



Cyberscope

# Audit Report

## **Tea-Fi**

December 2024

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>Risk Classification</b>	<b>2</b>
<b>Review</b>	<b>3</b>
Audit Updates	3
Source Files	3
<b>Overview</b>	<b>5</b>
<b>Findings Breakdown</b>	<b>6</b>
<b>Diagnostics</b>	<b>7</b>
CR - Code Repetition	8
Description	8
Recommendation	9
Team Update	9
CCR - Contract Centralization Risk	10
Description	10
Recommendation	11
Team Update	11
TSI - Tokens Sufficiency Insurance	13
Description	13
Recommendation	13
Team Update	13
<b>Functions Analysis</b>	<b>14</b>
<b>Inheritance Graph</b>	<b>18</b>
<b>Flow Graph</b>	<b>19</b>
<b>Summary</b>	<b>20</b>
<b>Disclaimer</b>	<b>21</b>
<b>About Cyberscope</b>	<b>22</b>

# Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation:** This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation:** This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical:** Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium:** Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor:** Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative:** Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
● Critical	Highly Likely / High Impact
● Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
● Minor / Informative	Unlikely / Low to no Impact

# Review

## Audit Updates

Initial Audit	19 Dec 2024
Corrected Phase 2	31 Dec 2024

## Source Files

Filename	SHA256
<b>SynthStaking.sol</b>	4f1136824b896078cd9b6a7062061b4c26 4bca137e7de34dbf34bde9e0963376
<b>interfaces/ZeroAddressError.sol</b>	96a30953a9b30293688e8d159a503b6e67 63013274b295a5d946b775b7dd9a6b
<b>interfaces/ITokenPermitSignatureDetails.sol</b>	4621780d35e831ef8270085cababbe6827 05812b793555a09909cf02b63a8b1e
<b>interfaces/ISynthStaking.sol</b>	9a7daf722c0fc3aacebc0928c89a99b3589 7ffa40abf41ee447d630d88820d24
<b>interfaces/IAllowanceTransfer.sol</b>	e6d68e521724e091a35046f89b912f9be8d cbe1ddd3e21bfac2ef9c6f8f5f655
<b>components/Storage.sol</b>	44cc3e4c33b3fd0fa92cec45e286f889f9f8 859dbac5359dd9f9244f74677067
<b>components/StakingService.sol</b>	a7ff8aa2bf2e8cf33f338530c1d880c249c2 c416b2f5cc9bdf097d843b650d07
<b>components/RewardsService.sol</b>	edca7db0f3b1a166ec60e54ad3aab83aa2 3a49ecc69b11f3ff261ce36e0a2cfb
<b>components/Permitable.sol</b>	2127a07cc9361af368175faa3c38ee4f107 b975e6fdb33cc907ae09de8b01227

**components/EIP712Service.sol**

c7ca4c7a44843449dce0b27643c4a767a4  
314053c065fd9475ab1b34ae61c3d2

## Overview

The provided smart contracts constitute a comprehensive staking system named SynthStaking. This system enables users to stake ERC20 tokens, manage their allowances through signature-based permissions, and earn rewards based on their staked amounts. The architecture is modular, comprising several interfaces and abstract contracts that define distinct functionalities such as permission handling, reward distribution, and state management.

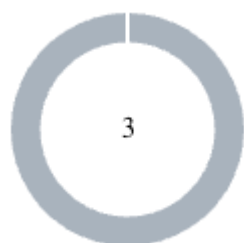
At the core of the system is the SynthStaking contract, which inherits from both StakingService and RewardsService. The StakingService manages the core staking and unstaking operations, ensuring that users can deposit and withdraw their tokens securely while adhering to predefined cooldown periods. It leverages the Permittable contract to handle token permissions via the Permit2 protocol, allowing users to approve token transfers through off-chain signatures. This enhances the user experience by reducing the number of on-chain transactions required for staking.

The RewardsService is responsible for the distribution and management of staking rewards. It utilizes the EIP712Service for secure and standardized signature verification, ensuring that reward claims are authenticated and authorized properly. The contract maintains meticulous records of rewards and implements cooldown mechanisms to mitigate abuse and ensure fair distribution among participants.

Security is a paramount consideration across all contracts. The system employs robust access control mechanisms through role-based permissions (ADMIN\_ROLE and OPERATOR\_ROLE) to restrict sensitive operations to authorized entities. Additionally, the contracts incorporate error handling for various edge cases, such as zero address validations and invalid signature detections, thereby enhancing the overall resilience of the staking platform.

In summary, the SynthStaking smart contracts provide a secure, efficient, and user-friendly framework for staking ERC20 tokens. By integrating advanced permission schemes, meticulous state management, and comprehensive reward distribution mechanisms, the system ensures a reliable and scalable solution for decentralized staking applications.

## Findings Breakdown



● Critical	0
● Medium	0
● Minor / Informative	3

Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	0	0	0	0
● Medium	0	0	0	0
● Minor / Informative	0	3	0	0

## Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	CR	Code Repetition	Acknowledged
●	CCR	Contract Centralization Risk	Acknowledged
●	TSI	Tokens Sufficiency Insurance	Acknowledged



## CR - Code Repetition

Criticality	Minor / Informative
Location	components/RewardsService.sol#L64 components/StakingService.sol#L124
Status	Acknowledged

### Description

The contract contains repetitive code segments. There are potential issues that can arise when using code segments in Solidity. Some of them can lead to issues like gas efficiency, complexity, readability, security, and maintainability of the source code. It is generally a good idea to try to minimize code repetition where possible.

```
function getWithdrawableRewardIds(address staker) external view returns
(WithdrawableIds[] memory ids) {
    uint256 length = stakerInfo[staker].rewardIds.length();
    ids = new WithdrawableIds[](length);

    for (uint256 i = 0; i < length; ++i) {
        uint256 rewardId = stakerInfo[staker].rewardIds.at(i);
        ids[i] = WithdrawableIds(rewardId, block.timestamp >
rewardRecords[rewardId].endCooldownTimestamp);
    }
}

function getWithdrawableStakeIds(address staker) external view returns
(WithdrawableIds[] memory ids) {
    uint256 length = stakerInfo[staker].unstakeIds.length();
    ids = new WithdrawableIds[](length);

    for (uint256 i = 0; i < length; ++i) {
        uint256 rewardId = stakerInfo[staker].unstakeIds.at(i);
        ids[i] = WithdrawableIds(rewardId, block.timestamp >
unstakeRecords[rewardId].endCooldownTimestamp);
    }
}
```

## Recommendation

The team is advised to avoid repeating the same code in multiple places, which can make the contract easier to read and maintain. The authors could try to reuse code wherever possible, as this can help reduce the complexity and size of the contract. For instance, the contract could reuse the common code segments in an internal function in order to avoid repeating the same code in multiple places.

## Team Update

The team replied with the following comment:

*“The team carefully examined the provided functions and noted that they are different and cannot be reused due to the fact that they operate on different fields of the WithdrawableIds structure array: rewardIds and unstakeIds. According to the team, creating a general function would only complicate the contract, as it would be necessary to include a selection by field in the WithdrawableIds structure.”*

## CCR - Contract Centralization Risk

<b>Criticality</b>	Minor / Informative
<b>Location</b>	components/Storage.sol#L192,201 components/EIP712Service.sol#L17
<b>Status</b>	Acknowledged

### Description

The contract's functionality and behavior are heavily dependent on external parameters or configurations. While external configuration can offer flexibility, it also poses several centralization risks that warrant attention. Centralization risks arising from the dependence on external configuration include Single Point of Control, Vulnerability to Attacks, Operational Delays, Trust Dependencies, and Decentralization Erosion.

