



# SMART CONTRACT AUDIT

 interfinetwork

 hello@interfi.network

 <https://interfi.network>

PREPARED FOR

**PIZZABRICK**



# INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	PizzaBrick
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	0x9a76148428230e4Ae4aea554B8c843fa6314226d
Blockchain	Binance Smart Chain
Centralization	Active Ownership
Commit	9d1901df81081564dc2dd7fcc018bb3e3fedc6f2
Telegram#1	<a href="https://t.me/PizzaBrick">https://t.me/PizzaBrick</a>
Telegram#2	<a href="https://t.me/pizza_brick_chat">https://t.me/pizza_brick_chat</a>
Twitter	<a href="https://twitter.com/PizzaBrickk">https://twitter.com/PizzaBrickk</a>
Report Date	November 01, 2023

 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	Major 🟡	Medium 🟠	Minor 🟢	Unknown 🟤	Informational 🟡
Open	0	1	3	0	2
Acknowledged	0	2	1	1	2
Resolved	0	1	2	0	0
Noteworthy Functions	Withdraw Money, Buy Bricks, Create Duel, Create Lottery				

📌 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.

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



# TABLE OF CONTENTS

TABLE OF CONTENTS .....	4
SCOPE OF WORK.....	5
AUDIT METHODOLOGY .....	6
RISK CATEGORIES .....	8
CENTRALIZED PRIVILEGES .....	9
AUTOMATED ANALYSIS.....	10
MANUAL REVIEW.....	13
DISCLAIMERS .....	30
ABOUT INTERFI NETWORK .....	33

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL



## SCOPE OF WORK

InterFi was consulted by PizzaBrick 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:

- PizzeriaGame.sol

 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:

Public Contract Link	
<a href="https://bscscan.com/address/0x9a76148428230e4ae4aea554b8c843fa6314226d#code">https://bscscan.com/address/0x9a76148428230e4ae4aea554b8c843fa6314226d#code</a>	
Contract Name	PizzeriaGame
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	<ul style="list-style-type: none"><li>○ Token Supply Manipulation</li><li>○ Access Control and Authorization</li><li>○ Assets Manipulation</li><li>○ Ownership Control</li><li>○ Liquidity Access</li><li>○ Stop and Pause Trading</li><li>○ Ownable Library Verification</li></ul>
----------------------	---



## 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.

 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
Major 🟡	These risks can be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
Medium 🟠	These risks are very important to fix, they carry an elevated risk of smart contract exploitation. This severity re-entrancy attacks are potent.
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.
Informational ⚪	These risks do not pose any risk to the contract or those who interact with it. These are compiler version, optimization and gas-related issues.

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.

 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

```

| **ReentrancyGuard** | Implementation | |||
| L | <Constructor> | Public ! |  | NO ! |
| L | _nonReentrantBefore | Private  |  | |
| L | _nonReentrantAfter | Private  |  | |
| L | _reentrancyGuardEntered | Internal  | | |
|||||
| **PizzeriaGame** | Implementation | ReentrancyGuard |||
| L | <Constructor> | Public ! |  | NO ! |
| L | buyBricks | External ! |  | nonReentrant |
| L | withdrawMoney | External ! |  | nonReentrant |
| L | getTotalPizzaPerHour | Public ! | | NO ! |
| L | _syncOven | Internal  |  | isRegistered |
| L | collectPizza | External ! |  | NO ! |
| L | swapPizzaToBricks | External ! |  | NO ! |
| L | upgradePizzeria | External ! |  | NO ! |
| L | sellPizzerias | External ! |  | NO ! |
| L | upgradeOven | External ! |  | NO ! |
| L | getPizzerias | External ! | | NO ! |

