#HashEx
Blockchain Security

PayFlow

smart contracts final audit report

January 2022


hashex.org


contact@hashex.org

Contents

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 PayFlow team to perform an audit of their smart contract. The audit was conducted between 19/01/2022 and 20/01/2022.

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 is available at address 0xe3B42852a85d38b18076Ab2dd96B0F894CC0636c.

2.1 Summary

Project name	PayFlow
URL	https://bscscan.com/address/0xe3b42852a85d38b18076ab2dd96b0f894cc0636c
Platform	Binance Smart Chain
Language	Solidity

2.2 Contracts

Name	Address
SafeMath	

CodeWithJoe	0xe3B42852a85d38b18076Ab2dd96B0F894CC0636c

3. Found issues

Medium	1 (25%)
Low	2 (50%)
Info	1 (25%)

C1. SafeMath

ID	Severity	Title	Status
C1-01	Low	Lack of error-messages	Acknowledged
C1-02	Info	Code formatting issue	Acknowledged

C2. CodeWithJoe

ID	Severity	Title	Status
C2-01	Medium	ERC20 approval race condition	Acknowledged
C2-02	Low	Unclear behaviour	Acknowledged

4. Contracts

C1. SafeMath

Overview

Library to perform arithmetic operations protecting from over-/underflow issues.

Issues

C1-01 Lack of error-messages

Low Acknowledged

Error messages are missing in the require-statements of the library.

C1-02 Code formatting issue

Info Acknowledged

At L32 a few functions are implemented within the line, which makes it hard to read the logic of the functions.

C2. CodeWithJoe

Overview

Standard ERC20-Token.

Issues

C2-01 ERC20 approval race condition

Medium Acknowledged

The token is not protected from the known ERC20 approval race condition attack.

The attack is possible because the `approve` method overwrites the current allowance

regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, rather than an account.

Recommendation

The risk of the attack could be mitigated by implementing functions increaseAllowance and decreaseAllowance.

C2-02 Unclear behaviour

Low Acknowledged

The view-function `totalSupply()` returns the result of the expression

```
_totalSupply  - balances[address(0)];
```

There is no obvious need to calculate the result of `totalSupply` instead of returning a value of `_totalSupply`. In case such behavior is still needed, it should be explained in a comment.

5. Conclusion

The contract supports the `ERC20` standard. High Severity issues are not found.

At the moment of the audit, most of the tokens (about 90%) are locked on a Vesting contract located at address 0x0C89C0407775dd89b12918B9c0aa42Bf96518820. While just ~0.33% of ones are in a liquidity pool at 0x0EbCD2E67027936f2A878DF1529bE65640d36ce3.

This audit includes recommendations on the code improving and preventing potential attacks.

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. 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

 contact@hashex.org

 @hashexbot

 blog.hashex.org

 linkedin

 github

 twitter

HashEx
Blockchain Security