InterFi
NETWORK

SMART CONTRACT AUDIT

interfinetwork

hello@interfi.network

https://interfi.network

PREPARED FOR

SACRED TAILS - LEGACY SHINSEI

INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	Sacred Tails
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Proxy	0x878eF413F193E0Bf8C356076C01267F48cC0dd7c
Implementation	0x8DdC3Ee0303cd66fcf5F69e0361A36e94DB0F26f
Blockchain	Polygon
Centralization	Active ownership
Commit	a6f10d59e058c41c6290cb5ee27db750710c826b
Website	https://www.reload.games/
Telegram	https://t.me/reloadgames/
X (Twitter)	https://twitter.com/r3loadgames/
Discord	https://discord.com/invite/reloadgames/
Report Date	January 21, 2024

ℹ Verify the authenticity of this report on our website: https://www.github.com/interfinetwork

EXECUTIVE SUMMARY

InterFi has performed the automated and manual analysis of solidity codes. Solidity codes were reviewed for common contract vulnerabilities and centralized exploits. Here's a quick audit summary:

Status	Critical 🔴	Major 🟠	Medium 🟡	Minor 🟢	Unknown 🟤
Open	0	0	0	3	0
Acknowledged	0	1	2	1	1
Resolved	0	2	0	1	0
Noteworthy Privileges	Mint to Treasury, Airdrop NFTs, Populate IDs, Update Box Price, Update Referral Discount Percent, Update Discount Percent, Withdraw				

ℹ Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.

ℹ Please note that centralization privileges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.

TABLE OF CONTENTS

SCOPE OF WORK

InterFi was consulted by Sacred Tails to conduct the smart contract audit of their solidity source codes.

The audit scope of work is strictly limited to mentioned solidity file(s) only:

- ShinseiNFT.sol

ℹ Please note that use of upgradeable contracts allows developers to modify contract logic in the future. While this feature provides flexibility for updates and improvements, it also introduces the potential risk of new vulnerabilities or exploits. Stakeholders should exercise caution and ensure continuous monitoring and security practices are in place to mitigate these risks.

ℹ If source codes are not deployed on the main net, they can be modified or altered before main-net deployment. Verify the contract's deployment status below:

Proxy Contract Link
https://polygonscan.com/address/0x878ef413f193e0bf8c356076c01267f48cc0dd7c#code
Implementation Contract Link
https://polygonscan.com/address/0x8ddc3ee0303cd66fcf5f69e0361a36e94db0f26f#code

Contract Name	ShinseiNFT
Compiler Version	0.8.20
License	MIT

AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit. Here's a brief overview of InterFi's auditing process and methodology:

CONNECT

- The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.

AUDIT

- Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:
    - Remix IDE Developer Tool
    - Open Zeppelin Code Analyzer
    - SWC Vulnerabilities Registry
    - DEX Dependencies, e.g., Pancakeswap, Uniswap
- Simulations are performed to identify centralized exploits causing contract and/or trade locks.
- A manual line-by-line analysis is performed to identify contract issues and centralized privileges. We may inspect below mentioned common contract vulnerabilities, and centralized exploits:

Centralized Exploits	○ Token Supply Manipulation ○ Access Control and Authorization ○ Assets Manipulation ○ Ownership Control ○ Liquidity Access ○ Stop and Pause Trading ○ Ownable Library Verification

Common Contract Vulnerabilities	○ Integer Overflow ○ Lack of Arbitrary limits ○ Incorrect Inheritance Order ○ Typographical Errors ○ Requirement Violation ○ Gas Optimization ○ Coding Style Violations ○ Re-entrancy ○ Third-Party Dependencies ○ Potential Sandwich Attacks ○ Irrelevant Codes ○ Divide before multiply ○ Conformance to Solidity Naming Guides ○ Compiler Specific Warnings ○ Language Specific Warnings

REPORT

- The auditing team provides a preliminary report specifying all the checks which have been performed and the findings thereof.
- The client's development team reviews the report and makes amendments to solidity codes.
- The auditing team provides the final comprehensive report with open and unresolved issues.

PUBLISH

- The client may use the audit report internally or disclose it publicly.

ℹ It is important to note that there is no pass or fail in the audit, it is recommended to view the audit as an unbiased assessment of the safety of solidity codes.

RISK CATEGORIES

Smart contracts are generally designed to hold, approve, and transfer tokens. This makes them very tempting attack targets. A successful external attack may allow the external attacker to directly exploit. A successful centralization-related exploit may allow the privileged role to directly exploit. All risks which are identified in the audit report are categorized here for the reader to review:

Risk Type	Definition
Critical 🔴	These risks could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
Major 🟠	These risks are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.
Medium 🟡	These risks should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. Low-risk re-entrancy-related vulnerabilities should be fixed to deter exploits.
Minor 🟢	These risks do not pose a considerable risk to the contract or those who interact with it. They are code-style violations and deviations from standard practices. They should be highlighted and fixed nonetheless.
Unknown 🟤	These risks pose uncertain severity to the contract or those who interact with it. They should be fixed immediately to mitigate the risk uncertainty.

All statuses which are identified in the audit report are categorized here for the reader to review:

Status Type	Definition
Open	Risks are open.
Acknowledged	Risks are acknowledged, but not fixed.
Resolved	Risks are acknowledged and fixed.

CENTRALIZED PRIVILEGES

Centralization risk is the most common cause of cryptography asset loss. When a smart contract has a privileged role, the risk related to centralization is elevated.

There are some well-intended reasons have privileged roles, such as:

- Privileged roles can be granted the power to `pause()` the contract in case of an external attack.
- Privileged roles can use functions like, `include()`, and `exclude()` to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale and to list on an exchange.

Authorizing privileged roles to externally-owned-account (EOA) is dangerous. Lately, centralization-related losses are increasing in frequency and magnitude.

- The client can lower centralization-related risks by implementing below mentioned practices:
- Privileged role's private key must be carefully secured to avoid any potential hack.
- Privileged role should be shared by multi-signature (multi-sig) wallets.
- Authorized privilege can be locked in a contract, user voting, or community DAO can be introduced to unlock the privilege.
- Renouncing the contract ownership, and privileged roles.
- Remove functions with elevated centralization risk.

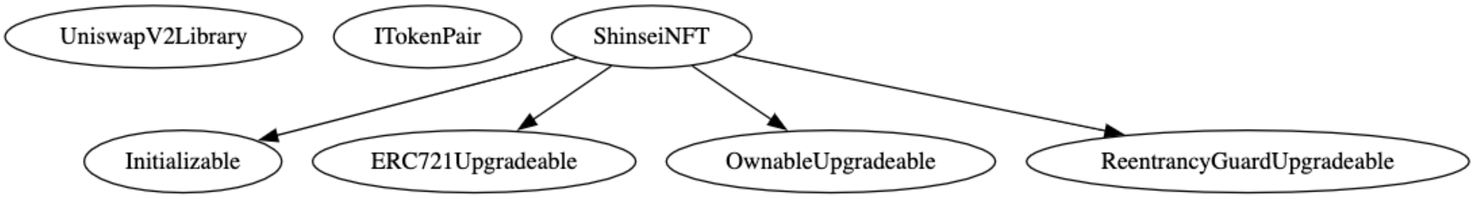
ℹ Understand the project's initial asset distribution. Assets in the liquidity pair should be locked. Assets outside the liquidity pair should be locked with a release schedule.

