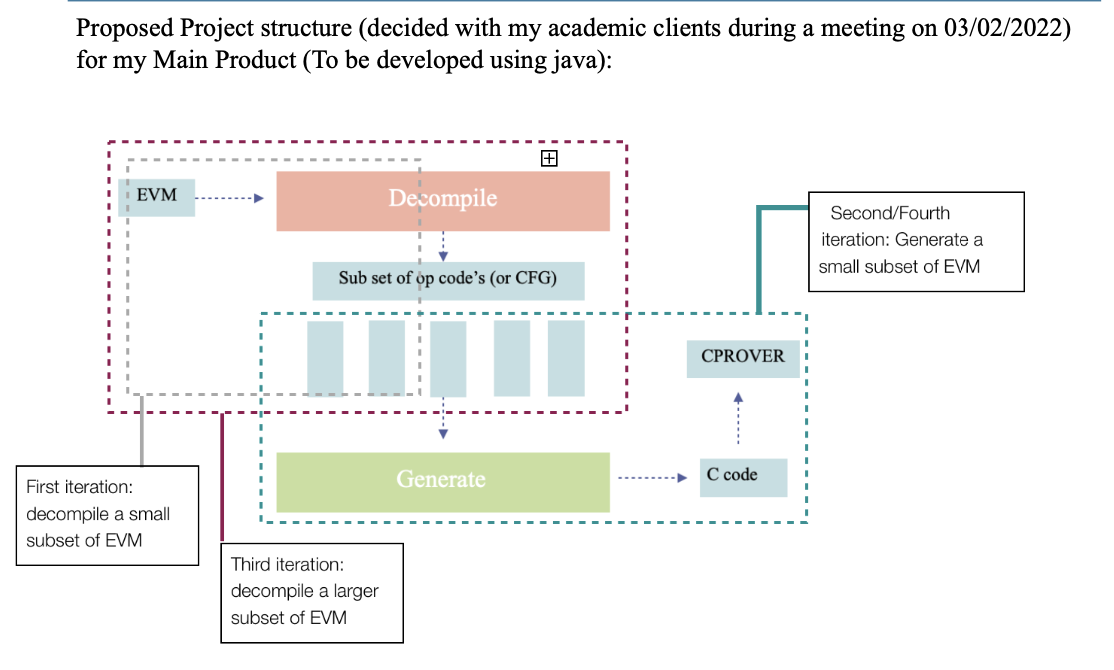
The Verification tool for C is CBMC and there are some verification tools for EVM or Solidity, etc.

## Project Objectives:

* Produce a subset of EVM byte code that has been decompiled to C code.
* Develop a program that displays this decomplication
* Ensure this subset (of EVM byte code) can be run through the CPROVER tool CBMC -Investigate to what degree do these tools allow us to analyze the EVM contracts, I.e. is it able to aid us in detecting bugs?

## Project Plan

### Project Structure:



### Iterative Development:

I will be splitting my project into three builds and will have **numbered objectives** for each:

1. **Minimum Viable Product:** Complete by mid-March:

* Research all existing implementations of similar projects (2-3 days)
* Write-up report (3-4 days)
* The program will take some EVM code and decompile it , translate into a list of Op code (or a CGF) (2+ weeks, depending on whether the development goes smoothly)
* Get client feedback, test that this product has no syntax or logical errors. (2-3 hours)

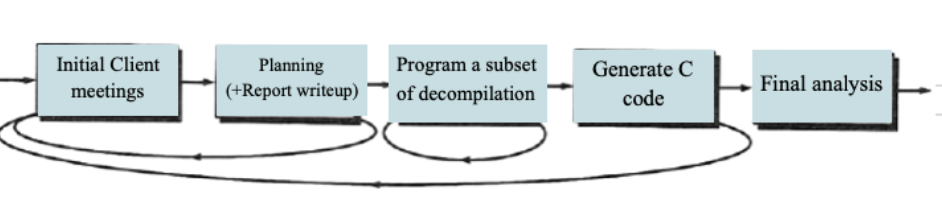
1. **The Main Product:** functional but with no extra subsets of op code: Complete by mid-April :

* Extend Report with updates (3-4 days)
* The program will take the decompiled Op code and generate reliable C code AND will successfully be usable by CPROVER tests(3+weeks)
* Get client feedback, test that this product has no syntax or logical errors. (2-3 hours)

1. **Additional Features:** Complete towards the start of May:

* Look at the behaviour of the decompiled C program and add additional features based on the main product’s current output(1+week(s))
* Add additional op code features (extending the decompiler and code generator made in the first two iterations) (2+ weeks)

Refer to this iterative workflow diagram (each stage must go through testing or a client meeting before moving on to the next step, and preferably each iteration SHOULD take 3 weeks (4 max)):



## Project Beneficiaries:

Beneficiaries of the project include:

1. My clients and I:
   * Martin is quite interested in the verification end of the problem we are trying to solve, while Michael is interested in the Ethereum cryptocurrency side of things. My aim is to pave a connection between the two.
2. Academics
   * The findings of this development may be of interests to academics in similar fields of work, such as cyber security, decomplication and more.
3. People using blockchain
   * If we can validate smart contracts, it can prevent huge financial losses for people that are part of the Ethereum blockchain.
   * In the past there have been times where hacking into the currency has led to cryptocurrency being destroyed. In 2017, “$300m of cryptocurrency was lost after a series of bugs in a popular digital wallet service led one curious developer to accidentally take control of and then lock up the funds” Locating these bugs can prevent similar cases from occurring, adding more security to the currency and more protection for people using block chain.

