# Stargate Smart Contract

# SMART CONTRACT AUDIT REPORT

March 2025

ExVul

www.exvul.com

# Table of Contents


**1. EXECUTIVE SUMMARY ..... 3**
1.1 Methodology ..... 3

**2. FINDINGS OVERVIEW ..... 6**
2.1 Project Info And Contract Address ..... 6
2.2 Summary ..... 6
2.3 Key Findings ..... 7

**3. DETAILED DESCRIPTION OF FINDINGS ..... 8**
3.1 The signature can be replay ..... 8
3.2 The signature can be replay ..... 10
3.3 Precision lost ..... 12
3.4 Initialization Function Lacks Authentication ..... 13

**4. CONCLUSION ..... 15**

**5. APPENDIX ..... 15**
5.1 Basic Coding Assessment ..... 15
*5.1.1 Apply Verification Control ..... 15*
*5.1.2 Authorization Access Control ..... 15*
*5.1.3 Forged Transfer Vulnerability ..... 16*
*5.1.4 Transaction Rollback Attack ..... 16*
*5.1.5 Transaction Block Stuffing Attack ..... 16*
*5.1.6 Soft Fail Attack Assessment ..... 16*
*5.1.7 Hard Fail Attack Assessment ..... 16*
*5.1.8 Abnormal Memo Assessment ..... 16*
*5.1.9 Abnormal Resource Consumption ..... 16*
*5.1.10 Random Number Security ..... 16*
5.2 Advanced Code Scrutiny ..... 17
*5.2.1 Cryptography Security ..... 17*
*5.2.2 Account Permission Control ..... 17*
*5.2.3 Malicious Code Behavior ..... 17*
*5.2.4 Sensitive Information Disclosure ..... 17*
*5.2.5 System API ..... 17*

**6. DISCLAIMER ..... 18**

**7. REFERENCES ..... 19**

# 1. EXECUTIVE SUMMARY

Exvul Web3 Security was engaged by WizzWoods to review smart contract implementation. The assessment was conducted in accordance with our systematic approach to evaluate potential security issues based upon customer requirement. The report provides detailed recommendations to resolve the issue and provide additional suggestions or recommendations for improvement.

The outcome of the assessment outlined in chapter 3 provides the system's owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and detailed recommendations that will resolve the underlying technical issue.

## 1.1 Methodology

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology [10] which is the gold standard in risk assessment using the following risk models:

- Likelihood: represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- Impact: measures the technical loss and business damage of a successful attack.
- Severity: determine the overall criticality of the risk.

Likelihood can be: High, Medium and Low and impact are categorized into for: High, Medium, Low, Informational. Severity is determined by likelihood and impact and can be classified into five categories accordingly, Critical, High, Medium, Low, Informational shown in table 1.1.

| Likelihood \ IMPACT | Informational | Low | Medium | High |
|---|---|---|---|---|
| High | Informational | Medium | High | Critical |
| Medium | Informational | Low | Medium | High |
| Low | Informational | Low | Low | Medium |

*Table 1.1 Overall Risk Severity*

To evaluate the risk, we will be going through a list of items, and each would be labelled with a severity category. The audit was performed with a systematic approach guided by a comprehensive assessment list carefully designed to identify known and impactful security issues. If our tool or analysis does not identify any issue, the contract can be considered safe regarding the assessed item.

For any discovered issue, we might further deploy contracts on our private test environment and run tests to confirm the findings. If necessary, we would additionally build a PoC to demonstrate the possibility of exploitation. The concrete list of check items is shown in Table 1.2.

- Basic Coding Bugs: We first statically analyze given smart contracts with our proprietary static code analyzer for known coding bugs, and then manually verify (reject or confirm) all the issues found by our tool.
- Code and business security testing: We further review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.
- Additional Recommendations: We also provide additional suggestions regarding the coding and development of smart contracts from the perspective of proven programming practices.

| Category | Assessment Item |
| --- | --- |
| **Basic Coding Assessment** | Apply Verification Control |
| | Authorization Access Control |
| | Forged Transfer Vulnerability |
| | Forged Transfer Notification |
| | Numeric Overflow |
| | Transaction Rollback Attack |
| | Transaction Block Stuffing Attack |
| | Soft Fail Attack |
| | Hard Fail Attack |
| | Abnormal Memo |
| | Abnormal Resource Consumption |
| | Secure Random Number |
| **Advanced Source Code Scrutiny** | Asset Security |
| | Cryptography Security |
| | Business Logic Review |
| | Source Code Functional Verification |
| | Account Authorization Control |
| | Sensitive Information Disclosure |
| | Circuit Breaker |
| | Blacklist Control |
| | System API Call Analysis |
| | Contract Deployment Consistency Check |

| Category | Assessment Item |
| --- | --- |
| **Additional Recommendations** | Semantic Consistency Checks |
| | Following Other Best Practices |

*Table 1.2: The Full List of Assessment Items*

To better describe each issue we identified, we categorize the findings with Common Weakness Enumeration (CWE-699) [14], which is a community-developed list of software weakness types to better delineate and organize weaknesses around concepts frequently encountered in software development.

