



# SMART CONTRACT SECURITY AUDIT OF SQUIDWORLD STAKING CONTRACT



SMART CONTRACT AUDIT | SOLIDITY DEVELOPMENT & TESTING | PROJECT EVALUATION

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN

# Audit Introduction

<b>Auditing Firm</b>	InterFi Network
<b>Audit Architecture</b>	InterFi Echelon Auditing Standard
<b>Language</b>	Solidity
<b>Client Firm</b>	SquidWorld
<b>Website</b>	<a href="https://www.squidworld.io/">https://www.squidworld.io/</a>
<b>Telegram</b>	<a href="http://t.me/SquidWorldOfficial/">http://t.me/SquidWorldOfficial/</a>
<b>Twitter</b>	<a href="https://twitter.com/SquidWorldTeam/">https://twitter.com/SquidWorldTeam/</a>
<b>Report Date</b>	August 30, 2022

## About SquidWorld

SquidWorld.io Team aims to become one of the best (if not the only one) Squid Game related meme site that will incorporate all the standard offering of crypto related utilities while providing a safe, transparent and trust for our investors. SquidWorld's Swap, Staking, NFT marketplace, P2E mini games, Lottery and the ultimate SquidWorld Island Metaverse Game will see you (as our valued investors) competing for different prizes as well as the SquidWorld Grand Finale Prize of minimum 3% (of total SquidWorld tokens share) as the Netflix Squid Game Season 2 series is approaching! Join us on this exciting ride!



# Audit Summary

InterFi team has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks. According to the audit:

- ❖ SquidWorld's staking solidity source code has **LOW RISK SEVERITY**
- ❖ SquidWorld's staking smart contract has an **ACTIVE OWNERSHIP**
- ❖ SquidWorld's staking code's centralization risk is **HIGH**
- ❖ Important contract privileges – **ACTIVATE POOL, CHANGE POOL LIMIT**
- ❖ High-risk contract privilege – **CLAIM STUCK TOKENS**

Be aware that smart contracts deployed on the blockchain aren't resistant to internal exploit, external vulnerability, or hack. For a detailed understanding of risk severity, source code vulnerability, exploitability, and audit disclaimer, kindly refer to the audit.

- 📄 Contract address (7 days): **0xe6Cad28f74Ce8Da96309B45eD9a5ad7ddc9A0b7**
- 📄 Contract address (30 days): **0x7330CA800507AE4F0008CdD22Fbe4f8C3de5AaFD**
- 📄 Contract address (90 days): **0x36fc40750A526FbB6148d6E363CBE85BeC7551Eb**

🔗 Blockchain: **Binance Smart Chain**

- ✅ Verify the authenticity of this report on InterFi's GitHub: <https://github.com/interfinetwork>



# Table Of Contents

## **Audit Information**

Audit Scope .....	5
-------------------	---

## **Echelon Audit Standard**

Audit Methodology .....	6
Risk Classification .....	8
Centralization Risk .....	9

## **Smart Contract Risk Assessment**

Static Analysis .....	10
Software Analysis .....	12
Manual Analysis .....	14
SWC Attacks .....	18
Risk Status & Radar Chart .....	20

## **Audit Summary**

Auditor's Verdict .....	21
-------------------------	----

## **Legal Advisory**

Important Disclaimer .....	22
About InterFi Network .....	25



## Audit Scope

InterFi was consulted by SquidWorld to conduct the smart contract security audit of their solidity source codes. The audit scope of work is strictly limited to the mentioned solidity file(s) only:

❖ SquidWorldStaking.sol

## Audit Hash

Solidity source code is audited at hash #6a9b83e507456c5b11b44538e53ab100398de659

# InterFi

.....

## Smart Contract Security Audit



# Audit Methodology

The scope of this report is to audit smart contract sources code of SquidWorld's Staking. InterFi has scanned contracts and reviewed codes for common vulnerabilities, exploits, hacks, and backdoors. Due to being out of scope, InterFi has not tested contracts on testnet to assess any functional flaws. Smart contract audits are conducted using a set of standards and procedures. A mutual collaboration is essential to perform an effective smart contract audit. Here's a brief overview of auditing process and methodology:

## **Connect**

- ❖ 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 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.
- ❖ Manual line by line analysis is performed to identify contract issues and centralized privileges.



