SALUS SECURITY

APR 2024

CODE SECURITY ASSESSMENT

BIGA

Overview

Project Summary

- Name: BIGA - Arcade
- Platform: EVM-compatible Chains
- Language: Solidity
- Repository: https://github.com/bigaarcade/arcade-sc
- Audit Scope: See Appendix - 1

Project Dashboard

Application Summary

Name	BIGA - Arcade
Version	v2
Type	Solidity
Date	Apr 03 2024
Logs	Mar 22 2024; Apr 03 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	2
Total Low-Severity issues	1
Total informational issues	1
Total	4

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

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	Cross-chain replay attacks are possible with `withdraw()`	Medium	Cryptography	Resolved
2	Centralization risk	Medium	Centralization	Acknowledged
3	Risky external calls cause Deposited events to be triggered unexpectedly	Low	Logging	Resolved
4	Missing two-step transfer ownership pattern	Informational	Business logic	Resolved

2.2 Notable Findings

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

1. Cross-chain replay attacks are possible with `withdraw()`

Severity: Medium	Category: Cryptography
Target: - contracts/BIGA.sol	

Description

contracts/BIGA.sol:L106-L121

```
function _validateWithdrawData(
    bytes calldata _signature,
    address _user,
    address _tokenIn,
    address _tokenOut,
    uint256 _amountOut,
    uint256 _nonce
) private {
    bytes memory data = abi.encodePacked(_user, _tokenIn, _tokenOut, _amountOut,
_nonce);
    data.verifySignature(_signature, validator, "GB04");

    bytes32 dataHash = keccak256(data);
    require(!hashUsed[dataHash], "GB05");

    hashUsed[dataHash] = true;
}
```

The _validateWithdrawData() function is used to validate the withdrawal signature of the validator. However, the critical chainId information that prevents cross-chain replay is missing from the signature field data.

This could lead to a cross-chain replay vulnerability in the BIGA contract when deployed on multiple chains with the same validator address. An attacker could exploit a single signature across multiple chains, resulting in asset losses for the project.

Recommendation

It is recommended that signatures follow the EIP712 standard and include the chainId as a signature domain to prevent cross-chain replay attacks.

Status

The team has resolved this issue in commit fefefe11.

2. Centralization risk

Severity: Medium	Category: Centralization
Target: - contracts/BIGA.sol	

Description

a) In the BIGA contract, the user deposits money using the deposit() function, but the contract does not record the user's deposit information. Therefore, all user data is stored in the off-chain database. This means that if the off-chain database is attacked, the user's on-chain assets may be compromised.

b) In addition, there is a privileged validator role. The admin role has the ability to sign withdrawal transactions.

If the validator's private key is compromised, the attacker can exploit the validator's role to sign malicious withdrawal transactions. In this scenario, users' assets would be severely compromised.

Recommendation

1. It is recommended to store the user's deposit and withdrawal information in the contract to avoid receiving off-chain attacks.

2. We recommend transferring the admin role to a multisig account with a timelock feature for enhanced security. This ensures that no single person has full control over the account and that any changes must be authorized by multiple parties.

Status

This issue has been acknowledged by the team.

3. Risky external calls cause Deposited events to be triggered unexpectedly

Severity: Low	Category: Logging
Target: - contracts/BIGA.sol	

Description

contracts/BIGA.sol:L61-L71

```
function deposit(address _tokenIn, address _tokenOut, uint _amountIn) external
nonReentrant {
    address user = msg.sender;

    _tokenIn.checkERC20("GB02");
    _tokenOut.checkERC20("GB02");
    require(_amountIn > 0, "GB03");

    IERC20(_tokenIn).safeTransferFrom(user, address(this), _amountIn);

    emit Deposited(user, _tokenIn, _tokenOut, _amountIn);
}
```

The BIGA protocol's deposit tracking relies on off-chain handling of the Deposited event. However, the deposit() function allows users to choose their tokenIn, allowing malicious actors to repeatedly trigger the Deposited event using fake ERC20 tokens.

This could undermine the integrity of off-chain systems tracking the Deposited event if they do not validate the tokenIn parameter, resulting in corrupted user deposit records.

Recommendation

Consider setting up a whitelist of tokenIn in the contract.

Status

The team has resolved this issue in commit fefefe11.

2.3 Informational Findings

4. Missing two-step transfer ownership pattern	
Severity: Informational	Category: Business logic
Target: - contracts/BIGA.sol	

Description

The BIGA contract inherits from the Ownable contract. This contract does not implement a two-step process for transferring ownership. Thus, ownership of the contract can easily be lost when making a mistake in transferring ownership.

Recommendation

Consider using the Ownable2Step contract from OpenZeppelin instead.

Status

The team has resolved this issue in commit fefefe11.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit dcb796b:

File	SHA-1 hash
BIGA.sol	16b9723c78c27367ba97efffb24c97207de4dcf2
SafeCheck.sol	d29f13a822681f26632d6b9cb5ab0f320bca5404
VerifySignature.sol	14c3471d18c40236b18b273c70439bed7ab8550c