Quantstamp

GYEN/ZUSD

Executive Summary

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

Type	Stablecoin
Timeline	2024-07-08 through 2024-07-11
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	README.md
Source Code	• https://github.com/trust-zcom/contracts-arbitrum #35e9abf • https://github.com/trust-zcom/contracts-optimism #9ec47c3
Auditors	• Danny Aksenov Senior Auditing Engineer • Adrian Koegl Auditing Engineer • Gereon Mendler Auditing Engineer

Documentation quality	Low
Test quality	Medium
Total Findings	5 Acknowledged: 5
High severity findings	0
Medium severity findings	1 Acknowledged: 1
Low severity findings	3 Acknowledged: 3
Undetermined severity findings	0
Informational findings	1 Acknowledged: 1

Summary of Findings

The audit report for the GYEN/ZUSD stablecoin contracts on Arbitrum and Optimism networks reveals several issues, mostly related to outdated dependencies and upgradeability mechanisms. While no high-severity vulnerabilities were found, there is one medium-severity and three low-severity findings that need attention. The contracts show good test coverage and passing tests, indicating a solid foundation. In terms of best practices, we recommend the client merge both repositories and update the solidity version and dependencies.

Update: The GMO team has acknowledged all audit findings. While some design choices are maintained for consistency with existing L1 contracts, the main recommendation to upgrade Solidity versions, dependencies, and implement more secure upgradeability mechanisms remains relevant for enhancing the project's overall security posture.

ID	DESCRIPTION	SEVERITY	STATUS
GMO-1	Outdated Solidity Versions and Dependencies	Medium	Acknowledged
GMO-2	Deprecated Upgradeability Mechanism	Low	Acknowledged
GMO-3	Token Name and Symbol Can Be Changed	Low	Acknowledged
GMO-4	Unprotected Initialization Function Can Be Front-Run	Low	Acknowledged
GMO-5	Potential Ether Lock in Proxy Contracts	Informational	Acknowledged

Assessment Breakdown

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

> **Disclaimer**
>
> Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

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

1. Code review that includes the following
    1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
    2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    3. 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:
    1. 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.
    2. 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, clarity, 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.

Scope

Files Included

Repo: https://github.com/trust-zcom/contracts-arbitrum(35e9abf98c81e9bbeb3e2d5142bc8ecd65f6b9f1) Files: contracts/*
Repo: https://github.com/trust-zcom/contracts-optimism(9ec47c396809d009f85c8a7dc138f4b6089821b5) Files: contracts/*

Files Excluded

Repo: https://github.com/trust-zcom/contracts-arbitrum(35e9abf98c81e9bbeb3e2d5142bc8ecd65f6b9f1) Files: contracts/Migrations.sol
Repo: https://github.com/trust-zcom/contracts-optimism(9ec47c396809d009f85c8a7dc138f4b6089821b5) Files: contracts/Migrations.sol

Operational Considerations

1. The contracts rely on the correct operation of the bridge relayers, through which tokens can be minted and burned. We assume the bridge relayers to be honest and not compromised.
2. The contracts are upgradeable. We assume that every upgrade will undergo a thorough audit.
3. Tokens of prohibited users can be burned by the `wiper` role. We assume that the `prohibiter` and `wiper` role will not abuse this power.
4. We assume that the keys of the `admin`, `pauser`, `prohibiter`, `rescuer`, and `wiper` are securely managed.
5. We assume the keys of the `owner` to be in a cold storage and not be used in a hot wallet.

Key Actors And Their Capabilities

1. The `pauser` can pause and unpause the token contract.
2. The `prohibiter` can blacklist users and prevent them from sending or receiving funds.
3. The `wiper` role can burn funds of prohibited user.

4. The `rescuer` role can transfer any ERC20 token out of the token contract.
5. The `admin` role can change all of the above roles. Therefore, the `admin` role can effectively burn funds of users.
6. The `owner` role can change the `admin` and `owner` role. As a result, the `owner` role can effectively burn user funds as well.

Findings

GMO-1 Outdated Solidity Versions and Dependencies

Medium | Acknowledged

> **Update**
> Marked as "Acknowledged" by the client.
> The client provided the following explanation:
>
> ```
> Our L2 contracts are prepared as a subset of L1 contracts, and their functionality and version are identical to the GYEN/ZUSD contracts currently operating on L1. The version issue that has been pointed out will be considered as a topic for review, including L1.
> ```

File(s) affected: `All files in scope`

Description: The contracts are currently using Solidity versions `0.5.13` and `0.5.16`, which lack the benefits of newer compiler optimizations, bug fixes, and security improvements. Additionally, older dependencies, particularly OpenZeppelin libraries, are less efficient and secure compared to their latest counterparts. Upgrading to newer versions will enhance gas efficiency, eliminate known bugs, and bolster security measures.

Recommendation: Upgrade to Solidity version `0.8.26` and use the most recent OpenZeppelin contracts. Update the contract logic as needed to align with the new versions and take advantage of the latest improvements.

