SX1411 - AssetToken Security Analysis

by Pessimistic

This report is public

June 3, 2022

Abstract 2
Disclaimer 2
Summary 2
General recommendations 2
Project overview 3
Project description 3
Code base update #1 3
Code base update #2 3
Code base update #3 3
Code base update #4 4
Code base update #5 4
Procedure 5
Manual analysis 6
Critical issues 6
Medium severity issues 7
Excessive restrictions (not an issue) 7
Bug (fixed) 7
Incorrect integration with OpenZeppelin (fixed) 8
Excessive approval for Guardian role (fixed) 8
Overpowered role 9
Low severity issues 10
Code quality 10
Gas consumption (fixed) 12
Project management (fixed) 12

Abstract

In this report, we consider the security of smart contracts of SX1411 - AssetToken project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, security audit is not an investment advice.

Summary

In this report, we considered the security of SX1411 - AssetToken smart contracts. We performed our audit according to the procedure described below.

The initial audit showed no critical issues. However, two bugs and incorrect usage of OpenZeppelin library were discovered. The contracts also depend heavily on users with special roles and include a number of low severity issues.

The details of interaction between the contracts and off-chain code are not documented.

After the initial audit, the code base was updated. In this update,Overpowered role and most of low severity issues were fixed. Also, code coverage was significantly improved.

After the recheck #1, another code base update was performed. In this update, all medium severity issues were fixed.

During a recheck of a new code base update, an Excessive restrictions issue of medium severity was discovered. However, after the update of the documentation, this occurred to be an intentional behavior of the code.

After the third recheck, the code base was updated again. This update had two new issues of medium severity, including Excessive approval for Guardian role, Overpowered role, and new issues of low severity.

During the recheck #5 of a new code base update, an Excessive approval for Guardian role issue of medium severity and most of low severity issues were fixed. The number of tests decreased.

General recommendations

We recommend fixing the mentioned issues.

Project overview

Project description

For the audit, we were provided with SX1411 - AssetToken project on a GitLab repository, commit 3f4aa34f9d12cca3dea86baa50173548928253bc.

The project has README.md file, and private documentation.

The project compiles with warnings and the compiler version in the config file is set incorrectly.

One test out of 16 does not pass, the code coverage is 20.56%.

The total LOC of audited sources is 405.

Code base update #1

For the recheck #1, we were provided with SX1411 - AssetToken project on a private GitHub repository, commit 23f1f08c6d5a944dede093d40d9ffcdf30ceb0e5.

In this update, a few issues were fixed. Also, new tests were added to the project, the code coverage increased to 99.56%.

Code base update #2

After the recheck #1, another code base update was performed. For the recheck #2, we were provided with commit 272df8639084bf53fb1eda826352ad8b38bcc806.

In this update, issues of medium severity were fixed and new tests were added to the project. However, the code coverage decreased to 99.2%. Also, developers added new functionality and provided a documentation.

Code base update #3

The code was moved to a new repo. For the recheck #3, we were provided with updated code base on commit 4ee565c0a5487709b749327e9edb3ae6c2a47f21.

In this update, the code was optimized for the contract to fit one block. Also, minor modifications were made to tests, the overall code coverage slightly increased to 99.26%.

After this update, the developers provided a new documentation for the project.

Code base update #4

For the recheck #4, we were provided with commit d146be2a1ce6464f1cbf9779a0bd93632508dc6e on a private GitHub repository. The scope of the code base included the following contracts: **AccessManager**, **AssetToken**, **AssetTokenData**.

In this update, two issues of medium severity and few of low severity were found. The number of tests changed, all 751 tests pass and the code coverage is 99.74%.

The developers also provided the additional part of the documentation for the project, sha1sum is fdfa6f1e3c0308d23655a626d33083ca9befaee1.

Code base update #5

For the recheck #5, we were provided with commit 57653569ade5e7a6bf1eb1607dc860c22fca472d on a private GitHub repository. One medium severity issue and the majority of low severity issues were fixed.

The number of tests decreased and all 744 tests passed. The code coverage is 99.73%.

Procedure

In our audit, we consider the following crucial features of the code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices.

