SCRL

# Full Audit Report

## Gptfusion Token Security Assessment

Document ID: 61D3952D33D07FE317C81E553DD1D2E3

Gptfusion Token Security Assessment

# FULL AUDIT REPORT

Security Assessment by SCRL on **Monday, May 12, 2025**

SCRL is deliver a security solution for Web3 projects by expert security researchers.

## Executive Summary

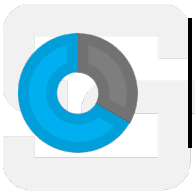
For this security assessment, SCRL received a request on Saturday, May 10, 2025

| Client | Language | Audit Method | Confidential | Network Chain | Contract |
|---|---|---|---|---|---|
| **GPTFUSION Token** | **Solidity** | **Whitebox** | **Public** | **BNB Chain** | **0xA4230A63Bd1d5804929c95A3eA0e658e8946A7bE** |

| Report Version | Twitter | Telegram | Website |
|---|---|---|---|
| **1.1** | **https://x.com/GPTFUSION_TOKEN** | **https://t.me/GPTFUSION_OFFICIALGROUP** | **https://gptfusion.cc/** |

**Scoring:**


## Vulnerability Summary



| 3 | 3 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|
| Total Findings | Unresolved | Resolved | Mitigate | Acknowledge | Decline |

| | | | | |
|---|---|---|---|---|
| ▪ | 0 | Critical | | Critical severity is assigned to security vulnerabilities that pose a severe threat to the smart contract and the entire blockchain ecosystem. |
| ▪ | 0 | High | | High-severity issues should be addressed quickly to reduce the risk of exploitation and protect users' funds and data. |
| ▪ | 0 | Medium | | It's essential to fix medium-severity issues in a reasonable timeframe to enhance the overall security of the smart contract. |
| ▪ | 0 | Low | - | While low-severity issues can be less urgent, it's still advisable to address them to improve the overall security posture of the smart contract. |
| ▪ | 0 | Very Low | | Very Low severity is used for minor security concerns that have minimal impact and are generally of low risk. |
| ▪ | 1 | Informational | 1 unresolved | Used to categorize security findings that do not pose a direct security threat to the smart contract or its users. Instead, these findings provide additional information, recommendations |
| ▪ | 2 | Gas-optimization | 2 unresolved | Suggestions for more efficient algorithms or improvements in gas usage, even if the current code is already secure. |

## Audit Scope:

| File | SHA-1 Hash |
|---|---|
| GPTFUSION_AuditSafe.sol | 4eff893758d8c7900d8e67d17adcef2b20dba4d3 |

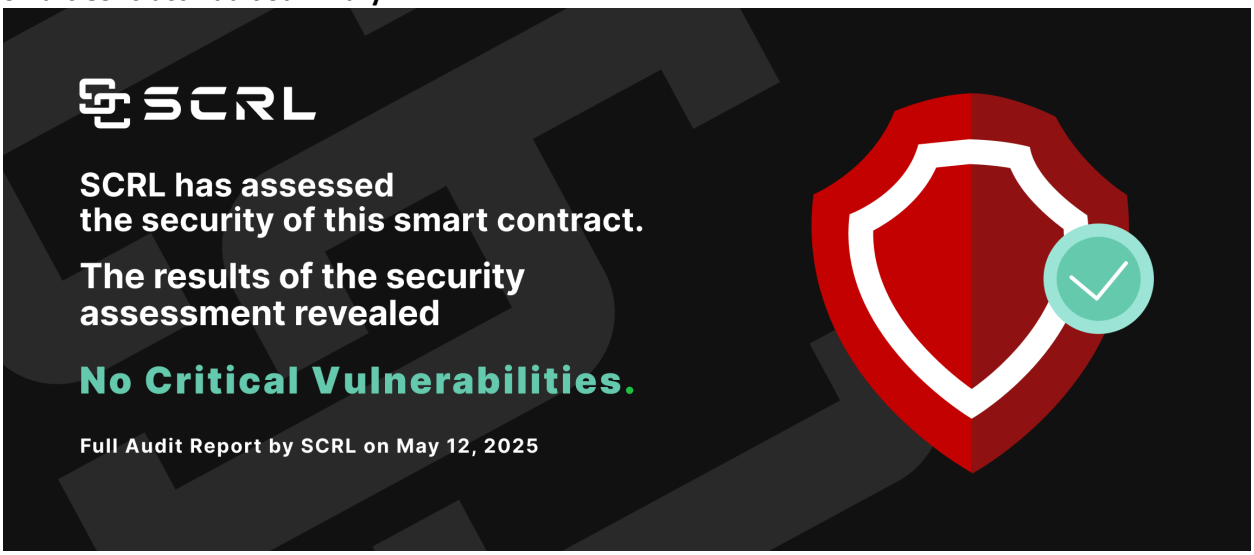
## Audit Version History:

| Version | Date | Description |
|---|---|---|
| 1.0 | Sunday, May 11, 2025 | Preliminary Report |
| 1.1 | Monday, May 12, 2025 | Full Audit Report |

## Audit information:

| Request Date | Audit Date | Re-assessment Date |
|---|---|---|
| Saturday, May 10, 2025 | Monday, May 12, 2025 | - |

## Smart Contract Audit Summary




## Security Assessment Author

| | | |
|---|---|---|
| Auditor: | **Mark K.** | [Security Researcher \| Redteam] |
| | **Kevin N.** | [Security Researcher \| Web3 Dev] |
| | **Yusheng T.** | [Security Researcher \| Incident Response] |
| Document Approval: | **Ronny C.** | CTO & Head of Security Researcher |
| | **Chinnakit J.** | CEO & Founder |

## Digital Sign

ID: 6004EDD6-2FC8-4CB2-8CD9-4F153836D13A
Digitally signed by <contact@scrl.io>
May 12, 2025 03:43 PM +07

## Disclaimer

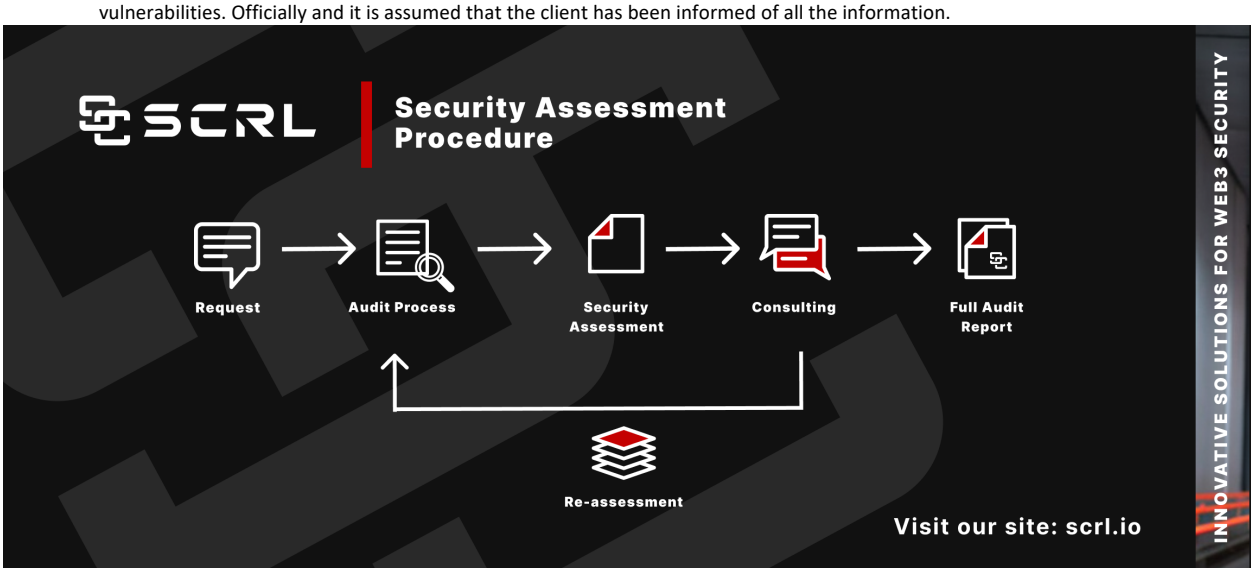
Regarding this security assessment, there are no guarantees about the security of the program instruction received from the client is hereinafter referred to as "**Source code**".
And **SCRL** hereinafter referred to as "**Service Provider**", the **Service Provider** will not be held liable for any legal liability arising from errors in the security assessment. The responsibility will be the responsibility of the **Client**, hereinafter referred to as "**Service User**" and the
**Service User** agrees not to be held liable to the **service provider** in any case. By contract
**Service Provider** to conduct security assessments with integrity with professional ethics, and transparency to deliver security assessments to users The **Service Provider** has the right to postpone the delivery of the security assessment. If the security assessment is delayed whether caused by any reason and is not responsible for any delayed security assessments.
If **the service provider** finds a vulnerability The **service provider** will notify the **service user** via the Preliminary Report, which will be kept confidential for security. The **service provider** disclaims responsibility in the event of any attacks occurring whether before conducting a security assessment. Or happened later All responsibility shall be sole with the **service user**.

