Assure DeFi®

THE VERIFICATION GOLD STANDARD

Security Assessment

ZeussToken

Date: 20/02/2025

Audit Status: FAIL

Audit Edition: Advanced

ASSURE DEFI®

THE VERIFICATION GOLD STANDARD

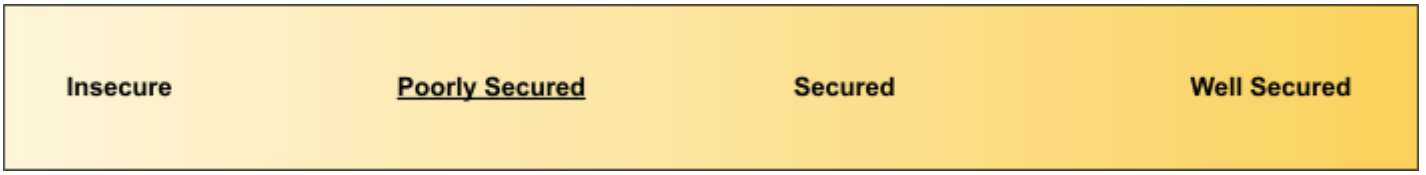
Risk Analysis

Vulnerability summary

Classification	Description
High	High-level vulnerabilities can result in the loss of assets or manipulation of data.
Medium	Medium-level vulnerabilities can be challenging to exploit, but they still have a considerable impact on smart contract execution, such as allowing public access to critical functions.
Low	Low-level vulnerabilities are primarily associated with outdated or unused code snippets that generally do not significantly impact execution, sometimes they can be ignored.
Informational	Informational vulnerabilities, code style violations, and informational statements do not affect smart contract execution and can typically be disregarded.

Executive Summary

According to the Assure assessment, the Customer's smart contract is **Poorly Secured.**



Scope

Target Code And Revision

For this audit, we performed research, investigation, and review of the ZeussToken contracts followed by issue reporting, along with mitigation and remediation instructions outlined in this report.

Target Code And Revision

Project	Assure
Language	Solidity
Codebase	Zeusstoken.sol [SHA256] https://etherscan.io/address/0x5967c987102876e631b5758174b99710141e42ab
Audit Methodology	Static, Manual

Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices.

Category	Item
Code review & Functional Review	• Compiler warnings. • Race conditions and Reentrancy. Cross-function race conditions. • Possible delays in data delivery. • Oracle calls. • Front running. • Timestamp dependence. • Integer Overflow and Underflow. • DoS with Revert. • DoS with block gas limit. • Methods execution permissions. • Economy model. • Private user data leaks. • Malicious Event log. • Scoping and Declarations. • Uninitialized storage pointers. • Arithmetic accuracy. • Design Logic. • Cross-function race conditions. • Safe Zeppelin module. • Fallback function security. • Overpowered functions / Owner privileges

AUDIT OVERVIEW

HIGH

1. Unlimited Minting Capability in the mint() Function

Issue: The current implementation of the mint() function grants the contract owner unrestricted authority to mint new tokens after deployment. The minting power poses significant risks, including:

- Economic Instability:
  Unlimited token creation can trigger rapid inflation, causing the token's value to plummet and undermining the project's long-term economic stability.
- Potential for Abuse:
  The owner could mint tokens arbitrarily for personal gain, opening the door to fraudulent practices and market manipulation, which would destroy investor trust.

Recommendation:
To mitigate these risks, implement one of the following measures:

- Disable Post-Deployment Minting:
  Permanently lock the minting functionality after the initial token distribution to prevent any further token creation.
- Introduce a Capped Supply:
  Set a hard cap on the total token supply, ensuring that minting beyond a predetermined limit is impossible.
- Adopt a Consensus-Based Upgrade Process:
  If additional minting is deemed necessary in the future, require a decentralized, consensus-driven mechanism to authorize any changes to the minting policy.