# 2. FINDINGS OVERVIEW

## 2.1 Project Info And Contract Address

Project Name: Stargate

Audit Time: March 10, 2025 – March 13, 2025

Language: Solidity

| File Name | Link |
|---|---|
| Stargate | https://github.com/YouNeedWork/fortuna/ |
| Commit hash | 98c6d66f788d23d1e3677e9e5c042cf3d64a27b1 |

## 2.2 Summary

| Severity | Found | |
|---|---|---|
| Critical | 2 | |
| High | 2 | |
| Medium | 0 | |
| Low | 0 | |
| Informational | 0 | |

## 2.3 Key Findings

| ID | Severity | Findings Title | Status | Confirm |
|---|---|---|---|---|
| NVE-001 | Critical | Signature Vulnerability to Replay Attacks | Fixed | Confirmed |
| NVE-002 | Critical | Signature Vulnerability to Replay Attacks | Fixed | Confirmed |
| NVE-003 | High | Precision lost | Fixed | Confirmed |
| NVE-004 | High | Initialization Function Lacks Authentication | Fixed | Confirmed |

*Table 2.3: Key Audit Findings*

# 3. DETAILED DESCRIPTION OF FINDINGS

## 3.1 The signature can be replay

| | | | |
|---|---|---|---|
| **ID:** | NVE-001 | **Location:** | |
| **Severity:** | Critical | **Category:** | Business Issues |
| **Likelihood:** | High | **Impact:** | High |

**Description:**

The signature is potentially susceptible to replay attacks due to the incorrect placement or handling of the nonce. This issue may allow unauthorized replay of the signature across different transactions, posing a security risk to the protocol. Proper nonce management should be enforced to ensure each signature is uniquely tied to a specific transaction.

```
pub fn claim(
    ctx: Context<Claim>,
    amount: u64,
    deposit_hash: String,
    timestamp: u64,
    out_trade_no: String,
    signature: [u8; 64],
) -> Result<()> {
    let config = &mut ctx.accounts.config.load()?;
    if amount < config.minimum_deposit {
        return Err(ErrorCode::InvalidAmount.into());
    }

    if amount > ctx.accounts.pool_token_account.amount {
        return Err(ErrorCode::InsufficientBalance.into());
    }

    let current_timestamp = Clock::get()?.unix_timestamp as u64;
    if (timestamp) < current_timestamp {
        return Err(ErrorCode::InvalidTimestamp.into());
    }

    let nonce = &mut ctx.accounts.nonce_account;
    nonce.nonce += 1;

    msg!(
        "msg params {} {} {} {}",
        amount,
        deposit_hash,
        timestamp,
        nonce.nonce
    );
```

**Recommendations:**

Exvul Web3 Security recommends placing the nonce in an appropriate position.

ic transaction.

```
pub fn claim(
    ctx: Context<Claim>,
    amount: u64,
    deposit_hash: String,
    timestamp: u64,
    out_trade_no: String,
    signature: [u8; 64],
) -> Result<()> {
    let config = &mut ctx.accounts.config.load()?;
    let nonce = &mut ctx.accounts.nonce_account;
    nonce.nonce += 1;

    if amount < config.minimum_deposit {
        return Err(ErrorCode::InvalidAmount.into());
    }

    if amount > ctx.accounts.pool_token_account.amount {
        return Err(ErrorCode::InsufficientBalance.into());
    }

    let current_timestamp = Clock::get()?.unix_timestamp as u64;
    if (timestamp) < current_timestamp {
        return Err(ErrorCode::InvalidTimestamp.into());
    }

    msg!(
        "msg params {} {} {} {}",
        amount,
        deposit_hash,
        timestamp,
        nonce.nonce
    );
```

**Result:** Fixed

## 3.2 The signature can be replay

| ID: | NVE-002 | Location: | |
|---|---|---|---|
| Severity: | Critical | Category: | Business Issues |
| Likelihood: | High | Impact: | High |

