Wanaka Security Analysis

Overview Abstract

In this report, we consider the security of the Wanaka project. Our main task is to find and describe security issues in the smart contracts of the platform to the team.

Limitations and use of the report

Broadly speaking, the assessments can not uncover all vulnerabilities of the given smart contracts, thus, it is not guaranteed that the system is secured even if no vulnerabilities are found. The focus of the assessments was limited to given smart contracts, other contracts were excluded (including external libraries or third party codes).

Summary

We have found **3** high severity issues, **2** medium severity issues and **7** low severity issues.
All of the issues have been adjusted and not presented in the latest source codes.

Recommendations

We recommend the team to fix all issues, as well as test coverage to ensure the security of the contracts.

Assessment Overview

Scope of the audit

Source codes for the audit were initially taken from the commit hash 2802f546c23df78e2dd86096f86390078262cb86. In-scope contracts for audit:

- All contracts in the ***contracts/*** folder, including: *assets*, *commons*, *erc20*, *interfaces*, *libs* and *OrderContract.sol*

Other source codes are out of scope for this report.

After the initial report for the source codes above, the team has fixed all findings. The final source codes for the audit are taken from the commit hash 31bfadc60648d80f3a66c2653cee4f0c05328df3

System Overview

Wanaka Farm setting is inspired by a dreamy beautiful town from the South Island of New Zealand, called Wanaka. The town of Wanaka is situated at the southern end of Lake Wanaka, surrounded by gorgeous natural beauty of mountains, lakes, and forest. The game feel will incorporate as much sight and sound of Wanaka to evoke natural beauty and immersive relaxing experience as possible.

At Wanaka Farm, players will immerse themselves in the role of a happy farmer by cultivating lands, crop farming, breeding pets, and decorating their own virtual
land. From then, each participant will contribute to building a whole metaverse combining multiple unique customized farms.

Wanaka provides lands and several in-game assets including Seed, Growing, Harvested, Product and Building items which are tradeable on Wanatrade - a Wanaka's marketplace.

The full details of the system and implementation can be found in the Green Paper.

Findings

We have found 3 high severity issues, 2 medium severity issues and 7 low severity issues.
All of the issues have been adjusted and not presented in the latest source codes.

[Fixed] [High] OrderContract: No withdrawal function to withdraw fees

In the **OrderContract**, it is taking fees when listing a new order or executing an order/offer. Fees are kept in the **OrderContract**, however, there is no withdrawal function to retrieve the fees.

Suggestion: Transferring fees to external addresses to avoid keeping fees in the contract. If adding a new withdrawal function, it must not withdraw tokens from buyers' offers.

Comment: This issue has been adjusted by using external addresses to store fees.

[Fixed] [High] OrderContract: An operator can take listing fees from sellers

In the **OrderContract**, when cancelling an order, the listing fee will be transferred back to the caller. In logic, a seller or an operator can call

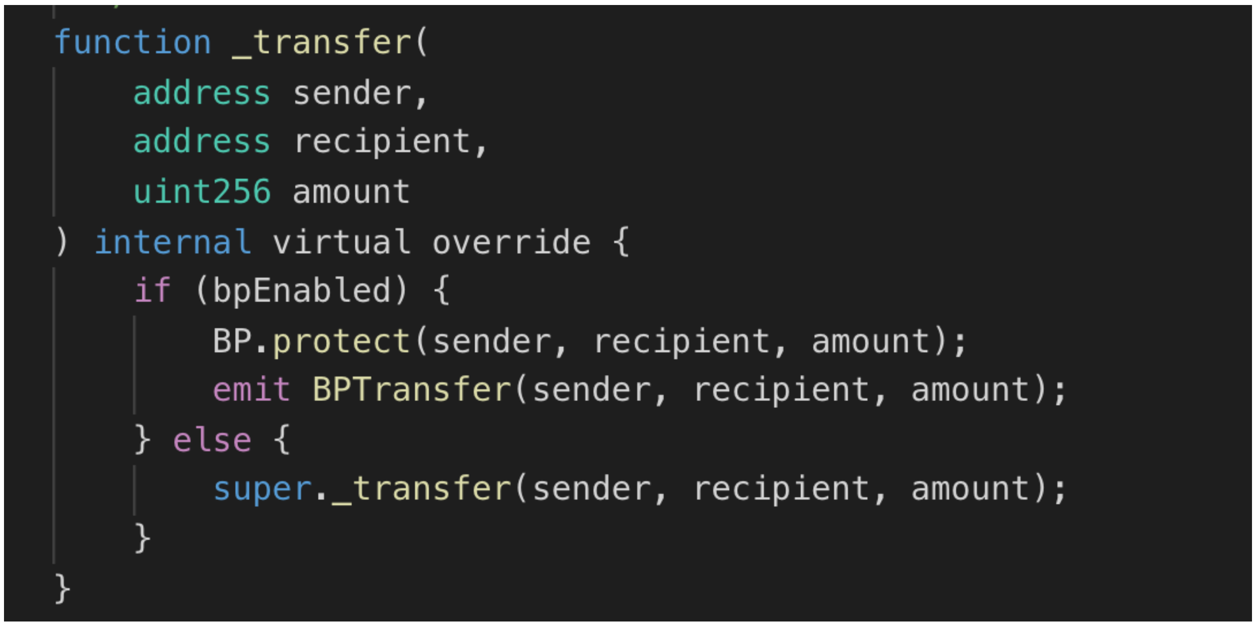
_cancelOrder function to cancel an order, thus, an operator can call **_cancelOrder** function to cancel orders of other sellers and get listing fees.

