



**InterFi**  
NETWORK



# **SMART CONTRACT SECURITY AUDIT ZOO PARTY**



**SMART CONTRACT AUDIT | TEAM KYC | PROJECT EVALUATION**

**RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA** 

# Summary

<b>Auditing Firm</b>	InterFi Network
<b>Architecture</b>	InterFi "Echelon" Auditing Standard
<b>Smart Contract Audit Approved By</b>	Chris   Blockchain Specialist at InterFi Network
<b>Project Overview Approved BY</b>	Albert   Project Specialist at InterFi Network
<b>Platform</b>	Solidity
<b>Audit Check (Mandatory)</b>	Static, Software, Auto Intelligent & Manual Analysis
<b>Project Check (Optional)</b>	KYC, Website & Socials Analysis (Not Applicable)
<b>Consultation Request Date</b>	September 25, 2021
<b>Report Date</b>	September 29, 2021

## Audit Summary

### Smart Contract Security Audit

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

- ❖ **Zoo Party's smart contract source code has **LOW RISK SEVERITY**.**
- ❖ **Zoo Party has successfully **PASSED** the smart contract audit.**

For the detailed understanding of risk severity, source code vulnerability, and functional test, kindly refer to the audit.



# Table Of Contents

## **Project Information**

Overview .....	4
----------------	---

## **InterFi “Echelon” Audit Standard**

Audit Scope & Methodology .....	6
InterFi’s Risk Classification.....	8

## **Smart Contract Risk Assessment**

Contract Overview .....	9
Static Analysis.....	10
Software Analysis .....	12
Manual Analysis.....	13
SWC Attacks.....	15
Risk Status & Radar Chart.....	17

## **Report Summary**

Auditor’s Verdict .....	18
-------------------------	----

## **Legal Advisory**

Important Disclaimer .....	19
About InterFi Network.....	20



# Project Overview

InterFi was consulted by Zoo Party on September 25, 2021 to conduct a smart contract security audit of their token source code.

Zoo Party is a community-driven Free to Play to Earn (F2P2E) game empowering users by rewarding them for their engagement and enjoyment. It is also a multiplayer RPG game where users can collect, breed, discover, and even sell virtual animals.

<b>Project</b>	ZOO PARTY
<b>Blockchain</b>	Binance Smart Chain / Binance Blockchain Explorer
<b>Language</b>	Solidity
<b>Contract</b>	0x2e2EF266FC5b710a22679Fb06b8c27c223e43D13
<b>Website</b>	<a href="https://zooparty.io">https://zooparty.io</a>
<b>Whitepaper</b>	<a href="https://zooparty.io/whitepaper.pdf">https://zooparty.io/whitepaper.pdf</a>
<b>Instagram</b>	<a href="https://instagram.com/zooparty.io/">https://instagram.com/zooparty.io/</a>
<b>Twitter</b>	<a href="https://twitter.com/ZooPartyio">https://twitter.com/ZooPartyio</a>
<b>Medium</b>	<a href="https://zooparty.medium.com">https://zooparty.medium.com</a>
<b>Telegram</b>	<a href="https://t.me/zooparty.io">https://t.me/zooparty.io</a>
<b>Contact</b>	<a href="mailto:metin@zooparty.io">metin@zooparty.io</a>



## Public logo



## Contract Source Code On Blockchain (BscScan Verified With Exact Match)

<https://bscscan.com/address/0x2e2ef266fc5b710a22679fb06b8c27c223e43d13#code>

Compiler Version: v0.8.7+commit.e28d00a7

Optimization Enabled: No with 200 runs

## Solidity Source Code On InterFi GitHub

<https://github.com/interfinetwork/audited-codes/blob/main/ZooParty.sol>

## GitHub Commits

Solidity source code committed at: 21b196f0482095d8d1c5a6cc0a8884da7c79f49e



# Audit Scope & Methodology

The scope of this report is to audit the smart contract source code of Zoo Party. The source code can be viewed in its entirety on

<https://github.com/interfinetwork/audited-codes/blob/main/ZooParty.sol>

InterFi has scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

## Category

---

### Smart Contract Vulnerabilities

- ❖ Re-entrancy (RE)
- ❖ Unhandled Exceptions (UE)
- ❖ Transaction Order Dependency (TO)
- ❖ Integer Overflow (IO)
- ❖ Unrestricted Action (UA)
- ❖ Ownership Takeover
- ❖ Gas Limit and Loops

### Source Code Review

- ❖ Deployment Consistency
- ❖ Repository Consistency
- ❖ Data Consistency
- ❖ Token Supply Manipulation
- ❖ Access Control and Authorization
- ❖ Operations Trail and Event Generation

### Functional Assessment

- ❖ Assets Manipulation
- ❖ Liquidity Access



## **InterFi's Echelon Audit Standard**

The aim of InterFi's "Echelon" standard is to analyze the smart contract and identify the vulnerabilities and the hacks in the smart contract. Mentioned are the steps used by ECHELON-1 to assess the smart contract:

1. Solidity smart contract source code reviewal:
  - ❖ Review of the specifications, sources, and instructions provided to InterFi to make sure we understand the size, scope, and functionality of the smart contract.
  - ❖ Manual review of code, which is the process of reading source code line-byline to identify potential vulnerabilities.
2. Static, Manual, and Automated AI analysis:
  - ❖ Test coverage analysis, which is the process of determining whether the test cases are covering the code and how much code is exercised when we run those test cases.
  - ❖ Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts

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

- ❖ Slither
- ❖ Consensys MythX
- ❖ Consensys Surya
- ❖ Open Zeppelin Code Analyzer
- ❖ Solidity Code Compiler



# InterFi's 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 which requires to own 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
! Critical	This level vulnerabilities could be exploited easily, and can lead to asset loss, data loss, asset manipulation, or data manipulation. They should be fixed right away.
! High	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 critical risk severity
! Medium	This level vulnerabilities are 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	This level vulnerabilities can be ignored. They are code style violations, and informational statements in the code. They may not affect the smart contract execution





# Smart Contract – Overview

## Contract information

Query	Result
Name	Zoo Party
Symbol	ZOOP
Decimals	6
Total Supply	200,000,000

```
contract ZooParty is ERC20 {  
    constructor() ERC20("Zoo Party", "ZOOP") {  
        _mint(msg.sender, 200000000 * 10 ** uint(decimals()));  
    }  
}
```












## Smart Contract Security Audit



# Smart Contract – Static Analysis

Symbol	Meaning
	Function can be modified
	Function is payable
	Function is locked
	Function can be accessed
!	Important functionality

```

| **IERC20** | Interface | |||
| L | totalSupply | External ! | |NO ! |
| L | balanceOf | External ! | |NO ! |
| L | transfer | External ! |  |NO ! |
| L | allowance | External ! | |NO ! |
| L | approve | External ! |  |NO ! |
| L | transferFrom | External ! |  |NO ! |
| |||||
| **IERC20Metadata** | Interface | IERC20 |||
| L | name | External ! | |NO ! |
| L | symbol | External ! | |NO ! |
| L | decimals | External ! | |NO ! |
| |||||
| **Context** | Implementation | |||
| L | _msgSender | Internal  | | |
| L | _msgData | Internal  | | |
| |||||
| **ERC20** | Implementation | Context, IERC20, IERC20Metadata |||
| L | <Constructor> | Public ! |  |NO ! |
| L | name | Public ! | |NO ! |
| L | symbol | Public ! | |NO ! |
| L | decimals | Public ! | |NO ! |
| L | totalSupply | Public ! | |NO ! |
| L | balanceOf | Public ! | |NO ! |
| L | transfer | Public ! |  |NO ! |
| L | allowance | Public ! | |NO ! |
| L | approve | Public ! |  |NO ! |
| L | transferFrom | Public ! |  |NO ! |
| L | increaseAllowance | Public ! |  |NO ! |
| L | decreaseAllowance | Public ! |  |NO ! |

```



```
| L | _transfer | Internal | 🔒 | 🔴 | | |
| L | _mint | Internal | 🔒 | 🔴 | | |
| L | _burn | Internal | 🔒 | 🔴 | | |
| L | _approve | Internal | 🔒 | 🔴 | | |
| L | _beforeTokenTransfer | Internal | 🔒 | 🔴 | | |
| L | _afterTokenTransfer | Internal | 🔒 | 🔴 | | |
|||||
| **ZooParty** | Implementation | ERC20 | |||
| L | <Constructor> | Public | ! | 🔴 | ERC20 |
```

# InterFi

## Smart Contract Security Audit

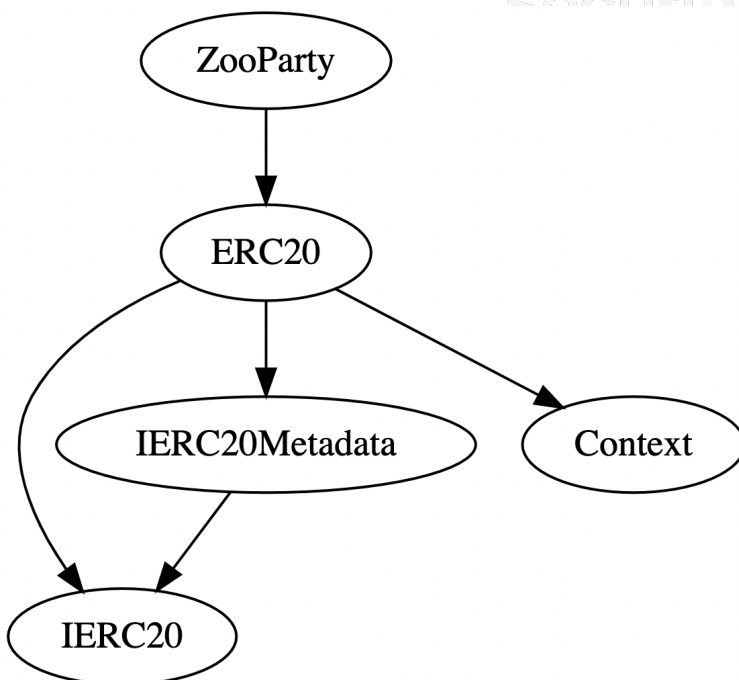


# Smart Contract – Software Analysis

## Callout functions – Sighash

Sighash	Function Signature
39509351	=> increaseAllowance(address,uint256)
18160ddd	=> totalSupply()
70a08231	=> balanceOf(address)
a9059cbb	=> transfer(address,uint256)
dd62ed3e	=> allowance(address,address)
095ea7b3	=> approve(address,uint256)
23b872dd	=> transferFrom(address,address,uint256)
06fdde03	=> name()
95d89b41	=> symbol()
313ce567	=> decimals()
119df25f	=> _msgSender()
8b49d47e	=> _msgData()
a457c2d7	=> decreaseAllowance(address,uint256)
30e0789e	=> _transfer(address,address,uint256)
4e6ec247	=> _mint(address,uint256)
6161eb18	=> _burn(address,uint256)
104e81ff	=> _approve(address,address,uint256)
cad3be83	=> _beforeTokenTransfer(address,address,uint256)
8f811a1c	=> _afterTokenTransfer(address,address,uint256)

## Callout functions – Inheritance Graph



# Smart Contract – Manual Analysis

Function	Description	Tested	Verdict
<b>TotalSupply</b>	provides information about the total token supply	Yes	<b>Passed</b>
<b>BalanceOf</b>	provides account balance of the owner's account	Yes	<b>Passed</b>
<b>Transfer</b>	executes transfers of a specified number of tokens to a specified address	Yes	<b>Passed</b>
<b>Approve</b>	allow a spender to withdraw a set number of tokens from a specified account	Yes	<b>Passed</b>
<b>Allowance</b>	returns a set number of tokens from a spender to the owner	Yes	<b>Passed</b>
<b>burn</b>	executes transfers of a specified number of tokens to a burn address	NA	NA

## Verified

- ❖ Owner can mint tokens at token launch.
- ❖ Owner can-not lock or burn user assets.
- ❖ Current Supply Owner: **0x5A68e6Eba67a2c8f3b7e056091e2c83305B32b1F**



## Important Information

**Zoo Party smart contract has 1 medium severity issue which may or may not create any functional vulnerability.**

- ❖ Zoo Party smart contract **does not** utilize "SafeMath" to prevent known vulnerabilities.

