Quantstamp Security Assessment Certificate

October 31st 2022 — Quantstamp Verified

StormX Token V2

This audit report was prepared by Quantstamp, the leader in blockchain security.

Executive Summary

Type	Ethereum
Auditors	Ed Zulkoski, Senior Security Engineer Martinet Lee, Senior Research Engineer Alex Murashkin, Senior Software Engineer
Timeline	2022-10-03 through 2022-10-06
EVM	Paris
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	README.md
Documentation Quality	High
Test Quality	High

Source Code

Repository	Commit
stormxio/stmx-token	8874148 initial audit
stormxio/stmx-token	8210787 fixes
stormxio/stmx-token	da8f664 fixes

Total Issues	**7** (6 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	**4** (3 Resolved)
Informational Risk Issues	**3** (3 Resolved)
Undetermined Risk Issues	0 (0 Resolved)

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
Undetermined	The impact of the issue is uncertain.

Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Fixed	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

The StormX V2 Token code and documentation is well-written. Only a few Low and Informational level issues were noted during the audit, and some of these have already been acknowledged via documentation. We recommend reviewing the report findings before using the code in production.
Update: All issues have been fixed, mitigated, or acknowledged as of commit `8210787`.
Update 2: Minor changes have been made to `Staking.sol` as of commit `da8f664`. No new issues were found.

ID	Description	Severity	Status
QSP-1	Privileged Roles and Ownership	Low	Acknowledged
QSP-2	Missing Input Validation	Low	Fixed
QSP-3	Owner Can Prevent Users From Unstaking Tokens Within Cooldown Period	Low	Fixed
QSP-4	Owner Can Front-Run to Update Penalty and Cooldown Period	Low	Mitigated
QSP-5	Allowance Double-Spend Exploit	Informational	Mitigated
QSP-6	Only-Owner Upgradability Is Not Tested	Informational	Fixed
QSP-7	`IERC20.sol` and `IERC20Upgradeable.sol` Are Equivalent	Informational	Fixed

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Slither v0.8.3

Steps taken to run the tools:

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Findings

QSP-1 Privileged Roles and Ownership

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `Staking.sol`, `STMX.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract.
In `Staking.sol`, the owner can change the staking cooldown and penalty values. However, this does not affect users that have already staked. Moreover, the owner is entitled to withdraw all new tokens from the `Convert` contract as well as pre-mint any amount of tokens into its own account.
The owner of `STMX` may also upgrade the contract arbitrarily.

Recommendation: This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to the owner. Take additional measures for protecting the admin account (e.g., using a multisig).

Update: From the StormX team: "StormX will do its best to take extra protective measures, such as using multi-sig."

QSP-2 Missing Input Validation

Severity: *Low Risk*

Status: Fixed

Related Issue(s): SWC-123

Description: It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. Specifically, in the following functions arguments of type `address` may be initialized with value `0x0`:

1. `Convert.constructor()` does not ensure that `oldToken_` and `newToken_` are non-zero.
2. `Staking.constructor()` does not ensure that `token_` and `treasury_` are non-zero.
3. `Staking.setTreasury()` does not ensure that `newTreasury` is non-zero.

Recommendation: We recommend adding the relevant checks.

QSP-3 Owner Can Prevent Users From Unstaking Tokens Within Cooldown Period

Severity: *Low Risk*

Status: Fixed

File(s) affected: `Staking.sol`

Description: OpenZeppelin's ERC20 token implementation prevents transfer to and from the `address(0)`. As a result, , if `treasury` is set to `address(0)` in `Staking.sol`, then `unstake()` will fail when the a user tries to unstake within the cooldown period as the function attempts to transfer to `treasury`.

Recommendation: Prevent `treasury` from being set to `address(0)` in both the constructor and `setTreasury()`.

QSP-4 Owner Can Front-Run to Update Penalty and Cooldown Period

Severity: *Low Risk*

Status: Mitigated

File(s) affected: `Staking.sol`

Description: The current design allows the owner to update penalty up to `100%` and no limitation on the cooldown period. Hence it should be noted that users' funds could be locked permanently if they were being front-runned.

