

November 6, 2019

# Audit Report

2key network smart contracts allowing Social Sourcing -  
Incentivised Activation of Result-Driven online virality.

*AUTHOR: DAN BOGDANOV – CALLISTO.NETWORK*

## ● TABLE OF CONTENTS

1. DISCLAIMER	3
2. INTRODUCTION	4
2.1 Audit request	4
2.2 In scope	4
3. METHODOLOGY	8
3.1 My review methodology	8
3.2 Classification of Detected Issues	9
4. SUMMARY OF FINDINGS	10
5. AUDIT FINDING	11
5.1 High	11
5.2 Medium	14
5.3 Low	20
5.4 Notes	24
5.5 Owner privileges	28
6. CONCLUSION	30

## ● 1. DISCLAIMER

This audit report presents the findings of a security review of the smart contracts under scope of the audit. The audit does not give any *warranties on the security of the code*. Using specially designed tools for debugging and using automated tests to verify minor changes and fixes.

I always recommend proceeding to several independent audits and a public bug bounty program to ensure the security of the smart contracts.

## ● 2. INTRODUCTION

### ✓ 2.1 Audit request

Smart contracts allowing Social Sourcing - Incentivised Activation of Result-Driven online virality. More can be found on <https://www.2key.network/> (litepaper, whitepaper etc.)

### ✓ 2.2 In scope

This document is a security audit report performed by danbogd <https://github.com/danbogd>, where 2key network <https://github.com/2key/contracts/tree/develop/contracts/2key> has been reviewed.

Commit hash **c85f3e1f3a04c56afd28bf673758367cc3df6609**.

- [TwoKeyAcquisitionCampaignERC20.sol](#).
- [TwoKeyAcquisitionLogicHandler.sol](#).
- [TwoKeyConversionHandler.sol](#).
- [TwoKeyPurchasesHandler.sol](#).
- [ArcERC20.sol](#).
- [TwoKeyCampaign.sol](#).
- [TwoKeyCampaignIncentiveModels.sol](#).
- [ERC20.sol](#).
- [InvoiceTokenERC20.sol](#).
- [TwoKeyDonationCampaign.sol](#).
- [TwoKeyDonationCampaignType.sol](#).
- [TwoKeyDonationConversionHandler.sol](#).
- [TwoKeyDonationLogicHandler.sol](#).
- [ITwoKeyAdminStorage.sol](#).
- [ITwoKeyBaseReputationRegistryStorage.sol](#).
- [ITwoKeyCampaignValidatorStorage.sol](#).
- [ITwoKeyCommunityTokenPoolStorage.sol](#).
- [ITwoKeyDeepFreezeTokenPoolStorage.sol](#).
- [ITwoKeyEventSourceStorage.sol](#).
- [ITwoKeyExchangeRateContractStorage.sol](#).
- [ITwoKeyFactoryStorage.sol](#).

- [ITwoKeyLongTermTokenPoolStorage.sol](#)).
- [ITwoKeyMaintainersRegistryStorage.sol](#).
- [ITwoKeyPlasmaEventsStorage.sol](#).
- [ITwoKeyPlasmaMaintainersRegistryStorage.sol](#).
- [ITwoKeyPlasmaRegistryStorage.sol](#).
- [ITwoKeyRegistryStorage.sol](#).
- [ITwoKeySignatureValidatorStorage.sol](#).
- [ITwoKeyUpgradableExchangeStorage.sol](#).
- [IDecentralizedNation.sol](#).
- [IGetImplementation.sol](#).
- [IERC20.sol](#).
- [IHandleCampaignDeployment.sol](#).
- [IKyberNetworkProxy.sol](#).
- [IMaintainingPattern.sol](#).
- [IStructuredStorage.sol](#).
- [ITwoKeyAcquisitionARC.sol](#).
- [ITwoKeyAcquisitionCampaignERC20.sol](#).
- [ITwoKeyAcquisitionCampaignStateVariables.sol](#).
- [ITwoKeyAcquisitionLogicHandler.sol](#).
- [ITwoKeyAdmin.sol](#).
- [ITwoKeyBaseReputationRegistry.sol](#).
- [ITwoKeyCampaignGetReferrers.sol](#).
- [ITwoKeyCampaignPublicAddresses.sol](#).
- [ITwoKeyCampaignValidator.sol](#).
- [ITwoKeyConversionHandler.sol](#).
- [ITwoKeyConversionHandlerGetConverterState.sol](#).
- [ITwoKeyDonationCampaign.sol](#).
- [ITwoKeyDonationCampaignFetchAddresses.sol](#).
- [ITwoKeyDonationConversionHandler.sol](#).
- [ITwoKeyDonationLogicHandler.sol](#).
- [ITwoKeyEventSource.sol](#).

- [ITwoKeyEventSourceEvents.sol](#).
- [ITwoKeyExchangeRateContract.sol](#).
- [ITwoKeyMaintainersRegistry.sol](#).
- [ITwoKeyPlasmaEvents.sol](#).
- [ITwoKeyPlasmaRegistry.sol](#).
- [ITwoKeyPurchasesHandler.sol](#).
- [ITwoKeyReg.sol](#).
- [ITwoKeyRegistry.sol](#).
- [ITwoKeyRegistryEvents.sol](#).
- [ITwoKeySingletonAddressStorage.sol](#).
- [ITwoKeySingletonRegistryFetchAddress.sol](#).
- [ITwoKeySingletonesRegistry.sol](#).
- [ITwoKeyWeightedVoteContract.sol](#).
- [IUpgradableExchange.sol](#).
- [Call.sol](#).
- [GetCode.sol](#).
- [IncentiveModels.sol](#).
- [SafeERC20.sol](#).
- [SafeMath.sol](#).
- [Utils.sol](#).
- [ITwoKeySingletonUtils.sol](#).
- [StandardTokenModified.sol](#).
- [TwoKeyAdmin.sol](#).
- [TwoKeyBaseReputationRegistry.sol](#).
- [TwoKeyCampaignValidator.sol](#).
- [TwoKeyCongress.sol](#).
- [TwoKeyEconomy.sol](#).
- [TwoKeyEventSource.sol](#).
- [TwoKeyExchangeRateContract.sol](#).
- [TwoKeyFactory.sol](#).
- [TwoKeyLockupContract.sol](#).

- [TwoKeyMaintainersRegistry.sol](#).
- [TwoKeyPlasmaEvents.sol](#).
- [TwoKeyPlasmaMaintainersRegistry.sol](#).
- [TwoKeyPlasmaRegistry.sol](#).
- [TwoKeyPlasmaSingletonRegistry.sol](#).
- [TwoKeyRegistry.sol](#).
- [TwoKeySignatureValidator.sol](#).
- [TwoKeySingletonsRegistry.sol](#).
- [TwoKeyUpgradableExchange.sol](#).
- [TokenPool.sol](#).
- [TwoKeyCommunityTokenPool.sol](#).
- [TwoKeyDeepFreezeTokenPool.sol](#).
- [TwoKeyLongTermTokenPool.sol](#).
- [Proxy.sol](#).
- [StructuredStorage.sol](#).
- [UpgradabilityStorage.sol](#).
- [UpgradeabilityProxy.sol](#).
- [Upgradeable.sol](#).
- [ProxyCampaign.sol](#).
- [UpgradabilityCampaignStorage.sol](#).
- [UpgradeableCampaign.sol](#).
- [DecentralizedNation.sol](#).
- [ERC20CustomToken.sol](#).
- [ERC20TokenMock.sol](#).
- [Ownable.sol](#).
- [TwoKeyAirdropCampaign.sol](#).
- [TwoKeyConversionStates.sol](#).
- [TwoKeyTypes.sol](#).
- [TwoKeyVoteToken.sol](#).
- [UpgradabilityProxyAcquisition.sol](#).

### 🟢 3. METHODOLOGY

#### ✔ 3.1 My review methodology

My smart contract review methodology involves automated and manual audit techniques. The applications are subjected to a round of dynamic analysis using tools like linters, program profilers and source code security scanners (publicly available automated Solidity analysis tools such as [Remix](#), [Oyente](#), Solidity static code analyzer [SmartCheck](#)). The contracts have their source code manually inspected for security vulnerabilities. This type of analysis has the ability to detect issues that are missed by automated scanners and static analyzers, as it can discover edge-cases and business logic-related problems. Special attention is directed towards the ability of an owner to manipulate contract.



### ✓ 3.2 Classification of detected issues

**High** - vulnerability can be exploited at any time and cause a loss of customers funds or a complete breach of contract operability. (Example: Parity Multisig hack, a user has exploited a vulnerability and violated the operability of the whole system of smart-contracts (Parity Multisigs). This could performed regardless of external conditions at any time.)

**Medium** - vulnerability can be exploited in some specific circumstances and cause a loss of customers funds or a breach of operability of smart-contract (or smart-contract system). (Example: ERC20 bug, a user can exploit a bug (or "undocumented opportunity") of transfer function and occasionally burn his tokens. A user can not violate someone else's funds or cause a complete breach of the whole contract operability. However, this leads to millions of dollars losses for Ethereum ecosystem and token developers.)

**Low** - vulnerability can not cause a loss of customers funds or a breach of contracts operability. However it can cause any kind of problems or inconveniences. (Example: Permanent owners of multisig contracts, owners are permanent, thus if it will be necessary to remove a misbehaving "owner" from the owners list then it will require to redeploy the whole contract and transfer funds to a new one.)

**Owner privileges** - the ability of an owner to manipulate contract, may be risky for investors.

**Note** - other code flaws, not security-related issues.


*The severity is calculated according to the OWASP risk rating model based on Impact and Likelihood:*


Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			


#### 4. SUMMARY OF FINDINGS

✓ In total, **30** issues were reported including:

✗ - 5 high severity issues.

 - 9 medium severity issues.

 - 6 low severity issues.

 - 6 notes.

 - 4 owner privileges (the ability of an owner to manipulate contract, may be risky for investors).

## 🟢 5. AUDIT FINDING

### ✔ HIGH SEVERTY ISSUES

#### ✖ 5.1 Anyone can get access to onlyMember functions

##### Description

Anyone can call `replaceMemberAddress` function and there is no check of membership. Therefore, an attacker can add any address to `isMemberInCongress` array and get access to all functions with `onlyMember` modifier.

##### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyCongress.sol#L187>

#### ✖ 5.2 There is no way to remove user

##### Description

Only the contract itself can call the `removeMember` function. But the contract does not have this call, respectively, there is no way to remove the user.

The `removeMember` function contains this code:

```
for (uint j = i; j < allMembers.length; j++){
    allMembers[j] = allMembers[j+1];
}
```

But the last assignment will not be performed. Because if the array consists of 5 elements then `allMembers[4] = allMembers[5]` expression attempts to access a nonexistent element(last index is 4). And transaction will be reverted.

*Note: Also access to nonexistent element is performed [here](#).*

##### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyCongress.sol#L256>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyCongress.sol#L273>

### ✖ 5.3 Anyone can transfer others tokens

#### Description

Using function `transferFrom` anyone can transfer other addresses voting points to himself. There is no checking for person who call this function, so he can pass his address as `to` address and transfer himself anyone's voting points.

#### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/TwoKeyVoteToken.sol#L109>

#### Recommendation

Better to use `transfer` function so only voting points' owners will be able to transfer their voting points.

### ✖ 5.4 Failure to delete a user is possible

#### Description

It may happen that the user cannot be removed in case if block gas limit is reached. When removed, two cycles are used and when a large number of users will not have enough gas to perform the function. And considering vulnerability 5.18. an attacker can intentionally spam an array of users.

```
while(allMembers[i] != targetMember) {
    if(i == allMembers.length) {
        revert();
    }
    i++;
}
//After member is found, remove his address from all members
for (uint j = i; j < allMembers.length; j++){
    allMembers[j] = allMembers[j+1];
}
```

#### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyCongress.sol#L265-L274>

## ✖ 5.5 Contract Access Following Code Hash

### Description

`GetCode` library function `at` access the code of another contract and load it into a bytes variable, this function is used in `TwoKeyVoteToken` to load contract code and calculate a hash that will be used to allow the contract or not to transfer tokens following the developers comments (check the code below).

However when the code is loaded no state variable that has changed after the deployment is taken into account meaning that an attacker can load the code of a pre approved contract, deploy a new contract with the admin address changed (an example can be if the contract set the admin address inside the constructor), this will allow the attacker to access the same privileges as the original allowed contract on `TwoKeyVoteToken` or any inherited contract.

Please note that no function is implemented to remove the privileges for a contract.

### Code snippet

<https://github.com/2key/contracts/blob/440b7016157e0f115294a38566d4f3d42a79dfed/contracts/2key/libraries/GetCode.sol#L10>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/TwoKeyVoteToken.sol#L37>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/TwoKeyVoteToken.sol#L46>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/TwoKeyVoteToken.sol#L54>

## ✔ MEDIUM SEVERTY ISSUES



### 5.6 Wrong value passing

#### Description

In function `setInitialParams` there is wrong value passing to function `setInitialParameters(_erc20Address, TWO_KEY_SINGLETON_REGISTRY);`, because in all other similar functions there is passing value from arguments of functions and also `TWO_KEY_SINGLETON_REGISTRY` variable not initialized in this contract.

```
function setInitialParams(
    address twoKeySingletonesRegistry,
    address _erc20Address,
    address _proxyStorage
)
external
{
    require(initialized == false);

    setInitialParameters(_erc20Address, TWO_KEY_SINGLETON_REGISTRY);

    PROXY_STORAGE_CONTRACT = ITwoKeyCommunityTokenPoolStorage(_proxyStorage);

    setUint("totalAmount2keys", 2000000000);
    setUint("annualTransferAmountLimit", 200000000);
    setUint("startingDate", block.timestamp);

    for(uint i=1; i<=10; i++) {
        bytes32 key1 = keccak256("yearToStartingDate", i);
        bytes32 key2 = keccak256("yearToTransferredThisYear", i);

        PROXY_STORAGE_CONTRACT.setUint(key1, block.timestamp + i*(1 years));
        PROXY_STORAGE_CONTRACT.setUint(key2, 0);
    }

    initialized = true;
}
```

#### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/token-pools/TwoKeyCommunityTokenPool.sol#L20>

#### Recommendation

There should be passed `twoKeySingletonesRegistry` instead of `TWO_KEY_SINGLETON_REGISTRY`.

```
setInitialParameters(_erc20Address, twoKeySingletonesRegistry);
```



## 5.7 Wrong membership address replacing

### Description

In function `replaceMemberAddress` there is setting old member info to new address in [line 199](#) , but `memberAddress` will still be old address, which should be replaced.

```
function replaceMemberAddress(
    address _newMemberAddress
)
public
{
    require(_newMemberAddress != address(0));
    // Update is member in congress state
    isMemberInCongress[_newMemberAddress] = true;
    isMemberInCongress[msg.sender] = false;

    //Update array containing all members addresses
    for(uint i=0; i<allMembers.length; i++) {
        if(allMembers[i] == msg.sender) {
            allMembers[i] = _newMemberAddress;
        }
    }

    //Update member object
    Member memory m = address2Member[msg.sender];
    address2Member[_newMemberAddress] = m;
    address2Member[msg.sender] = Member(
        {
            memberAddress: address(0),
            memberSince: block.timestamp,
            votingPower: 0,
            name: "0x0"
        }
    );
}
```

### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyCongress.sol#L180>

### Recommendation

Create new `Member` structure for replaced address with copying `memberSince`, `votingPower` and `name`.



## 5.8 Wrong condition checking

### Description

In the function `isCampaignEnded` there is possibility that `campaignRaisedAlready` will be more than `campaignHardCapWei` so in this case this function will return `false` instead of `true`.

```
function isCampaignEnded() internal view returns (bool) {
    if(checkIsCampaignActiveInTermsOfTime() == false) {
        return true;
    }
    if(endCampaignWhenHardCapReached == true && campaignRaisedAlready == campaignHardCapWei) {
        return true;
    }
    return false;
}
```

### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/acquisition-campaign-contracts/TwoKeyAcquisitionLogicHandler.sol#L280>

### Recommendation

Check that `campaignRaisedAlready` is at least equal to `campaignHardCapWei`.

```
if(endCampaignWhenHardCapReached == true && campaignRaisedAlready >= campaignHardCapWei) {
    return true;
}
```



## 5.9 Allowance function show wrong value

### Description

```
/**
 * @dev Function to check the amount of tokens that an owner allowed to a spender.
 * @param _owner address The address which owns the funds.
 * @return A uint256 specifying the amount of tokens still available for the spender.
 */
function allowance(
    address _owner,
    address _spender
)
public
view
returns (uint256)
{
    return balanceOf(_owner);
}
```

Function `allowance` show `balanceOf` functions' result and doesn't show the amount of tokens that an owner allowed to a spender as wrote in comment above function.



## Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/TwoKeyVoteToken.sol#L135>



## 5.10 Distribution date changing

### Description

In function [changeTokenDistributionDate](#) the [shift](#) value will be positive if [\\_newDate](#) less then [tokenDistributionDate](#) and negative otherwise. Therefore this value should be subtracted from [tokenUnlockingDate](#) instead of added.

Same issue occur in function [changeDistributionDate](#) in [TwoKeyPurchasesHandler.sol](#) contract

### Code snippet

```
uint shift = tokenDistributionDate - _newDate;
// If the date is changed shifting all tokens unlocking dates for the difference
for(uint i=0; i<numberOfVestingPortions+1;i++) {
    tokenUnlockingDate[i] = tokenUnlockingDate[i] + shift;
}
```

### Recommendation

Use subtraction shift instead of addition:

```
uint shift = tokenDistributionDate - _newDate;
// If the date is changed shifting all tokens unlocking dates for the difference
for(uint i=0; i<numberOfVestingPortions+1;i++) {
    tokenUnlockingDate[i] = tokenUnlockingDate[i] - shift;
}
```



## 5. 11 Underflow is possible

### Description

When removing a member, there is no safe mathematical operation [maxVotingPower-=votingPower](#). The following actions may cause an underflow:

1. [addMember](#) 0x1, [maxVotingPower](#) = 10
2. [removeMember](#) 0x1, [maxVotingPower](#) = 10
3. [replaceMemberAddress](#) 0x1 to 0x2, [maxVotingPower](#) = 10 (look at issue 5.1 where the removed user can restore his membership with his [votingPower](#))
4. [removeMember](#) 0x2, [maxVotingPower](#) = 0-10 (**underflow**)

And the same scheme underflow with [minimumQuorum](#) variable.

## Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyCongress.sol#L261>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyCongress.sol#L292>



## 5.12 Out of Gas

### Description

The contracts have loops the number of iterations of which depends on the variables in the storage and can increase over time. This can lead to "Out of Gas" error or to "Block Gas Limit". Need to create a mechanism for breaking cycles into parts.

*Example:*

```
function setVisitsListTimestamps(address _c, address _contractor, address _referrer) internal {
    uint[] memory visitListTimestamps = getVisitsListTimestamps(_c, _contractor, _referrer);
    uint[] memory newVisitListTimestamps = new uint[](visitListTimestamps.length + 1);
    for(uint i=0; i< visitListTimestamps.length; i++) {
        newVisitListTimestamps[i] = visitListTimestamps[i];
    }
    newVisitListTimestamps[visitListTimestamps.length] = block.timestamp;

    bytes32 keyHash = keccak256("visits_list_timestamps", _c, _contractor, _referrer);
    PROXY_STORAGE_CONTRACT.setUintArray(keyHash, newVisitListTimestamps);
}
```

## Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/donation-campaign-contracts/TwoKeyDonationConversionHandler.sol#L261-L27>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyPlasmaEvents.sol#L395-L397>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/acquisition-campaign-contracts/TwoKeyConversionHandler.sol#L368-L377>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyPlasmaEvents.sol#L251-L286>

<https://github.com/RideSolo/contracts-4/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyCongress.sol#L76>

## 5.13 Power Law (Rewards Distribution)

### Description

If a value of `factor` is higher than 2 in `powerLawRewards` function the cumulative reward for every address (`numberOfInfluencers`) will be higher than the total allocated reward `totalRewardEthWEI`.

### Code snippet

<https://github.com/RideSolo/contracts-4/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/libraries/IncentiveModels.sol#L48>

## 5.14 Maintainers Management

### Description

`getAllMaintainers` function implemented in of `TwoKeyMaintainersRegistry` and `TwoKeyPlasmaMaintainersRegistry` return an incomplete or a wrong array of maintainers addresses since it is only updated once when initialized but not updated when calling `addMaintainer` or `removeMaintainer` later on after initialization.

```
function getAllMaintainers()
public
view
returns (address[])
{
    return PROXY_STORAGE_CONTRACT.getAddressArray(keccak256("maintainers"));
}
```

```
function getAllMaintainers()
public
view
returns (address[])
{
    return PROXY_STORAGE_CONTRACT.getAddressArray(keccak256("maintainers"));
}
```

### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyMaintainersRegistry.sol#L104>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyPlasmaMaintainersRegistry.sol#L79>

✓ LOW SEVERTY ISSUES

● 5.15 No checking for zero address

## Description

Incoming addresses should be checked for an empty value(0x0 address) to avoid loss of funds or blocking some functionality.

## Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/TwoKeyAirdropCampaign.sol#L88>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/token-pools/TokenPool.sol#L24-L25>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/token-pools/TwoKeyCommunityTokenPool.sol#L21-L23>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/token-pools/TwoKeyCommunityTokenPool.sol#L52>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/donation-campaign-contracts/TwoKeyDonationCampaign.sol#L38>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/donation-campaign-contracts/TwoKeyDonationConversionHandler.sol#L82>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/acquisition-campaign-contracts/TwoKeyAcquisitionCampaignERC20.sol#L45>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyEventSource.sol#L134>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeySingletonesRegistry.sol#L55>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/upgradability/StructuredStorage.sol#L39>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/upgradability/UpgradabilityProxy.sol#L26>

## 5.16 ERC20 Compliance.

### **Description**

According to ERC20 standard, when initializing a token contract if any token value is set to any given address a transfer event should be emitted.

### **Code snippet**

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/ERC20CustomToken.sol#L12>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/ERC20TokenMock.sol#L12>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/acquisition-campaign-contracts/TwoKeyAcquisitionCampaignERC20.sol#L78>

## 5.17 Multi Transfer arrays length check.

### **Description**

There are several input arrays, but no check for the same length. In this case it is possible to skip some parameters.

### **Code snippet**

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyCongress.sol#L108-L109>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyExchangeRateContract.sol#L61>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/DecentralizedNation.sol#L117>

## 5.18 Not complete removal of a user from Congress

### Description

There are no restrictions for adding same user several times. In this case during the user removing only first record [will be removed from allMember array](#). The rest of the user records will be saved.

### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyCongress.sol#L239>

## 5.19 Known vulnerabilities of ERC-20 token

### Description

- It is possible to double withdrawal attack. More details [here](#)
- Lack of transaction handling mechanism issue. **WARNING!** This is a very common issue and it already caused millions of dollars losses for lots of token users! More details [here](#)

### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/donation-campaign-contracts/InvoiceTokenERC20.sol>

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyEconomy.sol>

### Recommendation

Add into a function `transfer(address _to, ... )` following code:

```
require( _to != address(this) );
```

## 5.20 Compare Function

### Description

compare function member of [TwoKeyCongress](#) compare only the first 3 bytes of the allowed function signature, if a not allowed function contain the same 3 first bytes the proposal can pass the validation phase.

```
function compare(
    bytes32 x,
    bytes y
)
public
pure
returns (bool)
{
    for(uint i=0;i<3;i++) {
        byte a = x[i];
        byte b = y[i];
        if(a != b) {
            return false;
        }
    }
    return true;
}
```

### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyCongress.sol#L143>

- Notes
- 5.21 Wrong transferFrom function

## Description

In function `transferFrom` there is adding `conversionQuota` to `_to` address balance, where `conversionQuota` is maximal ARC tokens that can be passed in `transferFrom`, but `value` will always be equal to 1. So in this function there will not be transferring tokens from one address to another, there will be burning from one and minting to another.

```
function transferFrom(
    address _from,
    address _to,
    uint256 _value
)
internal
returns (bool)
{
    // _from and _to are assumed to be already converted to plasma address (e.g. using plasmaOf)
    require(_value == 1);
    require(balances[_from] > 0);

    balances[_from] = balances[_from].sub(1);
    balances[_to] = balances[_to].add(conversionQuota);
    totalSupply_ = totalSupply_.add(conversionQuota.sub(1));

    if (received_from[_to] == 0) {
        twoKeyEventSource.joined(this, _from, _to);
    }

    received_from[_to] = _from;
    return true;
}
```

## Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/campaign-mutual-contracts/TwoKeyCampaign.sol#L59>



## 5.22 No checking for enough tokens

### Description

There is no checking that there will be enough tokens in [line 344](#) in function [buyTokensAndDistributeReferrerRewards](#) as wrote in **TODO** comment.

```
//TODO: add require that there's enough tokens at this moment
```

### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/acquisition-campaign-contracts/TwoKeyAcquisitionCampaignERC20.sol#L344>

### Recommendation

Add checking that there is enough tokens.

## 5.23 Empty if statement

### Description

**If** statement in [constructor](#) has no code inside of it and could be deleted.

```
if(inventoryAmount - maxNumberOfConversions*2*_numberOfTokensPerConverterAndReferralChain > 0 ) {  
  //TODO: This is sufficient balance which can't be used, and will be returned back to the contractor  
}
```

### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/TwoKeyAirdropCampaign.sol#L94>

## 5.24 Multiplication after division

### Description

Solidity operates only with integers. Thus, if the division is done before the multiplication, the rounding errors can increase dramatically.

```
totalBounty2keys = (_maxReferralRewardETHWei / (rate)) * (1000);
```

### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/acquisition-campaign-contracts/TwoKeyAcquisitionCampaignERC20.sol#L342>

## 5.25 Adding a member is possible only from constructor

### Description

The `addMember` function is used condition:

```
if(initialized == true) {  
    require(msg.sender == address(this));  
}
```

But after the initialization there is no way to use `addMember` because it is not being called from anywhere else in the contract.

### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyCongress.sol#L226-L228>

## 5.26 Unused contract

### Description

`TwoKeyVoteToken` contract is created but never used:

```
votingToken = new TwoKeyVoteToken(address(this));
```

### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/DecentralizedNation.sol#L115>

## ● SEVERTY OWNER PRIVILEGES (optional)

### Description

Contract owner allow himself to:

- upgrade contract and implement any logic in the new contract. And even if the new contract will be audited, at any time possible to change the address of the new contract again to not audited and insecure.

```
function upgradeTo(string _contractName, string _version, address _impl) public {  
    require(msg.sender == address(registry));  
    _implementation = _impl;  
}
```

### Code snippet

<https://github.com/2key/contracts/blob/295b13424a21ae9f52844a3a706af02aa4472fd4/contracts/2key/upgradability/UpgradeabilityProxy.sol#L26>

### Description

Contract owner allow himself to:

- to freeze all transfers on ERC for some period of time

```
function freezeTransfers()  
public  
onlyTwoKeyAdmin  
{  
    transfersFrozen = true;  
}
```

### Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyEconomy.sol#L58>

## Description

Contract owner allow himself to:

- can change any unit value in TwoKeyUpgradableExchange contract as buy and sell rates, weiRaised values, etc.

## Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyUpgradableExchange.sol#L400>

```
function updateUint(  
    string key,  
    uint value  
)  
public  
onlyMaintainer  
{  
    setUint(key, value);  
}
```

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/singleton-contracts/TwoKeyUpgradableExchange.sol>

## Description

Contract owner allow himself to:

- can change proxy logic contract any times and chan change it to not audited contract.

```
function setProxyLogicContract(address _proxyLogicContract) external onlyDeployer {  
    PROXY_LOGIC_CONTRACT = _proxyLogicContract;  
}
```

## Code snippet

<https://github.com/2key/contracts/blob/c85f3e1f3a04c56afd28bf673758367cc3df6609/contracts/2key/upgradability/StructuredStorage.sol#L48>

## 6. CONCLUSION

The audited smart contract **must not be deployed**. Reported issues must be fixed prior to the usage of this contract.