GasCore+ Inception

Team 5
Team Members: Abbinav H. Burhanuddin C.
Dhairya C. Ritesh M.
CSE 6324-001

Introduction

Smart contracts are software programs that are stored in blockchain that engage with highly valuable financial assets and are used to automate the execution of an agreement between two intermediaries [5]. Due to the sensitivity of the transactions, minor mishaps can lead to irreparable damage and major financial losses for users. Comprehensive testing plays a vital part in exposing errors within smart contract code and reduces any potential vulnerabilities. Smart Contracts deployed within the Ethereum Virtual Machine(EVM) are immutable and therefore traditional approaches of fixing defects after launching are not applicable and require runtime testing to patch vulnerabilities [5].

GitHub Repository

- Link: GitHub - Abbinav/CSE-6324-Team5: GasCore+ Implementation
- Version: 0.1

Project Vision

The major point of concern for the Smart Contract Technology is the handling of safety and reliability to avoid major financial losses. Static Analysis , Dynamic Analysis and Symbolic Execution are such testing techniques that can help evaluate the security of the smart contracts.

Static Analysis

Static Analysis examines the smart contract's byte code prior to execution. Static analyzers help us in detecting common vulnerabilities within the Ethereum Smart contracts without running the program [12].

Dynamic Analysis

Dynamic Analysis identifies issues within our smart contracts while executing during runtime and provides a report of vulnerabilities and property violations. Fuzzing is a common dynamic analysis technique for smart contracts that evaluates how smart contracts can handle invalid data and inputs [12].

Symbolic Execution

Symbolic execution is a testing technique that helps us find all execution paths of a program and generate the corresponding input for each path. This ensures that all inputs of a program are tested and all possible execution parts are explored[8].

The combination of these three techniques provides us a platform to develop an

analysis tool that could avoid a majority of security vulnerabilities while providing a safe platform for users to conduct transactions through the Ethereum blockchain. The vision of this project is to propose a tool called GasCore+ that utilizes a combination of two existing tools, GasGauge and MantiCore to provide a one stop analysis solution for smart contracts. GasCore+ will incorporate the Static and Dynamic Analysis of GasGage that focuses on gas related vulnerabilities within smart contracts and the Symbolic Execution of MantiCore that will focus on Input Generation and Error Discovery [9].

Features

- GasCore+ is a console based application and its source code will be available on github after the implementation.

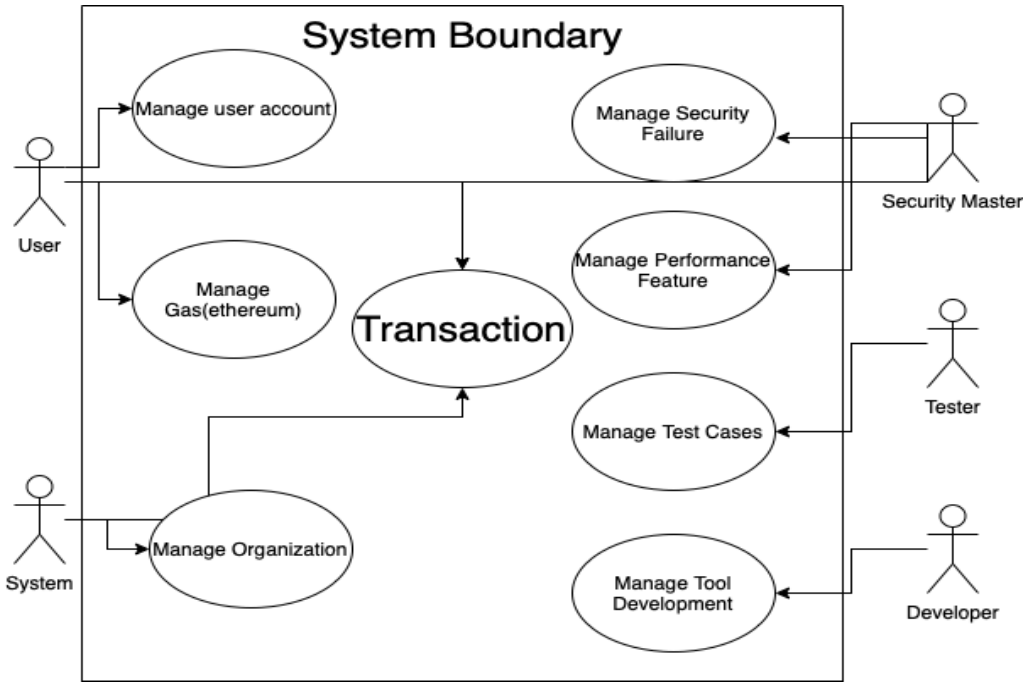
- GasCore+ estimates the gas cost of a running function and informs the users about any vulnerability. [13]

- GasCore+ is divided into three phases: the Detection phase, Identification phase, and Correction phase. All the stages of the tool can work alone or together to analyze Ethereum smart contracts. [13]

- GasCore+ will help us analyze gas related vulnerabilities before the execution in ethereum smart contracts ie It will be able to carry out **Static analysis**.[12]

- GasCore+ will also help us analyze gas related vulnerabilities after the execution in ethereum smart contracts ie It will be able to carry out **Dynamic analysis.**[12]

- GasCore+ will help us test all the execution paths within the program ie It will be able to carry out **Symbolic execution.**[8]



User Stories		
Type of User	**Goal/Objective**	**Benefit/Result**
As a End User/Customer	I want to carry out secure transactions	So that I don't end up loosing my Gas(Ethereum)
As a Developer	I want to build a tool that can include static analysis, dynamic analysis, and symbolic execution	So that I deliver a good Product
As a Owner of Company	I want to sell the right tool/ product	So that I don't loose the trust of my Customers
As a Tester/Smart Contract Executer	I want to find all the possible vulnerabilities inside the program	So that I can make sure I'm deploying the right program on EVM

Customers & Users

Customers

- Dr. Christopher Csallner
    - Dr. Csallner will provide the team assistance with any issues, concerns, or questions regarding Ethereum Smart Contracts, project deliverables, and deadlines.
    - Dr. Csallner will provide feedback to the team about their presentation performance during presentation days and feedback about written deliverables.

- GTA Mohammed Rifat
    - Mohammed will also provide the team assistance with any issues or concerns regarding the project's deliverables and deadlines.
    - Mohammed will also provide the team with feedback at the end of every iteration which the team will use to improve written deliverables, presentations, and code production/optimization.

- Shovon Niverd Pereria
    - Shovon will be the team's collaborator for this project, providing the team with information regarding the functionality of Etherum smart contracts and clarifying the team's questions.

- Review Team 6
    - Review Team 6 will provide the team with frequent feedback regarding the team's progress for every iteration presentation and deliverable.
    - Review Team 6 will also provide frequent feedback for the team's code for current running iteration with code reviews and comments on possible changes/improvements.

Users

- Class of CSE 6324-001
    - The class of CSE 6324-001 will be the team's target audience and possible future users for the project. The class will provide us feedback by asking questions about our project functionality. With the classes' feedback, the team will be able to improve the tool's functionality to better reflect the user's wants and needs that they are looking for.

Software Development Plan

The main goal of GasCore+ is to detect vulnerabilities in Ethereum Smart Contracts. The main tasks are to implement Static Analysis, Dynamic Analysis and Symbolic execution of Smart contracts to detect vulnerabilities in the execution.

Since GasCore+ aims to bring together the static analysis and dynamic analysis from GasGauge and Symbolic execution from MantiCore, the main task of the team will be to bring together these functionalities into one program.

The timeline to complete the development is two months and the project will be complete in four iterations.

Iteration 1 (2 weeks)

This iteration will include setting up a plan for development. Features of GasGauge and Manticore which will be implemented in GasCore+ will be listed and each team member will be assigned the responsibility for a part of the development, and the team will start working towards creating an initial prototype.

Iteration 2 (3 Weeks)

The team aims to have a working prototype of GasCore+ during this iteration. With a prototype in hand, the team will make decisions as to which features are not needed and which features will need to be added and the team will work towards implementing them.

Iteration 3 (3 weeks)

During this iteration, the testing of GasCore+ will commence. After adding all the necessary features, the team will pick available smart contracts to test and individually conduct static analysis, dynamic analysis and symbolic execution using GasCore+ to check its efficiency.

Iteration 4 (2 weeks)

During the final iteration, the team aims to have completed the development of GasCore+. Multiple tests will be run to ensure all the features and requirements of GasCore+ are functioning and the desired results are achieved.

Out of Scope:
User Interface - Due to a time constraint, the team will only work towards achieving all the basic core functionalities required for GasCore+. The team will not be developing a user interface for GasCore+.

Functionality - As the number of open source smart contracts are publicly available limited, GasCore+ will not be able to assure proper functionality and compatibility with all types of smart contracts.

Risks Management Plan

Top 3 Possible Project Risks

- Technical Risk
    - The team will attempt to combine two separate Python programs into a single program, but the team does not know if the combination of both programs will properly compile, run and produce the same results as the two separate programs.

- Resource Risk
    - The entire team is not familiar or does not have enough experience with the Ethereum Smart Contracts to work fluently in the EVM environment.

- Schedule Risk
    - The team's work schedules are in constant flux during the entire period of the project and will lead to conflicts with other classes' assignment due dates.

Mitigation Plan

- Technical Risk Mitigation
    - Mitigate: The team will try their best to understand the structure of both programs and will consult with an expert in the Python language for the best way to combine both programs that will reduce the chances of compilation errors and produce the best results similar to the separate Python programs.

- Resources Risk Mitigation
    - Mitigate: Team members can research Ethereum Smart Contracts in more depth for a better understanding and consult with Shovon to clarify any confusions regarding EVM and smart contract functionalities.

- Schedule Risk Mitigation
    - Mitigate: The team will remain flexible throughout the period of the project to take on any extra tasks while a team member is working on different class assignments. Using Microsoft Teams, the team will communicate

any issues or concerns regarding their tasks such as requesting extra time to complete the task and any deadlines approaching for assignments.

Risk Exposure

- Technical Risk Exposure
    - Probability of Risk Occurring (PR) = 50%
    - Effect of Risk Occurring (ER) = 10 hours will be spent
    - Risk Exposure (RE) = 50% (PR) * 10 hours (ER)
    - Risk Exposure (RE) = 5.0 extra hours

- Resources Risk Exposure
    - Probability of Risk Occurring (PR) = 30%
    - Effect of Risk Occurring (ER) = 8 hours will be spent
    - Risk Exposure (RE) = 30% (PR) * 8 hours (ER)
    - Risk Exposure (RE) = 2.4 extra hours

- Schedule Risk Exposure
    - Probability of Risk Occurring (PR) = 25%
    - Effect of Risk Occurring (ER) = 12 hours will be spent
    - Risk Exposure (RE) = 25% (PR) * 12 hours (ER)
    - Risk Exposure (RE) = 3.0 extra hours

Constraints

- Scheduled delivery date
    - The written deliverables, presentations and repo checks have to be ready for submission at the end of each iteration. The submission has to be ready to go to Dr. Christopher Csallner, GTA Mohammed Rifat, and Review Team 6 for reviewing, grading of quality, and feedback for improvement for the next iteration. Delays in submission would cause delays in reviewing, grading, and feedback resulting in an incomplete project by the final scheduled delivery date.

Competitors

Tool Name	Method	Description
GasCore+	**Static + Dynamic Analysis + Symbolic Execution**	**GasCore+ is a combination of Static/Dynamic Analysis and Symbolic Execution methods. GasCore+ utilizes the Static and Dynamic Analysis of GasGuage to identify any gas related vulnerabilities while also incorporating Slither as an API to write custom analysis. Additionally GasCore+ will use the Symbolic Execution from MantiCore for Input Generation and Error Discovery to provide an**

		overall analysis tool that will help identify the large majority of issues and vulnerabilities.
GasGuage	Static + Dynamic Analysis	GasGuage is a combination of Static and Dynamic Analysis that focuses on gas related vulnerabilities that ensures that Solidity gas does not surpass the gas limit during transactions. [6]
MantiCore	Symbolic Execution	MantiCore is a symbolic Execution tool for the analysis of Smart contracts. The main features of MantiCore include Program Exploration, Input Generation and Error Discovery. [9]
MPro	Static Analysis + Symbolic Execution	Mpro is a smart contract analyzer that utilizes Slither to combine the Static Analysis with the Symbolic Execution from Mythril. [8]
Slither	Static Analysis	Slither is a Solidity based Static Analyzer tool that detects vulnerabilities and provides an API to write custom analysis. [10]
Mythril	Symbolic Execution	Mythril utilizes Symbolic Execution and SMT solving to identify security vulnerabilities in smart contracts built for Ethereum.[12]

References

1] Risk Mitigation Plan Case Studies Slide Design - SlideModel | Case study template, Case study, Management infographic (pinterest.com), accessed 09/11/2022

[2] The Benefits of Risk Management Planning – Bellevue University Project Management Center of Excellence, accessed 09/11/2022

[3] ICEFLO | ScreenSteps, accessed 09/11/2022

[4] Risk Assessment Analysis: Exposure and Target Accounts for Risk Hedging - FinanceTrainingCourse.com, accessed 09/11/2022

[5] Bistarelli, S., Mazzante, G., Micheletti, M., Mostarda, L., Sestili, D., & Tiezzi, F. (2020). Ethereum smart contracts: Analysis and statistics of their source code and opcodes. *Internet of Things*,

[6] Ashraf, I., Ma, X., Jiang, B., & Chan, W. K. (2020). GasFuzzer: Fuzzing Ethereum Smart Contract Binaries to Expose Gas-Oriented Exception Security Vulnerabilities.

[7] (2020, March 19). *Crypto Market Pool - Gas in Solidity smart contracts*. Crypto Market Pool. https://cryptomarketpool.com/gas-in-solidity-smart-contracts/, accessed 09/11/2022

[8] W. Zhang, S. Banescu, L. Pasos, S. Stewart and V. Ganesh, "MPro: Combining Static and Symbolic Analysis for Scalable Testing of Smart Contract," in 2019

[9] Mark Mossberg, Felipe Manzano, Eric Hennenfent, Alex Groce, Gustavo Grieco, Josselin Feist, Trent Brunson, and Artem Dinaburg. 2019. Manticore: a user-friendly symbolic execution framework for binaries and smart contracts

[10] Liu, Y., Xu, J., Cui, B. (2022). Smart Contract Vulnerability Detection Based on Symbolic Execution Technology. In: Lu, W., Zhang, Y., Wen, W., Yan, H., Li, C. (eds) Cyber Security

[11] https://www.cprime.com/wp-content/uploads/2020/09/Untitled-3.png, Accessed 09/11/2022

[12] https://lightrains.com/blogs/solidity-static-analysis-tools/ , Accessed 09/10/2022.

[13]https://ieeexplore.ieee.org/document/9762279, Accessed 09/11/2022.