Immutable Internal Contract Audit

Gem Game Contract

Internal Auditors	Peter Robinson
Assessment Date	April 12, 2024
Final Report	April 12, 2024

Previous audits:

- None

Contents

Description

Gem Game allows people to check-in on-chain. They call a function that emits an event which includes their address and the time. This is used within the game to show that they have checked-in at that time.

Scope

Commit: 97f00aa69c7cfadddc67ca271593eaa0b1eac940

Asset	**Description**
Smart Contracts	GemGame.sol
Threat model	None

Team’s Greatest Concerns

- The team would like a general review. They have no specific concerns.

Basic Contract Analysis

The table below analyzes[1] the contracts.

Type	File	Lines	nSLOC	Complexity
Contract	GemGame.sol	63	27	30

[1] Analysis produced using the Solidity Metrics VSCode extension.

Smart Contracts

GemGame.sol

GemGame.sol allows people to “check-in” as part of a game. They call a function, earnGem, to indicate they are checking-in. The function emits an event recording the msg.sender and the time of the event. This contract allows administrators to pause the contract, to help mitigate attacks.

Roles:

- DEFAULT_ADMIN_ROLE: Role admin.
- _PAUSE: Role that can pause the contract.
- _UNPAUSE: Role that can unpause the contract.

External or Public Functions[2] that modify state:

Function Name	Function Selector	Authorisation Check	Notes
earnGem	ae56842b	Permissionless / None.	By design, anyone can call this function. The function’s only output, the event, is specific to the caller, and can not be in some way attributed to another account. The function can be paused.
grantRole	2f2ff15d	Only role admin.	
pause	8456cb59	Only _PAUSE role	
renounceRole	36568abe	Only msg.sender can renounce their own role.	
revokeRole	d547741f	Only role admin.	
unpause	3f4ba83a	Only _UNPAUSE role	

There are no External or Public Functions that do not modify state.

Function Name	Function Selector	Notes
DEFAULT_ADMIN_ROLE	a217fddf	
getRoleAdmin	248a9ca3	
hasRole	91d14854	
paused	5c975abb	
supportsInterface	01ffc9a7	

[2] The list of functions was determined using `forge inspect GemGame methods`

Upgradeable checks: This contract is not upgradeable.

Analysis of code logic:

- renounceRole allows the last admin to renounce their administrative role. This means that mistakenly, the last account with DEFAULT_ADMIN_ROLE role could mistakenly renounce their own ownership. If this happened, no one would be able to role administration for the _PAUSE and _UNPAUSE roles.
- This contract uses AccessControl, rather than AccessControlEnumerable. AccessControlEnumerable should be used as this variation provides APIs to fetch the number of accounts with a role and to fetch accounts with a role based on their index. Doing this allows for the list of accounts for a certain role to be fetched.

Abuser Test Cases and Findings

Overloading Downstream Services via Calls to earnGem

The following analysis was based upon code prior to the introduction of pausing (commit : bd62796f0a3532424e724e6758fcc2abc9099bc5).
Classification: Informational

Description: An attacker could call the earnGem function thousands of times per block, every block. This could cause downstream services that are going to do operations based on the event to become overloaded.

The gas cost of calling the earnGem is approximately 11,100 gas. A contract could be designed to call this function repeatedly. Given the 30 million gas limit, this function could be called approximately 2700 times per block. Given the block time of two seconds, this would mean the downstream service would be called at 1350 calls per second.

Action: The downstream service needs to be designed to handle this type of attack. Possible solutions:

- Only process the first event per transaction, thus allowing only one call to earnGem per transaction. Doing this would mean attackers would need to submit separate transactions, thus incurring the 21000 base transaction fee for each call, thus reducing the attack to approximately 467 times per second.
- Process all events emitted during the same block at once.
- Ordering the processing in such a way that recent events for the same account are quickly discarded.

Team Response:

The team has decided that making the contract Pausable will reduce the overall risk while maintaining the required flexibility to quickly iterate on this design. As a follow-up to this finding, we have now implemented the Pausable functionality.

For downstream services, we recognise that there is still some risk between the time the attack is initiated and when the contract is paused. Transactions that are within the same 24 hours are discarded.

Audit Response: This mitigates possible attacks.