Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

# SMART CONTRACT

## Security Audit Report

Project: Statera Token
Website: Stateratoken.com
Platform: Binance Smart Chain
Language: Solidity
Date: October 29th, 2021

# Table of contents

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

## Introduction

EtherAuthority was contracted by the Statera team to perform the Security audit of the Statera Token smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on October 29th, 2021.

**The purpose of this audit was to address the following:**

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

## Project Background

Statera (STA) token is a smart contract running on Binance Smart Chain. Every token transfer will burn 1% of the token. This smart contract also has cross-chain bridge functions, which means any other blockchain assets can be pegged with this tokenomics.

## Audit scope

| Name | Code Review and Security Analysis Report for Statera Token Smart Contract |
|---|---|
| **Platform** | **BSC / Solidity** |
| **Smart Contract Code** | https://github.com/rubinacci/anyswap-v1-core/blob/master/contracts/Statera-AnyswapV5ERC20.sol |
| **Github Commit** | 75e3f643ad4206382484534f22781dd770b6bda7 |
| **Audit Date** | October 29th, 2021 |

# Claimed Smart Contract Features

| Claimed Feature Detail | Our Observation |
|---|---|
| **Tokenomics:**<br>● Token Name: Statera Token<br>● Token Symbol: STA<br>● Decimals: 18<br>● Max Supply Limit: No Limit<br>● Token Burn on Every Transfer: 1% | **YES, This is valid.** |
| **Token Bridge**<br>● Owner can mint tokens<br>● Owner can burn tokens | **YES, This is valid.**<br>**Since this process contains centralized components, the owner must mint/burn tokens according to receiving/sending cross-chain assets.**<br>**And the private key of the owner's wallet must be handled securely.** |

# Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. This token contract does contain owner control, which does not make it fully decentralized.


We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

**We found 0 critical, 0 high, 0 medium and 1 low and some very low level issues. These issues are not critical ones, so it's good to go for the production.**

**Investors Advice:** Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

## Technical Quick Stats

| Main Category | Subcategory | Result |
|---|---|---|
| Contract Programming | Solidity version not specified | Passed |
| | Solidity version too old | Passed |
| | Integer overflow/underflow | Passed |
| | Function input parameters lack of check | Passed |
| | Function input parameters check bypass | Passed |
| | Function access control lacks management | Passed |
| | Critical operation lacks event log | Very Low Severity |
| | Human/contract checks bypass | Passed |
| | Random number generation/use vulnerability | N/A |
| | Fallback function misuse | Passed |
| | Race condition | Passed |
| | Logical vulnerability | Passed |
| | Features claimed | Passed |
| | Other programming issues | Passed |
| Code Specification | Function visibility not explicitly declared | Passed |
| | Var. storage location not explicitly declared | Passed |
| | Use keywords/functions to be deprecated | Passed |
| | Unused code | Passed |
| Gas Optimization | “Out of Gas” Issue | Passed |
| | High consumption ‘for/while’ loop | Passed |
| | High consumption ‘storage’ storage | Passed |
| | Assert() misuse | Passed |
| Business Risk | The maximum limit for mintage not set | Passed |
| | “Short Address” Attack | Passed |
| | “Double Spend” Attack | Passed |

**Overall Audit Result: PASSED**

## Code Quality

This audit scope has 1 smart contract file. Smart contracts contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in Statera Token are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Statera Token.

The Statera Token team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Some code parts are **not** well commented on smart contracts.

## Documentation

We were given a Statera Token smart contracts code in the form of a github link.The commit hash of that code is mentioned above in the table.

As mentioned above, some code parts are not well commented. But most parts are well commented, so it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

## Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

## AS-IS overview

| Sl. | Functions | Type | Observation | Conclusion |
|---|---|---|---|---|
| 1 | constructor | write | Passed | No Issue |
| 2 | onlyAuth | modifier | Passed | No Issue |
| 3 | onlyVault | modifier | Passed | No Issue |
| 4 | owner | read | Passed | No Issue |
| 5 | mpc | read | Passed | No Issue |
| 6 | setVaultOnly | external | access only Vault | No Issue |
| 7 | initVault | external | access only Vault | No Issue |
| 8 | setMinter | external | access only Vault | No Issue |
| 9 | setVault | external | access only Vault | No Issue |
| 10 | applyVault | external | access only Vault | No Issue |
| 11 | applyMinter | external | access only Vault | No Issue |
| 12 | revokeMinter | external | access only Vault | No Issue |
| 13 | getAllMinters | external | Passed | No Issue |
| 14 | changeVault | external | access only Vault | No Issue |
| 15 | changeMPCOwner | write | access only Vault | No Issue |
| 16 | mint | external | access only Auth | No Issue |
| 17 | burn | external | access only Auth | No Issue |
| 18 | Swapin | write | access only Auth | No Issue |
| 19 | Swapout | write | Passed | No Issue |
| 20 | totalSupply | external | Passed | No Issue |
| 21 | burnedSupply | external | Passed | No Issue |
| 22 | depositWithPermit | external | Passed | No Issue |
| 23 | depositWithTransferPermit | external | Passed | No Issue |
| 24 | deposit | external | Passed | No Issue |
| 25 | depositVault | external | access only Vault | No Issue |
| 26 | _deposit | internal | Passed | No Issue |
| 27 | withdraw | external | Passed | No Issue |
| 28 | withdrawVault | external | access only Vault | No Issue |
| 29 | _withdraw | internal | Passed | No Issue |
| 30 | _mint | internal | Passed | No Issue |
| 31 | _burn | internal | Passed | No Issue |
| 32 | _removeFromSupply | internal | Passed | No Issue |
| 33 | approve | external | Passed | No Issue |
| 34 | approveAndCall | external | Passed | No Issue |
| 35 | permit | external | Handle sig securely | No Issue |
| 36 | cut | write | Passed | No Issue |
| 37 | transferWithPermit | external | Passed | No Issue |
| 38 | verifyEIP712 | internal | Passed | No Issue |
| 39 | verifyPersonalSign | internal | Passed | No Issue |
| 40 | prefixed | internal | Passed | No Issue |
| 41 | transfer | external | Passed | No Issue |
| 42 | transferFrom | external | Passed | No Issue |
| 43 | transferAndCall | external | Passed | No Issue |

## Severity Definitions

| Risk Level | Description |
|---|---|
| Critical | Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc. |
| High | High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial |
| Medium | Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose |
| Low | Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution |
| Lowest / Code Style / Best Practice | Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored. |

# Audit Findings

## Critical Severity

No Critical severity vulnerabilities were found.

## High Severity

No High severity vulnerabilities were found.

## Medium

No Medium severity vulnerabilities were found.

## Low

(1) Critical functions lack event logs: It is recommended to fire appropriate event logs when significant state changes are happening. We recommend emitting events for the following functions.

- setVaultOnly
- applyVault
- applyMinter
- revokeMinter

## Very Low / Informational / Best practices:

(1) Please use the latest compiler version when deploying contract

```
pragma solidity ^0.8.2;
```

This is not a severe issue, but we suggest using the latest compiler version at the time of contract deployment, which is 0.8.9 at the time of this audit. Using the latest compiler version is always recommended which prevents any compiler level issues.

(2) Custom error messages missing:

Some “require” conditions in the code miss custom error messages. We suggest putting custom error messages in “require” conditions in the following functions, which will be helpful in debugging the code for the failed transactions.

- initVault
- applyVault
- applyMinter
- constructor
- _deposit
- permit
- transferWithPermit
- transfer
- transferFrom
- transferAndCall

(3) All functions which are not called internally, must be declared as external. It is more efficient as sometimes it saves some gas.

https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices

(4) Please make variables constant

```
string public name;
string public symbol;
```

These variable's values will be unchanged. So, please make it constant. It will save some gas. Just put a constant keyword.

## Centralization

These smart contracts have some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- setVaultOnly: The Vault owner can set vault enabled/disabled.
- initVault: The Vault owner can initialize vault value.
- setMinter: The Vault owner can set the minter.
- setVault: The Vault owner can set the vault.
- applyVault: The Vault owner can apply the vault.
- applyMinter: The Vault owner can apply the minter.
- revokeMinter: The Vault owner can revoke minter as an emergency function.
- changeVault: The Vault owner can change vault value.
- changeMPCOwner: The Vault owner can update the MPC owner.
- mint: The Auth wallet can mint tokens.
- burn: The Auth wallet can burn tokens.
- Swapin: The Auth wallet can trigger the SwapIn function by minting new tokens.
- depositVault: The Vault owner can deposit vault value.
- withdrawVault: The Vault owner can withdraw vault value.

# Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts, but they are not critical ones. So, **it's good to go to production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured".**

# Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

**Manual Code Review:**

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

**Vulnerability Analysis:**

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

**Documenting Results:**

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

**Suggested Solutions:**

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

# Disclaimers

## EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

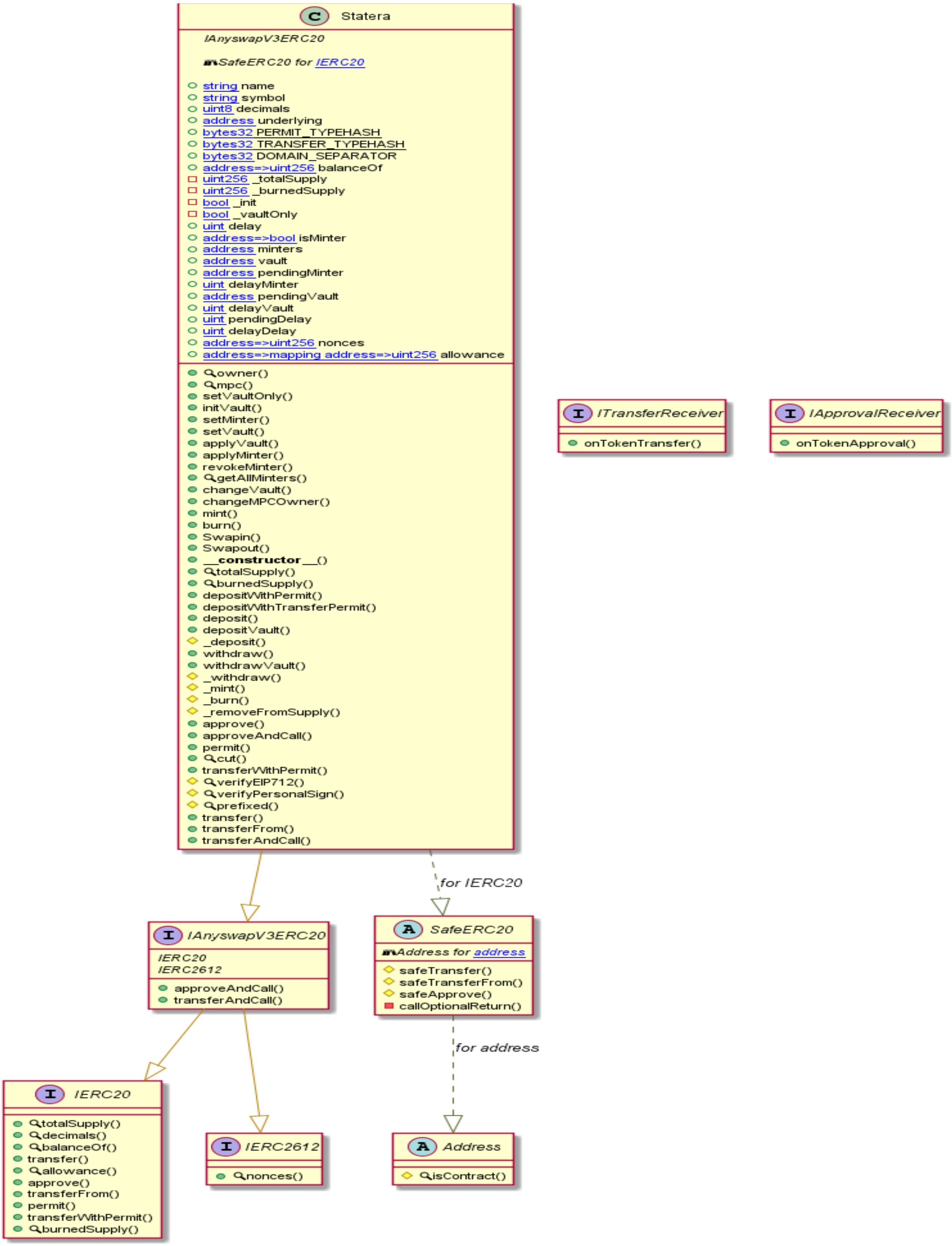
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

## Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

## Appendix

### Code Flow Diagram - Statera Token



# Slither Results Log

## Slither log >> Statera-AnyswapV5ERC20.sol

```
INFO:Detectors:
Statera.depositWithTransferPermit(address,uint256,uint256,uint8,bytes32,bytes32,address) (Statera-AnyswapV5ERC20.sol#329-332) ignores ret
urn value by IERC20(underlying).transferWithPermit(target,address(this),value,deadline,v,r,s) (Statera-AnyswapV5ERC20.sol#330)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Statera.initVault(address)._vault (Statera-AnyswapV5ERC20.sol#188) lacks a zero-check on :
                - vault = _vault (Statera-AnyswapV5ERC20.sol#190)
                - pendingVault = _vault (Statera-AnyswapV5ERC20.sol#191)
Statera.setMinter(address)._auth (Statera-AnyswapV5ERC20.sol#198) lacks a zero-check on :
                - pendingMinter = _auth (Statera-AnyswapV5ERC20.sol#199)
Statera.setVault(address)._vault (Statera-AnyswapV5ERC20.sol#203) lacks a zero-check on :
                - pendingVault = _vault (Statera-AnyswapV5ERC20.sol#204)
Statera.constructor(string,string,uint8,address,address)._underlying (Statera-AnyswapV5ERC20.sol#283) lacks a zero-check on :
                - underlying = _underlying (Statera-AnyswapV5ERC20.sol#287)
Statera.constructor(string,string,uint8,address,address)._vault (Statera-AnyswapV5ERC20.sol#283) lacks a zero-check on :
                - vault = _vault (Statera-AnyswapV5ERC20.sol#298)
                - pendingVault = _vault (Statera-AnyswapV5ERC20.sol#299)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Statera.deposit() (Statera-AnyswapV5ERC20.sol#334-338):
        External calls:
        - IERC20(underlying).safeTransferFrom(msg.sender,address(this),_amount) (Statera-AnyswapV5ERC20.sol#336)
        State variables written after the call(s):
        - _deposit(_amount,msg.sender) (Statera-AnyswapV5ERC20.sol#337)
                - _totalSupply += amount (Statera-AnyswapV5ERC20.sol#394)
        - _deposit(_amount,msg.sender) (Statera-AnyswapV5ERC20.sol#337)
                - balanceOf[account] += amount (Statera-AnyswapV5ERC20.sol#395)
Reentrancy in Statera.deposit(uint256) (Statera-AnyswapV5ERC20.sol#340-343):
        External calls:
        - IERC20(underlying).safeTransferFrom(msg.sender,address(this),amount) (Statera-AnyswapV5ERC20.sol#341)
        State variables written after the call(s):
        - _deposit(amount,msg.sender) (Statera-AnyswapV5ERC20.sol#342)
                - _totalSupply += amount (Statera-AnyswapV5ERC20.sol#394)
        - _deposit(amount,msg.sender) (Statera-AnyswapV5ERC20.sol#342)
                - balanceOf[account] += amount (Statera-AnyswapV5ERC20.sol#395)
Reentrancy in Statera.deposit(uint256,address) (Statera-AnyswapV5ERC20.sol#345-348):
        External calls:
        - IERC20(underlying).safeTransferFrom(msg.sender,address(this),amount) (Statera-AnyswapV5ERC20.sol#346)
```

```
        State variables written after the call(s):
        - _deposit(amount,to) (Statera-AnyswapV5ERC20.sol#347)
                - _totalSupply += amount (Statera-AnyswapV5ERC20.sol#394)
        - _deposit(amount,to) (Statera-AnyswapV5ERC20.sol#347)
                - balanceOf[account] += amount (Statera-AnyswapV5ERC20.sol#395)
Reentrancy in Statera.depositWithPermit(address,uint256,uint256,uint8,bytes32,bytes32,address) (Statera-AnyswapV5ERC20.sol#323-327):
        External calls:
        - IERC20(underlying).permit(target,address(this),value,deadline,v,r,s) (Statera-AnyswapV5ERC20.sol#324)
        - IERC20(underlying).safeTransferFrom(target,address(this),value) (Statera-AnyswapV5ERC20.sol#325)
        State variables written after the call(s):
        - _deposit(value,to) (Statera-AnyswapV5ERC20.sol#326)
                - _totalSupply += amount (Statera-AnyswapV5ERC20.sol#394)
        - _deposit(value,to) (Statera-AnyswapV5ERC20.sol#326)
                - balanceOf[account] += amount (Statera-AnyswapV5ERC20.sol#395)
Reentrancy in Statera.depositWithTransferPermit(address,uint256,uint256,uint8,bytes32,bytes32,address) (Statera-AnyswapV5ERC20.sol#329-33
2):
        External calls:
        - IERC20(underlying).transferWithPermit(target,address(this),value,deadline,v,r,s) (Statera-AnyswapV5ERC20.sol#330)
        State variables written after the call(s):
        - _deposit(value,to) (Statera-AnyswapV5ERC20.sol#331)
                - _totalSupply += amount (Statera-AnyswapV5ERC20.sol#394)
        - _deposit(value,to) (Statera-AnyswapV5ERC20.sol#331)
                - balanceOf[account] += amount (Statera-AnyswapV5ERC20.sol#395)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Statera.deposit() (Statera-AnyswapV5ERC20.sol#334-338):
        External calls:
        - IERC20(underlying).safeTransferFrom(msg.sender,address(this),_amount) (Statera-AnyswapV5ERC20.sol#336)
        Event emitted after the call(s):
        - Transfer(address(0),account,amount) (Statera-AnyswapV5ERC20.sol#396)
                - _deposit(_amount,msg.sender) (Statera-AnyswapV5ERC20.sol#337)
Reentrancy in Statera.deposit(uint256) (Statera-AnyswapV5ERC20.sol#340-343):
        External calls:
        - IERC20(underlying).safeTransferFrom(msg.sender,address(this),amount) (Statera-AnyswapV5ERC20.sol#341)
        Event emitted after the call(s):
        - Transfer(address(0),account,amount) (Statera-AnyswapV5ERC20.sol#396)
                - _deposit(amount,msg.sender) (Statera-AnyswapV5ERC20.sol#342)
Reentrancy in Statera.deposit(uint256,address) (Statera-AnyswapV5ERC20.sol#345-348):
```

```
        External calls:
        - IERC20(underlying).safeTransferFrom(msg.sender,address(this),amount) (Statera-AnyswapV5ERC20.sol#346)
        Event emitted after the call(s):
        - Transfer(address(0),account,amount) (Statera-AnyswapV5ERC20.sol#396)
                - _deposit(amount,to) (Statera-AnyswapV5ERC20.sol#347)
Reentrancy in Statera.depositWithPermit(address,uint256,uint256,uint8,bytes32,bytes32,address) (Statera-AnyswapV5ERC20.sol#323-327):
        External calls:
        - IERC20(underlying).permit(target,address(this),value,deadline,v,r,s) (Statera-AnyswapV5ERC20.sol#324)
        - IERC20(underlying).safeTransferFrom(target,address(this),value) (Statera-AnyswapV5ERC20.sol#325)
        Event emitted after the call(s):
        - Transfer(address(0),account,amount) (Statera-AnyswapV5ERC20.sol#396)
                - _deposit(value,to) (Statera-AnyswapV5ERC20.sol#326)
Reentrancy in Statera.depositWithTransferPermit(address,uint256,uint256,uint8,bytes32,bytes32,address) (Statera-AnyswapV5ERC20.sol#329-33
2):
        External calls:
        - IERC20(underlying).transferWithPermit(target,address(this),value,deadline,v,r,s) (Statera-AnyswapV5ERC20.sol#330)
        Event emitted after the call(s):
        - Transfer(address(0),account,amount) (Statera-AnyswapV5ERC20.sol#396)
                - _deposit(value,to) (Statera-AnyswapV5ERC20.sol#331)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Statera.mpc() (Statera-AnyswapV5ERC20.sol#177-182) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= delayVault (Statera-AnyswapV5ERC20.sol#178)
Statera.applyVault() (Statera-AnyswapV5ERC20.sol#208-211) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool)(block.timestamp >= delayVault) (Statera-AnyswapV5ERC20.sol#209)
Statera.applyMinter() (Statera-AnyswapV5ERC20.sol#213-217) uses timestamp for comparisons
        Dangerous comparisons:
```

```
Statera.applyMinter() (Statera-AnyswapV5ERC20.sol#213-217) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool)(block.timestamp >= delayMinter) (Statera-AnyswapV5ERC20.sol#214)
Statera.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (Statera-AnyswapV5ERC20.sol#462-479) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= deadline,Statera: Expired permit) (Statera-AnyswapV5ERC20.sol#463)
Statera.transferWithPermit(address,address,uint256,uint256,uint8,bytes32,bytes32) (Statera-AnyswapV5ERC20.sol#487-520) uses timestamp for
 comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= deadline,Statera: Expired permit) (Statera-AnyswapV5ERC20.sol#488)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (Statera-AnyswapV5ERC20.sol#77-83) uses assembly
        - INLINE ASM (Statera-AnyswapV5ERC20.sol#81)
Statera.constructor(string,string,uint8,address,address) (Statera-AnyswapV5ERC20.sol#283-311) uses assembly
        - INLINE ASM (Statera-AnyswapV5ERC20.sol#303)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
SafeERC20.safeApprove(IERC20,address,uint256) (Statera-AnyswapV5ERC20.sol#97-102) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (Statera-AnyswapV5ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in SafeERC20.callOptionalReturn(IERC20,bytes) (Statera-AnyswapV5ERC20.sol#103-114):
        - (success,returndata) = address(token).call(data) (Statera-AnyswapV5ERC20.sol#107)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter Statera.initVault(address)._vault (Statera-AnyswapV5ERC20.sol#188) is not in mixedCase
Parameter Statera.setMinter(address)._auth (Statera-AnyswapV5ERC20.sol#198) is not in mixedCase
Parameter Statera.setVault(address)._vault (Statera-AnyswapV5ERC20.sol#203) is not in mixedCase
Parameter Statera.revokeMinter(address)._auth (Statera-AnyswapV5ERC20.sol#220) is not in mixedCase
Function Statera.Swapin(bytes32,address,uint256) (Statera-AnyswapV5ERC20.sol#256-260) is not in mixedCase
Function Statera.Swapout(uint256,address) (Statera-AnyswapV5ERC20.sol#262-268) is not in mixedCase
Variable Statera.DOMAIN_SEPARATOR (Statera-AnyswapV5ERC20.sol#127) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
```

```
INFO:Detectors:
Statera.delay (Statera-AnyswapV5ERC20.sol#143) should be constant
Statera.delayDelay (Statera-AnyswapV5ERC20.sol#160) should be constant
Statera.pendingDelay (Statera-AnyswapV5ERC20.sol#159) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
owner() should be declared external:
        - Statera.owner() (Statera-AnyswapV5ERC20.sol#173-175)
changeMPCOwner(address) should be declared external:
        - Statera.changeMPCOwner(address) (Statera-AnyswapV5ERC20.sol#237-243)
Swapin(bytes32,address,uint256) should be declared external:
        - Statera.Swapin(bytes32,address,uint256) (Statera-AnyswapV5ERC20.sol#256-260)
Swapout(uint256,address) should be declared external:
        - Statera.Swapout(uint256,address) (Statera-AnyswapV5ERC20.sol#262-268)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Statera-AnyswapV5ERC20.sol analyzed (8 contracts with 75 detectors), 41 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

# Solidity Static Analysis

## Statera-AnyswapV5ERC20.sol

Security

**Check-effects-interaction:**

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeApprove(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 97:4:

**Check-effects-interaction:**

Potential violation of Checks-Effects-Interaction pattern in Statera.(string,string,uint8,address,address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 283:4:

**Check-effects-interaction:**

Potential violation of Checks-Effects-Interaction pattern in Statera.depositWithPermit(address,uint256,uint256,uint8,bytes32,bytes32,address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 323:4:

**Inline assembly:**

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 303:8:

**Block timestamp:**

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 178:12:

**Block timestamp:**

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 194:21:

**Block timestamp:**

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 205:21:

Gas & Economy

**Gas costs:**

Gas requirement of function Statera.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 119:4:

**Gas costs:**

Gas requirement of function Statera.symbol is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 120:4:

**Gas costs:**

Gas requirement of function Statera.decimals is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 121:4:

**Gas costs:**

Gas requirement of function Statera.setVault is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 203:4:

**Gas costs:**

Gas requirement of function Statera.getAllMinters is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 224:4:

**Gas costs:**

Gas requirement of function Statera.changeVault is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 229:4:

**Gas costs:**

Gas requirement of function Statera.mint is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 245:4:

ERC

**ERC20:**

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 10:4:

Miscellaneous

**Constant/View/Pure functions:**

IERC20.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 12:4:

**Constant/View/Pure functions:**

IERC20.approve(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 14:4:

**Constant/View/Pure functions:**

Statera.verifyPersonalSign(address,bytes32,uint8,bytes32,bytes32) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 532:4:

**Constant/View/Pure functions:**

Statera.prefixed(bytes32) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 539:4:

**Similar variable names:**

Statera.Swapin(bytes32,address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.

Pos: 257:14:

**Similar variable names:**

Statera.Swapin(bytes32,address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.

Pos: 257:23:

**Similar variable names:**

Statera._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.

Pos: 392:16:

**Similar variable names:**

Statera._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 396:43:

**Similar variable names:**

Statera._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 411:16:

**Similar variable names:**

Statera._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 413:18:

**Similar variable names:**

Statera._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 413:30:

**Similar variable names:**

Statera._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 414:24:

**Similar variable names:**

Statera.transferWithPermit(address,address,uint256,uint256,uint8,bytes32,bytes32) : Variables have very similar names "balanceOf" and "balance". Note: Modifiers are currently not considered by this static analysis.
Pos: 513:8:

**Similar variable names:**

Statera.transferWithPermit(address,address,uint256,uint256,uint8,bytes32,bytes32) : Variables have very similar names "balanceOf" and "balance". Note: Modifiers are currently not considered by this static analysis.
Pos: 513:28:

**Similar variable names:**

Statera.transferWithPermit(address,address,uint256,uint256,uint8,bytes32,bytes32) : Variables have very similar names "balanceOf" and "balance". Note: Modifiers are currently not considered by this static analysis.
Pos: 515:8:

**Similar variable names:**

Statera.transfer(address,uint256) : Variables have very similar names "balanceOf" and "balance". Note: Modifiers are currently not considered by this static analysis.
Pos: 550:8:

**Guard conditions:**

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 164:8:

**Guard conditions:**

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 169:8:

**Guard conditions:**

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 189:8:

**Guard conditions:**

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 214:8:

**Guard conditions:**

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 230:8:

**Guard conditions:**

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 238:8:

**Guard conditions:**

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 251:8:

**Data truncated:**

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 483:19:

# Solhint Linter

**Statera-AnyswapV5ERC20.sol**

```
Statera-AnyswapV5ERC20.sol:3:1: Error: Compiler version ^0.8.2 does
not satisfy the r semver requirement
Statera-AnyswapV5ERC20.sol:117:1: Error: Contract has 19 states
declarations but allowed no more than 15
Statera-AnyswapV5ERC20.sol:127:30: Error: Variable name must be in
mixedCase
Statera-AnyswapV5ERC20.sol:178:13: Error: Avoid to make time-based
decisions in your business logic
Statera-AnyswapV5ERC20.sol:194:22: Error: Avoid to make time-based
decisions in your business logic
Statera-AnyswapV5ERC20.sol:200:23: Error: Avoid to make time-based
decisions in your business logic
Statera-AnyswapV5ERC20.sol:205:22: Error: Avoid to make time-based
decisions in your business logic
Statera-AnyswapV5ERC20.sol:209:17: Error: Avoid to make time-based
decisions in your business logic
Statera-AnyswapV5ERC20.sol:214:17: Error: Avoid to make time-based
decisions in your business logic
Statera-AnyswapV5ERC20.sol:232:22: Error: Avoid to make time-based
decisions in your business logic
Statera-AnyswapV5ERC20.sol:240:22: Error: Avoid to make time-based
decisions in your business logic
Statera-AnyswapV5ERC20.sol:256:5: Error: Function name must be in
mixedCase
Statera-AnyswapV5ERC20.sol:262:5: Error: Function name must be in
mixedCase
Statera-AnyswapV5ERC20.sol:283:5: Error: Explicitly mark visibility
in function (Set ignoreConstructors to true if using solidity
>=0.7.0)
Statera-AnyswapV5ERC20.sol:300:22: Error: Avoid to make time-based
decisions in your business logic
Statera-AnyswapV5ERC20.sol:303:9: Error: Avoid using inline assembly.
It is acceptable only in rare cases
Statera-AnyswapV5ERC20.sol:463:17: Error: Avoid to make time-based
decisions in your business logic
Statera-AnyswapV5ERC20.sol:481:38: Error: Visibility modifier must be
first in list of modifiers
Statera-AnyswapV5ERC20.sol:488:17: Error: Avoid to make time-based
decisions in your business logic
```

**Software analysis result:**

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

**Email: audit@EtherAuthority.io**