SALUS SECURITY

SEP 2023

CODE SECURITY ASSESSMENT

HOOKED PROTOCOL

Overview

Project Summary

- Name: Hooked Protocol - Hooktopia
- Platform: BNB Smart Chain
- Language: Solidity
- Version: commit 12dd156
- Repository: https://github.com/DEVHooked/Hooktopia
- Audit Scope: See Appendix - 1

Project Dashboard

Application Summary

Name	Hooked Protocol - Hooktopia
Version	v2
Type	Solidity
Dates	September 4 2023
Logs	August 25 2023; September 4 2023

Vulnerability Summary

Total High-Severity issues	2
Total Medium-Severity issues	0
Total Low-Severity issues	1
Total informational issues	5
Total	8

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	Malicious users can obtain Hooktopia NFTs by burning someone else's Hooktopia Compass NFTs	High	Access control	Resolved
2	Signature replay may lead to unlimited token minting by users	High	Business Logic	Resolved
3	Centralization Risk	Low	Centralization	Mitigated
4	Missing zero address check	Information	Data Validation	Acknowledged
5	Can use immutable variable to save gas	Information	Gas Optimization	Resolved
6	Redundant code	Information	Redundancy	Resolved
7	Spelling error	Information	Code Quality	Resolved
8	Incorrect comments	Information	Code Quality	Resolved

2.2 Notable Findings

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

1. Malicious users can obtain Hooktopia NFTs by burning someone else's Hooktopia Compass NFTs

Severity: High	Category: Access control
Target: - contracts/HooktopiaCompass.sol	

Description

In HooktopiaCompass's burnCompass function, the caller should be the owner of the token or an authorized operator of the owner. However, in the code implementation, it erroneously requires the contract to be the caller's authorized operator.

This would allow attackers to burn anyone's Hooktopia Compass NFTs and obtain Hooktopia NFTs that they are not entitled to.

contracts/HooktopiaCompass.sol:L187

```
function burnCompass(uint256[] calldata tokenIds) external {
    require(burnOpen, "Burn not open");
    require(address(nftContract) != address(0), "NFT not set");

    for (uint256 i; i < tokenIds.length;) {
        uint256 tokenId = tokenIds[i];

        console.log("Burning token %s", tokenId);
        console.log("Owner of token %s", ownerOf(tokenId));
        console.log("Sender %s", msg.sender);
        // Validate ownership or approval before burning
        require(ownerOf(tokenId) == msg.sender || isApprovedForAll(msg.sender,
address(this)), "Not owner or approved");
        _burn(tokenId);

        uint8 tokenType = tokenTypes[tokenId];

        // Mint new NFT
        nftContract.redeem(msg.sender, followCompass(tokenType));

        // Increment counter
        i++;
    }
}
```

- **Proof of Concept**

```
function testBurnCompass() public {
    address bob = makeAddr("bob");
    address attacker = makeAddr("attacker");
    vm.startPrank(owner);
    hooktopiaCompass.ownerMint(bob, 1, 1);
```

```
    vm.stopPrank();

    uint256 balanceBefore = hooktopiaCompass.balanceOf(bob);

    // Attackers can burn tokens and mint them in Hooktopia by invoking
setApprovalForAll()
    vm.startPrank(attacker);
    hooktopiaCompass.setApprovalForAll(address(hooktopiaCompass), true);
    uint256[] memory tokens = new uint256[](1);
    tokens[0] = 0;
    hooktopiaCompass.burnCompass(tokens);
    vm.stopPrank();

    uint256 balanceAfter = hooktopiaCompass.balanceOf(bob);
    uint256 balance_Attacker = hooktopia.balanceOf(attacker);

    assertEq(balanceBefore, 1);
    assertEq(balanceAfter, 0);
    assertEq(balance_Attacker, 1);
}
```

Recommendation

Consider changing the aforementioned statement to: "isApprovedForAll(ownerOf(tokenId), msg.sender)".

Status

The team has resolved this issue in commit [d62da88](d62da88).

2. Signature replay may lead to unlimited token minting by users

Severity: High	Category: Business Logic

Target:
- contracts/HooktopiaCompass.sol

Description

The HooktopiaCompass contract places a restriction on the number of tokens authorized users can mint, as specified by MINTS_PER_WALLET.

However, the mint() function utilizes balanceOf() to determine the caller's previously minted token count. It should be noted though that a user's token balance and the number of tokens one has minted may not be the same. Users can bypass this verification by transferring tokens to other addresses.

Moreover, a single signature can be used multiple times, allowing users to repeatedly mint tokens once they acquire a valid signature.

contracts/HooktopiaCompass.sol:L96-L103

```
function mint(uint256 numberOfTokens, uint8 tokenType, uint256 goldPrice, uint256
silverPrice, bytes calldata signature) internal {
    require(numberOfTokens > 0, "numberOfTokens cannot be 0");
    require(numberOfTokens + balanceOf(msg.sender) <= MINTS_PER_WALLET, "Exceeds wallet
mint limit");
    require(tokenType == 0 || tokenType == 1, "Invalid token type");

    // Verify the signature
    bytes32 message =
ECDSA.toEthSignedMessageHash(keccak256(abi.encodePacked(MINT_HASH_TYPE, msg.sender)));
    require(SignatureChecker.isValidSignatureNow(mintSigner, message,
signature),"Invalid signature");
    //...
}
```

- **Proof of Concept**

```
function testMint()public{
    vm.startPrank(owner);
    hooktopiaCompass.setSaleState(HooktopiaCompass.SaleStates.PublicSale);
    vm.stopPrank();

    address bob = makeAddr("bob");
    address bob2 = makeAddr("bob2");
    vm.startPrank(bob);
    hooktopiaCompass.publicMint(3,0,getSig());

    assertEq(hooktopiaCompass.balanceOf(bob),3);

    hooktopiaCompass.transferFrom(bob,bob2,0);
    hooktopiaCompass.transferFrom(bob,bob2,1);
    hooktopiaCompass.transferFrom(bob,bob2,2);
```

```
    assertEq(hooktopiaCompass.balanceOf(bob),0);

    hooktopiaCompass.publicMint(3,0,getSig());

    assertEq(hooktopiaCompass.balanceOf(bob),3);
}
```

Recommendation

1. Consider designing a variable to track the count of tokens minted per user and use it to verify the minting process.
2. Consider preventing signature replay by tracking a nonce value for each user and using it to invalidate used signatures.

Status

The team has resolved this issue in commit d62da88.

3. Centralization Risk	
Severity: Low	Category: Centralization
Target: - contracts/HooktopiaCompass.sol	

Description

1. In the HooktopiaCompass contract, there is a crucial address, 'mintSigner', which is responsible for signing for users. The users need the signatures to mint tokens.

It should be noted that the 'mintSigner' address cannot be changed after it's set in the constructor. If the private key of this address is compromised, the team has no way to update the mintSigner address to a new one.

contracts/HooktopiaCompass.sol:L76

```
constructor(address _signer) ERC721("Hooktopia Compass ", "HC") {
    ...
    mintSigner = _signer;
    ...
}
```

2. When deploying the HooktopiaCompass contract, the deployer is granted the DEFAULT_ADMIN_ROLE and the MINTER_ROLE. If the deployer's private key is compromised, attackers can use the ownerMint function to mint tokens that are not entitled to them.

contracts/HooktopiaCompass.sol:L74-L75

```
constructor(address _signer) ERC721("Hooktopia Compass ", "HC") {
    _grantRole(DEFAULT_ADMIN_ROLE, msg.sender);
    _grantRole(MINTER_ROLE, msg.sender);
    ...
}
```

Recommendation

1. Consider adding the flexibility to update the mintSigner;
2. Consider transferring the privileged roles to multi-sig accounts.

Status

The team has mitigated this issue with commit d62da88.

