PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Bouje Finance

16 January 2022

paladinsec.co

info@paladinsec.co

Table of Contents

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 advocation 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.

1 Overview

This report has been prepared for Bouje Finance on the Fantom 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	Bouje Finance
URL	https://www.bouje.finance/
Platform	Fantom
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
`BoujeToken`	`0xE509Db88B3c26D45f1fFf45b48E7c36A8399B45A`	MATCH
`MasterChef`	`0x1277dd1dCbe60d597aAcA80738e1dE6cB95dCB54`	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	2	2	-	-
Medium	2	2	-	-
Low	6	4	-	2
Informational	8	6	2	-
Total	**18**	**14**	**2**	**2**

Classification of Issues

Severity	Description
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 BoujeToken

ID	Severity	Summary	Status
01	LOW	`mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef	ACKNOWLEDGED
02	INFO	Unused `MAX_CAP_SUPPLY` variable	RESOLVED
03	INFO	Governance functionality is broken	RESOLVED
04	INFO	`delegateBySig` can be frontrun and cause denial of service	RESOLVED

1.3.2 MasterChef

ID	Severity	Summary	Status
05	HIGH	Severely excessive rewards issue when reflective tokens are added	RESOLVED
06	HIGH	Token ownership can be transferred to mint and dump tokens	RESOLVED
07	MEDIUM	Unnecessary and inconsistent `devReward` logic	RESOLVED
08	MEDIUM	Setting `devAddress` or `feeAddress` to the zero address will break most functionality	RESOLVED
09	LOW	Users can be kicked out of inactive pools by the governance	ACKNOWLEDGED
10	LOW	Usage of block numbers on the Fantom network could cause irregular reward emissions	RESOLVED
11	LOW	The `pendingBouje` function will revert if `totalAllocPoint` is zero	RESOLVED
12	LOW	Minting can exceed max token supply	RESOLVED
13	LOW	`updateEmissionRate` has no maximum safeguard	RESOLVED
14	INFO	`boujeToken` can be made `immutable`	RESOLVED
15	INFO	Pools use the contract balance to figure out the total deposits	RESOLVED
16	INFO	References to FISH token	RESOLVED
17	INFO	`deposit`, `withdraw` and `emergencyWithdraw` can be made `external`	PARTIAL
18	INFO	Lack of events for `add`, `set`, `updateStartBlock`, `switchActivePool`, `forceWithdraw` and `transferBoujeTokenOwnership`	PARTIAL

2 Findings

2.1 BoujeToken

The BoujeToken token is a simple ERC-20 token which will be used as the main reward token for the Masterchef. The contract allows for BoujeToken tokens to be minted when the `mint` function is called by the owner of the contract, which at the time of deployment would be the Bouje team. Ownership of the contract is generally transferred to the Masterchef via the transferOwnership function for emission rewards to be minted and distributed to users staking in the Masterchef. Users should therefore carefully inspect that ownership of this contract has been transferred to the Masterchef. No maximum supply is enforced within the token contract.

The `mint` function can be used to pre-mint tokens for various uses including injection of initial liquidity, token presale, airdrops, and others.

2.1.1 Token Overview

Address	TBC
Token Supply	Unlimited
Decimal Places	18
Transfer Max Size	No maximum
Transfer Min Size	No minimum
Transfer Fees	None
Pre-mints	0

2.1.2 Privileged Roles

The following functions can be called by the owner of the contract:

- `mint`
- `renounceOwnership`
- `transferOwnership`

2.1.3 Issues & Recommendations

Issue #01	`mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef
Severity	LOW SEVERITY
Description	The `mint` function could be used to pre-mint tokens for legitimate uses including, but not limited to, the injection of initial liquidity, token presale, or airdrops; however, this function may also be used to pre-mint and dump tokens when the token contract has been deployed but before ownership is set to the Masterchef contract. This risk is prevalent amongst less-reputable projects, and any pre-mints can be prominently seen on the Blockchain.
Recommendation	Consider being forthright if this `mint` function is to be used by letting your community know how much was minted, where they are currently stored, if a vesting contract was used for token unlocking, and finally the purpose of the mints.
Resolution	ACKNOWLEDGED This issue will be marked as resolved once ownership has been transferred to the Masterchef.

