BitGenie Smart Contract

April 2024

SMART CONTRACT AUDIT REPORT

ExVul

www.exvul.com

Table of Contents

1. EXECUTIVE SUMMARY

Exvul Web3 Security was engaged by BitGenie 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.

High risk finding is primarily related to pause ,staking end time set, create address takeover.

Medium risk finding is primarily related to fee-on-transfer token and staing logic.

Informational risk finding is primarily related to the mulsig.

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

Category	Assessment Item
	Contract Deployment Consistency Check
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: BitGenie

Audit Time: April13th, 2024 – April22th, 2024

Language: solidity

File Name	code
Staking.sol	https://github.com/Async-Finance/Staking/blob/f98fc618f3de00775561678e0b81428358ade8cd/contracts/Staking.sol
Swap	BitGenie Swap And Factory Code

2.2 Summary

Severity	Found	
Critical	0	
High	3	
Medium	3	
Low	0	
Informational	1	

2.3 Key Findings

High risk finding is primarily related to pause ,staking end time set, create address takeover.

Medium risk finding is primarily related to fee-on-transfer token and staing logic.

Informational risk finding is primarily related to the mulsig.

ID	Severity	Findings Title	Status	Confirm
NVE-001	High	pause/unpause functionalities not implemented in pausable contracts	Fixed	Confirmed
NVE-002	High	User are possibly to lose some part of their profits if owner expands the end time	Fixed	Confirmed
NVE-003	High	Malicious user could front-run to takeover pair address	Fixed	Confirmed
NVE-	Medium	User can stake before contract staking starts	Fixed	Confirmed

ID	Severity	Findings Title	Status	Confirm
004				
NVE-005	Medium	contract don't support fee-on-transfer token	Fixed	Confirmed
NVE-006	Medium	Unstake shouldn’t be able to paused.	Fixed	Confirmed
NVE-007	Informational	Use multisig to set fee related config	Fixed	Confirmed

Table 2.1: Key Audit Findings

3. DETAILED DESCRIPTION OF FINDINGS

3.1 pause/unpause functionalities not implemented in pausable contracts

ID:	NVE-001	Location:	Staking.sol
Severity:	High	Category:	Business Issues
Likelihood:	Medium	Impact:	High

Description:

In contract function `stake`, `unstake`, `redeem`,we use modifier `whenNotPaused` from OZ contract

In Oz `Pausable,sol`, `_pause` and `_unpause` those functions are internal, the contract must implement two other public/external pause and unpause functions to allow the manager to pause and unpause the contracts when necessary. None of the function implement those functions, which means even if those contracts are supposed to be pausable (and have the pause/unpause functionalities), none of them can be paused.

Recommendations:

ExVul Web3 Labs recommends adding a pause and unpause function

```
9     /**
10    * @dev Triggers stopped state.
11    * Contract must not be paused
12    */
13   function pause() external onlyOwner {
14
15       _pause();
16   }
17
18   /**
19    * @dev Returns to normal state.
20    * Contract must be paused
21    */
22   function unpause() external onlyOwner {
23       _unpause();
24   }
25
```

Result: Confirmed

Fix Result: Fixed

```
function pause() external onlyOwner {
    _pause();
}
```

3.2 User are possibly to lose some part of their profits if owner expands the end time

ID:	NVE-002	Location:	Staking.sol
Severity:	High	Category:	Business Issues
Likelihood:	Medium	Impact:	High

Description:

Every time update, the `lastUpdateTime` will set time the `block.timestamp`,

we have function `setEndTime ` for owners to update end time.

Suppose one situation:

1. staking period endtime is 21th

2. alice redeem(or other function to call function update) at 22th, so the `lastUpdateTime` of `alice` is 22th.

3. owner want to expand the staking period, and set end time to 25th.

4. alice will lost the award from21th-22t

```
modifier update(address account) {
    reward[account] = available(account);
    lastUpdateTime[account] = block.timestamp;
    _;
}
```

Recommendations:

ExVul Web3 Labs recommends limit the updatetime max to endtime

```
modifier update(address account) {
    reward[account] = available(account);
    uint256 updateTime = Math.min(block.timestamp, endTime);
    lastUpdateTime[account] = block.timestamp;
    _;
}
```

Result: Confirmed

Fix Result: Fixed

We deleted setEndtime function.

3.3 Malicious user could front-run to takeover pair address

ID:	NVE-001	Location:	UniSwapV2Factory.sol
Severity:	High	Category:	Business Issues
Likelihood:	High	Impact:	High

Description:

the system itself may be fine but it could be a problem for integrations.

Lets say a normal user wants to create pair tokenA - tokenB , the expected pair address is pairAB

now a malicious user front-runs create a pair tokenX-tokenY , and the created pair address is pairAB ==> takeover the address that was about to pair tokenA-tokenB.

The original univ2 solves this problem by using create2 with unique salt ==> no duplicated pool address and no front-run to takeover the address

```
function createPair(
    address tokenA,
    address tokenB
) external override returns (address pair) {
    require(tokenA != tokenB, "UniswapV2: IDENTICAL_ADDRESSES");
    (address token0, address token1) = tokenA < tokenB
        ? (tokenA, tokenB)
        : (tokenB, tokenA);
    require(token0 != address(0), "UniswapV2: ZERO_ADDRESS");
    require(
        getPair[token0][token1] == address(0),
        "UniswapV2: PAIR_EXISTS"
    ); // single check is sufficient
    pair = address(new UniswapV2Pair(token0, token1));
    getPair[token0][token1] = pair;
    getPair[token1][token0] = pair; // populate mapping in the reverse direction
    allPairs.push(pair);
    emit PairCreated(token0, token1, pair, allPairs.length);
}
```

Figure 3.1.1 UniSwapV2Factory.sol

Recommendations:

ExVul Web3 Labs recommends use create2 to avoid pair address different with expect.
Result: Fixed
Sponser have use create2 to avoid this situation

3.4 User can stake before contract staking starts

ID:	NVE-003	Location:	Staking.sol
Severity:	Medium	Category:	Business Issues
Likelihood:	Medium	Impact:	High

Description:

There is no limit in stake that limit user should stake token before token start,during the time before start, user will not make any profit from this.

```
function stake(uint256 amount) external update(_msgSender()) nonReentrant whenNotPaused {
    require(block.timestamp < endTime, "Cannot stake. Stake is over");
    require(IERC20(stakeToken).allowance(_msgSender(), address(this)) >= amount, "StakingFund doesn't have enough allowance");
    require(amount + stakedValue <= hardCap, "Exceeds maximum stake amount");
    stakedValue += amount;
    IERC20(stakeToken).safeTransferFrom(_msgSender(), address(this), amount);
```

Recommendations:

ExVul Web3 Labs recommends add staking start check when user try to stake.

Result: Confirmed

Fix Result: fixed

Sponser:Staking before start time is allowed.

3.5 contract don't support fee-on-transfer token

ID:	NVE-004	Location:	Staking.sol
Severity:	Medium	Category:	Business Issues
Likelihood:	Medium	Impact:	High

Description:

As shown in the figure below in `stake`, the `balanceof` add the amount same as transefer amount,the problem is this will not work with fee-on-tranfer tokens, staking contract will get less token comparing to the transfer amount, so there will be some accounting problem in staking logic.

```
function stake(uint256 amount) external update(_msgSender()) nonReentrant whenNotPaused {
    require(block.timestamp < endTime, "Cannot stake. Stake is over");
    require(IERC20(stakeToken).allowance(_msgSender(), address(this)) >= amount, "StakingFund doesn't have enough allowance");
    require(amount + stakedValue <= hardCap, "Exceeds maximum stake amount");
    stakedValue += amount;
    IERC20(stakeToken).safeTransferFrom(_msgSender(), address(this), amount);
    balanceOf[_msgSender()] += amount;
```

Recommendations:

ExVul Web3 Labs recommends add support on fee-on-transfer token

Some example of support this type of token:

```
function _depositFunds(address _from, IERC20 _token, uint256 _amount) internal returns (uint256) {
    //:   fee on transfer support
    uint256 balanceBefore = _token.balanceOf(address(this));
    _token.safeTransferFrom(_from, address(this), _amount);
    uint256 balanceAfter = _token.balanceOf(address(this));

    return balanceAfter - balanceBefore;
}
```

Result: Confirmed

Fix Result: fixed

Sponser: contract don't use fee-on-transfer token.

3.6 Unstake shouldn't be able to paused.

ID:	NVE-005	**Location:**	Staking.sol
Severity:	Medium	**Category:**	Business Issues
Likelihood:	Medium	**Impact:**	Low

Description:

As shown in the figure below, there is a pause limit in unstake function, the problem is using the limit, user can not unstake their token if owner set contract pause, this will some kind of harmful to Project's reputation.

```
function unstake(uint256 amount) external update(_msgSender()) nonReentrant whenNotPaused {
    require(amount <= balanceOf[_msgSender()], "Exceeds balance");
    balanceOf[_msgSender()] -= amount;
    stakedValue -= amount;
    IERC20(stakeToken).safeTransfer(_msgSender(), amount);
    emit UnStakeEvent(_msgSender(), amount);
}
```

Recommendations:

ExVul Web3 Labs recommends remove this puase limt, user should be free to withdraw their assets at anytime,

Result: Confirmed

Fix Result: fixed

We have deleted the limit

```
function unstake(uint256 amount) external update(_msgSender()) nonReentrant {
    require(amount <= balanceOf[_msgSender()], "Exceeds balance");
    balanceOf[_msgSender()] -= amount;
    stakedValue -= amount;
    IERC20(stakeToken).safeTransfer(_msgSender(), amount);
    emit UnStakeEvent(_msgSender(), amount);
}
```

3.7 Use multisig to set fee related config

ID:	NVE-002	**Location:**	UniSwapV2Factory.sol
Severity:	Info	**Category:**	Business Issues
Likelihood:	Info	**Impact:**	Info

Description:

As shown in the figure below, the function `seeFeetTo()` allows owner to set the fee receiver, it's a important system config , once private key is leaked , whole system may be compromised, so we recommends use multisig to set important sys config.

```
function setFeeTo(address _feeTo) external override {
    require(msg.sender == feeToSetter, "UniswapV2: FORBIDDEN");
    feeTo = _feeTo;
}

function setFeeToSetter(address _feeToSetter) external override {
    require(msg.sender == feeToSetter, "UniswapV2: FORBIDDEN");
    feeToSetter = _feeToSetter;
}
```

Figure 3.1.1 UniSwapV2Factory.sol

Recommendations:

ExVul Web3 Labs recommends use multisig to set fee related config.

Result: Fixed

Sponser has known this and will try to avoid this situation.

4. CONCLUSION

In this audit, we thoroughly analyzed BitGenie 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