SALUS SECURITY

NOV 2023

CODE SECURITY ASSESSMENT

BINARYX

Overview

Project Summary

- Name: BinaryX - aigame
- Platform: BNB Smart Chain
- Language: Solidity
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	BinaryX - aigame
Version	v2
Type	Solidity
Dates	Nov 28 2023
Logs	Nov 20 2023; Nov 28 2023

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	2
Total Low-Severity issues	2
Total informational issues	6
Total	10

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	Use of the testnet address	Medium	Configuration	Acknowledged
2	Centralization risk	Medium	Centralization	Acknowledged
3	Setting role admins to self	Low	Access Control	Acknowledged
4	Missing events for functions that change critical state	Low	Logging	Acknowledged
5	Lack of expiration timestamp check	Informational	Business Logic	Acknowledged
6	Vulnerable to signature replay attacks	Informational	Cryptography	Acknowledged
7	Use of floating pragma	Informational	Configuration	Acknowledged
8	Redundant code	Informational	Redundancy	Acknowledged
9	Spelling mistake	Informational	Code Quality	Acknowledged
10	Gas optimization suggestions	Informational	Gas Optimization	Acknowledged

2.2 Notable Findings

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

1. Use of the testnet address

Severity: Medium	Category: Configuration
Target: - AbstractDeployer.sol	

Description

The state variable BNX has been set to a testnet address in the AbstractDeployer contract. This address will be used as the token parameter during the deployment of AiHero.

AbstractDeployer.sol:L11

```
address public BNX = 0x376c75638A8391849d44567A4af7D8fD069086b9;
```

Recommendation

Consider configuring the correct address for BNX before deploying to the mainnet.

Status

This issue has been acknowledged by the team.

2. Centralization risk

Severity: Medium	Category: Centralization

Target:
- Deployer.sol
- AiHero.sol
- BasicAccessControl.sol

Description

The AiHero contract has a privileged owner role. The owner of the AiHero contract has permissions for all roles and can modify all the configurations.

BasicAccessControl.sol:L38-L46

```
function _checkRole(bytes32 role, address account) internal view virtual override {
        if (!hasRole(role, account) && owner() != account) {
        revert(string(abi.encodePacked(
                "BasicAccessControl: account ",
                Strings.toHexString(account),
                " is missing role ",
                nameOfRole(role))));
        }
}
```

Should the owner's private key be compromised, an attacker can update the signer to an address he controls and mint tokens to his liking.

Recommendation

We recommend transferring privileged accounts to multi-sig accounts with timelock governors for enhanced security. This ensures that no single person has full control over the accounts and that any changes must be authorized by multiple parties.

Status

This issue has been acknowledged by the team.

3. Setting role admins to self

Severity: Low	Category: Access Control
Target: - BasicAccessControl.sol	

Description

The BasicAccessControl defines two roles: ROLE_DEPLOYER and ROLE_OPERATOR. In the constructor, the ROLE_DEPLOYER's admin is set to itself. This pattern is discouraged.

BasicAccessControl.sol:L12-L13

```
_setRoleAdmin(ROLE_DEPLOYER, ROLE_DEPLOYER);
_setRoleAdmin(ROLE_OPERATOR, ROLE_DEPLOYER);
```

For example, if the owner grants the deployer role to Alice through addDeployer() function of the Deployer contract, she can then revoke deployer role from the Deployer contract.

Recommendation

It is recommended to use the DEFAULT_ADMIN_ROLE as the admin for these roles and to grant this role to highly trusted addresses.

Status

This issue has been acknowledged by the team.

4. Missing events for functions that change critical state

Severity: Low	Category: Logging
Target: - AiHero.sol	

Description

Events allow capturing the changed parameters so that off-chain tools/interfaces can register such changes that allow users to evaluate them. Missing events do not promote transparency and if such changes immediately affect users' perception of fairness or trustworthiness, they could exit the protocol causing a reduction in protocol users.

In the AiHero contract, events are lacking in the privileged setter functions (e.g. setRecipient(), setPrice()).

Recommendation

It is recommended to emit events for critical state changes.

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

5. Lack of expiration timestamp check

Severity: Informational	Category: Business Logic
Target: - AiHero.sol	

Description

The deadline can be used to limit the execution of the signature within a certain period. In the mint() function, the signature includes a timestamp, but it's not checked against the current timestamp.

AiHero.sol:L169-L213

```
function mint(
        ...
        uint256 timestamp,
        bytes memory signature) public {
        ...
        data = data.concat(timestamp);
        bytes32 hash = ECDSA.toEthSignedMessageHash(keccak256(data));
        ...
}
```

Recommendation

Consider checking the timestamp to determine if the signature has expired.

Status

This issue has been acknowledged by the team.

