SMART CONTRACT SECURITY AUDIT REPORT

Client: Lofi The Yeti

Date: 8th of January 2025

Appendix

We thank Lofi The Yeti for allowing us to conduct a Smart Contract Audit. This document outlines our methodology, limitations, and results of the security audit.

Timeline: 07.01.2025 – 08.01.2025

Audit Summary

Lofi The Yeti should acknowledge all the risks summed up in the risks section of the report.

5	0	0	0
Total Findings	Resolved	Acknowledged	Mitigated

Findings by severity	Findings Number	Resolved	Mitigated	Acknowledged
Critical	0	0	0	0
High	0	0	0	0
Medium	0	0	0	0
Low	4	0	0	0
Informational	1			

Vulnerability	Severity
	Critical
	High
	Medium
L01. Redundant Variable Assignment in Mint Reduces Code Clarity, L02. Unnecessary Dummy Field in LOFI Struct Affects Code Maintainability, L03. Missing Event Emissions, L04. Unnecessary Mint Function Declaration Creates Misleading Availability	Low
I01. Missing Error Messages in Mint Function	Informational

Introduction

Hakflow was commissioned by Lofi The Yeti to perform a Smart Contract Security Audit of the LOFi token. This report details the findings from the security audit conducted between 07.01.2025 and 08.01.2025, including our remediation efforts and recommendations for enhancing the code's security posture.

The objective was to identify vulnerabilities, assess their impact and likelihood, and provide strategic insights into strengthening the client side security.

Scope:

LOFi Token:

- https://suiscan.xyz/mainnet/coin/0xf22da9a24ad027cccb5f2d496cbe91de953d363513db08a3a734d361c7c17503::LOFI::LOFI/txs

Package ID:

- https://suiscan.xyz/mainnet/object/0xf22da9a24ad027cccb5f2d496cbe91de953d363513db08a3a734d361c7c17503

Timeline:

- **Assessment Start Date:** 07.01.2025
- **Assessment End Date:** 08.01.2025
- **Report Submission:** 09.01.2025

Team Composition:

- **Project Lead**: Ensures quality control and oversees project delivery.
- **Security Auditor**: Conduct the security audit.

Overview

Network	SUI
Address	0xf22da9a24ad027cccb5f2d496cbe91de953d363513db08a3a734d361c7c17503::LOFI::LOFI
Explorer	https://suiscan.xyz/mainnet/coin/0xf22da9a24ad027cccb5f2d496cbe91de953d363513db08a3a734d361c7c17503::LOFI::LOFI
Creator	0x6fe47d26b342d668fd11d0d09802c66e1ff2e0538308fecbc37ae842336028ab
Package ID	0xf22da9a24ad027cccb5f2d496cbe91de953d363513db08a3a734d361c7c17503
PackageID Owner	Immutable
Total Supply	1,000,000,000
Decimals	9
Name	LOFI
Symbol	LOFI
Description	Lofi is everyone's favorite Yeti on Sui
Icon URL	https://i.ibb.co/fM8QZXh/LOFI-PFP.png

Executive Summary

The LOFI is a token deployed on the Sui blockchain using the Move smart contract programming framework. It is implemented in the module 0xf22da9a24ad027cccb5f2d496cbe91de953d363513db08a3a734d361c7c17503::LOFI::LOFI as part of the package with ID 0xf22da9a24ad027cccb5f2d496cbe91de953d363513db08a3a734d361c7c17503.

The token's metadata indicates a total supply of 1,000,000,000 tokens and custom metadata attributes.

The visual identity of the token is defined by the image located at https://i.ibb.co/fM8QZXh/LOFI-PFP.png. This makes the token distinct in its presentation across various platforms and marketplaces. The owner of the contract is set to Immutable, meaning further changes to metadata or logic are restricted.

Methodology

The security evaluation was conducted using a blend of automated scanning tools and manual inspection to identify and validate security weaknesses. The assessment included:

- **Manual Review:** Deep review of token source code.

Code Breakdown

The code implements the LOFI token on the SUI chain, which utilizes the Move language and follows a structured code format, enabling seamless integration with dApps and marketplaces within the ecosystem. The token's visuals and metadata are available through decentralized URLs, enabling secure and persistent access.

Token Name: LOFI
Symbol: LOFI
Supply: 1,000,000,000
Decimals: 9
Description: Lofi is everyone's favorite Yeti on Sui.
Icon URL: https://i.ibb.co/fM8QZXh/LOFI-PFP.png

Deployment Details:
The LOFI token is created using the Move smart contract framework, specifically within the module 0xf22da9a24ad027cccb5f2d496cbe91de953d363513db08a3a734d361c7c17503::LOFI. The package ID for this implementation is available in the following link: 0xf22da9a24ad027cccb5f2d496cbe91de953d363513db08a3a734d361c7c17503.

Key Functionalities:

1. Minting:
   The token is minted using the mint function, which generates two objects:

   - Treasury Cap(v0): Defines the total supply of tokens available for the currency.
   - Coin Metadata(v1): Contains essential token details, including decimals, name, symbol, description, and icon URL.

2. Supply:
   The initial supply is minted and transferred to the account @0x6fe47d26b342d668fd11d0d09802c66e1ff2e0538308fecbc37ae842336028ab during the initialization.

3. Public Freezing:
   After the minting process, both the Treasury Cap and Coin Metadata are frozen using public_freeze_object to ensure immutability and secure management of token metadata.

Metadata:

The LOFI token leverages decentralized URLs for its visual identity and metadata, ensuring persistent and secure access to its attributes across platforms.

Definitions

Table. Issue Severity Definition

Severity	Description
Critical	These issues present a major security vulnerability that poses a severe risk to the system. They require immediate attention and must be resolved to prevent a potential security breach or other significant harm.
High	These issues present a significant risk to the system, but may not require immediate attention. They should be addressed in a timely manner to reduce the risk of the potential security breach.
Medium	These issues present a moderate risk to the system and cannot have a great impact on its function. They should be addressed in a reasonable time frame, but may not require immediate attention.
Low	These issues present no risk to the system and typically relate to the code quality problems or general recommendations. They do not require immediate attention and should be viewed as a minor recommendation.
Informational	These findings provide insights into the system or code but do not represent immediate or direct risks to its functionality, security, or stability. They typically offer guidance for potential improvements and best practices.

Issue Status Definition

Status	Description
New	The issue was presented to the customer. The remediation after the initial discovery was not yet made.
Resolved	The issue has been fully addressed and fixed.
Acknowledged	The issue was not fixed as a result of the remediation. The customer was informed of the risks associated with it.
Mitigated	The issue was reduced or controlled, but not necessarily completely eliminated.

Low Severity Issues

L01. Redundant Variable Assignment in Mint Reduces Code Clarity

Severity	Low
Impact	Low
Likelihood	Low
Status	New

Description:

The contract sets the variable v2 to the value of v0 in the mint function, even though it could directly utilize the v0 variable. This redundant assignment adds unnecessary code complexity without providing additional functionality or value.

Such inefficiencies can reduce code readability and make the function less readable, potentially confusing for users.

Recommendation:

It is recommended to eliminate the redundant assignment of the v2 variable, by directly using the v0 variable in the return statement. This will reduce unnecessary operations, making the function cleaner and more straightforward.

By avoiding redundant variables, the code becomes easier to read and understand, lowering the chances of confusion. Simplifying the function also ensures that users and developers can focus on the essential logic without being distracted by unnecessary elements.

Applicable Code:

This finding applies to the following lines of code in the mint function:

Line#7

```
let v2 = v0;
...
(v2, v1)
```

L02. Unnecessary Dummy Field in LOFI Struct Affects Code Maintainability

Severity	Low
Impact	Low
Likelihood	Low
Status	New

Description:

The LOFI struct contains a field named dummy_field that has no discernible purpose in the current implementation. This unnecessary field adds complexity to the code without providing value. Its presence adds unnecessary complexity to the codebase, increasing the risk of confusion among users or developers who might misinterpret its role or significance.

This redundant field also detracts from the maintainability and efficiency of the contract, as it introduces ambiguity and slightly inflates the memory footprint if the struct is stored.

Recommendation:

Remove the dummy_field from the LOFI struct if it is not required for future logic or functionality. Removing this field will simplify the code, improve readability, and ensure developers can focus on the essential aspects of the contract without being distracted by unused or unnecessary components.

Simplification also aids in reducing maintenance overhead and minimizing the risk of potential misinterpretations or errors in the future.

Applicable Code:

This finding applies to the following definition of the LOFI struct:

Line#2

```
struct LOFI has drop {
    dummy_field: bool,
}
```

L03. Missing Event Emissions

Severity	Low
Impact	Low
Likelihood	Low
Status	New

Description:

The contract lacks event emissions for significant actions such as token initialization and minting. The absence of events reduces the ability to track these crucial actions on-chain, making it difficult for users or external systems to monitor and verify contract activity effectively. This lack of traceability can impact transparency and hinder the integration of the contract with off-chain systems.

Recommendation:

Implement event emission for key actions like token initialization and minting. This addition will improve traceability and transparency by providing a clear on-chain record of significant events. Events can serve as a reliable mechanism for users and external systems to monitor contract interactions and verify important actions.

Applicable Code:

This finding applies to the init and mint functions:

Line#2

```
fun init(arg0: LOFI, arg1: &mut 0x2::tx_context::TxContext) {
    ...
}

public fun mint(arg0: LOFI, arg1: &mut 0x2::tx_context::TxContext) :
(0x2::coin::TreasuryCap<LOFI>, 0x2::coin::CoinMetadata<LOFI>) {
    ...
}
```

L04. Unnecessary Mint Function Declaration Creates Misleading Availability

Severity	Low
Impact	Low
Likelihood	Low
Status	New

Description:

The contract includes a mint function that is solely invoked during the init function to mint tokens. Since minting is only intended to occur during initialization, the mint function could be integrated directly into the init function. Keeping mint as a standalone function may mislead users into believing that minting functionality is still available outside of initialization, which could cause confusion and misinterpretation of the contract's capabilities.

Recommendation:

This finding applies to the mint function, which is invoked exclusively within the init function and does not serve any other purpose.

Applicable Code:

This finding applies to the init and mint functions:

Line#7,12

```
fun init(arg0: LOFI, arg1: &mut 0x2::tx_context::TxContext) {
    let (v0, v1) = mint(arg0, arg1);
    ...
}

public fun mint(arg0: LOFI, arg1: &mut 0x2::tx_context::TxContext) :
(0x2::coin::TreasuryCap<LOFI>, 0x2::coin::CoinMetadata<LOFI>) {
    ...
}
```

Informational Severity Issues

I01. Missing Error Messages in Mint Function

Severity	Info
Impact	N/A
Likelihood	N/A
Status	New

Description:

The contract lacks assertions to enforce crucial conditions and fails to provide error messages to users attempting to interact with its functions. Assertions are essential for ensuring that specific conditions are met during contract execution.

Without them, unexpected behaviors or invalid states may occur, leaving users without meaningful feedback on transaction failures. Specifically, the mint function, which is designed as public, cannot be executed successfully, but the contract does not provide a clear error message when users attempt to execute the mint function, resulting in confusion and a poor user experience.

Recommendation:

Enhance the contract by including assertions to enforce necessary conditions and adding descriptive error messages to all functions, particularly the mint function. These error messages should provide users with clear feedback on why the function cannot be executed and the conditions required for successful execution.

This will improve the robustness of the contract by ensuring invalid states are caught and enhance the user experience by providing precise reasons for transaction failures. By addressing these issues, the contract will provide a more reliable and user-friendly interface, reducing potential confusion and improving overall system feedback.

Applicable Code:

This finding applies to the mint function:

Line#6,12

```
public fun mint(arg0: LOFI, arg1: &mut 0x2::tx_context::TxContext) :
(0x2::coin::TreasuryCap<LOFI>, 0x2::coin::CoinMetadata<LOFI>) {
    let (v0, v1) = 0x2::coin::create_currency<LOFI>(arg0, 9, b"LOFI", b"LOFI", b"Lofi is
everyone's favorite Yeti on Sui",
0x1::option::some<0x2::url::Url>(0x2::url::new_unsafe_from_bytes(b"https://i.ibb.co/fM8Q
ZXh/LOFI-PFP.png")), arg1);
    let v2 = v0;
    0x2::coin::mint_and_transfer<LOFI>(&mut v2, 1000000000000000000,
@0x6fe47d26b342d668fd11d0d09802c66e1ff2e0538308fecbc37ae842336028ab, arg1);
    (v2, v1)
}
```

Disclaimers

Hakflow Disclaimer

The extension reviewed in this audit has been analyzed based on the best industry practices available at the time of this report. The evaluation includes assessments of the configurations, deployment strategies and functionalities to perform intended functions.

This audit does not provide any warranties regarding the security of the systems or code assessed. It should not be considered as an exhaustive evaluation of the security, reliability, or error-free operation, nor should it be seen as an endorsement of the overall safety.

While we have endeavored to conduct a thorough analysis and provide accurate findings in this report, it is important to recognize that this report should not be the sole resource relied upon. We strongly recommend engaging in additional independent audits and continuous security monitoring to ensure the ongoing security and integrity of the environment.

Technical Disclaimer

Systems and their underlying code are complex and involve multiple layers of technology, including software, and network components. Each of these components can have inherent vulnerabilities that could potentially be exploited. Therefore, despite the thoroughness of the audit, we cannot guarantee absolute security of the audited systems or code. Continuous vigilance, regular updates, and a proactive security strategy are crucial to maintaining the security and integrity of any environment.

Appendix 1. Severity Definitions

In assessing the project's security posture, we employ a risk-based approach that evaluates the potential impact and exploitability of identified vulnerabilities. This method utilizes a matrix of impact and likelihood, a standard tool in risk management, to aid in the assessment and prioritization of risks.

Risk Levels

Critical: Critical vulnerabilities in an environment are those that are easily exploitable and can lead to severe consequences such as complete service disruption or compliance violations. These vulnerabilities generally allow unauthorized access or control over resources, leading to a direct impact on business operations and security.

High: High vulnerabilities are more challenging to exploit and might require specific conditions to be met. While their impact may be somewhat less devastating than critical vulnerabilities, they can still result in considerable damage such as data leaks, partial service disruption, or compliance issues.

Medium: Medium vulnerabilities include issues that may lead to certain operational inefficiencies or security risks that are more difficult to exploit and usually do not result in direct asset loss. These might include minor compliance deviations, moderate data exposure risks, or inefficiencies in resource utilization. Such vulnerabilities often require specific conditions or sequences of actions to be exploited.

Low: Low risk vulnerabilities pose minimal risk to system operation or security. They are typically related to non-critical inefficiencies, minor coding practices, or low-level configuration issues that have a very limited chance of being exploited. While these findings may not immediately affect system performance or security, addressing them can still contribute to overall improvement.

Informational: Informational vulnerabilities provide helpful insights or observations without indicating any immediate or potential risk to system functionality, security, or performance. They are typically suggestions, or areas where further optimization could be beneficial.

Impact Levels

High Impact: Risks with a high impact are associated with severe consequences, such as significant financial losses, substantial reputational damage, or major operational disruptions. High impact issues in environments typically involve breaches leading to data loss, denial of service attacks disrupting service availability, or security incidents that affect regulatory compliance.

Medium Impact: Risks with medium impact could lead to moderate financial losses or reputational harm. These may include less severe breaches that result in limited data exposure, minor compliance issues, or performance degradations that affect user experience but do not completely halt operations.

Low Impact: Risks with low impact are unlikely to cause direct financial losses or significant operational disruption. These issues generally pertain to inefficiencies or minor non-compliance with best practices that may affect the system's optimal functioning or minor security best practices deviations which do not directly compromise security but could lead to vulnerabilities if not addressed.

Likelihood Levels

High Likelihood: Risks with a high likelihood are those that are expected to occur frequently or are very likely to occur due to existing vulnerabilities or weaknesses in the system configuration or security measures. This category also includes risks from common attack vectors that target prevalent vulnerabilities across similar platforms.

Medium Likelihood: Risks with a medium likelihood are possible but not as probable as those in the high likelihood category. These might be due to less critical vulnerabilities that require specific conditions to be exploited or are known to a smaller group of potential attackers, representing a moderate level of threat.

Low Likelihood: Risks of low likelihood are those that are unlikely to occur and may require highly specific conditions or complex strategies to exploit. These risks typically stem from obscure or theoretical vulnerabilities that are not generally targeted by attackers due to the high effort and low success rate associated.

Risk Level	High Impact	Medium Impact	Low Impact
High Likelihood	**Critical**	**High**	**Medium**
Medium Likelihood	**High**	**Medium**	**Low**
Low Likelihood	**Medium**	**Low**	**Low**

Document

Name	Smart Contract Audit Report for Lofi The Yeti
Approved By	George Kougias | BDM at Hakflow
Audited By	Dimitris Pallis | CEO at Hakflow
Website	https://hakflow.com
Changelog	07.01.2025 – Preliminary Report 08.01.2025 - Final Report

Hakflow is at the forefront of the cybersecurity industry, forging impactful partnerships with leading entities such as *Plume Network* and *VaasBlock* in the blockchain space and *NSFOCUS* in traditional cybersecurity. Our commitment to excellence is further demonstrated by our active participation in renowned security conferences, including *ETHSofia* and *GISEC Dubai*.

Client Testimonial - Rizwan Zareef, CEO of Darkguard

"*Through our collaboration with Hakflow, our clients have proactively strengthened their cybersecurity defenses and are now better prepared for the evolving threat landscape*"

Industry Certified

https://hakflow.com