Quantstamp Security Assessment Certificate

February 13th 2021 — Quantstamp Verified

GYEN & ZUSD Contracts

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

Executive Summary

Type	ERC20 Token
Auditors	Kacper Bąk, Senior Research Engineer Ed Zulkoski, Senior Security Engineer Sebastian Banescu, Senior Research Engineer Sung-Shine Lee, Research Engineer
Timeline	2019-10-09 through 2021-02-09
EVM	Byzantium
Languages	Solidity, Javascript
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	README.md
Documentation Quality	Medium
Test Quality	High

Source Code

Repository	Commit
contracts	ce551e8
gmotrust-stablecoin-contract	21ac3ba
None	9f2de67
None	04f0738

Goals

- Does the token feature centralization of power?
- Does the token conform to ERC20 standard?

Total Issues	**6** (5 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**2** (2 Resolved)
Low Risk Issues	**2** (1 Resolved)
Informational Risk Issues	**2** (2 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

The code adheres to some best practices, however, it features centralization of power (minter and capper could perform unlimited token minting at any time).
Update: 1 the team updated some documentation and specification. The team provided detailed explanations regarding the mechanics by which tokens are minted, burned, and capped. The mechanism does rely on a centralized entity controlling the process, which cannot be enforced by the smart contract code. The maintenance of the 1:1 peg between fiat and tokens is maintained off-chain.
Update: 2 we have found a few more issues. We recommend addressing them before deploying the code.
Update: 3 the team addresses the issues by updating the documentation.

ID	Description	Severity	Status
QSP-1	Collusion between minter and capper may lead to arbitrary minting	Medium	Mitigated
QSP-2	Anybody can burn their own tokens	Medium	Fixed
QSP-3	No data validation in constructor / initializer	Low	Resolved
QSP-4	Privileged Roles and Ownership	Low	Acknowledged
QSP-5	Unlocked pragma and dependencies	Informational	Resolved
QSP-6	Allowance Double-Spend Exploit	Informational	Resolved

Quantstamp Audit Breakdown

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

Toolset

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

Setup

Tool Setup:

- Slither v0.6.13

Steps taken to run the tools:

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

Findings

QSP-1 Collusion between minter and capper may lead to arbitrary minting

Severity: Medium Risk

Status: Mitigated

File(s) affected: `Token_v1.sol`

Description: Although the function `mint()` is protected by the modifier `notMoreThanCapacity`, it allows the minter to perform arbitrary minting in case when the minter colludes with the capper. The latter can increase the cap via the function `cap()`.

Recommendation: We recommend updating the token so that it has a constant supply. This way the token value would not be diluted by minting. Alternatively, specification and documentation should explain why arbitrary minting is necessary.
Update: the team provided detailed explanations regarding the mechanics by which tokens are minted, burned, and capped in the documentation in the `README.md` file (commit `4961a30`). The mechanism does rely on a centralized entity controlling the process, which cannot be enforced by the smart contract code. The maintenance of the 1:1 peg between fiat and tokens is maintained off-chain. However this is a relatively common structure of stablecoins and relies on the auditing of the issuing entity.

QSP-2 Anybody can burn their own tokens

Severity: Medium Risk

Status: Fixed

File(s) affected: `Token_v1.sol`

Description: The `burn()` function in `Token_v1.sol` can be called by anyone. Consequently, anyone could burn their own tokens without sending them back to the centralized party and have the party perform the burning. This might not be the intended behavior as according to the README:

> `Customers can transfer tokens to their own addresses provided by GMO-Z.com Trust Company during the purchase applications. We burn tokens of theses addresses, then we send the same amount of fiat currency fund 1:1 with burned token amount to their bank accounts.`

Recommendation: We recommend checking whether this behavior is intended.
Update: the team clarified in the file `README.md` that this is the intended behavior.

QSP-3 No data validation in constructor / initializer

Severity: Low Risk

Status: Resolved

File(s) affected: `Token_v1.sol`

Description: In `Token_v1.sol`, the function `initialize()` sets the addresses of all roles, but does not check if these addresses differ from `0x0`.

Recommendation: We recommend adding `require()` statements to ensure that none of the addresses is equal to `0x0`.

QSP-4 Privileged Roles and Ownership

Severity: *Low Risk*

Status: Acknowledged

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract.

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.

