#HashEx
BLOCKCHAIN SECURITY

Purplesale

smart contracts final audit report

April 2024

hashex.org

contact@hashex.org

Contents

1. Disclaimer 3

2. Overview 4

3. Project centralization risks 7

4. Found issues 8

5. Contracts 11

6. Conclusion 19

Appendix A. Issues' severity classification 20

Appendix B. Issue status description 21

Appendix C. List of examined issue types 22

Appendix D. Centralization risks classification 23

1. Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and HashEx and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (HashEx) owe no duty of care towards you or any other person, nor does HashEx make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and HashEx hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HashEx hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HashEx, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed. HashEx owns all copyright rights to the text, images, photographs, and other content provided in the following document. When using or sharing partly or in full, third parties must provide a direct link to the original document mentioning the author (hashex.org).

2. Overview

HashEx was commissioned by the PurpleSale team to perform an audit of their smart contracts. The audit was conducted between 08/04/2024 and 10/04/2024.

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts
- Formally check the logic behind given smart contracts.

Information in this report should be used for understanding the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

The code was provided directly in .sol files.

SHA-1 hashes of the audited files are:

Idatabase.sol `dab50cc1bec0466d0fb69ce19731a22b1b15b770`

IPancakeRouter.sol `70431d3132d17524fb198a9f72adb6de4f534194`

IPancakePair.sol `c6c84bb5f6acd9380fe4e6f5ea4680201d46553c`

IPancakeFactory.sol `ab14c38a99b110a33f4f088b9b3791d892da3423`

FairLaunch/ICO.sol `44185f81a37ff5ff03313f4e804b163cbdda2590`

FairLaunch/BasicStructs.sol `353edc062ba36b33b76ee0e1cabd20f4d4a9b1d5`

FairLaunch/Launch.sol `a0830de8adc54eddd06b59fb993f93118f1a41fb`

Presale/ICO.sol `7fccaaa44c2b2d4d37b3c76ec086819ae2d21796`

Presale/BasicStructs.sol `b512798c052e9605138092c68e8d864545551e59`

Presale/Launch.sol `0cd71fea912bcf155f5b34db82bfe59aa03a62b0`

Update. The PurpleSale team has responded to this report. SHA-1 hashes of the updated files are:

FairLaunch/ICO.sol `fc7a8460fb8fcb0a41b4c61f871455add857a7d3`

FairLaunch/Launch.sol `a5a9189e7270ed5f6fdfa5cd5f84ddf33cf2932e`

Presale/ICO.sol `dc35a9b67de9ff5473352d1e1ee58252f7c17371`

Presale/Launch.sol `7e12524a88bc01320fee3d345c3c61598499d62d`

Please note that no code verification for deployed contracts was made. Deployed code may not be the same as what was audited. To ensure that the deployed contracts are the same as those audited, users must independently verify the SHA-1 hashes of the verified contract code.

2.1 Summary

Project name	Purplesale
URL	https://purplesale.finance
Platform	Binance Smart Chain
Language	Solidity
Centralization level	High
Centralization risk	High

2.2 Contracts

Name	Address
FairLaunch ICO	
Presale ICO	
LaunchFairLaunch	
Presale Launch	
Multiple contracts	

3. Project centralization risks

The project is highly centralized and all sales may become malfunctioned even after successful deployment.

4. Found issues

Critical	1 (6%)
High	8 (47%)
Medium	4 (24%)
Info	4 (23%)

Cd2. FairLaunch ICO

ID	Severity	Title	Status
Cd2I54	High	Wrong calculations in the buyTokens() function	Resolved
Cd2I61	High	Arbitrary router address	Resolved
Cd2I56	High	No checks in the setVesting() function	Resolved
Cd2I58	Medium	Wrong totalWhitelisted calculation	Resolved
Cd2I53	Info	Wrong function name	Partially fixed
Cd2I55	Info	Front-run threat	Acknowledged
Cd2I57	Info	Whitelist functionality isn't working	Resolved

Cd3. Presale ICO

ID	Severity	Title	Status
Cd3I5b	Critical	No checks in the setVesting() function	Resolved
Cd3I62	High	Arbitrary router address	Resolved
Cd3I64	High	Simultaneous cancelling and finalization	Resolved
Cd3I5c	Medium	Wrong totalWhitelisted calculation	Resolved
Cd3I59	Medium	Function adminCancelPresale() isn't working	Resolved
Cd3I67	Medium	Minimal amount limit	Acknowledged
Cd3I5a	Info	Front-run threat	Acknowledged

Cd4. LaunchFairLaunch

ID	Severity	Title	Status
Cd4I65	High	Lack of input validation	Partially fixed

Cd5. Presale Launch

ID	Severity	Title	Status
Cd5I66	High	Lack of input validation	Partially fixed

Cd6. Multiple contracts

ID	Severity	Title	Status
Cd6I69	High	Lack of tests and documentation	Acknowledged

5. Contracts

Cd2. FairLaunch ICO

Overview

The contract allows users to purchase tokens using a native token for a user. The price for the token on a sale is determined dynamically at the end of the sale.