AUTOMATED ANALYSIS

Symbol	Definition
🛑	Function modifies state
💵	Function is payable
🔒	Function is internal
🔐	Function is private
❗	Function is important

```
| **UniswapV2Library** | Library |  |||

| └ | getAmountOut | Internal 🔒 |   | |

||||||

| **ITokenPair** | Interface |  |||

| └ | getReserves | External ❗ |   |NO❗ |

||||||

| **ShinseiNFT** | Implementation | Initializable, ERC721Upgradeable, OwnableUpgradeable,
ReentrancyGuardUpgradeable |||

| └ | initialize | Public ❗ | 🛑  | initializer |

| └ | mintToTreasury | External ❗ | 🛑  | onlyOwner |

| └ | airdropNFTs | External ❗ | 🛑  | onlyOwner |

| └ | purchaseShinsei | External ❗ |  💵 | nonReentrant |

| └ | claimCashback | External ❗ | 🛑  | nonReentrant |

| └ | populateIds | Public ❗ | 🛑  | onlyOwner |

| └ | randomGenerator | Internal 🔒 | 🛑  | |

| └ | calculateAmountOut | Public ❗ |   |NO❗ |

| └ | updateBaseURI | Public ❗ | 🛑  | onlyOwner |
| └ | tokenURI | Public ❗️ |   |NO❗️ |
| └ | updateBoxPriceUSDT | External ❗️ | 🛑  | onlyOwner |
| └ | updateReferralDiscountPercentage | External ❗️ | 🛑  | onlyOwner |
| └ | updateDiscountPercentage | External ❗️ | 🛑  | onlyOwner |
| └ | withdraw | Public ❗️ | 🛑  | onlyOwner |
| └ | <Receive Ether> | External ❗️ |  💵 |NO❗️ |
| └ | <Fallback> | External ❗️ |  💵 |NO❗️ |
```

INHERITANCE GRAPH



MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized privileges	Major 🟠
CEN-02	Centralized role can withdraw entire ether and USDT balance from smart contract	

Important `onlyOwner` centralized privileges are listed below:

```
mintToTreasury()
airdropNFTs()
populateIds()
updateBaseURI()
updateBoxPriceUSDT()
updateReferralDiscountPercentage()
updateDiscountPercentage()
withdraw()
```

Smart contract defines a range of token IDs from `minId` to `maxId`. These values are used in functions like `populateIds` to generate a list of available token IDs. The range between these two values implicitly limits the total number of NFTs. However, mint functions do not explicitly enforce mint limit all minting operations, e.g., `mintToTreasury` and `airdropNFTs` functions require token IDs to be less than `minId`, which may be an inconsistency.

RECOMMENDATION

Deployers, contract owners, administrators, access controlled, and all other privileged roles' private-keys/access-keys/admin-keys should be secured carefully. These entities can have a single point of failure that compromises the security of the project.

Require multiple signatures from different parties to execute certain sensitive functions within contracts. This spreads control and reduces the risk of a single party having complete authority.

Implement a governance model that enables token holders or other stakeholders to participate in decision-making processes. This can include voting on contract upgrades, parameter changes, or any other critical decisions that impact the contract's functioning.

RESOLUTION

Sacred Tails team acknowledged that privileged roles are used as required, and accepted to use Gnosis multi-sig to manage centralized privileges whenever possible.

Currently, a privileged role is a prerequisite for our smart contract. This role is specifically designed to enable pre-minting of NFTs and the distribution of select NFTs to our existing NFT holders. To manage the availability of our NFTs for public sale effectively, we are implementing a cap. The surplus NFTs will be offered exclusively to the holders of our previous NFTs.

In accordance with these requirements, the privileged role is granted exclusive authority to withdraw USDT or Ether directly from the contract. Given these considerations, the following functions are reserved for access by the privileged role only:

```
mintToTreasury()
AirdropNFTs()
populateIds()
updateBaseURI()
updateBoxPriceUSDT()
updateReferralDiscountPercentage()
updateDiscountPercentage()
withdraw()
```

Looking ahead, we plan to introduce a governance model and transition towards utilizing multi-signature functionality as our platform evolves and our functional needs increase. This strategic approach ensures that our smart contract operations remain secure, transparent, and aligned with our growth trajectory.

Identifier	Definition	Severity
CEN-09	Use of proxy and upgradeable contracts	Major ●

Privileged role can initiate contract implementation. Contract upgradeability allows privileged roles to change current contract implementation.

```
contract ShinseiNFT is Initializable, ERC721Upgradeable, OwnableUpgradeable,
ReentrancyGuardUpgradeable{
```

Add `_disableInitializers` in upgradeable implementation to add a safety measure that prevents initializer functions from being called more than once, reducing the risk of unintended behavior or vulnerabilities.

RECOMMENDATION

Test and validate current contract thoroughly before deployment. While proxy contracts are great for robust deployments while maintaining the upgradeable flexibility, proxy codes are prone to new security or logical issues that may compromise the project.

RESOLUTION

Sacred Tails team commented that, to keep our contract adaptable and future-proof within evolving game ecosystem, we have implemented upgradeability features. This strategic decision allows us to make necessary modifications or enhancements to our contract's functionality as the need arises.

Identifier	Definition	Severity
LOG-01	Lack of appropriate maximum price boundary	Minor 🟢

Below mentioned function is set without any arbitrary upper limit:

```
updateBoxPriceUSDT()
```

RECOMMENDATION

Set a maximum allowable `boxPriceUSDT` input limit.

ACKNOWLEDGEMENT

Sacred Tails team acknowledged this finding, and commented that, we considered adding a price boundary check but decided against it due to the unpredictable and unstable nature of asset prices.

Identifier	Definition	Severity
LOG-02	Potential front-running	Medium 🟡

Front-running is a concern in blockchain transactions, particularly on public networks like Ethereum. It occurs when someone sees a pending transaction and manages to get their own transaction confirmed first, potentially for profit or advantage. Below mentioned functions are vulnerable to front-running:

`purchaseShinsei()`
`calculateAmountOut()`
Random Number Generation in `purchaseShinsei()`

RECOMMENDATION

Use commit-reveal scheme to hide transactions, use verifiable random function (VRF) such as Chainlink VRF for random number generation, and use better pricing model than direct LP spot price in `calculateAmountOut()`.

ACKNOWLEDGEMENT

Sacred Tails team acknowledged this finding, and commented that, our implementation of random number generation on the blockchain, while not entirely unpredictable, significantly reduces the risk of front-running attacks through its complexity. By incorporating multiple variables beyond the timestamp and block number, such as a dynamically populated array of public IDs influenced by `populateIds`, and employing a unique sorting method for this array, we make predicting the next random number challenging.

Identifier	Definition
LOG-03	Re-entrancy

Below mentioned functions are used with re-entrancy guard to prevent re-entrancy attacks:

```
purchaseShinsei()
claimCashback()
```

Identifier	Definition	Severity
LOG-04	Unchecked return values	Minor 🟢

In `withdraw()` function, external calls to transfer `ERC20` tokens without checking the return value.

RECOMMENDATION

Check return values of external calls, especially for `ERC20` token transfers.

Identifier	Definition	Severity
LOG-05	Authorization through `tx.origin`	Major ●

Major issue with `tx.origin` is its vulnerability to phishing attacks. If smart contract checks `tx.origin` for authentication, an attacker can create a malicious contract that calls smart contract. `tx.origin` would still be the original user, not the attacker's contract. This way, the attacker can trick users into executing transactions that they didn't intend to.

Using `tx.origin` for authorization could make the contract vulnerable as it refers to the original external account that started the transaction.

```
purchaseShinsei()
```

RECOMMENDATION

Avoid authorizations via global variables wherever necessary. Use `msg.sender` instead.

ACKNOWLEDGEMENT

Sacred Tails team acknowledged this finding, and commented that, we enhance our security by validating `msg.sender` directly, in addition to `tx.origin`, to prevent manipulation. This, along with checks on payment and parameter validity, adds an extra layer of security to our functions.

Identifier	Definition
COD-01	Note regarding keccak256 secure hashing

Note that the keccak256 function is not collision-resistant, and therefore there is a possibility of two different messages producing the same hash. Generating strong random input data, and properly securing and managing keys is recommended for fortification of keccak256.

Identifier	Definition	
COD-02	Timestamp manipulation via `block.timestamp` and `block.number`	Medium 🟡
COD-12	Randomness	

`randomGenerator()` function is using variables like `block.timestamp`, `block.number`, and `msg.sender` to generate a random number. While this method provides some level of uniqueness, it may not be sufficiently unpredictable, as parts of the input – such as message sender and timestamp of blocks are publicly visible on the blockchain.

RECOMMENDATION

Integrate with an external oracle like Chainlink VRF to supply provably-fair and verifiable random numbers. However, using external services introduces dependencies and potential points of failure.

