A CONSENSYS DILIGENCE AUDIT REPORT

Skyweaver

Date	April 2020
Auditors	Sergii Kravchenko

1 Executive Summary

This report presents the results of our engagement with **Horizon Games** to review the security of their smart contracts for the **SkyWeaver** game.

The review was conducted over one week, from April 1st to April 10th by Sergii Kravchenko. A total of 5 person-days were spent.

During this week, I focused primarily on factories that produce new cards/items/tokens, trust-related issues, the RNG mechanism, and re-entrancy attacks (especially within ERC-1155 token callbacks).

The review identified 7 issues of different severities (see section 4).

The allotted time for the review (1 person-week) was deemed insufficient from the start to do a comprehensive audit of the whole system. As such, some compromises had to be made on the completeness of the review.

2 Scope

Our review is focused on the commit hash `bde0c184db6168bf86656a48b12d5747950b54d9` of https://github.com/horizon-games/SkyWeaver-contracts repository. The list of files in scope can be found in the Appendix.

3 Security Specification

This section describes, **from a security perspective**, the expected behavior of the system under the review. It is not a substitute for documentation.

3.1 Actors

The relevant actors are listed below with their respective abilities:

- ***System owners*** - there are multiple contracts in the system that have an `owner` that potentially can be different addresses in different contracts, but logically it's the same owner (Horizon admins).
  - `SilverCardsFactory`
    - can register/deregister card IDs.
    - can change the price.
    - withdraw arcadeum coins from the contract.
  - `EternalHeroesFactory`
    - can register/deregister item IDs.
    - ***cannot*** change price, pricing is set during the contract creation.
    - withdraw arcadeum coins from the contract.
  - `GoldCardsFactory`
    - can register/deregister card IDs.
    - can change the price.
    - can change refund.
    - can change `rngDelay`.
  - `WeaveFactory`
    - can mint `weave` for anyone, minting is limited per time. The rate is fixed and cannot be changed.
  - `SWSupplyManager`
    - can manage factories.
    - can set maximum supply for each token ID.
    - can lock some token IDs.
- ***Token holders*** - anyone who has enough tokens (arcadeum coins or weave) can buy cards/items from factories.
- ***Anyone*** - can mine gold card for any already committed gold order and recommit.

3.2 Trust Model

In any smart contract system, it's important to identify what trust is expected/required between various actors. For this review, we established the following trust model:

The system has owners that have some centralized power, like changing prices/refunds. From other perspectives, the system is pretty much permissionless for the users, anyone with tokens can buy/melt items for a predefined/limited price.

Random number generation is not completely trustless; it may potentially be manipulated by miners. However, unless gold cards become expensive, there is no clear incentive for miners to do so.

3.3 Important Security Properties

The following is a non-exhaustive list of security properties that were discovered during this review:

- To decrease trust assumptions, `owner` can lock some token IDs in `SWSupplyManager` to ensure that only already existing factories with already existing permissions can mint items with these IDs.
- Sometimes price can be changed by the admin using a front-running technique. In that case, users have mechanisms that allows them to cancel their order if the price is too high. Also, the owner can modify the gold cards list between committing to mine a gold card and actually mining it.
- In some factories prices include decimals, in some - not.

4 Issues

Each issue has an assigned severity:

- Minor issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.

- Medium issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- Major issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Critical issues are directly exploitable security vulnerabilities that need to be fixed.

4.1 Gold order size should be limited Major ✓ Addressed

Resolution

Addressed in horizon-games/SkyWeaver-contracts#9 by adding a limit for cold cards amount in one order.

Description

When a user submits an order to buy gold cards, it's possible to buy a huge amount of cards. `_commit` function uses less gas than `mineGolds`, which means that the user can successfully commit to buying this amount of cards and when it's time to collect them, `mineGolds` function may run out of gas because it iterates over all card IDs and mints them:

code/contracts/shop/GoldCardsFactory.sol:L375-L376

```
// Mint gold cards
skyweaverAssets.batchMint(_order.cardRecipient, _ids, amounts, "");
```

Recommendation

Limit a maximum gold card amount in one order.

4.2 Price and refund changes may cause failures Major ✓ Addressed

Resolution

Addressed in horizon-games/SkyWeaver-contracts#3.

> Fix involves burning the weave when the commit occurs instead of when the minting of the gold cards occur.

Description

Price and refund for gold cards are used in 3 different places: commit, mint, refund.