## **Report**

- ❖ Auditing team provides a preliminary report specifying all the checks which have been performed and findings thereof.
- ❖ Client's development team reviews the report, and makes amendments in solidity codes.
- ❖ Auditing team provides the final comprehensive report with open and unresolved issues.

## **Publish**

- ❖ Client may use the audit report internally or disclose it publicly.
- ❖ 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.

## **Automated 3P frameworks used to assess the smart contract vulnerabilities**

- ❖ Remix IDE Developer Tool
- ❖ Open Zeppelin Code Analyzer
- ❖ SWC Vulnerabilities Registry
- ❖ DEX Dependencies, e.g., Pancakeswap, Uniswap



# Risk Classification

Smart contracts are generally designed to manipulate and hold funds denominated in ETH/BNB. This makes them very tempting attack targets, as a successful attack may allow the attacker to directly steal funds from the contract. Below are the typical risk levels of a smart contract:

**Vulnerable:** A contract is vulnerable if it has been flagged by a static analysis tool as such. As we will see later, this means that some contracts may be vulnerable because of a false positive.

**Exploitable:** A contract is exploitable if it is vulnerable and the vulnerability could be exploited by an external attacker. For example, if the “vulnerability” flagged by a tool is in a function that requires owning the contract, it would be vulnerable but not exploitable.

**Exploited:** A contract is exploited if it received a transaction on the main network which triggered one of its vulnerabilities. Therefore, a contract can be vulnerable or even exploitable without having been exploited.

Risk severity	Meaning
<b>! High</b>	This level vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
<b>! Medium</b>	This level vulnerabilities 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.
<b>! Low</b>	This level vulnerabilities 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.
<b>! Informational</b>	This level vulnerabilities may be ignored. They are code style violations and deviations from standard practises.





# Centralization Risk

Centralization risk is the most common cause of decentralized finance hacks. When a smart contract has an active contract ownership, the risk related to centralization is elevated. There are some well-intended reasons to be an active contract owner, such as:

- ❖ Contract owner can be granted the power to `pause()` or `lock()` the contract in case of an external attack.
- ❖ Contract owner 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 a full centralized power to a single body can be dangerous. Unfortunately, centralization related risks are higher than common smart contract vulnerabilities. Centralization of ownership creates a risk of rug pull scams, where owners cash out tokens in such quantities that they become valueless. **Most important question to ask here is, how to mitigate centralization risk?** Here's InterFi's recommendation to lower the risks related to centralization hacks:


- ❖ Smart contract owner's private key must be carefully secured to avoid any potential hack.
- ❖ Smart contract ownership should be shared by multi-signature (multi-sig) wallets.
- ❖ Smart contract ownership can be locked in a contract, user voting, or community DAO can be introduced to unlock the ownership.

## Centralization Status












































- ❖ SquidWorld's staking smart contract has an **active ownership**.



# Static Analysis

Symbol	Meaning
	Function can modify state
	Function is payable
	Function is locked
	Function can be accessed
	Important functionality

```

| **Address** | Library | |||
| L | isContract | Internal  | | |
| L | sendValue | Internal   | | |
| L | functionCall | Internal   | | |
| L | functionCall | Internal   | | |
| L | functionCallWithValue | Internal   | | |
| L | functionCallWithValue | Internal   | | |
| L | functionStaticCall | Internal  | | |
| L | functionStaticCall | Internal  | | |
| L | functionDelegateCall | Internal   | | |
| L | functionDelegateCall | Internal   | | |
| L | verifyCallResult | Internal  | | |
| |||||
| **Context** | Implementation | |||
| L | _msgSender | Internal  | | |
| L | _msgData | Internal  | | |
| |||||
| **IERC20** | Interface | |||
| L | totalSupply | External  | | NO  |
| L | decimals | External  | | NO  |
| L | balanceOf | External  | | NO  |
| L | transfer | External   | NO  |
| L | allowance | External  | | NO  |
| L | approve | External   | NO  |
| L | transferFrom | External   | NO  |
| |||||
| **Ownable** | Implementation | Context |||
| L | <Constructor> | Public   | NO  |
| L | owner | Public  | | NO  |
| L | renounceOwnership | Public   | onlyOwner |
| L | transferOwnership | Public   | onlyOwner |

