STAYSAFU
AUDIT

JANUARY 12TH, 2023

BOMB Money

TABLE OF CONTENTS

AUDIT SUMMARY

This report was written for BOMB Money in order to find flaws and vulnerabilities in the BOMB Money project's source code, as well as any contract dependencies that weren't part of an officially recognized library.

A comprehensive examination has been performed, utilizing Static Analysis, Manual Review, and BOMB Money Deployment techniques. The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors
- Assessing the codebase to ensure compliance with current best practices and industry standards
- Ensuring contract logic meets the specifications and intentions of the client
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders
- Through line-by-line manual review of the entire codebase by industry expert

AUDIT OVERVIEW

PROJECT SUMMARY

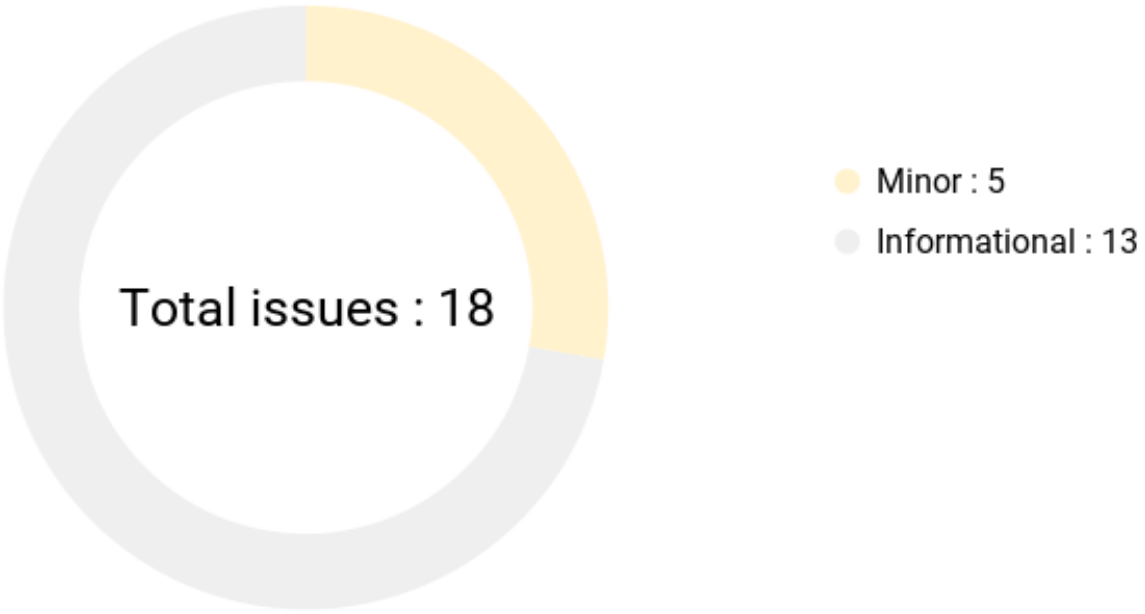
Project name	BOMB Money
Description	BOMB Money is on a mission to become the largest & safest crypto staking platform in the world. We already have an established DeFi ecosystem and are preparing for the launch of our mobile app that will break down the barriers to DeFi.
Platform	…
Language	Solidity
Codebase	https://github.com/bombchain/bluechip-staking-contracts/

FINDINGS SUMMARY

Vulnerability	Total
Critical	0
Major	0
Medium	0
Minor	5
Informational	13

EXECUTIVE SUMMARY

There have been no major or critical issues related to the codebase and all findings listed here are minor or informational. The major issues that have been found are centralization of major privileges and dependance on external protocols.



AUDIT FINDINGS

Code	Title	Severity
CON-1	Constructor	Informational
CLAIM-1	claimAndCompoundMulti	Informational
GET-1	getTotalEarnedAmount	Informational
GET-2	getTotalValueAtMaturity	Informational
GET-3	getTotalYieldAtMaturity	Informational
GOV-1	governanceRecoverUnsupported	Informational
DEP-1	deployStake	Informational
DEPO-1	_deposit	Informational
WITD-1	_withdraw	Informational

ASSET-1	_registerAsset	Informational
DEP-2	deployFunds	Informational
RET-1	returnDeployedFunds	Informational
APPR-1	_approveTokenIfNeeded	Informational
STAKE-1	stake, _stake	Minor
WITHD-1	withdraw	Minor
WITHD-2	withdrawMulti	Minor
CLAIM-1	claimAndCompound	Minor
GOV-2	governanceRecoverUnsupported	Minor

Contract | Operator.sol

The contract itself is like an Owner.sol which actually inherits OpenZeppelin's Ownable.sol. The contract establishes an additional role, known as the "operator,", and It can be changed by the owner. The contract adds new modifiers;

1. onlyOperator
2. onlyOwnerOrOperator

With new functions;

1. operator()
2. isOperator()
3. transferOperator(address newOperator_) (only owner can use.)

Also, this contract adds a new event called "OperatorTransferred" which gets triggered in the function `transferOperator`. The contract is coded in a very secure way, and the events are using the "indexed" keyword to help filter events.

Contract | StakingPositions.sol

The contract did not show any major vulnerabilities during the audit, and it was coded according to good practice with the correct variable types, such as "uint16" for the "PERCENT_DENOMENATOR" variable.
Almost every action is emitted as an event which is good practice.

