BankX Audit

August 2022

By CoinFabrik

Introduction

CoinFabrik was asked to audit the contracts for the BankX project. First we will provide a summary of our discoveries, and then we will show the details of our findings.

Scope

The audited files are from the git repository located at https://github.com/Lance-Parker/BankX git repository, commit `e99ff5f8cfe5697f2ddae7b1eb90cc3176659cf4`, and in the https://github.com/Lance-Parker/CD git repository, commit `14a850049458348e6d4b1e8b2ce1e97ba678ca02`.

The audited files are:

- `BankX/contracts/BankX/BankXToken.sol`: Staking token implementation
- `BankX/contracts/BankX/BankX_premint.sol`: Wrapper to BankXToken
- `BankX/contracts/XSD/Pools/XSDWETHpool.sol`: XSD/WETH pool
- `BankX/contracts/XSD/Pools/BankXWETHpool.sol`: BankX/WETH pool
- `BankX/contracts/ERC20/ERC20Custom.sol`: Modified ERC20 contract
- `BankX/contracts/XSD/Pools/CollateralPool.sol`: Collateral pool
- `BankX/contracts/XSD/Pools/CollateralPoolLibrary.sol`: Pool utility functions
- `BankX/contracts/Uniswap/Router.sol`: XSD/ETH/BankX Swap contract
- `CD/contracts/interfaces/BankXInterface.sol`: Interface to BankX Token
- `CD/contracts/CD.sol`: Certificate of Deposit main contract
- `CD/contracts/GlobalsAndUtility.sol`: Utility functions and global variables
- `CD/contracts/StakeableToken.sol`: Staking logic

The scope of the audit is limited to those files. No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

Analyses

Without being limited to them, the audit process included the following analyses:

- Arithmetic errors

- Outdated version of Solidity compiler
- Race conditions
- Reentrancy attacks
- Misuse of block timestamps
- Denial of service attacks
- Excessive gas usage
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

Summary of Findings

We found a critical issue, a medium issue and a minor issue. Also, several enhancements were proposed.

Security Issues

ID	Title	Severity	Status
CR-01	Manipulation of Daily Payout	Critical	Acknowledged
ME-01	Solidity Compiler Version	Medium	Resolved
MI-01	Hardcoded Interest Rate And Launch Time	Minor	Resolved

Security Issues Found

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. They must be fixed **immediately**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.

- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of, but can be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**.

Issues Status

An issue detected by this audit can have four distinct statuses:

- **Unresolved**: The issue has not been resolved.
- **Acknowledged**: The issue remains in the code, but is a result of an intentional decision.
- **Resolved**: Adjusted program implementation to eliminate the risk.
- **Partially resolved**: Adjusted program implementation to eliminate part of the risk. The other part remains in the code, but is a result of an intentional decision.
- **Mitigated**: Implemented actions to minimize the impact or likelihood of the risk

Critical Severity Issues

CR-01 Manipulation of Daily Payout

Location:

- `contracts/GlobalsAndUtility.sol:495 (CD)`

The `dailyDataUpdate()` calculates the daily payout taking in account the BankX token total supply amount:

```
rs._allocSupplyCached = bankXContract.totalSupply() + g._lockedXsTotal;
```

And then at `_dailyRoundCalc()`, line 471:

```
rs._payoutTotal = rs._allocSupplyCached * 10000 / 171779836;
```

And then at `_dailyRoundCalcAndStore()`, line 483:

```
dailyData[day].dayPayoutTotal = uint72(rs._payoutTotal);
```

But the bankX supply can be manipulated by any authorized pool, In this way the daily payout can be manipulated, by increasing or decreasing the supply of BankX tokens and calling `dailyDataUpdate()` at appropriate times.

Recommendation
Restrict the `dailyDataUpdate()` function so the payout cannot be manipulated by an external entity.

Status
Acknowledged. According to the development team, this issue will be mitigated because after launch, admin keys will be renounced.

Medium Severity Issues

ME-01 Solidity Compiler Version

All the audited files use the statement:

```
pragma solidity 0.5.13;
```

This causes that very old solidity compiler version being used which may lead into hitting already fixed bugs. Normally this issue is of minor severity but due to the age of the solidity compiler specified (0.5.13) and the amount of bug fixes accumulated since that version, it was upgraded to Medium severity.

Recommendation
It is better to lock to a specific compiler version (for example, `pragma solidity 0.8.13;`) and keep it up to date, fully testing on each upgrade.

Status
Resolved.

Minor Severity Issues

MI-01 Hardcoded Interest Rate And Launch Time

Location:

- `contracts/GlobalsAndUtility.sol:103,471 (CD)`

The Launch date is hardcoded:

```
uint256 internal constant LAUNCH_TIME = 1575331200;
```

Also, the interest rate used to calculate the daily payout is a hard-coded constant:

```
  rs._payoutTotal = rs._allocSupplyCached * 10000 / 171779836;
```

Recommendation

The `LAUNCH_TIME` constant could be a variable initialized at contract initialization. Also, a mechanism to update the interest rate should be implemented.

