Quantstamp Security Assessment Certificate

February 8th 2023 — Quantstamp Verified

FiNANCiE Token

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

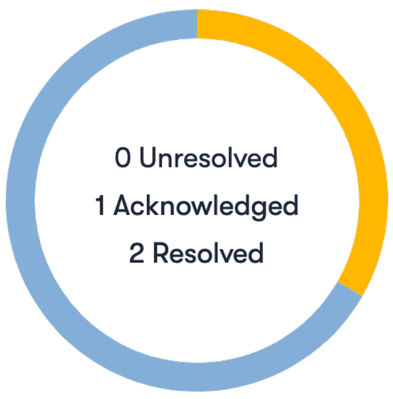
Executive Summary

Type	ERC20 Token
Auditors	Guillermo Escobero, Security Auditor
Timeline	2022-03-28 through 2022-04-07
EVM	Arrow Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	FiNANCiE Whitepaper ver.0.2 (unpublished)
Documentation Quality	Low
Test Quality	Undetermined

Source Code

Repository	Commit
fnct-contracts	212c853

Total Issues	**3** (2 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	0 (0 Resolved)
Informational Risk Issues	**2** (1 Resolved)
Undetermined Risk Issues	**1** (1 Resolved)



ALL ISSUES ADDRESSED

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

FNCT token was audited without any security vulnerabilities found. It implements the ERC20 standard with the correct configuration, using well-known libraries in the industry. Three informational issues were identified. It is recommended to improve code documentation, i.e. including a README file with the project's purpose.

Fix review: The FiNANCie team addressed all the findings. Quantstamp recommends creating public technical documentation about the purpose of this token if it is part of a system. In that case, it is recommended to create integration tests.

ID	Description	Severity	Status
QSP-1	Allowance Double-Spend Exploit	○ Informational	Acknowledged
QSP-2	Unlocked Pragma	○ Informational	Fixed
QSP-3	Ownable Contract Not Used	? Undetermined	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.

DISCLAIMER:
Quantstamp was requested to only audit `FNCT_flatten.sol`.

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
- Security issues in the compiler version
- Unsafe use of fallback functions
- Unsafe use of `tx.origin`
- Compliance with ERC20 standard
- Variables shadowing
- Illegal function calls
- Unsafe use of low-level functions
- Unsafe use of `selfdestruct`
- Wrong implementation of PRNG

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 v0.8.2

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: *Informational*

Status: Acknowledged

File(s) affected: `contracts/fnct/FNCT_flatten.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)
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 use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance()` and `decreaseAllowance()`.
Pending community agreement on an ERC standard that would protect against this exploit, 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. Teams who decide to wait for such a standard should make these recommendations to app developers who work with their token contract.

Update: The FiNANCiE team replied: "*As with other ERC20 tokens, we will not(and can't) resolve the issue with Solidity code, so we'd inform the fact of this potential issue and "0- reset-before-new-approval" workaround to our token's developer.*"

QSP-2 Unlocked Pragma

Severity: *Informational*

Status: Fixed

File(s) affected: `contracts/fnct/FNCT_flatten.sol`

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.8.*`. 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.

Update: The FiNANCie team stated that will use `0.8.3` version when compiling `FNCT` contract.

QSP-3 Ownable Contract Not Used

Severity: *Undetermined*

Status: Fixed

File(s) affected: `contracts/fnct/FNCT_flatten.sol`

Description: `FNCT` is labeled as `Ownable`. However, the modifier `onlyOwner` is not used. This can confuse users and will waste gas.

Recommendation: Please clarify why Ownable is needed. If not needed, remove `Ownable` and the related `import` statement.

Update: The issue was fixed by removing the `Ownable` library. The FiNANCie team addressed the issue in commit `ad42d6b`.

Automated Analyses

Slither

We have run the latest version of the Slither analyzer on this repository's Solidity code. The tool has identified 26 issues in total, which were filtered out as false positives.

```
ERC20PresetFixedSupply.constructor(string,string,uint256,address).name (FNCT_flatten.sol#678) shadows:
    - ERC20.name() (FNCT_flatten.sol#286-288) (function)
    - IERC20Metadata.name() (FNCT_flatten.sol#211) (function)
ERC20PresetFixedSupply.constructor(string,string,uint256,address).symbol (FNCT_flatten.sol#679) shadows:
    - ERC20.symbol() (FNCT_flatten.sol#294-296) (function)
    - IERC20Metadata.symbol() (FNCT_flatten.sol#216) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Context._msgData() (FNCT_flatten.sol#25-27) is never used and should be removed
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (FNCT_flatten.sol#8) allows old versions
Pragma version^0.8.0 (FNCT_flatten.sol#36) allows old versions
Pragma version^0.8.0 (FNCT_flatten.sol#114) allows old versions
Pragma version^0.8.0 (FNCT_flatten.sol#200) allows old versions
Pragma version^0.8.0 (FNCT_flatten.sol#230) allows old versions
Pragma version^0.8.0 (FNCT_flatten.sol#615) allows old versions
Pragma version^0.8.0 (FNCT_flatten.sol#655) allows old versions
Pragma version^0.8.0 (FNCT_flatten.sol#690) allows old versions
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

renounceOwnership() should be declared external:
    - Ownable.renounceOwnership() (FNCT_flatten.sol#84-86)
transferOwnership(address) should be declared external:
    - Ownable.transferOwnership(address) (FNCT_flatten.sol#92-95)
name() should be declared external:
    - ERC20.name() (FNCT_flatten.sol#286-288)
symbol() should be declared external:
    - ERC20.symbol() (FNCT_flatten.sol#294-296)
decimals() should be declared external:
    - ERC20.decimals() (FNCT_flatten.sol#311-313)
totalSupply() should be declared external:
    - ERC20.totalSupply() (FNCT_flatten.sol#318-320)
balanceOf(address) should be declared external:
    - ERC20.balanceOf(address) (FNCT_flatten.sol#325-327)
transfer(address,uint256) should be declared external:
    - ERC20.transfer(address,uint256) (FNCT_flatten.sol#337-341)
approve(address,uint256) should be declared external:
    - ERC20.approve(address,uint256) (FNCT_flatten.sol#360-364)
transferFrom(address,address,uint256) should be declared external:
    - ERC20.transferFrom(address,address,uint256) (FNCT_flatten.sol#382-391)
increaseAllowance(address,uint256) should be declared external:
    - ERC20.increaseAllowance(address,uint256) (FNCT_flatten.sol#405-409)
decreaseAllowance(address,uint256) should be declared external:
    - ERC20.decreaseAllowance(address,uint256) (FNCT_flatten.sol#425-434)
burn(uint256) should be declared external:
    - ERC20Burnable.burn(uint256) (FNCT_flatten.sol#629-631)
burnFrom(address,uint256) should be declared external:
    - ERC20Burnable.burnFrom(address,uint256) (FNCT_flatten.sol#644-647)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
. analyzed (8 contracts with 77 detectors), 26 result(s) found
```

Adherence to Specification

As the token supply is preset before deployment, it is a common practice to declare it as a constant state variable or hardcoded in the constructor to improve readability.
Update: Reply from the FiNANCiE team: *We also understand it should be hardcoded, but unfortunately the token supply is not fixed yet on our business plan. However, the "Fixed and Audited code" is required before the plan fixed, that's why we could not specify the value as constants or hardcoded-number.*

Code Documentation

There is no README file for this project. Each repository should contain a file named README.md that contains the key points and purpose of the project. Also, it should include basic instructions on how to run the test suite and any prerequisites for running tests. For each of these prerequisites also specify the versions supported/tested.

Update: The issue was acknowledged.

Adherence to Best Practices

All recommendations are discussed in previous sections.

Test Results

Test Suite Results

No tests were provided.

Code Coverage

No code coverage was 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

`7ebbf95e7a192f59dda33076d4bb3584f80936e6bdc8204cfcdf2102a08f30de ./contracts/fnct/FNCT_flatten.sol`

Changelog

- 2022-03-31 - Initial report
- 2022-04-07 - Updated the report according to commit `ad42d6b`.

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers 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 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

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

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 or other making available of the report to you by Quantstamp.

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. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites&aspo; 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 any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims 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. You agree that your access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. 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. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or 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, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. 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.