CON-1 | constructor

Description

Constructs the contract and sets the vault's owner as operator. There are 7 parameters, I would recommend using a struct that has those parameters for cleaner code.

STAKE-1 | stake, _stake

Description

You can set the amounts of tokens you want to stake with "uint256 _amountStaked". If the "_amountStaked" is "0", the contract sets it to your token balance.

If the "_transferStakeToken" variable is set; the staked tokens are transferred to the vault contract.

After that, the contract mints the sender an NFT token that holds the staking information of the sender. Then an event called "CreateStake" gets emitted.

Some of the variables like "totalYieldAtMaturity" and "_yieldAtMaturity" are calculated without an overflow check. SafeMath for variables like this is recommended.

There are 6 parameters, I would recommend using a struct for cleaner code.

The "msg.sender" can set variables like "_transferStakeToken" and "_allowWithdrawEarly" that can change the staking process. There are no checks for those variables. The "_fromCompound" variable is unnecessary and only used to emit an event.

WITHD-1 | withdraw

Description

The "_user" variable is set by "msg.sender", that means if the transaction is coming from a relayer like a proxy contract, the "_user" variable will be equal to the proxy contract's address instead of the real sender. I would recommend using "_msgSender" from OpenZeppelin.

There are no checks for "_isEarlyWithdraw". The sender can easily change the parameter and bypass the "Must acknowledge the early withdraw due to loss of tokens" error.

There are no checks for "allowWithdrawEarly" variable when the "Stake" struct gets created in the function "_stake". The sender can easily bypass the "This position is not eligible for early withdraw" error when opening a staking position.

With those variables set, the sender can withdraw half of his staked tokens. I recommend checking or setting the variables in the contract. Any stake position can be unstaked with this.

"UnstakeTokens" event gets emitted after the process, and the token that holds the staking info gets burned with "_burn".

WITHD-2 | withdrawMulti

Description

Same as “withdraw”, “isWithdrawEarly” parameter is also not checked. “_tokenIds” are an array but “isWithdrawEarly” is not. There could be two staking positions with one withdrawn early and one withdrawn after the stake time has passed but since “isWithdrawEarly” is not an array they would have the same outcome no matter if one of them is not withdrawn earlier.

CLAIM-1 | claimAndCompound

Description

There are no checks for the sender to be equal to the "_user". Someone that can predict the stake position's owner can call this function.

CLAIM-2 | claimAndCompoundMulti

Description

It's a helper function to multicall "claimAndCompound" and like the "withdrawMulti", not all parameters are arrays.

GET-1 | getTotalEarnedAmount

GET-2 | getTotalYieldAtMaturity

GET-3 | getTotalValueAtMaturity

Description

The math on those functions has no overflow checks. Calculating variables with libraries like "SafeMath" are recommended.

GOV-1 | governanceRecoverUnsupported

Description

The owner or the vault can access to contract's token balances. They can withdraw any token anywhere they want.

Contract | StakingVault.sol

This is the contract that holds the staked tokens. It's also "ReentrancyGuard" contract which is good point, and instead of normal `msg.sender` the contract uses `_msgSender` which allows proxies to send transactions too.

DEP-1 | deployStake

Description

The parameters are too long. Using a struct for parameters is recommended for cleaner code.

The variables "_endTime", "_capacity" and "_stakeToken" should be checked for invalidness. For example; "_endTime" should be more than "block.number" and "_stakeToken" shouldn't be zero address or dead address.

This function deploys a new "StakingPosition" contract and transfers the owner.

DEPO-1 | _deposit

Description

"_stakeAsset.stakedAmount += _amount;" line should be overflow checked. SafeMath is recommended. "Deposit" event should be emitted after the tokens are transferred.

Underscores are for private variables or functions but "_deposit" function is external. It should be renamed as something else.

WITD-1 | _withdraw

Description

Just like "_deposit", the calculations should be checked for overflow and the "Withdraw" event should be emitted after the tokens are transferred.

ASSET-1 | _registerAsset

Description

It's just a helper function to register new assets. The only thing that might cause a problem is the "active" property of the "StakeAsset" struct. It's always true, that if an asset is created with this function It's gonna be forced to be active.

DEP-2 | deployFunds

Description

The event is emitted after the process, that is ok but the calculations still need an overflow check. SafeMath is recommended.

RET-1 | returnDeployedFunds

Description

“require(stakeAssets[_stakeId].created > 0, "Stake does not exist");” is used at the start of so many functions. Converting that to a modifier is recommended for cleaner code.

“stakePosition.deployedAmount -= _amount;” should be checked for overflow. SafeMath is recommended.

APPR-1 | _approveTokenIfNeeded

Description

This function has never been used anywhere and can be removed.

GOV-2 | governanceRecoverUnsupported

Description

Owner can access to vault contract's token balance. That means the owner can withdraw any token in the vault.

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement.

This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement.

This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without StaySAFU's prior written consent.This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts StaySAFU to perform a security assessment.

This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the

technologies proprietors, business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project.

This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk.

StaySAFU's position is that each company and individual are responsible for their own due diligence and continuous security. StaySAFU's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or fun.