Cyberscope

Audit Report

Tea-Fi Token

October 2024

File	TeaToken
SHA256	c732455e6d4273a03f4edee779509b6bd80bc462e776c02268d5005b8818cb6e
Audited by	© cyberscope

Table of Contents

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

Contract Name	TeaToken
Testing Deploy	https://testnet.bscscan.com/address/0xfba763aa9f1e5aac245b1ade0b18fe522cdf9a92
Symbol	TS-TEA
Decimals	18
Total Supply	300,000,000

Audit Updates

Initial Audit	11 Oct 2024

Source Files

Filename	SHA256
TeaToken.sol	c732455e6d4273a03f4edee779509b6bd80bc462e776c02268d5005b8818cb6e
interfaces/ZeroAddressError.sol	83642b852ae173732f849ec7dfe02b6ba5bf0fbc54f4571253c95628ae2cd1aa

Overview

The `TeaToken` is a standard ERC20 token with additional features for burning, minting, and voting. It includes meta-transaction support via the `ERC2771Context` and is designed to be governable with voting features enabled through `ERC20Votes` .

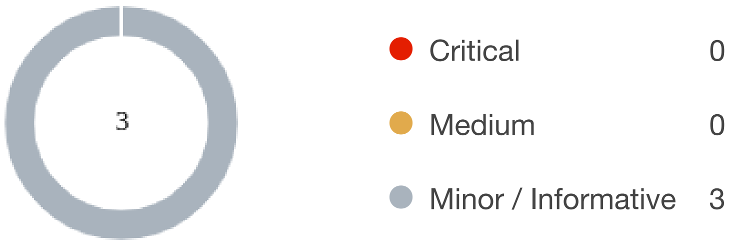
`mint` Functionality

The `mint` function allows the owner of the contract to mint new tokens to a specified recipient. This function is controlled by ownership, ensuring that only the designated owner can issue new tokens.

Other Functionalities

The constructor initializes the token with a name, symbol, trusted forwarder, multisig wallet, and initial supply. The contract supports token burning and integrates with `ERC20Votes` for governance-related voting. It also overrides functions from `ERC20Permit` and `ERC2771Context` for compatibility with meta-transactions and permit functionality.

Findings Breakdown



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
●	MT	Mints Tokens	Acknowledged
●	MC	Missing Check	Acknowledged
●	L04	Conformance to Solidity Naming Conventions	Acknowledged

MT - Mints Tokens

Criticality	Minor / Informative
Location	TeaToken.sol#L41
Status	Acknowledged

Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the `mint` function. As a result, the contract tokens will be highly inflated.

```
    function mint(address recipient, uint256 amount) external
onlyOwner {
        // recipient is checked in ERC20._mint
        _mint(recipient, amount);
    }
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

Team Update

The team has acknowledged that this is not a security issue and states:

In order to minimize this risk, the team introduced the following measures:

- *A multi-signature wallet has been introduced as a contract owner that requires more than two signatures to authorize any transaction.*
- *There are plans to develop a smart-contract that limits the amount and timing of token mints based on tokenomics to manage the token inflaction. Once the smart contract has been created and audited, ownership will be transferred to it.*

MC - Missing Check

Criticality	Minor / Informative
Location	TeaToken.sol#L24
Status	Acknowledged

Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

Specifically, the contract is missing a check to verify that the addresses are not set to the zero address.

```
    constructor(
        string memory name_, // "Tea-Fi Token"
        string memory symbol_, // "TEA"
        address trustedForwarder_,
        address multisigWallet,
        address _treasury,
        uint256 initialSupply // 300M
    ) ERC20(name_, symbol_) ERC2771Context(trustedForwarder_)
ERC20Permit(name_) Ownable(multisigWallet) {
        // treasury is checked in ERC20._mint
        _mint(_treasury, initialSupply);
    }
```

Recommendation

The team is advised to properly check the variables according to the required specifications.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	TeaToken.sol#L57,65
Status	Acknowledged

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
function CLOCK_MODE() public pure override returns (string
memory) {
        return "mode=timestamp";
    }
address _owner
```

Recommendation

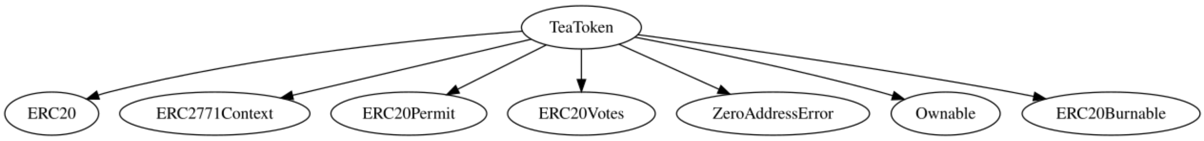
By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation https://docs.soliditylang.org/en/stable/style-guide.html#naming-conventions.

