DeHacker

Code Security Assessment

Nexchain

April 24th, 2025

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best prac tices. Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	Inifinitar
Platform	Ethereum(ETH)
Website	https://nexchain.ai
Type	DeFi
Language	Solidity
Codebase	Token

Vulnerability Summary

Vulnerability Level	**Total**	**Mitigated**	**Declined**	**Acknowledged**	**Partially Resolved**	**Resolved**
Critical	0	0	0	0	0	0
Major	2	0	0	0	2	0
Medium	0	0	0	0	0	0
Minor	1	0	0	0	0	1
Informational	2	0	0	1	0	1
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
OWN	@openzeppelin/contracts/access/Ownable.sol	38578bd71c0a909840e67202db527cc6b4e6b437e0f39f0c909da32c1e30cb81
IER	@openzeppelin/contracts/interfaces/draft-IERC6093.sol	56380323009ef4a119d44550b910fde1bff9cedde8f7f4c690152c7629bc3338
IEC	@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	9e7c70ec72d2f7d592e23ea84f3852b04f91f6f644ce57e0263493046b36afb9
ERE	@openzeppelin/contracts/token/ERC20/ERC20.sol	c322867b0b30f6d5f791e5ab9af5e3f5ff1c8c127906748f6725e592a5f40719
IEE	@openzeppelin/contracts/token/ERC20/IERC20.sol	30edf7394bab78d48b7db3a059248e1ea7c2c77d2ec0e37a13bb91415aafbe5a
COT	@openzeppelin/contracts/utils/Context.sol	847fda5460fee70f56f4200f59b82ae622bb03c79c77e67af010e31b7e2cc5b6
NCH	contracts/NexchainCoin.sol	ac17a803810a5921d07af3b40b8c7538eb54a5be82cbc665214d3c09948e059e

Findings

ID	Title	Severity	Status
NEX-01	Initial Token Distribution	Major	Partially Resolved
NEX-02	Centralization Risks In NexchainCoin.Sol	Major	Partially Resolved
NEX-03	State Variable Should Be Declared Constant	Minor	Resolved
NEX-04	Usage Of `Block.Timestamp	Informational	Resolved
NEX-05	Burn Process Can Be Finished By Anyone AndRestarted By Privilege User	Informational	Acknowledged

MAJOR

NEX-01 | Initial Token Distribution

Issue	Severity	Location	Status
Centralization	Major	contracts/NexchainCoin.sol (base): 21, 21	Partially Resolved

Description

All of the NEX tokens are sent to the contract deployer or one or several externally-owned account (EOA) addresses. This isa centralization risk because the deployer or the owner(s) of the EOAs can distribute tokens without obtaining the consensusof the community. Any compromise to these addresses may allow a hacker to steal and sell tokens on the market, resultingin severe damage to 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 to the private keys of the deployer account or EOAs. A multi-signature (⅔, ⅗) 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 vesting schedule for long-term success, and deanonymize the project team with a third-party KYC provider to create greater accountability.

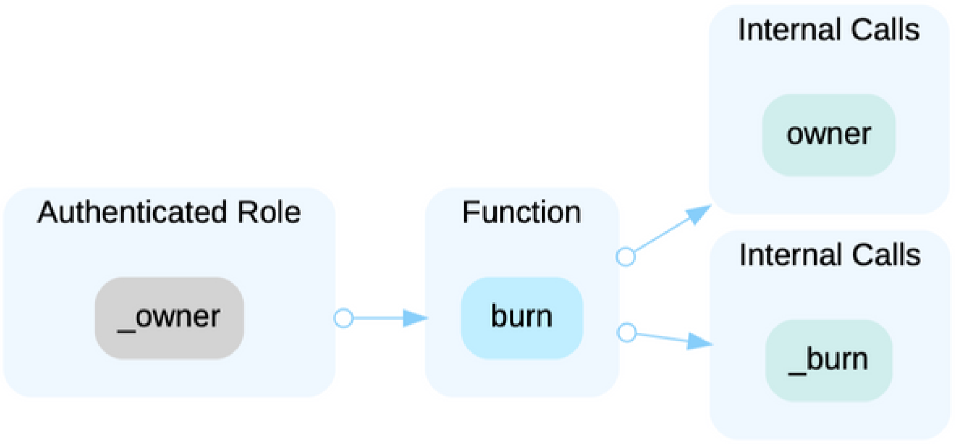
MAJOR

NEX-02 | Centralization Risks In NexchainCoin.Sol

Issue	Severity	Location	Status
Centralization	Major	contracts/NexchainCoin.sol (base): 24	Partially Resolved

Description

In the contract NexchainCoin , the role _owner has authority over the functions shown in the diagram below. Anycompromise to the _owner account may allow the hacker to take advantage of this authority and burn a specified tokenamount.



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 would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:
Timelock and Multi sign (⅔, ⅗) combination mitigate by delaying the sensitive operation and avoiding a single point of key management 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 key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

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

MINOR

NEX-03 | State Variable Should Be Declared Constant

Issue	Severity	Location	Status
Coding Issue	Minor	contracts/NexchainCoin.sol (update_20250417): 23	Resolved

Description

State variables that never change should be declared as constant to save gas.

```
23      uint256 public burnPreparationDuration = 2 * 24 * 60 * 60;
```

State variables that never change should be declared as constant to save gas.
Description

Recommendation

We recommend adding the constant attribute to state variables that never change.

INFORMATIONAL

NEX-04 | Usage Of ‘Block.Timestamp

Issue	Severity	Location	Status
Logical Issue	Informational	@openzeppelin/contracts/governance/utils/Votes.sol (update_20250417): 143~162; @openzeppelin/contracts/token/ERC20/extensions/ERC20Permit.sol (update_20250417): 44~67; @openzeppelin/contracts/utils/types/Time.sol (update_20250417): 74~80; contracts/NexchainCoin.sol (update_20250417): 35~40, 42~53, 55~62	Resolved

Description

block.timestamp is used for comparison, which can be risky since timestamp can be influenced by miners. That means the miner creating the block can manipulate the block.timestamp , to some degree, and change the outcome of the transaction.

```
        require(_burnPossibleFromTime != 0, "Not open");
```

```
        require(_burnPossibleFromTime <= block.timestamp,
"Time has not come yet");
```

```
        require(_burnPossibleFromTime == 0, "Already open");
        if (_burnPossibleFromTime != 0) {
```

```
        if (block.timestamp > deadline) {
```

```
         if (block.timestamp > expiry) {
```

```
         return effect <= timepoint ? (valueAfter, 0, 0) : (valueBefore,
valueAfter, effect);
```

Recommendation

We recommend against relying on block.timestamp .Reference:
https://swcregistry.io/docs/SWC-116

INFORMATIONAL

NEX-05 | Burn Process Can Be Finished By Anyone And Restarted By Privilege User

Issue	Severity	Location	Status
Design Issue	Informational	contracts/NexchainCoin.sol (update_20250417): 42, 55	Acknowledged

Description

Any user can burn the tokens by calling finishBurn function in the NexchainCoin contract when _burnPossibleFromTimereached the block.timestampAfter anyone calls the finish burn and success, owner can re-open the burn process by calling openBurn function. This maybbring similar centralization risk if the owner account has been compromised.

Recommendation

We recommend the team to double confirm this is an intended design

Disclaimer

Appendix

Finding Categories

Centralization / Privilege
Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style
Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code
Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Logical Issue
Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum" command against the target file.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

Ethereum
Cosmos
Eos
Substrate

TECH STACK

Python
Solidity
Rust
c++

CONTACTS

https://dehacker.io
https://twitter.com/dehackerio
https://github.com/dehacker/audits_public
https://t.me/dehackerio
https://blog.dehacker.io/

DeHacker

April 24th 2025