HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Catgirl
Date: 19 May, 2023

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for Catgirl
Approved By	Marcin Ugarenko | Lead Solidity SC Auditor at Hacken OU
Type	Marketplace
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://catgirl.io
Changelog	14.03.2023 - Initial Review 20.04.2023 - Second Review 19.05.2023 - Third Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by Catgirl (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project is review and security analysis of smart contracts in the repository:

Initial review scope

Repository	https://github.com/catgirlcoin/marketplace_contracts
Commit	6512502cc605c4d50765d4da3f28b5fd510369c5
Functional Requirements	https://github.com/catgirlcoin/marketplace_contracts/blob/main/MysteryBoxManager%20documentation.pdf
Technical Requirements	https://github.com/catgirlcoin/marketplace_contracts/blob/main/MysteryBoxManager%20documentation.pdf
Contracts	File: ./exchange/Exchange.sol SHA3: 0cbca25d7bbf30ecc6cb720d4a5b06ef156bd8b4755e0a46874a72b1e6a4716a File: ./exchange/ExchangeCore.sol SHA3: 4d85d7c7663b5af05328ce15c8a8373c0a3a05bd5c1a2ee051859f2ea4a2412a File: ./libraries/ArrayUtils.sol SHA3: 9fc72348ed434111167d1bc6d81e3788a7d6fb0d9422945eb799b01ce8334e9d File: ./libraries/SaleKindInterface.sol SHA3: cd3aa48f036e52520a7cdd3d0977f65b865b8e42ad2851034d5e9abc7fd3508b File: ./Marketplace.sol SHA3: ed3e444f25aa1a0df7fb2bc1c85da007355511221b0509026f8b060c0cbc1ecb File: ./MysteryBoxManager.sol SHA3: f8885db1feed4a42d364a0f17c30272dc732b70da82262b816d3222eab632c97

Second review scope

Repository	https://github.com/catgirlcoin/marketplace_contracts
Commit	a8ea50c472b7044344a6062eb823a4fe2dc9395b
Functional Requirements	https://github.com/catgirlcoin/marketplace_contracts/blob/main/CatgirlNFT%20documentation.pdf https://github.com/catgirlcoin/marketplace_contracts/blob/main/Marketplace%20documentation.pdf https://github.com/catgirlcoin/marketplace_contracts/blob/main/MysteryBoxManager%20documentation.pdf https://github.com/catgirlcoin/marketplace_contracts/blob/main/PriceOracle%20documentation.pdf

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Technical Requirements	https://github.com/catgirlcoin/marketplace_contracts/blob/main/CatgirlNFT%20documentation.pdf https://github.com/catgirlcoin/marketplace_contracts/blob/main/Marketplace%20documentation.pdf https://github.com/catgirlcoin/marketplace_contracts/blob/main/MysteryBoxManager%20documentation.pdf https://github.com/catgirlcoin/marketplace_contracts/blob/main/PriceOracle%20documentation.pdf
Contracts	File: contracts/CatgirlNFT.sol SHA3: b236aec9981be471f1c1a04461088899e2508efdcf4aaf22c00d8337a341bcf1 File: contracts/Marketplace.sol SHA3: 6a38dda296a3eb193c761cfbd57d2dec4d19895324955a09c5293aeca45744a6 File: contracts/MysteryBoxManager.sol SHA3: 0de6ec8537725a69f8945b469090bcb050b6cfa5f574b86df1182c957641b7f8 File: contracts/PriceOracle.sol SHA3: 64a115b6070e7eb51b01c8d0eb581e2d674197281324db281bb92e575dd36563 File: contracts/exchange/Exchange.sol SHA3: 5761add4321f279b26b84bc5041a69e560db05eb64367bd6bab560b578ca2d9f File: contracts/exchange/ExchangeCore.sol SHA3: 3b2710bdbc60eced31020ce8eef52971d75bdc549a6dba21bb3bfd91a0d79442 File: contracts/interfaces/ICatgirlNFT.sol SHA3: 362a8d623ccd16b8359ee6b6661fe04f0e2d343214bf670764d45507183aa0cd File: contracts/interfaces/IERC20Extended.sol SHA3: ef718830fe8a4f4b4d85c7bfb48bdef8a95d7096e87be1cae53026a9f69a8c8d File: contracts/interfaces/IPancakePair.sol SHA3: cb61938781b7f6a3304ca6b338313b3be28da1960e7f2dda1324615ef6b80350 File: contracts/interfaces/IPancakeRouter01.sol SHA3: e475d1323e82c026d63f01e938bdfe5da84edb68ceacec0d758645f36632b39c File: contracts/interfaces/IPancakeRouter02.sol SHA3: 15937e607c938b626ff02955e28ac28b6257fc8e85c6e6d9adf28ecab4501d47 File: contracts/interfaces/IPriceOracle.sol SHA3: ea655b5eae23bead8766d85c9585715ba61d4393895343cef57835a427f45232 File: contracts/interfaces/VRFConsumerBaseV2.sol SHA3: 10c0e625fa22b99aef33933288a180cd1298b1d68be7bc9361450f93f73ea272 File: contracts/libraries/ArrayUtils.sol SHA3: c6c118e141d7898cff3448ae3a1960d14d0da28ea0141889a875c4ba3f89b197 File: contracts/libraries/SaleKindInterface.sol SHA3: 88d8c543349db0db60dfc633c7b44463ddadbee37e876b02e2899b97b978b97c File: contracts/libraries/UniformRandomNumber.sol SHA3: c6326dd1e82bb7332147fa5a39c7c9bee78bf343de989e3041c37cabc13d48cc

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Third review scope

Repository	https://github.com/catgirlcoin/marketplace_contracts
Commit	6e29e5981525b5654d521b627b5ae85fc19353be
Functional Requirements	https://github.com/catgirlcoin/marketplace_contracts/blob/main/CatgirlNFT%20documentation.pdf https://github.com/catgirlcoin/marketplace_contracts/blob/main/Marketplace%20documentation.pdf https://github.com/catgirlcoin/marketplace_contracts/blob/main/MysteryBoxManager%20documentation.pdf https://github.com/catgirlcoin/marketplace_contracts/blob/main/PriceOracle%20documentation.pdf
Technical Requirements	https://github.com/catgirlcoin/marketplace_contracts/blob/main/CatgirlNFT%20documentation.pdf https://github.com/catgirlcoin/marketplace_contracts/blob/main/Marketplace%20documentation.pdf https://github.com/catgirlcoin/marketplace_contracts/blob/main/MysteryBoxManager%20documentation.pdf https://github.com/catgirlcoin/marketplace_contracts/blob/main/PriceOracle%20documentation.pdf
Contracts	File: contracts/CatgirlNFT.sol SHA3: f9aad2e6bcbf388e221116abc64f66d892ba9b8c715e0670de3f31964b2021a4 File: contracts/Marketplace.sol SHA3: 06ddb4b36983ebe7a1d0c9f756bcc041b71040eaabd2cf3bde47170a2f67697c File: contracts/MysteryBoxManager.sol SHA3: a873c23d751d18a3917875dca54f1f10bd507261c414620ce5311bc966fb870a File: contracts/PriceOracle.sol SHA3: 5fdc740368f94cf02af67ce32436e4f6f016ca3d786686c73daefe2a8beab4b7 File: contracts/exchange/Exchange.sol SHA3: 48701af49b733acdce9eaf77e44d1677d287c58883f87563a7a5a80a3e1b893e File: contracts/exchange/ExchangeCore.sol SHA3: 88634ac44e6770dbdca2f3d8287f24ac5f6f6b3b9f74f0e456d7fda757ae8188 File: contracts/interfaces/ICatgirlNFT.sol SHA3: 362a8d623ccd16b8359ee6b6661fe04f0e2d343214bf670764d45507183aa0cd File: contracts/interfaces/IERC20Extended.sol SHA3: ef718830fe8a4f4b4d85c7bfb48bdef8a95d7096e87be1cae53026a9f69a8c8d File: contracts/interfaces/IPancakePair.sol SHA3: cb61938781b7f6a3304ca6b338313b3be28da1960e7f2dda1324615ef6b80350 File: contracts/interfaces/IPancakeRouter01.sol SHA3: e475d1323e82c026d63f01e938bdfe5da84edb68ceacec0d758645f36632b39c File: contracts/interfaces/IPancakeRouter02.sol SHA3: 15937e607c938b626ff02955e28ac28b6257fc8e85c6e6d9adf28ecab4501d47 File: contracts/interfaces/IPriceOracle.sol SHA3: ea655b5eae23bead8766d85c9585715ba61d4393895343cef57835a427f45232 File: contracts/interfaces/VRFConsumerBaseV2.sol SHA3: 10c0e625fa22b99aef33933288a180cd1298b1d68be7bc9361450f93f73ea272

	File: libraries/ArrayUtils.sol SHA3: c6c118e141d7898cff3448ae3a1960d14d0da28ea0141889a875c4ba3f89b197 File: libraries/SaleKindInterface.sol SHA3: 88d8c543349db0db60dfc633c7b44463ddadbee37e876b02e2899b97b978b97c File: libraries/UniformRandomNumber.sol SHA3: c6326dd1e82bb7332147fa5a39c7c9bee78bf343de989e3041c37cabc13d48cc

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors.
Medium	Medium vulnerabilities are usually limited to state manipulations but cannot lead to asset loss. Major deviations from best practices are also in this category.
Low	Low vulnerabilities are related to outdated and unused code or minor gas optimization. These issues won't have a significant impact on code execution but affect code quality

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- The functional requirements are provided.
- The technical description is provided.
- NatSpec is provided.
- The development environment is described.

Code quality

The total Code Quality score is **10** out of **10**.

- The development environment is configured.

Test coverage

Code coverage of the project is **90.18%** (branch coverage).

- The coverage for ArrayUtils can be improved with dedicated tests.

Security score

As a result of the audit, the code contains no issues. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.6**.

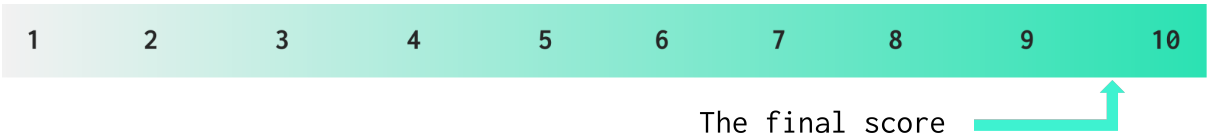

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
14 March 2023	18	12	9	6
20 April 2023	9	4	5	1
19 May 2023	0	0	0	0

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Risks

- The CatgirlNFT contract is heavily centralized. Admins are able to:
  - mint new NFTs with the desired attributes (season, score, rarity, character) bypassing the generation process.
  - assign new arbitrary attributes to any NFT through the function rebornCatgirl().
  - if the *_expectedCatgirlPerUSD* parameter provided to *processPriceUpdate()* is wrong for any backend issue, the function will fail and the price will not be updated.
- The admin of the MysteryBoxManager contract can withdraw the contract's funds not yet distributed, by using the function emergencyWithdraw(). The funds are supposed to be distributed every day, and the only risk incurred by the users is that 5% of that sum will not be burned as it should happen in the normal operation flow.
- The variable maximumNumberOfBox must be calibrated not to be incurred in the Block gas Limit in the claim() function. The client should pay attention also to the gas consumed in the openPendingBoxes() function of the CatgirlNFT contract, invoked in the claim() function. That function takes care to open and claim all the user's pending boxes, so its operations are gas intensive.
- All of the contracts are upgradeable so they can be changed after deployment, and any further change could introduce critical vulnerabilities.
- Wyvern protocol is designed in such a way that only the user who signed an order is able to cancel the related listing. This leads to a dangerous situation: if a user opens a listing for an NFT, then transfers that NFT to another wallet and then withdraws it to the original wallet, the listing is still active, possibly at a price lower than the floor price since some time may have passed. The frontend part of the protocol should put in place some measures to mitigate this situation.
- It is worth mentioning that Wyvern protocol is now at its third version, which still comes with a permissive license. The client instead designed the marketplace around the V1 of Wyvern protocol, which is outdated and contains multiple publicly disclosed vulnerabilities.
- The MysteryBoxManager contract relies on the prices of the BNB and Catgirl tokens, driven from the internal, centralized PriceOracle contract. The prices of BNB and Catgirl are calculated based on data from Chainlink, but the relevance of this data is determined by the maintenance done by the PRICE_UPDATER role. If the data inside the PriceOracle contract is not refreshed for long periods, the prices used for Catgirl NFT sales will not be accurate.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

System Overview

The audit scope covers the marketplace module of the Catgirl Token ecosystem.

- The main part is the marketplace itself. It is based on Wyvern protocol V1, with some features removed.
  Involved contracts:
    - exchange/Exchange.sol
    - exchange/ExchangeCore.sol
    - Marketplace.sol
    - libraries/ArrayUtils.sol
    - libraries/SaleKindInterface.sol
- The audit scope also contains MysteryBoxManager.sol, a contract where users can buy NFTs with different characters and rarity. There is no direct NFT sale: users can buy boxes that can then be opened on CatgirlNFT (an out of scope contract) to redeem the NFTs.
  The contract uses Chainlink to get random numbers to assign to the user boxes. Users can pay for the boxes using BNB and Catgirl tokens.

Privileged roles

- MysteryBoxManager.sol - all roles are assigned to the deployer by default:
    - DEFAULT_ADMIN_ROLE: can assign new roles, call the emergencyWithdraw() function, set the CatgirlNFT address, set the Catgirl token address, set the pancakeRouter address, set the farmingAddress, set the developmentAddress.
    - UPGRADER_ROLE: can upgrade the contract.
    - SETTER_ROLE: can set the mister box price, set the maximum number of unopened boxes held by each user, update the fund distribution percenage, update the burn percentage, set the current season, distribute the Catgirl Token and BNB funds according to the set shares, configure the oracle, pause and unpause the contract.
    - FACTORY_ROLE: not used.
- Marketplace.sol
    - UPGRADER_ROLE: can upgrade the contract.
    - SETTER_ROLE: can update MAKER_RELAYER_FEE and TAKER_RELAYER_FEE.

Recommendations

- In the MysteryBoxContract, the functions processDistributionCatgirl() and processDistributionBNB() are protected by the role SETTER_ROLE, but this is not required as users calling those functions would only benefit the protocol with the saved gas fees.

- Throughout the code, most variables are declared as uint, but it is recommended to be explicit and declare them as uint256.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Not Relevant
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Not Relevant
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	Passed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Passed
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Not Relevant
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155 EIP-712	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery. EIP-712 should be followed during a signer verification.	Passed
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Passed
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of Unused Variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP Standards Violation	EIP	EIP standards should not be violated.	Passed
Assets Integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Passed
User Balances Manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Passed
Token Supply Manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer.	Not Relevant
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Passed
Style Guide Violation	**Custom**	Style guides and best practices should be followed.	Passed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed
Secure Oracles Usage	**Custom**	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Passed
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be sufficient, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Passed
Stable Imports	**Custom**	The code should not reference draft contracts, which may be changed in the future.	Passed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Findings

■■■■ Critical

C01. Access Control Violation

SETTER_ROLE and *UPGRADER_ROLE* cannot be changed in the future because *DEFAULT_ADMIN_ROLE* is not assigned in the *Marketplace.sol* contract inside the *initialize()* function.

Path:
./contracts/Marketplace.sol : initialize()

Recommendation: Assign *DEFAULT_ADMIN_ROLE* inside the *initialize()* function.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

C03. Data Consistency

The function *hashOrder()* can produce collisions if provided with cleverly crafted parameters by an attacker.

This can lead to multiple exploits, for example, by allowing an attacker to fulfill an offer without sending or even owning the NFT the buyer wanted, while still being paid the victim's WETH.

A detailed analysis of the issue can be found at:
https://nft.mirror.xyz/VdF3BYwuzXgLrJglw5xF6CHcQfAVbqeJVtueCr4BUzs

Path:
./contracts/ExchangeCore.sol : hashOrder()

Recommendation: The issue can be fixed by assigning a fixed length to the signed bytes parameter : *callData* and *replacementPattern*.
The fix implemented by *OpenSea* together with *Wyvern core developers* and other parties can be found in the Wyvern V2.3 deployed contract.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

C04. Invalid Calculations

The function *guardedArrayReplace()* implements the wrong check in the if statement. It checks for *words > 0* while it should be checking for the remainder of the division *array.length / 0x20*, that is *array.length % 0x20*.

This wrong computation leads to unexpected consequences that are possibly exploitable, resulting in arbitrary storage writes.

A detailed analysis of the issue can be found at:

https://blocksecteam.medium.com/a-new-memory-overwrite-vulnerability-discovered-in-wyvern-protocol-5285996c297d

Path:
./libraries/ArrayUtils.sol : guardedArrayReplace()

Recommendation: Replace *words > 0* with the originally intended check *array.length % 0x20*.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

C05. Requirement Violation

The requirement on *maximumNumberOfBox* is poorly enforced. During buy operations, it should be checked against the box already owned by the user plus the number of boxes the user is buying.

buyCommonBoxWithBNB() checks it only against the boxes being bought, while *buyCommonBoxWithCatgirl()* only checks it against the boxes already owned by the user.

This issue can lead to a Denial Of Service due to a Block Gas Limit in the *claim()* function if users use the *buyCommonBoxWithBNB()* function.

Path:
./contracts/MysteryBoxManager.sol : buyCommonBoxWithBNB(), buyCommonBoxWithCatgirl()

Recommendation: Properly enforce the *maximumNumberOfBox* requirement during buy operations.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

C06. Unverifiable Logic

The *MysteryBoxManager* contract presents multiple interactions with out of scope contracts.

Path:
./contracts/MysteryBoxManager.sol

Recommendation: Add the *PriceOracle.sol* and *VRFConsumerBaseV2Upgradeable.sol* contracts to the scope of the audit.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

C07. Weak Source of Randomness

During the computation of *nyaScore*, the randomness from Chainlink is lost.

The *_rand* value gets computed and stored in a previous transaction, so it is known at the time of the *openPendingBoxes()* invocation.

Inside *internalMint()*, the *_rand* value gets combined with *block.number* to generate *nyaScore*. Since both *_rand* and *block.number* are known, the output value can easily be predicted by any user.

Path:
./contracts/CatgirlNFT.sol : internalMint()

Recommendation: Don't use *block.number* as a source of randomness. Note: the recommendation for this issue is highly linked to the M15 issue.

Found in: a8ea50c

Status: Fixed (Revised commit: 6e29e59)

High

H01. Signed Message Replay Attack

ExchangeCore.sol handles signatures but lacks the correct checks to avoid a signed message replay attack.

The signature could be replayed on the same chain, as there is no address validation, or crosschain, as there is no chainId used.

Path:
./contracts/ExchangeCore.sol

Recommendation: Implement *EIP-712*.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

H02. Malleable Signature

Signatures in Ethereum can be altered without the private key and remain valid. For example, elliptic key cryptography consists of three variables: *v*, *r*, and *s*, and if these values are modified in the right way, you can obtain a valid signature with an invalid private key.

Path:
./contracts/ExchangeCore.sol : validateOrder()

Recommendation: Implement *OpenZeppelin's ECDSA library* and use it instead of the *ecrecover()* function.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

H03. Missing Validation

The *ordersCanMatch()* function does not check that the *howToCall* parameter is the same in *buy* and *sell Order objects*, while they must be the same for the orders to match.

Path:
./contracts/ExchangeCore.sol : validateOrder()

Recommendation: Enforce the equality of *buy.howToCall* and *sell.howToCall*.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

H04. Requirement Violation

According to the documentation, *developmentPercentage* and *burnPercentage* should have constant values of 20% and 5% respectively.

The contract allows the change of these values via the function *setFundDistributionPercentage()*.

Path:
./contracts/MysteryBoxManager.sol : validateOrder()

Recommendation: Remove the setter function or make it explicit in the documentation that the values are not fixed.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

H05. Front-Running Attack

The function *swapBNBForCatgirl()* is not enforcing a minimum output value for the swap operation, potentially resulting in frontrunning every time the function is used.

The client implemented mitigation steps in the form of a getAmountsOut() call to the Catgirl/WBNB pair, with minimal slippage used, but these operations are done on-chain.

During the execution of *swapBNBForCatgirl()*, those mitigation steps had no effect, as the attack can be performed in the form of a sandwich attack, with before and after transfection.

Path:
./contracts/MysteryBoxManager.sol : swapBNBForCatgirl()

Recommendation: minAmountReceive has been set, but obtained on-chain, posing no mitigation for the frontrunning attack. The minimum amount should be provided as an external input, as the getAmountsOut() return value from the Pancake router can be easily manipulated.

Found in: 6512502

Status: Fixed (Revised commit: 6e29e59)

H08. Data Consistency

In the *buyCommonBoxWithBNB()* function, the user submitting an order is not refunded if he overpays in *BNB*.

Path:
./contracts/MysteryBoxManager.sol : buyCommonBoxWithBNB()

Recommendation: Refund the user with the difference between what he paid and the price he was supposed to pay.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

H09. Upgradeability Issues

Inside the *MysteryBoxManager.sol* contract, *ReentrancyGuardUpgradable* and *VRFConsumerBaseV2Upgradeable* are not initialized.

The extent of the consequences cannot be properly estimated because the *VRFConsumerBaseV2Upgradeable.sol* contract is imported locally but is not within the scope of the audit, nevertheless, contracts that are upgradeable should be initialized.

Path:
./contracts/MysteryBoxManager.sol : initialize()

Recommendation: Initialize *VRFConsumerBaseV2Upgradeable* and *ReentrancyGuardUpgradeable*.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

H09. Non-Finalized Code

The code should not import “hardhat/console.sol”.

The production code should not contain any functions or variables that are used solely in the test environment. This will allow malicious parties to manipulate the code or users to trigger them accidentally.

Path:
./contracts/ExchangeCore.sol

Recommendation: Remove debug import “hardhat/console.sol” from the production contract.

Found in: a8ea50c

Status: Fixed (Revised commit: 6e29e59)

H10. Race Condition

During the call of the *processPriceUpdate()* function, the price for the Catgirl Token is derived from the *Chainlink* result of the USD per BNB price feed and the Catgirl/WBNB PancakeSwap pair pool reserves.

The reserves of *Catgirl/WBNB PancakeSwap pair pool* can be manipulated before the *processPriceUpdate()* function call, resulting in an incorrect price of the Catgirl token stored on-chain and used for Catgirl NFT sales.

This could lead to the Catgirl NFT getting sold for less than its true value.

The external view function *getRealTimeCatgirlPerUSD()* that uses the same mechanism is safe to be used off-chain, but if any external contract uses it to get the current price of the Catgirl token, it can be manipulated. This function should only be used off-chain.

Path:
./contracts/PriceOracle.sol : processPriceUpdate()

Recommendation: Get the Catgirl token price by calling getRealTimeCatgirlPerUSD() through the frontend (as this is safe) and provide the output value as a parameter to the processPriceUpdate() function.

Compare the provided price with the one computed on-chain. If the deviation of the price is within a threshold of, for example, 1%, the computed on-chain value can be used with confidence that it was not manipulated.

Found in: a8ea50c

Status: Fixed (Revised commit: 6e29e59)

H11. Highly Permissive Role Access

MINTER_ROLE can burn NFTs from any address at any point, without the need for an approval.

Path:
./contracts/CatgirlNFT.sol : burnToken(), burnMultiple()

Recommendation: Restrict the access to this function for *MINTER_ROLE*.

Found in: a8ea50c

Status: Fixed (Revised commit: 6e29e59)

H12. Denial of Service Vulnerability

The function *openPendingBoxes()* checks if *setting.maxPurchase >= _numberOfPendingBoxes[i]*, this could create a Denial of Service, because the value of *_numberOfPendingBoxes[i]* is decided in the contract *MysterBoxManager.sol*, and the *maximumNumberOfBoxPending* variable inside of *MysteryBoxManager.sol* could be higher than the *setting.maxPurchase* variable of *CatgirlNFT.sol*, if this happens, there could be a Denial of Service.

Path:
./contracts/CatgirlNFT.sol : openPendingBoxes()

Recommendation: Leave only the check for *maximumNumberOfBoxPending* in the *MysteryBoxContract.sol*, don't check the boxes after they have already been sold.

Found in: a8ea50c

Status: Fixed (Revised commit: 6e29e59)

Medium

M01. Missing Validation

The input of the functions is not validated, its value must be within *10000* (100% in basis points) according to *ExchangeCore.INVERSE_BASIS_POINT*.

If left unvalidated, the function can result in human errors and a malicious increase.

Path:
./contracts/Marketplace.sol : setMakerFee(), setTakerFee()

Recommendation: Bound the input values to their allowed range. It is also advised to bind them to reasonable values, for users' safety.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

M02. Unscalable Functionality

Well-known contracts from projects should be imported directly from the source as the projects are in development and may update the contracts in the future.

This is the case for *ABDKMath64x64.sol*, which can be found in the abdk-libraries-solidity npm package.

Path:
./contracts/MysteryBoxManager.sol

Recommendation: Import abdk-libraries-solidity.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

M03. Missing Validation

Seasons are supposed to be incremental, but this requirement is not enforced.

Path:
./contracts/MysteryBoxManager.sol : setCurrentSeason()

Recommendation: Check that the provided *_season* is greater than the *currentSeason*.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

M05. Missing SafeERC20

The function *buyCommonBoxWithCatgirl()* does not use the *SafeERC20* library for checking the result of *ERC20 token transfers*. Tokens may not follow the *ERC20 standard* and return false in case of transfer failure or not return any value at all.

Path:
./contracts/MysteryBoxManager.sol : buyCommonBoxWithCatgirl()

Recommendation: Implement *OpenZeppelin's SafeERC20* library for *ERC20 tokens* interactions.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

M06. Inefficient Gas Model

The storage variable *pendingBox[msg.sender]* is repeatedly read in the *for* loop, while it can be loaded and read from memory to save gas on computations.

The same applies for the variable *pendingBox[user]* in function *getTotalPendingBox()*

Path:
./contracts/MysteryBoxManager.sol : claim(), getTotalPendingBox()

Recommendation: Load repeatedly read storage variables to memory and use them from there.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

M07. Missing Validation

The functions to buy boxes do not check that *_numberOfBoxes* is *> 0*. This can lead to a Denial of Service in the *claim()* function, as it loops the orders (the variable names are misleading; see issue L11) while the bound check is only on the number of boxes.

Path:
./contracts/MysteryBoxManager.sol : buyCommonBoxWithCatgirl(), buyCommonBoxWithBNB()

Recommendation: Enforce *_numberOfBoxes > 0* when users submit buy orders.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

M08. Best Practice Violation

The *ExchangeCore.sol* contract inherits from *ReentrancyGuardUpgradable* and *OwnableUpradeable*, but those contracts are never initialized in the flow of the contract, also, *Exchange.sol* inherits from *ExchangeCore.sol* and *Marketplace.sol* inherits from *Exchange.sol*, but the only initialized contracts are *AccessControlUpgradeable* and *UUPSUpgradable* inside the *Marketplace.sol initialize()* function.

The functions inherited from *OwnableUpgradeable* are never used, and not initializing *ReentrancyGuardUpgradeable* is not a security risk, but it is still best practice to initialize all upgradeable contracts that implement an initializer.

Path:
./contracts/Exchange.sol
./contracts/ExchangeCore.sol
./contracts/Marketplace.sol

Recommendation: Remove *OwnableUpgradable* from the *ExchangeCore.sol* contract because it is never used, and initialize the contracts properly inside the *Marketplace.sol* contract.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

M09. Contradiction

According to the NatSpec, only *SETTER_ROLE* should be able to update the contract state. However, in the implementation *DEFAULT_ADMIN_ROLE* can call various functions and change the contract state doing so.

Path:
./contracts/MysteryBoxManager.sol : setAddress(), setPancakeRouter(), setCatGirlTokenAddres(), setCatgirlNFTAddress()

Recommendation: Fix the mismatch.

Found in: 6512502

Status: Fixed (Revised commit: 6e29e59)

M10. Inefficient Gas Model

The number of pending boxes gets computed by *getTotalPendingBox()*, which at every invocation loops through all user's pending orders.

Path:
./contracts/MysteryBoxManager.sol : getTotalPendingBox()

Recommendation: It would be way cheaper to store this number directly in a mapping, and update it when users buy or claim new orders.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

M11. Contradiction

The *random()* function is never used nor documented.

Inside the structs *PendingBox* and *RequestBuyBoxData* there is no variable *tokenType* as described in the documentation.

The *DEFAULT_ADMIN_ROLE* role is never documented.

Path:
./contracts/MysteryBoxManager.sol

Recommendation: Fix the mismatches.

Found in: 6512502

Status: Fixed (Revised commit: 6e29e59)

M12. Variables Not Initialized

sSubscriptionId, *sKeyHash*, *callbackGasLimit*, *requestConfirmations*, *numWords* and *vrfCoordinator* are not initialized, they are set in the *setConfigVfr()* function, but this is not enforced.

Buying boxes before these variables are set would lead to an incorrect logic followed in the contract.

Path:
./contracts/MysteryBoxManager.sol : initialize()

Recommendation: Enforce the setting of those variables in the initializer function or check that they are not the default value when the boxes are being bought.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

M13. Upgradeable Errors

The initializer functions *__PriceOracle_init_unchained()* and *__CatgirlNFT_init_unchained()* should use the modifier *'onlyInitializing'* instead of *'initializer'*.

If the contracts get deployed and initialized in separated transactions, the initializations would revert.

Path:
./contracts/PriceOracle.sol
./contracts/CatgirlNFT.sol

Recommendation: Replace the initializer modifier with onlyInitializing.

Found in: a8ea50c

Status: Fixed (Revised commit: 6e29e59)

M14. Contradiction

Inside the struct *BoxSetting* there is no variable *tokenType* as described in the documentation.

Path:
./contracts/CatgirlNFT.sol

Recommendation: Fix the mismatches.

Found in: a8ea50c

Status: Fixed (Revised commit: 6e29e59)

Low

L01. Missing Event Parameters

Two parameters are left out of the emitted *order* *approve* events: *order.howToCall* and *order.replacementPattern*, which can be useful to track the contract state by off-chain logic.

Path:
./contracts/ExchangeCore.sol : approveOrder()

Recommendation: Add the missing parameters to emitted events.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

L02. Function Visibility

Many public functions can be declared *external* to save gas on invocations.

This is the case for all the functions of Exchange.sol.

Paths:
./contracts/Exchange.sol
./contracts/MysteryBoxManager.sol : processDistributionBNB(), processDitributionCatgirl(), emergencyWithdraw()

Recommendation: Change the visibility to external for functions not meant to be called internally.

Found in: 6512502

Status: Fixed (Revised commit: 6e29e59)

L03. Redundant Return Statement

The function *Exchange.atomicMatch_()* is missing the "*returns(type)*" part of the header, but in the function body it tries to return a value. It does not throw an error because it tries to return the output of *ExchangeCore.atomicMatch()*, which does not return anything.

The same also happens in other functions.

Path:
./contracts/Exchange.sol : atomicMatch_(), approveOrder_(), cancelOrder()

Recommendation: Improve code readability by removing the redundant return.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

L04. Explicit Uint Size

The code makes extensive use of implicit *uint* size declarations, which, while not posing any security issues, worsen the code's readability.

Paths:
./contracts/ExchangeCore.sol
./contracts/Exchange.sol

Recommendation: Replace uint declarations with explicit *uint256* type.

Found in: 6512502

Status: Fixed (Revised commit: 6e29e59)

L05. Redundant Function

The function *calculateCurrentPrice()* accepts the order as a parameter and returns *order.basePrice* with no other logic in it.

Path:
./contracts/ExchangeCore.sol : calculateCurrentPrice()

Recommendation: The function can be replaced by *order.basePrice* to improve code readability.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

L06. Storage Variables Packing

Storage variable declarations in *MysteryBoxManager.sol* can be rearranged to minimize the storage slots used and save gas on deployment.

Path:
./contracts/MysteryBoxManager.sol

Recommendation: Rearrange storage variables declarations.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

L07. Unused Variable

The storage variable *catgirlToken* is never used. Its upgradable counterpart, *uCatgirlToken,* is used instead.

Path:
./contracts/MysteryBoxManager.sol

Recommendation: Drop the unused variable.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

L08. Inefficient Gas Model

The variable *mysteryBoxFiatPrice* is set with *18 decimals*, but every time it's used, these *18 decimals* are removed, making them redundant.

