SECURITY AUDIT OF

NEXTON SMART CONTRACTS

Public Report

May 22, 2024

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
TON	The TON network, or Telegram Open Network, was a blockchain platform developed by the messaging app Telegram, aiming to provide a decentralized infrastructure for various services and applications.
FunC	FunC is a low-level, statically-typed programming language specifically designed for writing smart contracts on the TON (The Open Network) blockchain. It offers fine-grained control over contract execution and optimization, catering to developers who need efficient and secure contract implementations.
Tact	Tact is a programming language designed for writing smart contracts. It aims to provide a more developer-friendly environment, enhancing security and efficiency compared to previous languages. Tact simplifies the creation and management of smart contracts on the TON blockchain, making the process more accessible for developers.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on May 22, 2024. We would like to thank the NexTon for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the NexTon Smart Contracts. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified some vulnerable issue in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About NexTon Smart Contracts

Nexton is a yield optimizer based on liquid staking that offers liquidity for staked assets. It appeals to large TON holders who are not active in Defi protocols by providing them with a secure way to earn maximum staking profits. Additionally, it guarantees LPs a maximum of 40% APY, making it an attractive option for entering the TON ecosystem.

Simply put, LPs provide liquidity to enable nominators to execute a feature for maximizing staking rewards while LPs receive incentives, on the other side, nominators are going to earn optimum rewards while they retain NFT during the lock-up period. Under the hood, this protocol is connected to the TON nominator pool. It means all of the provided liquidity and the requested amount of staking assets are staked, then each stakeholder could optimize profits. Furthermore, it would be connected to existing staking-related protocols as long as we could synergize with them.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the NexTon Smart Contracts. It was conducted on commit `9aa979d249dbb1b6de97667876c9d48de814e193` from git repository link: *https://github.com/Nex-TON/Nexton_Contracts*

There are 2 files in our audit scope:

- NexTon.tact
- nft_item.fc

1.3. Audit methodology

The security audit process includes three steps:

- Mechanism Design is reviewed to look for any potential problems.
- Source codes are scanned/tested for commonly known and more specific vulnerabilities using public and our in-house security analysis tool.
- Manual audit of the codes for security issues. The source code is manually analyzed to look for any potential problems.

1.3.1. Vulnerability severities

For vulnerabilities, we categorize the findings into categories, depending on their criticality:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the functioning; creates a critical risk; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the code with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the code with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

NexTon acknowledges that the security services provided by Verichains, are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. NexTon understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, NexTon agrees that Verichains shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process.

1.5. Acceptance Minute

This final report served by Verichains to the NexTon will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the NexTon, the final report will be considered fully accepted by the NexTon without the signature.

2. AUDIT RESULT

2.1. Overview

The NexTon Smart Contracts was written in `Tact` and `Func` languages.

2.1.1. NexTon Contract

The staking contract is written in the `Tact` language. This contract is designed to allow users to stake their TON and earn rewards. During the lock period, users will hold an NFT item that collects their staking information.

2.1.2. NftItem Contract

The contract is written in the FunC language. It handles the logic for users unstaking their TON in the NexTon contract. Users will transfer ownership of their NFT items to the NexTon contract, during which the contract will check the balance and transfer rewards to the users.

2.2. Findings

During the audit process, the audit team had identified some vulnerable issue in the contract code.

2.2.1. NexTon.tact - Error when admin tries to withdraw all funds LOW

The OwnerWithdraw action allows the administrator to withdraw an amount less than or equal to the contract's balance. However, attempting to withdraw the entire balance will trigger an error due to the contract's need for a minimum native reserve to maintain functionality. Additionally, the contract's use of SendIgnoreError and SendPayGasSeparate modes when sending funds can make it difficult for the administrator to precisely calculate the maximum withdrawable amount.

```
receive(msg: OwnerWithdraw) {
        self.requireOwner();
        nativeThrowUnless(1000, msg.amount <= myBalance());
        nativeReserve(MIN_CONTRACT_BALANCE, 0);
        send(SendParameters{
            to: sender(),
            value: msg.amount, // Error when msg.amount == myBalance(), it also error when
msg.amount = myBalance() - MIN_CONTRACT_BALANCE
            mode: SendIgnoreErrors + SendPayGasSeparately + SendRemainingValue,
            body: "Assets withdrawn".asComment()
        });
    }
```

RECOMMENDATION

Update the logic, and consider removing the SendPayGasSeparately mode. The contract can withdraw myBalance() - MIN_CONTRACT_BALANCE when SendPayGasSeparately mode is not used.

UPDATES

- **May 22, 2024**: This issue has been acknowledged and fixed in commit `442d51254e64535b610bc26b7e2a6cda21ecf133` by NexTon team.

2.2.2. NexTon.tact - Avoid adding (+) base mode with optional flags INFORMATIVE

Using the `+` operator to combine base modes with optional flags can lead to possibily of excess value. Use the `|` operator to combine base modes with optional flags.

RECOMMENDATION

Update the code to use the `|` operator to combine base modes with optional flags.

UPDATES

- **May 22, 2024**: This issue has been acknowledged and fixed in commit `a52c4e2c90599981076acae542f268bb77dc2a1c` by NexTon team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*May 22, 2024*	Public Report	Verichains Lab

Table 2. Report versions history