**Description:**

The signature is potentially susceptible to replay attacks due to the incorrect placement or handling of the nonce. This issue may allow unauthorized replay of the signature across different transactions, posing a security risk to the protocol. Proper nonce management should be enforced to ensure each signature is uniquely tied to a specific transaction.

```
pub fn transfer_fee(
    ctx: Context<TransferFee>,
    amount: u64,
    timestamp: u64,
    out_trade_no: String,
    signature: [u8; 64],
) -> Result<()> {
    let mut config = &mut ctx.accounts.config.load_mut()?;

    let maximum_transfer = ctx.accounts.pool_token_account.amount * 5 / 1
00;
    if amount > maximum_transfer {
        return Err(ErrorCode::InvalidAmount.into());
    }

    let current_timestamp = Clock::get()?.unix_timestamp as u64;
    if timestamp < current_timestamp {
        return Err(ErrorCode::InvalidTimestamp.into());
    }

    config.nonce += 1;

    msg!(
        "msg params {} {} {} {}",
        amount,
        out_trade_no,
        timestamp,
        config.nonce
    );
```

**Recommendations:**

Exvul Web3 Security recommends placing the nonce in an appropriate position.

```
pub fn transfer_fee(
    ctx: Context<TransferFee>,
    amount: u64,
    timestamp: u64,
    out_trade_no: String,
    signature: [u8; 64],
) -> Result<()> {
    let mut config = &mut ctx.accounts.config.load_mut()?;
    config.nonce += 1;
    let maximum_transfer = ctx.accounts.pool_token_account.amount * 5 / 1
00;
    if amount > maximum_transfer {
        return Err(ErrorCode::InvalidAmount.into());
    }

    let current_timestamp = Clock::get()?.unix_timestamp as u64;
    if timestamp < current_timestamp {
        return Err(ErrorCode::InvalidTimestamp.into());
    }



    msg!(
        "msg params {} {} {} {}",
        amount,
        out_trade_no,
        timestamp,
        config.nonce
    );
```

**Result:** Fixed

## 3.3 Precision lost

| ID: | NVE-003 | Location: | |
|---|---|---|---|
| Severity: | High | Category: | Business Issues |
| Likelihood: | High | Impact: | Low |

**Description:**

Precision loss caused by division operations can potentially result in financial discrepancies or loss of funds.

```
let fundation_amount = amount * 1 / 100;

utils::transfer_token_from_pool(
    &ctx.accounts.pool_token_account,
    ctx.accounts.payer_token_account.to_account_info(),
    ctx.accounts.mint.to_account_info(),
    amount - fundation_amount,
    &ctx.accounts.token_program,
    &ctx.accounts.global_account,
    ctx.accounts.mint.decimals,
    ctx.bumps.global_account,
)?;

utils::transfer_token_from_pool(
    &ctx.accounts.pool_token_account,
    ctx.accounts.fundation_token_account.to_account_info(),
    ctx.accounts.mint.to_account_info(),
    fundation_amount,
    &ctx.accounts.token_program,
    &ctx.accounts.global_account,
    ctx.accounts.mint.decimals,
    ctx.bumps.global_account,
)?;
```

**Recommendations:**

Exvul Web3 Security recommends implementing a "multiply first, then divide" approach to minimize precision loss caused by division operations.

**Result:** Fixed

## 3.4 Initialization Function Lacks Authentication

| | | | |
|---|---|---|---|
| **ID:** | NVE-004 | **Location:** | |
| **Severity:** | High | **Category:** | Business Issues |
| **Likelihood:** | High | **Impact:** | High |

**Description:**

The initialization function lacks proper authorization mechanisms, creating a vulnerability during contract deployment. This could result in the function being front-run by malicious actors, potentially allowing them to insert harmful configuration files.

```
use anchor_lang::prelude::*;

#[derive(Accounts)]
pub struct Initialize<'info> {
    #[account(mut)]
    pub payer: Signer<'info>,

    #[account(
        init,
        payer = payer,
        space = 8 + std::mem::size_of::<AccountInfo>(),
        seeds = ["GLOBAL".as_bytes()],
        bump,
    )]
    /// CHECK: Use as Seed
    pub global_account: AccountInfo<'info>,

    #[account(
        init,
        payer = payer,
        space = 8 + std::mem::size_of::<Config>() ,
        seeds = ["CONFIG".as_bytes()],
        bump,
    )]
    pub config: AccountLoader<'info, Config>,

    pub system_program: Program<'info, System>,
}
```

