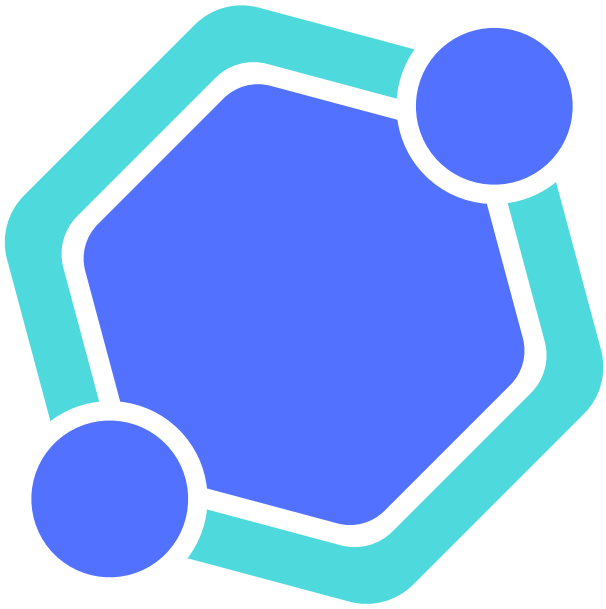
# AUDIT REPORT

April 2023



Audit conducted by

RICARDO PONTES

# Summary

| | |
|---|---|
| **Auditing Firm** | Crypto Hub |
| **Architecture** | Crypto Hub Auditing Standard |
| **Smart Contract Audit Approved By** | Ricardo \| Blockchain Dev at Crypto Hub |
| **Platform** | Solidity |
| **Mandatory Audit Check** | Static, Software & Manual Analysis |
| **Consultation Request Date** | April 11, 2023 |
| **Report Date** | April 11, 2023 |

## Audit Summary

Crypto Hub team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

- ★ Flutterfly smart contract source code has **LOW RISK SEVERITY**.
- ★ Flutterfly has **PASSED** the smart contract audit.

For the detailed understanding of risk severity, source code vulnerability, and functional test, kindly refer to the audit.

✅ Verify the authenticity of this report on Crypto Hub Website: https://www.cryptohub.agency/

# Table Of Contents

# Project Overview

Crypto Hub was consulted by Flutterfly to conduct the smart contract security audit of their solidity source code.

| | |
|---|---|
| Project | Flutterfly |
| Blockchain | Arbitrum One |
| Language | Solidity |
| Contracts | 0xe4f2DD5a7B9E212f59d112d73b74A13A9f765CC5 |
| Website: | https://flutterchain.com/ |

Public logo:

Solidity Source Code On Blockchain (Verified Contract Source Code)

https://arbiscan.io/token/0xe4f2dd5a7b9e212f59d112d73b74a13a9f765cc5#code

Contract Name: StandardToken

Compiler Version: v0.8.4

Optimization Enabled: yes

SHA-1 Hash

Solidity source code is audited at hash

#941010340f93883467220cfef58553a6d66b6e53

# Audit Scope & Methodology

The scope of this report is to audit the above smart contract source code and Crypto Hub has scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Smart Contract Vulnerabilities

- ☐ Re-entrancy
- ☐ Unhandled Exceptions
- ☐ Transaction Order Dependency
- ☐ Integer Overflow
- ☐ Unrestricted Action
- ☐ Incorrect Inheritance Order
- ☐ Typographical Errors
- ☐ Requirement Violation

Source Code Review

- ☐ Ownership Takeover
- ☐ Gas Limit and Loops
- ☐ Deployment Consistency
- ☐ Repository Consistency
- ☐ Data Consistency
- ☐ Token Supply Manipulation

Functional Assessment

- ☐ Access Control and Authorization
- ☐ Operations Trail and Event Generation
- ☐ Assets Manipulation
- ☐ Liquidity Access

# Crypto Hub Audit Standard

The aim of Crypto Hub standard is to analyze the smart contract and identify the vulnerabilities and the hacks in the smart contract. Mentioned are the steps used by ECHELON-1 to assess the smart contract:

1. Solidity smart contract source code reviewal:
   - Review of the specifications, sources, and instructions provided to Crypto Hub to make sure we understand the size, scope, and functionality of the smart contract.
   - Manual review of code, which is the process of reading source code line-by-line to identify potential vulnerabilities.