Any user can withdraw his position in the native token before the end of the sale (except in the case when the sale is canceled). The contract will transfer some percent of his position (in native tokens) to the admin wallet address.

A presale can be canceled by the owner of the presale or by the protocol admin. If this happens users will be able to refund their funds.

Only the owner of a presale can finalize it.

The contract doesn't have tests at all and cannot be considered completely verified and completely working. This contract may contain critical vulnerabilities since tests are necessary for complete confirmation that the contract is fully functioning.

Also, the contract doesn't have any checks on inputs coming from the owner of a presale. Any user who wants to use this contract should fully trust the owner of the presale because if the owner sets bad values for the contract, the contract may break and the user can lose his funds.

Issues

Cd2I54 Wrong calculations in the buyTokens() function

High Resolved

For example, there are 10 tokens for sale and the token has 18 decimals. Also, users sent 100 BNB to the contract to buy tokens. In this case, the global variable `ratePerBNB` will be equal to `(10 * 10**18 * 10**18) / (10**18 * 100 * 10**18) = 10 / 100 = 0`. Because of that the

sale will be broken.

Recommendation

There is no need for calculation of the `ratePerBNB` variable in the `buyTokens()` function. In the `Claim()`, `_normalClaim()`, `getUsersPurchase()`, and `remainingTokens()` functions amount of tokens can be calculated using this formula: `icoInfo.data.presaleSupply * bnbAmount / raisedBNB`.

Update

The issue remains for the `remainingTokens()` function, which is the only function using the `ratePerBNB` variable.

Cd2I61 Arbitrary router address

High Resolved

Any user can set any router address he wants. The owner of a presale can set a malicious router address and using that router steal all funds that will come to it.

Recommendation

Add in the factory contract a whitelist of routers that can be used in presales.

Cd2I56 No checks in the setVesting() function

High Resolved

In the function `setVesting()` there are no checks that this function is called only once and also there is no check that the `vesting.firstPercent` variable is less than 100%.

The owner of a presale can withdraw all tokens from the contract by setting very large `vesting.firstPercent` variable.

Recommendation

These checks can be added to fix this issue:

```
require(maxPercent == 0);
require(vesting.firstPercent <= 10000);
```

Cd2I58 Wrong totalWhitelisted calculation

Medium Resolved

In the `addToWhiteList()` and `removeWhiteList()` there are no checks that addresses are unique. If they are not unique then the variable `totalWhitelisted` will have a wrong value.

Cd2I53 Wrong function name

Info Partially fixed

Function `getLiquiditySupply()` doesn't return liquidity supply. It returns a different value.

Cd2I55 Front-run threat

Info Acknowledged

A call to the `endSale()` function can be front-run because there are no slippage cheks.

Cd2I57 Whitelist functionality isn't working

Info Resolved

In the contract there is a whitelist functionality that isn't used anywhere in the contract.

Cd3. Presale ICO

Overview

The contract allows whitelisted users to purchase tokens using a native token. The price for the token on a sale is static and determined at the deployment.

Any user can withdraw his position in the native token before the end of the sale (except in the case when the sale is canceled). The contract will transfer some percent of his position (in native tokens) to the admin wallet address.

A presale can be canceled by the owner of the presale or by the protocol admin. If this happens users will be able to refund their funds.

Only the owner of a presale can finalize it.

The contract doesn't have tests at all and cannot be considered completely verified and completely working. This contract may contain critical vulnerabilities since tests are necessary for complete confirmation that the contract is fully functioning.

Also, the contract doesn't have any checks on inputs coming from the owner of a presale. Any user who wants to use this contract should fully trust the owner of the presale because if the owner sets bad values for the contract, the contract may break and the user can lose his funds.

Issues

Cd3I5b No checks in the setVesting() function

Critical Resolved

In the function `setVesting()` there are no checks that this function is called only once and also there is no check that the `vesting.firstPercent` variable is less than 100%.

The owner of a presale can withdraw all tokens from the contract by setting very large `vesting.firstPercent` variable.

Recommendation

These checks can be added to fix this issue:

```
require(maxPercent == 0);
require(vesting.firstPercent <= 10000);
```

Update

The issue remains as vesting schedule still can be updated multiple times to exceed 100% in total. Moreover, introduced checks allow any user to abuse schedule re-setting.

Cd3I62 Arbitrary router address

High Resolved

Any user can set any router address he wants. The owner of a presale can set a malicious router address and using that router steal all funds that will come to it.

Recommendation

Add in the factory contract a whitelist of routers that can be used in presales.

Cd3I64 Simultaneous cancelling and finalization

High Resolved

The sale owner can cancel the sale, returning all ERC20 tokens, and then call the `endSale()` function to obtain part of collected BNB. In this scenario, users of the sale will have nothing in return.

Recommendation

Deny sale finalization if it has been cancelled.