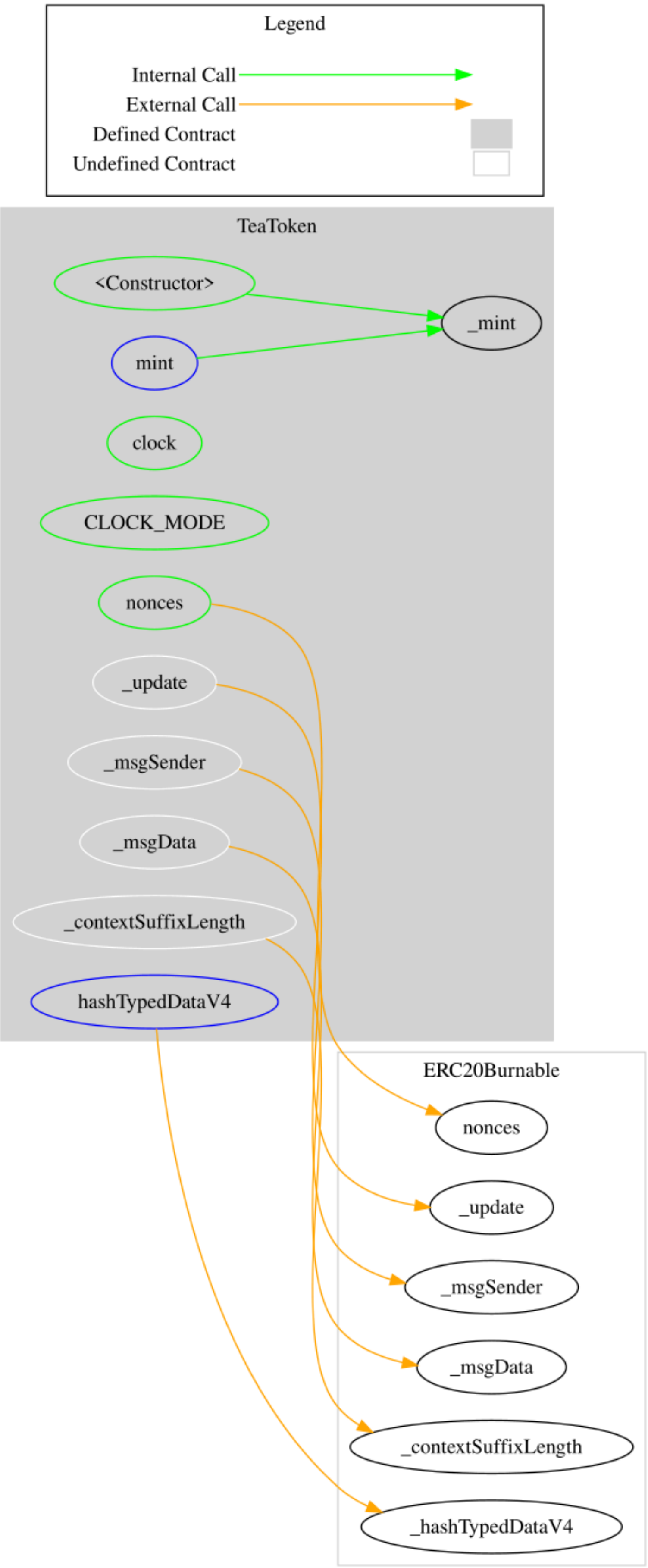
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
TeaToken	Implementation	ERC20, ERC2771Context, ERC20Permit, ERC20Votes, ZeroAddressError, Ownable, ERC20Burnable		
		Public	✓	ERC20 ERC2771Context ERC20Permit Ownable
	mint	External	✓	onlyOwner
	clock	Public		-
	CLOCK_MODE	Public		-
	nonces	Public		-
	_update	Internal	✓	
	_msgSender	Internal		
	_msgData	Internal		
	_contextSuffixLength	Internal		
	hashTypedDataV4	External		-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
TeaToken
<Constructor>
_mint
mint
clock
CLOCK_MODE
nonces
_update
_msgSender
_msgData
_contextSuffixLength
hashTypedDataV4
ERC20Burnable
nonces
_update
_msgSender
_msgData
_contextSuffixLength
_hashTypedDataV4

Summary

The Tea-Fi contract implements a token mechanism that allows the owner to mint it. This audit focuses on identifying security issues, assessing the contract’s logic, and suggesting potential improvements. The team has acknowledged the findings.

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