



January 6th 2021 — Quantstamp Verified

flexUSD

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

Executive Summary

Type	Stable Coin				
Auditors	Poming Lee, Research Engineer Jake Goh Si Yuan, Research Engineer Shunsuke Tokoshima, Software Engineer				
Timeline	2020-10-29 through 2021-01-06				
EVM	Muir Glacier				
Languages	Solidity				
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review				
Specification	None				
Documentation Quality	<div><div></div></div> Medium				
Test Quality	<div><div></div></div> Low				
Source Code	<table><tr><td>Repository</td><td>Commit</td></tr><tr><td>flexAssets</td><td>79ce4e1</td></tr></table>	Repository	Commit	flexAssets	79ce4e1
Repository	Commit				
flexAssets	79ce4e1				

Goals

- Formal Audit

Total Issues	11 (4 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	3 (1 Resolved)
Low Risk Issues	0 (0 Resolved)
Informational Risk Issues	8 (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).
Resolved	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

Throughout this audit, we have found nine potential issues of various severities, including three medium-risk issues and six informational-level. Plus, we have made multiple best practice recommendations. They are listed in the [Adherence to Best Practices](#) section and [Adherence to Specification](#) section. We recommend addressing the findings before going to production.

**** 2020-12-14 update **:** during this first reaudit, CoinFlex team has either brought the status of findings into fixed or acknowledged, leaving only one finding unresolved. The folder structure of the project has been modified, and a `test` folder was added. However, 13 out of 16 tests did not pass. In addition, we found two informative-severity issues in this round of reaudit.

**** 2021-01-06 update **:** during this second reaudit, CoinFlex team has either brought the status of findings into fixed or acknowledged, leaving two finding unresolved. In addition, all tests were removed.

ID	Description	Severity	Status
QSP-1	Allowance Double-Spend Exploit	^ Medium	Acknowledged
QSP-2	Lack of Tests	^ Medium	Fixed
QSP-3	Misleading Usage of <code>approve()</code> in <code>transferFrom()</code>	^ Medium	Acknowledged
QSP-4	Clone-and-Own	○ Informational	Acknowledged
QSP-5	Unlocked Pragma	○ Informational	Fixed
QSP-6	Centralization of Power	○ Informational	Acknowledged
QSP-7	Unknown Contract Artifact in Deployment Code	○ Informational	Fixed
QSP-8	Missing Input Validation	○ Informational	Fixed
QSP-9	Hardcoded Infura Key	○ Informational	Acknowledged
QSP-10	Upgradeable contract	○ Informational	Unresolved
QSP-11	Modifier <code>ispause()</code> would not revert when the contract is not paused	○ Informational	Unresolved

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](#) v0.6.12

Steps taken to run the tools:

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

Findings

QSP-1 Allowance Double-Spend Exploit

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `FLexUSD.sol`

Related Issue(s): [SWC-114](#)

Description: As it presently is constructed, the contract is vulnerable to the [allowance double-spend exploit](#), as with other ERC20 tokens.
** 2020-12-14 update **: Two functions `increaseAllowance` and `decreaseAllowance` are added and recommended to be used by end users.

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)
2. 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
3. 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
4. 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
5. 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 the use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance` and `decreaseAllowance`. The Quantstamp team recommends that the existence of `increaseApproval` and `decreaseApproval` be clearly communicated to the users.

QSP-2 Lack of Tests

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `FLexUSD.sol`

Description: The current implementation lacks test scripts to verify whether the implementation works as expected. Writing test code is the minimal QA effort expected from any project willing to identify errors and/or flaws in the implementation.

Recommendation: It is highly recommended to write some test code.

QSP-3 Misleading Usage of `approve()` in `transferFrom()`

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `FLexUSD.sol`

Description: In `transferFrom()`, the allowance of `msg.sender` is expected to be reduced after the transfer operation takes place via the proxy of the `msg.sender`, according to ERC20 standard. Given that this was one of the requirements of the specifications, we will hold this codebase to this standard. In this case, the allowance was reduced with `_approve()` method. This is highly unusual, as it is not a typical setter method for the allowance mapping as one would expect, given that the name `_approve()` suggesting that this is an additional action. At the same time, `_approve()` emits two events which could be very misleading to an off-chain observer, who may think or regard that a new allowance was set instead of a transfer event happening which is what is taking place.
** 2020-12-14 update **: CoinFlex team decided to keep the calling of the function `_approve` and remove one of the event emits.

Recommendation: It is recommended to directly modify the `_allowances` mapping value without using the `_approve` method.

QSP-4 Clone-and-Own

Severity: *Informational*

Status: Acknowledged

File(s) affected: `FLexUSD.sol`

Description: The clone-and-own approach involves copying and adjusting open source code at one's own discretion. From the development perspective, it is initially beneficial as it reduces the amount of effort. However, from the security perspective, it involves some risks as the code may not follow the best practices, may contain a security vulnerability or may include intentionally or unintentionally modified upstream libraries.
** 2020-12-14 update **: the CoinFlex team does not regard QSP 4 to be related to security and stated that "it is useful to have a public repo with flattened code so that interested parties may compare that code directly to the necessarily flattened version available on a verified contract on Etherscan. In this context, it is a feature".

Recommendation: Rather than the clone-and-own approach, a good industry practice is to use the Truffle framework for managing library dependencies. This eliminates the clone-and-own risks yet allows for following best practices, such as using libraries.

QSP-5 Unlocked Pragma

Severity: *Informational*

Status: Fixed

File(s) affected: [FlexUSD.sol](#)

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.6.*`. The caret (^) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".