Recommendation: Consider adding reasonable limitations in the configuration. Communicate the behaviour clearly to the users.

Update: StormX changed the `setCooldown()` method so that the upper limit is now 365 days. This is also now explained in the README file. However, the owner can still front-run with 100% penalty.

QSP-5 Allowance Double-Spend Exploit

Severity: *Informational*

Status: Mitigated

File(s) affected: `STMX.sol`

Description: As it presently is constructed, the contract is vulnerable to the allowance double-spend exploit, as with other ERC20 tokens.

Exploit Scenario: 1. Alice allows Bob to transfer `N` amount of Alice's tokens (`N>0`) by calling the `approve()` method on `Token` smart contract (passing Bob's address and `N` as method arguments)

1. After some time, Alice decides to change from `N` to `M` (`M>0`) the number of Alice's tokens Bob is allowed to transfer, so she calls the `approve()` method again, this time passing Bob's address and `M` as method arguments
2. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls the `transferFrom()` method to transfer `N` Alice's tokens somewhere
3. If Bob's transaction will be executed before Alice's transaction, then Bob will successfully transfer `N` Alice's tokens and will gain an ability to transfer another `M` tokens

4. Before Alice notices any irregularities, Bob calls `transferFrom()` method again, this time to transfer `M` Alice's tokens.

Recommendation: The exploit (as described above) is mitigated through use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance()` and `decreaseAllowance()`. Furthermore, we recommend that developers of applications dependent on `approve()` / `transferFrom()` should keep in mind that they have to set allowance to 0 first and verify if it was used before setting the new value.

QSP-6 Only-Owner Upgradability Is Not Tested

Severity: *Informational*

Status: Fixed

File(s) affected: `STMX.sol`

Description: The README has a requirement: "Only the owner can upgrade the token implementation". However, it is not tested in the provided tests.

Recommendation: Add a test verifying the upgradeability requirement.

QSP-7 `IERC20.sol` and `IERC20Upgradeable.sol` Are Equivalent

Severity: *Informational*

Status: Fixed

File(s) affected: `Convert.sol`

Description: The two interfaces `IERC20` and `IERC20Upgradeable` are equivalent as they are merely referencing the definition of ERC20 standard. Hence importing them both would only increase the contract size and no other meaningful use.

Recommendation: Use either `IERC20.sol` or `IERC20Upgradeable.sol` and remove the other.

Automated Analyses

Slither

Slither noted that within `Convert.withdraw()`, the return value of `newToken.transfer(owner(),reserves)` is not checked. However, since the `newToken` contract is internally developed, and `withdraw()` is an `onlyOwner` function, this is a non-issue.

Adherence to Best Practices

1. In `Convert.sol`, we would suggest a more generic method of withdrawing tokens of any kind (with a token address as a parameter).