```



```

| L | _transferOwnership | Internal | 🔒 | 🔴 | |
|||||
| **SquidStake** | Implementation | Ownable, ReentrancyGuard | |||
| L | <Constructor> | Public | ! | 🔴 | NO ! |
| L | <Receive Ether> | External | ! | 🔒 | NO ! |
| L | poolActivation | External | ! | 🔴 | onlyOwner |
| L | changePoolLimit | External | ! | 🔴 | onlyOwner |
| L | changeFeeWallet | External | ! | 🔴 | onlyOwner |
| L | PoolStake | External | ! | 🔴 | nonReentrant |
| L | claimPool | External | ! | 🔴 | nonReentrant |
| L | calculateRewards | Internal | 🔒 | | |
| L | rewardsCalculate | Public | ! | | NO ! |
| L | maxPayoutOf | External | ! | | NO ! |
| L | tokenBalance | Public | ! | | NO ! |
| L | ethBalance | Public | ! | | NO ! |
| L | retrieveEthStuck | External | ! | 🔴 | nonReentrant onlyOwner |
| L | retrieveERC20TokenStuck | External | ! | 🔴 | nonReentrant onlyOwner |
| L | maturityDate | Public | ! | | NO ! |
| L | fullMaturityReward | Public | ! | | NO ! |
| L | isContract | Internal | 🔒 | | |
|||||
| **ReentrancyGuard** | Implementation | | |||
| L | <Constructor> | Public | ! | 🔴 | NO ! |
|||||
| **SafeERC20** | Library | | |||
| L | safeTransfer | Internal | 🔒 | 🔴 | |
| L | safeTransferFrom | Internal | 🔒 | 🔴 | |
| L | safeApprove | Internal | 🔒 | 🔴 | |
| L | safeIncreaseAllowance | Internal | 🔒 | 🔴 | |
| L | safeDecreaseAllowance | Internal | 🔒 | 🔴 | |
| L | _callOptionalReturn | Private | 🔒 | 🔴 | |
|||||
| **SafeMath** | Library | | |||
| L | tryAdd | Internal | 🔒 | | |
| L | trySub | Internal | 🔒 | | |
| L | tryMul | Internal | 🔒 | | |
| L | tryDiv | Internal | 🔒 | | |
| L | tryMod | Internal | 🔒 | | |
| L | add | Internal | 🔒 | | |
| L | sub | Internal | 🔒 | | |
| L | mul | Internal | 🔒 | | |
| L | div | Internal | 🔒 | | |
| L | mod | Internal | 🔒 | | |
| L | sub | Internal | 🔒 | | |
| L | div | Internal | 🔒 | | |
| L | mod | Internal | 🔒 | | |

```



# Software Analysis

## Function Signatures

```

16279055 => isContract(address)
17686422 => calculateRewards(uint256,address)
24a084df => sendValue(address,uint256)
a0b5ffb0 => functionCall(address,bytes)
241b5886 => functionCall(address,bytes,string)
2a011594 => functionCallWithValue(address,bytes,uint256)
d525ab8a => functionCallWithValue(address,bytes,uint256,string)
c21d36f3 => functionStaticCall(address,bytes)
dbc40fb9 => functionStaticCall(address,bytes,string)
ee33b7e2 => functionDelegateCall(address,bytes)
57387df0 => functionDelegateCall(address,bytes,string)
946b5793 => verifyCallResult(bool,bytes,string)
119df25f => _msgSender()
8b49d47e => _msgData()
18160ddd => totalSupply()
313ce567 => decimals()
70a08231 => balanceOf(address)
a9059cbb => transfer(address,uint256)
dd62ed3e => allowance(address,address)
095ea7b3 => approve(address,uint256)
23b872dd => transferFrom(address,address,uint256)
8da5cb5b => owner()
715018a6 => renounceOwnership()
f2fde38b => transferOwnership(address)
d29d44ee => _transferOwnership(address)
4262afc0 => poolActivation(bool)
63760f73 => changePoolLimit(uint256)
3e4d0310 => changeFeeWallet(address)
99fcb5d8 => PoolStake(uint256)
b6bfd24b => claimPool()
6d26538f => rewardsCalculate(address)
8959af3c => maxPayoutOf(uint256)
eedc966a => tokenBalance(address)
4e6630b0 => ethBalance()
5eac4247 => retrieveEthStuck()
65415c48 => retrieveERC20TokenStuck(address,uint256)
66ac49c9 => maturityDate(address)
38ec9691 => fullMaturityReward(address)
d0c407e1 => safeTransfer(IERC20,address,uint256)
5beae096 => safeTransferFrom(IERC20,address,address,uint256)
d6dcec8d => safeApprove(IERC20,address,uint256)
390cc046 => safeIncreaseAllowance(IERC20,address,uint256)
5164ffed => safeDecreaseAllowance(IERC20,address,uint256)
becc5a20 => _callOptionalReturn(IERC20,bytes)
884557bf => tryAdd(uint256,uint256)