Recommendation: For consistency and to prevent unexpected behavior in the future, it is recommended to remove the caret to lock the file onto a specific Solidity version.

QSP-6 Centralization of Power

Severity: *Informational*

Status: Acknowledged

File(s) affected: [FlexUSD.sol](#)

Description: Smart contracts often have special roles to designate entities with privileges to make modifications to smart contracts. In this project, the admin can arbitrarily mint and burn tokens from/to any account and pause any accounts' token transfer transactions.

**** 2020-12-14 update **:** CoinFlex team stated that it will be added to the documentation surrounding the contract.

Recommendation: This centralization of power needs to be made clear to the users in the documentation. Users should be made aware of any risks associated with such centralization.

QSP-7 Unknown Contract Artifact in Deployment Code

Severity: *Informational*

Status: Fixed

Description: In [2_deploy_smartcontract.js](#), an unknown artifact called `CURRYSWAP` is referred to.

Recommendation: It is recommended to refer to [FlexUSD](#) instead of `CURRYSWAP` and remove the unnecessary deployment of `ERC20` in [2_deploy_smartcontract.js](#)

QSP-8 Missing Input Validation

Severity: *Informational*

Status: Fixed

File(s) affected: [FlexUSD.sol](#)

Description: The input parameter `account` of `TransferOwnership()` is missing validation. Providing incorrect values may result in a later discovery that the smart contract is not working as intended (e.g., Unupdatable multiplier).

Recommendation: It is recommended to add a check that ensures `account` is different from `0x0`.

QSP-9 Hardcoded Infura Key

Severity: *Informational*

Status: Acknowledged

Description: In [truffle-config.js](#) and [buidler.config.js](#), Infura API key is hardcoded.

**** 2020-12-14 update **:** CoinFlex team stated that these files will not be available in the public repo for the contracts.

Recommendation: It is recommended to include it in [.secrets.json](#).

QSP-10 Upgradeable contract

Severity: *Informational*

Status: Unresolved

Description: `contracts\fUSD.sol`: The contract is upgradeable and the implementation of code is not freeze and could be modified in the future.

Recommendation: Should add this information to [README.md](#) and make this information explicit to the end users.

QSP-11 Modifier `ispaused()` would not revert when the contract is not paused

Severity: *Informational*

Status: Unresolved

Description: `contracts\fUSD.sol`: The modifier `ispaused()` would not revert when the contract is not paused. It is used in multiple functions; for instance, function `setMultiplier`, `increaseAllowance` and `decreaseAllowance`.

Recommendation: Make sure the modifier is implemented correctly and added to the right functions. If this is the case, consider rename the modifier to `isnotpaused`.

[Automated Analyses](#)

Slither

Slither detected 23 items. However, all the items were either of:

- minor best practices such as the naming convention and the access level of functions (i.e. gas optimization context)
- some false-positives.

Adherence to Specification

- `setMultiplier()` has a validation on L264 that the multiplier should be greater than the previous multiplier. However, given that the comparison is `_multiplier >= multiplier`, it means that this will pass when `_multiplier == multiplier` violating the rule and making an ineffective change. It is recommended to set the condition to `_multiplier > multiplier`

Adherence to Best Practices

- It is recommended to use a linter for Solidity (e.g. solhint). There are some inconsistencies in Coding style such as the spacing. For example, `FlexUSD.sol` L440 has 5 spaces at the beginning of the line whereas L436 has 7 spaces. [In general](#), it is recommended to use 4 spaces per indentation level.
- `FlexUSD.sol`: It is recommended to inherit `ERC20Pausable.sol` to implement `Pausing` functionality.
- `FlexUSD.sol`: It is recommended to rename modifier `ispaused` to a less misleading name such as `whenNotPaused` if it is difficult to use `ERC20Pausable.sol`.
- `FlexUSD.sol`: It is recommended to define `dec` as a `constant` and explicitly set its visibility level.
- `FlexUSD.sol` L269: It is recommended to correct the inconsistent usage of `uint` and `uint256`.
- `FlexUSD.sol` L276: input parameter of `inputTotalSupply` should read `inputTotalSupply`.
- `FlexUSD.sol` L298: It is recommended to correct the redundant parentheses around `around.mul(dec)`
- `FlexUSD.sol` L312-314: `allowance()` calculates `externalAmt`, instead of `internalAmt`. It is recommended to rename the variable to `externalAmt`
- Mixed case is [conventionally](#) used for function/modifier names. It is recommended to rename `AddToBlacklist()`, `TransferOwnership()`, `RemoveFromBlacklist()`, and `Notblacklist()`.
- `FlexUSD.sol` L430: This line is not needed because `onlyAdmin()` is applied to this function.
- `FlexUSD.sol` L445: As `getpause` is a boolean, it is recommended to replace `getpause == false` with `!getpause`.

Test Results

Test Suite Results

No tests provided.

Code Coverage

No tests provided.

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

63be8650feaa4b370a71c12302da36664e3e4a1b3e5ddfd155b7e3f61375020 . /202101B-reaudit-79ce4e1/fUSD.sol
cb6b780cc4a5a7aaa6706b116ac42f704c12c1f9a1cccbaec1d19e339f5d018a . /202101B-reaudit-79ce4e1/fUSDStorage.sol

Changelog

- 2020-11-09 - Initial report
- 2020-12-14 - reaudit report
- 2021-01-06 - reaudit report

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure blockchain platforms at scale using computer-aided reasoning tools, with a mission to help boost the adoption of this exponentially growing technology.

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected \$5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp, Inc. (Quantstamp). Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on the website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.