



PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Davos

17 October 2023



paladinsec.co



info@paladinsec.co

Table of Contents

Table of Contents	2
Disclaimer	3
1 Overview	4
1.1 Summary	4
1.2 Contracts Assessed	4
1.3 Findings Summary	5
1.3.1 General Issues	6
1.3.2 BribeFactory	6
1.3.3 Bribe	6
2 Findings	7
2.1 General Issues	7
2.1.1 Issues & Recommendations	8
2.2 BribeFactory	11
Core Functionalities	11
2.2.1 Privileged Functions	12
2.2.2 Issues & Recommendations	13
2.3 Bribe	14
2.3.1 Privileged Functions	14
2.3.2 Issues & Recommendations	15

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocacy for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains the right to re-use any and all knowledge and expertise gained during the audit process, including, but not limited to, vulnerabilities, bugs, or new attack vectors. Paladin is therefore allowed and expected to use this knowledge in subsequent audits and to inform any third party, who may or may not be our past or current clients, whose projects have similar vulnerabilities. Paladin is furthermore allowed to claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for VoteBoost on the Arbitrum network. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	VoteBoost
URL	https://voteboost.xyz
Platform	Polygon
Language	Solidity
Preliminary Contracts	Bribe: https://github.com/voteboost/Bribes/commit/bc28e0b9a0f533dd3b407bf59edb0fbb57da73b7
Resolution 1	e923fc7136fa597ca2b9084c0de150b317d51506
Resolution 2	https://github.com/voteboost/Bribes commit: 13ea0c04776e58a3767fab48273cffb701cf2ac7 Fixed decimals

1.2 Contracts Assessed

Name	Contract	Live Code Match
BribeFactory		
Bribe		

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
● Governance	0	-	-	-
● High	2	1	-	1
● Medium	1	1	-	-
● Low	5	2	-	3
● Informational	4	2	-	2
Total	12	6	-	6

Classification of Issues

Severity	Description
● Governance	Issues under this category are where the governance or owners of the protocol have certain privileges that users need to be aware of, some of which can result in the loss of user funds if the governance's private keys are lost or if they turn malicious, for example.
● High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
● Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
● Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
● Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 General Issues

ID	Severity	Summary	Status
01	MEDIUM	The protocol does not work with tokens with decimals other than 18	✓ RESOLVED
02	LOW	Redundant auth functions	ACKNOWLEDGED
03	LOW	The whole codebase lacks events for key state changes	✓ RESOLVED
04	INFO	Project does not work with tokens that have a fee on transfer	ACKNOWLEDGED
05	INFO	uint and uint256 is used across contracts	✓ RESOLVED

1.3.2 BribeFactory

ID	Severity	Summary	Status
06	LOW	Lack of events for key state changes	✓ RESOLVED

1.3.3 Bribe

ID	Severity	Summary	Status
07	HIGH	Stale active period will miscalculate rewards	✓ RESOLVED
08	HIGH	Admin can steal all the ERC20 tokens from the contract	ACKNOWLEDGED
09	LOW	rewardTokens can run out of gas	ACKNOWLEDGED
10	LOW	_getRewardsHelper calculates the wrong start epoch of a new user	ACKNOWLEDGED
11	INFO	Users can claim rewards for other users which may affect their taxes without them knowing	ACKNOWLEDGED
12	INFO	Gas optimizations	✓ RESOLVED





2 Findings



2.1 General Issues



The issues listed in this section apply to the protocol as a whole. Please read through them carefully and take care to apply the fixes across the relevant contracts.





2.1.1 Issues & Recommendations

Issue #01		The protocol does not work with tokens with decimals other than 18
Severity	 MEDIUM SEVERITY	
Description	<p>Throughout the contracts, the computations assume that the tokens have 18 decimals as multiple computations are done with 1e18.</p> <p>e.g. Bribe:</p> <pre>uint256 _rewards = (_rewardPerToken * _balance) / 1e18;</pre>	
Recommendation	Consider not using any token that is different than 18 decimals, otherwise the computations will not work as expected.	
Resolution	 RESOLVED	<p>The protocol uses now the decimals of the tokens that are used within various contracts.</p>
Issue #02		Redundant auth functions
Severity	 LOW SEVERITY	
Description	<p>The following code containing authentication functions is present in the Bribe and BribeFactory contracts:</p> <pre>// --- Auth --- mapping (address => uint) public wards; function rely(address guy) external auth { wards[guy] = 1; } function deny(address guy) external auth { wards[guy] = 0; } modifier auth { require(wards[msg.sender] == 1, "Bribe/ not-authorized"); _; }</pre>	
Recommendation	Consider creating a specific contract called Auth.sol and inheriting it in the previously mentioned contracts.	
Resolution	 ACKNOWLEDGED	<p>The contracts that use the wards approach mimic the MakerDAO format.</p>

Issue #03 The whole codebase lacks events for key state changes	
Severity	 LOW SEVERITY
Description	<p>In most of the contracts, there are several functions that change ownership, permissions, or update key states of the codebase but are missing events.</p> <p>Some example of functions missing events:</p> <ul style="list-style-type: none"> - Bribe.sol: setMinter(), setVoter(), addRewardTokens()
Recommendation	Add events for each function and emit them accordingly.
Resolution	 RESOLVED

Issue #04 Project does not work with tokens that have a fee on transfer	
Severity	 INFORMATIONAL
Description	The whole architecture has several sections which will not work with tokens with a fee on transfer as this will break the whole system.
Recommendation	Consider not using such tokens.
Resolution	 ACKNOWLEDGED



Issue #05	uint and uint256 is used across contracts
Severity	 INFORMATIONAL
Description	Throughout the contracts, uint and uint256 types are used inconsistently.
Recommendation	Consider using only uint256 for a better readability of the codebase.
Resolution	 RESOLVED



2.2 BribeFactory

BribeFactory acts as the foundational layer for deploying and managing individual Bribe contracts within the decentralized ecosystem. Serving as a control tower, it ensures that every Bribe contract is instantiated with the necessary parameters and provides tools for holistic management over their lifecycle.

Core Functionalities

1. Bribe Deployment

- **Automated Creation:** The BribeFactory facilitates the automated generation and deployment of Bribe contracts, allowing for a streamlined process without the need for repeated manual code deployment.
- **Standardized Structure:** Each Bribe contract deployed follows a predetermined template, ensuring consistency and reliability across all Bribes.

2. Bribe Management

- **Contract Permissions:** With the ever-evolving nature of decentralized systems, there may be a need to adjust or modify permissions associated with a specific Bribe contract. The BribeFactory ensures administrators can effectively manage and alter these permissions as required.
- **Reward Token Modification:** The flexibility of the system is showcased in the BribeFactory's ability to change the reward token associated with any Bribe contract. This ensures that the reward system remains adaptable to shifts in market dynamics, token utility, or strategic alignment.

3. Holistic Overview

- **Centralized Repository:** The BribeFactory serves as a centralized point of reference to view and manage all deployed Bribe contracts. This consolidated view aids in efficient management and audit of the system.



- State Monitoring: Beyond deployment, the BribeFactory continually monitors the state of each Bribe, ensuring they operate within desired parameters and enabling timely intervention if anomalies are detected.

2.2.1 Privileged Functions

- addDefaultRewardToken
- removeDefaultRewardToken
- setBribeVoter
- setBribeMinter
- transferBribeOwnership
- recoverERC20AndUpdateData
- recoverERC20From
- setVoter
- setImplementation
- addRewardTokensToBribes
- addRewardTokensToBribe



2.2.2 Issues & Recommendations

Issue #06	Lack of events for key state changes
Severity	 LOW SEVERITY
Description	Functions like <code>setVoter()</code> , <code>addDefaultRewardToken()</code> , <code>removeDefaultRewardToken()</code> and <code>transferBribeOwnership()</code> perform important state changes on the contract but do not emit an event to reflect them.
Recommendation	Consider emitting events in these functions.
Resolution	 RESOLVED

2.3 Bribe

Bribe is responsible for giving the users the possibility to earn rewards based on their voting power at a respective epoch. Each epoch is currently set to 2 weeks.

The usual flow for a user is to vote for specific gauges and then the rewards are distributed to those gauges. As an extra incentive, the Bribe contract is used as a mechanism to incentivize users to be active and vote to receive extra rewards.



The admin is able to add other reward tokens, change permission and withdraw any ERC20 token in the contract.


2.3.1 Privileged Functions

- `addRewardTokens`
- `setVoter`
- `setMinter`
- `recoverERC20AndUpdateData`
- `emergencyRecoverERC20`



2.3.2 Issues & Recommendations

Issue #07	Stale active period will miscalculate rewards
Severity	 HIGH SEVERITY
Description	<p>Rewards are claimed through <code>getRewards()</code>, which calls the view only function <code>earnedByOwner()</code> where the calculation of rewards is done. During the calculation of the rewards, <code>getEpochStart()</code> is called, which returns the last active period from the minter.</p> <p>The minter contract is the <code>EpochsTimer</code> contract that is within this scope.</p> <p>The active period from the <code>EpochsTimer</code> contract is the timestamp when the minter was last updated. In the current implementation, the timestamp is stale and will miscalculate the rewards.</p>
Recommendation	<p>Call <code>update_period()</code> from the minter (<code>EpochsTimer</code>) contract before using <code>getEpochStart()</code>.</p> <p>When claiming rewards, <code>notifyRewardAmount</code> will also be called at the end.</p> <p><code>notifyRewardAmount</code>, in this case, will have the correct active period with the proposed fix, but <code>update_period()</code> should also be called inside <code>notifyRewardAmount</code> for any instance where it is called independently or in other functions of the codebase.</p>
Resolution	 RESOLVED
	<p><code>update_period</code> is called when the <code>notifyRewardAmount</code> is called.</p>


Issue #08**Admin can steal all the ERC20 tokens from the contract****Severity** HIGH SEVERITY**Description**

The following function allows the auth role to sweep all the tokens an authorized party wants from the contract:

```
function emergencyRecoverERC20(address tokenAddress, uint256 tokenAmount) external auth {  
  
    require(tokenAmount <=  
IERC20(tokenAddress).balanceOf(address(this)));  
    IERC20(tokenAddress).safeTransfer(owner(), tokenAmount);  
    emit Recovered(tokenAddress, tokenAmount);  
}
```


Recommendation

Consider only using multi-signature wallets with KYC-ed participants for privileged roles within the whole architecture. Additionally, consider if an Access Control List approach might be desired for such high privileges, where a special role can be implemented under a specific multi-signature that can access only this function, in this way, the risk can be lowered in case a specific multi-signature is exploited.

Resolution ACKNOWLEDGED

The team has stated that all the owners of the bribes will be under multi-signature contracts.




Issue #09**rewardTokens can run out of gas****Severity** LOW SEVERITY**Description**

rewardTokens is an array that contains all the reward tokens that were used within the Bribe. As this array only increases and never decreases, at some point, it might to run out of gas if enough tokens accumulate over the time.

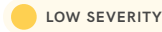
Recommendation

Consider implementing a view function with pagination
`getRewardTokens(uint256 start, uint256 length)`
that can be used to get the tokens using pagination.

Resolution ACKNOWLEDGED

The team has stated that the rewardTokens will not go out of bounds for now.



Severity**Description**

Within the `_getRewardsHelper`, if a user claims rewards for the first time, `_userLastTime` is set as the first bribe timestamp - `TWO_WEEKS`.

Line 131

```
if (_userLastTime < firstBribeTimestamp) _userLastTime =  
firstBribeTimestamp - TWO_WEEKS;
```

Then, within `_earnedAtEpoch`, a function `balanceOfOwnerAt` is called again which subtracts `TWO_WEEKS`, which puts the user balance two weeks in the past than the first active period.

After this, the `rewardPerToken` is called to get the reward at that particular epoch start, which again the initial timestamp is lowered by 2 weeks, and will return 0 rewards as that epoch does not exist.

```
function _earnedAtEpoch(address _owner, address  
_rewardToken, uint256 _timestamp) internal view returns  
(uint256) {  
    uint256 _balance = balanceOfOwnerAt(_owner, _timestamp);  
    if (_balance == 0) return 0;  
    else {  
        uint256 _rewardPerToken = rewardPerToken(_rewardToken,  
_timestamp);  
        uint256 _rewards = (_rewardPerToken * _balance) /  
1e18;  
        return _rewards;  
    }  
}
```


As we understand, the behavior should be as such:

If the user never claimed a reward, the initial start time should be the first epoch (which is the first `active_period` of the minter + two weeks). Then the balance for the reward calculation should be taken at the beginning of the epoch, hence the subtraction of the `TWO_WEEKS` within the `balanceOfOwnerAt`.

Recommendation

Consider setting the `_userLastTime` as the `firstBribeTimestamp`.

Resolution

 ACKNOWLEDGED

The client has stated: "Because bribes are recorded for the next epoch based on the weights in the current epoch, it is necessary to keep in mind that the portion of rewards for the next bribes (or next epoch) are determined by the weights in the current epoch of the voter contract. Hence, `rewardPerToken` and `balanceOfAt` subtract `TWO_WEEKS` because both functions are in the context of a bribe contract. Furthermore, the start of the loop from `_userLastTime` which is `TWO_WEEKS` (or one epoch) behind is for safety purposes."

Issue #11

Users can claim rewards for other users which may affect their taxes without them knowing

Severity

 INFORMATIONAL


Description

Users can claim rewards for any other user through `getRewardsForOwner()`. This can backfire in the future because it does change the balance of the `_rewardToken` for the end user, which is a taxable event they might not be aware of.

Recommendation

Consider making it very clear to users that anyone can claim their rewards for them, or remove the function if it is not strictly necessary.

Resolution

 ACKNOWLEDGED

Issue #12**Gas optimizations****Severity** INFORMATIONAL**Description****Examples**

```
uint256 k = 0;  
uint256 reward = 0;
```

Default initialization of variables consumes gas. Consider removing the initialization with the default value.

Line 160-161

```
address _owner = msg.sender;  
_getRewardsHelper(_owner, tokens);
```

This owner variable is obsolete, consider calling the `_getRewardsHelper` directly with `msg.sender` to save gas.

Line 146-147

```
uint256 _rewards = (_rewardPerToken * _balance) / 1e18;  
return _rewards;
```

The `_rewards` variable is obsolete, consider returning the computation result directly.

Recommendation

Consider implementing the gas optimizations mentioned above.

Resolution RESOLVED



PALADIN
BLOCKCHAIN SECURITY