Audit Report Review

Version 1.0

Roqbell

January 19, 2024

Retik-Finance Audit Report Review

Roqbell

January 19, 2024

Prepared by: Roqbell Lead Security Researchers:

- xxxxxxx

Table of Contents

* [Min-01] Logical Issue: the change of `Retik_Finance::Owner` by function `Retik_Finance::transferOwnership` overrides a previously set `owner` with the new one not guarateeing whether it has the ability to actuate on-cain transactions
* [Info-01] Coding Style: Incomplete Event Emission for Sensitive Functions possible cause of ambiguity
* [Info-02] Observation Regarding Bot Identification and Transaction Sequencing
* [Info-03] Unused Event Declaration

Protocol Summary

The `Retik Finance` is a trailblazer DeFi utilizing open-source protocols and rapid product development platforms to construct crypto-fiat bridging systems. The focus of this audit is the token contract.

Disclaimer

The Sec Research team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
Likelihood	High	H	H/M	M
	Medium	H/M	M	M/L
	Low	M	M/L	L

Audit Details

The finding described in this document correspond with the following commit hash:

```
- Commit Hash:  0x26ebb8213fb8d66156f1af8908d43f7e3e367c1d
```

Scope

```
./src/
#-- Retik_Finance.sol
```

External Dependencies

The folowing are external addresses used within the contracts:

Megalink.sol

- *_owner*: owner account of the contract
- *dexPair*: the address of the DEX pair.

Executive Summary

How the audit went?, Type of thigs you found, etc. We spent X hours with Y auditors using Z tools. etc

Privileged Functions

In the `Retik` project, multiple roles are adopted to ensure the dynamic runtime updates of the project, which were specifiedin the findings RFH-01.The advantage of this privileged role in the codebase is that the client reserves the ability to adjust the protocol according tothe runtime required to best serve the community. It is also worth of note the potential drawbacks of these functions, whichshould be clearly stated through the client's action/plan. Additionally, if the private key of the privileged account iscompromised, it could lead to devastating consequences for the project.To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community.Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the `Timelock` contract

Issues found

Severity	Numbers of issues found
Critical	0
Major	2
Medium	0
Minor	1
Info	3
Total	6

Findings

[Maj-01] Centralization Risks, the owner has authority over major functions in the contract, any compromise to this account can cause an hacker to take authority and wreck the protocol

Description: In the contract **Ownable** the role *_owner* has authority over the functions shown in the diagram below. Any compromise tothe *_owner* account may allow the hacker to take advantage of this authority to transfer ownership In the contract `Retik_Finance` the role _owner has authority over `Retik_Finance::transferOwnership`, `Retik_Finance::renounceOwnership` and the state variables. Any compromise to the *_owner* account may allow the hacker to take advantage of this authority to interfere with governance processes.

Recommendation: The risk describes the current project design and potentially makes iterations to improve in the security operation and level ofdecentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefullymanage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommendcentralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accountswith enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that wouldalso mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

- **Short Term:** Timelock and Multi sign (2/3, 3/5) combination mitigate by delaying the sensitive operation and avoiding a single point of keymanagement failure
  - Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;AND
  - Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private keycompromised; AND

  - A medium/blog link for sharing the timelock contract and multi-signers addresses information with the publicaudience

- **Long Term:** Timelock and DAO, the combination, mitigate by applying decentralization and transparency

  - Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations; AND
  - Introduction of a DAO/governance/voting module to increase transparency and user involvement. AND
  - A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

- **Permanent:** Renouncing the ownership or removing the function can be considered fully resolved

  - Renounce the ownership and never claim back the privileged roles.OR
  - Remove the risky functionality.

Mitigation:

- **[Retik 1/9/2024]:** We will lock most of the tokens according to our tokenomics plan. 40% will be sent to the presale contract.
- **[CertiK 1/9/2024]:** It is suggested to implement the aforementioned methods to avoid centralized failure. Also, it stronglyencourages the project team to periodically revisit the private key security management of all addresses related tocentralized roles.

[Maj-02] Centralization Issue, `RETIK` tokens were sent to the owner upon deployment causing the deployer/owner to have absolute authority to sell or distribute the tokens without community consensus and any compromise to this particular account poses a severe damage to the project

Description: `1_000_000_000 * 1e18` `RETIK` tokens were sent to the owner when deploying the contract. This is a centralization riskbecause the deployer or the owner(s) of the EOAs can distribute tokens without obtaining the consensus of the community.Any compromise to these addresses may allow a hacker to steal and sell tokens on the market, resulting in severe damageto the project

Impact: This is a serious cenralization issue and can lead to a great loss for the project.

Recommendation: It is recommended that the team be transparent regarding the initial token distribution process. The token distribution planshould be published in a public location that the community can access. The team should make efforts to restrict access tothe private keys of the deployer account

or EOAs. A multi-signature (2/3, 3/5) wallet can be used to prevent a single point offailure due to a private key compromise. Additionally, the team can lock up a portion of tokens, release them with a vestingschedule for long-term success, and deanonymize the project team with a third-party KYC provider to create greater accountability.

Mitigation:

- **[Retik 1/9/2024]:** We will lock most of the tokens according to our tokenomics plan. 40% will be sent to the presale contract.

[Min-01] Logical Issue: the change of `Retik_Finance::Owner` by function `Retik_Finance::transferOwnership` overrides a previously set `owner` with the new one not guarateeing whether it has the ability to actuate on-cain transactions

Description: The change of `Retik_Finance::Owner` by function `Retik_Finance::transferOwnership` overrides the previously set `Retik_Finance::Owner` with the new one without guaranteeing the new owner has the ability to actuate transactions on-chain.

Impact: This presents a critical logical flaw within the contract. In the event that the new owner lacks the capability to execute on-chain transactions, it could render the protocol nearly incapacitated, emphasizing the urgency of resolution.

Recommendation: Refactoring te linked codes below is strongly advised:

PoC

Consider refactoring

```
1      69 address public pendingOwner;
2      70
3      71 function renounceOwnership() public onlyOwner {
4      72      _owner = address(0);
5      73      pendingOwner = address(0);
6      74      emit OwnershipTransferred(_owner, address(0));
7      75  }
8      76
9      77 function transferOwnership(address newOwner) public onlyOwner {
10     78      require(address(0) != newOwner, "pendingOwner set to the
           zero address.");
11     79      pendingOwner = newOwner;
12     80 }
13     81
14     82 function claimOwnership() public {
15     83      require(msg.sender == pendingOwner, "caller != pending
           owner");
16     84
17     85      _owner = pendingOwner;86 pendingOwner = address(0);
```

```
87         emit OwnershipTransferred(_owner, pendingOwner);
88     }
```

Mitigation:

- **[CertiK 1/9/2024]:** The team acknowledged this issue and decided not to change the codebase this time.

[Info-01] Coding Style: Incomplete Event Emission for Sensitive Functions possible cause of ambiguity

Description: It is crucial to ensure comprehensive event emission within sensitive functions governed by centralized roles. The current code lacks the necessary events to convey crucial information and updates effectively. This gap in event emission needs to be addressed to enhance traceability and maintain proper oversight in an audit context.

Impact: This deficiency hampers traceability, hindering the ability to monitor crucial information and updates effectively, this gap undermines the overall transparency and oversight, potentially leading to oversight lapses and compromising the integrity of the codebase. Immediate attention is required to mitigate these risks and uphold the robustness of the coding practices.

Recommendation: To enhance transparency and accountability within the smart contract, it is advised to implement event emission within functions that involve critical operations controlled by centralized roles. Emitting events in such scenarios provides a clear and auditable record of state changes, ensuring that stakeholders and external observers can monitor and verify the execution of sensitive functionalities.

This practice not only facilitates a more comprehensive understanding of contract behavior but also contributes to improved transparency and adherence to best practices in smart contract development.

Mitigation:

- **[CertiK 1/9/2024]:** The team acknowledged this issue and decided not to change the codebase this time.

[Info-02] Observation Regarding Bot Identification and Transaction Sequencing

Description: The current Anti-Bot mechanism exhibits a limitation in its ability to prevent the initial entry of bots. Specifically, in lines 292 to 296 of the code, the system identifies and flags “from” and “to” addresses as bots. However, due to the sequence of operations, the check for bot status occurs

earlier in the transaction process. This ordering may lead to a scenario where, even if the “from” or “to” address is identified as a bot during the ongoing transaction, the transaction can proceed without impediment:

PoC

Consider refactoring

```
    283 require(!isBot[from], "RETIK: Bot detected");
    284
    285 if (from != owner() && to != owner()) {
    286     // trading disable till launch
    287     require(trading, "RETIK: trading is disable");
    288     // antibot
    289     if (
    290         block.timestamp < launchedAt + snipingTime291 ) {
    292         if (dexPair == from) {
    293             isBot[to] = true;
    294         } else if (dexPair == to) {
    295                 isBot[from] = true;
    296         }
    297     }
    298 }
```

</details

The question we want to discuss is whether we need to penalize bots or block transactions the first time they enter.

Impact: Impact: The current sequencing issue in the Anti-Bot mechanism allows transactions involving flagged bot addresses to proceed unaffected, compromising the system’s intended security and functionality.

Recommendation: We recommend reviewing the logic again and ensuring it is as intended.

Mitigation:

- **[Retik 1/9/2024]:** We are looking to penalize the bots

[Info-03] Unused Event Declaration

Description: The SwapAndLiquify event is declared in the code but remains unused. This event is not leveraged in the existing logic

Recommendation: Removal of the unused SwapAndLiquify event is suggested, as it is not actively employed..

This practice not only facilitates a more comprehensive understanding of contract behavior but also contributes to improved transparency and adherence to best practices in smart contract development.

Mitigation:

- **[CertiK 1/9/2024]:** The team acknowledged this issue and decided not to change the codebase this time.