```

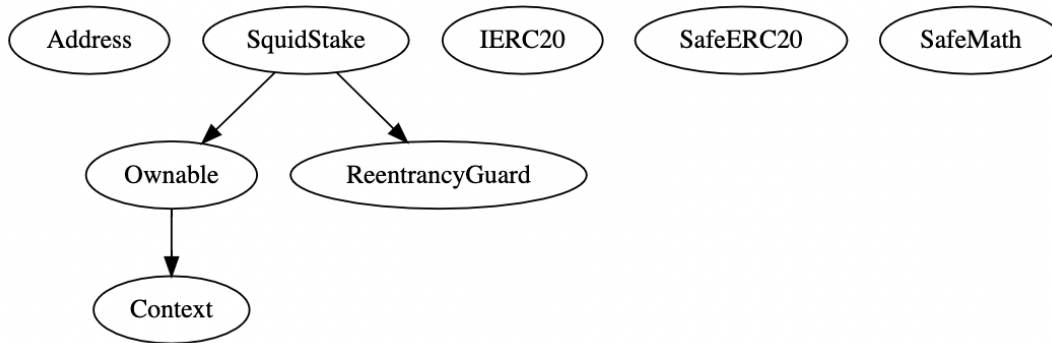


```

a29962b1 => trySub(uint256,uint256)
6281efa4 => tryMul(uint256,uint256)
736ecb18 => tryDiv(uint256,uint256)
38dc0867 => tryMod(uint256,uint256)
771602f7 => add(uint256,uint256)
b67d77c5 => sub(uint256,uint256)
c8a4ac9c => mul(uint256,uint256)
a391c15b => div(uint256,uint256)
f43f523a => mod(uint256,uint256)
e31bdc0a => sub(uint256,uint256,string)
b745d336 => div(uint256,uint256,string)
71af23e8 => mod(uint256,uint256,string)

```

### Inheritance Graph



## Smart Contract Security Audit



# Manual Analysis

Function	Description	Available	Status
<b>Total Supply</b>	provides information about the total token supply	Yes	Passed
<b>Balance Of</b>	provides account balance of the owner's account	Yes	Passed
<b>Transfer</b>	executes transfers of a specified number of tokens to a specified address	Yes	Passed
<b>Approve</b>	allow a spender to withdraw a set number of tokens from a specified account	Yes	Passed
<b>Allowance</b>	returns a set number of tokens from a spender to the owner	Yes	Passed
<b>Staking</b>	executes transfers of a specified stake reward to a specified address	Yes	Passed
<b>Transfer Ownership</b>	executes transfer of contract ownership to a specified wallet	Yes	Passed
<b>Renounce Ownership</b>	executes transfer of contract ownership to a dead address	Yes	Passed



## Notable Information

- ❖ Smart contract owner can **activate stake pool** and **change pool limit**.

```
function poolActivation(bool status) external onlyOwner {
    PoolInfo storage pool = poolInfo[_poolId];
    pool.active = status;
function changePoolLimit(uint256 amount) external onlyOwner {
    PoolInfo storage pool = poolInfo[_poolId];
    pool.poolLimit = amount * 10 ** (pool.stakeToken).decimals();
```

