



AuditAce

FROM INCEPTION TO SUCCESS



EHZ

Smart Contract Audit Report



Overview

AUDITACE team has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks.

Audit Result: **Passed with No Risk**

Audit Date: Jan 22, 2024

Audit Team: TEAM AUDITACE

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	0	0	0
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0



Used Tools

Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
- Back-doors
- Vulnerability
- Accuracy
- Readability

Tools

- DE
- Open Zeppelin
- Code Analyzer
- Solidity Code
- Compiler
- Hardhat

Manual Review:

A line by line code review has been performed by audit ace team.

Slither :

The code has undergone static analysis using Slither.



Token Summary

Parameter	Result
Token Address	0xA90bED1eDe77CAd3245f18b52cC1a5E7Ca307F92
Owner	0xF72b1DeD253ad7C5Ca7693f73A033F8c2Cc2a2Fc
Deployer	0xF72b1DeD253ad7C5Ca7693f73A033F8c2Cc2a2Fc
Token Type	ERC 20
Contract Checksum	Be1c3a4fbb6e83e8393a57617b5a5b22
Decimals	18
Supply	9,000,000,000
Platform	Ethereum Chain
Compiler	v0.6.1+commit.e6f7d5a4
Token Name	EHZ
Symbol	EHZ
License Type	default
Language	Solidity



STATIC ANALYSIS

A static analysis of the code was performed using Slither. No issues were found.

```
ZToken._name (EHZTOKEN.sol#480) shadows:
- ERC20._name (EHZTOKEN.sol#281)
ZToken._symbol (EHZTOKEN.sol#481) shadows:
- ERC20._symbol (EHZTOKEN.sol#282)
ference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
IFO:Detectors:
C20.constructor(string,string,uint256,address).name (EHZTOKEN.sol#292) shadows:
- ERC20.name() (EHZTOKEN.sol#384-386) (function)
C20.constructor(string,string,uint256,address).symbol (EHZTOKEN.sol#292) shadows:
- ERC20.symbol() (EHZTOKEN.sol#312-314) (function)
C20.constructor(string,string,uint256,address).totalSupply (EHZTOKEN.sol#292) shadows:
- ERC20.totalSupply() (EHZTOKEN.sol#336-338) (function)
- IERC20.totalSupply() (EHZTOKEN.sol#180) (function)
ference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
IFO:Detectors:
feMath.div(uint256,uint256) (EHZTOKEN.sol#117-119) is never used and should be removed
feMath.div(uint256,uint256,string) (EHZTOKEN.sol#132-138) is never used and should be removed
feMath.mod(uint256,uint256) (EHZTOKEN.sol#151-153) is never used and should be removed
feMath.mod(uint256,uint256,string) (EHZTOKEN.sol#166-169) is never used and should be removed
feMath.mul(uint256,uint256) (EHZTOKEN.sol#92-104) is never used and should be removed
feMath.sub(uint256,uint256) (EHZTOKEN.sol#63-65) is never used and should be removed
ference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
IFO:Detectors:
pragma version0.6.1 (EHZTOKEN.sol#5) allows old versions
lc-0.6.1 is not recommended for deployment
ference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
IFO:Detectors:
instant EHZToken._name (EHZTOKEN.sol#480) is not in UPPER_CASE_WITH_UNDERSCORES
instant EHZToken._symbol (EHZTOKEN.sol#481) is not in UPPER_CASE_WITH_UNDERSCORES
ference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
IFO:Detectors:
ZToken.slitherConstructorConstantVariables() (EHZTOKEN.sol#478-490) uses literals with too many digits:
- INIT_TOTALSUPPLY = 900000000000 * 10 ** 18 (EHZTOKEN.sol#483)
ference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
IFO:Slither:EHZTOKEN.sol analyzed (5 contracts with 93 detectors), 16 result(s) found
```

CONTRACT FUNCTION SUMMARY

Can edit Tax?

NOT DETECTED

Can take back Ownership?

NOT DETECTED

Is Blacklisted?

NOT DETECTED

Is Whitelisted?

NOT DETECTED

Is Mintable?

NOT DETECTED

Can transfer Pausable?

NOT DETECTED

Is Trading with CooldownTime?