For instance, the `DEFAULT_ADMIN_ROLE` has the authority to set the `unstakeCooldownDuration` and `rewardCooldownDuration` to a very large value. As a result, stakers will not be able to withdraw their unstaked tokens and their rewards respectively.

```
function setUnstakeCooldownDuration(uint256 duration) external
onlyRole(DEFAULT_ADMIN_ROLE) {
    _setUnstakeCooldownDuration(duration);
}
function setRewardCooldownDuration(uint256 duration) external
onlyRole(DEFAULT_ADMIN_ROLE) {
    _setRewardCooldownDuration(duration);
}
function _verifyCollectRewardSignature(
    address from,
    CollectRewardParam calldata param
) internal returns (bool result, string memory errorReason) {
    bytes memory encodedData = abi.encode(
        COLLECT_REWARD_TYPEHASH,
        param.staker,
        param.rewardAmount,
        param.nonce,
        param.deadline
    );
    return
        _verifySignature(encodedData, from, param.operator, param.nonce,
        param.deadline, param.v, param.r, param.s);
}
```

## Recommendation

To address this finding and mitigate centralization risks, it is recommended to evaluate the feasibility of migrating critical configurations and functionality into the contract's codebase itself. This approach would reduce external dependencies and enhance the contract's self-sufficiency. It is essential to carefully weigh the trade-offs between external configuration flexibility and the risks associated with centralization.

## Team Update

The team replied with the following comment:

*“The DEFAULT\_ADMIN\_ROLE is intentionally designed to be decentralized and not concentrated within a single entity. It is assigned to a secure multi-signature wallet that requires multiple approvals (e.g., 2 out of 3 or 3 out of 5 confirmations) for executing actions. This approach distributes decision-making authority and mitigates risks associated with a single point of failure.”*

*Operators, while possessing limited authority, operate under strict role-based access control and are restricted to performing only the actions explicitly authorized by the contract. Additionally, the rigorous integration of the EIP-712 protocol, combined with a robust watcher service for monitoring and flagging anomalies in rewards distribution, ensures a secure and reliable synth staking process without compromising the system's decentralization.*

*The reliance on configurable parameters does not undermine decentralization but enhances the protocol's adaptability to unforeseen circumstances. To exemplify, the ability to adjust parameters such as `unstakeCooldownDuration` and `rewardCooldownDuration` allows the protocol to respond effectively to network changes, such as congestion or shifts in user behavior."*

## TSI - Tokens Sufficiency Insurance

Criticality	Minor / Informative
Location	components/RewardsService.sol#L75
Status	Acknowledged

### Description

The tokens are not held within the contract itself. Instead, the contract is designed to provide the tokens from an external administrator. While external administration can provide flexibility, it introduces a dependency on the administrator's actions, which can lead to various issues and centralization risks.

```
teaToken.safeTransferFrom(treasury, to, amount);
```

### Recommendation

It is recommended to consider implementing a more decentralized and automated approach for handling the contract tokens. One possible solution is to hold the tokens within the contract itself. If the contract guarantees the process it can enhance its reliability, security, and participant trust, ultimately leading to a more successful and efficient process.

### Team Update

The team replied with the following comment:

*"The contract's architecture has been designed to rely on the treasury for token transfers, so that a clear separation of responsibilities is guaranteed. This approach allows treasury owners to dynamically manage liquidity and allocations without the need to deploy new contract versions or risk losing tokens locked within the contract. The treasury will operate as a secure multi-signature wallet, requiring 2 out of 3 or 3 out of 5 confirmations for any action. All treasury activities are logged and monitored by dedicated watcher service, which tracks balances and promptly flags any anomalies or potential depletion risks."*

## Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
<b>SynthStaking</b>	Implementation	StakingService, RewardsService, ERC2771Context		
		Public	✓	validAddress Storage Permitable ERC2771Context EIP712
	emergencyWithdrawErc20	External	✓	validAddress onlyRole
	emergencyWithdrawEth	External	✓	validAddress onlyRole
	_msgSender	Internal		
	_msgData	Internal		
	_contextSuffixLength	Internal		
<b>ISynthStakingErrors</b>	Interface	ZeroAddress Error		
<b>ISynthStaking</b>	Interface	ISynthStakingErrors		
	stake	External	✓	-
	stake	External	✓	-
	stake	External	✓	-
	unstake	External	✓	-