In general, building tools that are useful is the main benefit of this project. Finding and ideally preventing bugs with evidence that it works would be the ideal outcome of this project.

## Risks of the project:

### Project Management:

* The EVM byte code may be difficult to decompile into C and may not produce perfect C code. This should not be an issue if my prior research is well done at the start of the project. This research will be continued throughout the project in case I discover more up to date examples.
* What the CPROVER identifies as a bug may not be a bug for the EVM code... is this an accurate assumption to assume C code bugs are the same as EVM bugs? To refrain this from preventing the success of my project I will be working using an iterative approach, so if the first iterations fail, I will have enough time to plan a new method.

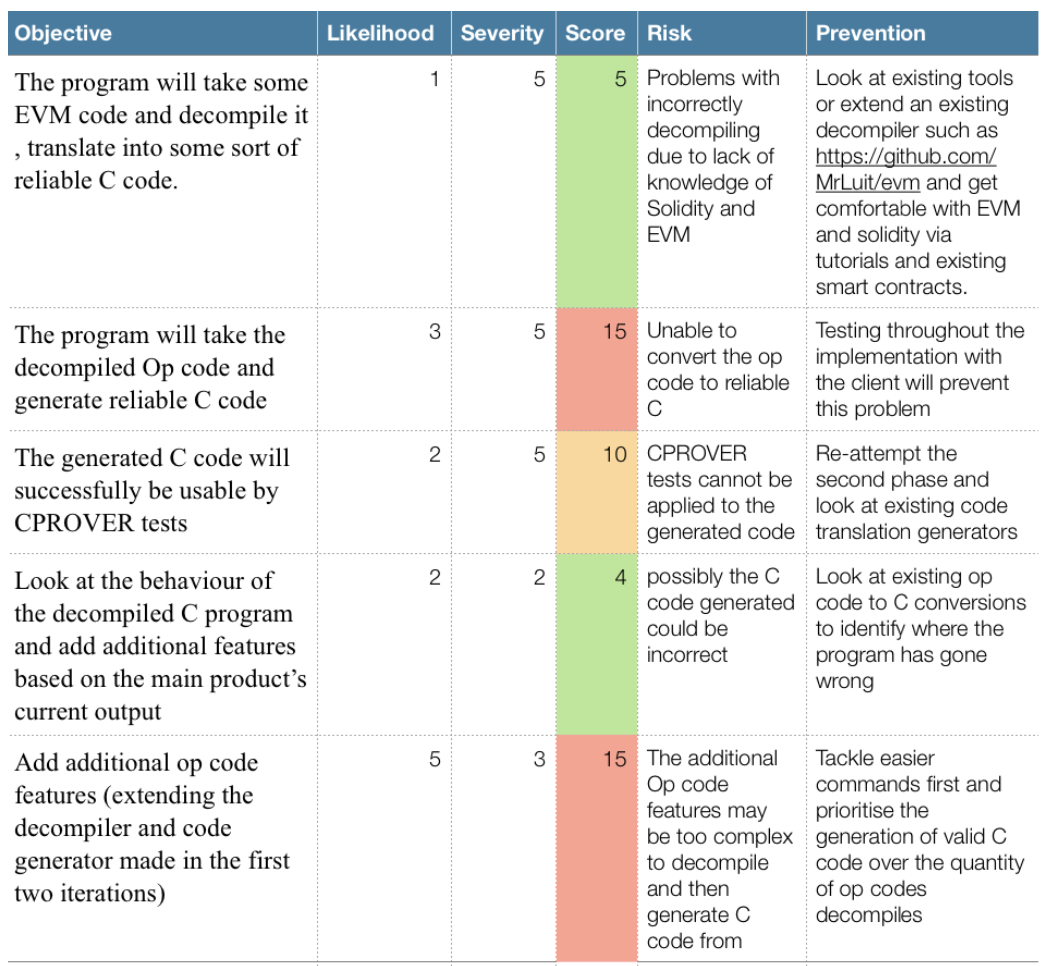
### Technical:

* What if the systems don’t work, I.e., platforms and breaking of tools? I will be displaying mitigation and proof of my attempts for the solutions, I am also prepared to use virtual machines, (for windows dependencies as I am using a MacBook.)
* Am I able to produce a program to satisfy the Academic Client? Regular Client meetings every week will prevent this from happening

### Personal:

* There may be Covid restrictions that prevent some meetings with the client. To prevent this, I am prepared to communicate using online meetings if the situation calls for it.
* Will the detection of bugs possibly be incorrect? As we are dealing with cryptocurrency the risks of being responsible for the authentication of smart contracts could cause losses to people relying on the tool if it is used and is incorrect.
* How will the tool be used? And what for? This project could be an example of dual use technology, while I am aware of possible concerning uses of this type of tool, if I find bugs, I will not be using my findings in a malicious manner.
* If we find a bug this could be ethically wrong if it's used for personal gain. If the tool is used as a form of Responsible disclosure, then it is ethical. However, if the tool is used unethically then knowledge of bugs in EVM programs, that are supposed to be safe, could put a lot of smart contracts in danger.

### Risk assessment:

* Likelihood (of this being a risk): 1-5 (1 being very unlikely, 5 being very likely)
* Severity (how necessary this is): 1-5 (1 being not impactful, 5 being very impactful)
* Score is the Likelihood\*Severity, (0-6 is low risk, 6-14 is medium risk, 15+ is high risk) 

## References:

1. The Guardian. (2017). *“$300m in cryptocurrency” accidentally lost forever due to bug*. [online] Available at: https://www.theguardian.com/technology/2017/nov/08/cryptocurrency-300m-dollars- stolen-bug-ether [Accessed 3 Feb. 2022].