

Abstract

Blockchain wallet transaction security is a long-standing challenge. Currently, there are many security challenges in blockchain wallets, such as private key loss, transaction tampering, and wallet hacking. These security risks have brought huge financial losses to users. Therefore, designing a secure scanning platform for blockchain wallet transaction becomes crucial. I proposed a platform with three main functions, including wallet address transaction history query and visualization, risk address and website detection, smart contract security scanning. By implementing these functions, the platform will act as a security tool by providing users with the security scans, helping them avoid potential security risks and financial losses, thereby increasing user confidence in blockchain transactions. For developers, smart contract security scanning can help them discover security vulnerabilities in smart contract code and improve development efficiency. Finally, I will enhance the practical significance and application value of the blockchain wallet transaction security platform, providing users with an effective solution that is expected to promote a safer transaction environment in the blockchain field.

Table of Contents

Abstract	1
1. Introduction.....	5
1.1 Dissertation Structure.....	5
1.2 Motivation.....	6
1.3 Aim and Objectives	8
1.3.1 Project Aim	8
1.3.2 Project Objectives.....	8
1.3.3 Description of Objectives.....	8
1.4 Project Plan	9
1.5 Ethical Considerations	10
2. Technical Background.....	10
2.1 What is Blockchain Wallet?	10
2.1.1 How Blockchain Wallets Work.....	11
2.1.2 Features of Blockchain Wallet.....	12
2.2 Blockchain Wallet Research.....	13
2.2.1 Usability Analysis in Different Blockchain Wallets.....	14
2.2.2 Security Problems in Different Blockchain Wallets	16
2.3 Relative Platform Research.....	18
2.4 Data Resources Descriptions / Solutions.....	22
2.5 Summary	23
3. What was done, and how?	24
3.1 Development Process.....	24
3.2 Development Requirements	25
3.2.1 Front-end Development	25
3.2.2 Back-end Function Development	25
3.2.3 Testing and Debugging	25
3.3 Main Tools and Technologies.....	25
3.4 System Architecture Design	28
3.4.1 System Architecture Diagram and Description.....	28
3.4.2 Components Diagram and Description.....	29

3.5	System User Interface Design.....	31
3.5.1	Transaction Page Design and Description.....	31
3.5.2	Address Page Design and Description.....	32
3.5.3	Smart Contract Page Design and Description.....	33
3.5.4	Web Page Components Description.....	34
3.6	Front-end, Back-end Development.....	35
3.6.1	Master Files in Front-end and Back-end.....	35
3.6.2	Transaction Page and Functions Development.....	36
3.6.3	Address Page and Functions Development.....	38
3.6.4	Smart Contract Page and Functions Development.....	40
3.7	Testing Methodology.....	42
3.7.1	Front-end Usability Testing.....	42
3.7.2	Back-end Unit Testing.....	42
3.8	Summary.....	43
4.	Result & Evaluation.....	44
4.1	Result of Testing Methodology.....	44
4.1.1	Front-end Usability Testing.....	44
4.1.2	Back-end Unit Testing.....	48
4.2	Project Evaluation.....	52
4.2.1	Test Result Evaluation.....	52
4.2.2	Development Process Evaluation.....	53
4.2.3	Develop Requirements Evaluation.....	55
4.3	Summary.....	56
5.	Conclusions.....	57
5.1	Review of Aim and Objectives.....	57
5.1.1	Project Aim.....	57
5.1.2	Project Objectives.....	57
5.2	Updated Project Plan.....	58
5.3	Personal Conclusions.....	59
5.3.1	What I Learned.....	59
5.3.2	What Went Well.....	60
5.3.3	What Could Be Better.....	60

5.4	Work in The Future and Research.....	61
5.5	Final Thoughts	62
6.	References	63
7.	Appendices.....	65
7.1	Appendix A – APIs Response Data.....	65
7.2	Appendix B – Data Processing Test Data	69
7.3	Appendix C – Tools Call Test Data	74

1. Introduction

In this chapter, I present the complete structure of the dissertation. After that, I will introduce the background of the project and current problems, use data to prove the seriousness of the problem, propose the problems I need to solve and the development direction of the project. I also fully describe the aim of the project and the objectives that need to be accomplished. Finally, a complete Gantt chart project plan will be presented, and ethical considerations are provided.

1.1 Dissertation Structure

My dissertation will be divided into five parts:

1. Introduction: this chapter provides the background of the project and the problems that need to be solved and proposes the direction of development of the project. It also gives a complete description of the project objectives and what needs to be achieved. At the end of the introduction is a complete Gantt chart project plan and ethical considerations.
2. Technical Background: technical background research is important when starting a project. This chapter will introduce the blockchain wallet and cover the investigation I conducted during the technical background research, showing all the information and knowledge gathered, as well as the technology I will use in the project.
3. What was done, and how: this part will show the process of my project development. It covers the topic of each work I did in the project and provides a detailed description of the corresponding work. At the same time, it will show the problems I encountered during the project and the ways to solve them.
4. Results & Evaluation: after the project is developed, it is the results and evaluation of the project. I will provide a complete summary of the project, providing feedback on the results of the functional implementation and practical testing. It also considers the achievement of the project objectives to ensure that the aim of the project is met.
5. Conclusions: I take conclusions as the end of the article. I will give a full reflective summary of the whole project in this chapter. Indicate what I have learned in the project, assess the integrity of the project, and return to the overall aim. At the same time, provide my objectives in the future work.

1.2 Motivation

With the rapid development of cryptocurrencies, an increasing number of people are using them for transactions and investments. The cryptocurrencies market has experienced rapid growth and has become an important part of the global financial system. Howarth(2022) provided a data, there are over 295 million crypto currency users all around the world until 2024. [1] Blockchain wallet is an important tool for holders to store, manage, and trade their crypto assets. With the increase in user numbers and the rise in demand, users have higher requirements for the security and convenience of wallets. Although cryptocurrencies technology has a high level of security, users' blockchain wallets still face various security risks. Issues such as loss of private keys, tampering with transactions, and wallet hacking occur from time to time, causing users significant losses and troubles.

Yearly total value stolen in crypto hacks and number of hacks,
2016 - 2023

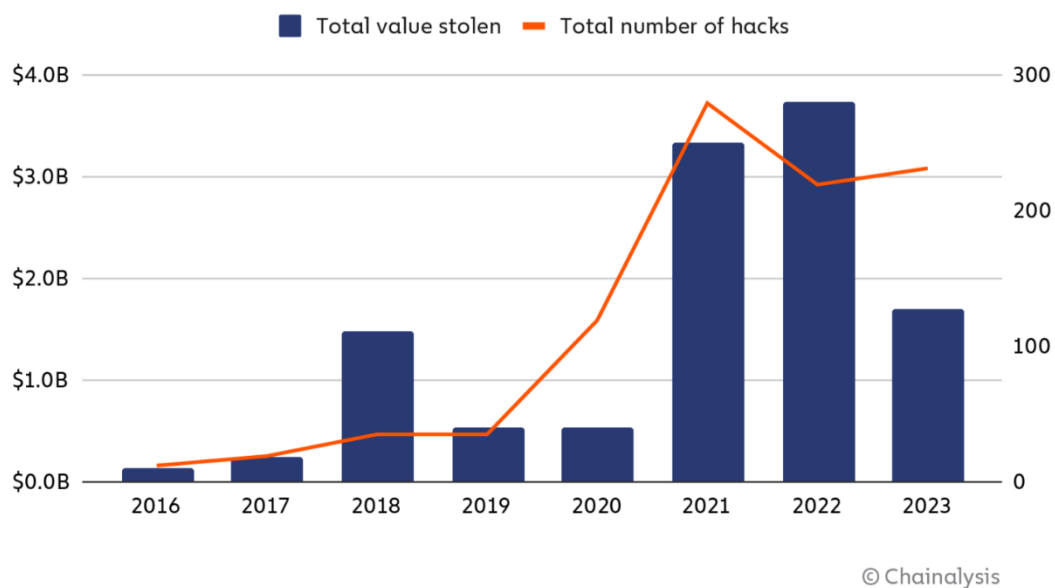


Figure 1.2.1: Crypto hacks from 2016 to 2024. [2]

Based on the Figure 1.2.1, it shows in the past two years, more than \$5.4 billion has been stolen from cryptocurrencies, while the number of stolen incidents has also increased to 231 in 2023. [2] Therefore, designing a secure scanning platform for blockchain wallet transactions is crucial to address these security issues.

Currently, the major problem is the blockchain wallets often lack effective transaction security checks and security detection before smart contract interaction. As a result, users face security risks such as risk wallet addresses, and phishing attacks when using blockchain wallets for operations and transfers.

Risk wallet addresses may deceive users into transferring funds to risk addresses through fake transaction requests or by inducing users to provide their private keys, resulting in financial losses. Also, when the user uses the blockchain wallet to interact with the smart contract, user cannot determine the authorized function of the smart contract to their wallet, which has huge security risks.

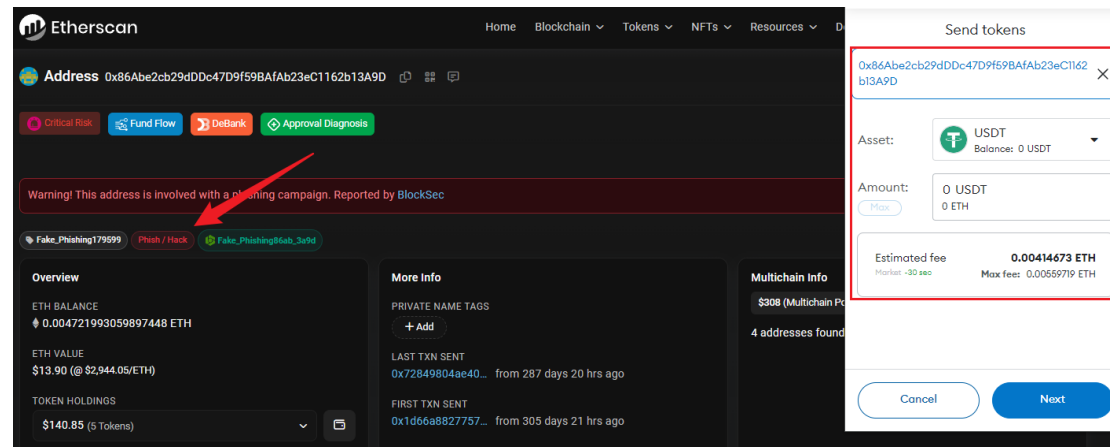


Figure 1.2.2: blockchain wallet lack security check.

In the Figure 1.2.2, even if an address has been flagged as a phishing attack or risk, the blockchain wallet on the right does not provide any security tips and allows transactions to continue. Furthermore, phishing attacks often exploit disguised information from seemingly trustworthy sources, such as emails, social media, or instant messages, to lure users into clicking malicious links, obtaining the private keys and authorizations of their blockchain wallets, thereby stealing their cryptocurrency assets.

To solve these problems, I will design a platform with three main functions in the project to help users avoid these problems: wallet address transaction history query and visualization, blacklist address and website detection, and smart contract security scanning. By implementing these functions, the platform can serve as a security scanning tool, providing users with the security scans they need before make a transaction or interact with the smart contract, helping them avoid potential security risks and financial losses.

1.3 Aim and Objectives

1.3.1 Project Aim

My aim in this project is develop a secure scanning platform to provide users with security scans before they make a blockchain wallet transaction, which is include three main functions:

1. Transaction history query and visualization: develop a function to allow users to query and visualize their transaction history from public blockchain data. Implement a function to generate transaction flow graphs for transaction tracking. In addition, this function will also flag in each transaction record whether it is a risky transaction.
2. Blacklist address and website detection: connect to public database through APIs, users can check if a specified wallet address or website is associated with risk or blacklisted activities.
3. Smart contract security scanning: provide users with the capability to perform security scans on smart contracts before interacting, safeguarding their funds from external calls. Also, it can provide a scan report to developers, help them to find the bugs in smart contract quickly.

1.3.2 Project Objectives

1. Analyze existing blockchain anti-money laundering tracking platforms.
2. Develop a functional understanding of blockchain wallet.
3. Collect public data APIs to implement fraud detection for addresses and websites.
4. Obtain public transaction data on the blockchain and visualization.
5. Tool learning and integration for smart contract security.
6. User interface design, usability testing and function development.

1.3.3 Description of Objectives

Objective 1: The tracking and visualization of blockchain transactions is an important security feature for cryptocurrencies in anti-money laundering. I need to study and analyze the existing platform, to have a deep understanding of its functional implementation and design process, provide references for the development.

Objective 2: I need to improve my understanding and use of blockchain wallets, including money transfers and on-chain transaction processes. This can give me an idea of the security issues that may occur when users use blockchain wallets for transactions.

Objective 3: I need to find and collect public databases of blacklisted addresses and websites, use APIs (application programming interface) to integrate data, provide users with secure queries before making transactions. It will solve the problem of wallets not performing security checks on transaction addresses.

Objective 4: The acquisition of on-chain transaction data is the prerequisite for transaction tracking and visualization. I need to get that type of data using the blockchain explorer's resources.

Objective 5: To study the methods for security scanning of smart contracts, I will need to learn how to use static analyzer to scan smart contracts to ensure the security of users when interacting with smart contracts and provide a report for the developer.

Objective 6: I will use Python and Flask as the basis for platform development. In addition, I will design a user-friendly GUI for the platform and evaluate all platform functions to enhance user experience after the functions are implemented.

1.4 Project Plan

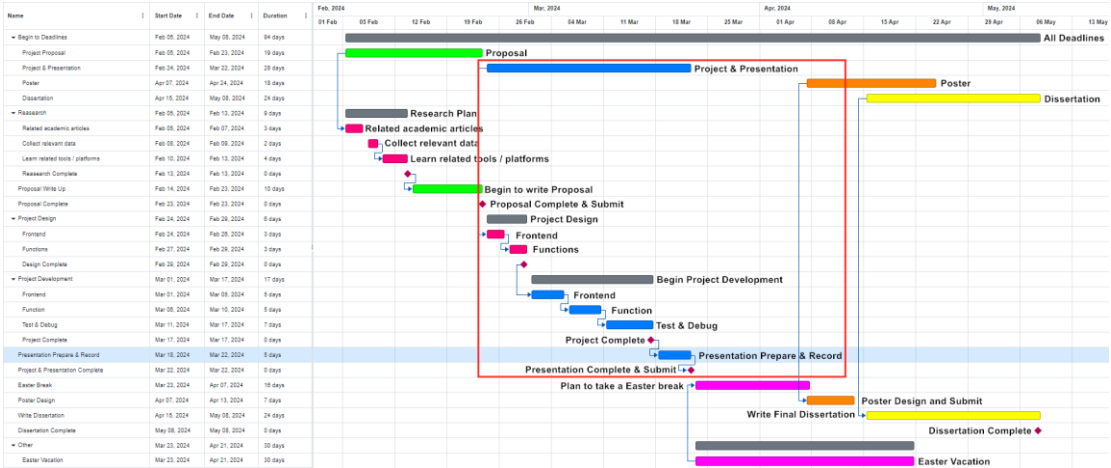


Figure 1.4.1: Project plan in Gantt chart.

Here is my whole project plan in Figure 1.4.1, the development process is inside the red box. The project plan includes the dates and durations, I classify the plan according to four deadlines, I can meet the project requirements according to different deadlines. I used the same color boxes as the deadlines boxes to ensure that my plan had a clear connection to my project goals and complete all deadlines on time.

1.5 Ethical Considerations

The blockchain wallet address is public, any wallet address or smart contract address can be viewed on blockchain explorer, wallet addresses are not tied to a specific identity, so they do not reveal who owns the wallet. I will give a notice in my platform and will not save the address when user provide their address. My project will not have any interact with users' wallet and collect any user data to avoid ethics and financial security problem, and all data in my project comes from the blockchain public data and databases.

2. Technical Background

Technical background research is an important beginning of a project. In this chapter, I will provide a complete description and investigation of blockchain wallets, provide the results of the investigation of transaction security, discuss the security issues existing in current blockchain wallets, and use relevant academic articles to explain and illustrate the security issues. Then, I will discuss platforms of the same nature as my project, point out their advantages and disadvantages and improve them in my project. Finally, I will list the data resources and descriptions that I used in the project.

2.1 What is Blockchain Wallet?

Blockchain wallets are designed for storing and managing cryptocurrencies. They provide a secure, decentralized way to protect your crypto assets.

The core of the wallet is the private key and the public key. The private key is the password used to access and control the cryptocurrency and must be kept secret, while the public key is the address used to receive the cryptocurrency and can be shared publicly. Wallets allow users to interact with the blockchain network to send, receive, and check cryptocurrency transactions.

At the same time, to ensure the safety of users' funds, most wallets provide a mnemonic phrase (multiple different words) backup options so that users can restore their wallet if needed. Many wallets also support cross-platform use to provide greater flexibility and convenience.



