LatteVault

Smart Contract Audit Report
Prepared for LatteSwap

Date Issued: Oct 26, 2021
Project ID: AUDIT2021034
Version: v1.0
Confidentiality Level: Public

inspeX
CYBERSECURITY PROFESSIONAL SERVICE

Report Information

Project ID	AUDIT2021034
Version	v1.0
Client	LatteSwap
Project	LatteVault
Auditor(s)	Peeraphut Punsuwan
Author	Peeraphut Punsuwan
Reviewer	Weerawat Pawanawiwat
Confidentiality Level	Public

Version History

Version	Date	Description	Author(s)
1.0	Oct 21, 2021	Full report	Peeraphut Punsuwan

Contact Information

Company	Inspex
Phone	(+66) 90 888 7186
Telegram	t.me/inspexco
Email	audit@inspex.co

Table of Contents

1. Executive Summary

As requested by LatteSwap, Inspex team conducted an audit to verify the security posture of the LatteVault smart contracts between Oct 19, 2021 and Oct 20, 2021. During the audit, Inspex team examined all smart contracts and the overall operation within the scope to understand the overview of LatteVault smart contracts. Static code analysis, dynamic analysis, and manual review were done in conjunction to identify smart contract vulnerabilities together with technical & business logic flaws that may be exposed to the potential risk of the platform and the ecosystem. Practical recommendations are provided according to each vulnerability found and should be followed to remediate the issue.

1.1. Audit Result

In the initial audit, Inspex found 1 high, 1 medium, and 1 very low-severity issues. With the project team's prompt response in resolving the issues found by Inspex, all issues were resolved or mitigated in the reassessment. Therefore, Inspex trusts that the LatteVault smart contract has high-level protections in place to be safe from most attacks.

1.2. Disclaimer

This security audit is not produced to supplant any other type of assessment and does not guarantee the discovery of all security vulnerabilities within the scope of the assessment. However, we warrant that this audit is conducted with goodwill, professional approach, and competence. Since an assessment from one single party cannot be confirmed to cover all possible issues within the smart contract(s), Inspex suggests conducting multiple independent assessments to minimize the risks. Lastly, nothing contained in this audit report should be considered as investment advice.

2. Project Overview

2.1. Project Introduction

LatteSwap is a decentralized exchange with integrated NFT functionalities operating on the Binance Smart Chain (BSC). It is a one-stop-shop for traders, yield farmers, and NFT collectors across the Blockchain ecosystem.

LatteVault is implemented for the users to earn yields on LatteVault by depositing their $LATTE to get rewards. The LatteVault will deposit users' $LATTE into the MasterBarista contract and compound the farming reward for the users for higher yield.

Scope Information:

Project Name	LatteVault
Website	https://app.latteswap.com/
Smart Contract Type	Ethereum Smart Contract
Chain	Binance Smart Chain
Programming Language	Solidity

Audit Information:

Audit Method	Whitebox
Audit Date	Oct 19, 2021 - Oct 20, 2021
Reassessment Date	Oct 21, 2021

The audit method can be categorized into two types depending on the assessment targets provided:

1. **Whitebox**: The complete source code of the smart contracts are provided for the assessment.
2. **Blackbox**: Only the bytecodes of the smart contracts are provided for the assessment.

2.2. Scope

The following smart contract was audited and reassessed by Inspex in detail:

Initial Audit: (Commit: 50a7f03e31f83c1cfc114f86473b131bb42e46bb)

Contract	Location (URL)
LatteVault	https://github.com/latteswap-official/latteswap-contract/blob/50a7f03e31/contracts/farm/LatteVault.sol

Reassessment: (Commit: cd1a218c4342b146bcde429bcfaa46ce5c376286)

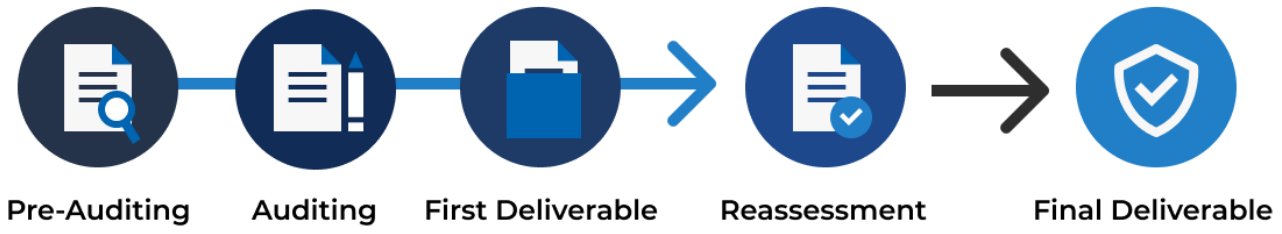
Contract	Location (URL)
LatteVault	https://github.com/latteswap-official/latteswap-contract/blob/cd1a218c43/contracts/farm/LatteVault.sol

The assessment scope covers only the in-scope smart contract and the smart contracts that it inherits from.

3. Methodology

Inspex conducts the following procedure to enhance the security level of our clients' smart contracts:

1. **Pre-Auditing**: Getting to understand the overall operations of the related smart contracts, checking for readiness, and preparing for the auditing
2. **Auditing**: Inspecting the smart contracts using automated analysis tools and manual analysis by a team of professionals
3. **First Deliverable and Consulting**: Delivering a preliminary report on the findings with suggestions on how to remediate those issues and providing consultation
4. **Reassessment**: Verifying the status of the issues and whether there are any other complications in the fixes applied
5. **Final Deliverable**: Providing a full report with the detailed status of each issue



3.1. Test Categories

Inspex smart contract auditing methodology consists of both automated testing with scanning tools and manual testing by experienced testers. We have categorized the tests into 3 categories as follows:

1. **General Smart Contract Vulnerability (General)** - Smart contracts are analyzed automatically using static code analysis tools for general smart contract coding bugs, which are then verified manually to remove all false positives generated.
2. **Advanced Smart Contract Vulnerability (Advanced)** - The workflow, logic, and the actual behavior of the smart contracts are manually analyzed in-depth to determine any flaws that can cause technical or business damage to the smart contracts or the users of the smart contracts.
3. **Smart Contract Best Practice (Best Practice)** - The code of smart contracts is then analyzed from the development perspective, providing suggestions to improve the overall code quality using standardized best practices.

3.2. Audit Items

The following audit items were checked during the auditing activity.

General
Reentrancy Attack
Integer Overflows and Underflows
Unchecked Return Values for Low-Level Calls
Bad Randomness
Transaction Ordering Dependence
Time Manipulation
Short Address Attack
Outdated Compiler Version
Use of Known Vulnerable Component
Deprecated Solidity Features
Use of Deprecated Component
Loop with High Gas Consumption
Unauthorized Self-destruct
Redundant Fallback Function
Insufficient Logging for Privileged Functions
Invoking of Unreliable Smart Contract
Use of Upgradable Contract Design
Advanced
Business Logic Flaw
Ownership Takeover
Broken Access Control
Broken Authentication
Improper Kill-Switch Mechanism

Improper Front-end Integration
Insecure Smart Contract Initiation
Denial of Service
Improper Oracle Usage
Memory Corruption
Best Practice
Use of Variadic Byte Array
Implicit Compiler Version
Implicit Visibility Level
Implicit Type Inference
Function Declaration Inconsistency
Token API Violation
Best Practices Violation

3.3. Risk Rating

OWASP Risk Rating Methodology[1] is used to determine the severity of each issue with the following criteria:

- **Likelihood**: a measure of how likely this vulnerability is to be uncovered and exploited by an attacker.
- **Impact**: a measure of the damage caused by a successful attack