2.3 Informational Findings

4. Missing zero address check

Severity: Information	Category: Data Validation
Target: - contracts/Hooktopia.sol - contracts/HooktopiaCompass.sol	

Description

It is considered a security best practice to verify addresses against the zero address during initialization or setting.

The zero address check is missing for the following functions:

contracts/Hooktopia.sol:L14

```
constructor(address _compass) ERC721("Hooktopia", "HT") {
    compassAddress = _compass;
}
```

contracts/HooktopiaCompass.sol:L76

```
constructor(address _signer) ERC721("Hooktopia Compass ", "HC") {
    ...
    mintSigner = _signer;
    ...
}
```

contracts/HooktopiaCompass.sol:L155

```
function setNFTContract(address _nftContract) external onlyRole(DEFAULT_ADMIN_ROLE) {
    nftContract = INFT(_nftContract);
}
```

Recommendation

Consider adding zero address checks for these address variables.

Status

This issue has been acknowledged by the team.

5. Can use immutable variable to save gas

Severity: Information	Category: Gas Optimization
Target: - contracts/Hooktopis.sol - contracts/HooktopiaCompass.sol	

Description

"MintSigner" and "CompassAddress" were set within the constructor and remain unchanged throughout the code. Therefore, they can be defined as immutable to save gas.

contracts/HooktopiaCompass.sol:L56

```
address public mintSigner;
```

contracts/Hooktopis.sol:L10

```
address public compassAddress;
```

Recommendation

Consider mark "mintSigner" and "compassAddress" as immutable.

Status

The team has resolved this issue in commit d62da88.

6. Redundant code

Severity: Information	Category: Redundancy
Target: - contracts/HooktopiaCompass.sol	

Description

Commented-out code, code for testing purposes (e.g. import “hardhat/console.sol”, console.log...) and unused code (e.g. unused variable) should be removed before deploying the contract to mainnet.

We have identified the following redundant codes.

- contracts/HooktopiaCompass.sol:L12,L183-L185,L71

Recommendation

Consider removing the redundant codes.

Status

The team has resolved this issue in commit d62da88.

7. Spelling error

Severity: Information	Category: Code Quality
Target: - contracts/HooktopiaCompass.sol	

Description

contracts/HooktopiaCompass.sol:L248

```
string memory description = tokenTypes[tokenId] == 0 ? "A compss made of gold can indicate the location of strong miracles which in hooktopia.According to the legend, miracles can bring powerful blessing to nourish your land" : "A compss made of silver can indicate the location of ordinary miracles which in hooktopia.According to the legend, miracles can bring powerful blessing to nourish your land";
```

In the tokenURI(), there is a typo: 'compss' should be 'compass'.

Recommendation

Consider correcting the typo error.

Status

The team has resolved this issue in commit d62da88.

8. Incorrect comments

Severity: Information	Category: Comment error
Target: - contracts/HooktopiaCompass.sol	

Description

The comments state that there are 666 gold compasses and 6000 silver compasses, which differs from the constants defined earlier in the code.

contracts/HooktopiaCompass.sol:L78,L81

```
constructor(address _signer) ERC721("Hooktopia Compass ", "HC") {
    //...
    for (uint64 i = 0; i < maxGoldSupply + reservedGold; i++) { // Gold from 0 to 665
        remainingGoldCompass.push(i);
    }
    for (uint64 i = uint64(maxGoldSupply + reservedGold); i < maxSupply; i++) { //
Silver from 666 to 6665
        remainingSilverCompass.push(i);
    }
}
```

Recommendation

It is recommended to update the comments. The comments should be modified to “Gold from 0 to 1110” and “Silver from 1111 to 6665” to align with the values of the constants.

Status

The team has resolved this issue in commit d62da88.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 12dd156:

File	SHA-1 hash
Hooktopia.sol	ea95660d8af83faec0d231eba83d391fcc2f1e45
HooktopiaCompass.sol	549cee733267c2c86acf4475702a68651a7b05c4