Audit Report for 2Key on October 14th, 2019.

Summary

Audit Report prepared by Solidified for 2Key covering referral smart contracts (and inherited dependencies).

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The debrief took place on October 14th, 2019, and the final results are presented here.

Audited Files

The following contracts were covered during the audit:

```
2key/
├── acquisition-campaign-contracts
│   ├── TwoKeyAcquisitionCampaignERC20.sol
│   ├── TwoKeyAcquisitionLogicHandler.sol
│   ├── TwoKeyConversionHandler.sol
│   └── TwoKeyPurchasesHandler.sol
├── campaign-mutual-contracts
│   ├── ArcERC20.sol
│   ├── TwoKeyCampaign.sol
│   ├── TwoKeyCampaignConversionHandler.sol
│   ├── TwoKeyCampaignIncentiveModels.sol
│   └── TwoKeyCampaignLogicHandler.sol
├── donation-campaign-contracts
│   ├── InvoiceTokenERC20.sol
│   ├── TwoKeyDonationCampaign.sol
│   ├── TwoKeyDonationCampaignType.sol
│   ├── TwoKeyDonationConversionHandler.sol
│   └── TwoKeyDonationLogicHandler.sol
├── libraries
│   ├── Call.sol
│   ├── GetCode.sol
│   ├── IncentiveModels.sol
│   ├── SafeMath.sol
│   └── Utils.sol
├── non-upgradable-singletons
```

```
│   ├── ITwoKeySingletonUtils.sol
│   ├── TwoKeyCongress.sol
│   ├── TwoKeyEconomy.sol
│   ├── TwoKeyPlasmaSingletoneRegistry.sol
│   └── TwoKeySingletonesRegistry.sol
├── singleton-contracts
│   ├── TwoKeyAdmin.sol
│   ├── TwoKeyBaseReputationRegistry.sol
│   ├── TwoKeyCampaignValidator.sol
│   ├── TwoKeyEventSource.sol
│   ├── TwoKeyExchangeRateContract.sol
│   ├── TwoKeyFactory.sol
│   ├── TwoKeyMaintainersRegistry.sol
│   ├── TwoKeyPlasmaEvents.sol
│   ├── TwoKeyPlasmaMaintainersRegistry.sol
│   ├── TwoKeyPlasmaRegistry.sol
│   ├── TwoKeyRegistry.sol
│   ├── TwoKeySignatureValidator.sol
│   └── TwoKeyUpgradableExchange.sol
├── singleton-storage-contracts
    ├── TwoKeyAdminStorage.sol
    ├── TwoKeyBaseReputationRegistryStorage.sol
    ├── TwoKeyCampaignValidatorStorage.sol
    ├── TwoKeyCommunityTokenPoolStorage.sol
    ├── TwoKeyDeepFreezeTokenPoolStorage.sol
    ├── TwoKeyEventSourceStorage.sol
    ├── TwoKeyExchangeRateStorage.sol
    ├── TwoKeyFactoryStorage.sol
    ├── TwoKeyLongTermTokenPoolStorage.sol
    ├── TwoKeyMaintainersRegistryStorage.sol
    ├── TwoKeyPlasmaEventsStorage.sol
    ├── TwoKeyPlasmaMaintainersRegistryStorage.sol
    ├── TwoKeyPlasmaRegistryStorage.sol
    ├── TwoKeyRegistryStorage.sol
    ├── TwoKeySignatureValidatorStorage.sol
    └── TwoKeyUpgradableExchangeStorage.sol
```

SOLIDIFIED

Audit Report for 2Key on October 14th, 2019.

Notes

The audit was performed on commit `de171eb80e9ce1c6566a5d782f0a0549ab0151ad` of repository https://github.com/2key/contracts. Follow ups for fixes were made up to commit `dea12aa44e45bba60a59043987c2dad29fe25732`.

Intended Behavior

The contracts implement complete link referral incentive scheme, rewarding users for link sharing as part of viral marketing. The contracts include a nested signature scheme, a plasma implementation, upgradability and governance voting scheme.

Issues Found

Critical

1. Transactions may hit block gas limit when referral chain becomes too long

Transactions may always fail because of hitting the block gas limit when the referrers list becomes too long, as the block gas limit may be hit. For example in `TwoKeyBaseReputationRegistry.sol`: `updateOnConversionExecutedEvent()`

Recommendation
Avoid looping over variable size arrays or loop in pre-defined manageable blocks in several transactions.

Instances of unbounded loops
TwoKeyAcquisitionCampaignERC20, function distributeArcsBasedOnSignature
TwoKeyCampaignLogicHandler, function updateRefchainRewards
TwoKeyDonationCampaign, function distributeArcsBasedOnSignature
TwoKeyPlasmaEvents, function visited

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `c03ec07dd170d45447747de3ac55c7551c35b6b8`.

2. Error in vote counting

The voting system implementation in `TwoKeyCongress.sol` is incorrect. In the current implementation, votes are added when in favor of proposal and subtracted when voting against. However, the proposal executing code checks for > 50 % majority compared to the total voting power.

Example: total voting power: 10, votes in favor 7, votes against 3, result = 7 – 3 = 4. In this case the proposal would fail despite majority.

Recommendation
Count votes against a proposal as 0.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

Major

3. Discrepant values between converted amount and msg.value

On the `function ConvertFiat`, of `TwoKeyAquisitionCampaingERC20`, all sanity checks are performed on `msg.value`, but conversion is created using `conversionAmountFiatWe`i.

Recommendation:
Make sure that msg.value is equivalent with conversionAmountFiatWei

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

4. Functionality to Remove Member is unusable

The logic of the RemoveMember function (`TwoKeyCongress.sol`) checks for an out of bounds position in the allMembers array, making every transaction to it fail. Also note that the function is internal, and there is no external/public function making a call to it, effectively making it inaccessible once deployed.

Recommendation
Stop the loop on length -1.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`. The second loop could be avoided, by copying the last item of the array into the place of the deleted member and then subtracting 1 from the array length.

5. Congress does not allow for adding members after deployment

Congress does not allow for the inclusion of new members after deployment. For the congress to function over a long period of time it's implied that members will change, consider adding logic to allow for new members to be voted in.

Follow up [18.11.19]
The issue was fixed and is no longer present in commit `dea12aa44e45bba60a59043987c2dad29fe25732`.

6. Restriction on methods called by congress is ineffective

The existence of the function `getAllowedMethods`, and the state variable `allowedMethodSignatures` imply that congress should be restricted in its actions to a predetermined set of method calls.

The mentioned state variable is never set within the contract, nor verified before external calls.

Recommendation
Consider reviewing the restriction on calls, setting up a list of allowed calls initially, and allowing it to change only through voting.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

7. Maintainers can bypass congress upgrades

The function `createProxy` allows for mantainers to redeploy proxy contracts and substitute the existing version, effectively bypassing Congress on upgradability.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `c03ec07dd170d45447747de3ac55c7551c35b6b8`.

Minor

8. Missing Enforcement of Invocation Restriction in TwoKeyBaseReputationRegistry.sol

The methods `updateOnConversionExecutedEvent()` and `updateOnConversionRejectedEvent()` can only be called from the `TwoKeyConversionHandler` contract according to the documentation. However, there is nothing in the code to enforce this.
This issue might be due to a commenting issue, as `validateCall()` performs checks on validated campaigns.

Recommendation
Enforce call restriction or update commenting if applicable.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

9. The initial token distribution does not match the events created.

The token distribution implemented `TwoKeyEconomy.sol` does not match the event created during token deployment.

Recommendation
Change events to match distribution or vice versa.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

10. Signature Verification does not Check for Malleable Addresses

The signature verification does not check for malleable addresses which are still allowed by `ecrecover`.

Recommendation
We recommend adding checks for malleable addresses. Here is an example of how to check for this:
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/cryptography/ECDSA.sol

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `c03ec07dd170d45447747de3ac55c7551c35b6b8`.

11. replaceMember function can lead to inconsistent state

If the caller passes his own address to the function he is effectively removed from the list. Passing an existing address will also throw the allMembers array in inconsistent state.

Recommendation
Validate the address passed to the function against the list of existing members.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

12. Error in function signature

`report2KEYWithdrawnFromNetworkInternal` is not declared as internal, and can therefore be called by anyone, bypassing the control in `report2KEYWithdrawnFromNetwork` (`onlyValidatedContracts`).

Recommendation
Adjust the function visibility modifier.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

13. Lack of input validation on campaign creation

Bothe the `TwoKeyFactory` and the campaigns contract do not perform any kind of validation on the given parameters, making it possible to create unusable campaigns.

Recommendation
It's recommended to do some basic checks checks, for example that the referral percent is lower than 100%.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

14. Unbounded loops can block execution

If a function performs a loop on an unbounded array it can grow larger than the block gas limit, blocking execution. This may happen in:

`numberOfContractsCurrently` in `TwoKeyUpgradableExchange.sol`
`numberOfMaintainersTotal` in `TwoKeyMantainersRegistry.sol`
`stateToConverter` in `TwoKeyConversionHandler.sol`

Recommendation
Check on limits and try to bound these loops to values that will fit within the gas limit. If any of the loops have to be unbounded, consider including logic allowing it to stop mid processing and being called again in a subsequent transaction.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

15. Contracts can be upgraded to non-existing versions

`upgradeContract` does not check whether the version passed as an argument returns an actual address, making possible to set the implementation address to `0x0`.

Recommendation
Check the input on contract name and version.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

16. Possible clash of addresses in twoKeyEconomy

`_twoKeyAdmin` is passed as a parameter, but the state variable is set with the `twoKeyAdmin` present in the singletons registry. The token balance is given to the one passed as a parameter, allowing for the balance to be given to anyone.

The admin can also change himself by granting this right to other address, possibly once again making the admins inconsistent with the ones on the registry.

Recommendation
Define and use a single source of truth for administrative addresses.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

17. Exchanges rates are a point of centralization

The defined exchange rate can heavily influence the live campaigns and being set by the maintainers it can be seen as a point of centralization in the project.

Recommendation
Consider having it voted by congress and stored in one of the registries, and then consumed by the campaigns.

Follow up [01.11.19]
Auditee's response: 2key will use use Band-Protocol in order to decentralize the exchange rates registry.

18. ArcERC20 and InvoiceTokenERC20 are not compliant to the ERC20 Standard

`ArcERC20` does not implement a few of the required functions in the standard, which is necessary to be compatible.

`InvoiceToken` reverts on approve and transferFrom functions.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

19. Influencers can set their own cut (percentage of the bounty to be received)

In `TwoKeyAcquisitionCampaignERC20.sol`, the influencers can set their own cut (or percentage of the referrals they will receive). This allows the influencer to override the system cuts.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

21. startCampaignWithInitialParams can be called multiple times

`startCampaignWithInitialParams` in `TwoKeyCampaign.sol` can be called multiple times, allowing the contractor to change campaign data, as well as his own.

Follow up [01.11.19]
2key informed us that this is by design. We confirmed that there are no negative effects in the smart contracts from changing campaign public and private hashes during execution, though effects beyond that, in other parts of the application, were not assessed.

Notes

22. Add Error Messages to require Statements

Currently all `require()` statements revert without specifying an error. However, it is considered good practise to always supply a string indicating the reason for failure.

Recommendation
Add error messages.

23. No events emitted

In `TwoKeyCampaignLogicHandler.sol`, functions used to set minimum and maximum contributions are not emitting events, as indicated by the documentation.
Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`. (Documentation updated)

24. Unnecessary variable

`TwoKeyBaseReputationRegistry.sol`: The variable `d` in method `updateOnConversionExecutedEvent()` is unnecessary, as it just used in the loop and equivalent to the `index i+1`.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

25. Modifiers are implemented as normal function

Throughout the code base restrictions are implemented as functions returning boolean values and used in `require()` statements. Most of these could be implemented as modifiers making the code much cleaner.

Recommendation
Convert functions used as modifiers into modifiers.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

26. Potential loop optimization in TwoKeyCongress.sol

The loop in line 114 can be terminated with a break statement to safe gas once the member to be replaced has been found.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

27. Event emitted with wrong value

The event Transfer to `twoKeyUpgradableExchange` emitted from `TwoKeyEconomy` is passing the wrong value.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

28. Access control on view function is useless

The function `CallgetReferrerBalanceAndTotalEarningsAndNumberOfConversions` is protected to be called only by mantainers and the referrer itself. But this isn't very effective both because all data is already available on chain and that signatures can be replayed.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

29. Anonymous conversions are still accessible

Flagging a transaction as anonymous does not affect the data stored on chain, making it effectively public.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit `bd60f48e945b586a5cb8b72584a7008f56470384`.

30. Transaction bytecode is stored on chain

`transactionBytecode` is stored in proposal unnecessarily, storing only the hash of the transaction, and then submitting it in the execution phase will result in considerable gas savings.

31. Delete unused parameters

The flag param on the function `getSuperStatistics` is unused and could be safely deleted(saving gas on the execution).

Follow up [01.11.19]
The issue was fixed and is no longer present in commit
`bd60f48e945b586a5cb8b72584a7008f56470384`.

32. Move literal strings to constants

Instead of using literal strings it is a better practice to use constants to avoid spelling mistakes now and in future developments to make wrong storage access less possible.

Follow up [01.11.19]
The issue was fixed and is no longer present in commit
`bd60f48e945b586a5cb8b72584a7008f56470384`.

33. Consider updating the compiler

The contracts still rely on Solidity 0.4. We recommend updating it to the latest versions, as they not only prevent some vulnerabilities (unassigned storage pointers, function missing visibility modifiers), but also correct all the known bugs up to now (https://solidity.readthedocs.io/en/latest/bugs.html).

SOLIDIFIED

Audit Report for 2Key on October 14th, 2019.

Closing Summary

Several issues were encountered, including a number of critical and major severity bugs. We recommend 2key reviews not only the issues, but also take a holistic review of the codebase.

There is heavy duplication of code in campaigns, also in some singletons. consider a modular approach, avoiding duplication (and the inevitable situation when two functions with the same name/signature start to behave differently because of inconsistencies, this already happens in smaller functions/modifiers such as onlyMaintainer()).

The complexity of the codebase is high, including several upgradable modules and an accompanying proxying scheme, as well as contracts interconnected in numerous ways creating a quasi-infinite number of combinations. There is a residual risk, inherent to complex designs, that is partly mitigated by 2key's ability to upgrade parts of the system.

Disclaimer