Implementing these controls will enhance the integrity of the token's economic model and safeguard against potential abuses.

MEDIUM

No medium severity issues were found.

LOW

No low severity issues were found.

INFORMATIONAL

No informational issues were found.

Testing coverage

During the testing phase, custom use cases were written to cover all the logic of contracts. *Check “Annexes” to see the testing code.*

ZeussToken contract tests:

```
tests/test_zeuss_token.py::test_transfer RUNNING
Transaction sent: 0xaf84aedcbe5ef00b6c959d5e5b80d1d28fef878707245ab982774cbe28aac1b5
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  ZeussToken.constructor confirmed   Block: 1   Gas used: 720411 (6.00%)
  ZeussToken deployed at: 0x3194cBDC3dbcd3E11a07892e7bA5c3394048Cc87

Transaction sent: 0xd7c923488c07229c10c9ad3846b67b14a7cfaee34a068b8363a3eed3ccb7f277
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  ZeussToken.transfer confirmed   Block: 2   Gas used: 51098 (0.43%)

Transaction sent: 0xb62af70e319677c6c9a3fa7e229f6e1778fbb5a35b292adba11f3027dbedd26f
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 1
  ZeussToken.transfer confirmed   Block: 3   Gas used: 51086 (0.43%)

Transaction sent: 0xdf5ecba5f6fc78a5abc1fcbcc490ee5c47a8b729b33fdeb883a336a68d3f1ded
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  ZeussToken.transfer confirmed   Block: 4   Gas used: 36086 (0.30%)

tests/test_zeuss_token.py::test_transfer PASSED
tests/test_zeuss_token.py::test_mint RUNNING
Transaction sent: 0xe061c3d9630c3db393393c5579a08e993fd9e76128c32f2a22e8fbab846c0733
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 1
  ZeussToken.constructor confirmed   Block: 5   Gas used: 720411 (6.00%)
  ZeussToken deployed at: 0x602C71e4DAC47a042Ee7f46E0aee17F94A3bA0B6

Transaction sent: 0xb126df4afd953211798ea1251f125dc352bd80ee39145f07657b84856d8c8d46
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 1
  ZeussToken.mint confirmed (reverted)   Block: 6   Gas used: 22969 (0.19%)

Transaction sent: 0xcca501c6f8532874f84a668f98e2b542c5c4388d0f657764276db79a3408c05e
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  ZeussToken.mint confirmed   Block: 7   Gas used: 51762 (0.43%)

Transaction sent: 0xe501a78db29d347cb02b5734bce2c5386df9ea479569d50ae0ce176d4cbaf6b9
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 3
  ZeussToken.mint confirmed   Block: 8   Gas used: 51774 (0.43%)

tests/test_zeuss_token.py::test_mint PASSED
tests/test_zeuss_token.py::test_transfer_from_contract RUNNING
Transaction sent: 0xee0dbba4a12cef110c324313b2d43cffaaf20fd481ad3c7dab13a7461dbb5af0