Status

Resolved.

Enhancements

These items do not represent a security risk. They are best practices that we suggest implementing.

Table

ID	Title	Status
EN-01	SafeMath Not Needed in Solidity >=0.8	Implemented
EN-02	Lack of Zero-Address Verification	Implemented
EN-03	Different Reentrancy-Guard Methods	Implemented

Details

EN-01 SafeMath not needed in Solidity >=0.8

Location:

- `contracts/BankXToken.sol:2`
- `contracts/BankX_premint.sol:2`

Solidity version 0.8 and over automatically reverts on integer overflows, not needing the `SafeMath.sol` contract anymore.

Recommendation

Remove the `SafeMath.sol` contract from the imports section and refactor the code appropriately.

Status
Implemented.

EN-02 Lack of zero-address Verification

Location:

- `contracts/BankXWETHpool.sol:223`

A zero-check on the new address should be added to the function `setSmartContractOwner()`.

Status
Implemented.

EN-03 Different Reentrancy-Guard Methods

Location:

- `contracts/CollateralPool.sol:69`

CollateralPool.sol uses a custom lock, while XSDWETHpoo.sol and BankXWETHpool.sol uses OpenZepellins's reentrancy guards, which is a better approach.

Recommendation
Replace custom guard code for OpenZepellins's reentrancy guards.

Status
Implemented.

Other Considerations

Arbitrary Bonus Creation

Location:

- `contracts/NFTBonus.sol:22 (CD)`

Any user can get an arbitrarily big bonus by setting any number of OwnerIDs using `NFTBonus.sol:setOwnerIds()`. The function `stackStartBonusXs()` then will return an arbitrarily big bonus using this operation:

```
return bonusXs + bonusXs * nftBonusContract.getNftsCount(msg.sender) / 10;
```

Here, the `getNftsCount(msg.sender)` can return a multiplier controlled by the attacker, so bonusXs can be arbitrarily big. This is a critical bug as it allows stealing

tokens from the contract but as this contract was out-of-scope for this audit, it was not added to the list of security issues, however it's documented in this section.

Recommendation

Restrict the NFTBonus contract so only the owner can add or remove NFT ids.

Status

BankX responded that this was a test contract and because of that, it has no security impact.

Centralization

The contracts `Pools/XSDWETHpool.sol`, `Pools/BankXWETHpool.sol` and others use the Chainlink service as the price oracle.

Using an average or a backup Oracle solution might mitigate centralization, but are more costly, and security of a multi-oracle solution is not guaranteed.

Oracles are complex and approaches to decentralize come with security, accuracy and reliability tradeoffs. Chainlink oracles are powered by independent, professional and decentralized node operators that use multiple layers of aggregation - https://blog.chain.link/chainlink-price-feeds-secure-defi/

Upgrades

No upgrade mechanism exists in any contract audited.

Privileged Roles

These are the privileged roles that we identified on each of the audited contracts.

BankX/contracts/BankX/BankXToken.sol

Owner

The Owner can: Set the pool, set the Timelock address, set the Treasure address, set the XSD token address.

Timelock

The Timelock address can: Set the pool, set the Timelock address, set the Treasure address, set the XSD token address.

Pools

The XSD pools can: Mint new BankX tokens, burn BankX tokens.

BankX/contracts/BankX/BankX_premint.sol

Owner

The Owner can: Set the BankX token address, set the XSD token address.

BankX/contracts/XSD/Pools/BankXWETHpool.sol

Owner

The Owner can: Initialize the contract, change the owner address.

BankX/contracts/XSD/Pools/CollateralPool.sol

Owner

The Owner can: set the pool parameters, set the PID controller address.

BankX/contracts/Uniswap/Router.sol

Owner

The Owner can: Initialize the contract, change the owner address.

Treasury

The Treasury address can: Add new liquidity Tokens, add liquidity ETHs.

CD/contracts/GlobalsAndUtility.sol

Owner

The Owner can: Set the BankX contract address, set the NFTBonus contract address.

Changelog

- 2022-08-05 – Initial report based on commit `e99ff5f8cfe5697f2ddae7b1eb90cc3176659cf4` for repository https://github.com/Lance-Parker/BankX and `14a850049458348e6d4b1e8b2ce1e97ba678ca02` for the repository https://github.com/Lance-Parker/CD.
- 2022-08-23 – Fixes checked on commit `f127336bd6a8390c52fa6b2a93bde1306f529745 for repository` https://github.com/Lance-Parker/BankX and E096bbce99cafb6e363b16dffec1ce3604c837d6 for repository https://github.com/Lance-Parker/CD.

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the BankX project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.