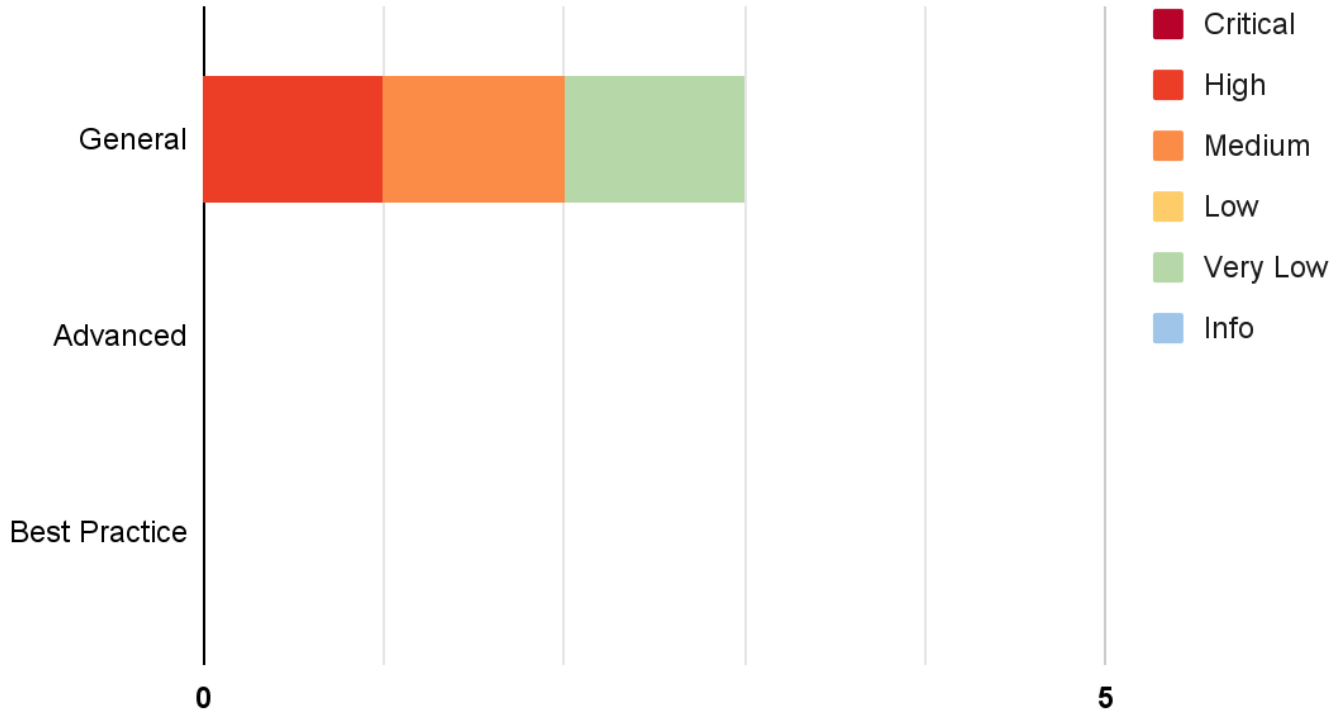
Both likelihood and impact can be categorized into three levels: **Low**, **Medium**, and **High**.

Severity is the overall risk of the issue. It can be categorized into five levels: **Very Low**, **Low**, **Medium**, **High**, and **Critical**. It is calculated from the combination of likelihood and impact factors using the matrix below. The severity of findings with no likelihood or impact would be categorized as **Info**.

Impact \ Likelihood	Low	Medium	High
Low	**Very Low**	**Low**	**Medium**
Medium	**Low**	**Medium**	**High**
High	**Medium**	**High**	**Critical**

4. Summary of Findings

From the assessments, Inspex has found 3 issues in three categories. The following chart shows the number of the issues categorized into three categories: **General**, **Advanced**, and **Best Practice**.



The statuses of the issues are defined as follows:

Status	Description
Resolved	The issue has been resolved and has no further complications.
Resolved *	The issue has been resolved with mitigations and clarifications. For the clarification or mitigation detail, please refer to Chapter 5.
Acknowledged	The issue's risk has been acknowledged and accepted.
No Security Impact	The best practice recommendation has been acknowledged.

The information and status of each issue can be found in the following table:

ID	Title	Category	Severity	Status
IDX-001	Use of Upgradable Contract Design	General	High	Resolved *
IDX-002	Centralized Control of State Variable	General	Medium	Resolved *
IDX-003	Insufficient Logging for Privileged Functions	General	Very Low	Resolved

* The mitigations or clarifications by LatteSwap can be found in Chapter 5.

