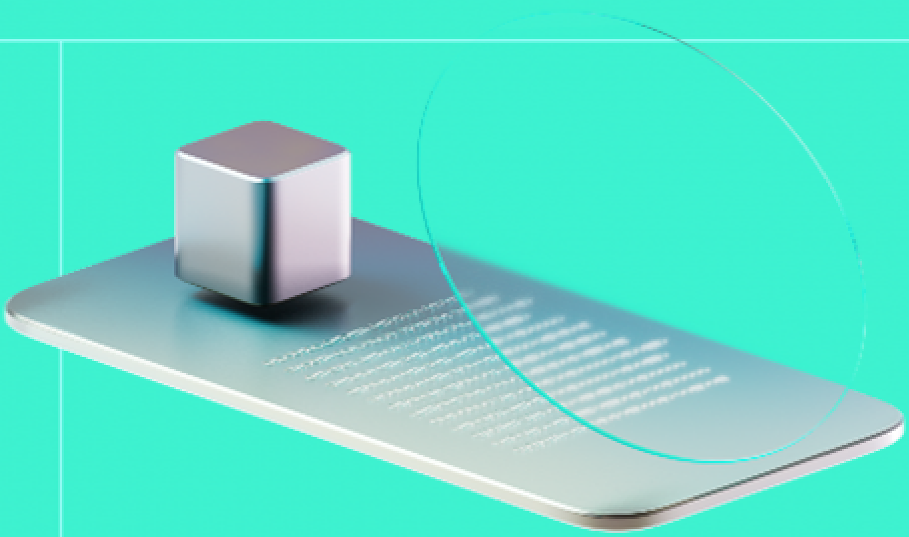
Smart Contract Code Review And Security Analysis Report



Customer: Electroneum

Date: 27.12.2023

We thank Electroneum for allowing us to conduct a Smart Contract Security Assessment. This document outlines our methodology, limitations, and results of the security assessment.

ETNBridge is a one-way bridge that was created to migrate all users' balances from a Monero-based chain to an Ethereum-based chain.

Platform: EVM

Language: Solidity

Tags: Bridge

Timeline: 18.12.2023-27.12.2023

Methodology: https://hackenio.cc/sc_methodology

Last Review Scope

Repository	https://github.com/electroneum/electroneum-sc-contracts
Commit	9c9b250

Audit Summary

10/10	7/10	57%	0/10
Security Score	Code quality score	Test coverage	Documentation quality score

Total 9/10

The system users should acknowledge all the risks summed up in the risks section of the report

0	0	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	0
Medium	0
Low	0

Document

Name	Smart Contract Code Review and Security Analysis Report for Electroneum
Audited By	David Camps Novi, Viktor Lavrenenko
Approved By	Przemyslaw Swiatowiec
Website	https://electroneum.com
Changelog	27/12/2023 - Final Report

Table to Contents

System Overview **6**

Privileged Roles 6

Executive Summary **7**

Documentation Quality 7

Code Quality 7

Test Coverage 7

Security Score 7

Summary 7

Risks **8**

Findings **9**

Vulnerability Details 9

Observation Details 9

F-2023-0189 - Floating Pragma - Info 9

F-2023-0198 - Unused Variables - Info 10

F-2023-0248 - Use Of Transfer Instead Of Call To Send Native Assets - Info 11

Disclaimers 12

Hacken Disclaimer 12

Technical Disclaimer 12

Appendix 1. Severity Definitions 13

Appendix 2. Scope 14

System Overview

The Electroneum system consists of a single smart contract: `ETNBridge`.

- The Electroneum project is migrating from their legacy Monero-based chain to a new Ethereum-based chain. To do so, the team created an uni-directional bridge that will be used to migrate all user balances from the old chain to the new one.
- User balances in the old chain will be burned, whilst the same amount will be sent on the new chain to the corresponding holders.
- Using a centralized off-chain system, the team will send the required tokens of the new chain into the `ETNBridge` contract, so that the balances can be distributed.
- The ETNBridge owner will call the function `crosschainTransfer()` to migrate the balances sequentially.

Privileged roles

- Owner:
  - Can pause/unpause the contract.
  - Executes the migration via `crosschainTransfer()` function.
  - Can upgrade the contract via `upgradeTo/upgradeToAndCall()` functions.

Executive Summary

This report presents an in-depth analysis and scoring of the customer's smart contract project. Detailed scoring criteria can be referenced in the scoring methodology.

Documentation quality

The total Documentation Quality score is **0** out of **10**.

- Functional requirements are not provided.
  - The project's purpose is not described.
  - The project's features are not explained.
  - Use cases are missing.
- The technical description is not provided:
  - The technical specification is missing.
  - Deployment instructions are not provided.
  - NatSpec is not sufficient for the main functionality of the `ETNBridge.sol`.

Code quality

The total Code Quality score is **10** out of **10**.

- Best practices are followed.
- The development environment is configured.

Test coverage

Code coverage of the project is **57%** (branch coverage):

- For projects with less than 250 LOC (Lines of Code) the test coverage is not mandatory, and it is not accounted for in the final score.

Security score

Upon auditing, the code was found to contain no issues, leading to a security score of **10** out of **10**.

All identified issues are detailed in the “Findings” section of this report.

Summary

The comprehensive audit of the customer's smart contract yields an overall score of **9.** This score reflects the combined evaluation of documentation, code quality, test coverage, and security aspects of the project.