ACKNOWLEDGEMENT

Sacred Tails team acknowledged this finding, and commented that, random numbers can't be 100% unpredictable on blockchain but the way we have implemented our logic makes it very hard for any potential timestamp manipulation.

Identifier	Definition	Severity
COD-03	Pricing logic	Minor 🟢

Pricing logic in `purchaseShinsei()` depends on the number of boxes, discount percentage, and potentially MATIC/USDT exchange rate.

RECOMMENDATION

Use an oracle for price feeds instead of relying on spot prices from liquidity pool. Oracles like Chainlink provide time-weighted average prices that are more resistant to manipulation.

Identifier	Definition	Severity
COD-10	Direct and indirect dependencies	Unknown ●

Smart contract is interacting with third party protocols e.g., Market Makers, External contracts such as Pair contract, Sushi-LP contract, USDT contract, Web 3 applications, OpenZeppelin tools. The scope of the audit treats these entities as black boxes and assumes their functional correctness. However, in the real world, all of them can be compromised, and exploited. Moreover, upgrades in these entities can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

RECOMMENDATION

Inspect third party dependencies regularly, and mitigate severe impacts whenever necessary.

ACKNOWLEDGEMENT

Sacred Tails team will inspect dependencies periodically, and provide amendments when possible.

Identifier	Definition	Severity
COD-13	Lack of event-driven architecture	Minor 🟢

Smart contract uses function calls to update state, which can make it difficult to track and analyze changes to the contract over time.

RECOMMENDATION

Use events to track state changes. Events improve transparency and provide a more granular view of contract activity.

Identifier	Definition	Severity
COM-01	Floating compiler status	Minor ●

Compiler is set to `^0.8.0`

RECOMMENDATION

Pragma should be fixed to the version that you’re indenting to deploy your contracts with.

RESOLUTION

Sacred Tails team will deploy smart contract with stable compiler.

DISCLAIMERS

InterFi Network provides the easy-to-understand audit of solidity source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high levels of technical risks and uncertainty. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. This audit report could include false positives, false negatives, and other unpredictable results.

CONFIDENTIALITY

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without InterFi Network's prior written consent.

NO FINANCIAL ADVICE

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. This audit report should not be used in any way to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

FOR AVOIDANCE OF DOUBT, SERVICES, INCLUDING ANY ASSOCIATED AUDIT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

TECHNICAL DISCLAIMER

ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, INTERFI NETWORK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO SERVICES, AUDIT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, INTERFI NETWORK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

WITHOUT LIMITING THE FOREGOING, INTERFI NETWORK MAKES NO WARRANTY OF ANY KIND THAT ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CLIENT'S OR ANY OTHER INDIVIDUAL'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.

TIMELINESS OF CONTENT

The content contained in this audit report is subject to change without any prior notice. InterFi Network does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.

LINKS TO OTHER WEBSITES

This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than InterFi Network. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites' and social accounts' owners. You agree that InterFi Network is not responsible for the content or operation of such websites and social accounts and that InterFi Network shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.

ABOUT INTERFI NETWORK

InterFi Network provides intelligent blockchain solutions. We provide solidity development, testing, and auditing services. We have developed 150+ solidity codes, audited 1000+ smart contracts, and analyzed 500,000+ code lines. We have worked on major public blockchains e.g., Ethereum, Binance, Cronos, Doge, Polygon, Avalanche, Metis, Fantom, Bitcoin Cash, Velas, Oasis, etc.

InterFi Network is built by engineers, developers, UI experts, and blockchain enthusiasts. Our team currently consists of 4 core members, and 6+ casual contributors.

Website: https://interfi.network

Email: hello@interfi.network

GitHub: https://github.com/interfinetwork

Telegram (Engineering): https://t.me/interfiaudits

Telegram (Onboarding): https://t.me/interfisupport interfinetwork

hello@interfi.network

https://interfi.network

SMART CONTRACT AUDITS | SOLIDITY DEVELOPMENT AND TESTING

RELENTLESSLY SECURING PUBLIC AND PRIVATE BLOCKCHAINS