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  ZeussToken.constructor confirmed   Block: 9   Gas used: 720411 (6.00%)
  ZeussToken deployed at: 0xE7eD6747FaC5360f88a2EFC03E00d25789F69291

Transaction sent: 0xee54f77489383ad84e0bcb75ad661b6542369f8e6b39b90e67d0349378fe4f63
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  ZeussToken.transferFromContract confirmed (reverted)   Block: 10   Gas used: 22903 (0.19%)

Transaction sent: 0xb719d5c8420cbd24f5543973e96ed442dc07cdde878cdf0a00b336e69c85c002
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 4
  ZeussToken.transferFromContract confirmed (Amount must be greater than 0)   Block: 11   Gas used: 22889 (0.19%)

Transaction sent: 0xaf39a939304cc8a14a6d81bf9f387da2fd44786bbafd02253fa841d34aee636d
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 5
  ZeussToken.transfer confirmed   Block: 12   Gas used: 51086 (0.43%)

Transaction sent: 0x946504e2d923e04c40ccb37e8f3e2565572eb73a797d2d326900efe04d54e0c1
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 6
  ZeussToken.transferFromContract confirmed   Block: 13   Gas used: 51884 (0.43%)

Transaction sent: 0x5548102c7b7ef2ee3396d59ef58ef4ce3c0c76b1a1a0453eba30febd13015077
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 7
  ZeussToken.transferFromContract confirmed   Block: 14   Gas used: 36872 (0.31%)

tests/test_zeuss_token.py::test_transfer_from_contract PASSED
```

```
tests/test_zeuss_token.py::test_burn RUNNING
Transaction sent: 0x9cdbc0f55393bf6cc40c26983ea1c3cef09c53a59341dcd940f3b9a984d3931a
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 3
  ZeussToken.constructor confirmed   Block: 15   Gas used: 720411 (6.00%)
  ZeussToken deployed at: 0x6951b5Bd815043E3F842c1b026b0Fa888Cc2DD85

Transaction sent: 0x49ba9042ef208c2aef350709247c2faa350513dee666be28105d33021dae0570
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 8
  ZeussToken.transfer confirmed   Block: 16   Gas used: 51098 (0.43%)

Transaction sent: 0x850b1a5734adaca0462f2b96787dd8fbb285ffde676a41563d32143d5e56ff30
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 9
  ZeussToken.transfer confirmed   Block: 17   Gas used: 51086 (0.43%)

Transaction sent: 0xd2745b344254bf89a88005738e082f577f0cbd8105de293b4c4e9b8e7598573d
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 3
  ZeussToken.burn confirmed   Block: 18   Gas used: 35408 (0.30%)

Transaction sent: 0x89d93d77fa6b01e4e0661abfe404f59627eb00fe667c2f419c8b041472c80431
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  ZeussToken.burn confirmed   Block: 19   Gas used: 35408 (0.30%)

tests/test_zeuss_token.py::test_burn PASSED
```