GMO-2 Deprecated Upgradeability Mechanism

Low | Acknowledged

> **Update**
> Marked as "Acknowledged" by the client.
> The client provided the following explanation:
>
> ```
> Our L2 contracts are prepared as a subset of L1 contracts, and their functionality and version are identical to the GYEN/ZUSD contracts currently operating on L1. The version issue that has been pointed out will be considered as a topic for review, including L1.
> ```

File(s) affected: `contracts/GYEN.sol`, `contracts/ZUSD.sol`

Description: The upgradeability mechanism utilized in the `GYEN` and `ZUSD` contracts is deprecated by OpenZeppelin. Specifically, the use of `AdminUpgradeabilityProxy` is no longer recommended due to the risk of selector clashing, which can compromise contract functionality and security. Selector clashing occurs when different functions share the same function selector, potentially leading to unintended function calls and vulnerabilities.

Recommendation: There are two best practices currently recommended for upgradeable contracts:

1. Use the `TransparentUpgradeableProxy` contract, which mitigates the risk of selector clashing.
2. Implement the `Universal Upgradeable Proxy Standard (UUPS)`, which moves the upgrade logic into the implementation contract, reducing the reliance on a central proxy admin contract.

Adopting either of these practices will enhance the security and reliability of your upgradeable contracts.

GMO-3 Token Name and Symbol Can Be Changed

Low | Acknowledged

> **Update**
> Marked as "Acknowledged" by the client.
> The client provided the following explanation:
>
> ```
> We have made it possible to change the token in case we need to rebrand it in the future. Furthermore, the private key that enables this change is strictly managed under our security policy.
> ```

File(s) affected: `contracts/OpToken_v1.sol`, `contracts/ArbToken_v1.sol`, `contracts/ArbToken_v2.sol`

Description: The ability to change the name and symbol of a token poses significant risks to the integrity and reliability of other contracts, such as marketplaces, that depend on these attributes for identification and functionality. Altering the name and symbol can lead to severe disruptions, causing confusion, loss of trust, and potential financial losses for users who interact with these contracts.

Recommendation: Ensure that the name and symbol of the token are immutable once set during the initial deployment. Remove the functionality to change name and symbol.

GMO-4 Unprotected Initialization Function Can Be Front-Run

Low Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
ArbToken_v2 is a logic-only contract without actual state. ArbTokenV2 can only be executed once when renaming, and for this ArbTokenV2 to be deployed and potentially negatively impact our Token, the deployer key is required, which is strictly managed.
```

File(s) affected: `contracts/ArbToken_v2.sol` ,,

Description: The `initializeV2` function in `ArbToken_v2.sol` is missing a modifier to prevent it from being front-run by a malicious actor. Note: This is mitigated by the `updateMetadata()` function. Additionally, we would like to point out that this issue is related to the token name and symbol being dynamic as referenced in GMO-3. If the token name and symbol were set at deployment and immutable, then the `initializeV2()` would not be capable of changing any sensitive information.

Recommendation: Add a modifier such as `onlyRescuer` to ensure this function can only be called by a truster user. Alternatively , consider removing the `initializeV2()` function altogether and rely on the `updateMetadata()` function to update the domain separator fields.
Note: We would like to reiterate that having functionality to change these fields is not recommended in general. We urge you to consider incrementing the version number instead as referenced in GMO-S-4 .

GMO-5 Potential Ether Lock in Proxy Contracts

Informational Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
We will take into consideration in operation so that unnecessary Ether is not locked.
```

File(s) affected: `contracts/GYEN.sol` , `contracts/ZUSD.sol`

Description: The constructor of both GYEN and ZUSD contracts, which inherit from `AdminUpgradeabilityProxy` , are payable. This could lead to Ether being permanently locked in the contract if sent during deployment.

Recommendation: Consider introducing a mechanism to withdraw any potentially locked funds.