Suggestion: Listing fees should be transferred back to the order's seller instead of **msg.sender**.

Comment: This issue has been adjusted by sending listing fees back to the order's seller.

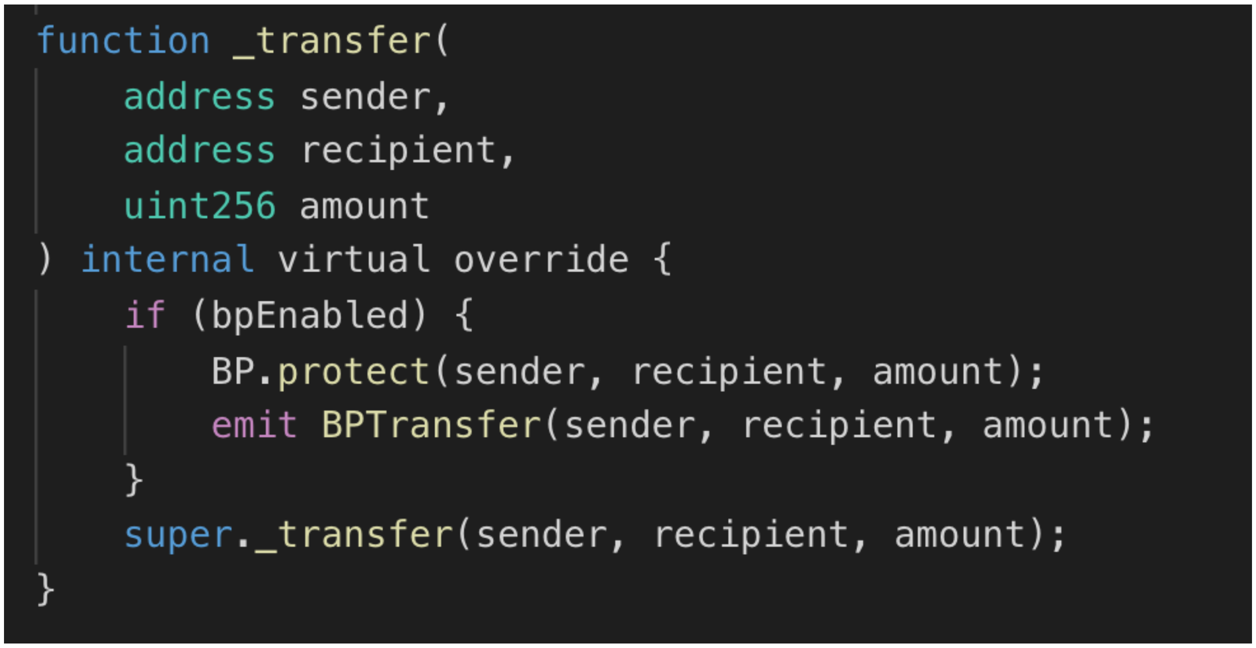
[Fixed] [High] WanakaFarm: Wrong implementation for BotProtection

In the **WanakaFarm**, it is using **BotProtection** to check for every transfer action. However, the implementation is wrong:



```
function _transfer(
    address sender,
    address recipient,
    uint256 amount
) internal virtual override {
    if (bpEnabled) {
        BP.protect(sender, recipient, amount);
        emit BPTransfer(sender, recipient, amount);
    } else {
        super._transfer(sender, recipient, amount);
    }
}
```

<u>Suggestion:</u> The correct implementation should be always calling **super._transfer()** as below.



```
function _transfer(
    address sender,
    address recipient,
    uint256 amount
) internal virtual override {
    if (bpEnabled) {
        BP.protect(sender, recipient, amount);
        emit BPTransfer(sender, recipient, amount);
    }
    super._transfer(sender, recipient, amount);
}
```

Comment: This issue has been adjusted and not presented in the latest code.