Risks

- The ETNBridge contract is upgradeable, which means the protocol owners can change the logic of the contract without prior notice.
- Most of the bridge management is handled off-chain and in a highly-centralized manner. As such, only a small part of the whole system could be audited, which means that a high security score in this audit does not guarantee the safety of the system for users.
- The token supply in both the old legacy chain and the new smart chain is out-of-scope and thus the correct management of supply and its inflation cannot be supervised.
- The protocol owner is not multi-signature managed, although it has a critical role in the system. A 3/5 multi-signature should be used at least.
- Since the balance of tokens are sent to the ETNBridge prior to user distribution, it is not guaranteed that the contract contains the necessary tokens. It may be the case that some balances are temporarily on hold until new tokens are sent to the contract.
- The contract can be paused at will by the contract owner, effectively putting user funds on hold at will.
- The lack of signatures creates a centralization risk since the input data can be altered if the oracle goes malicious.

Findings

Vulnerability Details

Observation Details

F-2023-0189 - Floating Pragma - Info

Description:	A **floating pragma** in Solidity refers to the practice of using a pragma statement that does not specify a fixed compiler version but instead allows the contract to be compiled with any compatible compiler version. This issue arises when pragma statements like pragma solidity ^0.8.0 are used without a specific version number, allowing the contract to be compiled with the latest available compiler version. This can lead to various compatibility and stability issues. **Version Compatibility:** Using a floating pragma makes the contract susceptible to potential breaking changes or unexpected behavior introduced in newer compiler versions. Contracts that rely on specific compiler features or behaviors may break when compiled with a different version. **Interoperability Issues:** Contracts compiled with different compiler versions may have compatibility issues when interacting with each other or with external services. This can hinder the interoperability of the contract within the Ethereum ecosystem. The project uses floating pragma ^0.8.13.
Assets:	• ETNBridge.sol
Status:	Fixed

Recommendations

Recommendation:	Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment. Consider known bugs for the compiler version that is chosen. **Remediation** (Revised commit: 9c9b250) : The floating pragma issue was addressed by locking the pragma to a Solidity version 0.8.23.

F-2023-0198 - Unused Variables - Info

Description: The `crosschainBalance` mapping variable is declared in `ETNBridge` contract. However, it is not used anywhere in the code.

Unused variables increase deployment costs, decrease code readability, and make the code look like some functionalities are unfinished.

Assets:

- ETNBridge.sol

Status: Fixed

Recommendations

Recommendation: Remove the unused variable (`crosschainBalance`) or implement the necessary functionality to use it.
Remediation (Revised commit: 9c9b250): Additional functionality was added to the `crosschainTransfer()` function to track the total amount of ETN migrated per address as well as the function `getAddressCosschainAmount()` to read this value.

F-2023-0248 - Use Of Transfer Instead Of Call To Send Native Assets - Info

Description: The contract uses built-in `transfer()` function for transferring native tokens.

The `transfer()` function was commonly used in earlier versions of Solidity for its simplicity and automatic reentrancy protection. However, it was identified as potentially problematic due to its fixed gas limit of `2300`.

Some of the EVM-compatible chains like ZkSync Era use a dynamic and divergent gas measurement method. Using transfer() can exceed the 2300 gas limit, causing the transaction to revert automatically.

Given that the audited solution is a contract that is part of the bridge and the transfer of substantial user funds, the consequences of using `transfer()` can be severe. A failure in the transfer process due to gas limit constraints could lead to significant, irreversible financial losses. This presents a high risk to the contract's operational integrity and the security of the funds, with potential impacts including protocol-level Denial of Service (DoS) and no viable recovery options.

Assets:

- ETNBridge.sol

Status: Accepted

Recommendations

Recommendation: It is recommended to use the built-in call() function instead of transfer() to transfer native assets. This method does not impose a gas limit, it provides greater flexibility and compatibility. Furthermore, the code using the call() function will be able to work even when the gas costs of the opcodes will be increased and surpass 2300.

Resolution (Revised commit: 9c9b250): The team is aware of the consequences of using `transfer()` and communicated to us the following: "the `ETNBridge` should always transfer to EOA (fixed gas) anyway and the limit imposed by `transfer()` is actually beneficial in this scenario.

External References:

- Stop using transfer
- ZkSync Era Failure

Disclaimers

This audit report focuses exclusively on the security assessment of the contracts within the specified review scope. Interactions with out-of-scope contracts and off-chain functionality are presumed to be correct and are not examined in this audit.

While we have diligently identified and mitigated potential security risks within the defined scope, it is important to note that our assessment is confined to the isolated contracts within this scope. The overall security of the entire system, including external contracts and integrations beyond our audit scope, cannot be guaranteed.

Users and stakeholders are urged to exercise caution when assessing the security of the broader ecosystem and interactions with out-of-scope functionality. For a comprehensive evaluation of the entire system, additional audits and assessments outside the scope of this report are necessary.

This report serves as a snapshot of the security status of the audited contracts within the specified scope at the time of the audit. We strongly recommend ongoing security evaluations and continuous monitoring to maintain and enhance the overall system's security.

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing 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 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.

Technical Disclaimer

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, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Appendix 1. Severity Definitions

When auditing smart contracts, Hacken is using a risk-based approach that considers **Likelihood**, **Impact**, **Exploitability** and **Complexity** metrics to evaluate findings and score severities.

Reference on how risk scoring is done is available through the repository in our Github organization:

hknio/severity-formula

Severity	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.
Medium	Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.
Low	Major deviations from best practices or major Gas inefficiency. These issues will not have a significant impact on code execution, do not affect security score but can affect code quality score.

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Scope Details

Repository	https://github.com/electroneum/electroneum-sc-contracts
Commit	9b41cd8804e5fe0f2ce6309a7d65faf86f3f7f1c
Whitepaper	N/A
Requirements	N/A
Technical Requirements	N/A

Contracts in Scope

./contracts/ETNBridge.sol