Issue #02	Unused MAX_CAP_SUPPLY variable
Severity	INFORMATIONAL
Location	Line 565 `uint256 constant MAX_CAP_SUPPLY = 88888 ether;`
Description	The code contains a constant variable that denotes a maximum supply of 88,888 tokens. However, this variable is unused throughout the contract and there is therefore no maximum supply.
Recommendation	Consider removing the unused variable.
Resolution	RESOLVED `MAX_CAP_SUPPLY` is replaced by `MAX_VIVE_SUPPLY` in the Masterchef.

Issue #03	Governance functionality is broken
Severity	INFORMATIONAL
Description	Although there is YAM-related delegation code in the token contract which is usually used for governance and voting, the delegation code can be abused as the delegates are not moved during transfers and burns. This allows for double spending attacks on the voting mechanism. It should be noted that this issue is present in pretty much every single farm out there including PancakeSwap and even SushiSwap.
Recommendation	The broken delegation-related code can be removed to reduce the size of the contract. If voting is ever desired, it can still be done through snapshot.org, used by many of the larger projects.
Resolution	RESOLVED YAM-related delegation code has been removed.

Issue #04	`delegateBySig` can be frontrun and cause denial of service
Severity	INFORMATIONAL
Description	Currently if `delegateBySig` is executed twice, the second execution will be reverted. It is thus in theory possible for a bot to pick up `delegateBySig` transactions in the mempool and execute them before a contract can. The issue with this is that the rest of said contract functionality would be lost as well. This could be a problem in case it would have been executed by a contract that would have rewarded you for your delegation for example.
Recommendation	Similar to the broken governance functionality issue, this can just be removed.
Resolution	RESOLVED YAM-related delegation code has been removed.

2.2 MasterChef

The Bouje Masterchef is a modified fork of the Goose Masterchef. A notable feature of forking the latter is the removal of the migrator function from Pancakeswap, which has been used maliciously to steal tokens staked in the Masterchef. Within the Masterchef, a minting limit of 8,888 has been set while the Masterchef is the owner of the token contract.

We commend Bouje Finance on their decision to fork a relatively safer version of the Masterchef. Deposit fees are capped at 5%, which will be highly appreciated by users. Finally, the emission schedule is currently set to 0.05 tokens per second with 10% of the emissions minted extra to the `devAddress`.

2.2.1 Issues & Recommendations

Issue #05	Severely excessive rewards issue when reflective tokens are added
Severity	HIGH SEVERITY
Description	When tokens with a transfer tax are added to the pools, this will result in significant excessive rewards due to the way the Masterchef handles the reward mechanism. Rewards can be heavily inflated when the balance of the Masterchef no longer matches that of user deposits. This happens for example with reflective tokens. This flaw in the Masterchef has been exploited on a significant number of projects. In all cases, their native tokens went to $0 because the exploit resulted in an egregiously large number of tokens being minted and dumped. This issue was also present in SushiSwap (the original Masterchef) and is present in pretty much every Masterchef since. Since they were never meant to have any tokens but LP tokens, it was not a problem there but has become a problem to projects who have started forking it for usage with less standard tokens.

Recommendation	Consider using the current standard of handling deposits, which is based on how Uniswap handles transfer fees:
	
	`uint256 balanceBefore = pool.lpToken.balanceOf(address(this));`
	`pool.lpToken.transferFrom(msg.sender, address(this), _amount);`
	`_amount = pool.lpToken.balanceOf(address(this)).sub(balanceBefore);`

Resolution	RESOLVED The recommendation has been implemented.

Issue #06	Token ownership can be transferred to mint and dump tokens
Severity	HIGH SEVERITY
Description	By calling the `transferBoujeTokenOwnership` function, the owner of this contract can transfer token ownership to any address, which would then be able to mint tokens and dump these on the market. Note that transferring token ownership would result in `updatePool` failing to mint tokens, and `deposit` and `withdraw` will therefore revert as well.
Recommendation	Although the comments state that this will only ever be used if the Masterchef is to be upgraded, it nonetheless poses a very big risk to users of the protocol. We highly recommend that this function be removed for ultimate peace of mind. Nonetheless, should this function be desired, then possibly introducing a 3-7 day delay before the function can be executed would give users plenty of time to react accordingly.
Resolution	RESOLVED `transferBoujeTokenOwnership` has been removed from the contract.

Issue #07	Unnecessary and inconsistent devReward logic
Severity	MEDIUM SEVERITY

Description	Within the `updatePool` function, 10% of the normal emission rate is minted extra to the `devAddress`. However, the `deposit` function still subtracts it from the rewards to the harvester but this is not necessary as the 10% is minted on top of these rewards and not as a part of them.
	
	Lines 1600-1601
	`uint256 devReward = pending.div(10);`
	`safeBoujeTransfer(msg.sender, pending.sub(devReward));`
	
	Furthermore, within the `withdraw` function, a different harvest is done, presumably by accidental error. Within the `withdraw` function, this `devReward` is not subtracted and the variable is unused.
	
	Lines 1641-1642
	`uint256 devReward = pending.div(10);`
	`safeBoujeTransfer(_user, pending);`

Recommendation	Consider adding an `lpSupply` variable to the `PoolInfo` that keeps track of the total deposits.
Resolution	RESOLVED Rewards transferred on deposit no longer subtracts from `pending`. Harvest on `withdraw` is fixed as well, and `lpSupply` has been added in `PoolInfo`.

Issue #08	**Setting `devAddress` or `feeAddress` to the zero address will break most functionality**
Severity	MEDIUM SEVERITY
Description	Within token contracts, minting or transferring tokens to the zero address will revert the transaction. Deposits and withdrawals will break if the `feeAddress` or `devAddress` is ever set to the zero address. Harvesting would fail as well. It should be noted that the `feeAddress` only affects deposits while the `devAddress` affects both deposits and withdrawals.

Recommendation	To prevent this from ever happening by accident and to limit governance risks, consider adding a requirement like so:
	`require(_devAddress != address(0), “non-zero!”);`
	`require(_feeAddress != address(0), “non-zero!”);`
	to the relevant configuration functions.

Resolution	RESOLVED The recommendation has been implemented.

Issue #10	Usage of block numbers on the Fantom network could cause irregular reward emissions
Severity	LOW SEVERITY
Description	Since block rates are very inconsistent on the Fantom network, using block numbers to account for time is not considered a good practice and could lead to very volatile emission rates.
Recommendation	Consider using `block.timestamp` everywhere instead of block numbers. It is very important that all places where block numbers were used to account for time are replaced with timestamps as using both by accident would be very bad.
Resolution	RESOLVED

Issue #11	The `pendingBouje` function will revert if `totalAllocPoint` is zero
Severity	LOW SEVERITY
Description	In the `pendingBouje` function, at some point a division is made by the `totalAllocPoint` variable. If all pools have their rewards set to zero, this variable will be zero as well. The requests will then revert with a division by zero error.
Recommendation	Consider only calculating the accumulated rewards since the `lastRewardBlock` if the `totalAllocPoint` variable is greater than zero. This check can simply be added to the existing check that verifies the `block.number` and `lpSupply`, like so: `if (block.timestamp > pool.lastRewardTime && pool.lpSupply != 0 && totalAllocPoint > 0) {`
Resolution	RESOLVED The recommendation was implemented.

Issue #12	Minting can exceed max token supply
Severity	LOW SEVERITY

Description	Currently, the hard-cap of 88,888 tokens is enforced as follows:
	
	Lines 1568-1571
	`if (currentSupply >= MAX_BOUJE_SUPPLY) {`
	`  pool.lastRewardBlock = block.number;`
	`  return;`
	`}`
	
	However, this only checks that the limit is not reached before the mint. This could mean that for example if the current supply is 88,887 tokens and there is a request to mint 2 tokens, this request will still pass since the supply is not reached and the final supply will be 88,889 tokens.
	
	❗ The `pendingBouje` function furthermore does not incorporate the aforementioned maximum. It might therefore disclose a wrong value compared to the actual pending tokens. This would however only affect the frontend.

Recommendation	Consider not minting the tokens in case the supply exceeds the total supply, like so:
	
	`if (boujeToken.totalSupply().add(boujeToken.mul(11).div(10)) <= MAX_BOUJE_SUPPLY) {`
	`  // The whole emission can be mint`
	`  uint256 devReward = boujeReward.div(10);`
	`  boujeToken.mint(devAddress, devReward);`
	`  boujeToken.mint(address(this), boujeReward);`
	
	`} else if (boujeToken.totalSupply() < MAX_BOUJE_SUPPLY) {`
	`  boujeToken.mint(address(this), MAX_BOUJE_SUPPLY.sub(boujeToken.totalSupply()));`
	`}`
	
	This will only ever mint the total supply at most.

Resolution	RESOLVED The `pendingVive` function was fixed as recommended to show the correct value at frontend. `updatePool` has been updated as well but with a different approach.

Issue #13	**updateEmissionRate has no maximum safeguard**
Severity	LOW SEVERITY
Description	Projects sometimes accidentally update their emission rate to a severely high number either by accident or with malicious intent.
Recommendation	Consider adding a `MAX_EMISSION_RATE` variable and setting it to a reasonable value. `require(_boujeTokenPerBlock <= MAX_EMISSION_RATE,"Too high");`
Resolution	RESOLVED There is now a maximum safeguard of 10 tokens.

Issue #14	**boujeToken can be made immutable**
Severity	INFORMATIONAL
Description	Variables that are only set in the constructor but never modified can be indicated as such with the immutable keyword. This is considered best practice since it makes the code more accessible for third-party reviewers and saves gas.
Recommendation	Consider making `boujeToken` explicitly immutable.
Resolution	RESOLVED

Issue #15	Pools use the contract balance to figure out the total deposits
Severity	INFORMATIONAL
Description	As with pretty much all Masterchefs, the total number of tokens in the Masterchef contract is used to determine the total number of deposits. This can cause dilution of rewards when people accidentally send tokens to the masterchef. More severely, because the native token is constantly minted, this will cause severe dilution on the native token pool.
Recommendation	Consider adding an `lpSupply` variable to the `PoolInfo` that keeps track of the total deposits.
Resolution	RESOLVED `lpSupply` has been added to `PoolInfo` as recommended.

Issue #16	References to FISH token
Severity	INFORMATIONAL
Description	Throughout the contract, within the code comments there are references to a FISH token. As there is no such token within this project, we assume this is due to accidental copying.
Recommendation	Consider going through the code comments to remove the obsolete references.
Resolution	RESOLVED References to FISH have been removed.

Issue #17	**`deposit`, `withdraw` and `emergencyWithdraw` can be made `external`**
Severity	INFORMATIONAL
Description	The above functions can be changed from `public` to `external`. Apart from being a best practice when the function is not used within the contract, this can lead to a lower gas usage in certain cases.
Recommendation	Consider making the above functions `external`.
Resolution	PARTIALLY RESOLVED `deposit` and `withdraw` has been made external. `emergencyWithdraw` remains public.

Issue #18	**Lack of events for `add`, `set`, `updateStartBlock`, `switchActivePool`, `forceWithdraw` and `transferBoujeTokenOwnership`**
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for the above functions.
Resolution	PARTIALLY RESOLVED `switchActivePool` and `forceWithdraw` still have no events.


PALADIN
BLOCKCHAIN SECURITY