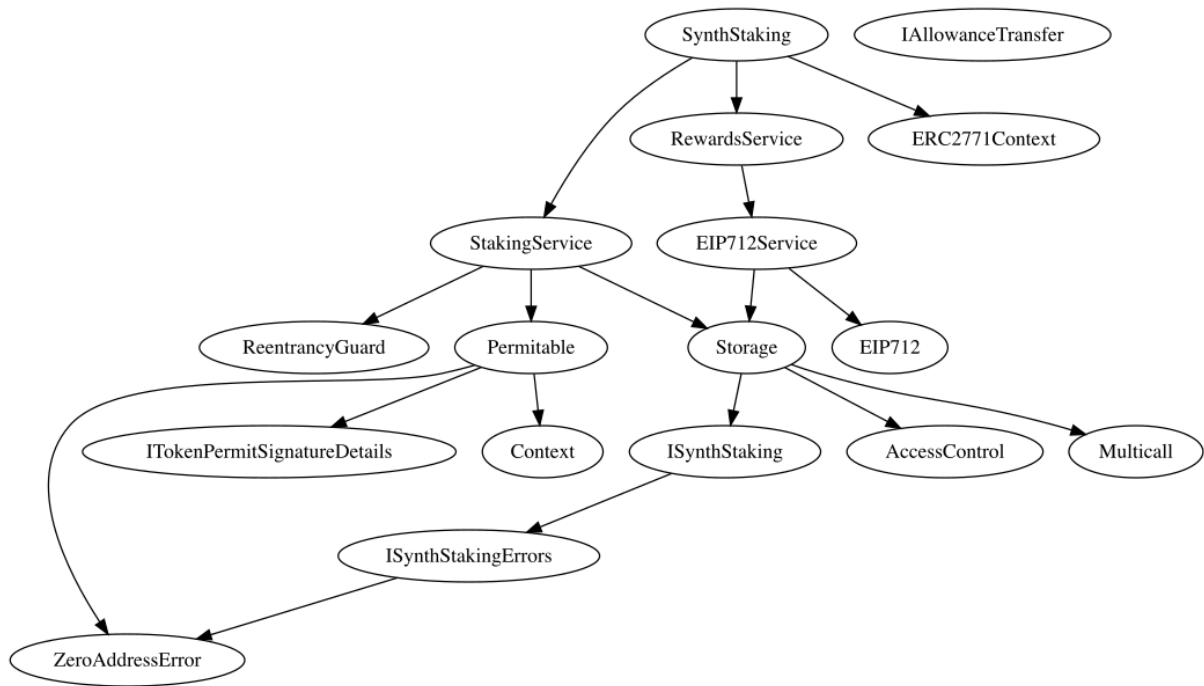
	withdrawStake	External	✓	-
	collectReward	External	✓	-
	withdrawReward	External	✓	-
	getPoolInfo	External		-
	getStakerInfo	External		-
<b>Storage</b>	Implementation	ISynthStaking, AccessControl, Multicall		
		Public	✓	validAddress
	createPool	External	✓	onlyRole validAddress
	setUnstakeCooldownDuration	External	✓	onlyRole
	setRewardCooldownDuration	External	✓	onlyRole
	setTreasury	External	✓	onlyRole
	setTotalAllocation	External	✓	onlyRole
	getPoolInfo	External		-
	getStakerInfo	External		-
	getTotalStakedByStakerByPool	External		-
	_setupRoles	Private	✓	
	_setUnstakeCooldownDuration	Private	✓	isNotZeroUint256
	_setRewardCooldownDuration	Private	✓	isNotZeroUint256
	_setTreasury	Private	✓	validAddress
	_setTotalAllocation	Private	✓	isNotZeroUint256



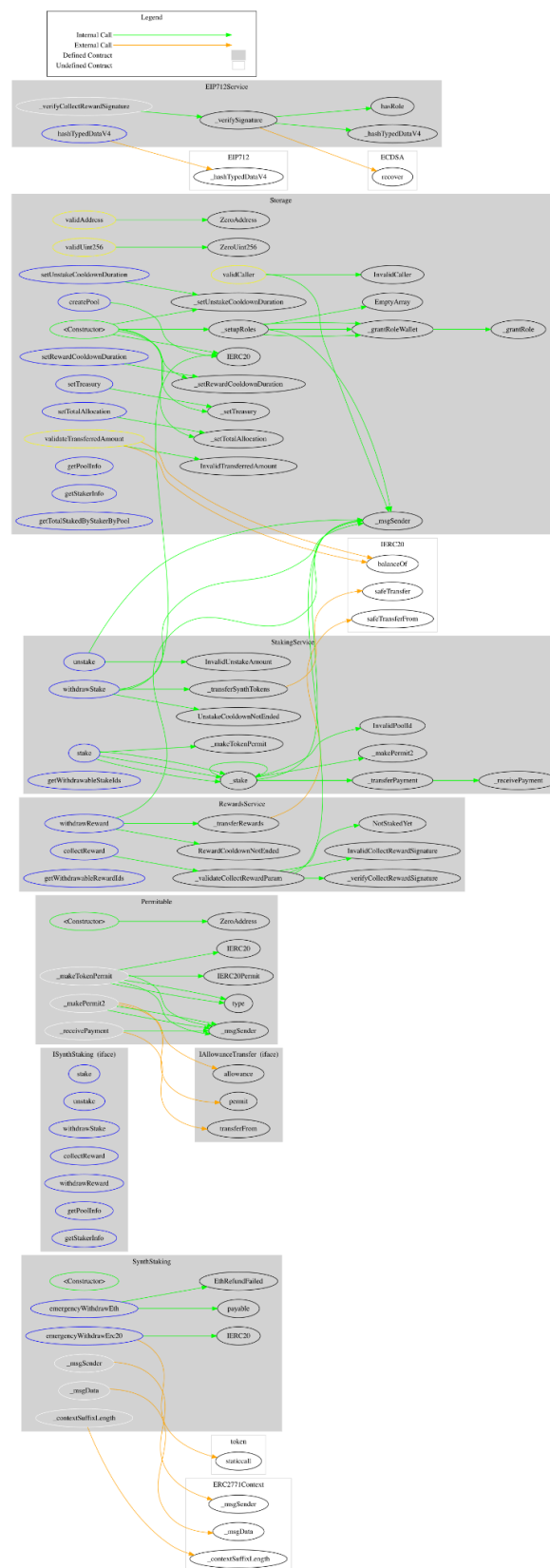
	_grantRoleWallet	Private	✓	validAddress
<b>StakingService</b>	Implementation	Storage, Permitable, ReentrancyGuard		
	stake	External	✓	-
	stake	External	✓	-
	stake	External	✓	-
	unstake	External	✓	validCaller isNotZeroUint256
	withdrawStake	External	✓	validCaller
	getWithdrawableStakeIds	External		-
	_stake	Internal	✓	
	_stake	Internal	✓	isNotZeroUint256 nonReentrant
	_transferPayment	Private	✓	validateTransferredAmount
	_transferSynthTokens	Private	✓	validateTransferredAmount
<b>RewardsService</b>	Implementation	EIP712Service		
	collectReward	External	✓	-
	withdrawReward	External	✓	validCaller
	getWithdrawableRewardIds	External		-
	_transferRewards	Private	✓	validateTransferredAmount
	_validateCollectRewardParam	Private	✓	validAddress isNotZeroUint256 validCaller

<b>Permitable</b>	Implementation	ZeroAddress Error, ITokenPermit SignatureDet ails, Context		
		Public	✓	-
	_makeTokenPermit	Internal	✓	
	_makePermit2	Internal	✓	
	_receivePayment	Internal	✓	
<b>EIP712Service</b>	Implementation	Storage, EIP712		
	_verifyCollectRewardSignature	Internal	✓	
	_verifySignature	Internal	✓	
	hashTypedDataV4	External		-

# Inheritance Graph



## Flow Graph



## Summary

Tea-Fi contracts implement a staking and rewards mechanism. This audit investigates security issues, business logic concerns, and potential improvements.

## Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security. Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

# About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.



**The Cyberscope team**

[cyberscope.io](https://cyberscope.io)