6. Vulnerable to signature replay attacks

Severity: Informational	Category: Cryptography
Target: - AiHero.sol	

Description

To prevent replay attacks (both cross-chain and multiple AiHero implementations), the signature should depend on the chainId and the AiHero address. However, none of these are taken into account in the current signature.

AiHero.sol:L16

```
bytes32 public constant ARGS_PREFIX = keccak256("AiHero.mint()");
```

AiHero.sol:L181-L190

```
bytes memory data;
data = data.concat(ARGS_PREFIX);
data = data.concat(account);
data = data.concat(avatarId);
data = data.concat(race);
data = data.concat(career);
data = data.concat(attributes);
data = data.concat(timestamp);
bytes32 hash = ECDSA.toEthSignedMessageHash(keccak256(data));
address recoveredSigner = ECDSA.recover(hash, signature);
```

Recommendation

Consider adding the chainId and contract address in the calculation of the hash.

Status

This issue has been acknowledged by the team.

7. Use of floating pragma

Severity: Informational	Category: Configuration
Target: - All	

Description

```
pragma solidity ^0.8.0;
```

The AiHero codebase uses a floating compiler version ^0.8.0.

Using a floating pragma ^0.8.0 statement is discouraged, as code may compile to different bytecodes with different compiler versions. Use a locked pragma statement to get a deterministic bytecode. Also use the latest Solidity version to get all the compiler features, bug fixes and optimizations.

Recommendation

It is recommended to use a locked Solidity version throughout the project. It is also recommended to use the most stable and up-to-date version.

Status

This issue has been acknowledged by the team.

8. Redundant code

Severity: Informational	Category: Redundancy

Target:
- AbstractDeployer.sol
- AiHero.sol
- BasicAccessControl.sol

Description

AiHero.sol:L9

```
import "./SignatureVerifier.sol";
```

The SignatureVerifier library is imported but never used.

AbstractDeployer.sol:L17-L22

```
function _init(BasicAccessControl a) internal returns (address) {
        a.grantDeployer(address(this));
        ...
}
```

The Deployer has been granted with ROLE_DEPLOYER during the deployment of AiHero. Thus, the line highlighted above is redundant.

BasicAccessControl.sol:L24-L27

```
modifier onlyOperator() {
        _checkRole(ROLE_OPERATOR);
        _;
}
```

The ROLE_OPERATOR is defined but never used.

Recommendation

Consider removing the redundant part of the code.

Status

This issue has been acknowledged by the team.

9. Spelling mistake

Severity: Informational	Category: Code Quality
Target: - AiHero.sol	

Description

_heros should be _hero**e**s.

AiHero.sol:L35

```
mapping(uint256 => Hero) private _heros;
```

Recommendation

Consider fixing the typo.

Status

This issue has been acknowledged by the team.

10. Gas optimization suggestions

Severity: Informational	Category: Gas Optimization

Target:
- AbstractDeployer.sol
- AiHero.sol

Description

AbstractDeployer.sol:L8-L13

```
address public RECIPIENT = 0xc9fc6362e6627bC981368C5284fe71de814408BB;
address public OPERATOR = 0x1d642bA5Bd1c4b6d669310E9ECf874C429f11960;
address public BNX = 0x376c75638A8391849d44567A4af7D8fD069086b9;
uint256 public AI_HERO_INITIAL_PRICE = 10000000000000000000;
```

The above state variables can be declared as constant to save gas.

AiHero.sol:L37

```
address public _token;
```

_token is only set in the constructor and can be declared as immutable.

AiHero.sol:L156

```
function setImageBaseURI(string memory v) public onlyDeployer
```

AiHero.sol:L169-L175

```
function mint(
        uint256 avatarId,
        uint256 race,
        uint256 career,
        uint256[] memory attributes,
        uint256 timestamp,
        bytes memory signature) public
```

The data location of parameters highlighted above could be calldata.

AiHero.sol:L192

```
require(_mintRequests[hash] == false, "AiHero: mint request has been processed");
```

It's unnecessary to compare `_mintRequests[hash]`, a boolean variable, to a boolean literal.

Recommendation

Consider applying the gas optimizations where needed.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files provided by the client:

File	SHA-1 hash
AbstractDeployer.sol	22c904689dfbafef03115a4d5aa490955bbc773f
AiGameV1.sol	255d2d2fee260117c1ab90431e76de2ade27ffb5
AiHero.sol	f5e64ad67922d6a21767d0510af276b48ec7499e
BasicAccessControl.sol	7b8c2141116c1cb0e497f40be3568548be2e4555
Bytes.sol	0b46415ed16c75334b4d11cb8efa7a207691d73b
Deployer.sol	39f4ef6f57af472de17ec0243a19dcb8fe3fca22
SignatureVerifier.sol	4d138b50095a45947a43b4b5fa3e6fedfe23a8f6
UseSigner.sol	4030bd47ff993c1c3ed13bebf127780b88bb2adf