PUBLIC

Code Assessment

of the Firetoken Smart Contracts

August 15, 2023

Produced for

by

Contents

1 Executive Summary

Dear all,

Thank you for trusting us to help Fire Group Ltd. with this security audit. Our executive summary provides an overview of subjects covered in our audit of the latest reviewed contracts of Firetoken according to Scope to support you in forming an opinion on their security risks.

Fire Group Ltd. implements an ERC-20 and ERC-1404 compliant FireToken, which is governed by the owner and features restricted token transfers.

The most critical subjects covered in our audit are functional correctness, access control and standard compliance. Security regarding all the aforementioned subjects is high.

The general subjects covered are code complexity and quality of specification documentation. Fire Group Ltd. did not provide any specifications, test cases, git commits or the framework setup.

In summary, we find that the codebase provides a high level of security.

It is important to note that security audits are time-boxed and cannot uncover all vulnerabilities. They complement but don't replace other vital measures to secure a project.

The following sections will give an overview of the system, our methodology, the issues uncovered and how they have been addressed. We are happy to receive questions and feedback to improve our service.

Sincerely yours,

ChainSecurity

1.1 Overview of the Findings

Below we provide a brief numerical overview of the findings and how they have been addressed.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	0

2 Assessment Overview

In this section, we briefly describe the overall structure and scope of the engagement, including the code commit which is referenced throughout this report.

2.1 Scope

The assessment was performed on the source code files inside the Firetoken repository based on the documentation files.

The scope consists of the following solidity smart contracts:

1. contracts/FireToken.sol
2. contracts/IERC1404.sol

The table below indicates the code versions relevant to this report and when they were received.

V	Date	Commit Hash	Note
1	07 Jul 2023	zip File no Commit provided	Initial Version

For the solidity smart contracts, the compiler version `0.5.17` was chosen.

2.1.1 Excluded from scope

Any other file not explicitly mentioned in the scope section. In particular tests, scripts, external dependencies, and configuration files are not part of the audit scope.

2.2 System Overview

This system overview describes the initially received version (Version 1) of the contracts as defined in the Assessment Overview.

Furthermore, in the findings section, we have added a version icon to each of the findings to increase the readability of the report.

Fire Group Ltd. offers an ERC-20 and ERC-1404 compliant FireToken, which features restricted token transfers.

2.2.1 FireToken

The FireToken contract is governed by an owner which is set to `msg.sender` upon deployment. The owner has the privileges of:

- `addUserListToKycRole()` - grant a KYC-ed role to a list of addresses.
- `removeUserFromKycRole()` - revoke a KYC-ed role from an address.
- `addTransferBlock()` - add a transfer block to an address.
- `removeTransferblock()` - remove the transfer block from an address.
- `mintTo()` - mint any amount of FireToken to an address.
- `burn()` - burn any amount of FireToken of an address.
- `setParent()` - Set the parent's address of a child's address.

- Update the mapping of a code to a human-readable string explaining an execution result by: `setRestrictionCode()`, `setBurnCode()`, `setMintCode()`, `setBlockCode()`, `removeRestrictionCode()`, `removeBurnCode()`, `removeMintCode()` and `removeBlockCode`.

FireToken functions as a normal ERC-20 token with 5 decimals, while the token transfers are restricted in the following cases:

- The sender has insufficient balance.
- The sender or the recipient does not have `_kyc` role.
- The sender or the recipient has `_transferblock` role.

In case the call to `transfer()` or `transferFrom()` reverts, the user can retrieve the restriction code by `detectTransferRestriction()`. For admin's operations to burn, mint, and update transfer blocks, users can retrieve a human-readable strings by querying the following functions with the execution `code` from the emitted events:

- `messageForTransferRestriction()` - returns a human-readable message for a given restrictioncode
- `messageForBurnCode()` - returns a human-readable message for a given burncode.
- `messageForMintCode()` - returns a human-readable message for a given mintcode.
- `messageForBlockCode()` - returns a human-readable message for a given blockcode.

Note the human-readable message for a given `code` can be changed by the owner of the contract by removing the `code` and re-adding again. In addition, the human-readable message of a given `code` may not exist upon users' query. Though the contract enforces that the `code` must exist at the time of the owner's privileged operation, the owner can easily remove the `code` after the operation.

A view function `getParent()` is provided to retrieve the parent address of a child address.

2.2.2 Roles and Trust Model

The owner role is the only admin of FireToken contract. We assume the owner is always trusted and never acts against the interest of the legitimate token users, otherwise it can:

- Manipulate the account balance by minting or burning tokens.
- Add a KYC-ed role to any address or revoke it from any address.
- Block any address to restrict all incoming or outgoing transfers.
- Modify or remove any operation code.

3 Limitations and use of report

Security assessments cannot uncover all existing vulnerabilities; even an assessment in which no vulnerabilities are found is not a guarantee of a secure system. However, code assessments enable the discovery of vulnerabilities that were overlooked during development and areas where additional security measures are necessary. In most cases, applications are either fully protected against a certain type of attack, or they are completely unprotected against it. Some of the issues may affect the entire application, while some lack protection only in certain areas. This is why we carry out a source code assessment aimed at determining all locations that need to be fixed. Within the customer-determined time frame, ChainSecurity has performed an assessment in order to discover as many vulnerabilities as possible.

The focus of our assessment was limited to the code parts defined in the engagement letter. We assessed whether the project follows the provided specifications. These assessments are based on the provided threat model and trust assumptions. We draw attention to the fact that due to inherent limitations in any software development process and software product, an inherent risk exists that even major failures or malfunctions can remain undetected. Further uncertainties exist in any software product or application used during the development, which itself cannot be free from any error or failures. These preconditions can have an impact on the system's code and/or functions and/or operation. We did not assess the underlying third-party infrastructure which adds further inherent risks as we rely on the correct execution of the included third-party technology stack itself. Report readers should also take into account that over the life cycle of any software, changes to the product itself or to the environment in which it is operated can have an impact leading to operational behaviors other than those initially determined in the business specification.

4 Terminology

For the purpose of this assessment, we adopt the following terminology. To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

- *Likelihood* represents the likelihood of a finding to be triggered or exploited in practice
- *Impact* specifies the technical and business-related consequences of a finding
- *Severity* is derived based on the likelihood and the impact

We categorize the findings into four distinct categories, depending on their severity. These severities are derived from the likelihood and the impact using the following table, following a standard risk assessment procedure.

Likelihood	Impact		
	High	Medium	Low
High	Critical	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Low

As seen in the table above, findings that have both a high likelihood and a high impact are classified as critical. Intuitively, such findings are likely to be triggered and cause significant disruption. Overall, the severity correlates with the associated risk. However, every finding's risk should always be closely checked, regardless of severity.

5 Findings

In this section, we describe our findings. The findings are split into these different categories:

Below we provide a numerical overview of the identified findings, split up by their severity.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	0

6 Informational

We utilize this section to point out informational findings that are less severe than issues. These informational issues allow us to point out more theoretical findings. Their explanation hopefully improves the overall understanding of the project's security. Furthermore, we point out findings which are unrelated to security.

6.1 Gas Optimizations

Informational **Version 1**

CS-FRTK-001

- The state variables `_name`, `_symbol` and `_decimals` could be declared as constants. As a result, the compiler does not reserve a storage slot for these variables, and every occurrence is replaced by the respective value. Compared to regular state variables, the gas costs of constant and immutable variables are much lower. For a constant variable, the expression assigned to it is copied to all the places where it is accessed and also re-evaluated each time. This allows for local optimizations. Immutable variables are evaluated once at construction time and their value is copied to all the places in the code where they are accessed. For these values, 32 bytes are reserved, even if they would fit in fewer bytes. Due to this, constant values can sometimes be cheaper than immutable values. (see Solidity docs)
- `_transfer()` does not need to use safeMath when modifying the `_balances` of the sender and the recipient. Because the sender's balance has already been checked in `detectTransferRestriction()` to ensure sufficient funds for the transfer. And the recipient's balance is always less or equal to `totalSupply`, in case `totalSupply` does not overflow during minting, the recipient's balance will never overflow.
- `_mint()` does not need to use safeMath when updating `_balances[account]`. The balance of any account is always less or equal to `totalSupply`.In case the previous update to `_totalSupply` does not overflow, `_balances[account]` will not overflow as well. This applies to the updates of `_totalSupply` in `_burn()` as well.
- The check of `onlyOwner` is redundant for internal function `_mint()`, because `_mint()` is only called by the external function `mintTo()`, which is already marked with modifier `onlyOwner`. This applies to the internal function `_burn()` as well.

6.2 Indexed Fields of Events

Informational **Version 1**

CS-FRTK-002

The events `Burn`, `Mint`, `Block` and `Unblock` do not mark the field `code` as indexed. Indexing fields in events allows to easily search for certain events. `code` is not a random number but is a limited set and could be indexed.

Fire Group Ltd. states:

```
The field code is more an add-on-info for the reason of events. At the time of
writing the contract, searching based on codes seemed not to be a requirement.
Thus, it was decided to not index the code part of the event.
```

6.3 Missing Events of KYC Roles Updates

Informational **Version 1**

CS-FRTK-003

The contract owner's call to `addUserListToKycRole()` and `removeUserFromKycRole()` will update the KYC roles, nevertheless, no events will be emitted to reflect the storage modification.

Fire Group Ltd. states:

```
KYC data is stored off-chain while executing the KYC processes. Thus, it was
decided to not emit events for adding and removing KYC roles as this information
is available off-chain.
```

6.4 Redundant Transfer Restriction

Informational **Version 1**

CS-FRTK-004

Without specifications it is unclear if `address(0)` is an address that has transfer restrictions in the `_kyc` set or not. In case it does have transfer restrictions and is not part of the set, the requires in `_transfer` are redundant.

6.5 Unused Variable `_propertyAmountLocks`

Informational **Version 1**

CS-FRTK-005

The contract defines a state variable called `_propertyAmountLocks` but it is not used.

7 Notes

We leverage this section to highlight further findings that are not necessarily issues. The mentioned topics serve to clarify or support the report, but do not require an immediate modification inside the project. Instead, they should raise awareness in order to improve the overall understanding.

7.1 No Way to Recover ETH or Token Sent to the Contract

Note Version 1

The contract has no functionality to recover ETH or token sent to the contract. All funds sent to the contract will be locked forever.

7.2 Reserve Code <100 for Contract Internal Use

Note Version 1

When the owner inserts a new code into the mapping, the code is required to be larger than 100. Whereas only less than 6 are used in the constructor, the rest are unused.

7.3 Unusual Decimals

Note Version 1

The token has only 5 decimals. Most contracts have 18 decimals which is the standard base in the Ethereum network. Many issues can arise when a token with other than 18 decimals shall be included in third party protocols. Hence, Fire Group Ltd. should carefully evaluate if it is necessary to use 5 decimals and state this very clearly everywhere (in-line documentation, online documentation, website and if other protocols are using the token) to mitigate future issues.