Weave tokens spent during the commit phase

code/contracts/shop/GoldCardsFactory.sol:L274-L279

```
function _commit(uint256 _weaveAmount, GoldOrder memory _order)
  internal
{
  // Check if weave sent is sufficient for order
  uint256 total_cost = _order.cardAmount.mul(goldPrice).add(_order.feeAmount
  uint256 refund_amount = _weaveAmount.sub(total_cost); // Will throw if ins
```

but they are burned `rngDelay` blocks after

code/contracts/shop/GoldCardsFactory.sol:L371-L373

```
// Burn the non-refundable weave
uint256 weave_to_burn = (_order.cardAmount.mul(goldPrice)).sub(_order.cardAm
weaveContract.burn(weaveID, weave_to_burn);
```

If the price is increased between these transactions, mining cards may fail because it should burn more `weave` tokens than there are tokens in the smart contract. Even if there are enough tokens during this particular transaction, someone may fail to melt a gold card later.

If the price is decreased, some `weave` tokens will be stuck in the contract forever without being burned.

Recommendation

Store `goldPrice` and `goldRefund` in `GoldOrder`.

4.3 Re-entrancy attack allows to buy `EternalHeroes` cheaper Major ✓ Addressed

Resolution

Addressed in horizon-games/SkyWeaver-contracts#4.

> Minting tokens before sending refunds. Subsequent PR will also add re-entrancy guard for all shops.

And re-entrancy guard added here: horizon-games/SkyWeaver-contracts#10

Description

When buying eternal heroes in `_buy` function of `EternalHeroesFactory` contract, a buyer can do re-entracy before items are minted.

code/contracts/shop/EternalHeroesFactory.sol:L278-L284

```
uint256 refundAmount = _arcAmount.sub(total_cost);
if (refundAmount > 0) {
  arcadeumCoin.safeTransferFrom(address(this), _recipient, arcadeumCoinID, r
}

// Mint tokens to recipient
factoryManager.batchMint(_recipient, _ids, amounts_to_mint, "");
```

Since price should increase after every `N` items are minted, it's possible to buy more items with the old price.

Recommendation

Add re-entrancy protection or mint items before sending the refund.

4.4 Supply limitation misbehaviors Medium ✓ Addressed

Resolution

Logic remains unchanged as it's the desired behaviour. But the issue is mitigated in horizon-games/SkyWeaver-contracts#5 by renaming the term "currentSupply" to "currentIssuance" and "maxSupply" to "maxIssuance" for maximum clarity.

Description

In `SWSupplyManager` contract, the `owner` can limit supply for any token ID by setting `maxSupply`:

code/contracts/shop/SWSupplyManager.sol:L149-L165

```
function setMaxSupplies(uint256[] calldata _ids, uint256[] calldata _newMaxS
  require(_ids.length == _newMaxSupplies.length, "SWSupplyManager#setMaxSupp

  // Can only *decrease* a max supply
  // Can't set max supply back to 0
  for (uint256 i = 0; i < _ids.length; i++ ) {
    if (maxSupply[_ids[i]] > 0) {
      require(
        0 < _newMaxSupplies[i] && _newMaxSupplies[i] < maxSupply[_ids[i]],
        "SWSupplyManager#setMaxSupply: INVALID_NEW_MAX_SUPPLY"
      );
    }
    maxSupply[_ids[i]] = _newMaxSupplies[i];
  }

  emit MaxSuppliesChanged(_ids, _newMaxSupplies);
}
```

The problem is that you can set `maxSupply` that is lower than `currentSupply`, which would be an unexpected state to have.

Also, if some tokens are burned, their `currentSupply` is not decreasing:

code/contracts/shop/SWSupplyManager.sol:L339-L345

```
function burn(
  uint256 _id,
  uint256 _amount)
  external
{
  _burn(msg.sender, _id, _amount);
}
```

This unexpected behaviour may lead to burning all of the tokens without being able to mint more.

Recommendation

Properly track `currentSupply` by modifying it in `burn` function. Consider having a following restriction `require(_newMaxSupplies[i] > currentSupply[_ids[i]])` in `setMaxSupplies` function.

4.5 Owner can modify gold cards distribution after someone committed to buy Medium Won't Fix

Resolution

The client decided not to fix this issue with the following comment:

> This issue will be addressed by having the owner be a delayed multisig, such that users will have time to witness a change in the distribution that is about to occur.

Description

When a user commits to buying a gold card (and sends weave), there is an expected distribution of possible outcomes. But the problem is that `owner` can change distribution by calling `registerIDs` and `deregisterIDs` functions.

Additionally, `owner` can buy any specific gold card avoiding RNG mechanism. It can be done by deleting all the unwanted cards, mining the card and then

returning them back. And if `owner` removes every card from the list, nothing is going to be minted.

Recommendation

There are a few possible recommendations:

- Fix a distribution for every order after `commit` (costly solution).
- Make it an explicit part of the trust model (increases trust to the admins).
- Cancel pending orders if gold cards IDs are changed.

4.6 A buyer of a gold card can manipulate randomness

Medium Won't Fix

Resolution

The client decided not to fix this issue with the following comment:

> We hereby assume that Horizon will always be willing to mine gold cards even at a loss considering the amount of gold cards that can be created per week is limited. If in practice this becomes a problem, we can upgrade this factory.

Description

When a user is buying a gold card, `_commit` function is called. After `rngDelay` number of blocks, someone should call `mineGolds` function to actually mint the card. If this function is not called during 255 blocks (around 1 hour), a user should call `recommit` to try to mint a gold card again with a new random seed. So if the user doesn't like a card that's going to be minted (randomly), user can try again until a card is good. The issue is medium because anyone can call `mineGolds` function in order to prevent this behaviour. But it costs money and there's no incentive for anyone to do so.

Recommendation

Create a mechanism to avoid this kind of manipulation. For example, make sure there is an incentive for someone to call `mineGolds` function

4.7 A refund is sent to recipient Medium Won't Fix

Resolution

The client decided not to fix this issue with the following comment:

> It's unlikely users will send inexact amount since price is fixed. If this becomes a problem in practice we can re-deploy the factory with this added functionality.

Description

When a refund is sent, it's sent to `recipient`. In case if a user wants to keep game items and money separate, it makes sense to send a refund back to `from` address.

Recommendation

Since there may be different use cases, consider adding `refundAddress` to order structure.

4.8 Randomness can be manipulated by miners Minor Won't Fix

Resolution

The client decided not to fix this issue with the following comment:

> For miners to be able to profit, they would have to forfeit multiple blocks and the desired gold cards would have to be very expensive in the first place (e.g. in the $10k) for it to be worth it for them. In practice, there are also other oppotunities

for miners that offer better returns, but if it ever turned out to be a problem, we would see it coming and we can then use a more secure and expensive source of RNG as the gold cards would be very expensive and the additional cost would be worth it.

Description

Random number generator uses future `blockhash` as a seed. So it's possible for miners to manipulate that value in order to get a better gold card. The issue is minor because it only makes sense if the cost of the card is high enough to do the extra work on the miner side.

Recommendation

Use better RNG algorithms if the price of gold cards is high enough for the miners to start manipulation.

Appendix 1 - Files in Scope

This audit covered the following files:

File	SHA-1 hash
shop/EternalHeroesFactory.sol	ba8e5e4
shop/GoldCardsFactory.sol	1240d2c
shop/SWSupplyManager.sol	f9c1825
shop/SilverCardsFactory.sol	23a3cb2
shop/WeaveFactory.sol	a4058da
tokens/SkyweaverAssets.sol	1e1f884
tokens/SkyweaverCurrencies.sol	8a5ebc8
abstract/AbstractERC1155MintBurn.sol	5d2e041
utils/Ownable.sol	15e6d2b

Appendix 2 - Disclosure

ConsenSys Diligence (“CD”) typically receives compensation from one or more clients (the “Clients”) for performing the analysis contained in these reports (the “Reports”). The Reports may be distributed through other means, including via ConsenSys publications and other distributions.

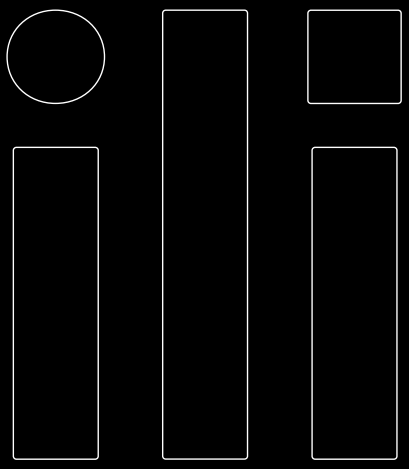
The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any Third-Party by virtue of publishing these Reports.

PURPOSE OF REPORTS The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of Solidity code and only the Solidity code we note as being within the scope of our review within this report. The Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond Solidity that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty.

CD makes the Reports available to parties other than the Clients (i.e., “third parties”) – on its website. CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

LINKS TO OTHER WEB SITES FROM THIS WEB SITE You may, through hypertext or other computer links, gain access to web sites operated by persons other than ConsenSys and CD. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that ConsenSys and CD are not responsible for the content or operation of such Web sites, and that ConsenSys and CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or mean that ConsenSys and CD endorses the content on that Web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the Reports. ConsenSys and CD assumes no responsibility for the use of third party software on the Web Site and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

TIMELINESS OF CONTENT The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice. Unless indicated otherwise, by ConsenSys and CD.



BLOG
TOOLS
RESEARCH
ABOUT
CONTACT
CAREERS
PRIVACY POLICY

Stay up-to-date on our latest offerings, tools, and the world of blockchain security.

Email*

e-mail address

→

POWERED BY CONSENSYS