2. Static, Manual, and Software analysis:
   - Test coverage analysis, which is the process of determining whether the test cases are covering the code and how much code is exercised when we run those test cases.
   - Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts

Automated 3P frameworks used to assess the smart contract vulnerabilities

- Slither
- Consensys MythX
- Consensus Surya
- Open Zeppelin Code Analyzer
- Solidity Code Complier

## Crypto Hub's Risk Classification

Smart contracts are generally designed to manipulate and hold funds denominated in ETH/BNB. This makes them very tempting attack targets, as a successful attack may allow the attacker to directly steal funds from the contract. Below are the typical risk levels of a smart contract: Vulnerable:

A contract is vulnerable if it has been flagged by a static analysis tool as such. As we will see later, this means that some contracts may be vulnerable because of a false-positive.

Exploitable: A contract is exploitable if it is vulnerable and the vulnerability could be exploited by an external attacker. For example, if the "vulnerability" flagged by a tool is in a function which requires to own the contract, it would be vulnerable but not exploitable.

Exploited: A contract is exploited if it received a transaction on the main network which triggered one of its vulnerabilities. Therefore, a contract can be vulnerable or even exploitable without having been exploited.

| Risk severity | Meaning |
|---|---|
| ! Critical | This level of vulnerability could be exploited easily, and can lead to asset loss, data loss, asset manipulation, or data manipulation. They should be fixed right away. |
| ! High | This level vulnerabilities are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to critical risk severity |
| ! Medium | This level of vulnerabilities should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. |
| ! Low | This level of vulnerability can be ignored. They are code style violations, and informational statements in the code. They may not affect the smart contract execution |

# Smart Contract Risk Assessment

## Contract Snapshot

```
contract StandardToken is IERC20, Ownable, BaseToken {
    using SafeMath for uint256;

    uint256 public constant VERSION = 1;

    mapping(address => uint256) private _balances;
    mapping(address => mapping(address => uint256)) private _allowances;

    string private _name;
    string private _symbol;
    uint8 private _decimals;
    uint256 private _totalSupply;

    constructor(
        string memory name_,
        string memory symbol_,
        uint8 decimals_,
        uint256 totalSupply_,
        address serviceFeeReceiver_,
        uint256 serviceFee_
    ) payable {
        _name = name_;
        _symbol = symbol_;
        _decimals = decimals_;
        _mint(owner(), totalSupply_);

        emit TokenCreated(owner(), address(this), TokenType.standard, VERSION);

        payable(serviceFeeReceiver_).transfer(serviceFee_);
    }
```

# Static / Quick Analysis

## Contract Security

### Contract source code verified

This token contract is open source. You can check the contract code for details. Unsourced token contracts are likely to have malicious functions to defraud their users of their assets.

### No proxy

There is no proxy in the contract. The proxy contract means contract owner can modifiy the function of the token and possibly effect the price.

### No mint function

Mint function is transparent or non-existent. Hidden mint functions may increase the amount of tokens in circulation and effect the price of the token.

### No function found that retrieves ownership

If this function exists, it is possible for the project owner to regain ownership even after relinquishing it

### Owner can't change balance

The contract owner is not found to have the authority to modify the balance of tokens at other addresses.

### No hidden owner

No hidden owner address was found for the token. For contract with a hidden owner, developer can still manipulate the contract even if the ownership has been abandoned.

### This token can not self destruct

No self-destruct function found. If this function exists and is triggered, the contract will be destroyed, all functions will be unavailable, and all related assets will be erased.

### No external call risk found

External calls would cause this token contract to be highly dependent on other contracts, which may be a potential risk.

## Honeypot Risk

### This does not appear to be a honeypot.

We are not aware of any malicious code.

### No codes found to suspend trading.

If a suspendable code is included, the token maybe neither be bought nor sold (honeypot risk).

### No trading cooldown function

The token contract has no trading cooldown function. If there is a trading cooldown function, the user will not be able to sell the token within a certain time or block after buying.

### No anti_whale(Unlimited number of transactions)

There is no limit to the number of token transactions. The number of scam token transactions may be limited (honeypot risk).

### Anti whale can not be modified

The maximum trading amount or maximum position can not be modified.

### Tax cannot be modified

The contract owner may not contain the authority to modify the transaction tax. If the transaction tax is increased to more than 49%, the tokens will not be able to be traded (honeypot risk).

