SALUS SECURITY

AUG 2023

CODE SECURITY ASSESSMENT

X WINNER

Overview

Project Summary

- Name: X WINNER
- Platform: EVM-compatible chains
- Commit: 1065926
- Repository: https://github.com/xwinner-dao/platform-contracts
- Language: Solidity
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	X WINNER
Version	v2
Type	Solidity
Dates	August 25 2023
Logs	August 14 2023; August 25 2023

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	1
Total Low-Severity issues	1
Total informational issues	4
Total	6

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction **4**

1.1 About SALUS 4

1.2 Audit Breakdown 4

1.3 Disclaimer 4

Findings **5**

2.1 Summary of Findings 5

2.2 Notable Findings 6

1. Chain reorgs can lead to change in award distribution 6

2. Unchecked return value of transfer() and transferFrom() 7

2.3 Informational Findings 8

3. Commented-out code across the codebase 8

4. State variables can be marked as immutable 9

5. Event parameters should be indexed 10

6. Use of magic number 11

Appendix **12**

Appendix 1 - Files in Scope 12

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Chain reorgs can lead to change in award distribution	Medium	Configuration	Acknowledged
2	Unchecked return value of transfer() and transferFrom()	Low	Data Validation	Acknowledged
3	Commented-out code across the codebase	Informational	Code Quality	Resolved
4	State variables can be marked as immutable	Informational	Code Quality	Resolved
5	Event parameters should be indexed	Informational	Logging	Acknowledged
6	Use of magic number	Informational	Code Quality	Acknowledged

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Chain reorgs can lead to change in award distribution	
Severity: Medium	Category: Configuration
Target: - RandomGame.sol	

Description

According to the VRF Security Considerations documentation: “In principle, miners/validators of your underlying blockchain could rewrite the chain's history to put a randomness request from your contract into a different block, which would result in a different VRF output. Note that this does not enable a miner to determine the random value in advance. It only enables them to get a fresh random value that might or might not be to their advantage. By way of analogy, they can only re-roll the dice, not predetermine or predict which side it will land on.” and “You must choose an appropriate confirmation time for the randomness requests you make.”
The requestConfirmations value in the RandomGame contract tells the Chainlink VRF service how many blocks the VRF service waits before writing a fulfillment to the chain. This value is currently set to 3, which is insufficient for chains prone to frequent reorganizations. Considering the application's anticipated launch on Polygon (as indicated by the Polygon coordinator address in the comment), we can use Polygon's statistics to demonstrate this issue. Here we can see there are more than 5 block reorganizations a day with depth that is more than 3 blocks. In this article we can even see a recent event where there was a 156 block depth chain reorg on Polygon.
Consequently, there is a high probability that users who have won rewards will lose them in the event of a chain reorganization.
Example Scenario:

1. User plays the game and wins.
2. Chain reorg happens and thus miners/validators put a new randomness request which would result in different VRF output.
3. With the new different VRF output, the user finds he is not winning.
4. This unexpected outcome may confuse the user and harm the project's reputation.

Recommendation

Use a larger requestConfirmations value.

Status

The team has acknowledged this issue.

2. Unchecked return value of transfer() and transferFrom()	
Severity: Low	Category: Data Validation
Target: - Platform.sol - PlatformSFT.sol - AgentPool.sol - GamePool.sol - InitiatorPool.sol	

Description

As defined in the ERC20 specification, the transfer()/transferFrom() function returns a bool variable that signals the success of the call. However, throughout the codebase, the value returned from calls to transfer() and transferFrom() is ignored. Examples of this are:
- Line 127 of GamePool.sol
- Line 132 of GamePool.sol

Recommendation

To handle calls to transfer()/transferFrom() safely, even when interacting with ERC20 implementations that, incorrectly, do not return a boolean, consider using the safeTransfer()/safeTransferFrom() function in OpenZeppelin's SafeERC20 contract.

Status

The team has acknowledged this issue.

2.3 Informational Findings

3. Commented-out code across the codebase	
Severity: Informational	Category: Code quality
Target: - All	

Description

There are various instances in the codebase where the code is commented out. Removing these commented-out code would improve code readability. Some examples include RandomGame.sol, IPlatformSFT.sol etc

Recommendation

Remove the commented-out code.

Status

The team has resolved the issue with commit e34bedd5.

4. State variables can be marked as immutable	
Severity: Informational	Category: Code quality
Target: - GamePool.sol	

Description

The state variable FT is set in the constructor and it is not updated anywhere in the file. Hence, it can be marked as immutable for gas savings.

Recommendation

Mark the variable FT as immutable.

Status

The team has resolved the issue with commit 62c2a9e6.

5. Event parameters should be indexed	
Severity: Informational	Category: Logging
Target: - InitiatorPool.sol	

Description

The event Withdraw emits FT, owner and amount. The first two event parameters should be indexed for better filtering by off-chain clients.

Recommendation

Make the event parameters indexed.

Status

The team has acknowledged this issue.

6. Use of magic number	
Severity: Informational	Category: Code quality
Target: - AgentPool.sol - RandomGame.sol - RewardArithmetic.sol	

Description

The mentioned files use magic numbers throughout for calculations. To enhance readability and user understanding, it is advisable to avoid this practice.

Recommendation

Instead of magic numbers, use constant variables.

Status

The team has acknowledged this issue.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 1065926:

File	SHA-1 hash
Platform.sol	3462f9c06a8d2cb915c846d1fc2aa17fc2630bc1
PlatformSFT.sol	3b7eee066b568425c4054ea6985997cbd2bc0434
RandomGame.sol	c54388892fa3f7dadf79563132a5ffab99839d6a
RewardArithmetic.sol	32ed21b594e0bc6789dddef1ab9d88d3b0550a48
Roles.sol	3a1828a27a80a635bb6ce00e755c71314361868f
Pool.sol	0066b5d38dc47ce547606b9733ec89b967f647ad
AgentPool.sol	80a4bf931a5a3ca400ea2d932a096b944c3666b1
GamePool.sol	56d535ef733347d6aed979bac60651c30618e892
InitiatorPool.sol	fe6709c69c85f864666c1401678c9751b477ac97
ShareCalculator.sol	7d7c7cbca2c7cb0201731c310b9f724d53a600d0