Test Results

Test Suite Results

```
Convert
  Deploying
     ✓ reverts if deployment is using zero-address oldToken
     ✓ reverts if deployment is using zero-address newToken
  Convert
     ✓ converts the tokens and emits "Converted" events (47ms)
     ✓ reverts in case of closed contract
     ✓ reverts in case of not enought old token balance
     ✓ reverts in case of failed transfer from the old token
     ✓ reverts in case of not enough reserves
  Close
     ✓ allows the owner to close the contract and emits "Closed" event
     ✓ prevents non-owner from setting the cooldown
  Withdraw
     ✓ allows the owner to withdraw remaining tokens when contract is closed
     ✓ prevents the owner from withdrawing the tokens when contract is not closed
     ✓ prevents non-owner from withdrawing the tokens

STMX
  ERC20
     ✓ has correct name
     ✓ has correct symbol
     ✓ has correct number of decimals
     ✓ has correct total supply
     ✓ has correct owner assigned
  Transfers
     ✓ sends transfer successfully
     ✓ uses transferFrom successfully
     ✓ uses transfers successfully
     ✓ reverts if input lengths do not match in transfers
     ✓ reverts if any transfer fails
  Upgradability
     ✓ should correctly upgrade the contract by the owner
     ✓ prevents non-owner from upgrading the contract
     ✓ reverts if initialize() called more than once

Staking
  Deploying
     ✓ reverts if deployment is using zero-address token
     ✓ reverts if deployment is using zero-address treasury
  Staking
     ✓ has correct owner assigned
     ✓ allows to stake the tokens
     ✓ allows to unstake the tokens and transfer the tokens before the cooldown period (44ms)
     ✓ allows to unstake the tokens and transfer the tokens after the cooldown period (42ms)
     ✓ emits "Staked" event when staking
     ✓ emits "Unstaked" event when unstaking
     ✓ prevents from staking more tokens than the balance
     ✓ prevents from staking zero tokens
     ✓ prevents from unstaking more tokens than already staked
     ✓ prevents from unstaking zero tokens
  Cooldown
     ✓ allows the owner to set the cooldown and emits "CooldownChanged" event
     ✓ prevents non-owner from setting the cooldown
```

```
      ✓ prevents from setting cooldown period over the maximum 365 days
   Penalty
      ✓ allows to calculate the penalty before actually unstaking
      ✓ allows the owner to set the penalty and emits "PenaltyChanged" event
      ✓ prevents non-owner from setting the penalty
      ✓ prevents the owner from overflowing the penalty
   Treasury
      ✓ allows the owner to set the treasury and emits "TreasuryChanged" event
      ✓ prevents non-owner from setting the treasury
      ✓ reverts when setting zero-address treasury


 47 passing (4s)
```

Code Coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	90.32	100	100	
Convert.sol	100	90.91	100	100	
STMX.sol	100	100	100	100	
Staking.sol	100	88.89	100	100	
All files	**100**	**90.32**	**100**	**100**	

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

```
e81011e5b76b2320bfe52b122ad16e3ac58bb92ed25c967460f6e08093c2d705 ./contracts/STMX.sol

3a2a58ea76fc3cfb677951ec0f38581e13cd6b2c16d52f1d9064854247c8c6d4 ./contracts/Convert.sol

3f9fb9b7221b28095ccd1816236f468723264095aa7f6ed5f9d49ad56818e989 ./contracts/Staking.sol
```

Tests

```
c2da87bda24977fb9e9614782da5a9c3bebde212c13d11a2cdf0978935b09970 ./Staking.spec.ts

9a7a4551dfe30c26a6df1114c846c4841eebc2d5fd60639b9f99bde96810af4b ./STMX.spec.ts

96bcdf8d3c1bf11a91752eda3cc93d7892dce47cb3cfc20684b6b9c3a4d75730 ./shared.ts

937f47428b2e5ee918f4145652511a67b50555e808d49e46c8b8874bd90acbd3 ./types.ts

d298e20965f7a02205df466437339e575d519524b08f2d32fb3bdc0e4d26857b ./Convert.spec.ts

d1a90bd2e782670e5357704e152d7814dc00c8e2d554252e3f1ee709b0482988 ./test/STMXv3.sol

440bb25b2a4c91c39a255cb5cf4c90f77a061c0b0f02076ebc676845a99ecb38 ./test/TestERC20.sol
```

Changelog

- 2022-10-07 - Initial report
- 2022-10-10 - Updated report based on commit `8210787`
- 2022-10-31 - Updated report based on commit `da8f664`

About Quantstamp

Quantstamp is a global leader in blockchain security backed by Pantera, Softbank, and Commonwealth among other preeminent investors. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its white glove security and risk assessment services.

The team consists of web3 thought leaders hailing from top organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Many of the auditors hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 250 audits and secured over $200 billion in digital asset risk from hackers. In addition to providing an array of security services, Quantstamp facilitates the adoption of blockchain technology through strategic investments within the ecosystem and acting as a trusted advisor to help projects scale.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Aave, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT