TENSET

Smart Contract Audit

LUCKY LEMU

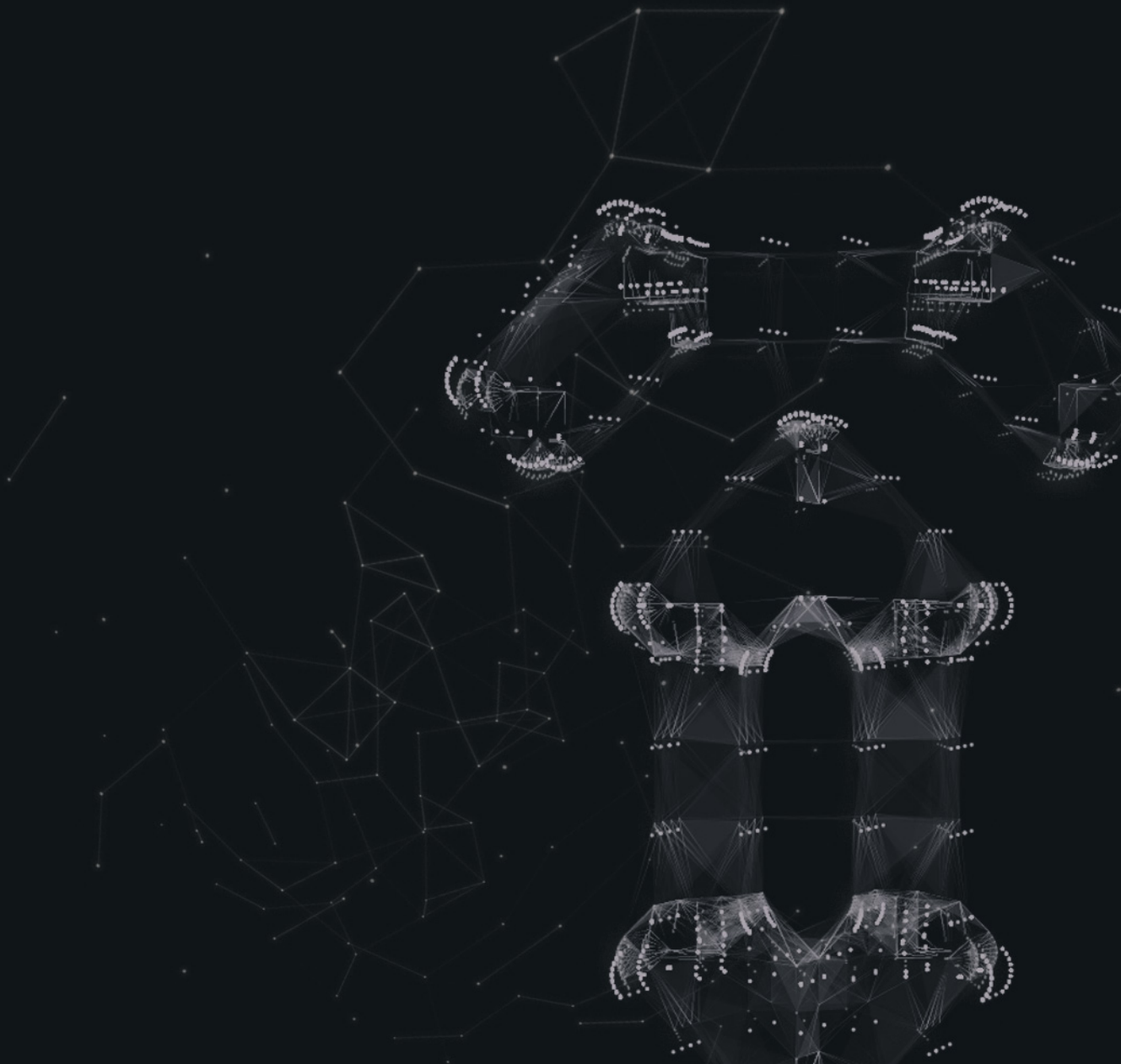

Table of Content

Executive Summary

Project Summary

Language	Codebase	Commits	Platform
Solidity	Private	0-f8b3904e52019acd2b57f7356315d0039561565	Base

Audit Scope

ID	File	SHA256
LM	Lemu.sol	9e858dc210d80379cea96f1587bfb b26e9d7b857ec3e7816a2a9faf5bc1a08f8
ERR	Errors.sol	dbc9dd2345652ee911e6981091f78 956ce53ee953d19652faec158776b6b8374
WTH	Withdrawable.sol	f47589a7f2169a9e6965749204796 498a792889b1f09c8a651c7d582fa41de93

General security assumptions

In conducting our security audit of the LUCKY LEMU, we have made a number of important security assumptions. These assumptions are fundamental to our analysis and should be considered when evaluating the overall security and potential vulnerabilities of the system:

Security of the Underlying Blockchain
Our security analysis is predicated on the assumption that the underlying blockchain (for example, Ethereum) is secure and functions as intended. Potential vulnerabilities or flaws in the blockchain protocol are outside the scope of this audit.

Trusted Environment
We assume that users of the smart contract will interact with it in a secure and trusted environment. This encompasses the use of secure, up-to- date software and hardware, which is free from malware or any potential interference from malicious third parties.

Private Key Security
Users of the smart contract are assumed to safeguard their private keys effectively. This involves not sharing private keys, securely storing key backups, and using hardware wallets or other secure means for key management.

These security assumptions form the basis of our audit. Deviations from these assumptions could potentially lead to risks or vulnerabilities not covered in this analysis. Therefore, the effective management of these factors is critical to ensure the ongoing security of the LUCKY LEMU.

Findings

Audit Overview

Issues

Severity	Found	Resolved	Partially Fixed	Acknowledged
Critical	0	0	0	0
High	0	0	0	0
Medium	0	0	0	0
Low	0	0	0	0
Informational	0	2	0	0
Total	**0**	**2**	**0**	**0**

LM-1
Pack variables into one slot for optimization

Informational Resolved

Description

Variables `buybackStrategy` and `supervisedTransfersEndAt` can be optimized for storage efficiency. These variables can be packed into a single storage slot, reducing gas costs. Additionally, changing the data type of `supervisedTransfersEndAt` to `uint64` can further optimize storage usage, as it only requires 8 bytes compared to the 20 bytes required for an address.

Recommendation

- Pack the variables `buybackStrategy` and `supervisedTransfersEndAt` into a single storage slot to reduce gas costs.

- Change the data type of `supervisedTransfersEndAt` to `uint64` to optimize storage usage.

ERR-1
Incorrect NatSpec Documentation

Informational Resolved

Description

In the Errors.sol, the NatSpec documentation for the `UnacceptableReference` error states that the given reference is `address(0)`. However, analysis of the Catamoto.sol contract at line 55 reveals that the reference may not always be `address(0)`. Therefore, the NatSpec comment is inaccurate and should be corrected to provide a more appropriate description of the error condition.

Recommendation

Update the NatSpec comment for the `UnacceptableReference` error in Errors.sol to accurately reflect the error condition. Instead of stating that the given reference is `address(0)`, describe the error as 'Given reference is unsupported' to align with the actual behavior observed in the contract.

Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted to and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, Consultant cannot guarantee the explicit security of the audited smart contracts.