QSP-5 Unlocked pragma and dependencies

Severity: *Informational*

Status: Resolved

File(s) affected: `Common.sol`, `Pauser.sol`, `Prohibiter.sol`, `Capper.sol`, `Minter.sol`, `MinterAdmin.sol`, `Admin.sol`, `Owner.sol`, `Token_v1.sol`, `ZUSD.sol`, `GYEN.sol`, `Burning.sol`, `BurningFactory.sol`

Related Issue(s): SWC-103

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.4.*`. 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". Similarly, in `package.json`, versions of dependencies are also 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. Similarly, we recommend locking dependency versions in `package.json`.

QSP-6 Allowance Double-Spend Exploit

Severity: *Informational*

Status: Resolved

File(s) affected: `Token_v1.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.

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.

Automated Analyses

Slither

Slither reported no issues.

Adherence to Specification

- The code is not accompanied by any specification. **Update:** as of commit `4961a30` the file `README.md` provides sufficient specification.
- The `Capper` contract extends `Pauser`, however, it does not use any of the modifiers of the `Pauser`. Since there is no specification available, is unclear whether the `_cap` function should be guarded by the `whenNotPaused` modifier. **Update:** fixed, `Capper` extends `Common` as of commit `2917944`.
- The function `BurningFactory.deploy()` is not guarded by any modifier, which means that anyone on the network can call this function and use it to deploy a new `Burning` contract instance and emit an event. It is unclear whether this functionality is intentional or it's a mistake. **Update:** fixed.
- The role of wiper is not documented in `README.md`. **Update:** fixed.
- In `Token_v2.sol`, what happens with the underlying funds when tokens get wiped? **Update:** fixed.

Code Documentation

The code does not have any inline documentation.

Adherence to Best Practices

The code adheres to some best practices, however:

• `Admin.sol`, line 25, and `Owner.sol`, line 24, contain the comment /* whenNotPaused */. Instead of the comment, we recommend adding a description to the function to explain why the modifier must not be present. **Update:** fixed.

• the following events are unindexed parameters: `event Pauser.Pause(bool status, address sender);` and `event Prohibiter.Prohibition(address prohibited, bool status, address sender);`. We recommend indexing at least the parameters of type `address`. **Update:** fixed.

• few typos in `require()` messages:

. `Prohibiter.sol`, line 17: "this account is a prohibited" should be "this account is prohibited". **Update:** fixed.

. `Prohibiter.sol`, line 22: "this account is not a prohibited" should be "this account is not prohibited". **Update:** fixed.

. `Common.sol`, line 10: "this amount is not an natural number" should be "this amount is not a natural number". **Update:** fixed.

• Instead of using custom roles, as a best practice we would recommend using OpenZeppelin roles contracts.

• In `Token_v1.sol`, lines 55-57 could be replaced by `super.transferFrom(...)`. **Update:** fixed.

• In `Token_v1.sol`, instead of writing the custom function `burn()`, it could be inherited from OpenZeppelin `ERC20Burnable`.

• In `Common.sol`, `isNaturalNumber()` checks if the number is positive. We recommend renaming it to `isPositiveNumber()`.

Test Results

Test Suite Results

```
Contract: BurningFactory.sol
  Test constructor function
     ✓ should create BurningFactory instance (72ms)
     ✓ manager address should not be zero (57ms)
     ✓ burner address should not be zero (58ms)
  Test deploy function
     ✓ Can deploy new burning (58ms)
     ✓ non burner cannot deploy (52ms)
  Test changeBurner function
     ✓ Manager can change burner (67ms)
     ✓ non manager cannot change burner (44ms)
     ✓ burner address should not be zero (53ms)

Contract: Burning.sol
  Test burn function
     ✓ Burner can burn (210ms)
     ✓ Non burner cannot burn (167ms)
  Test transfer function
     ✓ Burner can transfer (198ms)
     ✓ Non burner cannot transfer (160ms)
     ✓ Transfer should fail when burning was prohibited (210ms)
     ✓ Transfer should fail when contract was paused (217ms)
     ✓ Transfer should fail when recipient was zero address (204ms)
     ✓ Transfer should fail when amount was more than balance (195ms)

Contract: GYEN.sol
  Test implementation of AdminUpgradeabilityProxy
     ✓ Admin can view address of implementation
     ✓ Non admin cannot view address of implementation
  Test admin of AdminUpgradeabilityProxy
     ✓ Admin can view address of admin
     ✓ Non admin cannot view address of admin
  Test changeAdmin of AdminUpgradeabilityProxy
     ✓ Admin can change admin of proxy (43ms)
     ✓ Non admin cannot change admin of proxy (49ms)
     ✓ Cannot change admin of proxy to zero address
  Test upgradeTo of AdminUpgradeabilityProxy
     ✓ Admin can upgrade the implementation of proxy (70ms)
     ✓ Non admin cannot upgrade the implementation of proxy (87ms)
     ✓ Cannot upgrade implementation to non contract address (78ms)
  Test initialize function
     ✓ Initialize cannot call multiple times (53ms)
  Test initializeWiper function
     ✓ initializeWiper cannot call multiple times (41ms)
  Test cap function
     ✓ set capacity success case
     ✓ non capper cannot set capacity
     ✓ paused contract cannot set capacity (70ms)
     ✓ set capacity less than the totalSupply should fail (79ms)
     ✓ cannot set cap to a non natural number (42ms)
  Test mint function
     ✓ minter can mint (57ms)
     ✓ non minter cannot mint (41ms)
     ✓ paused contract cannot be mint (81ms)
     ✓ mint should not above capacity (45ms)
     ✓ mint should not above capacity (86ms)
     ✓ mint address should not be zero (46ms)
     ✓ mint should change the totalSupply (77ms)
     ✓ cannot mint a non natural number (44ms)
  Test transfer function
     ✓ transfer success case (95ms)
     ✓ prohibited account cannot transfer (122ms)
     ✓ paused contract cannot do transfer (111ms)
     ✓ recipient address should not be zero (76ms)
     ✓ transfer with amount over balance should fail (82ms)
     ✓ transfer with amount over balance should fail (125ms)
     ✓ cannot transfer with amount is not a natural number (77ms)
  Test transferFrom function
     ✓ transferFrom success case (134ms)
     ✓ prohibited sender cannot transfer (155ms)
     ✓ paused contract cannot do transfer (153ms)
     ✓ transfer amount that hasn't been approved should fail (101ms)
     ✓ transfer amount exceed approved amount should fail (116ms)
     ✓ transfer amount exceed approved amount should fail (158ms)
     ✓ recipient address should not be zero (122ms)
     ✓ cannot transferFrom with amount is not a natural number (122ms)
  Test burn function
     ✓ burn success case (179ms)
     ✓ burn should change the totalSupply (115ms)
     ✓ burn exceed the balance of account should fail (89ms)
     ✓ burn exceed the balance of account should fail (179ms)
     ✓ cannot burn amount of non natural number (77ms)
  Test approve function
     ✓ initial allowance should be zero
     ✓ approve should change the allowance (109ms)
  Test wipe function
     ✓ wiper can wipe (131ms)
     ✓ non wiper cannot wipe (110ms)
     ✓ no prohibited address cannot be wipe (82ms)
     ✓ paused contract cannot be wipe (221ms)
     ✓ wipe should change the totalSupply (155ms)

Contract: Admin.sol
  Test changeCapper function
     ✓ admin can change the capper
     ✓ non admin cannot change the capper
     ✓ paused contract cannot change the capper (64ms)
     ✓ cannot change the capper to zero address
  Test changePauser function
     ✓ admin can change the pauser
     ✓ non admin cannot change the pauser (38ms)
     ✓ cannot change the pauser to zero address (39ms)
  Test changeProhibiter function
     ✓ admin can change the prohibiter
