HashEx
BLOCKCHAIN SECURITY

PlayLeap

smart contracts preliminary audit report for internal use only

December 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 LEAP team to perform an audit of their smart contracts. The audit was conducted between 2022-12-24 and 2022-12-26

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 0x53263d9EF74Db583b15fbC6D5D4e8B83833fa134 in the Ethereum mainnet and 0x6eEd9140F80F9E989CB23AeCBD20b97a29FFc80F in the Binance Smart Chain (BSC)

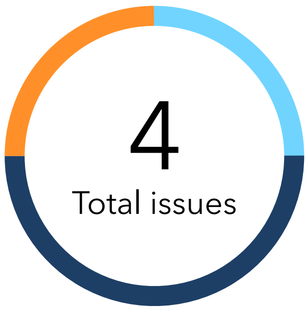
2.1 Summary

Project name	PlayLeap
URL	https://playleap.io
Platform	Binance Smart Chain, Ethereum
Language	Solidity

2.2 Contracts

Name	Address
LEAPToken	0x53263d9EF74Db583b15fbC6D5D4e8B83833fa134
MultiBridgeToken	0x6eEd9140F80F9E989CB23AeCBD20b97a29FFc80F

3. Found issues



●	High	1 (25%)
●	Low	1 (25%)
●	Info	2 (50%)

C1. LEAPToken

ID	Severity	Title	Status
C1-01	Info	Excessive inheritance	Open

C2. MultiBridgeToken

ID	Severity	Title	Status
C2-01	High	Mint is open for owner	Open
C2-02	Low	Gas optimizations	Open
C2-03	Info	Not indexed parameters in events	Open

4. Contracts

C1. LEAPToken

Overview

An implementation of the ERC-20 token standard built on OpenZeppelin contracts. Supports EIP-2612 permit approvals and voting, see ERC20Permit and ERC20Votes extensions by OpenZeppelin.

Issues

C1-01 Excessive inheritance

● Info ⓘ Open

LEAPToken contract inherits the ERC20, ERC20Permit, and ERC20Votes contracts, but ERC20 is already inherited from ERC20Permit, which in turn is inherited from ERC20Votes. Thus, LEAPToken could be compiled with a single parent of ERC20Votes, and no overrides would be needed in that case.

```
contract LEAPToken is ERC20, ERC20Permit, ERC20Votes {
  ...
}

abstract contract ERC20Votes is IVotes, ERC20Permit {
  ...
}

abstract contract ERC20Permit is ERC20, IERC20Permit, EIP712 {
  ...
}
```

C2. MultiBridgeToken

Overview

An ERC-20 standard token meant to be minted by bridge contracts. The contract owner has the ability to set mint capacity for desired bridges. No total supply cap is implemented, meaning the bridged token could be over-minted the original one in the Ethereum network.

Issues

C2-01 Mint is open for owner

High | Open

The contract owner can set the minting cap for an arbitrary address to any value of uint256 type. If the owner account gets compromised, the whole token project may fail, and the bridges may sustain losses.

```
function mint(address _to, uint256 _amount) external returns (bool) {
    Supply storage b = bridges[msg.sender];
    require(b.cap > 0, "invalid caller");
    b.total += _amount;
    require(b.total <= b.cap, "exceeds bridge supply cap");
    _mint(_to, _amount);
    return true;
}

function updateBridgeSupplyCap(address _bridge, uint256 _cap) external onlyOwner {
    bridges[_bridge].cap = _cap;
    emit BridgeSupplyCapUpdated(_bridge, _cap);
}
```

Recommendation

Limit the total supply according to the source token in the Ethereum network.

Secure the contract owner by transferring ownership to a Timelock contract with MultiSig admin.

C2-02 Gas optimizations

Low | Open

The `bridges[msg.sender].total` variable is read from the storage multiple times in the `mint()` and `_burnFrom()` functions.

C2-03 Not indexed parameters in events

Info | Open

Indexing of the parameters makes it possible to filter events. We recommend making the `bridge` parameter in the `BridgeSupplyCapUpdated` event indexed.

```
event BridgeSupplyCapUpdated(address bridge, uint256 supplyCap);
```

5. Conclusion

1 high, 1 low severity issues were found during the audit. No issues were resolved in the update.

The bridged LEAP token in BSC is highly dependent on the owner’s account. Users using the project have to trust the owner and that the owner's account is properly secured.

This audit includes recommendations on code improvement and the prevention of 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

@hashex_manager

blog.hashex.org

linkedin

github

twitter

HashEx

BLOCKCHAIN SECURITY