Audited by CRED SHIELDS

CredShields

Andar Bahar Smart Contract Audit

December 31, 2024 • CONFIDENTIAL

Description

This document details the process and result of the Smart Contract audit performed by CredShields Technologies PTE. LTD. on behalf of Allin Gaming between October 29th, 2024, and December 17th, 2024. A retest was performed on December 18th, 2024.

Author
Shashank (Co-founder, CredShields) shashank@CredShields.com

Reviewers
Aditya Dixit (Research Team Lead), Shreyas Koli (Auditor), Naman Jain (Auditor), Sanket Salavi (Auditor)

Prepared for
Allin Gaming

Table of Contents

1. Executive Summary ----------

Allin Gaming engaged CredShields to perform a smart contract audit from October 29th, 2024, to December 17th, 2024. During this timeframe, 4 vulnerabilities were identified. **A retest was performed on December 18th, 2024, and all the bugs have been addressed.**

High and Critical vulnerabilities represent the greatest immediate risk to "Allin Gaming" and should be prioritized for remediation, and fortunately, none were found.

The table below shows the in-scope assets and a breakdown of findings by severity per asset. Section 2.3 contains more information on how severity is calculated.

Assets in Scope	Critical	High	Medium	Low	info	Gas	Σ
Andar Bahar	0	0	0	2	2	0	**4**
	0	**0**	**0**	**2**	**2**	**0**	**4**

Table: Vulnerabilities Per Asset in Scope

The CredShields team conducted the security audit to focus on identifying vulnerabilities in Andar Bahar's scope during the testing window while abiding by the policies set forth by Allin Gaming's team.

State of Security

To maintain a robust security posture, it is essential to continuously review and improve upon current security processes. Utilizing CredShields' continuous audit feature allows both Allin Gaming's internal security and development teams to not only identify specific vulnerabilities but also gain a deeper understanding of the current security threat landscape.

To ensure that vulnerabilities are not introduced when new features are added, or code is refactored, we recommend conducting regular security assessments. Additionally, by analyzing the root cause of resolved vulnerabilities, the internal teams at Allin Gaming can implement both manual and automated procedures to eliminate entire classes of vulnerabilities in the future. By taking a proactive approach, Allin Gaming can future-proof its security posture and protect its assets.

2. The Methodology ----------

Allin Gaming engaged CredShields to perform the Andar Bahar Smart Contract audit. The following sections cover how the engagement was put together and executed.

2.1 Preparation Phase

The CredShields team meticulously reviewed all provided documents and comments in the smart contract code to gain a thorough understanding of the contract's features and functionalities. They meticulously examined all functions and created a mind map to systematically identify potential security vulnerabilities, prioritizing those that were more critical and business-sensitive for the refactored code. To confirm their findings, the team deployed a self-hosted version of the smart contract and performed verifications and validations during the audit phase.

A testing window from October 29th, 2024, to December 17th, 2024, was agreed upon during the preparation phase.

2.1.1 Scope

During the preparation phase, the following scope for the engagement was agreed upon:

IN SCOPE ASSETS
https://github.com/AllInGaming1/casino/tree/34b3423bc31dc3645b17e98e9a407665db0d0807/contracts/andar-bahar

2.1.2 Documentation

The Allin Gaming's team provided documentation for all the assets in scope and promptly answered all our questions.

2.1.3 Audit Goals

CredShields employs a combination of in-house tools and manual methodologies to conduct thorough security audits for Rust-based smart contracts. The audit process primarily involves manually reviewing the contract's source code, following best practices for Rust and WebAssembly (Wasm) development, and leveraging an internally developed, industry-aligned checklist. The team focuses on understanding key concepts, creating targeted test cases, and analyzing business logic to identify potential vulnerabilities.

2.2 Retesting Phase

Allin Gaming is actively partnering with CredShields to validate the remediations implemented towards the discovered vulnerabilities.

2.3 Vulnerability classification and severity

CredShields follows OWASP's Risk Rating Methodology to determine the risk associated with discovered vulnerabilities. This approach considers two factors - Likelihood and Impact - which are evaluated with three possible values - **Low**, **Medium**, and **High**, based on factors such as Threat agents, Vulnerability factors, and Technical and Business Impacts. The overall severity of the risk is calculated by combining the likelihood and impact estimates.

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	None	Low	Medium
		LOW	MEDIUM	HIGH
Likelihood				

Overall, the categories can be defined as described below -

1. **Informational**

   We prioritize technical excellence and pay attention to detail in our coding practices. Our guidelines, standards, and best practices help ensure software stability and reliability. Informational vulnerabilities are opportunities for improvement and do not pose a direct risk to the contract. Code maintainers should use their own judgment on whether to address them.

2. **Low**

   Low-risk vulnerabilities are those that either have a small impact or can't be exploited repeatedly or those the client considers insignificant based on their specific business circumstances.

3. **Medium**

   Medium-severity vulnerabilities are those caused by weak or flawed logic in the code and can lead to exfiltration or modification of private user information. These vulnerabilities can harm the client's reputation under certain conditions and should be fixed within a specified timeframe.

4. **High**

   High-severity vulnerabilities pose a significant risk to the Smart Contract and the organization. They can result in the loss of funds for some users, may or may not require specific conditions, and are more complex to exploit. These vulnerabilities can harm the client's reputation and should be fixed immediately.

5. **Critical**

   Critical issues are directly exploitable bugs or security vulnerabilities that do not require specific conditions. They often result in the loss of funds and Ether from Smart Contracts or users and put sensitive user information at risk of compromise or modification. The client's reputation and financial stability will be severely impacted if these issues are not addressed immediately.

6. **Gas**

   To address the risk and volatility of smart contracts and the use of gas as a method of payment, CredShields has introduced a "Gas" severity category. This category deals with optimizing code and refactoring to conserve gas.

2.4 CredShields staff

The following individual at CredShields managed this engagement and produced this report:

- Shashank, Co-founder CredShields shashank@CredShields.com

Please feel free to contact this individual with any questions or concerns you have about the engagement or this document.

3. Findings Summary ----------

This chapter presents the results of the security assessment. Findings are organized by severity and categorized by asset, with references to relevant classifications or standards. Each asset section includes a summary for clarity. The executive summary table provides an overview of the total number of identified security vulnerabilities for each asset, grouped by risk level.

3.1 Findings Overview

3.1.1 Vulnerability Summary

During the security assessment, 4 security vulnerabilities were identified in the asset.

VULNERABILITY TITLE	SEVERITY	Vulnerability Type
Dead Code	Low	Code With No Effects
Missing functionality to enable auto-bet	Low	Missing Functionality
Missing Ownership Transfer Mechanism	Informational	Insecure Ownership Transfer
Documentation Will Mislead Stakeholders, Affecting Clarity	Informational	Informational Discrepancy

Table: Findings in Smart Contracts

4. Remediation Status ----------

Allin Gaming is actively partnering with CredShields from this engagement to validate the discovered vulnerabilities' remediations. **A retest was performed on December 18th, 2024, and all the issues have been addressed.**

Also, the table shows the remediation status of each finding.

VULNERABILITY TITLE	SEVERITY	REMEDIATION STATUS
Dead Code	Low	**Fixed** [Dec 18, 2024]
Missing functionality to enable auto-bet	Low	**Fixed** [Dec 18, 2024]
Missing Ownership Transfer Mechanism	Informational	**Fixed** [Dec 18, 2024]
Documentation Will Mislead Stakeholders, Affecting Clarity	Informational	

Table: Summary of findings and status of remediation

5. Bug Reports ----------

Bug ID #1 [Fixed]

Dead Code

Vulnerability Type
Code With No Effects - SWC-135

Severity
Low

Description
In the submit_bet() function, the contract contains logic to validate bets by calling the validate_bet() function. Within validate_bet(), there are checks related to auto-bet. If auto-bet is set in bet it will revert. but in the submit_bet() function there is some code related to auto-betting, such as iterating over the iteration_count of auto-bets and validating stop loss/gain of auto-bets. However, auto-bet is explicitly disabled in the contract, and if detected, the function reverts with an error.

Affected Code

- https://github.com/AllInBetsCom/casino/blob/34b3423bc31dc3645b17e98e9a407665db0d0807/contracts/andar-bahar/src/helpers.rs#L139-L143
- https://github.com/AllInBetsCom/casino/blob/34b3423bc31dc3645b17e98e9a407665db0d0807/contracts/andar-bahar/src/contract.rs#L262-L273
- https://github.com/AllInBetsCom/casino/blob/34b3423bc31dc3645b17e98e9a407665db0d0807/contracts/andar-bahar/src/contract.rs#L244

Impacts
This does not impact the security aspect of the Smart contract but prevents confusion when the code is sent to other developers or auditors to understand and implement.
This reduces the overall size of the contracts and also helps in saving gas.

Remediation
It is recommended to remove any logic or iterations related to auto-betting in the submit_bet() functions, as auto-betting is not allowed in the contract. Or you can introduce a function to enable auto-bet if you want to use it in future.

Retest

This vulnerability has been fixed by adding a new storage access `ALLOW_AUTOBET.save(deps.storage, &false)` and a new function `UpdateAutobetFlag()`.

Bug ID #2 [Fixed]

Missing functionality to enable auto-bet

Vulnerability Type
Missing Functionality

Severity
Low

Description
In the current implementation of the contract, the validate_bet() function explicitly disallows auto-betting and reverts when it detects auto-bet-related parameters. Additionally, while there is dead code in the submit_bet() function that references auto-bet functionalities, there is no mechanism in the contract for the owner or authorised entity to enable or implement auto-betting functionality in the future.

This results in a design limitation that prevents the contract from adapting to future requirements, such as enabling auto-betting functionality. A scenario may arise where the owner wants to introduce auto-bet features, either to increase the usability of the platform or to support automated betting strategies, but they will not be able to do so without redeploying the contract, which is costly and inefficient.

Affected Code

- https://github.com/AllInBetsCom/casino/blob/34b3423bc31dc3645b17e98e9a407665db0d0807/contracts/andar-bahar/src/helpers.rs#L139-L143

Impacts
The contract cannot accommodate future enhancements related to auto-betting without significant modifications or redeployment.

Remediation
It is recommended to implement functionality to enable auto-bet.

Retest
This vulnerability has been fixed by adding a new storage access ALLOW_AUTOBET.save(deps.storage, &false)? and a new function UpdateAutobetFlag().

Bug ID #3 [Fixed]

Missing Ownership Transfer Mechanism

Vulnerability Type
Insecure Ownership Transfer

Severity
Informational

Description
The contract sets the initial administrator (owner) during deployment via the instantiate function. However, it does not provide any functionality to transfer ownership or update the admin after deployment. This creates a design limitation as the ownership cannot be changed, even if there is a need to transfer it to a new owner.

Ownership transfer is an essential feature for decentralized systems to ensure flexibility and recoverability. If the current admin's private keys are lost or compromised, the inability to transfer ownership could lead to the permanent loss of control over the contract.

Affected Code

- https://github.com/AllInBetsCom/casino/blob/34b3423bc31dc3645b17e98e9a407665db0d0807/contracts/andar-bahar/src/contract.rs

Impacts
If the admin loses access to their private keys or if the keys are compromised, there is no way to transfer ownership to a new secure address, leading to a permanent loss of control over the contract

Remediation
It is recommended to implement a two-step ownership transfer in contract.

Example code :

```
// Execute: Transfer Ownership
pub fn execute_transfer_ownership(
        deps: DepsMut,
        _env: Env,
        info: MessageInfo,
        new_admin: String,
```

```
) -> Result<Response, ContractError> {
        let admin = ADMIN.load(deps.storage)?;

        if info.sender != admin {
        return Err(ContractError::OnlyAdmin {});
        }

        if new_admin.is_empty() {
        return Err(ContractError::EmptyNewAdmin {});
        }

        PENDING_ADMIN.save(deps.storage, &Some(new_admin.clone()))?;
        Ok(Response::new().add_attribute("action",
"transfer_ownership").add_attribute("pending_admin", new_admin))
}

// Execute: Accept Ownership
pub fn execute_accept_ownership(
        deps: DepsMut,
        _env: Env,
        info: MessageInfo,
) -> Result<Response, ContractError> {
        let pending_admin = PENDING_ADMIN.load(deps.storage)?;

        if pending_admin.is_none() || pending_admin.as_ref().unwrap() !=  &info.sender.to_string()
{
        return Err(ContractError::OnlyPendingAdmin {});
        }

        ADMIN.save(deps.storage, &info.sender.to_string())?;
        PENDING_ADMIN.save(deps.storage, &None)?;

        Ok(Response::new().add_attribute("action",
"accept_ownership").add_attribute("new_admin", info.sender.to_string()))
}
```

Retest

This issue has been fixed by adding new functions: transfer_admin_control() and accept_admin_control().

Bug ID #4

Documentation Will Mislead Stakeholders, Affecting Clarity

Vulnerability Type
Informational Discrepancy

Severity
Informational

Description
The Andar Bahar betting system documentation mentions that a color-related feature has been implemented. However, upon reviewing the source code, it is clear that no such implementation exists. This inconsistency between the documentation and the actual code can lead to confusion for stakeholders, developers, and users referencing the documentation for accurate guidance.

Affected Code

- https://github.com/AllInBetsCom/casino/blob/main/contracts/andar-bahar/src/helpers.rs#L65-L125

Impacts
This discrepancy could result in stakeholders being misinformed about the capabilities of the system, leading to potential misunderstandings in the development or testing process. Developers or other teams relying on the documentation may waste time searching for a feature that does not exist, which could delay the project or cause unnecessary confusion.

Remediation
It is recommended that the color-related feature reference be removed from the documentation to align it with the actual implementation. This will ensure the accuracy and reliability of the documentation, preventing confusion and unnecessary issues for anyone using or referencing it.

Retest
-

6. The Disclosure ----------

The Reports provided by CredShields are not an endorsement or condemnation of any specific project or team and do not guarantee the security of any specific project. The contents of this report are not intended to be used to make decisions about buying or selling tokens, products, services, or any other assets and should not be interpreted as such.

Emerging technologies such as Smart Contracts and Solidity carry a high level of technical risk and uncertainty. CredShields does not provide any warranty or representation about the quality of code, the business model or the proprietors of any such business model, or the legal compliance of any business. The report is not intended to be used as investment advice and should not be relied upon as such.

CredShields Audit team is not responsible for any decisions or actions taken by any third party based on the report.

YOUR SECURE FUTURE STARTS HERE

CRED SHIELDS

At CredShields, we're more than just auditors. We're your strategic partner in ensuring a secure Web3 future. Our commitment to your success extends beyond the report, offering ongoing support and guidance to protect your digital assets

Audited by CRED SHIELDS

20A TANJONG PAGAR ROAD, SINGAPORE (088443)

info@credshields.com