**Recommendations:**

Exvul Web3 Security recommends enhancing access control measures on the initialization function.

```
pub struct Initialize<'info> {
    #[account(
        mut,
        address = crate::admin::id() @ ErrorCode::NotApproved
        )]
    pub payer: Signer<'info>,
```

**Result:** Fixed

# 4. CONCLUSION

In this audit, we thoroughly analyzed **Stargate** smart contract implementation. The problems found are described and explained in detail in Section 3. The problems found in the audit have been communicated to the project leader. We therefore consider the audit result to be **PASSED**. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

# 5. APPENDIX

## 5.1 Basic Coding Assessment

### 5.1.1 Apply Verification Control

- Description: The security of apply verification
- Result: Not found
- Severity: Critical

### 5.1.2 Authorization Access Control

- Description: Permission checks for external integral functions
- Result: Not found
- Severity: Critical

### 5.1.3 Forged Transfer Vulnerability

- Description: Assess whether there is a forged transfer notification vulnerability in the contract
- Result: Not found
- Severity: Critical

### 5.1.4 Transaction Rollback Attack

- Description: Assess whether there is transaction rollback attack vulnerability in the contract.
- Result: Not found
- Severity: Critical

### 5.1.5 Transaction Block Stuffing Attack

- Description: Assess whether there is transaction blocking attack vulnerability.
- Result: Not found
- Severity: Critical

### 5.1.6 Soft Fail Attack Assessment

- Description: Assess whether there is soft fail attack vulnerability.
- Result: Not found
- Severity: Critical

### 5.1.7 Hard Fail Attack Assessment

- Description: Examine for hard fail attack vulnerability
- Result: Not found
- Severity: Critical

### 5.1.8 Abnormal Memo Assessment

- Description: Assess whether there is abnormal memo vulnerability in the contract.
- Result: Not found
- Severity: Critical

### 5.1.9 Abnormal Resource Consumption

- Description: Examine whether abnormal resource consumption in contract processing.
- Result: Not found
- Severity: Critical

### 5.1.10 Random Number Security

- Description: Examine whether the code uses insecure random number.
- Result: Not found
- Severity: Critical

## 5.2 Advanced Code Scrutiny

### 5.2.1 Cryptography Security

- Description: Examine for weakness in cryptograph implementation.
- Results: Not Found
- Severity: High

### 5.2.2 Account Permission Control

- Description: Examine permission control issue in the contract
- Results: Not Found
- Severity: Medium

### 5.2.3 Malicious Code Behavior

- Description: Examine whether sensitive behavior present in the code
- Results: Not found
- Severity: Medium

### 5.2.4 Sensitive Information Disclosure

- Description: Examine whether sensitive information disclosure issue present in the code.
- Result: Not found
- Severity: Medium

### 5.2.5 System API

- Description: Examine whether system API application issue present in the code
- Results: Not found
- Severity: Low

## 6. DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without ExVul's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts ExVul to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. ExVul's position is that each company and individual are responsible for their own due diligence and continuous security. ExVul's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

# 7. REFERENCES

[1] MITRE. CWE- 191: Integer Underflow (Wrap or Wraparound).

https://cwe.mitre.org/data/ definitions/191.html.

[2] MITRE. CWE- 197: Numeric Truncation Error.

https://cwe.mitre.org/data/definitions/197. html.

[3] MITRE. CWE-400: Uncontrolled Resource Consumption.

https://cwe.mitre.org/data/ definitions/400.html.

[4] MITRE. CWE-440: Expected Behavior Violation.

https://cwe.mitre.org/data/definitions/440. html.

[5] MITRE. CWE-684: Protection Mechanism Failure.

https://cwe.mitre.org/data/definitions/ 693.html.

[6] MITRE. CWE CATEGORY: 7PK - Security Features.

https://cwe.mitre.org/data/definitions/ 254.html.

[7] MITRE. CWE CATEGORY: Behavioral Problems.

https://cwe.mitre.org/data/definitions/438. html.

[8] MITRE. CWE CATEGORY: Numeric Errors.

https://cwe.mitre.org/data/definitions/189.html.

[9] MITRE. CWE CATEGORY: Resource Management Errors.

https://cwe.mitre.org/data/ definitions/399.html.

[10] OWASP. Risk Rating Methodology.

https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

www.exvul.com

contact@exvul.com

@EXVULSEC

github.com/EXVUL-Sec

ExVul