

PlusCoin Token

October 15th 2019 — Quantstamp Verified

This smart contract audit was prepared by Quantstamp, the protocol for securing smart contracts.

1 (O Resolved)

2 (O Resolved)

Ο

Executive Summary

Type	Token Contract Audit		Severity Categories		
Auditors	Yohei Oka, Forward Deploy John Bender, Senior Resear Michael Teixeira, Software I Martin Derka, Senior Resea	rch Engineer Developer	A High	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.	
Timeline	2018-09-28 through 2018-1	10-02	^ Medium	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	
Languages	Solidity, Javascript				
Methods				financial impact.	
Specification	<u>PlusCoin Whitepaper</u>		∨ Low	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.	
Source Code	Repository	Commit			
	<u>NPLC</u>	<u>12fbe86</u>	 Informational 	The issue does not post an immediate risk, but is relevant	
				to security best practices or Defence in Depth.	
Total Issues	3 (O Resolved)		Undetermined	The impact of the issue is uncertain.	
High Risk Issues	Ο				
Medium Risk Issues	0				

3 issues

Goals

• 2018-10-09 - First Version

Changelog

Low Risk Issues

Informational Risk Issues

Undetermined Risk Issues

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

• Mishandled exceptions and call stack limits

Quantstamp Audit Breakdown

• Transaction-ordering dependence • Timestamp dependence

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

- Integer overflow / underflow • Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights

• Unsafe external calls

- Access control • Centralization of power
- Business logic contradicting the specification • Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- The Quantstamp auditing process follows a routine series of steps: Code review that includes the following Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart

describe.

those test cases.

Methodology

Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp

- Testing and automated analysis that includes the following: 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
 - Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute. 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.
- **Toolset** The below notes outline the setup and steps performed in the process of this audit.

Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

- Setup
- Tool Setup:

Steps taken to run the tools:

• Truffle v4.1.12

• Ganache v1.1.0

• MAIAN commit sha: ab387e1

• Oyente v1.2.5 • Mythril v0.2.7

> 1. Installed Truffle: npm install -g truffle 2. Installed Ganache: npm install -g ganache-cli

3. Installed the solidity-coverage tool (within the project's root directory): npm install --save-dev solidity-coverage 4. Ran the coverage tool from the project's root directory: ./node_modules/.bin/solidity-coverage

5. Flattened the source code using truffle-flattener to accommodate the auditing tools.

7. Ran the Mythril tool on each contract: myth -x path/to/contract 8. Installed the Oyente tool from Docker: docker pull luongnguyen/oyente

9. Migrated files into Oyente (root directory): docker run -v \$(pwd):/tmp - it luongnguyen/oyente 10. Ran the Oyente tool on each contract: cd /oyente/oyente && python oyente.py /tmp/path/to/contract

6. Installed the Mythril tool from Pypi: pip3 install mythril

12. Ran the MAIAN tool on each contract: cd maian/tool/ && python3 maian.py -s path/to/contract contract.sol

11. Cloned the MAIAN tool: git clone --depth 1 https://github.com/MAIAN-tool/MAIAN.git maian

Assessment

Findings

Severity: Low **Description:** The token contract features the following centralization characteristic:

Centralization of Power

• The owner can enable and disable transfer-related functionalities at will through the use of the isNotFrozen and isNotFrozenFrom modifier.

tokens. This issue is magnified in that it can be performed by all super admins, not only the contract owner. In the spirit of decentralization, we recommend taking away these functionalities.

Allowance Double-Spend Exploit

passing Bob's address and M as method arguments

communicated clearly to token holders upfront.