### No blacklist

The blacklist function is not included. If there is a blacklist, some addresses may not be able to trade normally (honeypot risk).

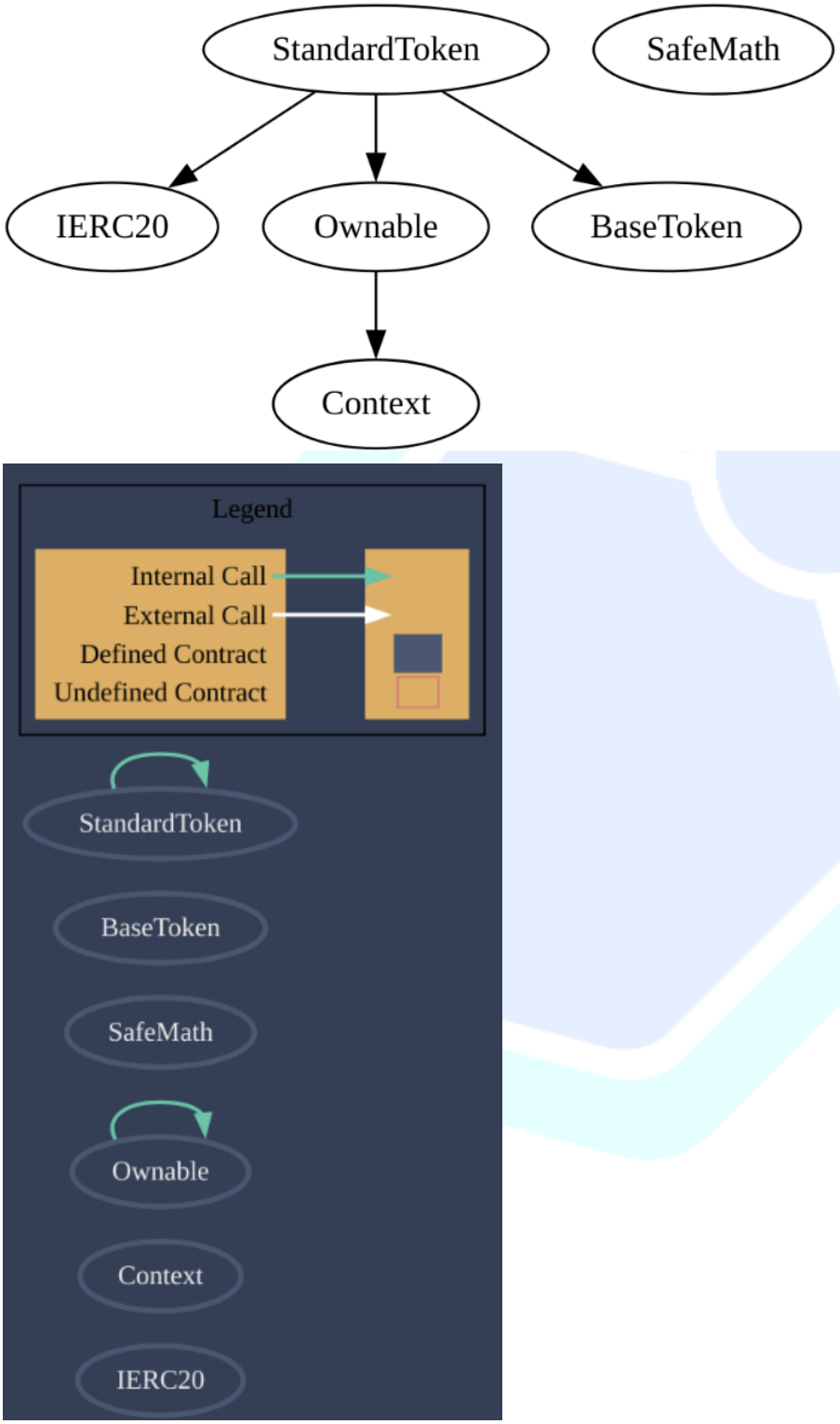
### No whitelist

The whitelist function is not included. If there is a whitelist, some addresses may not be able to trade normally (honeypot risk).