- ❖ Smart contract utilizes **safemath** function to avoid common smart contract vulnerabilities.

```
string private _name = "SquidWorldStaking";
library SafeMath {
function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    require(c >= a, "SafeMath: addition overflow");
function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    return sub(a, b, "SafeMath: subtraction overflow");
    uint256 c = a * b;
    require(c / a == b, "SafeMath: multiplication overflow");
    return c;
function div(uint256 a, uint256 b) internal pure returns (uint256) {
    return div(a, b, "SafeMath: division by zero");
function mod(uint256 a, uint256 b) internal pure returns (uint256) {
    return mod(a, b, "SafeMath: modulo by zero");
```

- ❖ Smart contract utilizes **re-entrancy guard** to prevent re-entrant calls.
- ❖ Smart contract owner can call `receive()` for fallbacks. It is executed on a call to the contract with empty call data. Make sure the contract can receive token through a regular transaction, and does not throw an exception.

```
receive() external payable {}
```



- ❖ Smart contract utilizes **redundant code** for `transferOwnership()`. Ideal transfer ownership code should look be written like:

```
function transferOwnership(address newOwner) public virtual onlyOwner {
    require(newOwner != address(0), "Ownable: new owner is the zero address");
    emit OwnershipTransferred(_owner, newOwner);
    _owner = newOwner;
}
```

- ❖ Smart contract **charges penalty fee** for pools claimed before maturity time.

```
uint256 penaltyAmount;

if (
    block.timestamp < (user.pool_deposit_time + pool.fullMaturityTime)
    calculatedRewards == 0;
    penaltyAmount = ((amount) * pool.poolPenaltyPercent) / 1000;
    if(penaltyAmount>0){
        pool.rewardToken.safeTransfer(penaltyFeeAddress, penaltyAmount);
        user.penaltyGiven += penaltyAmount;
        amount = amount.sub(penaltyAmount);
    }
}
```

- ❖ Smart contract owner can claim **stuck eth** and **stuck tokens** from the contract. Native tokens should not be available to withdraw from the contract.

```
function retrieveEthStuck()
    external
    nonReentrant
    onlyOwner
    returns (bool)
    payable(owner()).transfer(address(this).balance);
    return true;

function retrieveERC20TokenStuck(
    address _tokenAddr,
    uint256 amount
) external nonReentrant onlyOwner returns (bool) {
    IERC20(_tokenAddr).transfer(owner(), amount);
}
```





- ❖ Smart contract has an **informational severity issue** which may or may not create any functional vulnerability.

"Floating Pragma"

- ❖ Smart contract has an **informational severity issue** which may or may not create any functional vulnerability.

"Reward calculations may throw incorrect results"

- ❖ Smart contract has a **low severity issue** which may or may not create any functional vulnerability.

"Utilization of block.timestamp"

# InterFi

.....

## Smart Contract Security Audit



# SWC Attacks

SWC ID	Description	Status
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	Passed
SWC-103	Floating Pragma	! Informational
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELF-DESTRUCT Instruction	Passed
SWC-107	Re-entrancy	! Informational
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegate Call to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	! Low
SWC-117	Signature Malleability	Passed
SWC-118	Incorrect Constructor Name	Passed