5. Detailed Findings Information

5.1. Use of Upgradable Contract Design

ID	IDX-001
Target	LatteVault
Category	General Smart Contract Vulnerability
CWE	CWE-284: Improper Access Control
Risk	**Severity:** High **Impact:** High The logic of the affected contract can be arbitrarily changed. This allows the proxy owner to perform malicious actions e.g., stealing the user funds anytime they want. **Likelihood:** Medium This action can be performed by the proxy owner without any restriction.
Status	Resolved * The LatteSwap team has mitigated this issue by implementing a 24-hour delay `Timelock` over the admin of the `LatteVault` proxy contract. The admin of the `LatteVault` proxy contract is the `ProxyAdmin` contract, which is then owned by a timelock. The addresses of the related contracts are as follows: - `LatteVault` Proxy Address: `0x0201740f48158B43f72c30a39C122925008E1EE5` - `LatteVault` Implementation Address: `0xd839c004f5167a7ed1a7cd135c18fbf718589935` - `ProxyAdmin` Address : `0x02AF4337792a44aFb4005d57c36f9C3Bea6209bb` - `ProxyAdmin` Owner Address (`Timelock`) : `0x813879B5556B73c02A139e0340A33239C047957D` The platform users should monitor the timelock for the execution of privileged actions such as contract upgrading and act accordingly.

5.1.1. Description

Smart contracts are designed to be used as agreements that cannot be changed forever. When a smart contract is upgraded, the agreement can be changed from what was previously agreed upon.

As the `LatteVault` smart contract is upgradable, the contract logic can be modified by the owner anytime, making the smart contract untrustworthy.

5.1.2. Remediation

Inspex suggests deploying the contract without the proxy pattern or any solution that can make the smart contract upgradable.

However, if the upgradability is needed, Inspex suggests mitigating this issue by implementing a timelock mechanism with a sufficient length of time to delay the changes. This allows the platform users to monitor the timelock and be notified of the potential changes being done on the smart contract.

5.2. Centralized Control of State Variable

ID	IDX-002
Target	LatteVault
Category	General Smart Contract Vulnerability
CWE	CWE-710: Improper Adherence to Coding Standard
Risk	**Severity:** Medium **Impact:** Medium The controlling authorities can change the state variables without letting the users aware of the changes that may affect their funds. Thus, it is unfair to the other users. **Likelihood:** Medium There is nothing to restrict the changes from being done; however, these actions can only be performed by the contract owner.
Status	Resolved * The LatteSwap team has deployed the `LatteVault` to the BSC mainnet through a proxy contract. The addresses of the related contracts are as follows: - `LatteVault` Proxy Address: `0x0201740f48158B43f72c30a39C122925008E1EE5` - `LatteVault` Implementation Addresss: `0xd839c004f5167a7ed1a7cd135c18fbf718589935` - `LatteVault` Owner Address (LatteSwapDeployer): `0xE626fC6D9f4F1FAA17a157FB854d27fC55327283` At the time of the reassessment, the LatteSwap team has not transferred the contract ownership of the `LatteVault` to the `Timelock` contract yet. However, the LatteSwap team has confirmed that they will transfer the ownership to the `Timelock` contract with a 24-hour delay. So, the user should verify that the LatteSwap team has transferred the ownership to the `Timelock` contract before using it.

5.2.1. Description

The state variables can be updated any time by the controlling authorities. Changes in these variables can cause impacts to the users, so the users should accept or be notified before these changes are effective.

However, there is no constraint to prevent the authorities from modifying these variables without notifying the users.

The controllable privileged state update functions are as follows:

File	Contract	Function	Modifier
LatteVault (L: 197)	onlyOwner	setTreasury()	onlyOwner
LatteVault (L: 206)	onlyOwner	setPerformanceFee()	onlyOwner
LatteVault (L: 218)	onlyOwner	setWithdrawFee()	onlyOwner
LatteVault (L: 230)	onlyOwner	setWithdrawFeePeriod()	onlyOwner
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol (L: 60)	LatteVault	renounceOwnership()	onlyOwner
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol (L: 69)	LatteVault	transferOwnership()	onlyOwner
@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol (L: 143)	LatteVault	grantRole()	-
@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol (L: 158)	LatteVault	revokeRole()	-
@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol (L: 178)	LatteVault	renounceRole()	-

Please note that the `OwnableUpgradeable` and `AccessControlUpgradeable` contracts are inherited from OpenZeppelin's library by the `LatteVault` contract.

