SOLIDIFIED

Audit Report for Wyvern Exchange February 16, 2018.

Summary

Audit Report prepared by Solidified for Wyvern Exchange covering the Wyvern Protocol Audit Specification contracts.

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

Notes:
The audited versions are the ones present in the commit
`060314a275c9821e77c784763b9423fc38acecaa`

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 February 16, 2018.

Issues Found

1. Low level calls might succeed if contract is non-existent

If an order target is a contract that has self destruct mechanism, a malicious party can kill the target right before the atomic match and still get the transaction funds. This happens because low level calls return true for non-existent contracts.

This issue has a limited surface attack, but it's present in the report to raise awareness of such scenario.

2. Give preference for 'bytes32' over 'string'

For constant values smaller than 32 bytes, give preference for a bytes32 type, since they are much cheaper than string types, which are dynamically sized.

3. Remove empty constructor

WyvernProxyRegistry.sol - lines 22-28

```
/**
   * @dev Create a WyvernProxyRegistry instance
   */
  function WyvernProxyRegistry ()
      public
  {
  }
```

Since there's no code execution, there's no need for a constructor function.

4. Consider implementing versioning mechanism in the Exchange

Since there might be more than one deployed exchange at a time, it might be useful to implement a versioning mechanism in the WyvernExchange contract itself. That way users(and even the dapp frontend) can be sure of which version they are interacting with.

5. Consider using uint256 types

Although uint and uint256 are equivalent, it's a best practice to use the more explicit version.

6. Newer compiler version available

Changing the solidity version pragma to the latest version (`^0.4.20`) as of this writing, to enforce the use of an up-to-date compiler, is recommended.

List of known compiler bugs and their severity can be found here:
https://etherscan.io/solcbuginfo

Closing Summary

No critical or major issues were found, but it's advised to consider implementing a few of the suggestions present in this report.

Disclaimer

Audit Report for Wyvern Exchange February 16, 2018.

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.