Secret Backup Phrase

Your secret backup phrase makes it easy to back up and restore your account.

WARNING: Never disclose your backup phrase. Anyone with this phrase can take your Ether forever.

wool thought awful better jar
music slush give mechanic ginger
faculty portion

NEXT



Figure 2.1.1: Mnemonic phrase example.

Suratkar(2020) pointed out, if a user needs to perform any transaction on the blockchain, the user must sign the cryptocurrencies to their wallet address. When a transaction is required, the user can unlock the corresponding funds in the wallet through the wallet key for transactions. At the same time, an exchange of transaction data value is created on the blockchain. [3]

Blockchain wallet provide users with a secure and autonomous way to manage their crypto assets and are key to using blockchain. Everyone who intends to use the blockchain platform for any transaction must use a blockchain wallet.

2.1.1 How Blockchain Wallets Work

1. Key generation: a blockchain wallet generates a private key and a public key for the user. The private key is the password to access and control the cryptocurrency and must be kept strictly confidential. The public key is the address used to receive the cryptocurrency and can be shared publicly.
2. Transaction signature: when a user wants to send a transaction from a wallet, the transaction data is digitally signed with a private key. This signature proves that the initiator of the transaction is the wallet owner and prevents the transaction from being tampered with.
3. Broadcast transaction: the signed transaction will be broadcast to the blockchain network, waiting for miner nodes to pack and record in a new block. Once the transaction is confirmed and written to the blockchain, the funds are transferred from the sender to the receiver's wallet address.

4. **Wallet balance:** the wallet calculates the balance by scanning all transaction records related to the wallet address on the blockchain. If the public key address is known, the wallet can query and display the corresponding balance, but the public key address is safe to everyone knowing.
5. **Backup:** most wallets support backing up private keys in the form of mnemonic phrase that can be recovered if the wallet is damaged or lost. The corresponding private key and wallet can be regenerated when the mnemonic phrase is recovered.

In general, blockchain wallets send and receive cryptocurrencies on the blockchain, achieve autonomous control and management of funds through key generation, transaction signature and broadcast technologies.

2.1.2 Features of Blockchain Wallet

1. **Decentralization:** Instead of being controlled by a central authority or third party, blockchain wallets let users autonomously hold and manage their own cryptocurrency private keys. This leaves the money fully in the hands of the user, with no risk of a single point of failure or censorship.
2. **Anonymity:** most blockchain wallet addresses are just random alphanumeric combinations and are not directly linked to user identity. This provides users with a degree of transaction privacy and anonymity.
3. **Portability:** many software and mobile wallets support backup and recovery functions. If there is a mnemonic phrase, you can reconstruct and access the wallet anytime and anywhere, which is very portable.
4. **Cross-platform compatibility:** a good wallet tends to support multiple operating systems and devices, allowing users to easily manage their cryptocurrencies anywhere. Cross-platform usage brings great convenience.
5. **Multi-asset support:** many wallets not only support Bitcoin, but also are compatible with Ethereum, to facilitate users' asset management and transactions.

Blockchain wallets bring users many unique advantages. They become an important tool for managing crypto assets.

2.2 Blockchain Wallet Research

In the August of 2023, a CoinGecko author [4] provided a list about the most popular blockchain wallets, I would like to reference the top 10 blockchain wallets here, to conduct subsequent analysis.

Ranking	Blockchain Wallet	Total Installations & Users
1	MetaMask	22.66 million
2	Coinbase Wallet	11.00 million
3	Trust Wallet	10.40 million
4	Blockchain.com Wallet	10.00 million
5	Bitcoin.com Wallet	5.00 million
6	Phantom	2.54 million
7	Bitget Wallet	1.10 million
8	Crypto.com DeFi Wallet	1.07 million
9	Exodus	1.07 million
10	SafePal	1.06 million

Table 2.2.1: Top 10 popular blockchain wallets in 2023. [4]

The ranking data for this list is derived from the total number of installations of different blockchain wallets in browsers and mobile phones. Based on the Table 2.2.1, MetaMask is the most popular blockchain wallet, with more than 22.66 million app installations and users. Meanwhile, the top three most popular blockchain wallets have 44.06 million installations. Therefore, I will select these three most popular blockchain wallets for usability and security analysis and discuss whether users need to use security scanning functions before making transactions.

2.2.1 Usability Analysis in Different Blockchain Wallets

In chapter 2.2, I chose MetaMask, Coinbase Wallet and Trust Wallet for usability analysis. They vary in usability, depending on the tradeoff between security, portability, and complexity of use. I will show the usability of these three wallets through tables and different angles to explain them.

Blockchain Wallet	Type of Wallet	Bitcoin and Ethereum support	Hardware compatible	Customer Service
MetaMask	Software (browser extension and mobile)	Only Ethereum	Yes	No live customer support
Coinbase Wallet	Software (browser extension and mobile)	Yes	No	24/7 customer support
Trust Wallet	Software (browser extension and mobile)	Yes	Yes	No live customer support

Table 2.2.1.1: Usability Results table for three blockchain wallets

1. MetaMask

MetaMask is a decentralized blockchain wallet that prioritizes the security and anonymity of its users. It has a simple setup process and support for various Ethereum-based cryptocurrencies, providing versatility and accessibility. The wallet's compatibility with multiple platforms and integration with popular dApps (decentralized applications) make it a convenient choice for cryptocurrency enthusiasts.

However, the lack of real-time customer support and lack of bitcoin support can be drawbacks for some users. Nonetheless, MetaMask offers users secure and efficient Ethereum-based transactions, making it become a valuable blockchain wallet in the cryptocurrency space.

Advantages	Disadvantages
Simple setup process	Poor customer support
Supports a wide range of Ethereum-based cryptocurrencies	
Wide support for dApps	

Table 2.2.1.2: Pros and cons in MetaMask wallet

2. Coinbase Wallet

Coinbase Wallet is a popular unmanaged blockchain wallet that allows users to securely store their digital assets. Its mnemonic system ensures maximum security. Coinbase also allows user to back up mnemonics to Google Drive, which is convenient for users who might forget it.

This blockchain wallet is perfect for anyone who values security and likes to explore blockchain and will also find Coinbase helpful for users who like to switch between simple and professional modes.

Advantages	Disadvantages
Simple setup process	Limited support in NFTs
Simplified application layout	
Supports many cryptocurrencies	

Table 2.2.1.3: Pros and cons in Coinbase wallet

3. Trust Wallet

Trust Wallet has a simple user interface and support for multiple blockchain networks and crypto assets. Users can trade and store cryptocurrencies through a single interface, interacting with dApps. These features enable Trust Wallet to meet all blockchain user needs.

However, since the wallet contains multiple third-party exchanges, the fees for purchases can be confusing for new users. These providers charge different fees for the purchase of cryptocurrencies. Users need to research the purchase individually and determine the exact cost.

Advantages	Disadvantages
Simple setup process	Only support online software
Non-custodial	Poor customer support
In-app decentralized exchange	Private Android source code

Table 2.2.1.4: Pros and cons in Trust wallet

2.2.2 Security Problems in Different Blockchain Wallets

After analyzing the usability of three different blockchain wallets, I also analyzed their security problem separately. The main content of the security problems analysis is whether the three different blockchain wallets will perform security detection on the transaction address, such as identifying whether the address or website is risk such as fraud or phishing attacks. Another important security analysis is whether these three different blockchain wallets will perform security checks on smart contracts when users authorize blockchain wallets to interact with smart contracts. I will provide the results of the security analysis table and explain the three different blockchain wallets respectively.

Blockchain Wallet	Risk Address Detection Function	Smart Contract Detection	Risk Website Detection Function
MetaMask	Optional	Optional	No
Coinbase Wallet	Yes	Yes	No
Trust Wallet	Yes	No	No

Table 2.2.2.1: Security functions result in different wallet.

The results of the table above are derived from my investigation of the usage and functions of three different blockchain wallets. The three different wallets do check the risk addresses that have been collected, but they cannot detect risk website and Trust Wallet do not have smart contract detect function. MetaMask is optional for the first two functions, if the user does not turn on these two functions, they will not be able to use security functions.

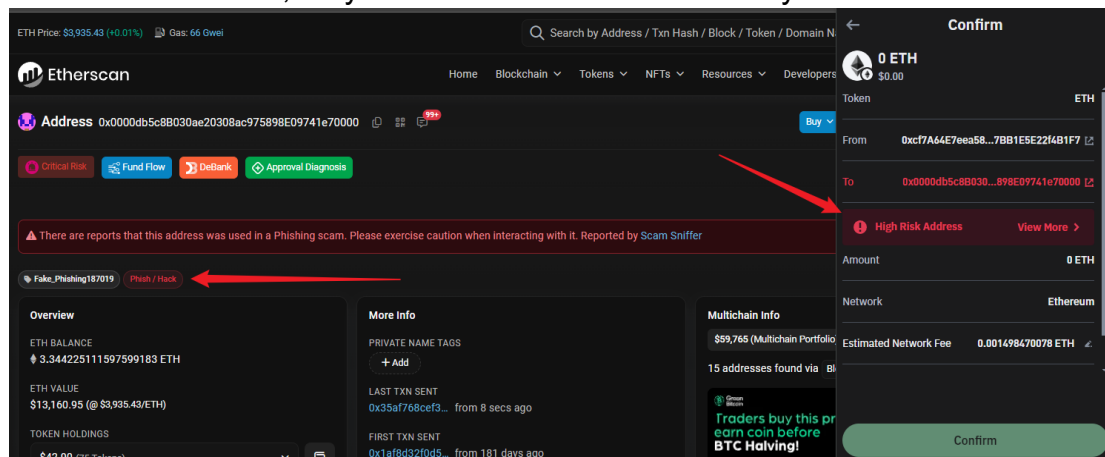


Figure 2.2.2.2: Officially marked addresses have notice on Trust Wallet

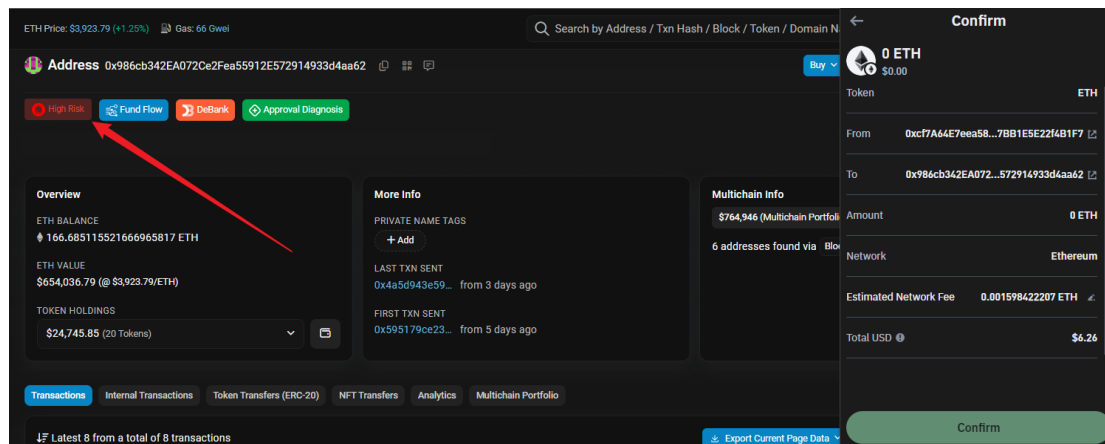


Figure 2.2.2.3: High-risk address does not have notice on Trust Wallet

In the two figures above, I show the state of Trust Wallet's transactions on risk addresses in two different situations.

In Figure 2.2.2.2, at the position of the red arrow on the left, Ethereum's official blockchain browser indicates that this wallet address is a phishing risk. On the right, I try to make a transaction using Trust Wallet, which displays a reminder of high-risk addresses. However, in Figure 2.2.2.3, the location of the red arrow on the left, the third-party blockchain security team alerted this address as a risk address, but Ethereum's official blockchain browser did not provide any security tips. As a result, when I made a transaction in the Trust Wallet on the right, it did not have any notice and the user could continue to make a transaction.

This means that the blockchain wallet has the problem of information delay for the function of security detection. This issue is not only appeared on the Trust Wallet, but also several other different blockchain wallets. The reason for this problem is that after the blockchain security team has identified the risk address and submitted it to the Ethereum official, it still takes a long time to publish the risk address information to different blockchain wallets or databases.

At the same time, based on my technical research and use, when users use blockchain wallets to authorize smart contracts, Coinbase Wallet has a complete detection system [5], MetaMask uses data provided by a third-party blockchain security organization, it provides an option to user that if they need security function [6], but the Trust Wallet does not have the risk detecting capabilities of smart contract security.

Scam Sniffer, a blockchain cybersecurity team, has published a report on cryptocurrency phishing attacks in 2023. In the report, the team noted that the attackers stole cryptocurrency assets from users blockchain wallets by deploying malicious smart contracts and phishing websites to trick users into authorizing blockchain wallets and signing malicious transactions. [10]

This situation shows that the functions of blacklist address, website detection and smart contract security scanning I designed in the project are important.

2.3 Relative Platform Research

I have chosen MetaSleuth (<https://metasleuth.io/>) as the relative platform to investigate. MetaSleuth is a blockchain tracking and investigation platform that helps trace the flow of cryptocurrencies from criminal activities. The platform has many functions, such as transaction tracking, address marking, and filtering capabilities.

Next, I will experience the main functions of this platform and give my comments.

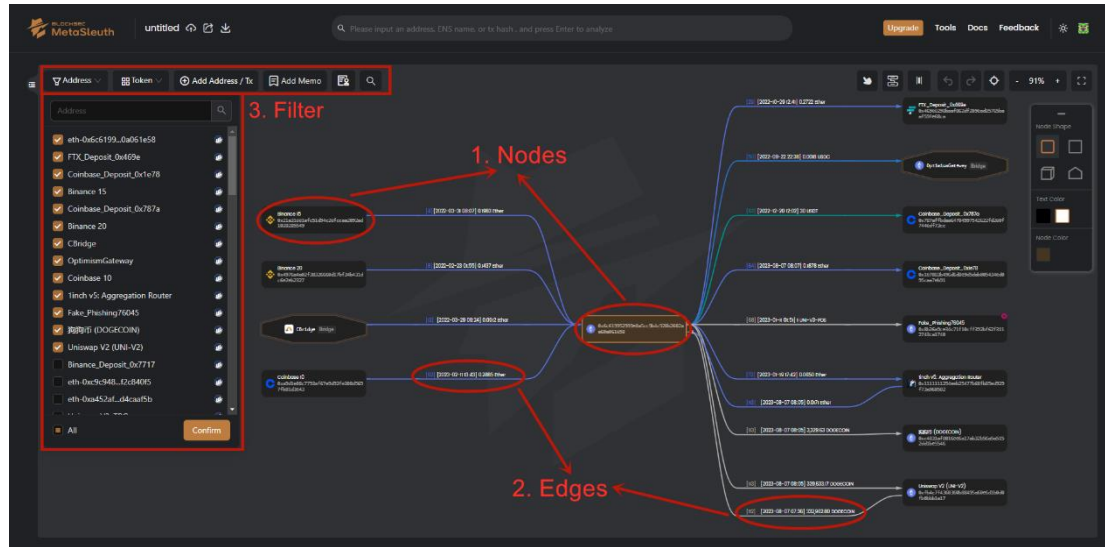


Figure 2.3.1: A functional demonstration of the MetaSleuth platform

In Figure 2.3.1, the main function of this platform is to visualize the transaction flow graph of the blockchain wallet address and display it in the canvas. I have marked three main functions: nodes, edges and filter. I will introduce and analyze these three functions respectively.

1. Nodes:

A blockchain wallet address node is a real address on the blockchain, uniquely identified by a combination of the blockchain and the address itself. Users can view various information about the address node, such as address, name label, etc.

