DeHacker

Code Security Assessment

Infinitar

April 17th, 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	Binance Smart Chain (BSC)
Website	https://www.infinitar.com
Type	GameFi
Language	Solidity
Codebase	Token

Vulnerability Summary

Vulnerability Level	Total	Mitigated	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	3	0	0	3	0	0
Medium	1	0	0	0	0	1
Minor	2	0	0	0	0	2
Informational	0	0	0	0	0	0
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
IGT	IGT_TOKEN.sol	b6ad58425e1be4edc1e3f479e406b3f733566476bad9ff8d46e93334e6c1934b
INF	INF_TOKEN.sol	72a297e0e9773b48d2d1a67424307bd08387fd7b21b8c7b229a1da62c3454dda
IGO	IGT_TOKEN.sol	90827c50171707d8405c0e6f803ca67967e79d10d062b871475fdb0a5a918314
INT	INF_TOKEN.sol	7e3e8533f7a2884f2aca4617778ff4194c2f8c5093558b65c68984c189c7b572

Findings

ID	Title	Severity	Status
IGT-02	Initial Token Distribution	Major	Acknowledged
INF-01	Centralized Balanced Manipulation	Major	Acknowledged
INF-02	Centralization Related Risks	Major	Acknowledged
INF-05	Irreversible Deactivation Mechanism In INFTokens Contract	Medium	Resolved
INF-03	Usage Of transfer / send For Sending Native Token	Minor	Resolved
INF-04	Missing Zero Address Validation	Minor	Resolved
IGT-01	Variables That Could Be Declared As Immutable	Minor	Resolved
TOK-01	State Variable Should Be Declared Constant	Minor	Resolved

MAJOR

IGT-02 | Initial Token Distribution

Issue	Severity	Location	Status
Centralization	Major	IGT_TOKEN.sol (08/26 - 1f17d2): 16~1	Acknowledged

Description

All of the IGT tokens (1 billion) are sent to the contract deployer or one or several externally-owned account (EOA)addresses. This is a centralization risk because the deployer or the owner(s) of the EOAs can distribute tokens withoutobtaining the consensus of the community. Any compromise to these addresses may allow a hacker to steal and sell tokenson the market, resulting in 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 tothe 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 vestingschedule for long-term success, and deanonymize the project team with a third-party KYC provider to create greateraccountability.

MAJOR

INF-01 | Centralized Balance Manipulation

Issue	Severity	Location	Status
Centralization	Major	INF_TOKEN.sol (08/26 - 1f17d2): 81, 89	Acknowledged

Description

In the INFToken contract, the admin role possesses the authority to update the token balance of any arbitrary accountwithout any restrictions. Specifically, the admin can mint any amount of tokens to any account and burn tokens from any account.Any compromise to the admin account may allow a hacker to take advantage of this authority and manipulate users' balances, the hacker could also update his/her balance to a large number, impact the circulation of tokens in the market.

Recommendation

We recommend the team makes efforts to restrict access to the private key of the privileged account. A strategy of multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. In addition, the team should be transparent and notify the community in advance whenever they plan to mint more tokens or engage in similar balance-related operations.Here are some feasible short-term and long-term suggestions that would mitigate the potential risk to a different level and suggestions that would permanently fully resolve the risk:

Short Term:

A multi signature (⅔, ⅗) wallet mitigate the risk by avoiding a single point of key management failure.

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to a private key compromised;

AND

- A medium/blog link for sharing the time-lock contract and multi-signers' addresses information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the gnosis address with ALL the multi-signer addresses for the verification process.
- Provide a link to the medium/blog with all of the above information included.

Long Term:

A DAO for controlling the operation mitigate the risk by applying transparency and decentralization.

- Introduction of a DAO, governance, or voting module to increase decentralization, transparency, and user involvement;
  AND
- A medium/blog link for sharing the multi-signers' addresses, and DAO information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the gnosis address with ALL the multi-signer addresses for the verification process.
- Provide a link to the medium/blog with all of the above information included.

Permanent:

The following actions can fully resolve the risk:

- Renounce the ownership and never claim back the privileged role.
  OR
- Remove the risky functionality.
  OR
- Add minting logic (such as a vesting schedule) to the contract instead of allowing the owner account to call the sensitive function directly.

Note: we recommend the project team consider the long-term solution or the permanent solution. The project team shallmake a decision based on the current state of their project, timeline, and project resources.

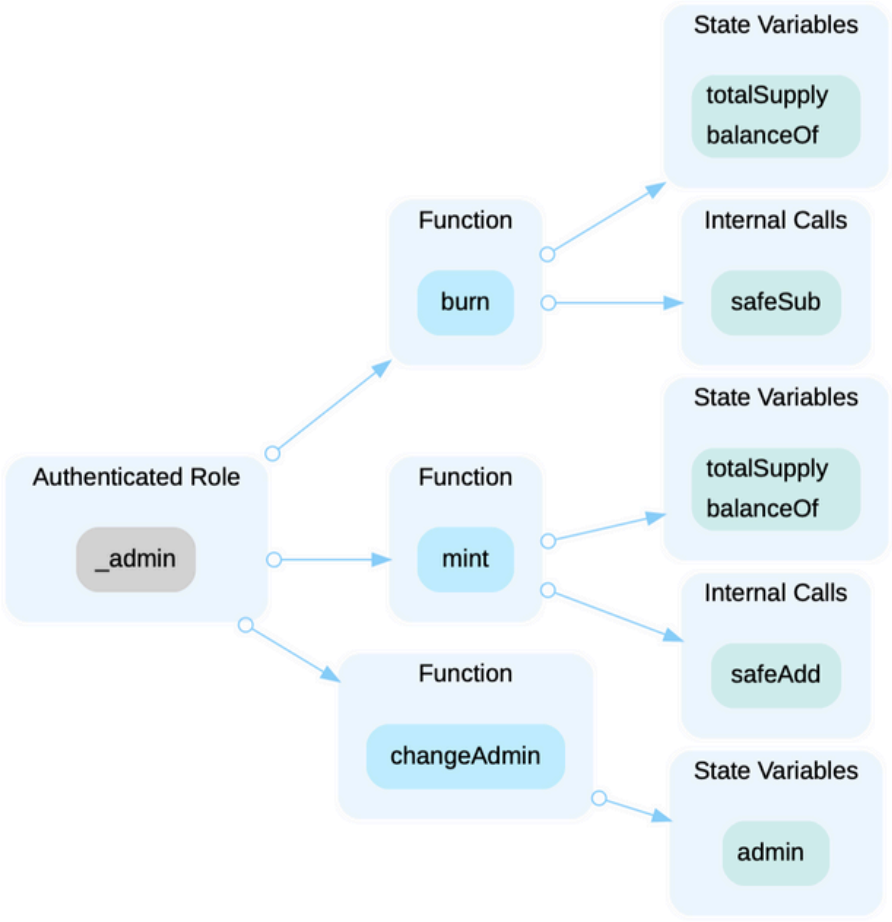
MAJOR

INF-02 | Centralization Related Risks

Issue	Severity	Location	Status
Centralization	Major	INF_TOKEN.sol (08/26 - 1f17d2): 81, 89, 98, 104, 109	Acknowledged

Description

In the contract INFToken , the role _admin has authority over the functions shown in the diagram below. Any compromiseto the _admin account may allow the hacker to take advantage of this authority and burn tokens from an address, minttokens to an address, change the admin address.



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 carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature 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:

Time-lock 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 role.
  OR
- Remove the risky functionality.

MEDIUM

INF-05 | Irreversible Deactivation Mechanism In INFTokenContract

Issue	Severity	Location	Status
Logical Issue	Medium	INF_TOKEN.sol (08/26 - 1f17d2): 104~107	Resolved

Description

The INF Token contract contains a contract Active boolean flag that effectively controls the operational status of most critical functions, including transfer/transfer From , mint , and burn . This flag is initialized to true , enabling all controlled functions upon deployment. The contract includes a function, deactivateContract() , which when invoked by the admin, sets contractActive to false , thereby disabling the aforementioned critical functions. Once deactivateContract() is called, contractActive is set to false , and there is no mechanism provided to reactivate the contract. This permanently disables all essential functionalities such as token transfers, minting, and burning.

Recommendation

The auditing team would like to confirm with the project team if the irreversible deactivation is the intended design and inquireabout the handling of the remaining tokens within the contract once the contractActive status is set to false .

MINOR

INF-03 | Usage Of transfer / send For Sending NativeTokens

Issue	Severity	Location	Status
Volatile Code	Minor	INF_TOKEN.sol (08/26 - 1f17d2): 110	Resolved

Description

It is not recommended to use Solidity's transfer() and send() functions for transferring native tokens, since some contracts may not be able to receive the funds. Those functions forward only a fixed amount of gas (2300 specifically) andthe receiving contracts may run out of gas before finishing the transfer. Also, EVM instructions' gas costs may increase in thefuture. Thus, some contracts that can receive now may stop working in the future due to the gas limitation.
The following function in the INFToken contract uses transfer() to send the native token.

```
109     function withdrawEther() public onlyAdmin {
110         payable(admin).transfer(address(this).balance);
111     }
```

Recommendation

We recommend using the Address.sendValue() function, from OpenZeppelin.

MINOR

INF-04 | Missing Zero Address Validation

Issue	Severity	Location	Status
Volatile Code	Minor	INF_TOKEN.sol (08/26 - 1f17d2): 81, 89	Resolved

Description

The INFToken contract lacks a check to ensure that the provided address is not address(0) in the mint() and burn()functions. This can lead to unintended behavior, such as tokens being minted or burned to/from the zero address.

```
    function mint(address _to, uint256 _value) public onlyAdmin isActive returns
(bool success) {
        ...
    }
```

```
    function burn(address _from, uint256 _value) public onlyAdmin isActive returns
(bool success) {
        ...
    }
```

Recommendation

We recommend adding a check the passed-in address is not address(0) to prevent unexpected errors.

MINOR

IGT-01 | Variables That Could Be Declared As Immutable

Issue	Severity	Location	Status
Gas Optimization	Minor	IGT_TOKEN.sol (08/26 - 1f17d2): 8	Resolved

Description

The totalSupply in the IGTToken contract assigned in the constructor can be declared as immutable . Immutable statevariables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. Abig advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variablessince they will not be stored in storage.

```
uint256 public totalSupply;

constructor() {
    totalSupply = 1000000000 * 10**uint256(decimals);
    ...
}
```

Recommendation

We recommend declaring these variables as immutable. Please note that the immutable keyword only works in Solidityversion v0.6.5 and up.

MINOR

TOK-01 | Variables That Could Be Declared As Immutable

Issue	Severity	Location	Status
Coding Issue	Minor	IGT_TOKEN.sol (08/26 - 1f17d2): 5, 6, 7; INF_TOKEN.sol (08/26- 1f17d2): 5, 6, 7	Resolved

Description

State variables that never change should be declared as constant to save gas.In the IGTToken contract:

```
5     string public name = "Infinitar Governance Token";
6     string public symbol = "IGT";
7     uint8 public decimals = 18;
```

- name , symbol and decimals can be declared constant.

In the INFToken contract:

```
5     string public name = "INFinitar coin";
6     string public symbol = "INF";
7     uint8 public decimals = 18;
```

- name , symbol and decimals can be declared constant.

Recommendation

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

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

April 17th 2025