5.2.2. Remediation

In the ideal case, the state variables should not be modifiable to keep the integrity of the smart contract. However, if modifications are needed, Inspex suggests limiting the use of these functions via the following options:

- Implementing community-run governance to control the use of these functions
- Using a Timelock contract to delay the changes for a sufficient amount of time, e.g., 24 hours

Note: When utilizing a `Timelock` contract to delay the owner's action. The effect will be applied to all `onlyOwner` modifiers. There are two functions that use the `onlyOwner` modifier but may not need a time delay:

- `emergencyWithdraw()` - The owner should be able to call it anytime for the emergency case, when the`MasterBarister` has a problem, the `emergencyWithdraw()` function can withdraw the $LATTE from `MasterBarister` to `LatteVault` contract and the users can withdraw their token from the contract.
- `inCaseTokensGetStuck()` - It is used to withdraw the token that's not the $LATTE from the `LatteVault` contract by the owner. This contract allows only the deposit of $LATTE, so when the

owner withdraws other tokens, there is no impact to the users.

In those cases, Inspex suggests creating a new role that can call these functions without the time delay.

5.3. Insufficient Logging for Privileged Functions

ID	IDX-003
Target	LatteVault
Category	General Smart Contract Vulnerability
CWE	CWE-778: Insufficient Logging
Risk	**Severity:** Very Low **Impact:** Low Privileged functions' executions cannot be monitored easily by the users. **Likelihood:** Low It is not likely that the execution of the privileged functions will be a malicious action.
Status	Resolved LatteSwap team has resolved this issue as suggested in commit `cd1a218c4342b146bcde429bcfaa46ce5c376286` by emitting events in the privileged functions.

5.3.1. Description

Privileged functions that are executable by the controlling parties are not logged properly by emitting events. Without events, it is not easy for the public to monitor the execution of those privileged functions, allowing the controlling parties to perform actions that cause big impacts to the platform.

For example, the owner can modify the `performanceFee` by executing `setPerformanceFee()` function in the `LatteVault` contract, and no event is emitted.

LatteVault.sol

```
206 function setPerformanceFee(uint256 _performanceFee) external onlyOwner {
207     require(
208         _performanceFee <= MAX_PERFORMANCE_FEE,
209         "LatteVault::setPerformanceFee::performanceFee cannot be more than
    MAX_PERFORMANCE_FEE"
210     );
211     performanceFee = _performanceFee;
212 }
```

The privileged functions without sufficient logging are as follows:

File	Contract	Function	Modifier
LatteVault (L: 197)	onlyOwner	setTreasury()	onlyOwner

LatteVault (L: 206)	onlyOwner	setPerformanceFee()	onlyOwner
LatteVault (L: 218)	onlyOwner	setWithdrawFee()	onlyOwner
LatteVault (L: 230)	onlyOwner	setWithdrawFeePeriod()	onlyOwner
LatteVault (L: 242)	onlyOwner	emergencyWithdraw()	onlyOwner
LatteVault (L: 249)	onlyOwner	inCaseTokensGetStuck()	onlyOwner

5.3.2. Remediation

Inspex suggests emitting events for the execution of privileged functions, for example:

LatteVault.sol

```
206 event SetPerformanceFee(uint256 _oldPerformanceFee, address
    _newPerformanceFee);
207 function setPerformanceFee(uint256 _performanceFee) external onlyOwner {
208     require(
209       _performanceFee <= MAX_PERFORMANCE_FEE,
210       "LatteVault::setPerformanceFee::performanceFee cannot be more than
    MAX_PERFORMANCE_FEE"
211     );
212     emit SetPerformanceFee(performanceFee, _performanceFee);
213     performanceFee = _performanceFee;
214 }
```

6. Appendix

6.1. About Inspex

Inspex is formed by a team of cybersecurity experts highly experienced in various fields of cybersecurity. We provide blockchain and smart contract professional services at the highest quality to enhance the security of our clients and the overall blockchain ecosystem.

Follow Us On:

Website	https://inspex.co
Twitter	@InspexCo
Facebook	https://www.facebook.com/InspexCo
Telegram	@inspex_announcement

6.2. References

[1] “OWASP Risk Rating Methodology.” [Online]. Available: https://owasp.org/www-community/OWASP_Risk_Rating_Methodology. [Accessed: 08-May-2021]

inspeX

CYBERSECURITY PROFESSIONAL SERVICE