Cd3I5c Wrong totalWhitelisted calculation

Medium Resolved

In the `addToWhiteList()` and `removeWhiteList()` there are no checks that addresses are unique. If they are not unique then the variable `totalWhitelisted` will have a wrong value.

Cd3I59 Function adminCancelPresale() isn't working

Medium Resolved

The function `adminCancelPresale()` isn't working because a transaction will revert due to underflow in the `remainingTokens()` function during the calculation of the `bal` variable.

All tokens are transferred from the contract in the `_cancelPresale()` function that is called before the `remainingTokens()` function. Because of that, this underflow will happen (balance is zero, `soldToken` is not zero).

Cd3I67 Minimal amount limit

Medium Acknowledged

Only the first purchase is checked against the `icoInfo.data.minAmount`, after that any other purchases can have zero value, and the corresponding users will be included into the contributors' address list.

Recommendation

Fix the logic or remove requirements.

Cd3I5a Front-run threat

Info Acknowledged

A call to the `endSale()` function can be front-run because there are no slippage cheks.

Cd4. LaunchFairLaunch

Overview

Using this contract a user can deploy his presale with any parameters he wants. He will need to transfer fees to a staking pool address and to an admin wallet address.

Issues

Cd4I65 Lack of input validation

High Partially fixed

The user-input data is not validated in any kind, allowing creation of malicious or vulnerable sales.

Recommendation

Include safety checks for the input data.

Cd5. Presale Launch

Overview

Using this contract a user can deploy his presale with any parameters he wants. He will need to transfer fees to a staking pool address and to an admin wallet address.

Issues

Cd5I66 Lack of input validation

High Partially fixed

The user-input data is not validated in any kind, allowing creation of malicious or vulnerable sales.

Recommendation

Include safety checks for the input data.

Cd6. Multiple contracts

Issues

Cd6I69 Lack of tests and documentation

High Acknowledged

The project doesn't contain any tests and documentation. We urgently recommend increasing test coverage. We also suggest providing the documentation section.

6. Conclusion

1 critical, 8 high, 4 medium severity issues were found during the audit. 1 critical, 5 high, 3 medium issues were resolved in the update. The reviewed contracts are highly dependent on the owner's account. See the centralization risks chapter.

This audit includes recommendations on code improvement and the prevention of potential attacks.

Please note that no code verification for deployed contracts was made. Deployed code may not be the same as what was audited. To ensure that the deployed contracts are the same as those audited, users must independently verify the SHA-1 hashes of the verified contract code.

Appendix A. Issues' severity classification

- **Critical.** Issues that may cause an unlimited loss of funds or entirely break the contract workflow. Malicious code (including malicious modification of libraries) is also treated as a critical severity issue. These issues must be fixed before deployments or fixed in already running projects as soon as possible.
- **High.** Issues that may lead to a limited loss of funds, break interaction with users, or other contracts under specific conditions. Also, issues in a smart contract, that allow a privileged account the ability to steal or block other users' funds.
- **Medium.** Issues that do not lead to a loss of funds directly, but break the contract logic. May lead to failures in contracts operation.
- **Low.** Issues that are of a non-optimal code character, for instance, gas optimization tips, unused variables, errors in messages.
- **Informational.** Issues that do not impact the contract operation. Usually, informational severity issues are related to code best practices, e.g. style guide.

Appendix B. Issue status description

Resolved. The issue has been completely fixed.

Partially fixed. Parts of the issue have been fixed but the issue is not completely resolved.

Acknowledged. The team has been notified of the issue, no action has been taken.

Open. The issue remains unresolved.

Appendix C. List of examined issue types

- Business logic overview
- Functionality checks
- Following best practices
- Access control and authorization
- Reentrancy attacks
- Front-run attacks
- DoS with (unexpected) revert
- DoS with block gas limit
- Transaction-ordering dependence
- ERC/BEP and other standards violation
- Unchecked math
- Implicit visibility levels
- Excessive gas usage
- Timestamp dependence
- Forcibly sending ether to a contract
- Weak sources of randomness
- Shadowing state variables
- Usage of deprecated code

Appendix D. Centralization risks classification

Centralization level

- **High.** The project owners can manipulate user's funds, lock user's funds on their will (reversible or irreversible), or maliciously update contracts parameters or bytecode.
- **Medium.** The project owners can modify contract's parameters to break some functions of the project contract or contracts, but user's funds remain withdrawable.
- **Low.** The contract is trustless or its governance functions are safe against a malicious owner.

Centralization risk

- **High.** Lost ownership over the project contract or contracts may result in user's losses. Contract's ownership belongs to EOA or EOAs, and their security model is unknown or out of scope.
- **Medium.** Contract's ownership is transferred to a contract with not industry-accepted parameters, or to a contract without an audit. Also includes EOA with a documented security model, which is out of scope.
- **Low.** Contract's ownership is transferred to a well-known or audited contract with industry-accepted parameters.

 contact@hashex.org

 @hashex_manager

 blog.hashex.org

 linkedin

 github

 twitter

#HashEx
BLOCKCHAIN SECURITY