```



```

| L | upgradeCookingSpeed | External ! | 🔴 | NO ! |
| L | createDuel | External ! | 🔴 | isRegistered |
| L | _randomNumber | Internal 🔒 | | |
| L | getDuelType | Internal 🔒 | | |
| L | _createDuel | Internal 🔒 | 🔴 | |
| L | _joinDuel | Internal 🔒 | 🔴 | |
| L | _fightDuel | Internal 🔒 | 🔴 | |
| L | isValidPrice | Public ! | | NO ! |
| L | isValidParticipantsCount | Public ! | | NO ! |
| L | hasJoinedLottery | Public ! | | NO ! |
| L | getLotteryParticipantsCount | Public ! | | NO ! |
| L | createLottery | External ! | 🔴 | isRegistered |
| L | getActiveLotteries | External ! | | NO ! |

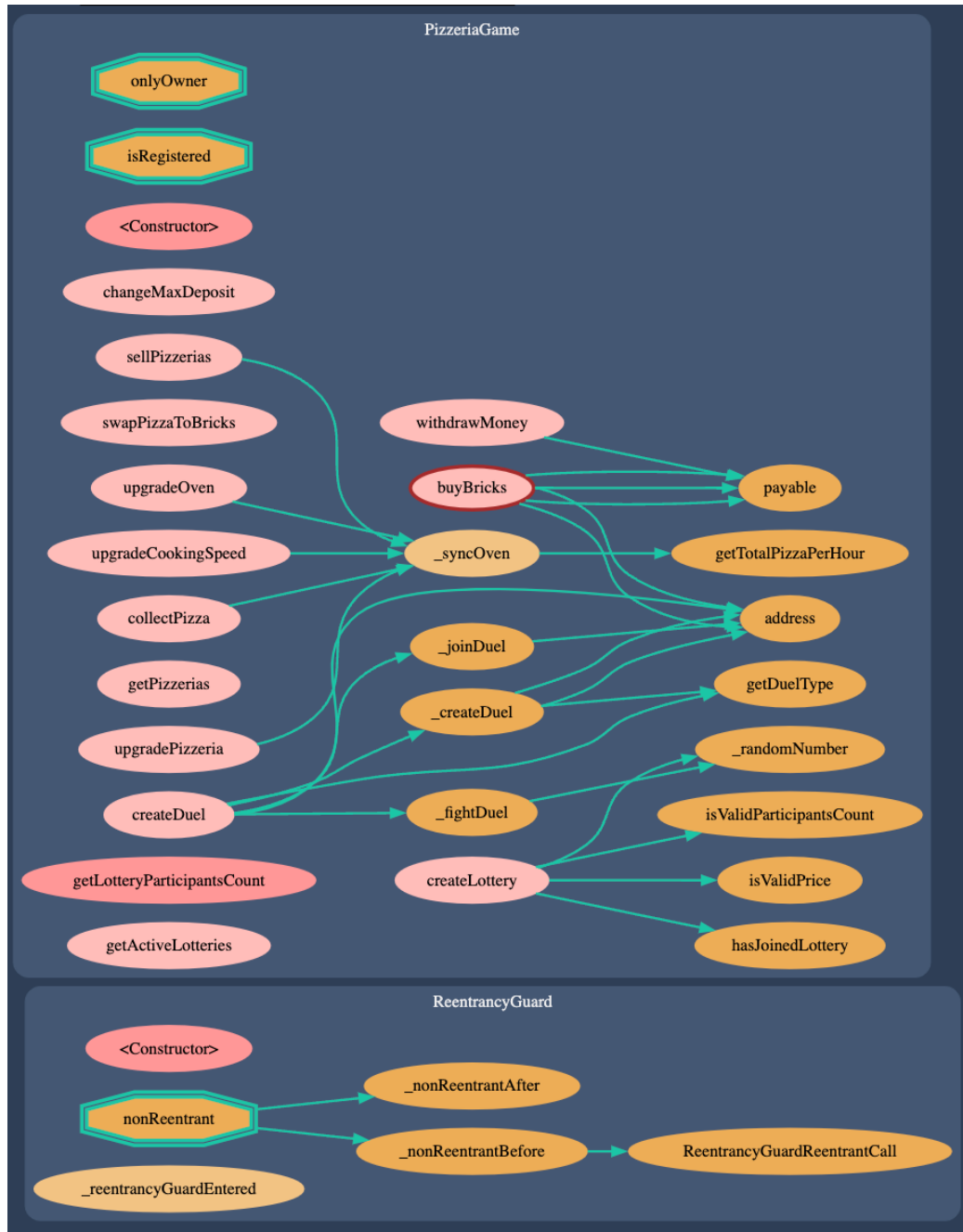
```

TERFI  
CONFIDENTIAL

INTERFI  
CONFIDENTIAL



# CALLOUT GRAPH



## MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized privileges	Medium ●
CEN-07	Authorizations and access control	
CEN-12	Hardcoded addresses	

buyBricks() and createLottery() functions send 6% of investment to different addresses (owner, manager1 and manager2). These central point of control poses some risks, as these addresses may be able to influence the game significantly by collecting a large portion of the funds.

```
address public owner = 0x8c23B6FF39A8C33c2725b540403AC3768E95FD79;
address public manager = 0x6B53F45F9c0D71ca054AcC2bdDEB48AFfAaC7462;
address public manager2 = 0x7300b977659B20C51d0082D5370771A829a64628;
```

User address is registered access restriction is provided to functions listed below:

```
_syncOven()
createDuel()
createLottery()
```

Mentioned addresses are hardcoded in the contract.

```
0x8c23B6FF39A8C33c2725b540403AC3768E95FD79
0x6B53F45F9c0D71ca054AcC2bdDEB48AFfAaC7462
0x7300b977659B20C51d0082D5370771A829a64628
```

## RECOMMENDATION

Deployers', owners', administrators', managers', 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.



