Confidential

DGLD Token Smart Contracts - Base

Final Security Audit

Confidential
October 2025

info@jitadigital.com

Jita Ltd

Executive Summary

This blockchain security audit covers DGLD repository https://github.com/goldtokensa/gtsa-base-smart-contracts , October 2025.

Code revision under audit: `eb9b5aeaf78713a19659635653c36b51499ff54d`
Updated code revision: `82cbf9c76064e5666dece5e88021f3b1e913d515`

Issues Found

- 0 critical issues ;
- 0 high severity issues ;
- 2 low severity issues ;
- 6 comments.

Key Audit Results

- 2 low severity issues resolved,
- 6 comments resolved

Issues Severity	Found	Resolved	Remaining
Critical	0	0	0
High	0	0	0
Low	2	2	0
Comments	6	6	0

DGLD Token Mainnet address: `0xA9299C296d7830A99414d1E5546F5171fA01E9c8`
DGLD Token TESTNET Base L2 address:
`0xd02f50E1017F493ffFFa70c8fCf09e349e11d6c9`

Audit result: Pass

1 Methodology

1.1 Audit Result

Jita's current methodology does not score audits, as such scores can be subjective, easy to manipulate, create a false sense of security and mislead people into comparing audits and security scores.

Instead, the final audit is only assigned a value PASS or FAIL.

Audit Result	
PASS	The issues found by the auditor(s) have all been resolved or explained by the development team.
FAIL	Some of the issues found by the auditor(s) have not all been resolved or explained by the development team.

1.2 Issues Severity Evaluation

The following section presents the audit findings.

Each issue is located as precisely as possible in the codebase, a description gives more information on the problem, and one or more recommendations are provided.

The client answer to each issue is included.

Each issue risk severity is estimated following the OWASP risk rating methodology and is assigned an overall risk severity which can take one of four values:

Issue Overall Risk Severity	
Critical	The issue compromises the project and must be fixed.
High	The issue is either very likely with low impact, or unlikely but with a big impact on the business. The issue must be fixed.
Low	The issue does not compromise the project, or it has a potentially minor impact on the system operation.
Comment	The issue does not compromise the project.

The issue overall risk severity is evaluated based on two metrics:
1) the likelihood of the issue being exploited (technical feasibility of exploit) and
2) the technical/business/financial impact of the issue.

An issue's overall risk severity is an estimate and may be revised higher or lower by the client company based on their own estimates of business impact.

2 Scope of Work

2.1 DGLD Base L2 Token

2.1.1 Github repositories

- git@github.com:goldtokensa/gtsa-base-smart-contracts.git
  Base L2 token for DGLD
  Git revision `eb9b5aeaf78713a19659635653c36b51499ff54d`

2.2 Assumptions and limitations

2.2.1 Time and version reliance

The security audits are conducted within a specified time frame and are reliant on the specific version of the target code, system and information provided by the client, its affiliates, or its partners. Therefore, the list of identified vulnerabilities and security issues identified during the audit should not be considered comprehensive and exhaustive. The audit results do not imply any guarantee that all potential vulnerabilities, flaws, or defects have been detected.

2.3 Disclaimer

Jita does not warrant that the Deliverables will identify all vulnerabilities.The Deliverables shall not be considered as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

Jita disclaims all other warranties express, implied, statutory, or otherwise, including, without limitation, the implied warranties of merchantability, non-infringement, and fitness for a particular purpose.

This audit is not financial advice.

2.4 Copyright

This report accessed through any source other than those controlled by Jita Digital or the Customer should not be considered authentic.

3 Audit Findings

3.1 /src

3.1.1 Low: Solidity language version pinning

3.1.1.1 Severity

Low

3.1.1.2 Files

`src/DGLDToken.sol`
`src/UpgradeableOptimismMintableERC20.sol`
`src/PausibleWithAccess.sol`
`src/Blacklistable.sol`

3.1.1.3 Functions

```
pragma solidity ^0.8.15;
```

3.1.1.4 Description

The solidity language version is set to 0.8.15 or above.
This allows developers to use different compiler versions, which can produce different bytecode versions.
Upstream files from the project use version pinning.

3.1.1.5 Recommendation

Use version pinning as follows:

```
pragma solidity 0.8.15;
```

3.1.1.6 Customer

This issue is addressed by configuring the Solidity version in foundry.toml.
solc_version = “0.8.15”

3.1.2 Low: minor risk of storage layout collisions during upgrades

3.1.2.1 Severity

Minor

3.1.2.2 Files

`src/DGLDToken.sol`
`src/UpgradeableOptimismMintableERC20.sol`

src/PausibleWithAccess.sol
src/Blacklistable.sol

3.1.2.3 Functions

```
uint256[50] private __gap;
```

3.1.2.4 Description

OpenZeppelin 4.x Proxy pattern uses an empty array of arbitrary size N at the end of an upgradeable smart contract. This allows developers to add up to N new state variables in future versions of the smart contract. The developer could make a mistake in a future version and still cause a collision. This is a low probability error and this pattern is recommended by Open Zeppelin 4.x.

3.1.2.5 Recommendation

Consider migrating to Open Zeppelin 5.x and Solidity 0.8.20 which introduce namespaced storage layout (see ERC-7201). This approach replaces the gap array with structs.
More info:
https://www.openzeppelin.com/news/introducing-openzeppelin-contracts-5.0#Simplifications

3.1.2.6 Customer

Resolved

3.1.3 Comment: replace String error messages with CustomErrors

3.1.3.1 Severity

Comment

3.1.3.2 Files

src/Blacklistable.sol

3.1.3.3 Functions

```
modifier notBlacklisted(address _account) {
        require(!blacklisted[_account], "Blacklistable: account is
blacklisted");
        _;
    }
```

3.1.3.4 Description

String error messages are non-standardized and gas consuming.

3.1.3.5 Recommendation

Use CustomErrors instead of require statements.
This lets developers standardize error processing and optimize gas consumption:

```
modifier notBlacklisted(address _account) {
        if(!blacklisted[_account]) {
            revert BlacklistedAccount(account);
        }
        _;
}
```

3.1.3.6 Customer

Resolved in fabaf64

3.1.4 Comment: replace String error messages with CustomErrors

3.1.4.1 Severity

Comment

3.1.4.2 Files

`src/UpgradeableOptimismMintableERC20.sol`

3.1.4.3 Functions

```
    /// @notice Internal function to check bridge authorization
    function _onlyBridge() internal view {
        require(msg.sender == BRIDGE, "OptimismMintableERC20: only
bridge can mint and burn");
    }
```

3.1.4.4 Description

String error messages are non-standardized and gas consuming.

3.1.4.5 Recommendation

Use CustomErrors instead of require statements.
This lets developers standardize error processing and optimize gas consumption:

```
modifier notBlacklisted(address _account) {
        if(!blacklisted[_account]) {
            revert BlacklistedAccount(account);
        }
```

```
        _;
}
```

3.1.4.6 Customer

Resolved in fabaf64

3.1.5 Comment: typo in file name

3.1.5.1 Severity

Comment

3.1.5.2 Files

`src/PausibleWithAccess.sol`

3.1.5.3 Functions

`N.A.`

3.1.5.4 Description

File name is PausibleWithAccess.sol

3.1.5.5 Recommendation

Rename to PausableWithAccess.sol

3.1.5.6 Customer

Resolved in 8bd7d79

3.1.6 Comment: minimize changes to upstream file version

3.1.6.1 Severity

Comment

3.1.6.2 Files

`src/PausibleWithAccess.sol`

3.1.6.3 Functions

```
pragma
Import {PausableUpgradeable}
```

3.1.6.4 Description

This file comes from https://github.com/coinbase/extended-optimism-mintable-token/blob/master/src/roles/PausableWithAccess.sol
There are minor changes that may not be necessary.
Minimizing changes from third-party files is a good practice and facilitates code reviews and updating to the latest version.

3.1.6.5 Recommendation

Revert changes to pragma (use version pinning for Solidity)
Add spaces to both imports smart contracts.

3.1.6.6 Customer

This issue is deprecated after migrating to OpenZeppelin 5 and Solidity 0.8.20.

3.1.7 Comment: minimize changes to upstream file version

3.1.7.1 Severity

Comment

3.1.7.2 Files

`src/Blacklistable.sol`

3.1.7.3 Functions

```
pragma
Import {AccessControlUpgradeable}
Modifier notBlacklisted(...)
```

3.1.7.4 Description

This file comes from /centrehq/centre-tokens/… and contains non-meaningful changes.
It is better practice to preserve the original file if possible, both for audits and for future updates / version tracking.

3.1.7.5 Recommendation

- Maintain solidity language version pinning
- Preserve spacing in `import { AccessControlUpgradeable }`
- Preserve formatting for modifier notBlacklisted (multi-line)

3.1.7.6 Customer

This issue is deprecated after migrating to OpenZeppelin 5 and Solidity 0.8.20.

3.1.8 Comment: _admin management key

3.1.8.1 Severity

Comment

3.1.8.2 Files

`DGLDToken.sol`

3.1.8.3 Functions

`initialize()`

3.1.8.4 Description

Parameter _admin is an Ethereum address that gets assigned all critical security roles: admin, pauser, blacklister.
That address is critical for correct business operation.

3.1.8.5 Recommendation

Make sure to use a SAFE multi sig wallet, another multi-sig wallet or a MPC wallet configured so that at least two signatures are required to authorize any function call using that key. Besides, it must be also possible to restore the wallet if one of the keys is lost.
Going further, consider adding a separate key for roles PAUSER and BLACKLISTER for day to day operations if these functions are frequently called.

3.1.8.6 Customer

Reviewed keys management plan and accelerated rollout.