DeHacker

Code Security Assessment

Kingnet

July 21th, 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	Kingnet
Platform	Binance Smart Chain (BSC)
Website	https://kingnetai.io
Type	DeFi
Language	Solidity
Codebase	0xD31b809aee5ACACE5d69E76aa7441eBBA3273B96

Vulnerability Summary

Vulnerability Level	Total	Mitigated	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	1	0	0	0	1	0
Medium	0	0	0	0	0	0
Minor	0	0	0	0	0	0
Informational	4	0	0	4	0	0
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
KIN	contracts/Kingnet.sol	f91ada538771d280063919795424d9844fe6eebd05a9e8cb5705c17d98a47725

Findings

ID	Title	Severity	Status
KIN-01	Centralization Risks in Kingnet.sol	Major	Partially Resolved
KIN-02	Missing Emit Events	Informational	Acknowledged
KIN-03	Redundant Ownership Transfer in Constructor	Informational	Acknowledged
KIN-04	Lack of Minimum Deposit Amount Restriction Enables DoS via Micro-Deposits	Informational	Acknowledged
KIN-05	Array missing `pop` function	Informational	Acknowledged

MAJOR

KIN-01 | CENTRALIZATION RISKS IN KINGNET.SOL

Issue	Severity	Location	Status
Centralization	Major	contracts/Kingnet.sol: 95, 104	Partially Resolved

Description

In the contract Kingnet , 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 and add a new deposit pool or withdraw thecontract balance to the owner.

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 accounts
with 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 (⅔, ⅗) 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

Recommendation

- 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 publicaudience.

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.

Alleviation

[Kingnet, 07/07/2025]: The team acknowledged the issue and adopted the multisign solution to ensure the private keymanagement process at the current stage. The Kingnet contract has transferred the ownership to a Gnosis Safe contractwith 2/4 signers in the sensitive function signing process.

- Transfer ownership to Gnosis Safe:0x505d07e18117d6aec899736a186ab757c0c4a36c39b86c826b00ae06ac81bfeb, Gnosis safe contract address 0x70C9423D81EFaE4Cb750188b9767B8e0d48154E3.
- The 4 multisign addresses:

1. EOA:0x1d953e0ceabebe97d3b0026e04c122ca17e0316c2.
EOA:0x894e9f53360a652d504484e45801093e2eba07833.
EOA:0x5730f93eefdfff4ba850dc04f4e95bef1110b4ad4.
EOA:0x6b74b538ef03bf044ac5b0a063e18f90d2390c0a

INFORMATIONAL

KIN-02 | MISSING EMIT EVENTS

Issue	Severity	Location	Status
Coding Style	Informational	contracts/Kingnet.sol: 95, 104	Acknowledged

Description

There should always be events emitted in sensitive functions that are controlled by centralization roles

Recommendation

It is recommended to emit events in sensitive functions that are controlled by centralization roles.

INFORMATIONAL

KIN-03 | REDUNDANT OWNERSHIP TRANSFER IN CONSTRUCTOR

Issue	Severity	Location	Status
Coding Style	Informational	contracts/Kingnet.sol: 40	Acknowledged

Description

The constructor of the Kingnet contract includes the following line:

39 constructor() {
40 transferOwnership(msg.sender);
41 // Initialize deposit pools (amounts in ETH)
42 pools.push(DepositPool(0.001 ether, 0, 0));
43 }

However, since Kingnet inherits from OpenZeppelin's Ownable contract, ownership is already initialized to msg.senderduring contract deployment via the Ownable constructor. Explicitly calling transferOwnership(msg.sender) again isredundant and unnecessary.

Recommendation

Consider removing the redundant transferOwnership(msg.sender) line from the constructor to simplify the code andavoid confusions

INFORMATIONAL

KIN-04 LACK OF MINIMUM DEPOSIT AMOUNT RESTRICTIONENABLES DOS VIA MICRO-DEPOSITS

Issue	Severity	Location	Status
Volatile Code	Informational	contracts/Kingnet.sol: 46, 95	Acknowledged

Description

The addPool function allows the contract owner to create a new deposit pool with any arbitrary amount. However, there isno lower bound on the _amount parameter. This introduces a potential DoS vector when extremely small deposit amounts(e.g., 1 wei) are added.

```
95 function addPool(uint256 _amount) external onlyOwner {
96 // Check if a pool with the same amount already exists
97 for (uint256 i = 0; i < pools.length; i++) {
98 require(pools[i].amount != _amount, "Pool amount already exists");
99 }
100 pools.push(DepositPool(_amount, 0, 0));
101 }
```

Because users can repeatedly deposit into these pools, a malicious user might abuse this by performing a large number oftiny deposits, causing the records array to grow excessively and generating a flood of Deposit events, which could lead toevent log spam and increased gas costs when reading records

Recommendation

Consider enforcing a minimum deposit amount threshold in addPool() to prevent micro-denomination abuse

INFORMATIONAL

KIN-05 ARRAY MISSING pop FUNCTION

Issue	Severity	Location	Status
Volatile Code	Informational	contracts/Kingnet.sol: 23, 24	Acknowledged

Description

Arrays without the pop operation in Solidity can lead to inefficient memory management and increase the likelihood of out-of-gas errors.

Recommendation

Consider adding functionality to remove elements from the array to prevent it from becoming too large over the lifetime of thecontract

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

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

July 21th 2025