IARD SOLUTIONS

Protecting Web3

PANDEXCHANGE SECURITY AUDIT

PUBLIC

03/05/2023

audit@iard.solutions

https://iard.solutions

Introduction

IARD Solutions is at the forefront of the cryptocurrency industry, dedicated to the mission of decentralizing services and ensuring the highest standards of security and reliability. This report represents a comprehensive assessment conducted by our expert team at IARD Solutions, specializing in Smart Contract audits.

At IARD Solutions, we take pride in our rigorous and comprehensive audit approach. Our team of experts follows industry-best practices to ensure a thorough evaluation of Smart Contracts. Our audit methodology encompasses a combination of manual review and automated analysis tools, providing a multi-faceted assessment of the audited project's codebase. We adhere to the principles of transparency and accountability, employing a systematic and structured process to identify potential vulnerabilities, logic flaws, and security risks. By leveraging both static and dynamic analysis techniques, we assess the code's functionality, security, and adherence to industry standards.

The purpose of this report is to provide a thorough analysis of the PandExchange project's Smart Contract, assessing its integrity, functionality, and security. In this document, we present a concise overview of the audited project, outlining key findings and recommendations to enhance the project's robustness and trustworthiness in the decentralized landscape. Note that the audit’s scope is limited to the source code provided to us at this time, referred in the audit scope section.

In the audit below, our findings are set in different categories depending on their impact on the Smart Contract:

- CRITICAL: Critical vulnerabilities pose an immediate and severe threat to the security, functionality, or integrity of the system. Exploiting these vulnerabilities could result in significant damage or loss.
- HIGH: High-severity vulnerabilities are serious but may not be as urgent as critical ones. They have the potential to cause significant harm if exploited and should be addressed promptly.
- MEDIUM: Medium-severity vulnerabilities are important but may not pose an immediate threat. They should be addressed in a timely manner to prevent potential security issues.
- LOW: Low-severity vulnerabilities are minor issues that have limited impact and are less likely to be exploited. They should still be addressed as part of a comprehensive security strategy.
- INFORMATIONAL: Informational findings provide non-critical, supplementary information that may be relevant for improving overall system understanding.

1. Audit Scope

The audit conducted by IARD Solutions for the PandExchange project was a comprehensive examination of its Smart Contract ecosystem. Our assessment covered various critical aspects, including but not limited to:

1. **Code Integrity:** We thoroughly reviewed the Smart Contract codebase to ensure it adheres to best coding practices, follows industry standards, and is free from vulnerabilities.
2. **Functionality:** We assessed the functionality of the Smart Contract to verify that it performs as intended, executes transactions correctly, and handles edge cases effectively.
3. **Security:** Security is a paramount concern throughout the audit. We scrutinized the code for potential vulnerabilities such as re-entrancy attacks, overflows, and underflows to name the most commons, ensuring that the Smart Contract is robust against malicious attempts.
4. **Gas Optimization:** In the context of blockchain, efficient gas usage is vital. We optimized the code for gas consumption, enhancing cost-effectiveness and overall performance.
5. **Documentation:** We also reviewed documentation to ensure that it provides clear and accurate guidance for users and developers, promoting ease of understanding and integration.

This comprehensive scope allowed us to provide a holistic evaluation of the PandExchange project's Smart Contract, identifying strengths and areas for improvement to enhance its overall reliability and security. We produced this audit using both automatic tools and manual analysis to reduce risks as much as possible.

The audit is limited to the code provided by PandExchange team. The audit was done on the commit “a0a48af6441c543ee0ea83bcbf2e974cc71bba6b” from their git repository.

It is essential to acknowledge that while IARD Solutions conducts thorough audits with the utmost diligence and expertise, our assessments may not uncover all potential vulnerabilities or risks. The primary purpose of our audit is to provide a comprehensive evaluation of the Smart Contract's codebase and security practices as of the audit date. Our findings and recommendations are based on the information available at the time of the assessment. The rapidly evolving nature of blockchain and cybersecurity means that new vulnerabilities can emerge, and project circumstances may change post-audit.

2. Audit Results

2.1 Contract Overview

In our audit of the Smart Contract for the PandExchange, we conducted a comprehensive examination of the codebase, which is written in Solidity. The Smart Contract was compiled using v0.8.19+commit.7dd6d404, and it relies on several external interfaces, including Uniswap IRouter01 and IRouter02. The project is deployed on BNB Chain, Polygon and Arbitrum.

The key functions of this Smart Contract play a pivotal role in its functionality and security. Some of the most critical functions include:

- createDCAPlan: This function is responsible for creating a DCA Plan for a user, registering in the Smart Contract and moving the users' funds inside the PandExchange Smart Contract. Note that the creation of a DCA Plan perform the first token swap instantly.
- executeDCAPlanOccurrence: This function is responsible for performing an occurrence in the DCA Plan, meaning that it will perform a swap with the specified amount of token in the stead of the DCA Plan owner and reward the caller with the set reward.
- deleteDCAPlan: This function allows for the deletion of a user's DCA Plan, recovering in the process every unexchanged tokens. It can be called at any time after the creation.

These functions, along with the overall structure of the Smart Contract, form the backbone of the project's functionality and security. Our audit focused on ensuring their correctness, efficiency, and adherence to best practices, as well as the other functions'.

2.2 Unit Testing Coverage

Unit test coverage is a crucial aspect of ensuring the reliability and functionality of the Smart Contract. During our audit, we evaluated the extent of unit test coverage to gauge the comprehensiveness of testing within the codebase. The unit tests are designed to validate individual components and functions of the Smart Contract, helping identify potential issues early in the development process. We assessed the percentage of code covered by unit tests, examining their effectiveness in ensuring code correctness and preventing regressions. Our findings regarding unit test coverage are integral to our overall assessment of the project's code quality and robustness.

The Unit Testing Coverage for this project is as follow:

File Name	% Statements	% Branch	% Functions	% Lines	Uncovered lines
PandExchange.sol	100%	90%	100%	100%	

Remaining uncovered branch corresponds to unreachable branches. The high coverage rate allows us to state that the PandExchange Smart Contract perform as expected in all the tested use cases and provide us with a high degree of confidence in the functionalities of PandExchange Smart Contract.

2.3 Audit Findings

Our audit team found the following problems in the PandExchange Smart Contract:

- CRITICAL: 0 vulnerabilities
- HIGH: 0 vulnerabilities
- MEDIUM: 1 vulnerability
- LOW: 3 vulnerabilities
- INFORMATIONAL: 2 vulnerabilities

2.2.1 Function that sends Ether to arbitrary destinations. MEDIUM

Explanation of the Problem:

The audit identified a function within the smart contract that allows the sending of Ether to arbitrary destinations. This presents a high-risk scenario, as it could lead to unauthorized transactions and potential loss of funds.

Recommended Solution:

To mitigate the risk associated with the function, it is recommended to implement proper access controls and validation checks to ensure that Ether transactions are only permitted to authorized destinations. This will enhance the security of the smart contract and prevent unauthorized fund transfers.

Update by the Team:

The team has successfully addressed this issue by implementing comprehensive access controls and validation checks. The function has been modified to restrict Ether transfers to only predefined and authorized destinations. This update ensures a secure and controlled flow of funds within the smart contract.

2.2.2 Re-entrancy Vulnerability Leading to Out-of-Order Events LOW

Explanation of the Problem:

The audit revealed a re-entrancy vulnerability in the smart contract, potentially leading to out-of-order execution of events. While the impact is assessed as low, it is crucial to address this vulnerability to maintain the integrity and intended order of events within the contract.

Recommended Solution:

To mitigate the re-entrancy vulnerability, the recommended solution involves implementing appropriate locks and checks to ensure that events are executed in the intended order. This will prevent any unexpected behaviour and maintain the contract's functionality as designed.

Update by the Team:

The team has successfully addressed the re-entrancy vulnerability by incorporating locks and checks within the smart contract. These measures guarantee the proper sequencing of events, eliminating the risk of out-of-order execution.

2.2.3 Dangerous Usage of block.timestamp LOW

Explanation of the Problem:

The audit identified a potentially risky use of block.timestamp within the smart contract. While the impact is assessed as low, such usage can lead to vulnerabilities related to timestamp manipulation and must be addressed to ensure the contract's resilience.

Recommended Solution:

To mitigate the risk associated with block.timestamp, it is recommended to explore alternative methods for timestamp handling or incorporate additional safeguards to prevent manipulation. This will enhance the security of the smart contract against timestamp-based attacks.

Update by the Team:

The team has addressed the potential risk by implementing alternative methods for timestamp handling within the smart contract. This update ensures a more secure approach to timestamp usage, mitigating the identified vulnerability.

2.2.4 Problem Detected: Low-Level Calls LOW

Explanation of the Problem:

The audit flagged two instances of low-level calls within the smart contract. While the impact is categorized as informational, it is important to note such occurrences for transparency and potential optimizations.

Recommended Solution:

Considering the informational nature of the issue, no direct action is mandated. However, it is recommended to assess the necessity of low-level calls and explore higher-level alternatives for improved readability and maintainability.

Update by the Team:

The team is currently evaluating the necessity of the identified low-level calls. Depending on the assessment, adjustments may be made for enhanced readability and potential optimizations. Further updates will be provided based on the team's decision.

2.2.5 Conformity to Solidity Naming Conventions INFORMATIONAL

Explanation of the Problem:

The audit identified deviations from Solidity naming conventions in 25 instances within the smart contract. While this issue is informational, adhering to naming conventions enhances code readability and maintainability.

Recommended Solution:

To address the naming convention discrepancies, it is recommended to update the variable and function names in alignment with Solidity conventions. This will contribute to code consistency and facilitate future code reviews.

Update by the Team:

The team has rectified the naming convention discrepancies by updating variable and function names according to Solidity conventions. This update enhances code consistency and aligns with best practices for improved readability and maintainability.

2.2.6 Conformance to Numeric Notation Best Practices INFORMATIONAL

Explanation of the Problem:

The audit highlighted instances where the smart contract could benefit from conforming to numeric notation best practices. While this issue is informational, adhering to best practices improves code readability and maintainability.

Recommended Solution:

To align with numeric notation best practices, it is recommended to update numeric representations in the smart contract. This will enhance code consistency and facilitate future code reviews.

Update by the Team:

The team has addressed the numeric notation discrepancies, ensuring conformity to best practices. This update contributes to improved code readability and maintains alignment with established standards.

3. Conclusion

In concluding our comprehensive security audit of the smart contract, IARD Solutions has diligently identified and addressed various aspects to fortify the integrity and reliability of the system. Our team, with a commitment to excellence, tackled high-impact vulnerabilities, including the function allowing Ether transfers to arbitrary destinations and a re-entrancy vulnerability affecting event order. These issues were promptly mitigated through the implementation of robust access controls and event sequencing safeguards.

It is crucial to note that while the audit provides a comprehensive overview of the smart contract's security posture at the time of assessment, the evolving nature of blockchain technology requires continuous vigilance. We recommend regular reassessment and adherence to security best practices to ensure the ongoing resilience of the smart contract.

IARD Solutions remains dedicated to the highest standards of security and is available for any further inquiries or support required to maintain the robustness of your decentralized solution. We appreciate the opportunity to contribute to the security and success of your project.