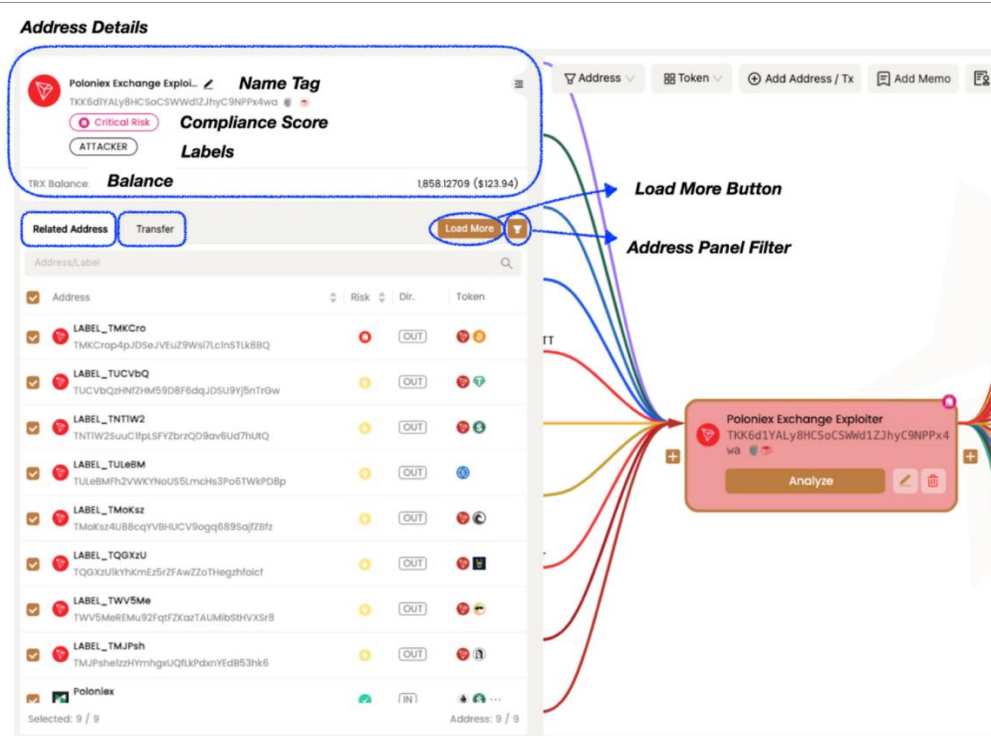


Figure 2.3.2: Node in the MetaSleuth platform

After clicking on the node of the blockchain wallet address, the details panel of the address expands from the left. In Figure 2.3.2, I marked the information content of the blockchain wallet address provided by the platform to the user.

However, this function has usability problems, and the information it presents to the user is not intuitive enough. Users need to manually click to see the detailed content, and no direct feedback is given after the query. For example, what the user queries is a risky blockchain wallet address, but the annotations in the platform are too small to attract the user's attention.

2. Edges:

When you click any edge on the platform, the Edge list panel expands. In this panel, you can view all asset transfers between the two addresses (based on currently available data). Each edge is uniquely identified by (from, to and asset).

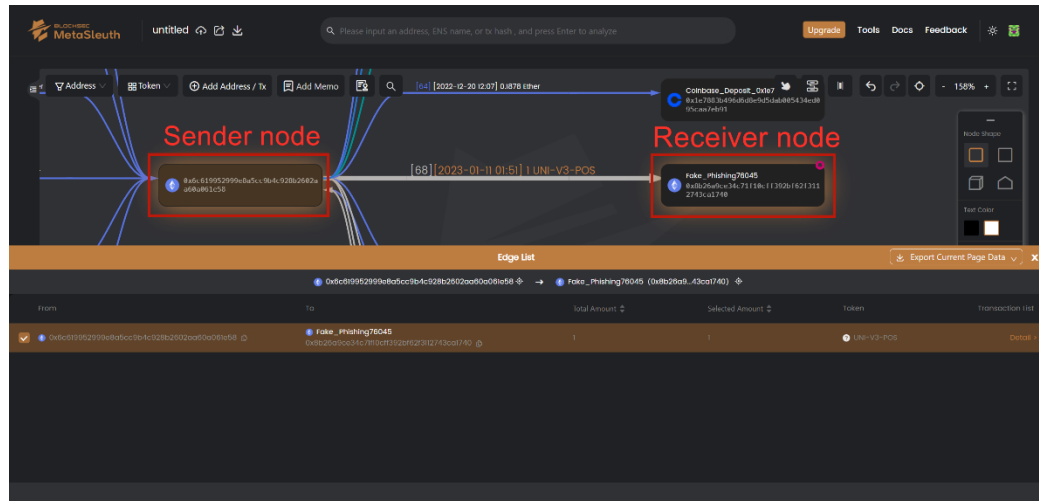


Figure 2.3.3: Edge in the MetaSleuth platform

I chose one of the edges to view. Each edge corresponds to the transaction flow of the wallet address. As you can see in Figure 2.3.3, this transaction has been sent to the recipient's blockchain wallet address. However, note the label given in the image, the recipient address indicates the risk of a phishing attack. This means that there are no special colors or markers used in the platform to alert the user that the transaction he is sending is a security risk.

3. Filter

This function mainly provides filtering and editing for the data queried by the user, user can filter whether the specified blockchain wallet address or the cryptocurrency of the transaction is displayed in the canvas.

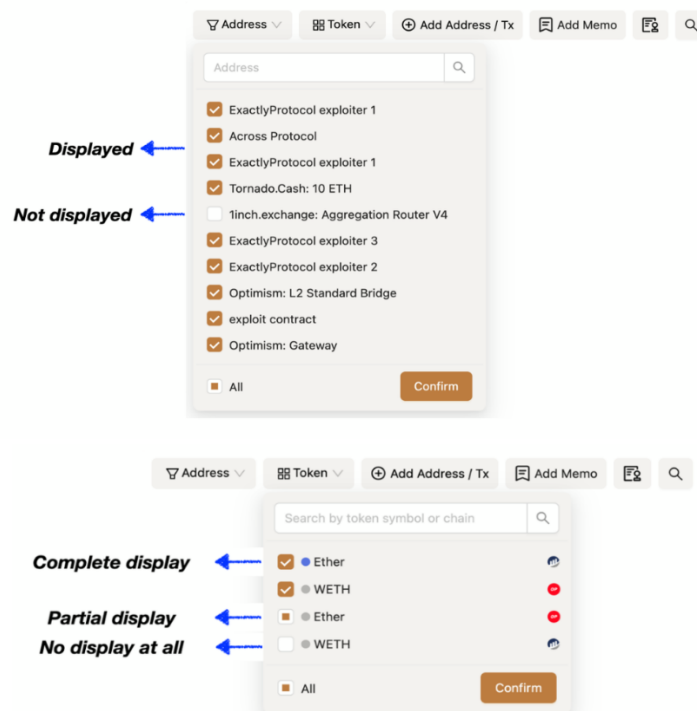


Figure 2.3.4: Filter function in MetaSleuth platform

This is a great function, especially when it comes to anti-money laundering tracking. However, this is of little use to the average user, who generally only wants to get more intuitive data.

In summary, MetaSleuth is a powerful platform. Its functions serve users of blockchain or blockchain security experts, especially in the part of blockchain transaction visualization and tracking. However, MetaSleuth is too complex to use for the average user, and much of the detailed security information requires a manual click. As a result, ordinary users or those transacting on the blockchain for the first time are unable to intuitively access useful security information. This also proves that the results provided to users in my project need to be more intuitive and concise.

2.4 Data Resources Descriptions / Solutions

The use of data is an important part of my project to provide users with the most accurate results to compensate for the missing features and issues of blockchain wallets. At the same time, to avoid ethical issues, I use publicly available data sources in my projects.

Below, I will list the data sources I collected during the technical background research phase and the data sources I used in the project. The use of the data will be categorized according to the functionality of the project.

1. Transaction history query and visualization function:

Etherscan: It is an Ethereum blockchain explorer, search, APIs and analytics platform for Ethereum. As a public way to access blockchain data, they provide a free API interface that allows users to directly access Ethereum blockchain data by request. [7] I will use “Accounts” API endpoint to get a list of transactions by different blockchain wallet addresses.

```
https://api.etherscan.io/api
?module=account
&action=txlist
&address={blockchain_wallet_address}
&startblock=0
&endblock=99999999
&page=1
&offset=10
&sort=desc
&apikey=etherscan_api_key
```

Table 2.4.1: API endpoint for transactions query

2. Blacklist address and website detection function:

GoPlus: GoPlus provides a free, real-time blockchain security inspection APIs, all sources of information are public. [8] I'm going to use the Malicious Address and Phishing Website Detection API in this function.

```
https://api.gopluslabs.io/api/v1/address_security/
{blockchain_wallet_address}
?chain_id=1
```

Table 2.4.2: API endpoint for malicious address query

```
https://api.gopluslabs.io/api/v1/phishing_site
?url={website_URL}
```

Table 2.4.3: API endpoint for phishing website query

3. Smart contract security scanning function:

GoPlus: GoPlus also provides security queries for smart contracts, which can provide functional detection results of smart contracts and the user's blockchain wallet authorization function queries. I will be using this API as a simple smart contract security scanning function.

<pre>https://api.gopluslabs.io/api/v1/token_security/1 ?contract_addresses={smart_contract_address}</pre>

Table 2.4.4: API endpoint for smart contract scanning

Slither: Slither is a static analyzer written in Python. It runs a suite of vulnerability detectors that print visual information about the details of the smart contract. Slither enables developers to discover vulnerabilities, enhance their code understanding, and provide developers with assistance in smart contract development. [9] I will combine this static analysis tool as a deep detection of smart contract security and use it in the smart contract security scan function.

2.5 Summary

In this chapter, I conduct technical background research on blockchain wallets. Firstly, I introduce the concept, working principle, characteristics and differences between blockchain wallet and traditional wallet. I analyzed the top 10 most popular blockchain wallets in 2023, focusing on MetaMask, Coinbase Wallet, and Trust Wallet for usability and security analysis, pointing out their problems with address, website and smart contract security detection.

Then, I introduced a blockchain transaction tracking platform as an evaluation object and commented on the advantages and disadvantages of the platform, arguing that it is powerful but too complex for the user and cannot provide more intuitive information.

Finally, the data resources that will be used in this project are listed.

This part makes a comprehensive technical investigation and analysis for the security of the blockchain wallet, determines the data resources and solutions that need to be used in the project, laying the foundation for the subsequent system design and implementation.

3. What was done, and how?

This part describes in detail the process and introduction that goes into developing a system. I will list the development process and development requirements, provide system architecture and component design diagrams and descriptions. At the same time, I will show the user interface design, discuss the implementation of the system. I will also describe the methodology used for development, the tools and techniques used, and the process of how to translate a complete system design into implementation.

3.1 Development Process

In this project, I will divide the system development into five stages:

1. **System architecture design:** This phase focuses on determining the overall structure of the system and the relationships between its components. I will plan each module of the system in detail and the way they interact with each other to ensure that the system has good scalability and maintainability.
2. **User interface design:** I will focus on designing the user interface to ensure that users can easily and intuitively interact with the system. I will consider user experience and user friendliness, design an interface that is easy to understand and operate.
3. **Front-end static development:** Once the user interface design is finalized, I will start the front-end static development phase. At this stage, I will turn the designed user interface into a static page, using techniques such as HTML and CSS to implement the structure, style, and interaction of the page.
4. **Back-end development and connectivity:** In this phase, I will work on developing the back-end part of the system and making sure that the front-end is connected to the back-end. I will implement the system functional logic, handle data interaction and display, ensure integration with other system functions.
5. **Testing and debugging:** I will plan the test method and conduct a comprehensive test of the system. At the same time, I will debug and fix any issues found until the system meets the expected quality standards.

By dividing system development into well-defined phases, I was able to effectively manage project progress and ensure that all aspects of development work were fully considered and implemented. In addition, this approach makes it easier to identify and solve problems, it can be adjusted and improved in time during the development process.

3.2 Development Requirements

To achieve the remaining project objectives, the following requirements will detail the requirements in development. They are divided into different parts, related to the function of the system.

3.2.1 Front-end Development

1. Need to follow the GUI design to develop.
2. Suitable for all device display sizes.
3. The data returned by the back-end must be displayed properly.

3.2.2 Back-end Function Development

1. API requests and returned data must be handled correctly.
2. Use visualization libraries to transform data into visual charts.
3. Use Slither in combination and process the returned results.
4. The data can be transfer correctly to the front-end.

3.2.3 Testing and Debugging

1. User interface usability testing.
2. Unit tests on each function.
3. Front-end and back-end communication testing and debugging.

3.3 Main Tools and Technologies

1. HTML

HTML is a markup language used to create the structure of web pages, which consists of a series of tags, each describing a different content or function on a web page. [11] HTML uses these tags to define elements such as text, images, links, tables, and so on, and to determine how they appear in the browser. The HTML language structure is very simple, it is composed of a pair of Angle brackets surrounded by tags, tags are usually divided into a start tag and an end tag, some are closed.

As the basic language of front-end development, HTML has the advantages of simple and easy to learn, cross-platform compatibility, semantics, and good cooperation with other technologies. I can use it to provide basic support for building web pages.

2. CSS

CSS (Cascading Style Sheets) is a style sheet language used to describe the style and layout of web pages, which is used in conjunction with HTML to control the appearance and typography of web page elements. [12] CSS uses selectors and attributes to select page elements and define their styles, making web pages more beautiful, readable, and easy to use.

I combine CSS with HTML for front-end development. As an integral part of front-end development, CSS provides powerful style control and layout capabilities for creating great web pages.

3. Bulma Framework

Bulma is a CSS framework based on the Flexbox layout for quickly building responsive, mobile-first websites and Web applications. [13] The framework provides a series of predefined CSS classes, provides powerful styling components and layout tools for front-end development. Through a simple HTML structure and class name, I can achieve a variety of complex layout and style effects, which helps me build high-quality platforms quickly in front-end development.

4. Python

Python is a high-level general-purpose interpreted programming language. Python has a robust standard library and a rich ecosystem of third-party libraries that cover development needs in various fields such as Web development, data analytics, artificial intelligence, and more. I chose to use Python for my project development because Python has a variety of good Web frameworks, I can use Python in combination with Flask frameworks. These frameworks provide complete development tools and components to improve development efficiency.

Python can also be used on multiple operating systems, allowing developers to develop and run on different platforms.

5. Flask Framework

Flask is a lightweight Python Web framework, it is simple and flexible, suitable for rapid development of Web applications. Flask provides a simple code structure and rich extension features that allow developers the flexibility to customize and extend Web applications. [14] I chose to use Flask framework in conjunction with Python for development because Flask framework is highly integrated with Python and can be easily integrated with Python standard libraries and third-party libraries. At the same time, Flask uses the Jinja2 template engine to separate front-end logic from back-end logic, which can be used in HTML to improve reusability and flexibility. [14]

6. Slither

Slither is a smart contract static analysis tool specifically designed for static analysis and vulnerability detection of Ethereum smart contracts. [9] It can detect various security vulnerabilities and potential problems in contracts. It can provide detailed reports and suggestions to help developers find and repair security problems in contracts, improve the security and reliability of smart contracts. Slither can also provide enough security references and suggestions for ordinary users.

In my project, I will use this tool as one of the functions of smart contract security detection and output the results of the tool in the front page.

7. Pyecharts

Pyecharts is a Python visualization library based on Echarts, which can quickly create various interactive charts. Pyecharts has rich chart types and flexible configuration options, and I was able to easily implement my data visualization needs. [15] At the same time, Pyecharts supports seamless integration with common tools in the Python environment such as Jupyter Notebook, Flask, Django, etc., providing powerful visualization solutions for data analysis, web application development and other fields.

8. Markdown

Markdown is a markup language designed to make documents easy to read and write. Using Markdown, I can markup text using simple markup syntax, without having to learn complex HTML tags. Markdown documents can be easily converted to HTML and other formats, it can be edited and read on a variety of editors and platforms.

9. Jinja2

Jinja2 is a popular Python based templating engine specifically designed to combine data with templates to generate text output. It uses a simple and flexible syntax that makes it easy for developers to create pages with dynamic content. Jinja2 allows developers to write modular templates, improving code reuse and maintainability.

3.4 System Architecture Design

System architecture design plays a crucial role in project development. System architecture design ensures the integrity and reliability of the website platform by defining the components, structure, interaction and data flow of the system. In this chapter, I will show the system architecture diagram and the system component diagram that I designed, describing them respectively.