     ✓ non admin cannot change the prohibiter
     ✓ paused contract cannot change the prohibiter (64ms)
```

```
      ✓ cannot change the prohibiter to zero address (38ms)
  Test changeWiper function
      ✓ admin can change the wiper
      ✓ non admin cannot change the wiper (40ms)
      ✓ paused contract cannot change the wiper (67ms)
      ✓ cannot change the wiper to zero address

Contract: MinterAdmin.sol
  Test changeMinter function
      ✓ MinterAdmin can change the minter
      ✓ non minterAdmin cannot change the minter
      ✓ paused contract cannot change the minter (67ms)
      ✓ cannot change the minter to zero address

Contract: Owner.sol
  Test changeOwner function
      ✓ owner can change the owner
      ✓ non owner cannot change the owner
      ✓ paused contract cannot change the owner (68ms)
      ✓ cannot change the owner to zero address
  Test changeAdmin function
      ✓ owner can change the admin (40ms)
      ✓ non owner cannot change the admin
      ✓ cannot change the admin to zero address (38ms)
  Test changeMinterAdmin function
      ✓ owner can change the minterAdmin
      ✓ non owner cannot change the minterAdmin
      ✓ paused contract cannot change the minterAdmin (63ms)
      ✓ cannot change the minterAdmin to zero address

Contract: Pauser.sol
  Test pause function
      ✓ pauser can pause the contract
      ✓ non pauser cannot pause the contract
      ✓ paused contract cannot pause again (59ms)
  Test unpause function
      ✓ pauser can unpause the contract (57ms)
      ✓ non pauser cannot unpause the contract (57ms)
      ✓ unpause contract cannot unpause again (101ms)

Contract: Prohibiter.sol
  Test prohibit function
      ✓ prohibiter can prohibit the account (55ms)
      ✓ non prohibiter cannot prohibit the account (42ms)
      ✓ paused contract cannot prohibit account (63ms)
      ✓ prohibited account cannot prohibit again (65ms)
      ✓ prohibited account cannot be zero (42ms)
  Test unprohibit function
      ✓ prohibiter can unprohibit the account (58ms)
      ✓ non prohibiter cannot unprohibit the account (64ms)
      ✓ paused contract cannot unprohibit account (96ms)
      ✓ non prohibited account cannot unprohibit
      ✓ unprohibit account cannot be zero

Contract: Token_v1.sol
  Test initialize function
      ✓ Initialize cannot call multiple times (112ms)
      ✓ cannot initialize owner to zero address (62ms)
      ✓ cannot initialize admin to zero address (78ms)
      ✓ cannot initialize capper to zero address (54ms)
      ✓ cannot initialize prohibiter to zero address (62ms)
      ✓ cannot initialize pauser to zero address (45ms)
      ✓ cannot initialize minterAdmin to zero address (47ms)
      ✓ cannot initialize minter to zero address (49ms)
  Test cap function
      ✓ set capacity success case (39ms)
      ✓ non capper cannot set capacity (45ms)
      ✓ paused contract cannot set capacity (62ms)
      ✓ set capacity less than the totalSupply should fail (68ms)
      ✓ cannot set cap to a non natural number
  Test mint function
      ✓ minter can mint (50ms)
      ✓ non minter cannot mint (38ms)
      ✓ paused contract cannot be mint (66ms)
      ✓ mint should not above capacity
      ✓ mint should not above capacity (74ms)
      ✓ mint address should not be zero
      ✓ mint should change the totalSupply (67ms)
      ✓ cannot mint a non natural number
  Test transfer function
      ✓ transfer success case (80ms)
      ✓ prohibited account cannot transfer (98ms)
      ✓ paused contract cannot do transfer (97ms)
      ✓ recipient address should not be zero (73ms)
      ✓ transfer with amount over balance should fail (67ms)
      ✓ transfer with amount over balance should fail (106ms)
      ✓ cannot transfer with amount is not a natural number (70ms)
  Test transferFrom function
      ✓ transferFrom success case (140ms)
      ✓ prohibited sender cannot transfer (141ms)
      ✓ paused contract cannot do transfer (138ms)
      ✓ transfer amount that hasn't been approved should fail (93ms)
      ✓ transfer amount exceed approved amount should fail (117ms)
      ✓ transfer amount exceed approved amount should fail (143ms)
      ✓ recipient address should not be zero (100ms)
      ✓ cannot transferFrom with amount is not a natural number (107ms)
  Test burn function
      ✓ burn success case (84ms)
      ✓ burn should change the totalSupply (94ms)
      ✓ burn exceed the balance of account should fail (67ms)
      ✓ burn exceed the balance of account should fail (100ms)
      ✓ cannot burn amount of non natural number (108ms)
  Test approve function
      ✓ initial allowance should be zero
      ✓ approve should change the allowance (279ms)

Contract: Upgrade GYEN/ZUSD from v1 to v2
  uses original storage slot positions
      ✓ retains original storage slots 0 through 65 (206ms)
      ✓ retains original storage slots for balances mapping
      ✓ retains original storage slots for allowed mapping
      ✓ retains original storage slots for prohibiteds mapping
  uses original storage slot positions
      ✓ retains original storage slots 0 through 65 (210ms)
      ✓ retains original storage slots for balances mapping
      ✓ retains original storage slots for allowed mapping
      ✓ retains original storage slots for prohibiteds mapping
  uses original storage slot positions
      ✓ retains original storage slots 0 through 65 (219ms)
      ✓ retains original storage slots for balances mapping
      ✓ retains original storage slots for allowed mapping
      ✓ retains original storage slots for prohibiteds mapping
  uses original storage slot positions
      ✓ retains original storage slots 0 through 65 (234ms)
      ✓ retains original storage slots for balances mapping
      ✓ retains original storage slots for allowed mapping
      ✓ retains original storage slots for prohibiteds mapping

Contract: ZUSD.sol
  Test implementation of AdminUpgradeabilityProxy
      ✓ Admin can view address of implementation
      ✓ Non admin cannot view address of implementation
  Test admin of AdminUpgradeabilityProxy
      ✓ Admin can view address of admin
      ✓ Non admin cannot view address of admin
  Test changeAdmin of AdminUpgradeabilityProxy
      ✓ Admin can change admin of proxy (50ms)
      ✓ Non admin cannot change admin of proxy (39ms)
      ✓ Cannot change admin of proxy to zero address (63ms)
  Test upgradeTo of AdminUpgradeabilityProxy
      ✓ Admin can upgrade the implementation of proxy (70ms)
      ✓ Non admin cannot upgrade the implementation of proxy (79ms)
      ✓ Cannot upgrade implementation to non contract address (75ms)
  Test initialize function
      ✓ Initialize cannot call multiple times (53ms)
  Test initializeWiper function
      ✓ initializeWiper cannot call multiple times (54ms)
  Test cap function
      ✓ set capacity success case
      ✓ non capper cannot set capacity (60ms)
      ✓ paused contract cannot set capacity (110ms)
      ✓ set capacity less than the totalSupply should fail (100ms)
      ✓ cannot set cap to a non natural number (39ms)
  Test mint function
      ✓ minter can mint (66ms)
      ✓ non minter cannot mint (41ms)
      ✓ paused contract cannot be mint (82ms)
      ✓ mint should not above capacity (41ms)
      ✓ mint should not above capacity (79ms)
      ✓ mint address should not be zero (40ms)
      ✓ mint should change the totalSupply (75ms)
      ✓ cannot mint a non natural number (48ms)
  Test transfer function
      ✓ transfer success case (88ms)
      ✓ prohibited account cannot transfer (115ms)
      ✓ paused contract cannot do transfer (107ms)
```

```
      ✓ recipient address should not be zero (105ms)
      ✓ transfer with amount over balance should fail (107ms)
      ✓ transfer with amount over balance should fail (140ms)
      ✓ cannot transfer with amount is not a natural number (75ms)
   Test transferFrom function
^[[A       ✓ transferFrom success case (144ms)
      ✓ prohibited sender cannot transfer (142ms)
      ✓ paused contract cannot do transfer (138ms)
      ✓ transfer amount that hasn't been approved should fail (88ms)
      ✓ transfer amount exceed approved amount should fail (116ms)
      ✓ transfer amount exceed approved amount should fail (184ms)
      ✓ recipient address should not be zero (110ms)
      ✓ cannot transferFrom with amount is not a natural number (113ms)
   Test burn function
      ✓ burn success case (90ms)
      ✓ burn should change the totalSupply (104ms)
      ✓ burn exceed the balance of account should fail (74ms)
      ✓ burn exceed the balance of account should fail (124ms)
      ✓ cannot burn amount of non natural number (80ms)
   Test approve function
      ✓ initial allowance should be zero
      ✓ approve should change the allowance (130ms)
   Test wipe function
      ✓ wiper can wipe (132ms)
      ✓ non wiper cannot wipe (111ms)
      ✓ no prohibited address cannot be wipe (77ms)
      ✓ paused contract cannot be wipe (145ms)
      ✓ wipe should change the totalSupply (148ms)


  225 passing (1m)
```