**Security Assessment Is Not Financial/Investment Advice Any loss arising from any investment in any project is the responsibility of the investor.**

**SCRL disclaims any liability incurred. Whether it's Rugpull, Abandonment, Soft Rugpull, Exploit, Exit Scam.**

## Security Assessment Procedure

1. **Request** The client must submit a formal request and follow the procedure. By submitting the source code and agreeing to the terms of service.
2. **Audit Process** Check for vulnerabilities and vulnerabilities from source code obtained by experts using formal verification methods, including using powerful tools such as Static Analysis, SWC Registry, Dynamic Security Analysis, Automated Security Tools, CWE, Syntax & Parameter Check with AI ,WAS (Warning Avoidance System a python script tools powered by SCRL).
3. **Security Assessment** Deliver Preliminary Security Assessment to clients to acknowledge the risks and vulnerabilities.
4. **Consulting** Discuss on risks and vulnerabilities encountered by clients to apply to their source code to mitigate risks.
    a. **Re-assessment** Reassess the security when the client implements the source code improvements and if the client is satisfied with the results of the audit. We will proceed to the next step.
5. **Full Audit Report** SCRL provides clients with official security assessment reports informing them of risks and vulnerabilities. Officially and it is assumed that the client has been informed of all the information.



## Risk Rating

Risk rating using this commonly defined: $Risk\ rating\ =\ impact\ *\ confidence$

**Impact** The severity and potential impact of an attacker attack
**Confidence** Ensuring that attackers expose and use this vulnerability

| Confidence<br>Impact [Likelihood] | Low | Medium | High |
|---|---|---|---|
| Low | **Very Low** | **Low** | **Medium** |
| Medium | **Low** | **Medium** | **High** |
| High | **Medium** | **High** | **Critical** |

**Severity** is a risk assessment It is calculated from the Impact and Confidence values using the following calculation methods,
$Risk\ rating\ =\ impact\ *\ confidence$
It is categorized into
**7 categories severity based**


For **Informational** & **Non-class/Optimization/Best-practices will** not be counted as **severity**

## Category

Centralization
**Centralization Risk** is The risk incurred by a sole proprietor, such as the Owner being able to change something without permission

Economics Risk
**Economics Risk** is Risks that may affect the economic mechanism system, such as the ability to increase Mint token

Logical Issue
**Logical Issue** is that can cause errors to core processing, such as any prior operations that cause background processes to crash.

Authorization
**Authorization** is Possible pitfalls from weak coding allows unrelated people to take any action to modify the values.

Mathematical
**Mathematical** Any erroneous arithmetic operations affect the operation of the system or lead to erroneous values.

Naming Conventions
**Naming Conventions** naming variables that may affect code understanding or naming inconsistencies

Security Risk
**Security Risk** of loss or damage if it's no mitigate

Coding Style
**Coding Style** is Tips coding for efficiency performance

Best Practices
**Best Practices** is suggestions for improvement

Optimization
**Optimization** is performance improvement

Gas Optimization
**Gas Optimization** is increase performance to avoid expensive gas

Dead Code
**Dead Code** having unused code This may result in wasted resources and gas fees.