Severity: Informational

Description: As it presently is constructed, the contract is vulnerable to the allowance double-spend exploit, as with other ERC20 tokens. An example of an exploit goes as follows:

While not necessarily a vulnerability, if the contract owner's private key is compromised or the owner becomes malicious, then token holders may be unable to transfer their

The PlusCoin team has communicated to Quantstamp that the ability to freeze the token and specific accounts is implemented for AML purposes. This should also be

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

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. 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

Unused Event Severity: Informational

Test Results Test Suite Results

Contract: Administratable addSuperAdmin ✓ contract owner can add a super Admin (45260 gas)

· Block limit: 6721975 gas

257610.97 krw/eth

56.57

59.77

59.75

41.66

68.22

39.72

· # calls

✓ a super admin can not add another super admin (23292 gas) ✓ a non super admin cannot add another super admin (23292 gas) ✓ cannot add a super admin that is already a super admin (68994 gas) removeSuperAdmin

✓ a super admin can not remove another super admin (184329 gas) ✓ a non super admin cannot add another super admin (23292 gas) ✓ cannot add a super admin that is already a super admin (68994 gas) Contract: Freezable freezeAccount ✓ contract owner can freeze token (66246 gas)

PlusCoin

PlusCoin

PlusCoin

PlusCoin

Deployments

```
✓ cannot add a frozen account that is already a frozen account (69994 gas)
Contract: PlusCoin
 constructor
     ✓ validate token minting
 transfer
     ✓ simple transfer case should succeed and change balance (52961 gas)
     ✓ transferFrom case should succeed and change balance (144945 gas)
     ✓ transfer fails when token is frozen (68331 gas)

√ transfer fails if sender is frozen (70829 gas)

√ transfer fails if destination is frozen (71163 gas)
     ✓ transfer fails if destination is not a valid address (22256 gas)

✓ when the spender is the frozen account (70061 gas)

     ✓ when the spender is the frozen token (92521 gas)
```

✓ when the spender is the frozen account(message sender) (95744 gas)

✓ a non super admin can not freeze another account (23952 gas)

✓ contract owner can remove a super Admin (60407 gas)

✓ contract owner can add a frozen account (45912 gas)

✓ non owner can not freeze token (22379 gas)

√ increase allowance (78722 gas) √ decrease allowance (78986 gas)

· approve

· freezeAccount

· freezeToken

· transfer

· transferFrom

decreaseAllowance ·

· increaseAllowance ·

· transferOwnership ·

· removeSuperAdmin

standard should make these recommendations to app developers who work with their token contract.

Description: The event Burn (PlusCoin.sol L12) doesn't appear to be used anywhere and should be removed.

✓ change owner and add super admin (115350 gas) ✓ try to drain contract from admin address (27765 gas) Gas 5 gwei/gas Methods Contract Method · Max Ava 45260 · Administratable · addSuperAdmin Administratable · removeSuperAdmin 15147 45912 · · freezeAccount 43917 Freezable · freezeToken 45590 . · addSuperAdmin

			615857	9.2 %	793.26						
Freezable .			930766	13.8 %	1198.88						
PlusCoin ·	3120608	3120672	3120619	46.4 %	4019.53						
27 passing (11s)											
Code Coverage											
The PlusCoin repository has very high test coverage.											
File		% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines					
contracts/		100	100	100	100						
Administratable.sol		100	100	100	100						
Freezable.sol		100	100	100	100						
PlusCoin.sol		100	100	100	100						
All files		100	100	100	100						
Automated Analyses											
Oyente											
Symbolic execution (the Oyente tool) did not detect any vulnerabilities of types Parity Multisig Bug 2, Transaction-Ordering Dependence (TOD), Callstack Depth Attack, Timestamp Dependency, Re-Entrancy Vulnerability, and Integer Over/Underflow.											

46445 •

46550 .

44423 •

46402

46385

32605 ·

44258 •

32341 ·

15301 •

52961

45539 •

· % of limit ·

30837

The Mythril tool detects defects such as Integer underflow, Unprotected functions, Missing check on call return value, Re-entrancy, Multiple sends in a single transaction,

The following are the SHA-256 hashes of the audited contracts and/or test files. A smart contract or file with a different SHA-256 hash has been modified, intentionally or otherwise, after the

Tests

./test/Administratable.test.js

audit. 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

External call to untrusted contract, delegatecall or callcode to untrusted contract, Timestamp dependence, Use of tx.origin, Predictable RNG, Transaction order

dependence, Use of assert() instead of require(), Use of deprecated functions, Detect tautologies. The tool found a potential integer underflow issue in the add()

46381 •

46220 .

44093 •

method of SafeMath.sol, but Quantstamp determined that it was a false positive. Mythril did not find any other issues. MAIAN The MAIAIN tool detects three kinds of bad contracts, Greedy, Prodigal and Suicidal. It found no defects.

Appendix

audit.

Contracts

File Signatures

Mythril

Adherence to Specification With minimal written specification we were unable to judge to what degree the code conforms to the specification. Private conversation with the PlusCoin team convinced

us that the contract code implements the desired functionality within the context of its intended usage.

./contracts/Administratable.sol

105236f1f4649fdfddbe7aa4857cc8466c2406e6ea30b668836e0f8f88f2cd27

2cf21015089404778ecb5a4ad22a3bfb7dac64709ec8444ab2d6bd5c4d3e7ab7

./test/Freezable.test.js ./contracts/Freezable.sol 6f222918c590e3f2ced22b752649aebfe7819c412b02c0db2954077ef3cb14c3 700c0904cfbc20dba65f774f54a476803a624372cc95d926b3eba9b4d0f0312e ./contracts/Migrations.sol ./test/PlusCoin.test.js 7a60bbce9324841a5534206bf2c0fe998c211e71fedb161c16889c51058275fe ./contracts/PlusCoin.sol

Quantstamp is a Y Combinator-backed company that helps to secure smart contracts at scale using computer-aided reasoning tools, with a mission to help boost

Quantstamp's team boasts decades of combined experience in formal verification, static analysis, and software verification. Collectively, our individuals have over 500

Google scholar citations and numerous published papers. In its mission to proliferate development and adoption of blockchain applications, Quantstamp is also developing

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;

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp, Inc. (Quantstamp). Such hyperlinks are

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

associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Cryptographic tokens are emergent technologies and carry with them

high levels of technical risk and uncertainty. The Solidity language itself and other smart contract languages remain under development and are subject to unknown risks

aspects that could present security risks. You may risk loss of tokens, Ether, and/or other loss. A report is not an endorsement (or other opinion) of any particular project or

and flaws. The review does not extend to the compiler layer, or any other areas beyond Solidity or the smart contract programming language, or other programming

team, and the report does not guarantee the security of any particular project. A report does not consider, and should not be interpreted as considering or having any

Timeliness of content

Notice of confidentiality

Links to other websites

About Quantstamp

adoption of this exponentially growing technology.

no obligation to update any information following publication.

a new protocol for smart contract verification to help smart contract developers and projects worldwide to perform cost-effective smart contract security audits. To date, Quantstamp has helped to secure hundreds of millions of dollars of transaction value in smart contracts and has assisted dozens of blockchain projects globally with its white glove security auditing 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. Finally, Quantstamp's dedication to research and development in the form of collaborations with leading academic institutions such as National University of Singapore and MIT (Massachusetts Institute of Technology) reflects Quantstamp's commitment to enable world-class smart contract innovation.

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 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.

42b4e37d142c1232023e30efaa659f6785019704fce61022a29fcd782cccbded

5dc8653aa162669f1f8651ac25effd79c96b06814a10297cdf0cc6dcb32dba0e

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

completeness of any outcome generated by such software.

bearing on, the potential economics of a token, token sale or any other product, service or other asset. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. To the fullest extent permitted by law, we disclaim all warranties, express 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 website, or any website or mobile application featured in any banner or other 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. You may risk loss of QSP tokens or other loss. 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.

PlusCoin Token Audit