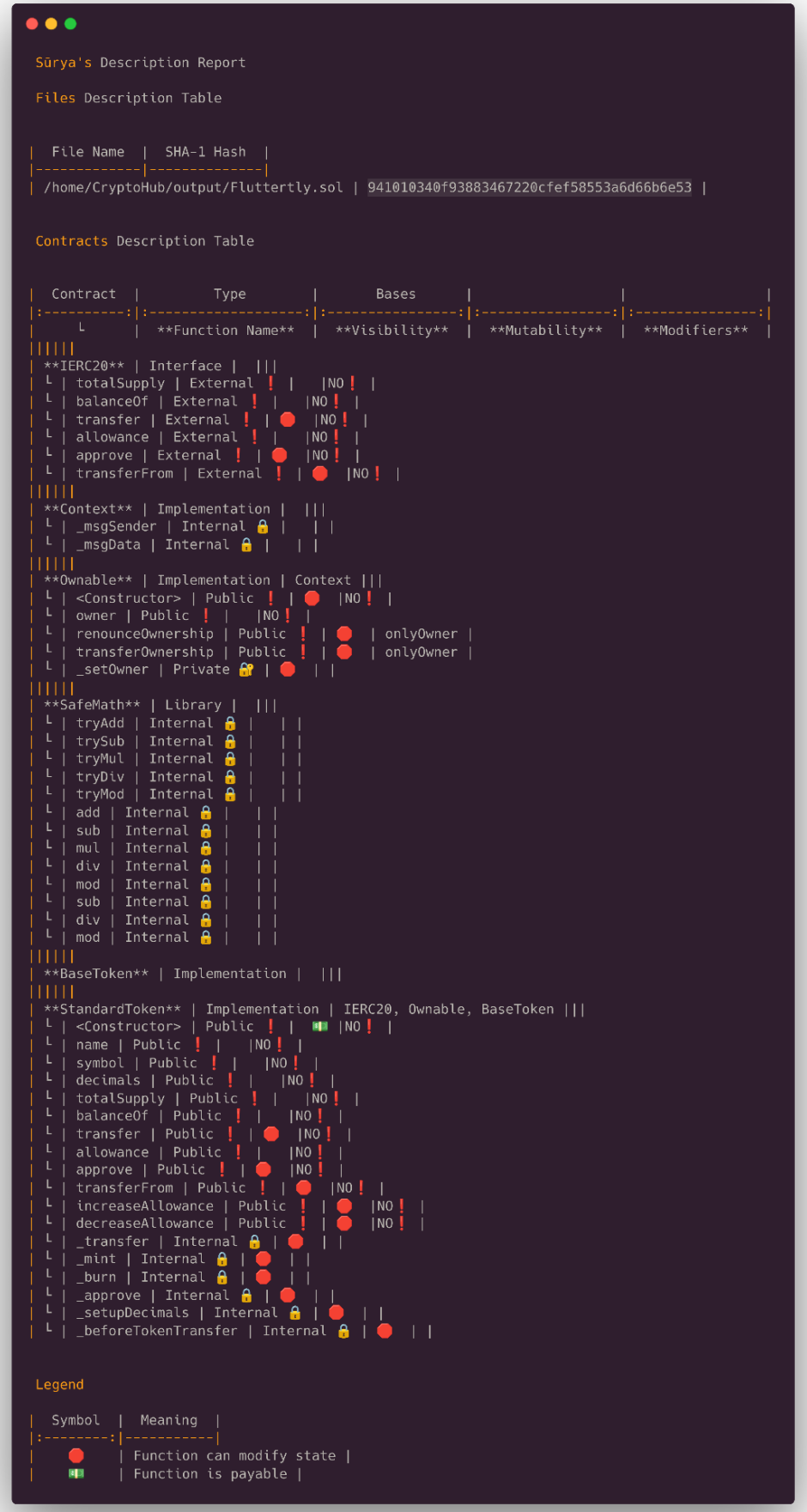
### No tax changes found for personal addresses

No tax changes were found for every assigned address. If it exists, the contract owner may set a very outrageous tax rate for assigned address to block it from trading.