3.4.1 System Architecture Diagram and Description

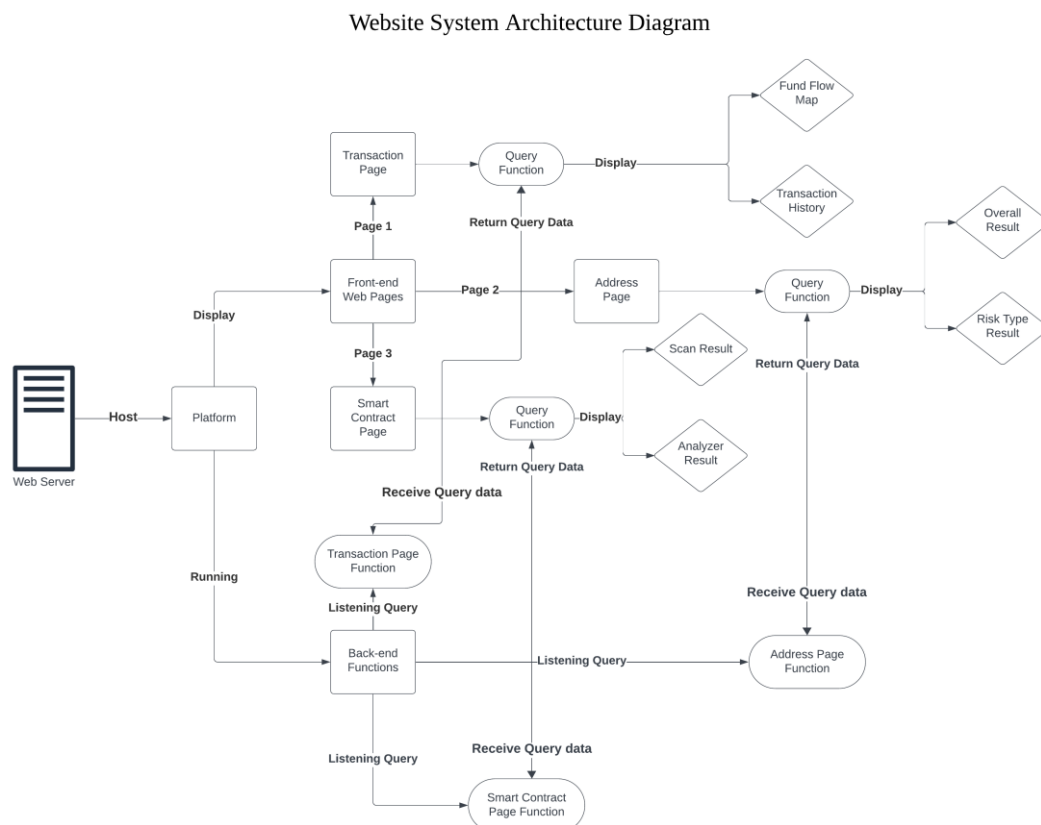


Figure 3.4.1.1: Website system architecture diagram

Figure 3.4.1.1 is the system architecture diagram I made. The platform I developed in the project runs on a web server. As shown in the figure, the platform will be divided into two parts: displaying front-end website pages and running the functions performed by the back-end. In the front-end website page, I will make three different pages corresponding to different needs, they also correspond to the three functions of the back-end operation.

The first page is the *Transaction* page, it contains a query function. After the user queries the data blockchain wallet address, the front-end page transfers the form data to the back-end *Transaction Page* function. This function will process the request and return the data to the front-end page. The page will display the transaction history and security, as well as a visual output of the

transaction flow.

The second page is the *Address* page, it contains a query function for the risk blockchain wallet address or website. The front-end page transfers the table data to the back-end *Address Page* function, it will process the request and returns the data to the front-end, which will display the security result of the risky blockchain wallet address or website and the judgment of the risk type respectively.

The third page is the *Smart Contract* page, it provides the user to enter the address of the smart contract for security detection and scanning of the smart contract. The front-end page transmits the form data to the back-end *Smart Contract Page* function, it will process the request and return the data to the front-end page, which will display the results of different risk types of smart contracts and the number of vulnerabilities, provide reports to users and developers.

3.4.2 Components Diagram and Description

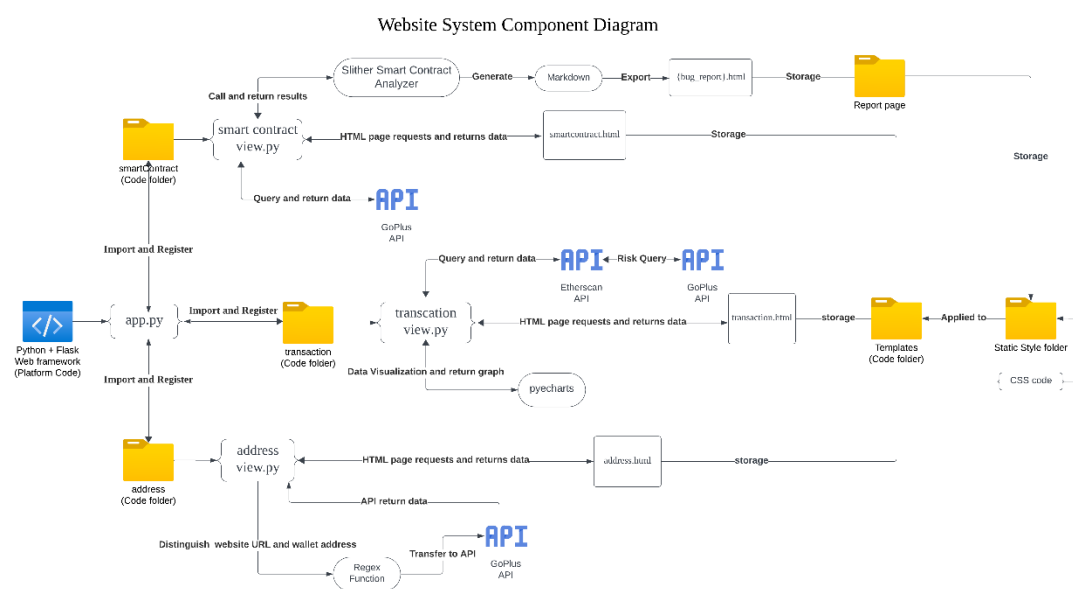


Figure 3.4.2.1: Website system back-end functions component diagram

Figure 3.4.2.1 is the component diagram of the back-end function of the website system, it shows the invocation process of the component in the function and reflects to the front-end page.

Based on diagram, the functional components are mainly implemented using Python, Flask frameworks and divided into three different functions.

The first function is applied to *view.py* in the *Transaction* code folder. This component will receive data from the front-end *Transaction* page, implement Etherscan and GoPlus API calls within the component to query transaction history and transaction risk data. Using the transaction history data returned by Etherscan, the component visualizes the transaction flow using the Pyecharts

visualization library. Finally, the data is packaged and sent back to the front-end Transaction page.

The second function works on *view.py* in the *Address* code folder. This component will receive user query data from the front-end *Address* page, first distinguish whether the user query data is a blockchain wallet address or a website through the regex function, and then use the API of GoPlus within the component to query the data of the risk blockchain wallet address and website. Finally, the data is packaged and sent back to the front-end *Address* page.

The third function is *view.py* that works in the *SmartContract* code folder. This functional component is divided into two parts: the first part receives the user query data from the front-end *SmartContract* page, queries the security information of the smart contract through the GoPlus API and returns the result. The second part is to transfer the smart contract address queried by users to Slither for vulnerability scanning of smart contracts, return the overall result of scanning to the component, then generate HTML static pages in Markdown format to provide users and developers with scanning reports. At last, data will display in the front-end *Smart Contract* page.

All pages will be stored in the *Templates* folder and styled using CSS code.

3.5 System User Interface Design

System user interface design is not only the bridge between users and the system, but also the core of user experience. Through well-designed user pages, an intuitive and friendly interface can be provided. At the same time, user page design is also an important tool for communicating with users during the project development process, to maintain a close fit between the product and the needs of users and projects. I will show the user interface design drawings of three different pages in the project and provide explanations.

3.5.1 Transaction Page Design and Description

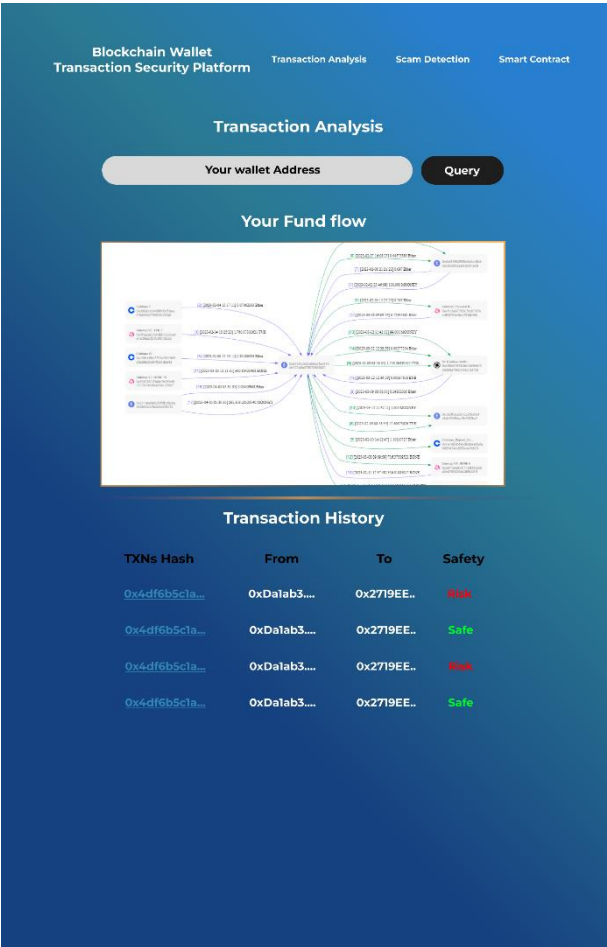


Figure 3.5.1.1: Transaction Page GUI Diagram

Figure 3.5.1.1 shows the GUI design of the Transaction front-end page. The “Transaction Analysis” section has an input field for entering the wallet address and a “Query” button. Below the input area, there is a visualization called “Your Fund Flow”, which shows a visualization of incoming and outgoing transactions with blockchain wallet addresses and transaction hashes, used to display historical transaction flow information for wallet addresses. The “Transaction History” section lists recent transaction information, including

details such as the hash value of each transaction for the blockchain wallet address queried by the user, the sender (“From”) address, the recipient (“To”) address, and the “Safety” status indicating whether each transaction is safe or risky. If the Risk transaction is displayed “Risk”, the Safe transaction is displayed “Safe”.

This page is designed to provide users with an overview of blockchain wallet transactions, fund flows and security analytics through visual and table data.

3.5.2 Address Page Design and Description

The image shows a web interface for a 'Blockchain Wallet Transaction Security Platform'. It features a navigation bar with links to 'Transaction Analysis', 'Scam Detection', and 'Smart Contract'. The main heading is 'Scam Address and Website Detection'. Below this is an input field labeled 'Input any wallet address or URL to check' and a 'Check' button. The result status is 'It may be Safe / Dangerous', with 'Data Source From: GoPlus / SlowMist'. A section titled 'Because...' contains a table of risk types and their results.

Risk Type	Result
Cybercrime	True / False
Money Laundering	True / False
Financial Crime	True / False
Darkweb TXN	True / False
Phishing Activity	True / False
Fake KYC	True / False
Stealing Attack	True / False
Blackmail Activity	True / False
Malicious Mining	True / False
Phishing Website	True / False

Figure 3.5.2.1: Address Page GUI Diagram

Figure 3.5.2.1 is the GUI design of the Address front-end page. The main function of this page is to detect whether the entered wallet address or website URL is a potential security risk.

There is an input box in the center where the user can enter any wallet address or website URL that needs to be checked, then click the “Check” button to trigger the risk detection.

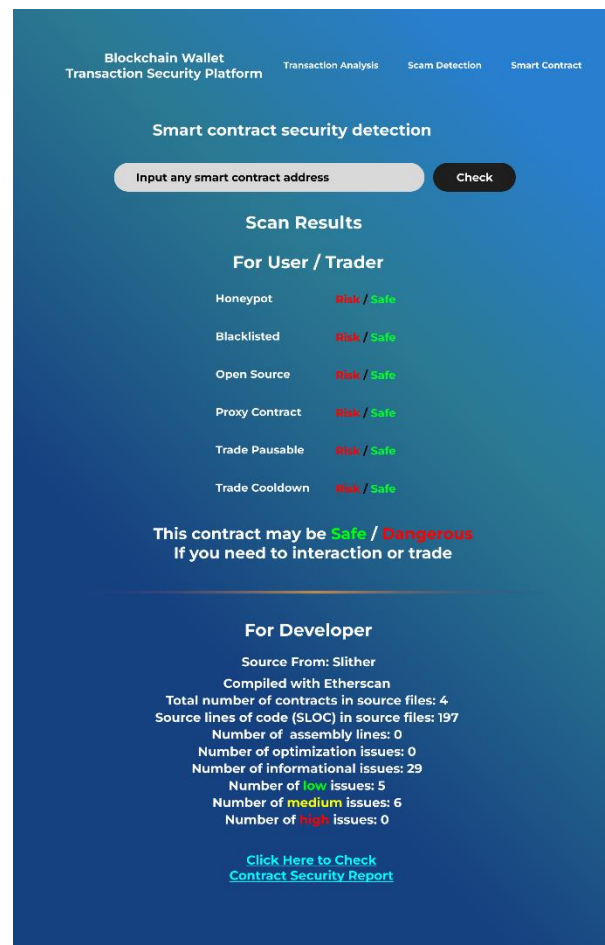
The total test results are presented in a “Safe” or “Dangerous” format, combined with analysis and risk assessment based on multiple data source.

Detailed risk types are listed below, each risk type gives a “True/False” result

indicating whether the risk exists. At the same time, the “Phishing Website” risk type will only be displayed when users query the website URL.

This page is designed with the objective of providing a security detection service to help users find and stay away from potential blockchain fraud risk, thereby protecting the security of funds and crypto assets. At the same time, by listing common types of scams, it also plays a role in warning.

3.5.3 Smart Contract Page Design and Description



The image shows a web interface for a Blockchain Wallet Transaction Security Platform. The top navigation bar includes links for Transaction Analysis, Scam Detection, and Smart Contract. The main heading is "Smart contract security detection". Below this is an input field labeled "Input any smart contract address" and a "Check" button. The results are divided into two sections: "For User / Trader" and "For Developer".

Smart contract security detection

Input any smart contract address

Scan Results

For User / Trader

Honeypot	Risk / Safe
Blacklisted	Risk / Safe
Open Source	Risk / Safe
Proxy Contract	Risk / Safe
Trade Pausable	Risk / Safe
Trade Cooldown	Risk / Safe

This contract may be Safe / Dangerous
If you need to interaction or trade

For Developer

Source From: Slither
Compiled with Etherscan
Total number of contracts in source files: 4
Source lines of code (SLOC) in source files: 197
Number of assembly lines: 0
Number of optimization issues: 0
Number of informational issues: 29
Number of low issues: 5
Number of medium issues: 6
Number of high issues: 0

[Click Here to Check Contract Security Report](#)

Figure 3.5.3.1: Smart Contract Page GUI Diagram

In the Figure 3.5.3.1, It shows the “Smart contract” front-end page of the platform, which is specifically designed to detect and evaluate the security of smart contracts.

At the top of the page there is an input box where the user can enter the address of the smart contract to be detected and then click the “Check” button to trigger a security scan. The scan results are divided into two parts: for users/traders and for developers.

The user/trader section can check whether the smart contract has honeypot traps, is blacklisted, is open source, is an agent contract, whether the trade can be suspended, and whether there is a cooldown period for the trade, each risk

type show with “Risk” or “Safe” status. Finally, the comprehensive security assessment results of the contract will be given.

In the developer section, detailed information about the source code of the contract is provided, including result sources, compilers used, number of lines of source code, etc. In addition, the number of issues found in the contract with different severity levels, such as the number of low/medium/high risk issues, as well as the number of other optimization and informational issues are listed. A link is provided at the end where developers can click to view the full contract security report.

The page provides users and developers with a comprehensive smart contract security assessment, which helps to discover and repair security risks, thus guaranteeing the code security of the smart contract and the security authorization of the blockchain wallet.

3.5.4 Web Page Components Description

In the GUI interface I designed, I used some common page components:

1. **Top navigation bar:** This is the main navigation bar for the entire platform and consists of three tabs : “Transaction Analysis”, “Fraud Detection” and “Smart Contracts”. Users can switch between pages by clicking on different tabs.
2. **Input field:** Each module page has a gray input box for the user to enter the content that needs to be checked, such as wallet address, URL, smart contract address, etc. The text prompt in the input box explains the type of content that should be entered.
3. **Button:** On the same line as the input box is a “Query” or “Check” button. After entering the content in the input box, the user needs to click the button to trigger the corresponding detection or analysis function.

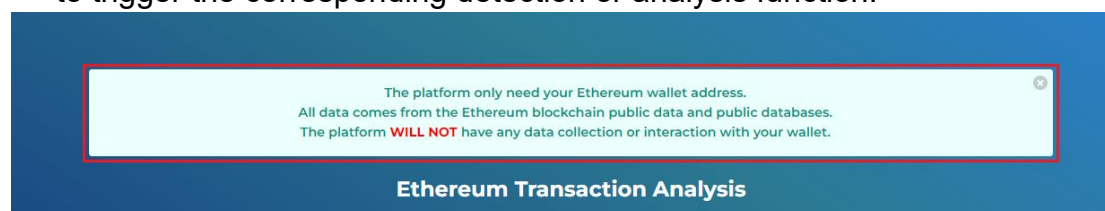


Figure 3.5.4.1: Notice box about ethical considerations

4. **Notice box:** In Figure 3.5.4.1, I added a notice box at the top of each page. The purpose of this function is to inform users that the platform only needs their blockchain wallet address and will not collect any other information. This is to satisfy the need for ethical considerations.

3.6 Front-end, Back-end Development

In system development, front-end and back-end development play different roles. Front-end development is responsible for building the interfaces that users interact with directly, including user interface design and implementation for web and mobile applications. Back-end development is responsible for the core logic and data processing of the system to support the functionality and data interaction required for the front-end interface. In this chapter, I will detail the development process of the project and provide corresponding diagrams.