Implement multi-signature wallets: 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.

Use a decentralized governance model: 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.

## ACKNOWLEDGEMENT

PizzaBrick team has argued that privileged roles are used as intended, and commented to use multi-signature wallets to manage centralization wherever possible.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT



Identifier	Definition	Severity
LOG-02	Note regarding front-running	Minor <span style="color: green;">●</span>

A malicious entity can manipulate the operation by front-running a transaction to purchase assets and make profits by back-running a transaction to sell assets. Users can see pending transactions and their details, hence, they can potentially front-run a transaction.


- `buyBricks()` – Users can observe the desired purchase amount and front-run with their own purchase.
- `createDuel()` – A malicious entity may front-run duel creations or joining for favorable matchups.
- `createLottery()` – A malicious entity may front-run to be the last participant in a lottery.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

## RECOMMENDATION

Use commit reveal scheme or group transactions within a certain time-frame before processing them.



Identifier	Definition	Severity
LOG-03	Re-entrancy	Informational 

Since these functions do not make direct external calls, they should not be vulnerable to re-entrancy attacks. Re-entrancy can happen in unusual ways; therefore, it is recommended to add re-entrancy guard:

```
createDuel()  
createLottery()
```

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

## RECOMMENDATION

Use mutex or re-entrancy guard to deter re-entrant calls.





Identifier	Definition	Severity
PZB-01	Use of block properties for source of randomness	Medium ●

Be aware that the timestamp of the block can be manipulated by a miner. When the contract uses the timestamp to seed a random number, the miner can actually post a timestamp within 15 seconds of the block being validated, effectively allowing the miner to precompute an option more favorable to their chances.

## RECOMMENDATION

Use a secure random number generator, such as Chainlink VRF (Verifiable Random Function) or decentralized oracles for random number generation.

## ACKNOWLEDGEMENT

PizzaBrick team has argued that `_randomNumber()` is used instead of external Chainlink VRF to reduce contract complexity and dependency. On public - EVM blockchains, miner manipulation is possible when there's an economic incentive for miners, there's not many ways to avoid that.



Identifier	Definition	Severity
COD-03	Hardcoded values	Informational ●

Constants like FEE\_PERCENT and REFERRAL\_BRICKS\_PERCENT are hardcoded, making contract less flexible for future value changes.

```
uint256 constant FEE_PERCENT = 6;
uint256 constant REFERRAL_BRICKS_PERCENT = 7;
uint256 constant REFERRAL_PIZZA_PERCENT = 3;
uint256 constant BRICKS_PER_PIZZA = 100;
uint256 constant MAX_ACTIVE_LOTTERIES = 10;
```

## RECOMMENDATION

If required by design, keep values changeable in future.

## ACKNOWLEDGEMENT

PizzaBrick team has commented that - these constants are intentionally hardcoded as the protocol will not be changing these values.




Identifier	Definition	Severity
COD-07	Note regarding keccak256 secure hashing	Informational <input type="radio"/>

Note that usage of keccak256 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.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL



Identifier	Definition	Severity
COD-08	Lack of fallback function	Informational 

Fallback functions are usually executed in one of the following cases: If a function identifier doesn't match any of the available functions in a smart contract. If there was no data supplied along with the function call.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT

## RECOMMENDATION

Use fallback function with empty data, and mark it external, and payable.

## ACKNOWLEDGEMENT

PizzaBrick team has acknowledged this finding, and has agreed to keep the code as-is.



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

Smart contract is interacting with external and internal protocols e.g., Market Makers, Random Number, External contracts, Web 3 applications, Open Zeppelin tools, Math libraries, Oracle library, etc. 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 dependencies regularly, and mitigate severe impacts whenever necessary.

## ACKNOWLEDGEMENT

PizzaBrick team has acknowledged to inspect third party dependencies regularly.



Identifier	Definition	Severity
COD-11	Inadequate access restrictions	Medium 🟡

Below mentioned functions have no ownership or permission check, allowing for unauthorized changes:

```
buyBricks()  
withdrawMoney()  
collectPizza()  
swapPizzaToBricks()  
upgradePizzeria()  
sellPizzerias()  
upgradeOven()  
upgradeCookingSpeed()  
createDuel()  
createLottery()
```

Even though some of these functions need to be accessible by user to play game, it is recommended to use modifiers to validate the access.

## RECOMMENDATION

Function access must be restricted adequately to deter malicious actors, and unwanted state changes.



## PARTIAL RESOLUTION

PizzaBrick team has argued that above mentioned functions need to be accessible by user to play game.

Below mentioned functions use `_syncOven()` to check if the user is registered:

```
collectPizza()  
upgradePizzeria()  
sellPizzerias()  
upgradeOven()  
upgradeCookingSpeed()
```

PizzaBrick team has added require statements to mentioned functions, however they are still missing custom modifiers:

```
createDuel()  
createLottery()
```



Identifier	Definition	Severity
COD-12	Lack of event-driven architecture	Minor ●

There are no events being logged in the contract, which makes it difficult to track and analyze the contract's activities.

## RECOMMENDATION

Use events to track state changes, such as upgrades, duels, lotteries. Events improve transparency and provide a more granular view of contract activity.





Identifier	Definition	Severity
COD-13	Note regarding flash loan vulnerabilities	Minor <span style="color: green;">●</span>

Smart contract does not interact with external contracts directly for token swapping or lending, so it is less susceptible to flash loan attacks, which usually exploit some form of arbitrage opportunity.

However, this smart contract has underlying Re-entrancy, and Checks-Effects-Interactions vulnerabilities, therefore, interaction with malicious contracts can technically introduce flash loan vulnerabilities.

## RECOMMENDATION

Due to the interconnected nature of DeFi contracts, smart contracts can be exploited using another malicious contract. Be cautious while interacting with third-party contracts, tokens, and protocols.

- Fix underlying Re-entrancy, and Checks-Effects-Interactions vulnerabilities.
- Limit how often a function can be called by a particular user within a given timeframe.
- Use a time delay or multi-step process for significant contract actions.

## ACKNOWLEDGEMENT

PizzaBrick team has added nonReentrant modifier to `buyBricks()` and `withdrawMoney()`.



Identifier	Definition	Severity
VOL-03	Design flaws	Medium 🟡


- In `createDuel()` and `createLottery()` functions, users are deducting bricks (`users[msg.sender].bricks -= ...`). If malicious entities create these games and win, they're essentially getting their money back, plus a potential bonus. This design is flawed, and can be exploited.
- In `buyBricks()` function, smart contract collects fees and sends them out, but any remaining ether (`msg.value` minus all deductions) remains trapped in the contract.
- In `withdrawMoney()` function, user's pizza is decremented before the transfer is attempted. If the external call fails (even though check for its success is added), user's pizza balance is still decreased, resulting in a potential loss of funds.

TERFI  
CONFIDENTIALINTERFI  
CONFIDENTIAL

## RECOMMENDATION

Fix non-conforming logic.



Identifier	Definition	Severity
COM-01	Floating compiler status	Minor 
PZB-02	Potential arithmetic vulnerabilities	

Compiler is set to:

`>=0.7.0 <0.9.0;`

Arithmetic operations like += and -= don't have overflow or underflow checks.

If smart contract is deployed using compiler below 8.0, technically arithmetic vulnerabilities may occur. Compilers over 8.0 have built-in underflow, overflow checks.

TERFI  
CONFIDENTIAL

INTERFI  
CONFIDENTIAL


## RECOMMENDATION

Pragma should be fixed to the version that you're indenting to deploy your contracts with. Use SafeMath – if you're planning to deploy contract with older compiler than 8.0.

## RESOLUTION

PizzaBrick team has fixed compiler version to 0.8.20.



Identifier	Definition	Severity
PZB-03	Unchecked underflow	Minor 

In the `withdrawMoney` function, subtraction from `users[msg.sender].pizza` without checking if amount is less than or equal to `users[msg.sender].pizza` will lead to underflow.

TERFI  
CONFIDENTIALINTERFI  
CONFIDENTIAL

## RECOMMENDATION

Use SafeMath library or the in-built Solidity's underflow/overflow checks using a version  $\geq 0.8.0$ .

## RESOLUTION

PizzaBrick team has fixed compiler version to `0.8.20`.



Identifier	Definition	Severity
COM-04	Unbounded loops	Minor <span style="color: green;">●</span>

Smart contract functions rely on mentioned unbounded loops for important state change, update, and execution:

```

users()
stage()
ticketsPrice()
maxParticipants()
participants()
activeLotteryCount()

```

For example, if activeLotteryCount becomes very large, the loop inside createLottery() function can consume excessive gas:

Users can continuously create duels, potentially making this storage grow indefinitely, leading to potential gas limit problems in the future.

## RECOMMENDATION

Fix infinite loop through and limit storage.



## 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.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL





## 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](mailto: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