# Table Of Content

## Source Units in Scope

Source Units Analyzed: 1
Source Units in Scope: 1 (**100%**)

| Type | File | Logic Contracts | Interfaces | Lines | nLines | nSLOC | Comment Lines | Complex. Score | Capabilities |
|---|---|---|---|---|---|---|---|---|---|
| | src/GPTFUSION_AuditSafe.sol | 1 | | 67 | 67 | 50 | 2 | 34 | |
| | **Totals** | **1** | | **67** | **67** | **50** | **2** | | |

Legend: [-]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

# Visibility, Mutability, Modifier function testing

## Components

| 📝Contracts | 📚Libraries | 🔍Interfaces | 🎨Abstract |
|---|---|---|---|
| 1 | 0 | 0 | 0 |

## Exposed Functions

**This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.**

| 🌐Public | 💰Payable |
|---|---|
| 4 | 0 |

| External | Internal | Private | Pure | View |
|---|---|---|---|---|
| 0 | 5 | 0 | 0 | 0 |

## StateVariables

| Total | 🌐Public |
|---|---|
| 7 | 7 |

## Capabilities

| Solidity Versions observed | 🧪 Experimental Features | 💰 Can Receive Funds | 🖥️ Uses Assembly | 💣 Has Destroyable Contracts |
|---|---|---|---|---|
| ^0.8.24 | | | | |

| 📩 Transfers ETH | ⚡ Low-Level Calls | 👥 DelegateCall | 🧱 Uses Hash Functions | 🪚 ECRecover | 🌀 New/Create/Create2 |
|---|---|---|---|---|---|
| | | | | | |

| ♻️ TryCatch | Σ Unchecked |
|---|---|
| | |

## Dependencies / External Imports

| Dependency / Import Path | Count |
|---|---|

# Vulnerability Findings

| ID | Vulnerability Detail | Severity | Category | Status |
|---|---|---|---|---|
| CEN-01 | Token Distribution | Informational | Centralization | **Acknowledge** |
| GAS-01 | Use Custom Errors | Gas-optimization | Gas Optimization | **Acknowledge** |
| GAS-02 | Functions guaranteed to revert when called by normal users can be marked `payable` | Gas-optimization | Gas Optimization | **Acknowledge** |

# CEN-01: Token Distribution

| Vulnerability Detail | Severity | Location | Category | Status |
|---|---|---|---|---|
| Token Distribution | Informational | Check on finding | Centralization | **Acknowledge** |

## Finding:

Despite the Token Contract **not containing any malicious functions** that can be executed by the Owner, But, it has been identified that token contracts do not adequately define token distribution, with only one token holder **owning 100% as of Sunday, May 11, 2025**. This presents a significant risk of centralization, and all potential participants must give careful consideration to this matter.

We strongly urge all participants **always promptly to verify token holdings** at
https://bscscan.com/token/0xA4230A63Bd1d5804929c95A3eA0e658e8946A7bE#balances

| Address | Quantity | Percentage |
|---|---|---|
| 0x786b1c125eA66a2A24a041eA2A926541e47831E0 | 100,000,000 | 100% |

***Note: Please note that SCRL is not responsible for any investments. And this document is not an investment recommendation document. If any project is in the pre-sale stage, please participate it at your own risk.
https://chat.scrl.io/hc/scrl-help-center/articles/1717548722-understand-the-risk-of-de_fi-web3

## Recommendation:

**We recommend creating a distribution token & liquidity lock contract to clearly define the distribution ratio for tokens such as Developer, Marketing, Liquidity, and further considerations below.**

In terms of timeframes, there are three categories: short-term, long-term, and permanent.

For short-term solutions, a combination of timelock and multi-signature (2/3 or 3/5) can be used to mitigate risk by delaying sensitive operations and avoiding a single point of failure in key management. This includes implementing a timelock with a reasonable latency, such as 48 hours, for privileged operations; assigning privileged roles to multi-signature wallets to prevent private key compromise; and sharing the timelock contract and multi-signer addresses with the public via a medium/blog link.

For long-term solutions, a combination of timelock and DAO can be used to apply decentralization and transparency to the system. This includes implementing a timelock with a reasonable latency, such as 48 hours, for privileged operations; introducing a DAO/governance/voting module to increase transparency and user involvement; and sharing the timelock contract, multi-signer addresses, and DAO information with the public via a medium/blog link.

Finally, permanent solutions should be implemented to ensure the ongoing security and protection of the system.

## Alleviation: -

## GAS-01: Use Custom Errors

| Vulnerability Detail | Severity | Location | Category | Status |
|---|---|---|---|---|
| Use Custom Errors | - | Check on finding | Gas Optimization | **Acknowledge** |

### Finding:

```
File: GPTFUSION_AuditSafe.sol

20:         require(msg.sender == owner, "Caller is not the owner");

34:         require(_to != address(0), "Invalid address");

35:         require(balanceOf[msg.sender] >= _value, "Insufficient balance");

51:         require(_to != address(0), "Invalid address");

52:         require(balanceOf[_from] >= _value, "Insufficient balance");

53:         require(allowance[_from][msg.sender] >= _value, "Allowance exceeded");

```
```

### Recommendation:

Replace error strings with custom errors to reduce deployment and runtime costs. This approach can lead to significant gas savings in Solidity 0.8.4 and above, as custom errors consume less gas than error strings. More information can be found in the Solidity documentation on custom errors.

### Alleviation:

-
```

## GAS-02: Functions guaranteed to revert when called by normal users can be marked `payable`

| Vulnerability Detail | Severity | Location | Category | Status |
|---|---|---|---|---|
| Functions guaranteed to revert when called by normal users can be marked `payable` | Gas-optimization | Check on finding | Gas Optimization | **Acknowledge** |

### Finding:

```
File: GPTFUSION_AuditSafe.sol

63:     function renounceOwnership() public onlyOwner {

```
```

### Recommendation:

Added the ‘payable’ keyword to the renounceOwnership function while preserving all existing functionality. This change reduces gas costs for legitimate callers without introducing new risks.

### Alleviation:

-
```

## SWC Findings

| ID | Title | Scanning | Result |
|---|---|---|---|
| SWC-100 | Function Default Visibility | Complete | No risk |
| SWC-101 | Integer Overflow and Underflow | Complete | No risk |
| SWC-102 | Outdated Compiler Version | Complete | No risk |
| SWC-103 | Floating Pragma | Complete | No risk |
| SWC-104 | Unchecked Call Return Value | Complete | No risk |
| SWC-105 | Unprotected Ether Withdrawal | Complete | No risk |
| SWC-106 | Unprotected SELFDESTRUCT Instruction | Complete | No risk |
| SWC-107 | Reentrancy | Complete | No risk |
| SWC-108 | State Variable Default Visibility | Complete | No risk |
| SWC-109 | Uninitialized Storage Pointer | Complete | No risk |
| SWC-110 | Assert Violation | Complete | No risk |
| SWC-111 | Use of Deprecated Solidity Functions | Complete | No risk |
| SWC-112 | Delegatecall to Untrusted Callee | Complete | No risk |
| SWC-113 | DoS with Failed Call | Complete | No risk |
| SWC-114 | Transaction Order Dependence | Complete | No risk |

| | | | |
|---|---|---|---|
| SWC-115 | Authorization through tx.origin | Complete | No risk |
| SWC-116 | Block values as a proxy for time | Complete | No risk |
| SWC-117 | Signature Malleability | Complete | No risk |
| SWC-118 | Incorrect Constructor Name | Complete | No risk |
| SWC-119 | Shadowing State Variables | Complete | No risk |
| SWC-120 | Weak Sources of Randomness from Chain Attributes | Complete | No risk |
| SWC-121 | Missing Protection against Signature Replay Attacks | Complete | No risk |
| SWC-122 | Lack of Proper Signature Verification | Complete | No risk |
| SWC-123 | Requirement Violation | Complete | No risk |
| SWC-124 | Write to Arbitrary Storage Location | Complete | No risk |
| SWC-125 | Incorrect Inheritance Order | Complete | No risk |
| SWC-126 | Insufficient Gas Griefing | Complete | No risk |
| SWC-127 | Arbitrary Jump with Function Type Variable | Complete | No risk |
| SWC-128 | DoS With Block Gas Limit | Complete | No risk |
| SWC-129 | Typographical Error | Complete | No risk |
| SWC-130 | Right-To-Left-Override control character (U+202E) | Complete | No risk |

| | | | |
|---|---|---|---|
| SWC-131 | Presence of unused variables | Complete | No risk |
| SWC-132 | Unexpected Ether balance | Complete | No risk |
| SWC-133 | Hash Collisions With Multiple Variable Length Arguments | Complete | No risk |
| SWC-134 | Message call with hardcoded gas amount | Complete | No risk |
| SWC-135 | Code With No Effects | Complete | No risk |
| SWC-136 | Unencrypted Private Data On-Chain | Complete | No risk |

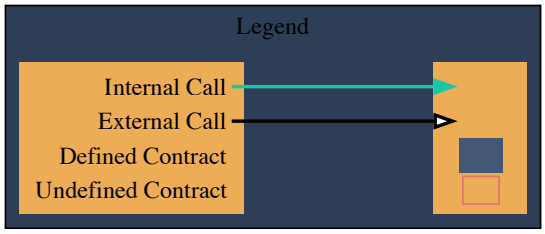
Contracts Description Table

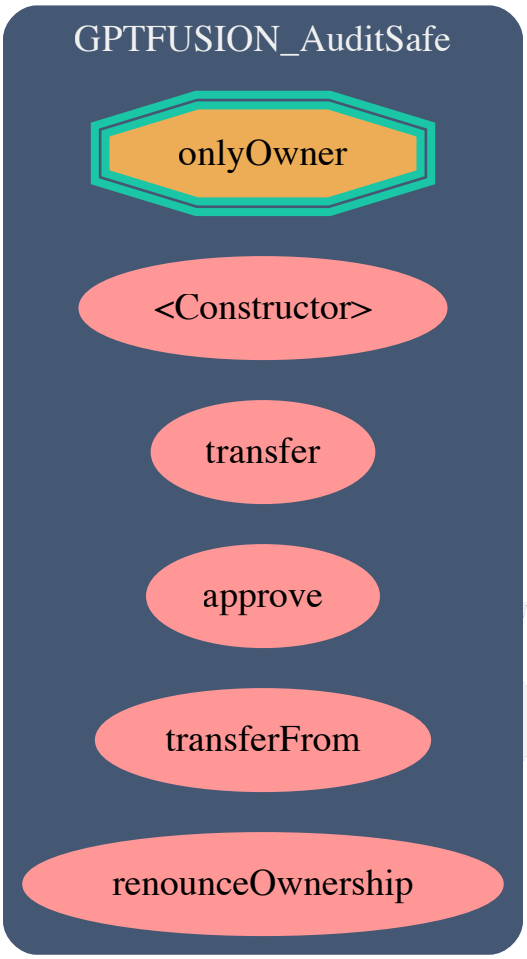
| Contract | Type | Bases | | |
|---|---|---|---|---|
| └ | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| **GPTFUSION_AuditSafe** | Implementation | | | |
| └ | | Public ! | 🛑 | NO ! |
| └ | transfer | Public ! | 🛑 | NO ! |
| └ | approve | Public ! | 🛑 | NO ! |
| └ | transferFrom | Public ! | 🛑 | NO ! |
| └ | renounceOwnership | Public ! | 🛑 | onlyOwner |

Legend

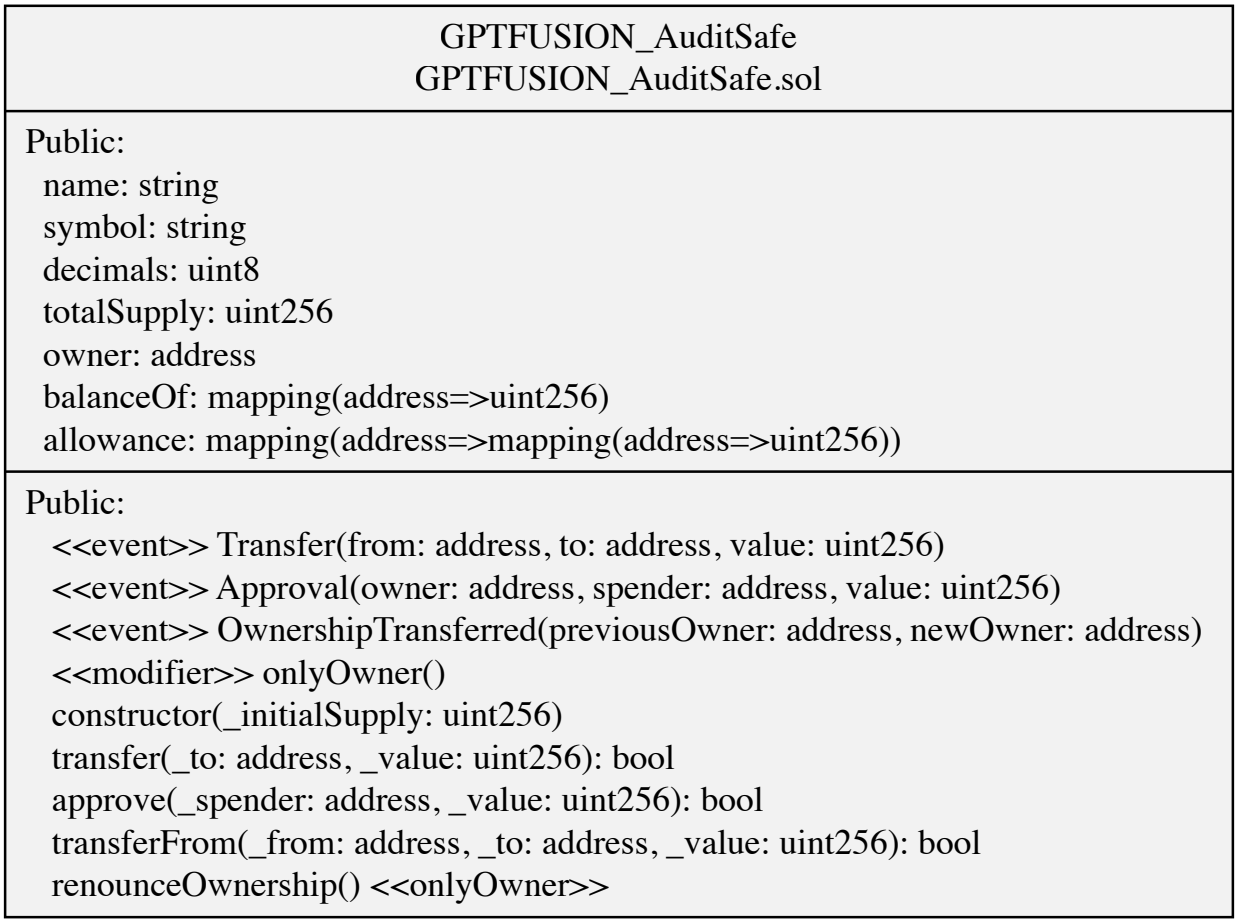
| Symbol | Meaning |
|---|---|
| 🛑 | Function can modify state |
| 💵 | Function is payable |

## Call Graph



## UML Class Diagram



## About SCRL

SCRL (Previously name SECURI LAB) was established in 2020, and its goal is to deliver a security solution for Web3 projects by expert security researchers. To verify the security of smart contracts, they have developed internal tools and KYC solutions for Web3 projects using industry-standard technology. SCRL was created to solve security problems for Web3 projects. They focus on technology for conciseness in security auditing. They have developed Python-based tools for their internal use called WAS and SCRL. Their goal is to drive the crypto industry in Thailand to grow with security protection technology.


**Follow Us On:**

| | |
|---|---|
| Website | https://scrl.io/ |
| Twitter | https://twitter.com/scrl_io |
| Telegram | https://t.me/scrl_io |
| Medium | https://scrl.medium.com/ |