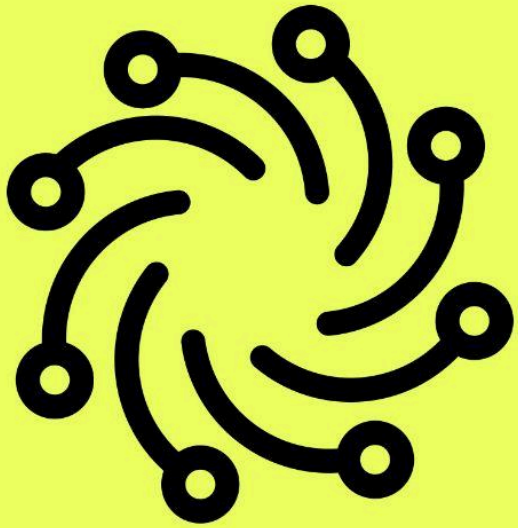
audita



Botto

Governance and Rewards

Smart Contract Security Audit

September 26th, 2024

Prepared for:
Botto – Decentralised Autonomous Artist (*botto.com*)

Presented By:
Audita Security (*audita.io*)

Document

The contents of this document may include confidential information pertaining to the IT systems, intellectual property, and possible vulnerabilities along with methods of exploitation that the Client may possess. The report that contains this confidential information can be utilized internally by the Client, and can be made available to the public after all vulnerabilities are addressed, depending on the decision of the Client.

Approved by: Jose Lopez, Reni Dimitrova @ Audita.io
Contracts: *BottoActiveRewards.sol, BottoGovernance.sol*
Network: Ethereum, Base
Programming language: Solidity
Method: Manual Audit
Client Website: https://www.botto.com/
Timeline: 20/09/2024 - 26/09/2024

Table of Contents

Executive Summary

Manual Audit

During the manual audit conducted by our experts, we did not identify any Critical or High severity vulnerabilities.

We identified 2 Medium and 1 Low severity vulnerabilities.

2 Informational issues were indicated, as well as 2 recommendations for Gas Optimization.

Overall Assessment

Botto's Governance and Rewards contracts are safe for deployment and pose no risks to the protocol and its users.

Severity	Count	Acknowledged
Critical	0	-
High	0	Yes
Medium	2	Yes
Low	1	Yes
Informational	2	Yes
GAS	2	Yes

Documentation

We recommend this report, as well as specific information from this report to be included in protocol's official Documentation, as soon as code is deployed.

Audita Vulnerability Classifications

Audita follows the most recent standards for vulnerability severities, taking into consideration both the possible impact and the likelihood of an attack occurring due to a certain vulnerability.

Severity	Description
Critical	Critical vulnerability is one where the attack is more straightforward to execute and can lead to exposure of users' data, with catastrophic financial consequences for clients and users of the smart contracts.
High	The vulnerability is of high importance and impact, as it has the potential to reveal the majority of users' sensitive information and can lead to significant financial consequences for clients and users of the smart contracts.
Medium	The issue at hand poses a potential risk to the sensitive information of a select group of individual users. If exploited, it has the potential to cause harm to the client's reputation and could result in unpleasant financial consequences.
Low	The vulnerability is relatively minor and not likely to be exploited repeatedly, or is a risk that the client has indicated is not impactful or significant, given their unique business situation.
Informational	The issue may not pose an immediate threat to ongoing operation or utilization, but it's essential to consider implementing security and software engineering best practices, or employing backup measures as a safety net.

Scope

The security assessment was scoped to the following smart contract in Botto's code repository:

Contract name
BottoActiveRewards.sol
BottoGovernance.sol

The codebase has been audited up to and including commit:

19c5fa401613c610d57199051d4033414a98d22b

Findings

Summary

Code	Description	Severity	Fixes
[BOTTO-01]	(*unstake*) No slashing and timelock mechanisms for `BottoGovernance	Medium	Acknowledged
[BOTTO-02]	(*stake*) No upper limit for staking amount in `BottoGovernance	Medium	Acknowledged
[BOTTO-03]	Lack of Two Step Ownership transferring	Low	Acknowledged
[BOTTO-04]	Lack of pause mechanism	Informational	Acknowledged
[BOTTO-05]	Missing event emission for granting roles	Informational	Acknowledged
[BOTTO-06]	The *botto* variable should be declared immutable	GAS	Acknowledged
[BOTTO-07]	Use a smaller 'uint' type for the nonces to save storage	GAS	Acknowledged

Detailed Findings

[BOTTO-01]	(*unstake*) No slashing and timelock mechanisms for BottoGovernance	`Medium`

Function: unstake
Details:
No penalty for unstaking could lead to governance attacks where users stake right before a vote and unstake immediately after. Users can unstake at any time, which could be problematic for ongoing governance decisions.

Recommendation:
Implement a mechanism to prevent unwanted governance participant behaviour.

[BOTTO-02]	(*stake*) No upper limit for staking amount in `BottoGovernance	`Medium`

Function: stake
Details:
No cap on the total amount that can be staked could lead to potential issues if a single user stakes an extremely large amount. This could lead to centralization of governance power as a single user could have a significant influence on the governance decisions.

Recommendation:
Confirm this is the desired system design, and introduce mechanisms to prevent centralization risks.

[BOTTO-03]	Lack of Two Step Ownership transferring	Low

Details:
The *BottoGovernance* contract uses the *OwnableUpgradeable* functionality, relying on the owner to recover the extra tokens sent.
In case of incorrect ownership transferring, the malicious user may withdraw those tokens or they may be locked.

Recommendation:
Use *Ownable2StepUpgradeable* to ensure the security of the privileged user functionality.

[BOTTO-04]	Lack of pause mechanism	Informational

Recommendation:
Both contracts do not have a pause mechanism. As a best practice, introduce a pause mechanism in the contract to stop the contract in case of any emergency.

[BOTTO-05]	Missing event emission for granting roles	Informational

Details:
It's best practice to emit events whenever a role is granted or revoked.
Found in `BottoGovernance.sol`:

```solidity
    function initialize() public initializer {
        __EIP712_init("BOTTO-ACTIVE-REWARDS", "1.0.0");
        __ReentrancyGuard_init();
```

```
@>        _grantRole(DEFAULT_ADMIN_ROLE, msg.sender); // Possible event for granting role
@>        _grantRole(Constants.REWARD_ROLE, msg.sender); // Possible event for granting
role
    }
```

Recommendation:
Implementing event emission for granting or revoking roles will help in tracking the role changes and can be useful for debugging and auditing purposes.

[BOTTO-06]	The *botto* variable should be declared immutable	GAS

Details:
The *botto* variable in the *BottoGovernance* contract is set in the constructor and is never changed.
Therefore, it may be declared as immutable to use less gas.

Recommendation:
Consider declaring *botto* variable as immutable.

[BOTTO-07]	Use a smaller ‘uint’ type for the nonces to save storage	GAS

Details:

The *claimNonce* and *recoverNonce* variables are of type ‘uint256’. In order for the nonce variables to overflow, the underlying functions will need to be called 2^256 times. This is not feasible in practice and a smaller ‘uint’ type can be used to save storage.

Recommendation:

If ‘uint128’ is used then both the *claimNonce* and *recoverNonce* can be stored in a single storage slot.

Potential Gas Savings:

* SSTORE (storing to a new storage slot): 20,000 gas
* SSTORE (updating an existing storage slot): 5,000 gas
* By packing these variables, you save one SSTORE operation per initialization/update.

Overall Assessment

Botto's Governance and Rewards contracts are safe for deployment and pose no risks to the protocol and its users.

Severity	Count	Acknowledged
Critical	0	-
High	0	Yes
Medium	2	Yes
Low	1	Yes
Informational	2	Yes
GAS	2	Yes

Recommendations

Audita has put forward the following recommendations for Botto's Governance and Rewards smart contracts:

- Implement slashing or timelock mechanism to prevent unwanted governance participant behavior.
- Confirm having no upper limit for staking amount is the desired system design, and introduce mechanisms to prevent centralization risks.
- Use *Ownable2StepUpgradeable* to ensure the security of the privileged user functionality.
- Both contracts do not have a pause mechanism. As a best practice, introduce a pause mechanism in the contract to stop the contract in case of any emergency.
- Implementing event emission for granting or revoking roles will help in tracking the role changes and can be useful for debugging and auditing purposes.
- Consider declaring *botto* variable as immutable to save on gas.
- If a smaller uint type is used, such as 'uint128' - both the *claimNonce* and *recoverNonce* can be stored in a single storage slot.

Fixes

Botto team is dedicated and responsive, cooperating to acknowledge and implement the above recommendations.

Disclaimer

This audit makes no statements or warranties on the security of the code. **While we have conducted the analysis to our best abilities and produced this report in line with latest industry developments, it is important to not rely on this report only.** In order for contracts to be considered as safe as possible, the industry standard requires them to be checked by several independent auditing bodies. Those can be other audit firms or public bounty programs.

Smart contract platforms, their programming languages, and other software components are not immune to vulnerabilities that can be exploited by hackers. As a result, although a smart contract audit can help identify potential security issues, it cannot provide an absolute guarantee of the audited smart contract's security.