SOLIDIFIED

Audit Report for BetX January 21, 2019.

Summary

Audit Report prepared by Solidified for NextGen covering the the BetX smart contract repository.

Process and Delivery

Three independent Solidified experts performed an unbiased and isolated audit of the below smart contracts. The debriefing took place on January 21, 2019.

Audited Files

The following files were covered during the audit:

- BaseTokenRegistry.sol
- ERC223ReceivingContract.sol
- ERC223StandardToken.sol
- Escrow.sol
- MarketRegistry.sol
- Migrations.sol
- TransactionFeePool.sol
- ChallengeDeposit.sol
- CommitteeManager.sol
- DisputeManager.sol
- DisputeVote.sol
- OptInList.sol
- OutcomeReporter.sol
- SystemParameters.sol
- LibBytes.sol
- LibMarket.sol
- LibOrder.sol
- LibOrderAmounts.sol
- LibString.sol
- LibVote.sol
- LibVoteResult.sol
- MarketCreatorWhitelist.sol
- OutcomeReporterWhitelist.sol
- SuperAdminRole.sol
- SystemParamsWhitelist.sol
- Whitelist.sol
- DummyERC223.sol
- TUSD.sol
- DetailedToken.sol
- AssertOrderValidator.sol
- BaseOrderValidator.sol
- CancelOrder.sol
- FeeSchedule.sol
- FillOrder.sol
- Fills.sol
- ReadOnlyOrderValidator.sol
- TokenTransferProxy.sol

Notes

The audit was based on the commit hash `6894de7`, solidity compiler `0.4.24+commit.e67f0147`

Issues Found

Critical

1. Committee is open to trustless collusion

Multiple owners can deposit tokens into a smart contract with the goal of influencing the committee vote result. It's possible to know beforehand if the group will have enough weight to influence voting to their desired output, making it risk-free to do so.

Additional note on committee structure: considering that members are already the biggest holders, that they get a proportional share of their weight from the `transactionFeePool` and that there is no mechanism for a forced rotation of members the committee can become sort of cartel that concentrate large amount of tokens over time. This can increase their ability to manipulate votes(since it's a weighted vote) in the long term.

Recommendation
Consider adding mechanisms for protecting against whale manipulation on the committee. This can be another layer of dispute, an additional "one member one vote round" or a combination of those.

Follow up [04.02.2019]
The BetX team implemented a time-lock that locks all committee member stakes for two weeks after optOut, and will also mitigate the aforementioned risks by distributing the tokens in ways that avoid capital concentration. Although these are steps on the right direction, we still recommend the implementation of other safeguards against committee collusion, through the ability of market participants to challenge the result of a dispute, and possibly implementing staking for reporters (and therefore enabling penalties for wrongly reported outcomes). We agree with BetX in that the risk of collusion is low in early stages of the platform, but the impact of a wrong committee to BetX's reputation would be significant.

Major

2. Committee member can deny some actions

Since the native token is an ERC223, members can implement malicious contracts to make the `tokenFallback` function always fail, denying some actions being taken, like being displaced from committee.

Recommendation
Consider adding extra methods that forcefully remove members and allow them to withdraw their stake afterwards.

Amended [04.02.2019]
The issue has been fixed and is no longer present in commit `340180326afe28ce7a7da112aa445151e80c1b4e`.

3. Problems with premature reveal and stealStake mechanism

If a member reveal it's vote prematurely he can rush to submit a new hash or he can make a transaction to steal his own deposit effectively not being punished, since he will receive his deposit back.

The `stealStake` function is also vulnerable to frontrunning by anyone to get free tokens.

Recommendation
Consider adding additional checks to that function, like restricting the use of stealStake to market participants (or committee members?).

Amended [04.02.2019]
The issue has been fixed and is no longer present in commit `340180326afe28ce7a7da112aa445151e80c1b4e`.

Minor

4. Outcome can be reported more than once

The function to report does not check if market has already been reported. It's also possible to change the reported outcome after a challenge deposit has already been made, making it count against the user desired outcome.

An example:

1) For a specific market, an outcome 1 is reported
2) Some user, disagreeing with the result submits a deposit above the threshold to challenge it to change to outcome 2.
3) In the meantime, the outcome is reported again, this time changing it to 2. Now, the user deposit is being counted towards challenge the latest outcome, which is not it was initially intended.

Recommendation
We recommend including a `require()` statement checking if the outcome has been previously reported, and not allowing it to be changed.

Amended [04.02.2019]
The issue has been fixed and is no longer present in commit `340180326afe28ce7a7da112aa445151e80c1b4e`.

5. Native Token safeERC20 implementation is incomplete

`Native.sol` inherits from `ERC223StandardToken`, in it only part of the `safeERC20` functions (increase/decrease allowance) is implemented (`increaseAllowance`). This results in transactions reducing the allowance still being vulnerable to frontrunning.

Recommendation
We recommend `decreaseAllowance` is also implemented.

Amended [04.02.2019]
The issue has been fixed and is no longer present in commit `340180326afe28ce7a7da112aa445151e80c1b4e`.

6. SuperAdminRole implementation do not allow for revoking access

`SuperAdminRole.sol` is used to control the keys that are allowed to manage access to the other profiles used throughout the smart contracts (`SystemParams`, `MarketCreator`, `OutcomeReporter`). It implements functions to add and to renounce ownership, not implementing, however, a function for revoking access rights. Revoking the access from a private key would be necessary in case a key is lost or compromised.

Additionally, there's no prevention regarding admin renouncing roles, so it's possible that at a given time no admin is left, making the contract unusable.

Recommendation
We recommend that a function for revoking `SuperAdmin` access is created.

Amended [04.02.2019]
The issue has been fixed and is no longer present in commit `340180326afe28ce7a7da112aa445151e80c1b4e`.

7. Markets can get frozen

There are a few scenarios where there's no planned course of action.
a. No vote is given to a schedule vote
b. No vote is revealed

While it seems unlikely, such scenarios can happen in situations where the network is clogged with transactions and gas prices are too high.

Recommendation
Consider implementing administrative functions that allow getting out of locked state

Amended [04.02.2019]
The issue has been fixed and is no longer present in commit `340180326afe28ce7a7da112aa445151e80c1b4e`.

Notes

8. `AssertOrderValidator` does not implement all the specified checks

`AssertOrderValidator.sol` checks the validity of an order, and reverts for invalid orders. Some of the specified checks were not included in `AssertOrderValidator`, such as total bet size different than zero, expiry greater than now, `percentageOdds` different than zero. These checks are however performed in other functions, hence the classification of this issue as a note.

Recommendation
We recommend carefully reviewing the contract specification and implementing the missing checks.

Amended [04.02.2019]
The issue has been fixed and is no longer present in commit `340180326afe28ce7a7da112aa445151e80c1b4e`.

9. Unsafe array length manipulation

In multiple instances (`DisputeManager.sol#L304,325` and `BaseTokenRegistry.sol#L51`), the length of the dynamic array is changed directly. Improperly handled, this can lead to a storage overlap attack.

Recommendation
Use `uint[] storage arrayName = new uint[](7)` to create a dynamic array of the desired length.
Use delete `arrayName` to clear a dynamic array.
Use `.push()` (instead of `.length++`) to write to the end of the dynamic array.

Amended [04.02.2019]
The issue has been fixed and is no longer present in commit `340180326afe28ce7a7da112aa445151e80c1b4e`.

10. Import interfaces instead of full contracts

When importing contracts, making use of interfaces greatly reduces the gas necessary to make a deployment, since they compile to a much smaller bytecode, while still being able to use the functions that importing multiple contracts would allow for.

NextGen's response:
"We will be implementing this later but decided to hold off for now for time reasons as this is simply a gas savings on deployment."

11. Numerous unbounded loops

A number of unbounded loops exist within the contracts (committee member total, user order for a given market), giving room for concern. If `array.length` is large enough, the function will exceed the block gas limit, and transactions calling it will revert. If an external actor were to influence the loop, it could potentially cause these issues (which may have downstream impact).

Recommendation
Avoid loops with big or unknown number of steps. If possible, bound loops where appropriate by supplying a known termination point.

NextGen's response:
"We carefully examined large loops on the critical path of the contracts and found roughly their upper bounds on the number of iterations based on targeted system parameters for launch:

- `removeToken()` on `BaseTokenRegistry.so`l at it's worst case (the token to remove is in the first slot in the array) can run through over 1000 iterations before running into the block limit.
- `committeeTransition()` on `CommitteeManager.sol` looks like it has a cap of around ~250 iterations before running into the block limit. For the initial committee size of 30, this means that it can handle around 7 base tokens.
- `updateLowestBalance()` on `OptInList.sol` can run through over 1000 iterations before running into the block limit. Similar complexity as removeToken.
- `resolveVotesForMember()` on `DisputeManger.sol` uses 2.8m gas with 150 scheduled votes.
- `resetVotes()` uses the most gas when there is `deferredVotes`# = `maxScheduledVotes` and `scheduledVotes`# = `maxScheduledVotes`. With a `maxScheduledVotes` of 30 and in this worst-case scenario, this function uses 1.6m gas."

Conclusion

Seven issues were found during the audit which could break the intended behavior, along with several informational notes. It is recommended for the NextGen team to address the issues. It is furthermore recommended to post the contracts on public bounty following the audit.

Disclaimer