NOT DETECTED

AUDIT METHODOLOGY

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
 - Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
 - Test coverage analysis determines whether the test cases are covering the code and how much code is exercised when we run the test cases.
 - Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
 - Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.
-



Issues Checking Status

No	Issue Description	Checking Status
1	Compiler warnings.	Passed
2	Race conditions and Reentrancy. Cross-function race conditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Front running.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed
10	Methods execution permissions.	Passed
11	Design Logic.	Passed
12	Cross-function race conditions.	Passed
13	Safe Zeppelin module.	Passed
14	Malicious Event log.	Passed
15	Scoping and Declarations.	Passed
16	Fallback function security.	Passed
17	Arithmetic accuracy.	Passed



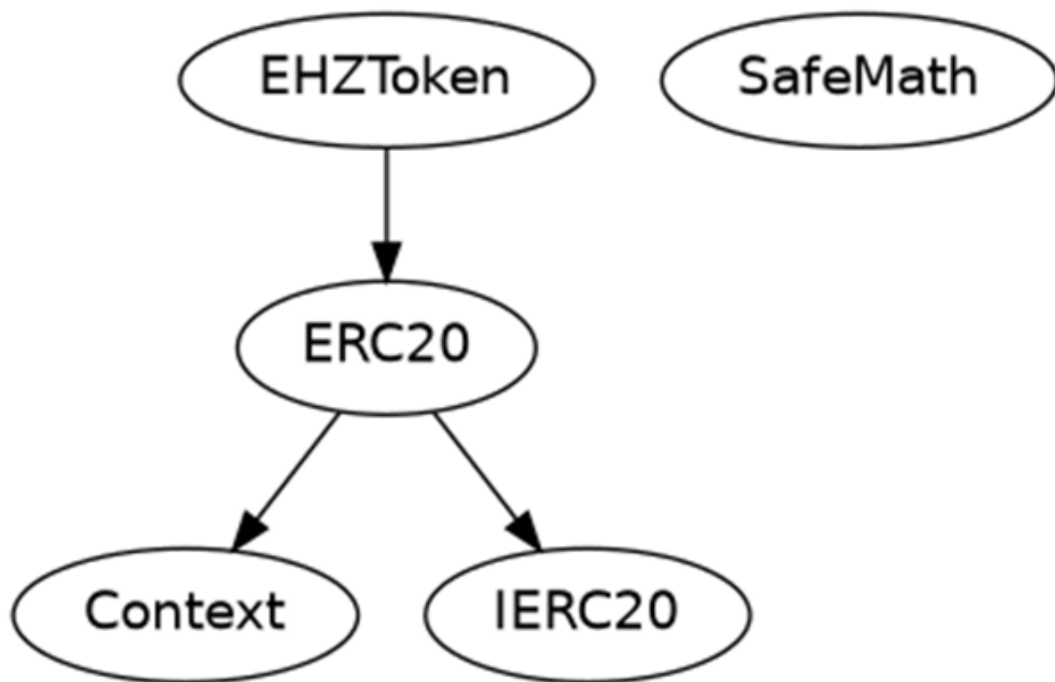
SWC ATTACK TEST

SWC ID	Description	Test Result
SWC-100	Function Visibility	Passed
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	Passed
SWC-103	Floating Pragma	Passed
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed
SWC-107	Re-entrancy	Passed
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegate Call to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	Passed



SWC ID	Description	Test Result
SWC-117	Signature Malleability	Passed
SWC-118	Incorrect Constructor Name	Passed
SWC-119	Shadowing State Variables	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-122	Lack of Proper Signature Verification	Passed
SWC-123	Requirement Violation	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-125	Incorrect Inheritance Order	Passed
SWC-126	Insufficient Gas Grieving	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-129	Typographical Error	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed
SWC-131	Presence of unused variables	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-133	Hash Collisions with Multiple Variable Length Arguments	Passed
SWC-134	Unencrypted Private Data On-Chain	Passed

Inheritance Tree



Classification of Risks

Severity

Description

◆ High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
◆ Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
◆ Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
◆ Gas Optimization /Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

Findings

Severity

Found

◆ High-Risk	0
◆ Medium-Risk	0
◆ Low-Risk	0
◆ Gas Optimization / Suggestions	0



Disclaimer

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Always Do your own research and protect yourselves. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research before making any investments. This document should not be presented as a reason to buy or not buy any particular token.

The Auditace team disclaims any liability for any resulting loss or profit.



ABOUT AUDITACE

Audit Ace is built, to combat financial fraud in the cryptocurrency industry, a growing security firm that provides audits, Smart contract creation, and end-to-end solutions to all crypto-related queries.

Website - <https://auditace.tech/>

Telegram - https://t.me/Audit_Ace

Twitter - https://twitter.com/auditace_

Github - <https://github.com/Audit-Ace>
