SOLIDIFIED

Audit Report for Wyvern Exchange. March 7, 2018.

Summary

Audit Report prepared by Solidified for Wyvern Exchange covering the Wyvern Protocol Audit Specification contracts. This report only refers to changes made since the last audit submitted on Feb 16, 2018 and based on commit `060314a275c9821e77c784763b9423fc38acecaa.`

Process and Delivery

Per client's request, one Solidified expert performed an unbiased and isolated audit of the below contracts. The final results are presented here.

Audited Files

The following files were covered during the audit:

- ProxyRegistry.sol
- AuthenticatedProxy.sol
- WyvernProxyRegistry.sol
- WyvernExchange.sol
- Exchange.sol
- Exchange-core
- SaleKindInterface.sol
- TokenRecipient.sol
- ReentrancyGuarded.sol
- ArrayUtils.sol
- WyvernAtomicizer.sol
- DelegateProxy.sol
- TokenTransferProxy.sol

Notes:
The audited versions are the ones present in the commit
`fce3320dcb262dabb6084f961f43baca22222275`

Intended Behavior

The purpose of these contracts is to facilitate exchange of non fungible assets in the ethereum blockchain

SOLIDIFIED

Audit Report for Wyvern Exchange. March 7, 2018.

Issues Found

1. Balance invariant can't be assured

It is possible to forcibly send ether to a contract, without triggering the execution of any code, and therefore, the check `require(this.balance == 0)` can fail and stop the contract from executing any atomic match operation.

Obs. This issue has also been raised during the simultaneous bug bounty on Solidified platform.

2. Protocol Fee can be avoided

If a transaction is being made using the protocol fee method, both parties can set all their order fee to zeros and still succeed. This happens because of this part:

ExchangeCore.sol line 305-308

```
/* If using the split fee method, order must have sufficient protocol fees. */
        if (order.feeMethod == FeeMethod.SplitFee && (order.makerProtocolFee <
minimumMakerProtocolFee || order.takerProtocolFee < minimumTakerProtocolFee)) {
            return false;
        }
```

If the chosen method is ProtocolFee, the first part of the if statement returns false and the execution proceeds without checking any fee parameter.

3. Beware of malicious token implementations

Accepting any ERC20 token opens the protocol, and its users, for a new class of attacks from malicious tokens, which are contracts that comply to the interface but not necessarily implement it correctly. It's fairly easy to disguise a bad token as an honest one, even so when considering that many users might not be proficient in reading token contracts.

An example attack is a malicious ERC20 that is built to avoid protocol fees. It's possible because the fee recipient is always known and therefore, the token can ignore any transfer to that specified address.

This is not a issue to the protocol itself, but Wyvern and its users might be greatly affected by it, so it's prudent to raise such issue.

Closing Summary

A major and a minor issues were found, and therefore, it's advised to fix it before proceeding.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of the Wyvern Exchange. This audit does not provide a security or correctness guarantee of the audited smart contracts. 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.