Scrysec.
Auditblock

Azteq (AZTEQ)

v0.8.17+commit.8df45f5f

v0.8.17

Low-Risk	Medium-Risk	High-Risk
Low-risk code	Medium-risk code	High-risk code

AZTEQ

0x9194d7e750217f9637d5f35592411459bd587dae

Executive Summary

Project Name Azteq

Overview The Azteq ecosystem is design to mint and stake erc tokens

Method Manual Review, Functional Testing, Automated Testing etc.

Scope of Audit The scope of this audit was to analyze the contract codebase for quality, security, and correctness.

High Medium

Low Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	0	0

Types of Severities

High

A high-severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Techniques and Methods

The overall quality of code.

- Use of best practices.
- Code documentation and comments match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrance and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, and their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms Used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Phase 1

Project - Azteq

High Severity Issues

No issues found

Medium Severity Issues

No issues found

Low Severity Issues

1. Ownership Methods (manipulate ownership)

```
function changeOwner(address newOwner) public {
    require(msg.sender == _owner, "Only the current owner can change the owner.");
    _owner = newOwner;
}
```

Description

Our auditor found this mistake manually. It is not a bug or vulnerability. We are simply acknowledging that the “changeOwner” Function has simple Detect missing zero address validation.

Recommendation

It is important to note that not define the ownership with external calls. You can use the Openzeppelin ownership contract. staying away from calling ownership with external calls. Just use a modifier and add o to avoid an attack for that. Using modifiers is good practice.

Status

Acknowledged

Informational Severity Issues

No issues found

Smart Contract Weakness Classification (SWC) Vulnerabilities for Attacks

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- Dangerous strict equalities
- Tautology or contradiction
- Missing Zero Address Validation
- Return values of low-level calls
- Revert/require functions
- Private modifier
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Phase 2

```
 Azteq.changeOwner(address) (contracts/Azteq.sol#562-565) should emit an
event for:
        - _owner = newOwner (contracts/Azteq.sol#564)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#missing-events-access-control

INFO:Detectors:
Azteq.changeOwner(address).newOwner (contracts/Azteq.sol#562) lacks a
zero-check on :
                - _owner = newOwner (contracts/Azteq.sol#564)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#missing-zero-address-validation
INFO:Detectors:
Azteq.mintDaily() (contracts/Azteq.sol#579-589) uses timestamp for
comparisons
        Dangerous comparisons:
        - require(bool,string)(currentTimestamp >= _lastMintTimestamp +
86400,Tokens can only be minted once per day.) (contracts/Azteq.sol#583-
586)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#block-timestamp



Context._msgData() (contracts/Azteq.sol#20-22) is never used and should
be removed
ERC20._burn(address,uint256) (contracts/Azteq.sol#408-423) is never used
and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (contracts/Azteq.sol#3) allows old versions
Pragma version^0.8.0 (contracts/Azteq.sol#28) allows old versions
Pragma version^0.8.0 (contracts/Azteq.sol#107) allows old versions
Pragma version^0.8.0 (contracts/Azteq.sol#134) allows old versions
Pragma version^0.8.0 (contracts/Azteq.sol#498) allows old versions
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#incorrect-versions-of-solidity

INFO:Detectors:
Azteq.slitherConstructorConstantVariables() (contracts/Azteq.sol#500-590)
uses literals with too many digits:
        - _MAX_SUPPLY = 4000000000 * (10 ** 18) (contracts/Azteq.sol#501)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#too-many-digits
```

Functional Testing

Some of the tests performed are mentioned below:

- ✔ Should Approve Tokens
- ✔ Should the Just Owner Change Ownership
- ✔ Should revert when none-owner calls onlyOwner methods
- ✔ Should work mint method correctly
- ✔ Should stake or unstake

Closing Summary

In this report, we have considered the security of Azteq. We performed our audit according to the procedure described above.

Several issues were identified during the audit process, and their severity levels have been classified. Recommendations and best practices have also been provided to enhance code quality and security posture. The team has acknowledged all identified issues.

Disclaimer

Scrysec does not provide security warranties, investment advice, or endorsements of any platform. This audit does not guarantee the security or correctness of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice. The authors are not liable for any decisions made based on the information in this document. Securing smart contracts is an ongoing process. A single audit is not sufficient. We recommend that the platform's development team implement a bug bounty program to encourage further analysis of the smart contract by other third parties

Scrysec

By Auditblock

Scrysec is a blockchain security company that provides professional services and solutions for securing blockchain projects. They specialize in smart contract audits on various blockchains and offer a range of services

100+
Audits Completed

$1B
Secured

300K
Lines of Code Audited

Telegram : @Scrysec

Audit Time
9 January 2024

Audit Report of Azteq