Auditor Suggestions

GMO-S6 Standard Functionalities Are Self-Implemented

Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
We are introducing functionality similar to the contracts already deployed on L1. To meet our security requirements, we are designing and implementing our own contracts independently.
```

Description: In the implementation of the contracts, some standard libraries are self-implemented. While we did not identify any security concerns with these implementations, using the libraries reduces risk and promotes user trust and readability. The following libraries could be used:

1. The domain separator calculation is defined in EIP712. OpenZeppelin provides the EIP712 library with functionality implemented in this contract.

2. The different roles like `pauser`, `admin`, `rescuer`, etc. are currently implemented in a separate contract each. OpenZeppelin's `AccessControl` contract provides the functionality to grant, revoke, and transfer roles. The `grantRole()` function can be overriden such that each role can only be granted to one address, if desired.

Recommendation: Consider using standard libraries in your implementation. Nevertheless, this is not a security concern and just serves as best practice advice.

GMO-S7 Optimism Docs Are Not Adjusted

Fixed

> **Update**
> Marked as "Fixed" by the client.
> Addressed in: `7b76d2f3c02c688d5a60c9809bc2258ec3048134`, `1853da35811a2e800d30b78606c6abb26ecc05cb`.
> The client provided the following explanation:
>
> ```
> As you pointed out, we have updated the Optimism README. We are also reviewing Arbitrum's README.
> ```

Description: The `README` documentation for the Optimism contracts seem to be a copy + paste of the Arbitrum `README` documentation. This causes `Arbitrum` to be mentioned throughout the documentation and incorrect function names such as `bridgeBurn()` which are called `burn()` in the Optimism contract.

Recommendation: In case the repos for Arbitum and Optimism will remain separate, adjust the Optimism `README` to reflect the function naming in the contract. However, we recommend merging both repositories.

GMO-S8 `permit()` Function Does Not Follow Current Standard

Acknowledged

> **Update**
> Marked as "Acknowledged" by the client.
> The client provided the following explanation:
>
> ```
> We will reconsider this when upgrading the Solidity version. We are currently using versions 0.5.13 and 0.5.16.
> ```

File(s) affected: `contracts/ArbToken_v1.sol`, `contracts/ArbToken_v2.sol`, `contracts/OpToken_v1.sol`

Description: The `permit()` function is self-implemented and doesn't follow the current standard. While we didn't identify any security issues, for cryptographic operations, it is safer to rely on standard implementations.

Recommendation: Extend the `ERC20Permit` contract by OpenZeppelin which implements a `permit()` function that also respects the chain ID through EIP712.

GMO-S9 Inefficient Handling of Contract Upgrade in `Arbtoken_v2.initializev2()`

Acknowledged

> **Update**
> Marked as "Acknowledged" by the client.
> The client provided the following explanation:
>
> ```
> Regarding contract's version control, we are using the same version control that has already been implemented in L1.
> ```

File(s) affected: `contracts/ArbToken_v1.sol`

Description: The `initializeV2()` function, which upgrades the `ArbToken` contract to v2, does not properly utilize the version field in the EIP-712 domain separator. Instead, it introduces a new `_NEW_DOMAIN_SEPARATOR` variable and associated functions. This approach is inefficient and goes against the intended use of versioning in EIP-712. In EIP-712, the version field in the domain separator is designed to handle contract upgrades elegantly. When a contract is upgraded, incrementing this version automatically invalidates all previous signatures without the need for a entirely new domain separator.

The current implementation:
1. Unnecessarily complicates the code by introducing a new domain separator.
2. Misses the opportunity to use the built-in versioning mechanism of EIP-712.
3. Requires additional overriding functions to handle the new domain separator.

Recommendation: Consider the following:

1. Remove the `_NEW_DOMAIN_SEPARATOR` variable and all associated functions.
2. Utilize the version field in the EIP-712 domain separator for upgrade management:
    - In the `initializeV2()` function, increment the version field.
    - Use this incremented version to construct the updated domain separator.
3. Implement a version check in the `initializeV2()` function to ensure it's only called once:

```
require(version == "1", "Contract already upgraded");
version = "2";
```

GMO-S10 Combine Repositories and Share Inheritance

Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
We will keep the repositories separate to ensure expandability to other networks and maintain
flexibility.
```

File(s) affected: `contracts/ArbToken_v1.sol`, `contracts/ArbToken_v2.sol`, `contracts/OpToken_v1.sol`

Description: A shared repository could cut down on duplicate code using inheritance, and consequently avoid mistakes during upgrades and fixes.

Recommendation: Considering, how similar the token contracts are for both Optimism and Arbitrum, we suggest using a unified token contract.

GMO-S11 Improve Input Validation

Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Thank you for your advice. Consider not to input incorrect data from an operational perspective during
deployment.
```

File(s) affected: `contracts/ArbToken_v1.sol`, `contracts/ArbToken_v2.sol`

Related Issue(s): SWC-contracts/OpToken_v1.sol

Description: The `initialize` function in `ArbToken_v1.sol`, `OpToken_v1.sol` and the `setName` and `setSymbol` functions in `ArbToken_v2.sol`, `OpToken_v1.sol` are missing input validation for the `name`, `symbol`, and `decimals` parameters.

Recommendation: Add appropriate checks to ensure the validity of these parameters (e.g., non-empty strings for name and symbol, reasonable range for decimals).

GMO-S12 Add Event Emissions for Initialization

Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
ArbToken_v1.sol and OpToken_v1.sol have already been deployed on the main network, and initialize can
only be executed once and will not be re-executed.
```

File(s) affected: `contracts/ArbToken_v1.sol`, `contracts/OpToken_v1.sol`

Description: The `initialize` functions in `ArbToken_v1.sol`, `OpToken_v1.sol` are missing event emissions for the initialization of various addresses.

Recommendation: Emit events for all address initializations to improve transparency and make it easier to track important contract changes off-chain.

GMO-S13 Use Override Keyword

Acknowledged

> **Update**
> Marked as "Acknowledged" by the client.
> The client provided the following explanation:
>
> ```
> We will reconsider this when upgrading the Solidity version. We are currently using versions 0.5.13 and 0.5.16.
> ```

File(s) affected: `contracts/ArbToken_v2.sol`

Description: The `DOMAIN_SEPARATOR` and `permit` functions in `ArbToken_v2.sol` are overriding functions from the parent contract but don't use the `override` keyword.

Recommendation: Although not required until Solidity `0.6.0`, using the `override` keyword makes it more apparent that these functions are overriding functions in the parent contract.

GMO-S14 Remove Duplicate Function

Fixed

> **Update**
> Marked as "Fixed" by the client.
> Addressed in: `dd8ccfd0f05b754819b86baaa7c64b3979bc1e22`.
> The client provided the following explanation:
>
> ```
> We removed the duplicate function you pointed out.
> ```

File(s) affected: `contracts/ArbToken_v2.sol`

Description: The `_calculateDomainSeparator` function is duplicated in both `ArbToken_v1.sol` and `ArbToken_v2.sol`.

Recommendation: Remove the duplicate function from `ArbToken_v2.sol` and use the one from `ArbToken_v1.sol` to avoid code duplication and potential inconsistencies.

GMO-S15 Improve Naming Convention

Acknowledged

> **Update**
> Marked as "Acknowledged" by the client.
> The client provided the following explanation:
>
> ```
> The functions of L1 are used as is.
> ```

File(s) affected: `contracts/Roles/Prohibiter.sol`, `contracts/Roles/Common.sol`

Description: There are two instances where naming conventions could be improved:

1. The `prohibiteds` mapping in `Prohibiter.sol` could have a more descriptive name.
2. The `isNaturalNumber` modifier in `Common.sol` could be renamed to better reflect its functionality.

Recommendation: 1. Rename `prohibiteds` to `prohibitedAddresses` in `Prohibiter.sol` for better clarity and adherence to naming conventions.

2. Rename `isNaturalNumber` to `isNonZero` in `Common.sol`, as it only checks if the number is greater than zero, not if it's a natural number in the strict mathematical sense.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.

- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – 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.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **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).

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.

Files

- `2dd...8c3 ./ArbToken_v1.sol`
- `755...d87 ./ArbToken_v2.sol`
- `600...333 ./GYEN.sol`
- `14a...50c ./ZUSD.sol`
- `f97...d5d ./contracts/GYEN.sol`
- `576...7ae ./contracts/ZUSD.sol`
- `a4b...de6 ./contracts/OpToken_v1.sol`
- `8cb...319 ./contracts/Roles/Prohibiter.sol`
- `556...94f ./contracts/Roles/Owner.sol`
- `980...bd6 ./contracts/Roles/Rescuer.sol`
- `588...014 ./contracts/Roles/Pauser.sol`
- `513...10c ./contracts/Roles/Wiper.sol`
- `a9d...e43 ./contracts/Roles/Common.sol`
- `055...aa7 ./contracts/Roles/Admin.sol`

Tests

- `f0b...015 ./OpToken_v1.js`
- `9a1...348 ./gyen.js`
- `dd9...f0e ./zusd.js`
- `490...fcb ./signERC2612Permit.js`
- `ccd...932 ./prohibiter.js`
- `cab...573 ./owner.js`
- `049...12c ./wiper.js`
- `923...8f5 ./pauser.js`
- `465...e53 ./admin.js`
- `d2d...186 ./rescuer.js`
- `4c8...75b ./test/ArbiToken_v1.js`
- `664...51d ./test/gyen.js`
- `788...069 ./test/zusd.js`
- `490...fcb ./test/utils/signERC2612Permit.js`
- `24e...db5 ./test/roles/prohibiter.js`
- `c48...a7d ./test/roles/owner.js`
- `120...c9e ./test/roles/wiper.js`
- `d12...5fb ./test/roles/pauser.js`
- `eee...cfc ./test/roles/admin.js`

- `376...bc3 ./test/roles/rescuer.js`

Automated Analysis

N/A

Test Suite Results

While there are issues with outdated dependencies, all tests are passing.

```
Contract: ArbToken_v1.sol
    Test initialize function
      ✔ Initialize cannot call multiple times (778ms)
      ✔ cannot initialize owner to zero address
      ✔ cannot initialize admin to zero address
      ✔ cannot initialize prohibiter to zero address
      ✔ cannot initialize pauser to zero address
      ✔ cannot initialize wiper to zero address
      ✔ cannot initialize rescuer to zero address
      ✔ cannot initialize l1Address to zero address
      ✔ cannot initialize l2Gateway to zero address
    Test bridgeMint function
      ✔ only l2Gateway can bridgeMint (46ms)
      ✔ non l2Gateway cannot bridgeMint
      ✔ bridgeMint address should not be zero
      ✔ bridgeMint should change the totalSupply (72ms)
    Test transfer function
      ✔ transfer success case (130ms)
      ✔ prohibited account cannot transfer (152ms)
      ✔ paused contract cannot do transfer (173ms)
      ✔ recipient address should not be zero (50ms)
      ✔ transfer with amount over balance should fail (48ms)
      ✔ transfer with amount over balance should fail (127ms)
      ✔ cannot transfer with amount is not a natural number (57ms)
      ✔ cannot transfer to prohibited recipient (109ms)
    Test transferFrom function
      ✔ transferFrom success case (154ms)
      ✔ prohibited sender cannot transfer (124ms)
      ✔ paused contract cannot do transfer (128ms)
      ✔ transfer amount that hasn't been approved should fail (53ms)
      ✔ transfer amount exceed approved amount should fail (83ms)
      ✔ transfer amount exceed approved amount should fail (151ms)
      ✔ recipient address should not be zero (93ms)
      ✔ cannot transferFrom with amount is not a natural number (120ms)
      ✔ cannot transferFrom to prohibited recipient address (151ms)
    Test bridgeBurn function
      ✔ bridgeBurn success case (111ms)
      ✔ bridgeBurn should change the totalSupply (123ms)
      ✔ bridgeBurn exceed the balance of account should fail (60ms)
      ✔ bridgeBurn exceed the balance of account should fail (114ms)
    Test approve function
      ✔ initial allowance should be zero
      ✔ approve should change the allowance (47ms)
    Test permit function
      ✔ can permit with signature (224ms)
      ✔ permit expired
      ✔ invalid permit (40ms)

  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 (55ms)
```

```
    ✔ Non admin cannot change admin of proxy
    ✔ Cannot change admin of proxy to zero address
  Test upgradeTo of AdminUpgradeabilityProxy
    ✔ Admin can upgrade the implementation of proxy (99ms)
    ✔ Non admin cannot upgrade the implementation of proxy (80ms)
    ✔ Cannot upgrade implementation to non contract address
  Test initialize function
    ✔ Initialize cannot call multiple times
  Test bridgeMint function
    ✔ l2Gateway can bridgeMint (55ms)
    ✔ non l2Gateway cannot bridgeMint
    ✔ bridgeMint address should not be zero
    ✔ bridgeMint should change the totalSupply (101ms)
  Test transfer function
    ✔ transfer success case (92ms)
    ✔ prohibited account cannot transfer (125ms)
    ✔ prohibited recipient account cannot receive (126ms)
    ✔ paused contract cannot do transfer (181ms)
    ✔ recipient address should not be zero (59ms)
    ✔ transfer with amount over balance should fail (52ms)
    ✔ transfer with amount over balance should fail (98ms)
    ✔ cannot transfer with amount is not a natural number (71ms)
  Test transferFrom function
    ✔ transferFrom success case (136ms)
    ✔ prohibited sender cannot transfer (151ms)
    ✔ prohibited recipient cannot receive (129ms)
    ✔ paused contract cannot do transfer (167ms)
    ✔ transfer amount that hasn't been approved should fail (54ms)
    ✔ transfer amount exceed approved amount should fail (145ms)
    ✔ transfer amount exceed approved amount should fail (141ms)
    ✔ recipient address should not be zero (106ms)
    ✔ cannot transferFrom with amount is not a natural number (105ms)
  Test bridgeBurn function
    ✔ bridgeBurn success case (131ms)
    ✔ bridgeBurn should change the totalSupply (115ms)
    ✔ bridgeBurn exceed the balance of account should fail (63ms)
    ✔ bridgeBurn exceed the balance of account should fail (96ms)
  Test approve function
    ✔ initial allowance should be zero
    ✔ approve should change the allowance (49ms)
  Test permit function
    ✔ can permit with signature (177ms)
    ✔ permit expired (99ms)
    ✔ invalid permit (95ms)
  Test initializeV2 function
    ✔ name and symbol changed (83ms)
    ✔ InitializeV2 cannot call multiple times (91ms)
  Test updateMetadata function
    ✔ updateMetadata and can permit with signature (164ms)
    ✔ Non rescuer cannot updateMetadata

Contract: Admin.sol
  Test changePauser function
    ✔ admin can change the pauser (102ms)
    ✔ non admin cannot change the pauser
    ✔ cannot change the pauser to zero address
  Test changeProhibiter function
    ✔ admin can change the prohibiter (56ms)
    ✔ non admin cannot change the prohibiter
    ✔ paused contract cannot change the prohibiter (63ms)
    ✔ cannot change the prohibiter to zero address
  Test changeWiper function
    ✔ admin can change the wiper (57ms)
    ✔ non admin cannot change the wiper
    ✔ paused contract cannot change the wiper (46ms)
    ✔ cannot change the wiper to zero address
  Test changeRescuer function
    ✔ admin can change the rescuer (47ms)
    ✔ non admin cannot change the rescuer
    ✔ paused contract cannot change the rescuer (61ms)
    ✔ cannot change the rescuer to zero address
```

```
Contract: Owner.sol
  Test changeOwner function
    ✔ owner can change the owner (40ms)
    ✔ non owner cannot change the owner
    ✔ paused contract cannot change the owner (41ms)
    ✔ cannot change the owner to zero address
  Test changeAdmin function
    ✔ owner can change the admin
    ✔ non owner cannot change the admin
    ✔ cannot change the admin to zero address

Contract: Pauser.sol
  Test pause function
    ✔ pauser can pause the contract (55ms)
    ✔ non pauser cannot pause the contract
    ✔ paused contract cannot pause again (54ms)
  Test unpause function
    ✔ pauser can unpause the contract (65ms)
    ✔ non pauser cannot unpause the contract (43ms)
    ✔ unpause contract cannot unpause again (97ms)

Contract: Prohibiter.sol
  Test prohibit function
    ✔ prohibiter can prohibit the account (52ms)
    ✔ non prohibiter cannot prohibit the account
    ✔ paused contract cannot prohibit account (59ms)
    ✔ prohibited account cannot prohibit again (45ms)
    ✔ prohibited account cannot be zero
  Test unprohibit function
    ✔ prohibiter can unprohibit the account (128ms)
    ✔ non prohibiter cannot unprohibit the account (50ms)
    ✔ paused contract cannot unprohibit account (104ms)
    ✔ non prohibited account cannot unprohibit
    ✔ unprohibit account cannot be zero

Contract: Rescuer.sol
  Test rescue function
    ✔ rescuer can rescue (209ms)
    ✔ non rescuer cannot rescue (67ms)
    ✔ paused contract cannot rescue (84ms)
    ✔ can not rescue more than balance (71ms)
    ✔ rescue should not change the totalSupply (118ms)

Contract: Wiper.sol
  Test wipe function
    ✔ wiper can wipe (147ms)
    ✔ non wiper cannot wipe (103ms)
    ✔ no prohibited address cannot be wipe (51ms)
    ✔ paused contract cannot be wipe (136ms)
    ✔ wipe should change the totalSupply (124ms)

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 (52ms)
    ✔ Non admin cannot change admin of proxy
    ✔ Cannot change admin of proxy to zero address
  Test upgradeTo of AdminUpgradeabilityProxy
    ✔ Admin can upgrade the implementation of proxy (107ms)
    ✔ Non admin cannot upgrade the implementation of proxy (78ms)
    ✔ Cannot upgrade implementation to non contract address
  Test initialize function
    ✔ Initialize cannot call multiple times
  Test bridgeMint function
    ✔ l2Gateway can bridgeMint (56ms)
    ✔ non l2Gateway cannot bridgeMint
    ✔ bridgeMint address should not be zero
```

```
    ✔ bridgeMint should change the totalSupply (50ms)
  Test transfer function
    ✔ transfer success case (87ms)
    ✔ prohibited account cannot transfer (84ms)
    ✔ prohibited recipient account cannot receive (92ms)
    ✔ paused contract cannot do transfer (123ms)
    ✔ recipient address should not be zero (47ms)
    ✔ transfer with amount over balance should fail (63ms)
    ✔ transfer with amount over balance should fail (109ms)
    ✔ cannot transfer with amount is not a natural number (63ms)
  Test transferFrom function
    ✔ transferFrom success case (148ms)
    ✔ prohibited sender cannot transfer (147ms)
    ✔ prohibited recipient cannot receive (170ms)
    ✔ paused contract cannot do transfer (135ms)
    ✔ transfer amount that hasn't been approved should fail (58ms)
    ✔ transfer amount exceed approved amount should fail (120ms)
    ✔ transfer amount exceed approved amount should fail (125ms)
    ✔ recipient address should not be zero (93ms)
    ✔ cannot transferFrom with amount is not a natural number (93ms)
  Test bridgeBurn function
    ✔ bridgeBurn success case (106ms)
    ✔ bridgeBurn should change the totalSupply (129ms)
    ✔ bridgeBurn exceed the balance of account should fail (59ms)
    ✔ bridgeBurn exceed the balance of account should fail (99ms)
  Test approve function
    ✔ initial allowance should be zero
    ✔ approve should change the allowance (50ms)
  Test permit function
    ✔ can permit with signature (153ms)
    ✔ permit expired (110ms)
    ✔ invalid permit (105ms)
  Test initializeV2 function
    ✔ name and symbol changed (73ms)
    ✔ InitializeV2 cannot call multiple times (79ms)
  Test updateMetadata function
    ✔ updateMetadata and can permit with signature (187ms)
    ✔ Non rescuer cannot updateMetadata


177 passing (1m)


Contract: GYEN.sol
  Test implementation of AdminUpgradeabilityProxy
    ✓ Admin can view address of implementation
    ✓ Non admin cannot view address of implementation (184ms)
  Test admin of AdminUpgradeabilityProxy
    ✓ Admin can view address of admin
    ✓ Non admin cannot view address of admin (43ms)
  Test changeAdmin of AdminUpgradeabilityProxy
    ✓ Admin can change admin of proxy (63ms)
    ✓ Non admin cannot change admin of proxy (74ms)
    ✓ Cannot change admin of proxy to zero address (62ms)
  Test upgradeTo of AdminUpgradeabilityProxy
    ✓ Admin can upgrade the implementation of proxy (242ms)
    ✓ Non admin cannot upgrade the implementation of proxy (117ms)
    ✓ Cannot upgrade implementation to non contract address (164ms)
  Test initialize function
    ✓ Initialize cannot call multiple times (56ms)
  Test mint function
    ✓ l2Gateway can mint (127ms)
    ✓ non l2Gateway cannot mint (59ms)
    ✓ mint address should not be zero (66ms)
    ✓ mint should change the totalSupply (72ms)
  Test transfer function
    ✓ transfer success case (118ms)
    ✓ prohibited account cannot transfer (89ms)
    ✓ prohibited recipient account cannot receive (86ms)
    ✓ paused contract cannot do transfer (95ms)
    ✓ recipient address should not be zero (47ms)
    ✓ transfer with amount over balance should fail (60ms)
```

```
    ✓ transfer with amount over balance should fail (82ms)
    ✓ cannot transfer with amount is not a natural number (103ms)
  Test transferFrom function
    ✓ transferFrom success case (115ms)
    ✓ prohibited sender cannot transfer (135ms)
    ✓ prohibited recipient cannot receive (157ms)
    ✓ paused contract cannot do transfer (134ms)
    ✓ transfer amount that hasn't been approved should fail (100ms)
    ✓ transfer amount exceed approved amount should fail (97ms)
    ✓ transfer amount exceed approved amount should fail (156ms)
    ✓ recipient address should not be zero (96ms)
    ✓ cannot transferFrom with amount is not a natural number (102ms)
  Test burn function
    ✓ burn success case (93ms)
    ✓ burn should change the totalSupply (115ms)
    ✓ burn exceed the balance of account should fail (94ms)
    ✓ burn exceed the balance of account should fail (75ms)
  Test approve function
    ✓ initial allowance should be zero
    ✓ approve should change the allowance (39ms)
  Test permit function
    ✓ can permit with signature (245ms)
    ✓ permit expired (84ms)
    ✓ invalid permit (106ms)
  Test updateMetadata function
    ✓ updateMetadata and can permit with signature (179ms)
    ✓ Non rescuer cannot updateMetadata (46ms)

Contract: OpToken_v1.sol
  Test initialize function
    ✓ Initialize cannot call multiple times (240ms)
    ✓ cannot initialize owner to zero address
    ✓ cannot initialize admin to zero address (38ms)
    ✓ cannot initialize prohibiter to zero address
    ✓ cannot initialize pauser to zero address
    ✓ cannot initialize wiper to zero address
    ✓ cannot initialize rescuer to zero address