[Fixed] [Medium] WLand: Wrong emitted index for a new release

When adding a new release, an event **NewReleaseAdded** with the new index equals the last index + 1 will be emitted. However, the event is using the same newIndex for the first and the second releases, which is wrong according to the design.
Example:

- For the first release, since it is the first release, the **lastReleaseIndex** is set to be 0, thus, the **newReleaseIndex** = **lastReleaseIndex** + 1 = 1.

- For the second release, the **lastReleaseIndex** is set to be 0 again (**releases.length - 1**), thus, the **newReleaseIndex** = **lastReleaseIndex** + 1 = 1.

Suggestion: The first release should use 0 as the new release index to emit in the event.

Comment: This issue has been adjusted by using (releases.length - 1) as the value to emit.

[Fixed] [Medium] GameItems: addItemType function allows to add an item to an existing itemId

In the **addItemType** function, it is not checked if the **itemId** exists before storing new data for the **itemId**, thus, it could potentially override an existing **itemId**.

Suggestion: Add a check if the itemId exists before storing data.

Comment: This issue has been adjusted and not presented in the latest code.

[Fixed] [Low] PermissionRight: No functions to retrieve the list of admins/operators

There is no function to get the list of admin/operators, only functions to check if an address is an admin/operator, which is difficult to manage.

Suggestion: Should use **EnumerableSet** from **OpenZeppelin** to implement the admin/operator groups.

Comment: This issue has been adjusted by using EnumerableSet for AddressSet.

[Fixed] [Low] Uncommon Return Bool for functions that are either reverted or true

In some functions like ... , it returns a bool value. However, in all functions, the result is either reverted or true, which is redundant to add a return value.

Suggestion: Should remove the return param.

Comment: This issue has been adjusted by removing returning params.

[Partial-Fixed] [Low] Shorten revert messages

In the code, some reverted messages are too long, which will increase the bytecode size of the contract, as well as gas consumption.

Suggestion: Should try to use shortened revert messages.

Comment: The team wants to have a clear revert message for debugging easier. Some messages are shorten, while the others are kept as current implementations.

[Fixed] [Low] Remove redundant imports, casting, unused attributes etc.

1. **GameItems**
    a. Redundant imports: **GameCurrency**, **IERC20**, **Counters**, **SafeCast.**
    b. Unused variables: **_packageIds.**

2. **NftItems**
    a. Redundant imports: **GameCurrency, SafeMath.**
    b. Redundant casts:
        i. **rarity** (from **Rarity** to **Rarity**).

3. **WLand**
    a. Redundant imports: **console.sol**, **SafeMath**.
    b. Unused variables: **endTime** in **ReleaseVersion**.
    c. Redundant casts:
        i. **environment** (from **Environment** to **Environment**).
        ii. **maxSupply**: define as uint128 type instead of uint256 and downcast to uint128.

4. **WanakaFarm**
    a. Redundant imports: **SafeMath** and **Context**.

5. **OrderContract**
    a. Redundant imports: **console.sol**, **SafeMath**, **Context**, **Strings**, **Utils.**

6. **utils**
    a. Redundant functions: **addressToString**, **concatenate**, **bytesToBytes32** are unused.

Suggestion: Remove redundant imports, functions, variables, castings, etc.

Comment: This issue has been adjusted and not presented in the latest code.

[Fixed] [Low] Design: Remove checks for balance and allowance before transferring tokens

In contracts like **WLand**, **OrderContract**, it is always checking for balance and allowance before making a transfer/transferFrom call. However, it is redundant as in the function like transfer/transferFrom, the balance and allowance have been checked by SafeMath.

Suggestion: Remove checking for balance and allowance, use **SafeERC20** for **ERC20** operations.

Comment: This issue has been adjusted and not presented in the latest code.

[Acknowledged] [Low] Design: Overload for admin/operator

There are too many operations that are only done by either an admin or an operator, some operations are frequent and the team needs to run a server to monitor and process, which means some of the keys have to be stored on the server, thus, it is easier to be compromised.

Comment: This issue has been adjusted and not presented in the latest code.

[Fixed] [Low] Others minor issues

There are other minor issues that we have found and discussed with the team internally to fix, mostly code convention/styling, typos, redundant logics, etc. The team has adjusted all issues in the latest version of the source codes.

Comment: All issues have been adjusted and not presented in the latest code.

Testings

The team has provided tests for all contracts, they do not provide 100% full coverage yet. We strongly recommend the team to add full test coverage to ensure the security of the codes.