Annexes

Testing code:

ZeussToken:

```
from brownie import (
    reverts,
)


from scripts.helpful_scripts import (
    ZERO_ADDRESS,
    get_account
)


from scripts.deploy import (
    deploy_zeuss_token
)


def test_transfer(only_local):
    # Arrange
    owner = get_account(0)
    other = get_account(1)
    extra = get_account(2)

    contract_owner = "0x2cc312F73F34BcdADa7d7589CB3074c7Dc06ebE9"

    token = deploy_zeuss_token(owner)

    tx = token.transfer(other, 1e18, {"from": contract_owner})
```

```
    assert tx.events['Transfer'][0]['from'] == contract_owner

    assert tx.events['Transfer'][0]['to'] == other

    assert tx.events['Transfer'][0]['value'] == 1e18


    tx = token.transfer(extra, 5e18, {"from": contract_owner})

    assert tx.events['Transfer'][0]['from'] == contract_owner

    assert tx.events['Transfer'][0]['to'] == extra

    assert tx.events['Transfer'][0]['value'] == 5e18


    tx = token.transfer(extra, 0.5e18, {"from": other})

    assert tx.events['Transfer'][0]['from'] == other

    assert tx.events['Transfer'][0]['to'] == extra

    assert tx.events['Transfer'][0]['value'] == 0.5e18


def test_mint(only_local):

    # Arrange

    owner = get_account(0)

    other = get_account(1)

    extra = get_account(2)


    contract_owner = "0x2cc312F73F34BcdADa7d7589CB3074c7Dc06ebE9"


    token = deploy_zeuss_token(owner)

    with reverts():

        token.mint(other, 5e18, {"from": other})


    token.balanceOf(extra) == 0

    tx = token.mint(extra, 5e18, {"from": contract_owner})
```

```
    assert tx.events['Transfer'][0]['from'] == ZERO_ADDRESS

    assert tx.events['Transfer'][0]['to'] == extra

    assert tx.events['Transfer'][0]['value'] == 5e18

    token.balanceOf(extra) == 5e18


    token.balanceOf(other) == 0

    tx = token.mint(other, 1e18, {"from": contract_owner})

    assert tx.events['Transfer'][0]['from'] == ZERO_ADDRESS

    assert tx.events['Transfer'][0]['to'] == other

    assert tx.events['Transfer'][0]['value'] == 1e18

    token.balanceOf(other) == 5e18


def test_transfer_from_contract(only_local):

    # Arrange

    owner = get_account(0)

    other = get_account(1)

    extra = get_account(2)


    contract_owner = "0x2cc312F73F34BcdADa7d7589CB3074c7Dc06ebE9"


    token = deploy_zeuss_token(owner)

    with reverts():

        token.transferFromContract(other, 5e18, {"from": other})

    with reverts("Amount must be greater than 0"):

        token.transferFromContract(other, 0, {"from": contract_owner})


    token.transfer(token.address, 1e18, {"from": contract_owner})

    tx = token.transferFromContract(other, 0.5e18, {"from": contract_owner})
```

```
    assert tx.events['Transfer'][0]['from'] == token.address

    assert tx.events['Transfer'][0]['to'] == other

    assert tx.events['Transfer'][0]['value'] == 0.5e18


    tx = token.transferFromContract(extra, 0.5e18, {"from": contract_owner})

    assert tx.events['Transfer'][0]['from'] == token.address

    assert tx.events['Transfer'][0]['to'] == extra

    assert tx.events['Transfer'][0]['value'] == 0.5e18


def test_burn(only_local):

    # Arrange

    owner = get_account(0)

    other = get_account(1)

    extra = get_account(2)


    contract_owner = "0x2cc312F73F34BcdADa7d7589CB3074c7Dc06ebE9"


    token = deploy_zeuss_token(owner)


    # transfer some tokens

    token.transfer(other, 1e18, {"from": contract_owner})

    token.transfer(extra, 1e18, {"from": contract_owner})


    tx = token.burn(0.5e18, {"from": other})

    assert tx.events['Transfer'][0]['from'] == other

    assert tx.events['Transfer'][0]['to'] == ZERO_ADDRESS

    assert tx.events['Transfer'][0]['value'] == 0.5e18
```

```
    tx = token.burn(0.1e18, {"from": extra})

    assert tx.events['Transfer'][0]['from'] == extra

    assert tx.events['Transfer'][0]['to'] == ZERO_ADDRESS

    assert tx.events['Transfer'][0]['value'] == 0.1e18
```

Technical Findings Summary

Findings

Vulnerability Level	Total	Pending	Not Apply	Acknowledged	Partially Fixed	Fixed
High	1					
Medium	0					
Low	0					
Informational	0					

Assessment Results

Score Results

Review	Score
Global Score	**70/100**
Assure KYC	Not completed
Audit Score	70/100

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum score is 100, however to attain that value the project must pass and provide all the data needed for the assessment. Our Passing Score has been changed to 84 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below. The Global Score is a combination of the evaluations obtained between having or not having KYC and the type of contract audited together with its manual audit.

Audit Failed

Following our comprehensive security audit of the token contract for the ZeussToken project, the project did not fulfill the necessary criteria required to pass the security audit.

Disclaimer

Assure Defi has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocating for the Project, and users relying on this report should not consider this as having any merit for financial adZeussToken in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment adZeussToken, nor should it be used to signal that any person reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided ‘as is, and Assure Defi is under no covenant to audit completeness, accuracy, or solidity of the contracts. In no event will Assure Defi or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment serZeussTokens provided by Assure Defi are subject to dependencies and are under continuing development. You agree that your access or use, including but not limited to any serZeussTokens, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The serZeussTokens may access, and depend upon, multiple layers of third parties.