This increases the Gas cost of the *calculateCatgirlSalePrice()* and *calculateBNBSalePrice()* invocations without any benefit.

Path:
./contracts/MysteryBoxManager.sol

Recommendation: Remove redundant decimal conversions and set the mysteryBoxFiatPrice value accordingly, to save gas on computations.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

L09. Missing Zero Address Check

The zero check for *developmentAddress* and *farmingAddress* is enforced in the functions that use them instead of in their setter functions.

The zero address check is also missing in other functions.

Path:
./contracts/MysteryBoxManager.sol : setAddress(), __MysteryBoxManager_init_unchained(), setPancakeRouter(), setCatGirlTokenAddress(), setCatgirlNFTAddress(), setConfigVfr();

Recommendation: Move the zero address check to the setter function.

Found in: 6512502

Status: Fixed (Revised commit: 6e29e59)

L10. Redundant Inheritance

Contracts *MysteryBoxManager.sol* and *Marketplace.sol* inherit from *Initializable.sol*, this is redundant because they also inherit from other upgradeable contracts.

Paths:
./contracts/MysteryBoxManager.sol
./contracts/Marketplace.sol

Recommendation: Remove the redundant inheritance.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

L11. Variable Naming

The name of the variable *pendingBox* is misleading, as it does not represent the pending boxes. It represents the pending orders, each of which contains a certain number of boxes.

Path:
./contracts/MysteryBoxManager.sol : buyCommonBoxWithCatgirl()

Recommendation: Change the variable name with something semantically consistent with its purpose.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

L13. Style Guide Violation

The provided projects should follow official guidelines.

Path:
./contracts/*

Recommendation: Follow the official Solidity guidelines.

Found in: 6512502

Status: Fixed (Revised commit: 6e29e59)

L13. Unused Function

The function *random()* is never used and can be dropped from the contract.

Path:
./contracts/MysteryBoxManager.sol : random()

Recommendation: Remove unused functions from the contract.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

L14. Unindexed Events

Having indexed parameters in the events makes it easier to search for these events using *indexed* parameters as filters.

Path:
./contracts/MysteryBoxManager.sol

Recommendation: Use the *"indexed"* keyword to relevant event parameters.

Found in: 6512502

Status: Fixed (Revised commit: 6e29e59)

L15. Missing Events

Events for critical state changes should be emitted for tracking things off-chain.

Path:
./contracts/MysteryBoxManager.sol : setMysteryBoxPrice(), setMaximumNumberOfBox(), setFundDistributionPercentage(), setAddress(), setPancakeRouter(), setCatgirlTokenAddress(), setCatgirlNFTAddress(), setCurrentSeason(), emergencyWithdraw(), processDistributionCatgirl(), processDistributionBNB(),

buyCommonBoxWithBNB(), setConfigVfr(), buyCommonBoxWithCatgirl(), claim()

Recommendation: Create and emit related events.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

L16. Deprecated Function

The *_setupRole()* function is deprecated, *_grantRole()* should be used instead.

Paths:
./contracts/Marketplace.sol : initialize()
./contracts/MysteryBoxManager.sol : __MysteryBoxManager_init_unchained()

Recommendation: Consider using a non deprecated function.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

L18. Redundant Code Block

Inside the function *buyCommonBoxWithCatgirl()* there is a require statement that checks the *balanceOf()* of the *msg.sender*, this is redundant since the *transferFrom()* function called right after will already perform this check.

Path:
./contracts/MysteryBoxManager.sol : buyCommonBoxWithCatgirl()

Recommendation: Delete the redundant code block.

Found in: 6512502

Status: Fixed (Revised commit: a8ea50c)

L18. Redundant Inheritance

PriceOracle does not need to inherit from Initializable, as it already inherits from other upgradeable contracts

CatgirlNFT does not need to inherit from Initializable and ERC721Upgradeable for the same reason.

Path:
./contracts/PriceOracle.sol
./contracts/CatgirlNFT.sol

Recommendation: Remove redundant inheritance.

Found in: a8ea50c

Status: Fixed (Revised commit: 6e29e59)

L19. Unindexed Events

Having indexed parameters in the events makes it easier to search for these events using *indexed* parameters as filters.

Path:
./contracts/CatgirlNFT.sol

Recommendation: Use the *"indexed"* keyword to track relevant event parameters.

Found in: a8ea50c

Status: Fixed (Revised commit: 6e29e59)

L20. Variable Type

canReborn variable should be a bool type instead of uint256, as its only meaningful values are 0 and 1.

Path:
./contracts/CatgirlNFT.sol

Recommendation: Change canReborn type to bool.

Found in: a8ea50c

Status: Fixed (Revised commit: 6e29e59)

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.