Code Coverage

The code features superb coverage.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
Burning.sol	100	100	100	100	
BurningFactory.sol	100	100	100	100	
GYEN.sol	100	100	100	100	
Token_v1.sol	100	100	100	100	
Token_v2.sol	100	100	100	100	
ZUSD.sol	100	100	100	100	
contracts/Roles/	100	100	100	100	
Admin.sol	100	100	100	100	
Capper.sol	100	100	100	100	
Common.sol	100	100	100	100	
Minter.sol	100	100	100	100	
MinterAdmin.sol	100	100	100	100	
Owner.sol	100	100	100	100	
Pauser.sol	100	100	100	100	
Prohibiter.sol	100	100	100	100	
Wiper.sol	100	100	100	100	
All files	**100**	**100**	**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

```
af6a532122a395bed59bfbfd75c7e47041e175c61a68b3df43a7b7a22a493e51 ./contracts/Burning.sol
4098d81546310eecd0656185e0d567dce69f323c9ec2ec27d6806c2b082b9b3d ./contracts/BurningFactory.sol
01e1f56dd89123e27a76aa5c66f61f3a967df69ac550c2af7bd90a1bfd0edfe9 ./contracts/GYEN.sol
92d547dc562cd58ea8dde04fd746d4ed54b4823a25a3f2b9b448fafd7ce745bd ./contracts/Migrations.sol
65de7dba0b7fa0af60b3ad010a1d528f846a94e67b4155b540d0ce04775a395a ./contracts/Token_v1.sol
5e269db93b7223b77630d49748945de55167301bd50c76dab667e204edf218c1 ./contracts/Token_v2.sol
98771afae3ef4ca08dd545374d2112df90b7f0befd10e79ee06b257d3a76e554 ./contracts/ZUSD.sol
06ee7bb58a96aaf5a34238b537a8708aaba92b55b51dc62aafdee6d019ff9710 ./contracts/Roles/Admin.sol
4393a3cf3d953c73dc436c970b9f1b1be2a500c4d670d9c6b6eedaa639c616a9 ./contracts/Roles/Capper.sol
6abeb5929f8af467c77f4d3bb366d2acb364f5debd115bd04f03629311527616 ./contracts/Roles/Common.sol
46310c3909cc3c203b1abe2da3d8d432cc861265555d68ced358c77bb50c48b9 ./contracts/Roles/Minter.sol
08d917fb583dfdb65edc157eef84af2b0b5ab679156f3fe3e28f4ccd1e1db7d8 ./contracts/Roles/MinterAdmin.sol
459fdc4094037a24f5f3d47c84b0d4ebc77f579cf18716c024a83a026a42ce02 ./contracts/Roles/Owner.sol
e679133d1dc43a1404fb20e2381aae67e967414068ae1c93a66ae250977e8ba6 ./contracts/Roles/Pauser.sol
43149121ea8befd879568723391d018a18b45537fc94ecd2f43233f52bba1d5e ./contracts/Roles/Prohibiter.sol
888c26f02e4eab5834e4998923a5d98d1a8c1591e99bba7eff0ec907247c657f ./contracts/Roles/Wiper.sol
```

Tests

```
4ed4fc953ed620f11e954a64248ebd2049a580e4bfa462548e1676cad5499d3e ./test/burning.js
731ca3a78c4a5065e030b8abe6abdfe72deb7230f1a8603cb16d66629687b54a ./test/burning_factory.js
c092d513150c6b2efcdb84bba64d0956e385cb75c6260f14365a759641442052 ./test/gyen.js
2d1cbd825d7fab3481fefb1f38435f142caa619b3215de6d83cf0c0a02526895 ./test/token_v1.js
52614aa88c47e9c47760740cbc328e8d4e669ecca27ee82f613d2c709f641a67 ./test/upgrade_gyen_zusd_to_v2.js
2de5c675facaded8110ef88f753b5e3c390a7d36a12ef824047ea3771b2287bc ./test/validate_storage_slots_helper.js
5487784f683d74e2f7f6e56165e3e71e2204a1f186b4671ac118512aff28c3f4 ./test/zusd.js
0a277e7226ba20364484c3c537f30e2563d8befc0126f27cb0b01bcfb1795d1d ./test/roles/admin.js
9fdd385017a266fcb0b164481175cab232b02b20f96296c206e14190e06dce53 ./test/roles/minterAdmin.js
2b287414f2f23950c7084a56778f3b939057f06e3347ccab6655ec1cd5bc7017 ./test/roles/owner.js
7932d96acafed90bd6f6e00e5527876b89135234e10c8f61df440637713fb1a3 ./test/roles/pauser.js
daf3664198e66eaa32b5c6fc75e13b476d0a2fc373259e74b780c727a132f0ae ./test/roles/prohibiter.js
```

Changelog

- 2019-10-11 - Initial report
- 2019-11-01 - Revised report based on commit `2917944`
- 2019-11-27 - Revised report based on commit `4961a30`
- 2019-11-29 - Revised report based on commit `7319662`
- 2021-01-20 - Revised report based on commit `21ac3ba`
- 2021-02-03 - Revised report based on commit `9f2de67`
- 2021-02-09 - Revised report based on commit `04f0738`

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.