3.6.1 Master Files in Front-end and Back-end

- **Front-end:** For the master file on the front-end, I use *base.html*, which is stored in the *Templates* folder. This HTML file will serve as the overall front-end framework of the site, it includes the CSS style of the Bulma framework and the style file written by myself, which contains all the web components in the user interface design. Based on Figure 3.6.1.1, the code includes the *Block Content* function of the Jinja2 template engine, bringing in templates for the other three front pages.

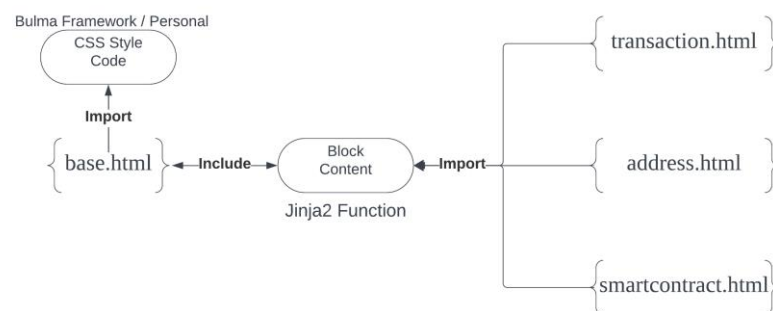


Figure 3.6.1.1: Front-end master file code logic

- **Back-end:** For the master back-end file, I use *app.py*, which is a Python file that is also required to run the Flask framework. In the code for this file, I use the *Transaction* page as the main page, and I use the *Blueprint* function to import the back-end python file for the other three pages, the code process is shown in Figure 3.4.2.1.

After introducing the development and process of the main file, I will introduce the front-end and back-end development content of the three pages respectively. Since the development of the main file has determined the general project framework, I will introduce the components and data presentation process of the front-end page in the following content, and the back-end functions will be explained using UML diagrams combined with descriptions.

3.6.2 Transaction Page and Functions Development

3.6.2.1 Front-end Development

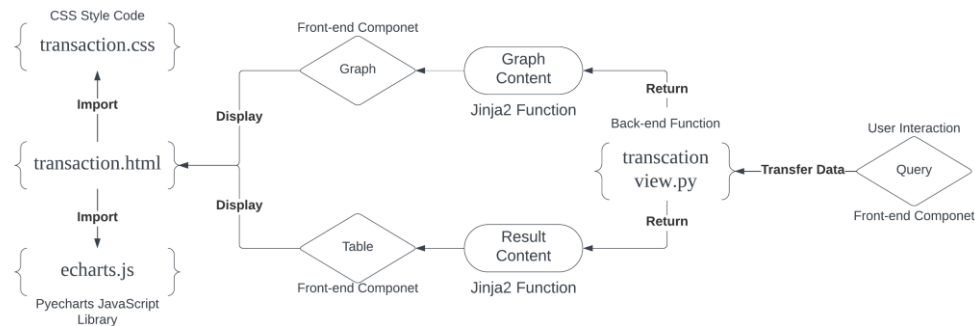


Figure 3.6.2.1.1: Transaction page front-end code logic

Figure 3.6.2.1.1 shows the front-end code logic of the *Transaction* page. When user makes a query, the front-end input field will transmit the data entered by the user to the back-end function for processing. The back-end function will return two different data: the first one is the transaction flow graph, through the Jinja2 custom *Graph* function, the function will determine whether to accept the data from the back-end and display the result to the corresponding HTML front-end page, it will show a graph generate by the Pyecharts in the web page. The second is the transaction history, through the Jinja2 custom *Result* function, if the data received from the back-end, the function would parse the JSON data and display in the *Table* component of the HTML page and show the table to users. This web page will import *transaction.css* as style and *echarts.js* to visualize the graph in the front-end, *echarts.js* is used to parse the received graph data from the back-end.

3.6.2.2 Back-end Development

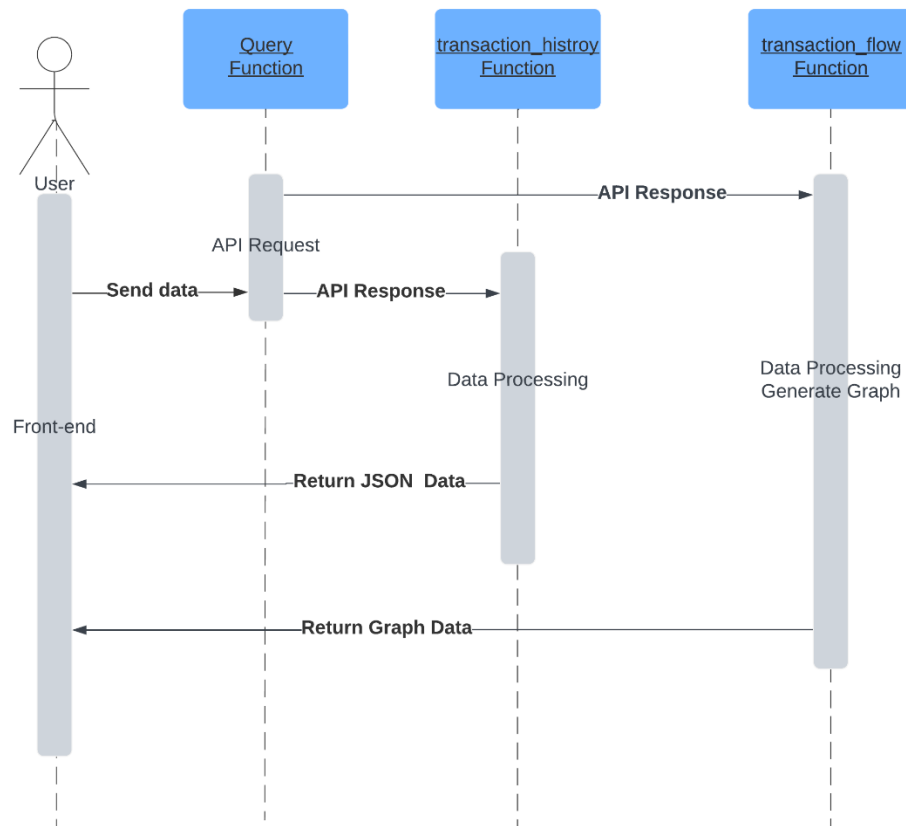


Figure 3.6.2.2.1: Transaction page back-end UML sequence diagram

For the back-end functionality of the *Transaction* page, I provided a UML sequence diagram in Figure 3.6.2.2.1. When the user sends the request data on the front-end, the Query function on the back-end will receive the data information, call the corresponding API and return the API data. The returned API data will sent to two different functions for processing. The first function is *transaction_histroy*, which will repackage the transaction history data for processing, query the blockchain wallet address security of the transaction through the GoPlus malicious address query API, return the data packaged into JSON format to the front-end Jinja2 function for processing and display. The second function is *transaction_flow*, which will use the Pyecharts library to distinguish between the sender (“From”) and the receiver (“To”) in the transaction data, generate the corresponding transaction flow graph, and transfer the graph in data format to the front-end page.

3.6.3 Address Page and Functions Development

3.6.3.1 Front-end Development

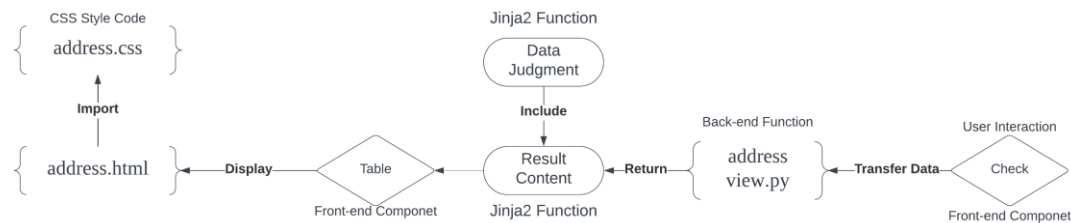


Figure 3.6.3.1.1: Address page front-end code logic

The front-end logic of the *Address* page is relatively simple for the other two pages. As shown in Figure 3.6.3.1.1, after the user makes a query, the front-end input field will transmit the input data to the back-end function for processing. The back-end function will return a JSON data and transfer it to Jinja2 custom *Result* function to parse. At the same time, I add a data judgment here, if the back-end returns the data of the website URL, only the security information of the website URL is displayed in the *Table* component of the front-end. If the back-end returns the query data of the blockchain wallet address, the front-end *Table* component will display the corresponding blockchain wallet address information. This web page will only import *address.css* as style in the *address.html*.

3.6.3.2 Back-end Development

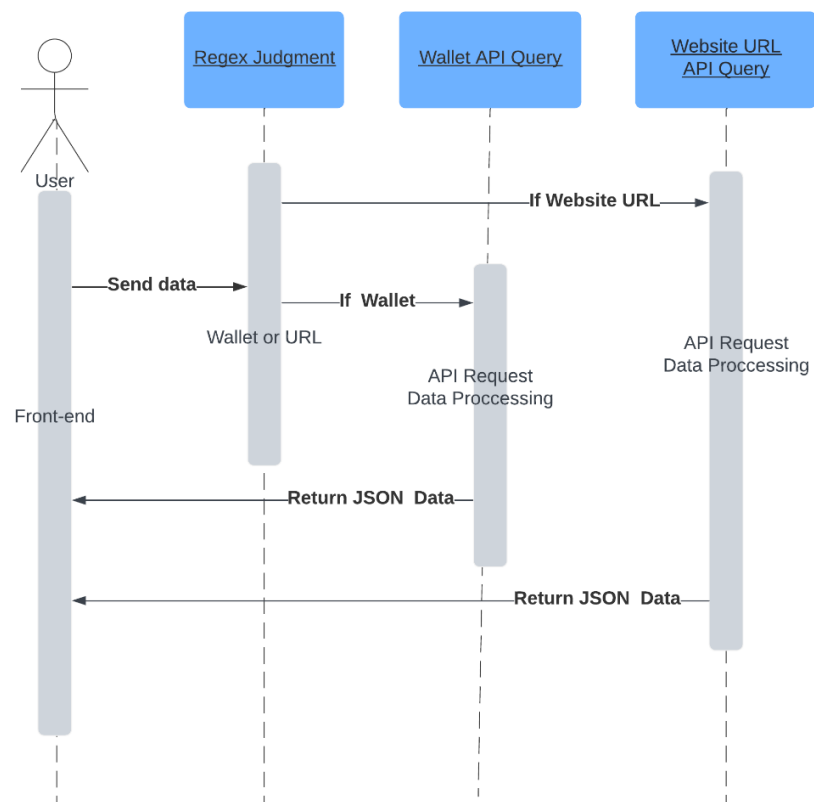


Figure 3.6.3.2.1: Address page back-end UML sequence diagram

This is a UML sequence diagram of the back-end functionality of the *Address* page in Figure 3.6.3.2.1. Unlike other back-end functions, it has a *Regex Judgment* function to distinguish whether the data transmitted by the user from the front-end is a blockchain wallet address or a website URL.

<pre># blockchain wallet address regex formular ethereum_address_pattern = r'^0x[a-fA-F0-9]{40}\$' # website URL regex formular website_link_pattern = r'^(https? http):\\(www\\.)?[\\s/\$. ?#].[^\\s]*\$'</pre>

Table 3.6.3.2.2: Regex judgment formular

Table 3.6.3.2.2 shows the regex formular to process data judgment. If user queries a blockchain wallet address, the data will be sent to the *Wallet API Query* function, which will use the Goplus malicious address query API to package the data back to the front-end in JSON format. If user queries a website URL, the data will be sent to the *Website URL API Query* function, which will use the Goplus phishing website query API to package the data back to the front-end in JSON format.

3.6.4 Smart Contract Page and Functions Development

3.6.4.1 Front-end Development

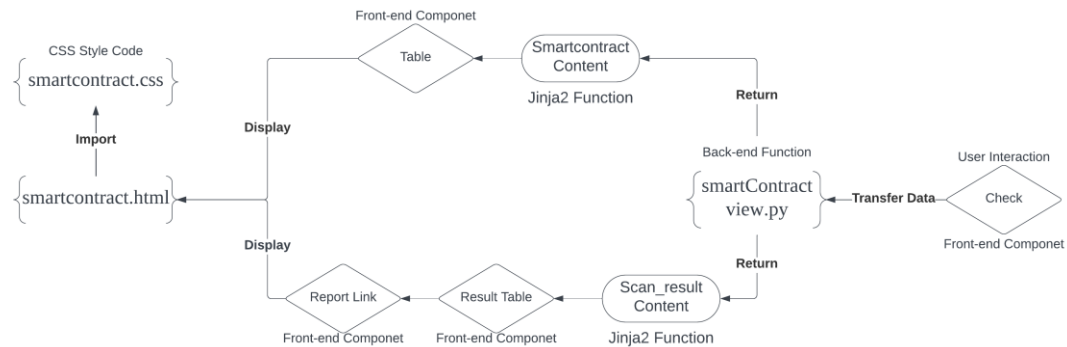


Figure 3.6.4.1.1: Smart contract page front-end code logic

The difference is the *SmartContract* page, which will display two different table data and a redirect link. Based on the display of the code logic diagram in Figure 3.6.4.1.1, after the back-end function receives the data queried by the user, it processes and returns two data, respectively transmitting to the Jinja2 custom function of *Smartcontract* and *Scan_result*. The *Smartcontract* function will process the data into tables that will be displayed on the page. However, the *Scan_result* function processes the data into a table and parses the static page to generate a link for the scan report, which is reflected in the front-end page. This page will import *smartcontract.css* as a style.

3.6.4.2 Back-end Development

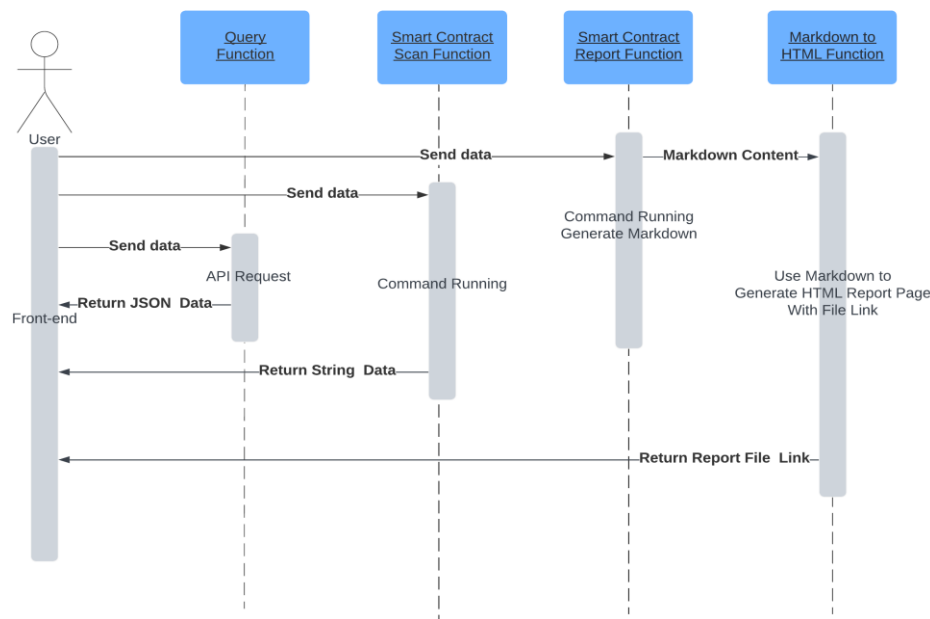


Figure 3.6.4.2.1: Smart contract page back-end UML sequence diagram

The back-end functionality of the *Smart contract* page will be relatively complex, based on the display in Figure 3.6.4.2.1. After the user transfers data from the front-end, the data will send to three different functions. The *Query* function will use the GoPlus smart contract scanning API, the data returned by the API will be processed in the function and packaged into JSON format and returned to the front-end.

```
slither {wallet_address} --print human-summary
```

Table 3.6.4.2.2: Smart contract security scan command in Slither

The second function is the *Smart Contract Scan* function, which will accept the smart contract address queried by the user, call the Slither smart contract static analysis tool through the system command for security scanning, and the returned result will be processed as a string and transmitted to the front-end.

```
slither {wallet_address} --checklist --show-ignored-findings
```

Table 3.6.4.2.3: Smart contract vulnerability report command in Slither

The third function is the *Smart Contract Report* function, it will receive the smart contract address queried by the user, call Slither through the system command to generate the vulnerability report of the smart contract, the report will be sent to the *Markdown to HTML* function through the Markdown format. It will be used to generate HTML static pages in the Markdown format, which eventually returns a link of the report path to the front-end.

3.7 Testing Methodology

I will use the following test methods to evaluate and verify the system, provide the test results and discussion in the Results & Evaluation chapter of the dissertation.

3.7.1 Front-end Usability Testing

For the front-end part, I will do the following tests:

1. Compatibility test:

- Test the front-end display in major desktop browsers (Chrome, Firefox, Safari).
- Test each web page component of the front-end under different operating systems (Windows, macOS, Linux).

2. User interface test:

- Manually test the layout, style, component display, etc. of each page of the website to check whether it meets the design requirements.
- Test user interaction, such as button click, form submission, page navigation, etc., to check whether the interactive feedback is normal.
- Different resolutions of the page adaptability test in desktop screen and mobile screen.

3.7.2 Back-end Unit Testing

For the back-end functions, I will use three different unit testing methods, including:

1. API call test:

- Test functions that call external APIs to ensure API requests are sent and the response data is processed correctly.
- Print the data returned by the APIs and check for compliance with the data usage requirements.

2. Data processing test:

- After the data is returned by the APIs, test the results of the parsing and formatting process, to ensure that the data is processed correctly.

3. Tools integration testing:

- Test functions that integrate external tools, to ensure that the tools are called correctly and gets the expected output.
- Simulate various normal and error outputs of the tool to test error handling logic.

3.8 Summary

In this chapter of my dissertation, I document the development of a secure scanning platform for blockchain wallet transactions.

Firstly, the development process of system development is divided into 5 stages, and the specific development requirements of each functional module are listed. Then, I introduce the main tools and techniques used in the development and explains why they were chosen. After that, the overall architecture design, module components and interactive flow of the system are explained. Also, I used the architecture diagram and component diagram to show them intuitively. This chapter also presents and describes the user interface design for the three main pages of the system. It focuses on the specific details of the development and implementation of the front and back-end.

Finally, I provide testing method to ensure the quality and reliability of the whole system.

4.Result & Evaluation

In the part of development results, I will describe the results of testing methodology for the project and discuss the reasons leading to the results. In the evaluation chapter, I will evaluate the test result and provide an evaluation about the development process. Finally, I will evaluate whether the project meets the aims and objectives.

4.1 Result of Testing Methodology

Multi-platform compatibility is an important aspect of my development projects. Being able to use on multiple operating systems and browsers increases the project usability. I will be testing it on different operating systems and browsers based on the Table 4.1.1.

Operating System	Browser
Windows 11	Chrome
MacOS Big Sur	Safari
Ubuntu 22	Firefox

Table 4.1.1: Testing environment in different system and browser

4.1.1 Front-end Usability Testing

Front-end usability testing will be divided into two parts, compatibility testing and user interface testing. I will describe the test results of them respectively.

4.1.1.1 Compatibility Test

1. Front-end Display Test:

I tested the front-end display in three different browsers and got the results.

Browser	Results
Chrome	Pass
Safari	Pass
Firefox	Fail

Table 4.1.1.1.1: Front-end display results in three browsers

Based on the front-end display results in Table 4.1.1.1.1, Chrome and Safari all pass the testing in desktop mode. They can display all front-end pages normally, and there is no problem with the style of the page. But Firefox was failed in this test.

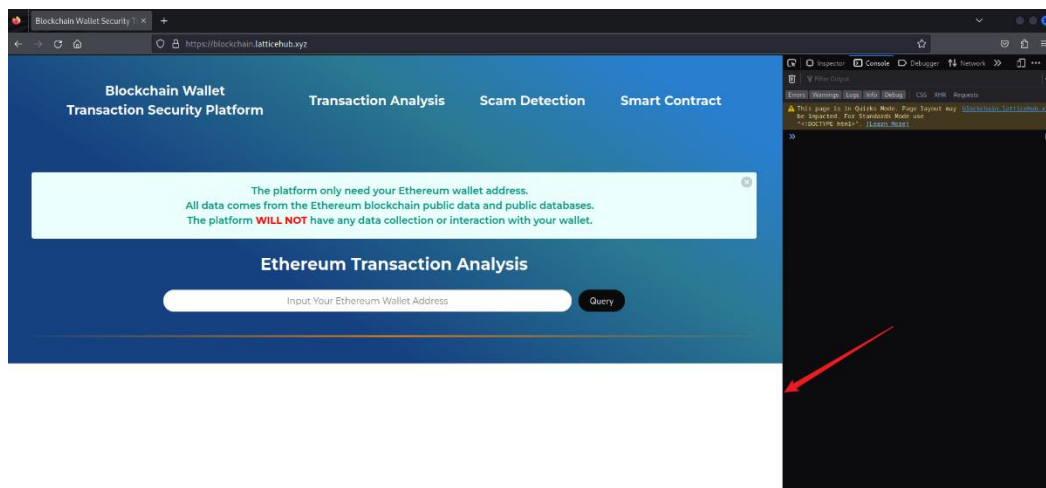


Figure 4.1.1.1.2: Display problem in the Firefox browser

In the Figure 4.1.1.1.2, the front-end page is not displayed properly in Firefox. A large white gap appears at the bottom of the page, however this issue does not appear in Google Chrome and Safari. The reason for this problem is the browser typesetting engine. For HTML files, I need to add identification code to the code to determine how the browser will handle the page. However, I did not handle the display mode, which caused a problem in Firefox.

2. Web Page Components Test:

Second, based on the four web page components in chapter 3.5.4, I tested each web page component that I developed in the front-end page.

Browser	Navigation Bar	Input Field	Button	Notice Box
Chrome	Pass	Pass	Pass	Pass
Safari	Pass	Pass	Pass	Pass
Firefox	Pass	Pass	Pass	Pass

Table 4.1.1.1.3: Web page components test results in three browsers

Table 4.1.1.1.3 provides the test results of web components. For the results in the table, these four web components passed the tests in three different browsers and did not have any display problems in desktop mode.

4.1.1.2 User Interface Test

1. Layout Test:

The layout test is to check whether the front-end page development is completed, and whether it meets the needs of the user interface design.

Browser	Results
Chrome	Pass
Safari	Pass
Firefox	Fail

Table 4.1.1.2.1: Layout test result in three browsers

In Table 4.1.1.2.1, the results of layout test are shown in three different browsers. According to my user interface design phase of the system, I centered all the display layouts, but Firefox had a problem in this test.

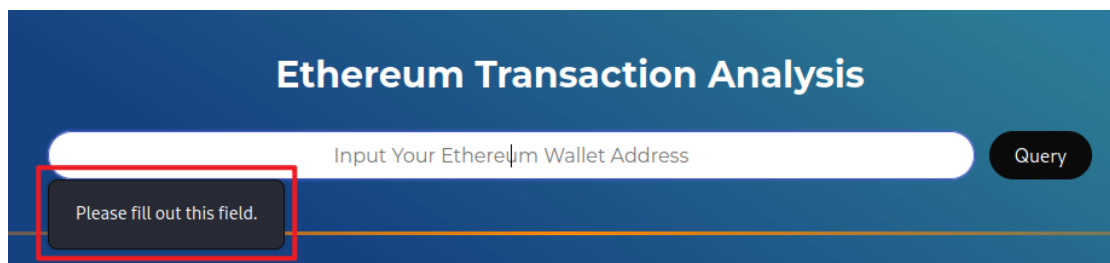


Figure 4.1.1.2.2: Layout problem in Firefox

In the figure above, the prompt of input field is not displayed in the center, it displayed on the left of input field, which does not meet the requirements of the user interface design.

2. Interaction Test:

I chose to conduct interactive testing to ensure that users can use each function of the system normally. For this, I tested the interactivity of the page components in different browsers.

Browser	Navigation Bar	Input Field	Button	Notice Box
Chrome	Pass	Pass	Pass	Pass
Safari	Pass	Pass	Pass	Pass
Firefox	Pass	Pass	Pass	Pass

Table 4.1.1.2.3: Interaction test in three browsers

Based on the results of the interactivity test, all the page components passed the test in each browser.

3. Page Adaptability Test:

The adaptability of the page is checked to ensure that the front page is compatible with different screen sizes. I tested the display of the front page using the size of the desktop screen and the mobile screen.

Screen Type	Results
Desktop Screen	Pass
Mobile Screen	Fail

Table 4.1.1.2.4: Page adaptability test results in different screen type

Table 4.1.1.2.4 shows the results of the page compatibility test under different screen types. It can be concluded from the results that the front-end page has a good compatibility experience and passed the test at the size of the desktop screen, but there will be display problems at the size of the mobile screen.

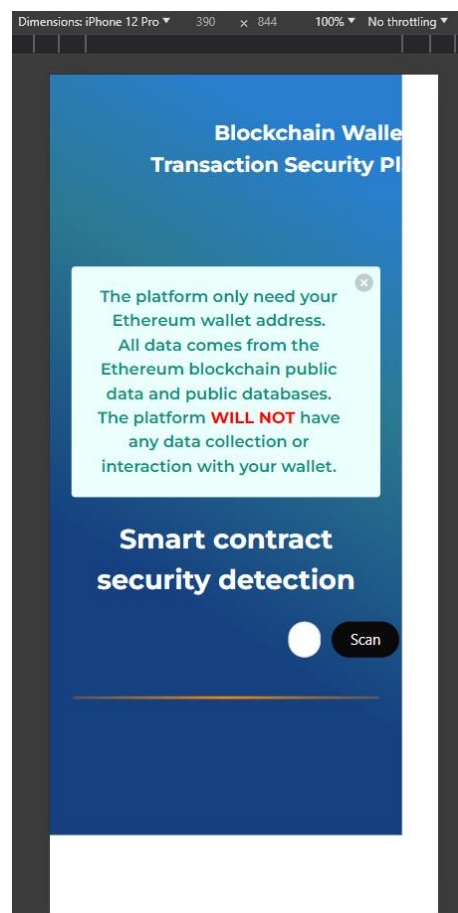


Figure 4.1.1.2.5: Adaptability problem in mobile screen

I simulated the screen size of the iPhone 12 Pro in Chrome for front-end page access. As can be seen in the figure above, the front page does not display properly in the screen size of the mobile. The reason for this problem is that I have not processed CSS style to be compatible with the size of the mobile screen. At this time, the size of the mobile screen cannot display and use the front-end page normally.

4.1.2 Back-end Unit Testing

Unit tests ensure functions are invoked correctly, processed, and returned to the data required after user requests. I will perform three different types of unit tests in this chapter.

Operating System	Environment
Windows 11	Python 3.10

Table 4.1.2.1: Back-end unit testing environment

The table above is my environment for unit testing. I will run the unit tests through the Python, providing the results and descriptions of the tests.

4.1.2.1 APIs Call Test

1. APIs Request Test:

APIs request testing is to ensure that each functional unit can normally request the required API. I will run tests based on the API interfaces described in chapter 2.4, describing the data they request and the results they return.

API Resource	API Description	Request Data	Request Status	Result
Etherscan	Transactions query	Wallet address	200	Pass
GoPlus	Malicious address query	Wallet address	200	Pass
GoPlus	Phishing website query	Website URL	200	Pass
GoPlus	Smart contract scanning	Smart contract address	200	Pass

Table 4.1.2.1.1: APIs request test result in different API

Table 4.1.2.1.1 shows the results of the API requests I used in the back-end. The criteria for determining whether the request was successfully processed is based on the returned request status code. When the request status returns to “200”, the request has been successfully processed by the API server and the requested resource has been returned. This proves that all four APIs I used in the back-end of the project were successful in making requests and getting the information.

2. APIs Response Data Test:

The purpose of testing the data returned by the API is to ensure that the back-end functions can correctly obtain and process the data they need. I will request the API used in the back-end function and print out the data returned by the API to check whether it passes the test.

API Resource	API Description	Response Data	Response Status	Result
Etherscan	Transactions query	Show in Appendix A 1	200	Pass
GoPlus	Malicious address query	Show in Appendix A 2	200	Pass
GoPlus	Phishing website query	Show in Appendix A 3	200	Pass
GoPlus	Smart contract scanning	Show in Appendix A 4	200	Pass

Table 4.1.2.1.2: APIs response data test result in different API

Due to the large amount of data, I have shown the returned data in Appendix A. Based on the above table, all the returned data is consistent with the requirements. The normal return of data can ensure that the subsequent data processing functions can be carried out normally.

4.1.2.2 Data Processing Test

Data processing test is to process the data returned after the request through the APIs and return the processed data to the front-end display. The data processing test is to ensure that the data can be correctly extracted and formatted into a format that the front-end can parse.

Back-end File	Function	Data	Result
Transaction view.py	historydata	Show in Appendix B 1.1	Pass
	graph	Show in Appendix B 1.2	Pass
Address view.py	address_result	Show in Appendix B 2.1	Pass
	website_result	Show in Appendix B 2.2	Pass
SmartContract vview.py	address_result	Show in Appendix B 3.1	Pass
	scan_result	Show in Appendix B 3.2	Pass
	scan_report	Show in Appendix B 3.3	Pass

Table 4.1.2.2.1: Data processing test results in different function

I have shown the data returned by each data processing function in Appendix B. Based on the results in Table 4.1.2.2.1, all the data processing functions effectively process the data returned by the API, and package the data into the format I need. This test ensures that the system can correctly provide effective data feedback.

4.1.2.3 Tools Integration Test

I integrated the external tools Slither and Pyecharts into the project. I chose to make tools integration testing to make sure the tools were being called properly and to get the data output I needed. At the same time, I simulated the correct and wrong output results of the tools to test the error processing logic.

1. Tools Call Test:

I will test the running in Slither and Pyecharts separately, observe the output of both tools, and provide the test results.

Tools	Call Function	Data	Result
Slither	Smart contract scan result	Show in Appendix C 1.1	Pass
	Smart contract scan report	Show in Appendix C 1.2	Pass
Pyecharts	Data Visualization	Show in Appendix C 2.1	Pass

Table 4.1.2.3.1: Tools integration test results in different tools

I have shown the tools output data results in Appendix C. Slither provides the results of the smart contract scan and the text report processed using Markdown. Pyecharts visualizes transaction data and provides users with interactive experience. Users can view the flow of transaction data according to the display in the visualization diagram. Based on the output, all functions were passing the test.

2. Tools Output Test:

The output test is to ensure the tools can provide data feedback to the user in the case of errors and avoid the situation of functional errors. For this part of the processing, I used the subprocess library in the python code for calling the tool. The subprocess library can output the correct and wrong results after the execution of the tool at the same time. I have written judgment rules through the code to deal with different result contents.

Tools	Call Function	Result
Slither	Smart contract scan result	Pass
	Smart contract scan report	Fail
Pyecharts	Data Visualization	Pass

Table 4.1.2.3.2: Tools output test results in different tools

The test results are shown in Table 4.1.2.3.2. As you can see in the table, *Smart contract scan result* and *Data Visualization* function passed the test. But *Smart contract scan report* function in Slither did not pass the test. After checking for errors, I discovered that subprocess library in Python had a

different way of executing commands when calling the tool. Because the function of *Smart contract scan report* is to generate static HTML pages through Markdown processing of Slither output, if there is a problem in the execution of commands, the function will not produce result.

4.2 Project Evaluation

Project evaluation is a systematic process, through a review and evaluation of project, to confirm the achievement of project objectives, identify problems and make suggestions for improvement. In this chapter, I will evaluate the test results, project development process and development requirements, provide corresponding explanations.

4.2.1 Test Result Evaluation

Through front-end and back-end testing, I conducted a comprehensive verification of the system functionality, performance and user experience. Below I will evaluate the front-end and back-end test results separately.

4.2.1.1 Front-end Test Results Evaluation

Front-end testing includes compatibility testing and user interface testing. In the compatibility test, I tested it across three major browsers. The results show that the front page can display normally on Chrome and Safari, but there are problems with the page layout and style display on Firefox. This issue is since I did not handle the typesetting engine of the browser properly. In addition, the main components of the front page, work properly and interact well in all three browsers. In terms of user interface testing, the page layout meets the design requirements on Chrome and Safari, but Firefox has a centered display issue. In addition, under the screen size of mobile devices, the adaptability of the front page is poor, the layout and interactive experience need to be optimized. These issues will be addressed in the future work to provide a better user experience.

4.2.1.2 Back-end Test Results Evaluation

Back-end test includes API call test, data processing test and tool integration test, the test content is centered on the main functions of the system.

The API call test verifies the interaction between the system and the third-party API, and the test results show that all API requests can be sent normally and the required data can be obtained, which lays the foundation for the subsequent data processing. The data processing test ensures that the system can properly extract and format the raw data returned by the API in preparation for the front-end presentation. The tool integration test includes calls to two external tools, Slither and Pyecharts. In the test, both tools were called and output result data

correctly, the error handling logic was verified. The successful integration of the tools brings enhanced functions such as smart contract analysis and data visualization to the system.

In general, the back-end test design is reasonable and the case is sufficient, which can comprehensively check the correctness and robustness of the back-end system, and provide guarantee for the reliable operation of the system. In the subsequent development process, I will continue to test and fix new problems.

4.2.2 Development Process Evaluation

In the process of system development, I formulated a clear development process, which divided the whole project into five stages: system architecture design, user interface design, front-end static development, back-end development and connection, test and debugging. This phased development mode is conducive to project schedule management, ensuring that all aspects of the work are fully considered and implemented, easy to find and solve problems.

