BITGRIT

SMART CONTRACTS REVIEW

zokyo

October 10th 2024 | v. 1.0

Security Audit Score

PASS

Zokyo Security has concluded that this smart contract passed a security audit.

ZOKYO AUDIT SCORING BITGRIT

1. Severity of Issues:
    - Critical: Direct, immediate risks to funds or the integrity of the contract. Typically, these would have a very high weight.
    - High: Important issues that can compromise the contract in certain scenarios.
    - Medium: Issues that might not pose immediate threats but represent significant deviations from best practices.
    - Low: Smaller issues that might not pose security risks but are still noteworthy.
    - Informational: Generally, observations or suggestions that don't point to vulnerabilities but can be improvements or best practices.
2. Test Coverage: The percentage of the codebase that's covered by tests. High test coverage often suggests thorough testing practices and can increase the score.
3. Code Quality: This is more subjective, but contracts that follow best practices, are well-commented, and show good organization might receive higher scores.
4. Documentation: Comprehensive and clear documentation might improve the score, as it shows thoroughness.
5. Consistency: Consistency in coding patterns, naming, etc., can also factor into the score.
6. Response to Identified Issues: Some audits might consider how quickly and effectively the team responds to identified issues.

SCORING CALCULATION:

Let's assume each issue has a weight:
- Critical: -30 points
- High: -20 points
- Medium: -10 points
- Low: -5 points
- Informational: -1 point

Starting with a perfect score of 100:
- 0 Critical issues: 0 points deducted
- 0 High issues: 0 points deducted
- 0 Medium issues: 0 points deducted
- 1 Low issue: 1 resolved = 0 points deducted
- 0 Informational issues: 0 points deducted

Thus, the score is 100

TECHNICAL SUMMARY

This document outlines the overall security of the Bitgrit smart contract/s evaluated by the Zokyo Security team.

The scope of this audit was to analyze and document the Bitgrit smart contract/s codebase for quality, security, and correctness.

Contract Status

There were 0 critical issues found during the review. (See Complete Analysis)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract/s but rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that can withstand the Ethereum network's fast-paced and rapidly changing environment, we recommend that the Bitgrit team put in place a bug bounty program to encourage further active analysis of the smart contract/s.

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the Bitgrit repository:
Repo: https://github.com/bitgrit-official/bgr-token/blob/main/contracts/BGR.sol

Last commit - `0c8723aaf0e1d3af32cd7be4bfc6f2ed2f6c3b80`

Within the scope of this audit, the team of auditors reviewed the following contract(s):

- BGR.sol

During the audit, Zokyo Security ensured that the contract:

- Implements and adheres to the existing standards appropriately and effectively;
- The documentation and code comments match the logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices, efficiently using resources without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the most recent vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of Bitgrit smart contract/s. To do so, the code was reviewed line by line by our smart contract developers, who documented even minor issues as they were discovered. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01 Due diligence in assessing the overall code quality of the codebase.

02 Cross-comparison with other, similar smart contract/s by industry leaders.

03 Thorough manual review of the codebase line by line.

Executive Summary

BitGrit shares the mission to uncover new possibilities for Artificial Intelligence technology and Blockchain solutions in order to create a global platform to empower data scientists through an AI competition marketplace platform which makes use of big data created by the community. Companies which require AI capabilities (competition providers) can request for a competition to an infrastructure provider on the BRG network. That marketplace component creates an exchange of AI models to solve business specific issues for a prize pool in USD or native BRG token.

The BRG token is a utility token that aims to power BitGrit's ecosystem while taking advantage of Solidity's truly decentralized capabilities. Zokyo was tasked with the security audit component of BitGrit's development process to ensure that there were no exploitable vulnerabilities in the Solidity code itself and assess tokenomics to ensure the token functions as expected within the ecosystem. Overall, the developers have built the contract with security in mind solely relying on battle hardened dependencies to carry the weight of the typical ERC20 compliant logic through the OpenZeppelin contracts as opposed to rewriting the contract from scratch which can create unnecessary high-severity security issues.

The contract audit yielded 1 Low severity finding which involved an issue around floating pragmas where an unlikely bug to trigger, still poses a security risk. Due to using tried and tested libraries that have withstood the test of time to support their design goals, the BGT has had favorable results during the security audit.

It's recommended that the development team formulates a plan to slowly distribute the token and ensure that tokens are spread amongst the public evenly perhaps through competitions, through staking or airdrops for active users to prevent a small number of users controlling a large amount of the total supply.

STRUCTURE AND ORGANIZATION OF THE DOCUMENT

For the ease of navigation, the following sections are arranged from the most to the least critical ones. Issues are tagged as “Resolved” or “Unresolved” or “Acknowledged” depending on whether they have been fixed or addressed. Acknowledged means that the issue was sent to the Bitgrit team and the Bitgrit team is aware of it, but they have chosen to not solve it. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Client or remains disregarded by the Client. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

FINDINGS SUMMARY

#	Title	Risk	Status
1	Usage of Floating Pragmas May Compile Contracts With an Unexpected Version	Low	Resolved

LOW-1 | RESOLVED

Usage of Floating Pragmas May Compile Contracts With an Unexpected Version

The BitGrit (BGR) token contract uses the pragma `^0.8.20`. By enabling the use of multiple compilers for this contract, they can lead to unexpected issues in the Solidity contracts as the code may have been tested with a different compiler than the one that is being used to deploy the contracts.

Recommendation:
It's recommended that the contract pins the pragma version which is used when testing the code.

	BGR.sol
Re-entrancy	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Unexpected Ether	Pass
Delegatecall	Pass
Default Public Visibility	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Short Address/ Parameter Attack	Pass
Unchecked CALL Return Values	Pass
Race Conditions / Front Running	Pass
General Denial Of Service (DOS)	Pass
Uninitialized Storage Pointers	Pass
Floating Points and Precision	Pass
Tx.Origin Authentication	Pass
Signatures Replay	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass

We are grateful for the opportunity to work with the Bitgrit team.

The statements made in this document should not be interpreted as an investment or legal advice, nor should its authors be held accountable for the decisions made based on them.

Zokyo Security recommends the Bitgrit team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo