SOLIDIFIED

Audit Report for Immutable on June 9th, 2020.

Summary

Audit Report prepared by Solidified for Immutable covering the Gods Unchained Season 1 smart contracts (and their associated components).

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The debrief took place on June 9th, 2020, and the final results are presented here.

Audited Files

Files contained in folder `https://github.com/immutable/platform-contracts/tree/develop/contracts/gods-unchained/contracts/s1`

Notes

The audit was initially performed on commit `280d7702c02e74915c432f5fc7ed56fbf84c52c0` of the repository listed in the section above, using Solidity compiler version `0.5.11`.

Intended Behavior

The smart contracts implement Gods Unchained Season 1 Packs, Chests, Referrals, Sale and Vendors.

SOLIDIFIED

Audit Report for Immutable on June 9th, 2020.

Issues Found

Critical

Major

1. S1Sale.sol and S1Vendor.sol: Multiple Issues

A. It's possible for anyone to get the funds currently in the `S1Sale`
If, for some reason, there're funds left on the sale contract, anyone can claim it by simply calling the `PurchaseFor` function with a destination address as `vendor`.

B. If it wasn't for issue **A**, the sale contract would roll over any funds from a previous purchase to the next one.
The contract makes no control of the initial and end balance inside a single transaction, making it possible to have leftovers from previous transactions, which wouldn't be possible to retrieve.

C. Both contracts implement a payable fallback with no logic, meaning they will accept any incoming ether with no data or with data that packs an erroneous call.

If funds are sent directly to the contracts, they will be used for the next purchase (either paying for the purchase or being reimbursed to the purchaser).

Recommendation
The idea of this contract is to make it simpler to perform multiple purchases to different vendors in a row, but it could be useful to add some more restrictions, instead of sending the whole balance back and forth. If there're any wrong address in the `requests[]` array, it might cause funds to be lost.

A better approach is to maintain a registry of known and valid Vendors, and disallow calls for other addresses. Another option is to implement a view function in `S1Vendor` that returns the amount of ETH necessary to the purchase and forward only that defined amount.

Lastly, remove the payable fallbacks, accepting funds only on direct calls to functions.

Amended [10.06.2020]
The issue was fixed and is no longer present in commit `c4e14af232d2d861a7000094f4cf931857d901ec`. The payable fallback functions mentioned in the issue are still present.

Minor

2. S1Vendor.sol: Return values of external calls is not checked

---

Function `purchaseFor` performs two low level calls with no data, to send ether to the referrer, and the change back to the buyer. The result of both calls is not checked. If the first call fails, the buyer will be reimbursed the value that was supposed to be sent to the referrer. If the second call fails, the change will remain in the contract, and be used (deliberately or not) for the next purchase performed (same effect as #1).

Recommendation
Check if external calls succeeded. Since the contract uses its own balance only while purchases are performed, implementing a `require(address(this).balance == 0)` before execution ends will prevent purchase values from remaining in the contract (and being used in the subsequent purchase).

Amended [10.06.2020]
The issue was fixed and is no longer present in commit `c4e14af232d2d861a7000094f4cf931857d901ec`.

Notes

3. Referral.sol: line 68: _haflDenom() is only ever called on 100

`_haflDenom()` is only ever called on 100, so will always return 50.

Recommendation
Consider removing it to improve readability and reduce unnecessary complexity.

Amended [10.06.2020]
The issue was fixed and is no longer present in commit `c4e14af232d2d861a7000094f4cf931857d901ec`.

4. S1Vendor: Users can get a discount by self-referring

There's no restriction on valid addresses for referrals, which basically allows for users self-referring and getting back some of the purchase.

Recommendation
There isn't a mitigation possible to make it sybil resistant.

Immutable's response [10.06.2020]
"This is known, we are comfortable with this."

5. RarityProvider: Mythic rarity isn't present anywhere in the code

The Rarity enum defines a `Mythic` rarity, but it isn't present anywhere else in the code on scope.

Amended [10.06.2020]
The issue was fixed and is no longer present in commit `c4e14af232d2d861a7000094f4cf931857d901ec`.

6. Consider updating compiler version

Contracts are using Solidity 0.5.11, consider updating to a later version, as it will contain the latest bug fixes, known bugs that were active in 0.5.11:

```
    "0.5.11": {
        "bugs": [
            "ImplicitConstructorCallvalueCheck",
            "TupleAssignmentMultiStackSlotComponents",
            "MemoryArrayCreationOverflow",
```

```
            "privateCanBeOverridden",
            "YulOptimizerRedundantAssignmentBreakContinue0.5"
        ],
        "released": "2019-08-12"
    }
```

None of these bugs directly affect the in scope smart contracts.

Immutable's response [10.06.2020]
"We've tried - unfortunately there are dependency issues blocking this for now. There are some behaviours in 0.5.11 that we require for old contracts, and we haven't yet set up a system capable of compiling some contracts with one compiler, and others with another."

SOLIDIFIED

Audit Report for Immutable on June 9th, 2020.

Closing Summary

Immutable's Gods Unchained Season 1 contracts contain two issues, with two one them being manor, and one of minor severity, along with several areas of note.

We recommend all issues are amended, while the notes are up to Immutable's discretion, as they mainly refer to improving the operation of the smart contract and best practices.

Amended [10.06.2020]
All issues were fixed and are no longer present in commit `c4e14af232d2d861a7000094f4cf931857d901ec`.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of the Immutable platform or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.