    ✓ cannot initialize l1Address to zero address
    ✓ cannot initialize l2Gateway to zero address
  Test mint function
    ✓ only l2Gateway can mint (39ms)
    ✓ non l2Gateway cannot mint
    ✓ mint address should not be zero
    ✓ mint should change the totalSupply (52ms)
  Test transfer function
    ✓ transfer success case (54ms)
    ✓ prohibited account cannot transfer (79ms)
    ✓ paused contract cannot do transfer (78ms)
    ✓ recipient address should not be zero (58ms)
    ✓ transfer with amount over balance should fail (65ms)
    ✓ transfer with amount over balance should fail (91ms)
    ✓ cannot transfer with amount is not a natural number (45ms)
    ✓ cannot transfer to prohibited recipient (101ms)
  Test transferFrom function
    ✓ transferFrom success case (115ms)
    ✓ prohibited sender cannot transfer (102ms)
    ✓ paused contract cannot do transfer (121ms)
    ✓ transfer amount that hasn't been approved should fail (49ms)
    ✓ transfer amount exceed approved amount should fail (78ms)
    ✓ transfer amount exceed approved amount should fail (108ms)
    ✓ recipient address should not be zero (102ms)
    ✓ cannot transferFrom with amount is not a natural number (90ms)
    ✓ cannot transferFrom to prohibited recipient address (113ms)
  Test burn function
    ✓ burn success case (79ms)
    ✓ burn should change the totalSupply (57ms)
    ✓ burn exceed the balance of account should fail (47ms)
    ✓ burn exceed the balance of account should fail (92ms)
  Test approve function
    ✓ initial allowance should be zero
    ✓ approve should change the allowance (45ms)
  Test permit function
```

```
    ✓ can permit with signature (84ms)
    ✓ permit expired (49ms)
    ✓ invalid permit (56ms)
  Test updateMetadata function
    ✓ updateMetadata and can permit with signature (107ms)

Contract: Admin.sol
  Test changePauser function
    ✓ admin can change the pauser
    ✓ non admin cannot change the pauser
    ✓ cannot change the pauser to zero address
  Test changeProhibiter function
    ✓ admin can change the prohibiter
    ✓ non admin cannot change the prohibiter
    ✓ paused contract cannot change the prohibiter (51ms)
    ✓ cannot change the prohibiter to zero address
  Test changeWiper function
    ✓ admin can change the wiper
    ✓ non admin cannot change the wiper
    ✓ paused contract cannot change the wiper (85ms)
    ✓ cannot change the wiper to zero address
  Test changeRescuer function
    ✓ admin can change the rescuer
    ✓ non admin cannot change the rescuer
    ✓ paused contract cannot change the rescuer (49ms)
    ✓ cannot change the rescuer 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 (44ms)
    ✓ cannot change the owner to zero address
  Test changeAdmin function
    ✓ owner can change the admin
    ✓ non owner cannot change the admin
    ✓ cannot change the admin 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 (42ms)
  Test unpause function
    ✓ pauser can unpause the contract (41ms)
    ✓ non pauser cannot unpause the contract (68ms)
    ✓ unpause contract cannot unpause again (101ms)

Contract: Prohibiter.sol
  Test prohibit function
    ✓ prohibiter can prohibit the account
    ✓ non prohibiter cannot prohibit the account
    ✓ paused contract cannot prohibit account (46ms)
    ✓ prohibited account cannot prohibit again (107ms)
    ✓ prohibited account cannot be zero (39ms)
  Test unprohibit function
    ✓ prohibiter can unprohibit the account (72ms)
    ✓ non prohibiter cannot unprohibit the account (41ms)
    ✓ paused contract cannot unprohibit account (79ms)
    ✓ non prohibited account cannot unprohibit
    ✓ unprohibit account cannot be zero

Contract: Rescuer.sol
  Test rescue function
    ✓ rescuer can rescue (148ms)
    ✓ non rescuer cannot rescue (58ms)
    ✓ paused contract cannot rescue (110ms)
    ✓ can not rescue more than balance (52ms)
    ✓ rescue should not change the totalSupply (107ms)

Contract: Wiper.sol
  Test wipe function
```

```
    ✓ wiper can wipe (167ms)
    ✓ non wiper cannot wipe (124ms)
    ✓ no prohibited address cannot be wipe (128ms)
    ✓ paused contract cannot be wipe (293ms)
    ✓ wipe should change the totalSupply (139ms)

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 (57ms)
    ✓ Non admin cannot change admin of proxy
    ✓ Cannot change admin of proxy to zero address
  Test upgradeTo of AdminUpgradeabilityProxy
    ✓ Admin can upgrade the implementation of proxy (82ms)
    ✓ Non admin cannot upgrade the implementation of proxy (65ms)
    ✓ Cannot upgrade implementation to non contract address (50ms)
  Test initialize function
    ✓ Initialize cannot call multiple times
  Test mint function
    ✓ l2Gateway can mint
    ✓ non l2Gateway cannot mint
    ✓ mint address should not be zero (52ms)
    ✓ mint should change the totalSupply (53ms)
  Test transfer function
    ✓ transfer success case (61ms)
    ✓ prohibited account cannot transfer (84ms)
    ✓ prohibited recipient account cannot receive (83ms)
    ✓ paused contract cannot do transfer (109ms)
    ✓ recipient address should not be zero (71ms)
    ✓ transfer with amount over balance should fail (84ms)
    ✓ transfer with amount over balance should fail (67ms)
    ✓ cannot transfer with amount is not a natural number (52ms)
  Test transferFrom function
    ✓ transferFrom success case (91ms)
    ✓ prohibited sender cannot transfer (132ms)
    ✓ prohibited recipient cannot receive (127ms)
    ✓ paused contract cannot do transfer (97ms)
    ✓ transfer amount that hasn't been approved should fail (57ms)
    ✓ transfer amount exceed approved amount should fail (83ms)
    ✓ transfer amount exceed approved amount should fail (113ms)
    ✓ recipient address should not be zero (67ms)
    ✓ cannot transferFrom with amount is not a natural number (69ms)
  Test burn function
    ✓ burn success case (56ms)
    ✓ burn should change the totalSupply (79ms)
    ✓ burn exceed the balance of account should fail (47ms)
    ✓ burn exceed the balance of account should fail (74ms)
  Test approve function
    ✓ initial allowance should be zero
    ✓ approve should change the allowance (65ms)
  Test permit function
    ✓ can permit with signature (91ms)
    ✓ permit expired (60ms)
    ✓ invalid permit (72ms)
  Test updateMetadata function
    ✓ updateMetadata and can permit with signature (168ms)
    ✓ Non rescuer cannot updateMetadata (44ms)


174 passing (34s)
```

Code Coverage

The tests are exhibiting healthy code coverage.

Arbitrum: ----------|----------|----------|----------|----------|----------------|

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	93.44	100	88.89	91.3	
ArbToken_v1.sol	94.74	100	88.89	92.86	110,111,112
ArbToken_v2.sol	91.3	100	85.71	88.89	61,62,63
GYEN.sol	100	100	100	100	
ZUSD.sol	100	100	100	100	
contracts/Roles/	100	100	100	100	
Admin.sol	100	100	100	100	
Common.sol	100	100	100	100	
Owner.sol	100	100	100	100	
Pauser.sol	100	100	100	100	
Prohibiter.sol	100	100	100	100	
Rescuer.sol	100	100	100	100	
Wiper.sol	100	100	100	100	

Optimism: ----------|----------|----------|----------|----------|----------------|

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	89.58	100	86.67	88.68	
GYEN.sol	100	100	100	100	
OpToken_v1.sol	89.58	100	84.62	88.68	... 133,134,137
ZUSD.sol	100	100	100	100	
contracts/Roles/	100	100	100	100	
Admin.sol	100	100	100	100	
Common.sol	100	100	100	100	
Owner.sol	100	100	100	100	
Pauser.sol	100	100	100	100	
Prohibiter.sol	100	100	100	100	
Rescuer.sol	100	100	100	100	
Wiper.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
All files	94.51	100	94.87	94.44	

Changelog

- 2024-07-12 - Initial report
- 2024-07-29 - Final report

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, 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

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.