## Software Analysis



```
Sighash   |   Function Signature
========================
39509351  =>  increaseAllowance(address,uint256)
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
fc201122  =>  _setOwner(address)
884557bf  =>  tryAdd(uint256,uint256)
a29962b1  =>  trySub(uint256,uint256)
6281efa4  =>  tryMul(uint256,uint256)
736ecb18  =>  tryDiv(uint256,uint256)
38dc0867  =>  tryMod(uint256,uint256)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
f43f523a  =>  mod(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
b745d336  =>  div(uint256,uint256,string)
71af23e8  =>  mod(uint256,uint256,string)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
a457c2d7  =>  decreaseAllowance(address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
4e6ec247  =>  _mint(address,uint256)
6161eb18  =>  _burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
61e9edb2  =>  _setupDecimals(uint8)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
```



Sūrya's Description Report

Files Description Table

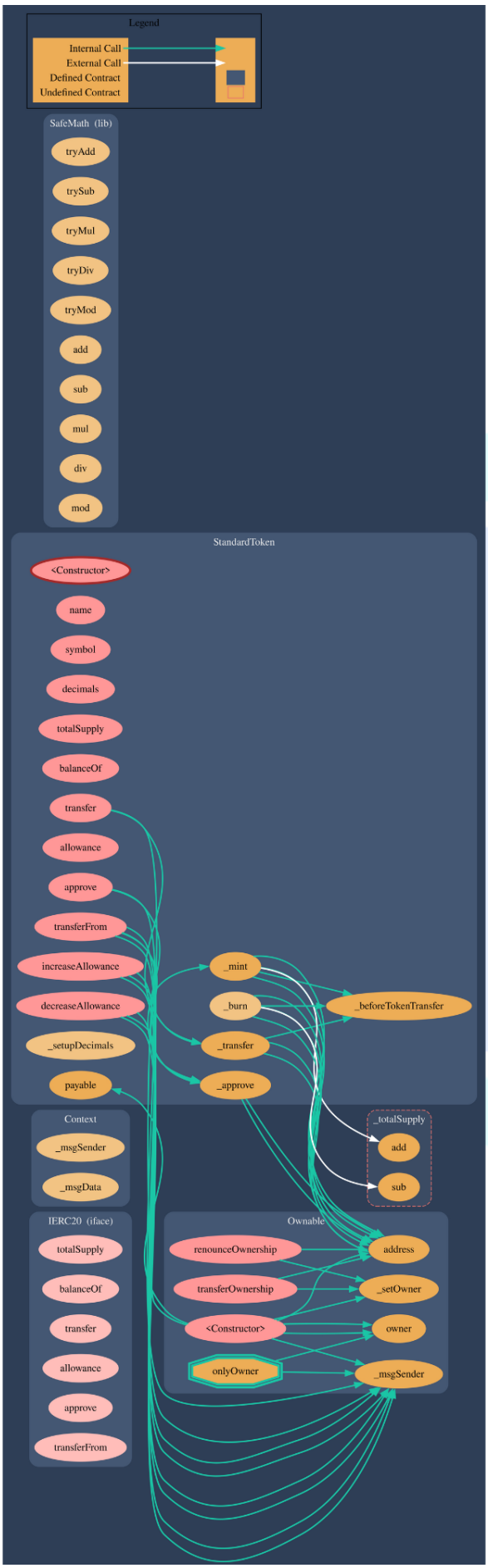
| File Name | SHA-1 Hash |
|---|---|
| /home/CryptoHub/output/Fluttertly.sol | 941010340f93883467220cfef58553a6d66b6e53 |

Contracts Description Table

| Contract | Type | Bases | | |
|---|---|---|---|---|
| └ | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| **IERC20** | Interface | ||| |
| └ | totalSupply | External ❗ | | NO ❗ |
| └ | balanceOf | External ❗ | | NO ❗ |
| └ | transfer | External ❗ | 🛑 | NO ❗ |
| └ | allowance | External ❗ | | NO ❗ |
| └ | approve | External ❗ | 🛑 | NO ❗ |
| └ | transferFrom | External ❗ | 🛑 | NO ❗ |
| **Context** | Implementation | ||| |
| └ | _msgSender | Internal 🔒 | | |
| └ | _msgData | Internal 🔒 | | |
| **Ownable** | Implementation | Context ||| |
| └ | <Constructor> | Public ❗ | 🛑 | NO ❗ |
| └ | owner | Public ❗ | | NO ❗ |
| └ | renounceOwnership | Public ❗ | 🛑 | onlyOwner |
| └ | transferOwnership | Public ❗ | 🛑 | onlyOwner |
| └ | _setOwner | Private 🔐 | 🛑 | |
| **SafeMath** | Library | ||| |
| └ | tryAdd | Internal 🔒 | | |
| └ | trySub | Internal 🔒 | | |
| └ | tryMul | Internal 🔒 | | |
| └ | tryDiv | Internal 🔒 | | |
| └ | tryMod | Internal 🔒 | | |
| └ | add | Internal 🔒 | | |
| └ | sub | Internal 🔒 | | |
| └ | mul | Internal 🔒 | | |
| └ | div | Internal 🔒 | | |
| └ | mod | Internal 🔒 | | |
| └ | sub | Internal 🔒 | | |
| └ | div | Internal 🔒 | | |
| └ | mod | Internal 🔒 | | |
| **BaseToken** | Implementation | ||| |
| **StandardToken** | Implementation | IERC20, Ownable, BaseToken ||| |
| └ | <Constructor> | Public ❗ | 💵 | NO ❗ |
| └ | name | Public ❗ | | NO ❗ |
| └ | symbol | Public ❗ | | NO ❗ |
| └ | decimals | Public ❗ | | NO ❗ |
| └ | totalSupply | Public ❗ | | NO ❗ |
| └ | balanceOf | Public ❗ | | NO ❗ |
| └ | transfer | Public ❗ | 🛑 | NO ❗ |
| └ | allowance | Public ❗ | | NO ❗ |
| └ | approve | Public ❗ | 🛑 | NO ❗ |
| └ | transferFrom | Public ❗ | 🛑 | NO ❗ |
| └ | increaseAllowance | Public ❗ | 🛑 | NO ❗ |
| └ | decreaseAllowance | Public ❗ | 🛑 | NO ❗ |
| └ | _transfer | Internal 🔒 | 🛑 | |
| └ | _mint | Internal 🔒 | 🛑 | |
| └ | _burn | Internal 🔒 | 🛑 | |
| └ | _approve | Internal 🔒 | 🛑 | |
| └ | _setupDecimals | Internal 🔒 | 🛑 | |
| └ | _beforeTokenTransfer | Internal 🔒 | 🛑 | |

Legend

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


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod
StandardToken
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
increaseAllowance
decreaseAllowance
_setupDecimals
payable
_mint
_burn
_transfer
_approve
_beforeTokenTransfer
Context
_msgSender
_msgData
_totalSupply
add
sub
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Ownable
renounceOwnership
transferOwnership
<Constructor>
onlyOwner
address
_setOwner
owner
_msgSender

## SWC Attacks

The following table contains an overview of the SWC registry. Each row consists of an SWC identifier (ID), weakness title, CWE parent and list of related code samples.

The auditor used a MythX tool, A static analyzer that parses the Soldity AST, a symbolic analyzer that detects possible vulnerable states, and a greybox fuzzer that detects vulnerable execution paths.

| ID | Description | Status |
|---|---|---|
| **SWC - 100** | Function Default Visibility | ✔️ **Passed** |
| **SWC - 101** | Integer Overflow and Underflow | ✔️ **Passed** |
| **SWC - 102** | Outdated Compiler Version | ✔️ **Passed** |
| **SWC - 103** | Floating Pragma | ✔️ **Passed** |
| **SWC - 104** | Unchecked Call Return Value | ✔️ **Passed** |
| **SWC - 105** | Unprotected Ether Withdrawal | ✔️ **Passed** |
| **SWC - 106** | Unprotected SELFDESTRUCT Instruction | ✔️ **Passed** |
| **SWC - 107** | Reentrancy Passed | ✔️ **Passed** |
| **SWC - 108** | State Variable Default Visibility | ✔️ **Passed** |
| **SWC - 109** | Uninitialized Storage Pointer | ✔️ **Passed** |
| **SWC - 110** | Assert Violation Passed | ✔️ **Passed** |
| **SWC - 111** | Use of Deprecated Solidity Functions | ✔️ **Passed** |
| **SWC - 112** | Delegatecall to Untrusted Callee | ✔️ **Passed** |
| **SWC - 113** | DoS with Failed Call | ✔️ **Passed** |
| **SWC - 114** | Transaction Order Dependence | ✔️ **Passed** |
| **SWC - 115** | Authorization through tx.origin | ✔️ **Passed** |
| **SWC - 116** | Block values as a proxy for time | ✔️ **Passed** |
| **SWC - 117** | Signature Malleability | ✔️ **Passed** |



| ID | Description | Status |
|---|---|---|
| **SWC - 118** | Incorrect Constructor Name | ✔ **Passed** |

| | | |
|---|---|---|
| **SWC - 119** | Shadowing State Variables | ✔ **Passed** |
| **SWC - 120** | Weak Sources of Randomness from Chain Attributes | ✔ **Passed** |
| **SWC - 121** | Missing Protection against Signature Replay Attacks | ✔ **Passed** |
| **SWC - 122** | Lack of Proper Signature Verification | ✔ **Passed** |
| **SWC - 123** | Requirement Violation Passed | ✔ **Passed** |
| **SWC - 124** | Write to Arbitrary Storage Location | ✔ **Passed** |
| **SWC - 125** | Incorrect Inheritance Order Passed | ✔ **Passed** |
| **SWC - 126** | Insufficient Gas Griefing | ✔ **Passed** |
| **SWC - 127** | Arbitrary Jump with Function Type Variable | ✔ **Passed** |
| **SWC - 128** | DoS With Block Gas Limit | ✔ **Passed** |
| **SWC - 129** | Typographical Error | ✔ **Passed** |
| **SWC - 130** | Right-To-Left-Override control character (U+202E) | ✔ **Passed** |
| **SWC - 131** | Presence of unused variables | ✔ **Passed** |
| **SWC - 132** | Unexpected Ether balance | ✔ **Passed** |
| **SWC - 133** | Hash Collisions With Multiple Variable Arguments | ✔ **Passed** |

| | | |
|---|---|---|
| **SWC - 134** | Message call with hardcoded gas amount | ✔ **Passed** |
| **SWC - 135** | Code With No Effects | ✔ **Passed** |
| **SWC - 136** | Unencrypted Private Data On-Chain | ✔ **Passed** |



## Manual Analysis

Extremely Basic Token with no Instances Detected..

## Risk Status

| Risk severity | Meaning |
|---|---|
| ! Critical | None critical severity issues identified |
| ! High | None high severity issues identified |
| ! Medium | None medium severity issues identified |
| ! Low | None low severity issues identified |
| Verified | 7 functions and instances verified and checked |
| Safety Score | 96 out of 100 |

# Report Summary

Crypto Hub team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks.

Flutterfly  smart contract source code has **LOW RISK SEVERITY.**
Flutterfly has **PASSED** the smart contract audit.

**Note for stakeholders:**

Be aware that active smart contract owner privileges constitute an elevated impact on smart contract's safety and security.
Make sure that the project team's KYC/identity is verified by an independent firm, e.g., Crypto Hub.
Always check if the contract's liquidity is locked. A longer liquidity lock plays an important role in the project's longevity. It is recommended to have multiple liquidity providers.
Examine the unlocked token supply in the owner, developer, or team's private wallets. Understand the project's tokenomics, and make sure the tokens outside of the LP Pair are vested or locked for a longer period of time.
Ensure that the project's official website is hosted on a trusted platform, and is using an active SSL certificate. The website's domain should be registered for a longer period of time.

# Audit & KYC Certificates

**We hereby certificate Flutterfly token smart contract as an audited project under the Crypto Hub enterprise umbrella. And to represent it as such we issued the following certificate:**

# Legal Advisory

## *Important Disclaimer*

Crypto Hub provides contract auditing and project verification services for blockchain projects. The purpose of the audit is to analyze the on-chain smart contract source code, and to provide a basic overview of the project. This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purposes without Crypto Hub prior written consent.

Crypto Hub provides the easy-to-understand assessment of the project, and the smart contract (otherwise known as the source code). The audit makes no statements or warranties on the security of the code. It also cannot be considered as an adequate assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have used all the data at our disposal to provide the transparent analysis, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Be aware that smart contracts deployed on a blockchain aren't resistant from external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security. Therefore, Crypto Hub does not guarantee the explicit security of the audited smart contract.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

This report should not be considered as an endorsement or disapproval of any project or team. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. Do conduct your own due diligence and consult your financial advisor before making any investment decisions.

## About Crypto Hub

Crypto Hub provides intelligent blockchain solutions. Crypto Hub is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. Crypto Hub's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy-to-use.

Crypto Hub is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 3+ core team members, and 6+ casual contributors. Crypto Hub provides manual, static, and automatic smart contract analysis, to ensure that the project is checked against known attacks and potential vulnerabilities.