4.2.2.1 Development Requirement Setting

First, I clearly stipulated the specific development requirements of front-end, back-end, test and debugging and other aspects, provided clear guidelines for subsequent implementation. These development requirements comprehensively cover all aspects of the system function, considering the aesthetics and practicality of the user interface, but also pay attention to the correctness of the system function and the visual presentation of the data, which is conducive high-quality project.

4.2.2.2 Rationality of Technology Selection

In terms of technology selection, I chose the appropriate development tools and frameworks according to the project requirements. The front-end uses HTML and CSS as web development technologies, with good cross-platform compatibility. The back-end uses the Python and Flask framework, with strong third-party library support, which helps to quickly build functional modules. I chose Slither and Pyecharts to make full use of external resources to enhance the system. The technology selection meets the actual needs and lays a foundation for efficient development.

4.2.2.3 Rationality of System Architecture Design

The system architecture diagram and component diagram I developed clearly describe the division of responsibilities, module division and interaction process at the front and back-end, showing the rationality of the system structure. Through modular design, different functional modules are relatively independent, which is conducive to the division of labor and cooperation development and later maintenance. The architecture design of front-end and back-end separation makes the interface and background logic do not interfere with each other, improves the development efficiency.

4.2.2.4 User Interface Design

I attach great importance to the user interface design of the system. I have designed intuitive and friendly user pages for different functional modules, rationally layout each component. During the design process, I fully considered the user experience and introduced enhanced functions such as data visualization and report generation to meet the needs of different user groups. At the same time, the experience of the interface is enhanced through guiding prompts and ethical reminders.

4.2.2.5 Front, Back-end Development

In terms of front-end development, I adopted a modular programming approach, developed the page components and functions separately, and assembled and integrated them in the main file. The front-end code is logically clear and in line with Web development best practices. In the back-end development, I used UML sequence diagram to model the interaction flow of each function module in detail and explained the key functions, reflecting the standardization and traceability of the development process.

4.2.3 Develop Requirements Evaluation

4.2.3.1 Front-end Development Evaluation

1. Need to follow the GUI design to develop

During the development process, I designed intuitive and friendly user interfaces for different functional modules, including three pages of transaction analysis, fraud detection and smart contract. This meets the need for front-end development based on GUI design.

2. Suitable for all device display sizes

I conducted a page adaptive test and found that the front page would have display problems under the moving screen size, which did not fully meet the adaptive requirements. However, I am aware of the problem and plan to optimize the display on different devices in future development.

3. The data returned by the back-end must be displayed properly.

My front-end development adopts a modular approach, using the Jinja2 template engine to correctly present the data returned from the back-end in different front-end components to meet this requirement.

4.2.3.2 Back-end Function Development

1. Properly handle API requests and return data

My back-end unit test includes API call test and response data test, verifying that the system can correctly send API requests and obtain required data, which meets this requirement.

2. Use visualization libraries to transform data into visual charts

I adopted the Pyecharts visualization library, which can convert transaction data into an interactive flow chart to meet the requirements of data visualization.

3. Use the smart contract tool and process the returned results

My back-end development has integrated Slither, which can scan for vulnerabilities and generate reports on contract addresses entered by users, to meet this need.

4. Properly transfer data to the front-end

I adopted the standardized front-end separation architecture, the back-end data was transmitted to the front-end in JSON formats after processing, it was correctly parsed and displayed by the front-end template engine, thus achieving the correct transmission of data.

4.2.3.3 Testing and Debugging

1. User interface usability testing

My test results document includes front-end usability testing that fully tests compatibility, layout, interaction, and adaptability to meet this requirement.

2. Unit tests on each function

In my back-end unit test, including API call test, data processing test and tool integration test, unit tests the core function modules of the system to meet this requirement.

3. Front-end and back-end communication testing and debugging

I involved the realization of the front and back-end data interaction in the development process, and carried out the corresponding testing and debugging work, which met the requirement of this part.

4.3 Summary

In this chapter, I evaluated the test results, expounded the problems and deficiencies found in the test, I will improve and optimize in the future. Then I evaluated the development process, decided the project was reasonable and feasible. I evaluated whether the front-end and back-end functions of the implementation met the development requirements. Most of the requirements were realized, but some details still needed to be improved.

5. Conclusions

After evaluating the test results and system development requirements, I will provide a complete summary in this chapter. I will review and determine whether the original aim and objectives of the project have been met, discuss the project plan and current progress. After that, I will make a personal summary and reflection, describe what I have learned in this project, conduct a complete review and personal evaluation of the project. According to the results of the development of the project, determine the work that needs to be carried out in the future of the project.

5.1 Review of Aim and Objectives

5.1.1 Project Aim

The aim of the project I have set is to develop a platform that will provide users with security scan before conducting blockchain wallet transactions.

According to my development process in chapter 3, I successfully developed a blockchain wallet transaction security platform, basically achieving the overall aim of the project. Specifically:

1. Transaction history query and visualization capabilities have been implemented, allowing users to query all transaction records of the wallet address, and visualize the flow of funds through an interactive flow chart, while flagging risky transactions.
2. Blacklisting addresses and website detection have been implemented, allowing users to check whether a particular wallet address or website is involved in fraudulent activity and to list common risk types for reference.
3. Smart contract security scan function has been implemented, users can carry out security detection of smart contract addresses, understand contract risks, and generate vulnerability reports for developers to help repair.

5.1.2 Project Objectives

1. **Objective 1** - Satisfy

I mentioned in the report that I investigated and analyzed the existing platform, which provided references for my subsequent development.

2. **Objective 2** - Satisfy

In the process of technical research, I demonstrated a deep understanding of blockchain wallet with transaction processing and security issues.

3. Objective 3 - Satisfy

I implemented address and website risk detection by integrating data from public APIs like GoPlus and Etherscan.

4. Objective 4 - Satisfy

I used API resources from blockchain explorer such as Etherscan to obtain transaction data and visualized the data using the Pyecharts library.

5. Objective 5 - Satisfy

I learned the Slither smart contract static analysis tool, added security scanning and reporting functions from Slither into the platform.

6. Objective 6 - Satisfy

I designed a user-friendly GUI interface and conducted usability tests. All the functions of the system are realized through the development of front and back-end.

5.2 Updated Project Plan

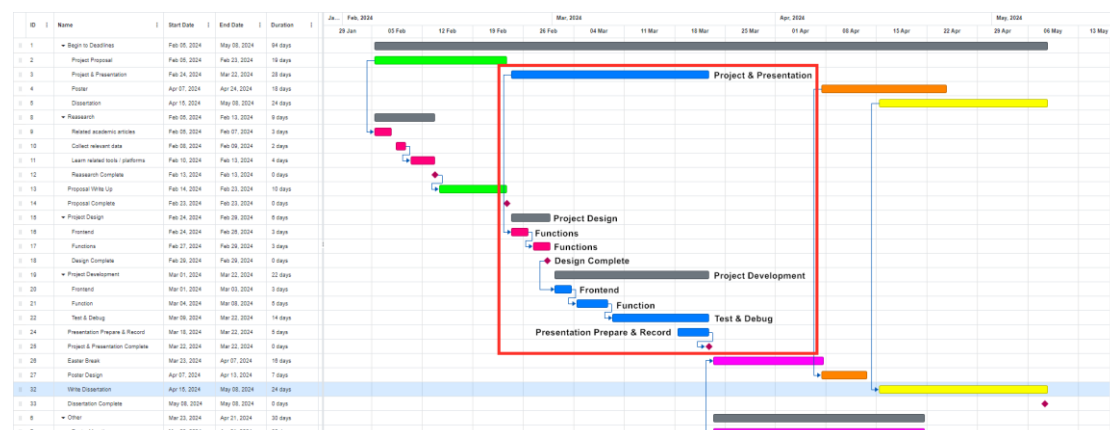


Figure 5.2.1: Updated project timetable

The timetable for project development is shown in the red box in Figure 5.2.1, which I have updated retrospectively. As shown in the figure, the development of the project is in accordance with the plan. The project design phase will take the same amount of time as previously planned. In the project development stage, I completed the development of the front-end page two days ahead of schedule, and the development of the back-end functions was the same as the previous plan. But in the testing and debugging phase, it took me a week longer than I had planned. I tried to use the Matplotlib library to generate a visualization of transaction flows, but the results did not meet my requirement. Because of this, I spent 3 days changing and using the Pyecharts, which is better to meet my needs and provided better user usability. Then, it took me 4 days to debug

and make sure the external tools were stable. In the last 7 days, I conducted a complete test of the project, and the problems were solved at this stage.

5.3 Personal Conclusions

5.3.1 What I Learned

Based on the progress and development process of the whole project, I have summarized what I have learned as follows:

1. **System architecture design ability**

I designed a reasonable layered architecture for the whole system, modular development and learned how to design a good system.

2. **Front and back-end development technology**

I am skilled in using front-end technologies for develop a modular front-end code, achieving user-friendly interface and good interactive experience. As a back-end development language, I have a solid grasp of Python. By using Flask framework and Jinja2 template engine, built a fully functional back-end system.

3. **Blockchain and smart contract technology**

In the process of development, I learned the use of blockchain wallet, transaction processing process and understanding of the security problems of blockchain wallet, which laid the foundation for the implementation of system functions. At the same time, I learned the Slither, integrated it into the system to provide users with smart contract security scanning and vulnerability report generation functions.

4. **Application of data processing and visualization techniques**

I investigated and integrated multiple public APIs to implement main functions in the platform, while correctly processing the data returned by the API. After data processing, I learned to use the Pyecharts visualization library to transform blockchain transaction data into interactive flow charts, improving the form of data presentation and user experience.

5.3.2 What Went Well

At the beginning of the project, I clearly set the overall aim and specific objectives, which cover the development of a blockchain wallet security scanning platform. This provides a foundation for the direction of the whole project. When choosing the development technology, I decided to use web technologies such as HTML in the front-end, Python and Flask framework in the back-end. The technology selection fully considered the project needs and was able to efficiently complete the development of various functions. The system architecture, module division and component interaction flow designed are reasonable, the front and back-end separation reflect good scalability. For the design of the front-end, I added enhanced functions such as data visualization and report generation, introduced operational guidelines and ethical considerations to show the humanized design concept. In the process of development, I have adopted a standardized development process, each process has been fully documented and evaluated. In testing and debugging, I conducted multi-dimensional usability testing in front-end testing. In the back-end, I used unit testing and integration testing, which can effectively discover and repair system problems.

5.3.3 What Could Be Better

After reviewing the test results and the system, I think there are a few things I can do better:

1. Improved Mobile adaptation

During front-end testing, I found that the front-end page adaptability on mobile device was poor, there were some problems in layout and interactive experience. While I am aware of this and plan to optimize in subsequent development, mobile device is an important factor that today's systems must consider and needs to be addressed as a priority in the future development.

2. Enhanced exception handling

I included exception handling of the tools output in the back-end tests. However, at the level of the whole system, I can further strengthen the exception and fault tolerance processing capabilities, especially for illegal user input, network anomalies and other situations to avoid system crash.

3. Use of version control

I do not use version control systems such as Git in my development process. Version control systems facilitate code collaboration, version management, and problem tracking. This is something I need to add to my project development.

5.4 Work in The Future and Research

I think the improvement of the project is the most important work in the future. According to what I have listed in chapter 5.1.2, I will first improve them in future work. Next, I need to continuously improve and strengthen the core functions of the system, which are mainly divided into the following three parts:

1. **Improve the transaction tracking function**

The improvement of the transaction tracking function is to enhance the anti-money laundering capabilities and enhance the security of blockchain transactions. In future work, I plan to expand the scope of tracking to include not only direct transactions of wallet addresses, but also the source and destination of transaction funds. According to my investigation findings in Chapter 2.3, it is more conducive to discover potential risky behaviors by analyzing the entire fund flow link.

2. **Smart contracts dynamic calling and debugging**

Supports dynamic deployment and invocation of smart contract methods, rather than just static scanning of existing contracts. I investigated Remix IDE, an online visual smart contract development tool. [16] I can reference it to provide an online smart contract development and debugging environment on the front-end. This can compensate for the shortcomings of existing analytics tools and provide sufficient convenience for smart contract developers and users.

3. **Make project as an open-source framework**

This part of the work is the main goal of the project in the future. After perfecting the system in this project, I hope to sort out the code structure, build reusable libraries and middleware, easy to integration with other projects. Develop perfect technical documentation for users in need and reduce the threshold for developers to get started. I also hope to build an active community where developers can ask questions and needs, work together to push the project forward.

5.5 Final Thoughts

I would like to review the whole process of the project in this part.

When I first came up with this idea, I was eager to contribute to the security of blockchain wallet transactions. Through investigation and analysis, I made it clear to build a security scanning platform that integrates transaction query, risk detection and smart contract audit.

While developing this project, I have benefited a lot. Whether it is system architecture design, front and back-end development, or blockchain wallet and smart contract technology, my ability has been greatly improved.

Even so, this project is not perfect, and there are still improvements to be made in terms of mobile adaptation and exception handling. But I am proud that I was able to objectively see my own shortcomings and make a future work plan to point the project in a new direction.

I have submitted the project code to supplementary material, everyone can use and test my project directly.

I hope I can use my own technology and passion to provide users with a more secure and reliable blockchain transaction experience.

6. References

- [1] - Howarth, J. (2022). 71+ Incredible Cryptocurrency Statistics (2022). [online] Exploding Topics. Available at: <https://explodingtopics.com/blog/cryptocurrency-stats> [Accessed 6 Apr. 2024].
- [2] - Chainalysis Team (2024). Stolen Crypto Falls in 2023, but Hacking Remains a Threat. [online] Available at: <https://www.chainalysis.com/blog/crypto-hacking-stolen-funds-2024/> [Accessed 6 Apr. 2024].
- [3] - Suratkar, S., Shirole, M. and Bhirud, S. (2020). Cryptocurrency Wallet: A Review. [online] IEEE Xplore. doi:<https://doi.org/10.1109/ICCCSP49186.2020.9315193>.
- [4] - Qian, L.Y. (2023). Most Popular Crypto Hot Wallets for Self-Custody. [online] Available at: <https://www.coingecko.com/research/publications/most-popular-crypto-hot-wallets> [Accessed: 07 Apr. 2024].
- [5] - Xu, Y., Rustandi, I., Ma, Y. and Dialani, V. (2023). Detecting the Undetectable: Coinbase ERC-20 Scam Token Detection System. [online] Available at: <https://www.coinbase.com/en-gb/blog/detecting-the-undetectable-coinbase-erc-20-scam-token-detection-system> [Accessed 9 Apr. 2024].
- [6] - MetaMask (2023). Using transaction security providers to protect yourself from scams. [online] Available at: <https://support.metamask.io/hc/en-us/articles/12539282795675-Using-transaction-security-providers-to-protect-yourself-from-scams> [Accessed 9 Apr. 2024].
- [7] - etherscan.io (2019). Ethereum (ETH) Blockchain Explorer. [online] Ethereum (ETH) Blockchain Explorer. Available at: <https://etherscan.io/> [Accessed 10 Apr. 2024].
- [8] - GoPlus (2021). GoPlus Security. [online] GoPlus Security. Available at: <https://gopluslabs.io/> [Accessed 10 Apr. 2024].
- [9] - Feist, J., Grieco, G. and Groce, A. (2019). Slither: A Static Analysis Framework for Smart Contracts. [online] IEEE Xplore. doi:<https://doi.org/10.1109/WETSEB.2019.00008>.

[10] - Researcher (2024). Scam Sniffer 2023: Crypto Phishing Scams Drain \$300 Million from 320,000 Users. [online] Scam Sniffer. Available at: <https://drops.scamsniffer.io/post/scam-sniffer-2023-crypto-phishing-scams-drain-300-million-from-320000-users/> [Accessed 10 Apr. 2024].

[11] - MDN Web Docs (2018). HTML: HyperText Markup Language. [online] MDN Web Docs. Available at: <https://developer.mozilla.org/en-US/docs/Web/HTML> [Accessed 12 Apr. 2024].

[12] - Mozilla (2019). CSS: Cascading Style Sheets. [online] MDN Web Docs. Available at: <https://developer.mozilla.org/en-US/docs/Web/CSS> [Accessed 12 Apr. 2024].

[13] - Bulma.io. (2017). Bulma: Free, open source, & modern CSS framework based on Flexbox. [online] Available at: <https://bulma.io/> [Accessed 12 Apr. 2024].

[14] - Ronacher, A. (2010). The Python micro framework for building web applications. [online] Flask. Available at: <https://flask.palletsprojects.com/en/3.0.x/> [Accessed 12 Apr. 2024].

[15] - pyecharts.org. (2017). pyecharts - A Python Echarts Plotting Library built with love. [online] Available at: <https://pyecharts.org/> [Accessed 12 Apr. 2024].