**Zoo Party smart contract has 1 low severity issue which may or may not create any functional vulnerability.**

- ❖ Expected pragma, import directive or contract/interface/library definition.

```
{
  "resource": "/Users/InterFi Cloud Data/ZooParty.sol",
  "owner": "_generated_diagnostic_collection_name_#0",
  "severity": 8,
  "message": "Expected pragma, import directive or contract/interface/library
definition.",
  "source": "solc",
  "startLineNumber": 123,
  "startColumn": 1,
  "endLineNumber": 123,
  "endColumn": 2
}
```



# Smart Contract – SWC Attacks

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



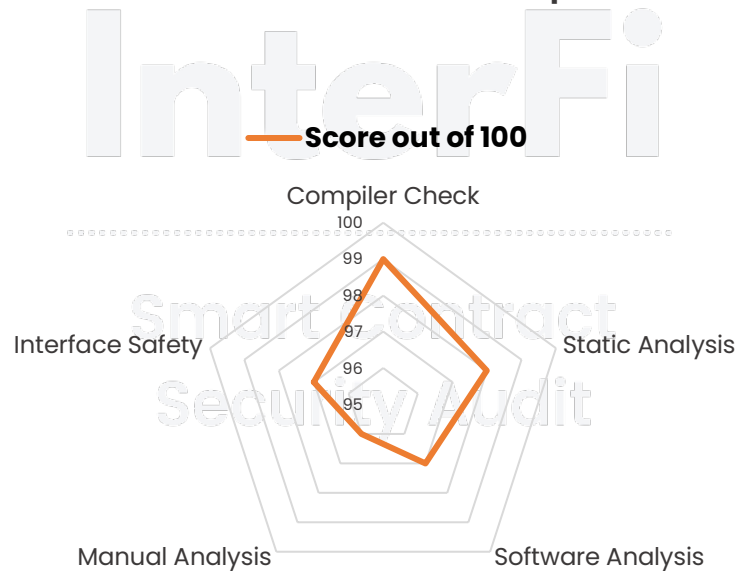
<b>SWC-119</b>	Shadowing State Variables	<b>Passed</b>
<b>SWC-120</b>	Weak Sources of Randomness from Chain Attributes	<b>Passed</b>
<b>SWC-121</b>	Missing Protection against Signature Replay Attacks	<b>Passed</b>
<b>SWC-122</b>	Lack of Proper Signature Verification	<b>Passed</b>
<b>SWC-123</b>	Requirement 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 hardcoded gas amount	<b>Passed</b>
<b>SWC-135</b>	Code With No Effects (Irrelevant/Dead Code)	<b>Passed</b>
<b>SWC-136</b>	Unencrypted Private Data On-Chain	<b>Passed</b>





# Smart Contract - Risk Status & Radar Chart

Risk Severity	Status
<b>! Critical</b>	None critical severity issues identified
<b>! High</b>	None high severity issues identified
<b>! Medium</b>	<b>1 Medium severity issue identified</b>
<b>! Low</b>	<b>1 Low severity issue identified</b>
<b>Passed</b>	<b>39 functions and instances verified and passed</b>



Compiler Check 99

Static Analysis 98

Software Analysis 97

Manual Analysis 96

Interface Safety 98



## Auditor's Verdict

InterFi team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks.

**Zoo Party's source code does not use "SafeMath" function to prevent known smart contract vulnerabilities.**

**Zoo Party's smart contract source code has LOW RISK SEVERITY.**

**Zoo Party has successfully PASSED the smart contract audit.**

# InterFi

## Smart Contract Security Audit

### General Note:

- ❖ Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security.
- ❖ Owner or developer KYC isn't checked and verified due to out of scope.
- ❖ Project's liquidity pair isn't checked and verified due to out of scope.
- ❖ Project website is not checked due to out of scope. The website hasn't been reviewed for SSL and lighthouse report.



# Important Disclaimer

InterFi Network provides contract auditing and project verification services for blockchain projects. The purpose of the audit is to analyse the on-chain smart contract source code, and to provide basic overview of the project. **This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purposes without InterFi's prior written consent.**

InterFi provides the easy-to-understand assessment of the project, and the smart contract (otherwise known as the source code). The audit makes no statements or warranties on the security of the code. It also cannot be considered as an enough assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have used all the data at our disposal to provide the transparent analysis, it is important to note that you should not rely on this report only – we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. **Be aware that smart contracts deployed on a blockchain aren't resistant from external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security. Therefore, InterFi does not guarantee the explicit security of the audited smart contract.**

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

**This report should not be considered as an endorsement or disapproval of any project or team.**

The information provided on 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. Do conduct your own due diligence and consult your financial advisor before making any investment decisions.



# 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>





**@INTERFINETWORK**

**RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA 🇨🇦**