We perform our audit according to the following procedure:

- Automated analysis
    - We scan project's code base with automated tools: Slither and SmartCheck.
    - We manually verify (reject or confirm) all the issues found by tools.
- Manual audit
    - We manually analyze code base for security vulnerabilities.
    - We assess overall project structure and quality.
- Report
    - We reflect all the gathered information in the report.

Manual analysis

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

The audit showed no critical issues.

Medium severity issues

Medium severity issues can influence project operation in current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

Excessive restrictions (not an issue)

`_beforeTokenTransfer` function of **AssetToken** contract includes multiple checks. This function is called internally within all ERC20 token operations. Checks at lines 154-156 partially disable `burn` function, which contradicts documentation:

```
'Burn' function can be called by any token holder in either Active
or Safeguard states.
```

A user cannot burn their tokens when the contract is in the Safeguard state unless the caller has SAFEGUARD_TRANSFER_ROLE role.

Additionally, multiple functions rely on the contract being an authorized holder of the tokens, which we do not consider an issue.

The documentation for the project was updated. The behavior of the contract is correct.

Bug (fixed)

- When a user stakes their wrapped tokens in **xToken** contract, their tokens are consumed twice. First, the tokens are transferred to the contract balance during `stake` function call. Second, the same amount of tokens are burned from the users balance when the Guardian approves redemption request.

- The same token rate is applied for mint and for burn in **xToken** contract. Consider the following example:

  Mint operation was initiated with `amount = 100`, and `rate = 2`, which updates user's balance `balanceOf(user) = 50`.

  For simplicity, rate does not change, and burn operation with `amount = 25` uses all the user's tokens.

  That is, off-chain `amount` values are `100` for direct operation and `25` for the reverse (`amount → amount / rate ^ 2`). However, one can reasonably expect these operations use the same value.

The issues have been fixed and are not present in the latest version of the code.

Incorrect integration with OpenZeppelin (fixed)

Function `_beforeTokenTransfer` of **xToken** contract at line 90 should be `virtual` and call `super._beforeTokenTransfer(from, to, amount)` as described in OpenZeppelin documentation.

The issue has been fixed and is not present in the latest version of the code.

Excessive approval for Guardian role (fixed)

During staking in **xToken** contract, the user gives a Guardian approval for amount at lines 224 and 238, which is not required by the contract. Therefore, the Guardian can withdraw additional amount of tokens from the user.

Update 4: The issue was fixed and was not present in the previous version of the code. However, in this update we found this problem again at lines 597, 638 in **AssetToken** contract.

Update 5: The issues have been fixed and are not present in the latest version of the code.

Overpowered role

The project has special roles with excessive powers. As a result, the system depends heavily on these roles.

`Issuer` role in Active state and `Guardian` role in Safeguard state can:

- freeze/unfreeze contract.
- make the requests for mint and burn and approve them or remove a redemption request (`Guardian` can not make the mint requests).
- change the address of **AssetTokenData** contract.
- set minimum redemption amount.
- add/remove agents and members from/to black list.
- delete asset token.

In Active state `Issuer` role also can mint any tokens they want so Safeguard state never begins.

We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig.

> *Comment from developers: Swarm Markets is a unique platform that bridges decentralized markets with investor protection, regulated as a financial entity under BaFin in Germany. As such we must comply with certain financial regulations that are designed to protect users. The primary use case for the SX1411 Asset Token smart contract system is to create a digital representation of an off-chain asset that has been placed into custody with a trusted entity such as a custodian or brokerage house, and then made available as a tradeable asset on the Swarm Markets exchange. The architecture defines participants in the system which have certain administrative powers within the contract, as defined in our compliance requirements. What is important to note is that every address authorized to hold or interact with any asset on Swarm Markets must successfully pass a KYC and AML process and be identified to the same standards as required by any regulated financial institution. In this system, this applies not only to token holders, but also those participants which have administrative powers, such as the Issuer, Guardian and any Agents We feel that this requirement addresses the concerns raised.*

Low severity issues

Low severity issues don't directly affect project's operations. However, they might lead to various problems in the future versions of the code. We recommend taking them into account.

Code quality

- `update` function of **CompoundRateKeeper** contract has TODO comment at line 48 and this is not implemented in the code.

  The issue has been fixed and is not present in the latest version of the code.

- Storage variables of **xToken** contract are considered `internal` by default. Declare visibility of `mintRequestID` , `redemptionRequestID`, `hundredPercent` variables explicitly to improve code readability.

  Same issues at line 5 in **AdminManager** contract and at lines 21, 14 in **MembersManager** contract.

  The issues have been fixed and are not present in the latest version of the code.

- Variables of **xToken** contract at line 48 are set via constructor and never changed later. Declare them as `immutable` to reduce gas consumption.

  This issue is not relevant anymore in the latest version of the code.

- **AgentManager** contract has `equalStrings` and `notEqualStrings` functions at lines 27-33 which are almost identical.

  This part of the code has been removed from the code base.

- Mark `transferOwnership` function of **AdminManager** contract as `external`. It improves readability and saves a bit of gas.

  Update 4: This part of the code had been removed from the code base in the previous updates. In this update, we found the same issue for `onlyIssuer` function in **AccessManager** contract.

  Update 5: The issue has been fixed and is not present in the latest version of the code.

- Literal `21979553151239153027` at line 42 in `setInterestRate` function of **CompoundRateKeeper** contract should be declared as constant. Also, consider adding a comment in the code that explains where this value comes from.

> *Comment from developers: Regarding the constant, this is a number intended to limit the possible interest rate that can be set per year. The original developer indicated we should limit it as at some point the high numbers would cause an error. The limit we selected was 100% per year, which when calculated per second, derives the constant in the contract. You can also see that using this formula, which converts the annual interest rate to a per-second interest rate. Using the value 2 to represent 100%, we get the same constant as is in the contract.*

- Consider replacing `10 ** 27` with `1e27` at line 17 in **CompoundRateKeeper** contract.

  Same issue in **xToken** contract at line 46.

  > *Comment from developers: Given in the **DSMath** library the `RAY` definition it's written like this `10**27`, I replicated that in the **CompoundRateKeeper**.*

- In **AssetToken** contract, `safeguardStake` function initializes fields of `redemptionRequests` struct consequently at lines 571–579. Consider declaring the struct within a single statement to optimize gas consumption, prevent some types of errors and improve code readability.

  The issue has been fixed and is not present in the latest version of the code.

- `_isContract` check is not necessary in `freezeContract`, `setContractToSafeguard`, `unfreezeContract` functions of **AccessManager** contract since there is the same check inside of token registration. And `onlyStoredToken` modifier checks whether this token address is registered or not.

  The issue has been fixed and is not present in the latest version of the code.

- `_index` parameter has `uint256` type. Thus, it is not necessary to check this variable for zero.

  The issue has been fixed and is not present in the latest version of the code.

- The following functions violate the CEI (checks-effects-interactions) pattern:
  - `_safeguardUnstake` function of **AssetToken** contract.
  - `safeguardStake` function of **AssetToken** contract.

  We highly recommend following CEI pattern to increase the predictability of the code execution and protect from some types of re-entrancy attacks.

  The issues have been fixed and are not present in the latest version of the code.

- Unnecessary assignment and check of `success` variable in `freezeContract` and `unfreezeContract` functions of **AccessManager** contract. This variable is always true.

Gas consumption (fixed)

- Additional storage variable assignments are redundant in **xToken** contract:
    - at line 172.
    - at line 127.

  The issues have been fixed and are not present in the latest version of the code.
- Reading `length` and `tokensData[_tokenAddress]` storage variables of **AccessManager** contract in loop consumes gas on each iteration. Declare local variable with this value before the loop:
    - at line 233 in `_agentHasContractsAssigned` function.
    - at line 669 in `getIndexByAuthorizationAddress` function.
    - at line 257 in `_changeAuthorizationOwnership` function.
    - at line 277 in `_removeFromAuthorizationArray` function.
    - at line 353 in `mustBeAuthorizedHolders` function.

  The issues have been fixed and are not present in the latest version of the code.
- Consider declaring parameters with `string` type as `calldata` instead of `memory` all over the code (except the constructor) to avoid extra copy.

  The issues have been fixed and are not present in the latest version of the code.

Project management (fixed)

- `package-lock.json` is in `.gitignore`. Best practice is to commit it to the repository.
- Some tools are added to the `dependencies` list rather than in `devDependencies`.

The issues have been fixed and are not present in the latest version of the code.

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Daria Korepanova, Security Engineer
Ivan Gladkikh, Junior Security Engineer
Irina Vikhareva, Project Manager

June 3, 2022