[16] - Remix (2020). Remix - Ethereum IDE & community. [online] Available at: <https://remix-project.org/>. [Accessed 15 Apr. 2024]

7. Appendices

7.1 Appendix A – APIs Response Data

1. Etherscan – Transactions Query:

```
{
  "status": "1",
  "message": "OK",
  "result": [
    {
      "blockNumber": "17861764",
      "timeStamp": "1691395667",
      "hash":
      "0x2ae7ec84c43530b22a7f0b0086287378f0d14349a0790c85b53f940d0cc9cecd",
      "nonce": "124",
      "blockHash":
      "0xea687c2dcde58f4cd639fcdc0ee62f7c22dbc13a4c9cf9a3827a1cca3d6c9dde",
      "transactionIndex": "117",
      "from": "0x6c619952999e8a5cc9b4c928b2602aa60a061e58",
      "to": "0x1e7883b496d6d8e9d5dab005434ed095caa7eb91",
      "value": "34163523716489304",
      "gas": "21000",
      "gasPrice": "17815044325",
      "isError": "0",
      "txreceipt_status": "1",
      "input": "0x",
      "contractAddress": "",
      "cumulativeGasUsed": "9880645",
      "gasUsed": "21000",
      "confirmations": "1628060",
      "methodId": "0x",
      "functionName": ""
    }
  ]
}
```

2. GoPlus – Malicious Address Query:

```
{
  "code": 1,
  "message": "ok",
  "result": {
    "cybercrime": "0",
    "money_laundering": "0",
    "number_of_malicious_contracts_created": "0",
    "gas_abuse": "0",
    "financial_crime": "0",
    "darkweb_transactions": "0",
    "reinit": "0",
    "phishing_activities": "0",
    "contract_address": "0",
    "fake_kyc": "0",
    "blacklist_doubt": "0",
    "fake_standard_interface": "0",
    "data_source": "",
    "stealing_attack": "0",
    "blackmail_activities": "0",
    "sanctioned": "0",
    "malicious_mining_activities": "0",
    "mixer": "0",
    "fake_token": "0",
    "honeypot_related_address": "0"
  }
}
```

3. GoPlus – Phishing Website Query:

```
{
  "code": 1,
  "message": "OK",
  "result": {
    "website_contract_security": [],
    "phishing_site": 0
  }
}
```

4. GoPlus – Smart Contract Scanning:

```
{
  "code": 1,
  "message": "OK",
  "result": {
    "0xc4831af8016ee6a17ab32b56a5a5152dd1be5546": {
      "anti_whale_modifiable": "0",
      "buy_tax": "0.01",
      "can_take_back_ownership": "0",
      "cannot_buy": "0",
      "cannot_sell_all": "0",
      "creator_address": "0x1f01f60f89d24faee17a2fdb06b6854d4628fa2d",
      "creator_balance": "6981217.154776467",
      "creator_percent": "0.006981",
      "dex": [
        {
          "liquidity_type": "UniV2",
          "name": "UniswapV2",
          "liquidity": "7300.96747917",
          "pair": "0xfb4c7f4360360b88495a60e6d1b0d0fb0bbb1a17"
        }
      ],
      "external_call": "0",
      "hidden_owner": "0",
      "holder_count": "538",
      "holders": [
        {
          "address": "0xfb4c7f4360360b88495a60e6d1b0d0fb0bbb1a17",
          "tag": "UniswapV2",
          "is_contract": 1,
          "balance": "762308315.498926192",
          "percent": "0.762308315498926192",
          "is_locked": 0
        }
      ],
      "honeypot_with_same_creator": "0",
      "is_anti_whale": "1",
      "is_blacklisted": "0",
      "is_honeypot": "0",
      "is_in_dex": "1",
      "is_mintable": "0",
      "is_open_source": "1",
      "is_proxy": "0",
      "is_whitelisted": "1",
```

```
"lp_holder_count": "2",
"lp_holders": [
  {
    "address": "0x0000000000000000000000000000000000000000dead",
    "tag": "",
    "value": null,
    "is_contract": 0,
    "balance": "1.152291605495523553",
    "percent": "0.999999999999999132",
    "NFT_list": null,
    "is_locked": 1
  },
  {
    "address": "0x0000000000000000000000000000000000000000",
    "tag": "Null Address",
    "value": null,
    "is_contract": 0,
    "balance": "0.0000000000000001",
    "percent": "0.000000000000000867",
    "NFT_list": null,
    "is_locked": 1
  }
],
"lp_total_supply": "1.152291605495524553",
"owner_address": "0x0000000000000000000000000000000000000000",
"owner_balance": "0",
"owner_change_balance": "0",
"owner_percent": "0.000000",
"personal_slippage_modifiable": "0",
"selfdestruct": "0",
"sell_tax": "0.01",
"slippage_modifiable": "0",
"token_name": "DOGECOIN",
"token_symbol": "DOGECOIN",
"total_supply": "100000000",
"trading_cooldown": "1",
"transfer_pausable": "0"
}
}
```

7.2 Appendix B – Data Processing Test Data

1. Transaction view.py

1.1 historydata:

```
{'hash':  
'0x2ae7ec84c43530b22a7f0b0086287378f0d14349a0790c85b53f940d0cc9cecd',  
'from': 'You', 'to': '0x1e7883b496d6d8e9d5dab005434ed095caa7eb91', 'safety':  
'Safe'}
```

1.2 graph:

```
{  
  "animation": true,  
  "animationThreshold": 2000,  
  "animationDuration": 1000,  
  "animationEasing": "cubicOut",  
  "animationDelay": 0,  
  "animationDurationUpdate": 300,  
  "animationEasingUpdate": "cubicOut",  
  "animationDelayUpdate": 0,  
  "aria": {  
    "enabled": false  
  },  
  "color": [  
    "#5470c6",  
    "#91cc75",  
    "#fac858",  
    "#ee6666",  
    "#73c0de",  
    "#3ba272",  
    "#fc8452",  
    "#9a60b4",  
    "#ea7ccc"  
  ],  
  "series": [  
    {  
      "type": "graph",  
      "layout": "force",  
      "symbolSize": 10,  
      "circular": {  
        "rotateLabel": false  
      },  
      "force": {  
        "repulsion": 8000,  
        "gravity": 0.2,  

```

```
        "edgeLength": 30,
        "friction": 0.6,
        "layoutAnimation": true
    },
    "label": {
        "show": false,
        "margin": 8
    },
    "lineStyle": {
        "show": true,
        "width": 1,
        "opacity": 1,
        "curveness": 0,
        "type": "solid",
        "color": "black"
    },
    "roam": true,
    "draggable": false,
    "focusNodeAdjacency": true,
    "data": [
        {
            "name": "You"
        },
        {
            "name":
"0x1e7883b496d6d8e9d5dab005434ed095caa7eb91"
        },
        {
            "name": "0x1111111254eeb25477b68fb85ed929f73a960582"
        },
        {
            "name":
"0xc4831af8016ee6a17ab32b56a5a5152dd1be5546"
        },
        {
            "name": "0x2719ee9eb38ff9bdbc0a0e5bdb354a2da3a9b7f3"
        },
        {
            "name": "0xb8c77482e45f1f44de1745f52c74426c631bdd52"
        },
        {
            "name": "0x16b2f56890583a38def7bc3372e170ec369fb2cb"
        }
    ],
```

```

"edgeLabel": {
  "show": false,
  "margin": 8
},
"edgeSymbol": [
  "circle",
  "arrow"
],
"edgeSymbolSize": [
  10,
  15
],
"links": [
  {
    "source": "You",
    "target":
"0x1e7883b496d6d8e9d5dab005434ed095caa7eb91"
  },
  {
    "source": "You",
    "target": "0x1111111254eeb25477b68fb85ed929f73a960582"
  },
  {
    "source": "You",
    "target": "0x1111111254eeb25477b68fb85ed929f73a960582"
  },
  {
    "source": "You",
    "target": "0x1111111254eeb25477b68fb85ed929f73a960582"
  },
  {
    "source": "You",
    "target":
"0xc4831af8016ee6a17ab32b56a5a5152dd1be5546"
  },
  {
    "source": "You",
    "target": "0x1111111254eeb25477b68fb85ed929f73a960582"
  },
  {
    "source": "You",
    "target":
"0x1e7883b496d6d8e9d5dab005434ed095caa7eb91"
  },

```

```
        {
            "source": "You",
            "target": "0x2719ee9eb38ff9bdbc0a0e5bdb354a2da3a9b7f3"
        },
        {
            "source": "You",
            "target": "0xb8c77482e45f1f44de1745f52c74426c631bdd52"
        },
        {
            "source": "You",
            "target": "0x16b2f56890583a38def7bc3372e170ec369fb2cb"
        }
    ],
    "itemStyle": {
        "color": "red"
    }
},
"legend": [
    {
        "data": [],
        "selected": {}
    }
],
"tooltip": {
    "show": true,
    "trigger": "item",
    "triggerOn": "mousemove|click",
    "axisPointer": {
        "type": "line"
    },
    "showContent": true,
    "alwaysShowContent": false,
    "showDelay": 0,
    "hideDelay": 100,
    "enterable": false,
    "confine": false,
    "appendToBody": false,
    "transitionDuration": 0.4,
    "textStyle": {
        "fontSize": 14
    },
    "borderWidth": 0,
    "padding": 5,
```



```
        "order": "seriesAsc"
    }
}
```

2. Address view.py

2.1 Address_result:

```
{'cybercrime': 'Safe', 'money_laundering': 'Safe', 'financial_crime': 'Safe',
'darkweb_transactions': 'Safe', 'phishing_activities': 'Risk', 'fake_kyc': 'Safe',
'stealing_attack': 'Risk', 'blackmail_activities': 'Safe', 'malicious_mining_activities':
'Safe', 'safety': 'Risk', 'data_source': 'SlowMist,ScamSniffer,BlockSec'}
```

2.2 Website_result:

```
{'phishing_site': 'Risk', 'data_source': 'GoPlus Security'}
```

3. SmartContract view.py

3.1 address_result:

```
{'is_honeypot': 'Safe', 'is_blacklisted': 'Safe', 'is_open_source': 'Safe', 'is_proxy':
'Safe', 'transfer_pausable': 'Safe', 'trading_cooldown': 'Risk', 'safety': 'Risk',
'data_source': 'GoPlus Security'}
```

3.2 scan_result:

```
['Total number of contracts in source files: 7', 'Source lines of code (SLOC) in source
files: 260', 'Number of assembly lines: 0', 'Number of optimization issues: 10',
'Number of informational issues: 20', 'Number of low issues: 6', 'Number of medium
issues: 3', 'Number of high issues: 3']
```

3.3 scan_report:

```
/static/reports/0xc4831af8016ee6a17ab32b56a5a5152dd1be5546_report.html
```

7.3 Appendix C – Tools Call Test Data

1. Slither

1.1 Smart Contract Scan Result:

'Total number of contracts in source files: 7', 'Source lines of code (SLOC) in source files: 260', 'Number of assembly lines: 0', 'Number of optimization issues: 10', 'Number of informational issues: 20', 'Number of low issues: 6', 'Number of medium issues: 3', 'Number of high issues: 3'

1.2 Smart Contract Scan Report:

Summary

- [reentrancy-no-eth](#) (2 results) (Medium)
- [missing-zero-check](#) (1 results) (Low)
- [reentrancy-benign](#) (5 results) (Low)
- [reentrancy-events](#) (6 results) (Low)
- [dead-code](#) (10 results) (Informational)
- [solc-version](#) (2 results) (Informational)
- [constable-states](#) (1 results) (Optimization)

reentrancy-no-eth

Impact: Medium Confidence: Medium - ID-0 Reentrancy in [DSAAuth.setOwner\(address\)](#): External calls: - [auth\(\)](#) - [authority.canCall\(src.this.sig\)](#) State variables written after the call(s): - [owner = owner_DSAAuth.owner](#) can be used in cross function reentrancies: - [DSAAuth.DSAAuth\(\)](#) - [DSAAuth.isAuthorized\(address.bytes4\)](#) - [DSAAuth.setOwner\(address\)](#)

crtyic-export/etherscan-contracts/0x9f8f72aa9304c8b593d555f12ef6589cc3a579a2-DSToken.sol#L83-L95

- ID-1 Reentrancy in [DSAAuth.setAuthority\(DSAAuthority\)](#): External calls:
 - [auth\(\)](#)
 - [authority.canCall\(src.this.sig\)](#) State variables written after the call(s):
 - [authority = authority_DSAAuth.authority](#) can be used in cross function reentrancies:
 - [DSAAuth.authority](#)
 - [DSAAuth.isAuthorized\(address.bytes4\)](#)
 - [DSAAuth.setAuthority\(DSAAuthority\)](#)

crtyic-export/etherscan-contracts/0x9f8f72aa9304c8b593d555f12ef6589cc3a579a2-DSToken.sol#L99-L111

missing-zero-check

Impact: Low Confidence: Medium - ID-2 [DSAAuth.setOwner\(address\).owner_](#) lacks a zero-check on : - [owner = owner_](#)

crtyic-export/etherscan-contracts/0x9f8f72aa9304c8b593d555f12ef6589cc3a579a2-DSToken.sol#L83

reentrancy-benign

Impact: Low Confidence: Medium - ID-3 Reentrancy in [DSToken.mint\(address,uint256\)](#): External calls: - [auth\(\)](#) - [authority.canCall\(src.this.sig\)](#) State variables written after the call(s): - [_balances\[guv\] = add\(_balances\[guv\].wad\)](#) - [_supply = add\(_supply.wad\)](#)

crtyic-export/etherscan-contracts/0x9f8f72aa9304c8b593d555f12ef6589cc3a579a2-DSToken.sol#L905-L913

- ID-4 Reentrancy in [DSToken.start\(\)](#): External calls:
 - [auth\(\)](#)
 - [authority.canCall\(src.this.sig\)](#) State variables written after the call(s):
 - [stopped = false](#)

crtyic-export/etherscan-contracts/0x9f8f72aa9304c8b593d555f12ef6589cc3a579a2-DSToken.sol#L531-L535

- ID-5 Reentrancy in [DSToken.burn\(address,uint256\)](#): External calls:
 - [auth\(\)](#)
 - [authority.canCall\(src.this.sig\)](#) State variables written after the call(s):
 - [_approvals\[guv\]\[msg.sender\] = sub\(_approvals\[guv\]\[msg.sender\].wad\)](#)
 - [_balances\[guv\] = sub\(_balances\[guv\].wad\)](#)
 - [_supply = sub\(_supply.wad\)](#)

crtyic-export/etherscan-contracts/0x9f8f72aa9304c8b593d555f12ef6589cc3a579a2-DSToken.sol#L915-L931

- ID-6 Reentrancy in [DSToken.setName\(bytes32\)](#): External calls:
 - [auth\(\)](#)
 - [authority.canCall\(src.this.sig\)](#) State variables written after the call(s):
 - [name = name_](#)

crtyic-export/etherscan-contracts/0x9f8f72aa9304c8b593d555f12ef6589cc3a579a2-DSToken.sol#L941-L945

- ID-7 Reentrancy in [DSToken.stop\(\)](#): External calls:
 - [auth\(\)](#)
 - [authority.canCall\(src.this.sig\)](#) State variables written after the call(s):
 - [stopped = true](#)

crtyic-export/etherscan-contracts/0x9f8f72aa9304c8b593d555f12ef6589cc3a579a2-DSToken.sol#L525-L529

reentrancy-events

Impact: Low Confidence: Medium - ID-8 Reentrancy in [DSToken.burn\(address,uint256\)](#): External calls: - [auth\(\)](#) - [authority.canCall\(src.this.sig\)](#) Event emitted after the call(s): - [Burn\(guv.wad\)](#)

crtyic-export/etherscan-contracts/0x9f8f72aa9304c8b593d555f12ef6589cc3a579a2-DSToken.sol#L915-L931

- ID-9 Reentrancy in [DSToken.stop\(\)](#): External calls:
 - [auth\(\)](#)
 - [authority.canCall\(src.this.sig\)](#) Event emitted after the call(s):
 - [LogNote\(msg.sig,msg.sender.foo,msg.value,msg.data\)](#)
 - [note\(\)](#)

crtyic-export/etherscan-contracts/0x9f8f72aa9304c8b593d555f12ef6589cc3a579a2-DSToken.sol#L525-L529

- ID-10 Reentrancy in [DSToken.start\(\)](#): External calls:
 - [auth\(\)](#)
 - [authority.canCall\(src.this.sig\)](#) Event emitted after the call(s):
 - [LogNote\(msg.sig,msg.sender.foo,msg.value,msg.data\)](#)
 - [note\(\)](#)

crtyic-export/etherscan-contracts/0x9f8f72aa9304c8b593d555f12ef6589cc3a579a2-DSToken.sol#L531-L535

- ID-11 Reentrancy in [DSAAuth.setAuthority\(DSAAuthority\)](#): External calls:
 - [auth\(\)](#)
 - [authority.canCall\(src.this.sig\)](#) Event emitted after the call(s):

2. Pyechart

2.1 Data Visualization:

