#HashEx
BLOCKCHAIN SECURITY

Metawin

smart contracts final audit report

October 2022

hashex.org

contact@hashex.org

Contents

1. Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and HashEx and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (HashEx) owe no duty of care towards you or any other person, nor does HashEx make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and HashEx hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HashEx hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HashEx, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed. HashEx owns all copyright rights to the text, images, photographs, and other content provided in the following document. When using or sharing partly or in full, third parties must provide a direct link to the original document mentioning the author (hashex.org).

2. Overview

HashEx was commissioned by the Metawins team to perform an audit of their smart contract. The audit was conducted between 13/07/2022 and 15/07/2022.

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts
- Formally check the logic behind given smart contracts.

Information in this report should be used for understanding the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

The code is available after 2bfad99 commit.

Update: the Metawins team has responded to this report. The updated code is available after 1e3af9c commit.

Update 2: the audited code with mainnet parameters was deployed to the Ethereum network at address 0x8652168693827ff341f2ad5AD9a091f2A975710A.

Update 3: the updated code was provided by the @metawins/raffles Bitbucket repository after d3e81cf commit.

Update 4: the updated code is located in the same repository after the 66f4834 commit.

The audited code was deployed to Ethereum network at address 0x99Efed3b4e43738d44c1F532135Cd16A240d983d.

2.1 Summary

Project name	Metawin
URL	https://metawin.xyz
Platform	Ethereum
Language	Solidity

2.2 Contracts

Name	Address
Basic721.sol	
IVRFConsumerv1.sol	
Manager.sol	0x99Efed3b4e43738d44c1F532135Cd16A240d983d

3. Found issues

High	3 (19%)
Medium	4 (25%)
Low	6 (38%)
Info	3 (18%)

C3. Manager.sol

ID	Severity	Title	Status
C3-01	High	Commission can reach 100%	Resolved
C3-02	High	Winner can be rerolled	Resolved
C3-03	High	Excessive owner's rights	Partially fixed
C3-04	Medium	Reentrancy in cancelRaffle()	Resolved
C3-05	Medium	Extra deposited funds are not returned	Resolved
C3-06	Medium	numEntriesPerUser are not updated	Resolved
C3-07	Medium	Funds may get locked on the contract	Resolved
C3-08	Low	Useful for search event fields are not indexed	Resolved
C3-09	Low	No events	Partially fixed
C3-10	Low	Possible indexation error	Acknowledged

C3-11	Low	Lack of mechanism to withdraw LINK excess	Acknowledged
C3-12	Low	Gas optimization	Partially fixed
C3-13	Low	getPriceStructForId() fails if non-existent id is passed	Partially fixed
C3-14	Info	No global setter for the platform commission	Resolved
C3-15	Info	Controversial variable naming	Resolved
C3-16	Info	Outdated oracle version	Acknowledged

4. Contracts

C1. Basic721.sol

Overview

NFT contract with pausable transfers. No issues were found.

C2. IVRFConsumerv1.sol

Overview

Chainlink oracle random number consumer interface. No issues were found.

C3. Manager.sol

Overview

Main lottery contract.

Issues

C3-01 Commission can reach 100%

High Resolved

Platform commission for the raised raffle funds may reach up to 100%.

```
function changePlatformPercentage(uint256 _raffleId, uint256 _percentage)
    public
    onlyRole(OPERATOR_ROLE)
{
    RaffleStruct storage raffle = raffles[_raffleId];
    // Don´t check the raffle status
    require(_percentage <= 100 && _percentage >= 0, "Wrong percentage");
    // the percentage must be set to basic points
```

```
    raffle.platformPercentage = _percentage * 100;
}
```

Recommendation

Limit the fee upper bound to a reasonable value and block the ability to change it after the raffle is in the accepted status.

Update

In the update platform commission, the percent change functionality was removed. Now it is only set during raffle creation and is required to be lower or equal to 50%.

C3-02 Winner can be rerolled

High Resolved

`EARLY_CASHOUT` and `CLOSING_REQUESTED` statuses supposedly already sent a request to VRF for a generation winning entry number. However, `setWinner()` doesn't consider that, and raffles with these statuses pass the requirement:

```
function setWinner(uint256 _raffleId)
    public
    payable
    nonReentrant
    onlyRole(OPERATOR_ROLE)
{
    ...
    require(
        raffle.status == STATUS.ACCEPTED ||
            raffle.status == STATUS.EARLY_CASHOUT ||
            raffle.status == STATUS.CLOSING_REQUESTED,
        "Raffle in wrong status"
    );
    ...
    IVRFConsumerv1 randomNumber = IVRFConsumerv1(chainlinkContractAddress);
    randomNumber.getRandomNumber(_raffleId, raffle.entries.length);
}
```

This results in winner reroll and gross gambling rules violation. The issue is also may be

referred to the Excessive owner's rights category since this can be abused by the owner to choose desired winner's address.

Recommendation

Remove `EARLY_CASHOUT` and `CLOSING_REQUESTED` from accepted status options in the requirement mentioned in the code snippet.

Update

Inappropriate statuses were removed from the function. The winner can't be re-rolled now.

C3-03 Excessive owner's rights

High Partially fixed

a. The admins can change the source of the VRF contract address;

b. Random number is actually obtained from backend service and can be compromised;

c. The operators can fill raffles entries with arbitrary addresses and draw an NFT among them;

d. The accounts with operator rights may cancel the raffle and all the funds sent by users for participating in the raffle will be sent to the address that is set by the owner.

Recommendation

a, c – Admins and operators must be `Timelock` contracts with a minimum delay of at least 24 hours. This won't stop the operators and admins from possible rights abuse but it will help users to be informed about upcoming changes.

b – Redesign contracts architecture that `Manager` is VRF consumer without any off-chain mediums except Chainlink's nodes.

d – implement redeem mechanism or remove cancel raffle functionality.

Update

The update fixed issues *a,b, and d*. The `Manager` contract is now derived from `VRFConsumerBase` and receives randomness directly from Chainlink oracle without backend intermediates. The raffle cancelation process was also reconsidered. Now, if a raffle is canceled, the users who bought entries in it can get a direct refund during the first 30 days after the raffle cancelation calling the `claimRefund()` function. Remained assets after the refund period will go to the owner. Issue *c* was acknowledged.

Team response

MetaWin intends to proceed with the current state of AMOE (Alternative Method of Entry) raffle entries by addressing the point.

Metawin is required by law in some jurisdictions to offer alternate methods of entry to prize draws.

In order to remain compliant, its contracts retain the flexibility to add entries by way of the manual process right up until the second the competition draws.

C3-04 Reentrancy in cancelRaffle()

Medium Resolved

The raffle status is changed after the external call. An arbitrary malefactor cannot exploit the vulnerability, but privileged users can sometimes drain all contract's assets.

NFT `seller` approves as an operator `destinationWallet` address. When the `destinationWallet` receives raised funds, in the callback it transfers NFT from `seller` to the `Manager` contract and cancel the same raffle again. Since the raffle status wasn't updated and the contract owns NFT, the transactions won't be reverted and NFT will be transferred to the `seller` and funds will be sent to `destinationWallet` once again causing double spent. The scheme can be repeated multiple times in one transaction and the number of funds spent is only limited by the initial `Manager` balance and max call depth stack in Solidity.

Recommendation

Update the raffle status to `CANCELLED` before funds transfer.

Update

Reentrancy exploit is not possible anymore.

C3-05 Extra deposited funds are not returned

Medium Resolved

While buying entries to the raffle, a user can attach greater `msg.value` than `priceStruct.price`. If so, the transaction won't be reverted and overpaid assets will be added to raffle funds.

```
function buyEntry(uint256 _raffleId, uint256 _id)
    public
    payable
    nonReentrant
{
    ...
    PriceStructure memory priceStruct = getPriceStructForId(_raffleId, _id);
    require(priceStruct.price > 0, "id not supported");
    require(
        msg.value >= priceStruct.price,
        "msg.value is less than the price"
    ); // 1722
   ...
}
```

Recommendation

Require `msg.value` to be equal to `priceStruct.price` or return the difference to the user.

Update

The requirement was changed to strict equality.

C3-06 numEntriesPerUser are not updated

Medium Resolved

While adding free players in entries with `giveBatchEntriesForFree()`, the method doesn't change the user's state in `numEntriesPerUser` mapping:

```
function giveBatchEntriesForFree(
    uint256 _raffleId,
    address[] memory _freePlayers
) public payable nonReentrant onlyRole(OPERATOR_ROLE) {
    ...
    uint256 freePlayersLength = _freePlayers.length;
    for (uint256 i = 0; i < freePlayersLength; i++) {
        address entry = _freePlayers[i];
        if (
            numEntriesPerUser[keccak256(abi.encode(entry, _raffleId))] +
                1 <=
            raffles[_raffleId].maxEntries
        ) raffles[_raffleId].entries.push(entry);
    }
    raffles[_raffleId].entriesLength = raffles[_raffleId].entries.length;
    ...
}
```

It only checks whether adding one to the current user's entries number won't exceed `maxEntries`, but doesn't increment the value then.

Recommendation

Increment user's entries in the raffle.

```
if (
    numEntriesPerUser[keccak256(abi.encode(entry, _raffleId))] +
        1 <=
    raffles[_raffleId].maxEntries
) {
    numEntriesPerUser[keccak256(abi.encode(entry, _raffleId))]++;
    raffles[_raffleId].entries.push(entry);
}
```

Update

The Metawins team applied the proposed solution.

C3-07 Funds may get locked on the contract

Medium Resolved

Functions `giveBatchEntriesForFree()` and `setWinner()` have a redundant `payable` modifier, although attached ETH is not required and `msg.value` is not used. Accidentally sent cryptocurrency will stay locked on the contract.

Recommendation

Remove the `payable` modifier from the functions.

C3-08 Useful for search event fields are not indexed

Low Resolved

The `nftAddress`, `nftId`, `winner`, `buyer` fields in events should be indexed as they can considerably ease the relevant logs searching process.

C3-09 No events

Low Partially fixed

We recommend emitting events on important value changes to be easily tracked off-chain. No events are emitted in `earlyCashOut()`, `setWinner()`, `addAdminRole()`, `addOperatorRole()`, `revokeOperatorRole()`, `setDestinationAddress()`, `setRandomNumberContractAddress()`, `changePlatformPercentage()`, `cancelRaffle()`, `emergencyChangeStatus()`, `setRequiredNFT()`.

Update

`setDestinationAddress()` still doesn't emit an event as well as `setFee()` added in the update.

C3-10 Possible indexation error

Low Acknowledged

In `createRaffle()` it is not checked `_prices` array length is less or equal to 5. If the array contains more `PriceStrictures` the transaction will end up with inconsistent indexation error

message. It can confuse an operator and negatively affects fault tolerance.

C3-11 Lack of mechanism to withdraw LINK excess

Low Acknowledged

The contract needs LINK tokens on its balance to get random values for raffles when choosing winners. These LINK tokens should be deposited to the contract beforehand, but there is no way to withdraw these tokens if in any case more LINK tokens than needed were sent to the contract.

Recommendation

We recommend adding a function that lets the contract owner withdraw any ERC20 tokens sent to the contract.

C3-12 Gas optimization

Low Partially fixed

1. Ended raffles can be deleted from the blockchain;

2. Excessive storage reads L350,356,357,360,361,362,377,378;

3. `keccak256` hash can be memorized L271,286;

4. All public functions visibility should be changed to external;

5. Role management functions are redundant and can be removed, `AccessControl` import includes the needed functionality;

6. Mappings data storing ended raffles info can be removed after rewards are transferred;

7. `VRFCoordinator`, `LinkToken` variables are not used and can be removed.

C3-13 getPriceStructForId() fails if non-existent id is passed

Low Partially fixed

Passing non-existent in raffle price structure id to `getPriceStructForId()` will cause the transaction to revert with an inconsistent indexation error message. Also `found` variable is redundant.

Recommendation

Change while loop on for loop with `(uint256 i = 0; i < 5; i++)` condition, where 5 is maximum `PriceStructure` array length in `prices` mapping.

Update

Regardless the Metawins team followed the recommendation, commented L384 results in the transactions with passed non-existent price ids won't be reverted and will eventually emit `EntrySold()` event.

C3-14 No global setter for the platform commission

Info Resolved

`changePlatformPercentage()` allows changing the developer's fee just for one specific raffle, but the default set value stays unchanged at 25% and can't be modified.

Update

Setter function was removed. Comission is set individually for a raffle.

C3-15 Controversial variable naming

Info Resolved

Actually implemented logic in the code contradicts to the `requiredCollections` variable naming and its description in the comment. Judging by the documented purpose, a user should have a token from all the collections in the list to buy an entry, although, in fact, it's sufficient to have just one token from available collections of a raffle. Consider giving the variable a name better reflecting its role in in the project.

C3-16 Outdated oracle version

Info Acknowledged

An old version of oracle is used. Consider calling the updated one.

Team response

When we started the Chainlink integration, VRF v2 looks still buggy while VRF v1 looks rock-solid and mature. The plan is to move to VRF v2 in the next version.

5. Conclusion

3 high, 4 medium, 6 low severity issues were found during the audit. 2 high, 4 medium, 1 low issues were resolved in the update.

The update considerably decreased the dependency level on the owner's account. Competitions are honest and winners can not be compromised. If a raffle is canceled refund assets are available for participants to claim during the first 30 days after raffle cancellation. Nevertheless, operators retain the right abuse opportunity allowing them to roll NFT among arbitrary addresses.

This audit includes recommendations on code improvement and the prevention of potential attacks. We recommend adding tests with coverage of at least 90% with any updates in the future.

Team response: The team has responded explaining the reason for the permission is to allow them to provide purely AMOE to all competitions as required by law in some locations.

Appendix A. Issues severity classification

- **Critical.** Issues that may cause an unlimited loss of funds or entirely break the contract workflow. Malicious code (including malicious modification of libraries) is also treated as a critical severity issue. These issues must be fixed before deployments or fixed in already running projects as soon as possible.
- **High.** Issues that may lead to a limited loss of funds, break interaction with users, or other contracts under specific conditions. Also, issues in a smart contract, that allow a privileged account the ability to steal or block other users' funds.
- **Medium.** Issues that do not lead to a loss of funds directly, but break the contract logic. May lead to failures in contracts operation.
- **Low.** Issues that are of a non-optimal code character, for instance, gas optimization tips, unused variables, errors in messages.
- **Info.** Issues that do not impact the contract operation. Usually, info severity issues are related to code best practices, e.g. style guide.

Appendix B. List of examined issue types

- Business logic overview
- Functionality checks
- Following best practices
- Access control and authorization
- Reentrancy attacks
- Front-run attacks
- DoS with (unexpected) revert
- DoS with block gas limit
- Transaction-ordering dependence
- ERC/BEP and other standards violation
- Unchecked math
- Implicit visibility levels
- Excessive gas usage
- Timestamp dependence
- Forcibly sending ether to a contract
- Weak sources of randomness
- Shadowing state variables
- Usage of deprecated code

 contact@hashex.org

 @hashex_manager

 blog.hashex.org

 linkedin

 github

 twitter

#HashEx
BLOCKCHAIN SECURITY