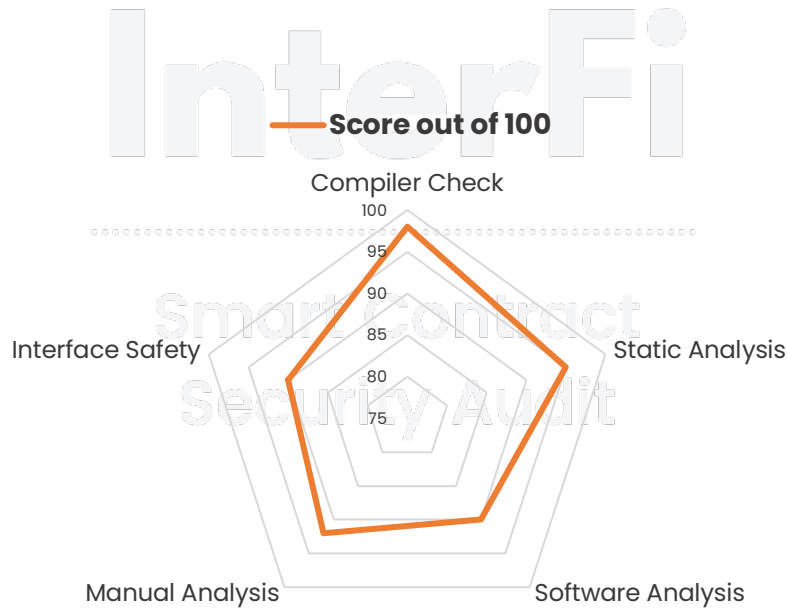


<b>SWC-119</b>	Shadowing State Variables	<b>Passed</b>
<b>SWC-120</b>	Weak Sources of Randomness from Chain Attributes	<b>Passed</b>
<b>SWC-121</b>	Missing Protection against Signature Replay Attacks	<b>Passed</b>
<b>SWC-122</b>	Lack of Proper Signature Verification	<b>Passed</b>
<b>SWC-123</b>	Requirement Violation	<b>Passed</b>
<b>SWC-124</b>	Write to Arbitrary Storage Location	<b>Passed</b>
<b>SWC-125</b>	Incorrect Inheritance Order	<b>Passed</b>
<b>SWC-126</b>	Insufficient Gas Griefing	<b>Passed</b>
<b>SWC-127</b>	Arbitrary Jump with Function Type Variable	<b>Passed</b>
<b>SWC-128</b>	DoS With Block Gas Limit	<b>Passed</b>
<b>SWC-129</b>	Typographical Error	<b>Passed</b>
<b>SWC-130</b>	Right-To-Left-Override control character (U+202E)	<b>Passed</b>
<b>SWC-131</b>	Presence of unused variables	<b>Passed</b>
<b>SWC-132</b>	Unexpected Ether balance	<b>Passed</b>
<b>SWC-133</b>	Hash Collisions With Multiple Variable Length Arguments	<b>Passed</b>
<b>SWC-134</b>	Message call with the hardcoded gas amount	<b>Passed</b>
<b>SWC-135</b>	Code With No Effects (Irrelevant/Dead Code)	<b>! Informational</b>
<b>SWC-136</b>	Unencrypted Private Data On-Chain	<b>Passed</b>



# Risk Status & Radar Chart

Risk Severity	Status
High	No high severity issues identified
Medium	No medium severity issues identified
Low	2 low severity issues identified
Informational	3 informational severity issues identified
Centralization Risk	Active contract ownership identified



## Auditor's Verdict

InterFi team has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks. According to the audit:

- ❖ SquidWorld's staking solidity source code has **LOW RISK SEVERITY**
- ❖ SquidWorld's staking smart contract has an **ACTIVE OWNERSHIP**
- ❖ SquidWorld's staking code's centralization risk is **HIGH**

# InterFi

.....

### Note for stakeholders

## Smart Contract Security Audit

- ❖ Be aware that active smart contract owner privileges constitute an elevated impact on smart contract safety and security.
- ❖ If the smart contract is not deployed on any blockchain at the time of the audit, the contract can be modified or altered before blockchain development. Verify contract's deployment status in the audit report.
- ❖ Make sure that the project team's KYC/identity is verified by an independent firm.
- ❖ Always check if the contract's liquidity is locked. A longer liquidity lock plays an important role in the project's longevity. It is recommended to have multiple liquidity providers.
- ❖ Examine the unlocked token supply in the owner, developer, or team's private wallets. Understand the project's tokenomics, and make sure the tokens outside of the LP Pair are vested or locked for a longer period.



# IMPORTANT DISCLAIMER

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.

## Smart Contract Security Audit

### **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. InterFi is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. **InterFi's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy to use.**

InterFi is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 6+ core team members, and 10+ casual contributors. **InterFi provides manual, static, and automatic smart contract analysis, to ensure that project is checked against known attacks and potential vulnerabilities.**

To learn more, visit <https://interfi.network>

To view our audit portfolio, visit <https://github.com/interfinetwork>

To book an audit, message <https://t.me/interfiaudits>





RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA 