TRUFY

Smart Contract Audit Report

for

IVIRSE

Preliminary Comments

September, 2022

Contents

1 Introduction **3**
1.1 Project Summary . . . 3
1.2 Vulnerability Summary . . . 3

2 Findings **4**

3 Detailed Results **5**
3.1 ID-01: Suggested modification for project structure . . . 5
3.1.1 Description . . . 5
3.1.2 Recommendation . . . 5
3.2 ID-02: Lack of contract information in the error message of `require()` . . . 6
3.2.1 Description . . . 6
3.2.2 Recommendation . . . 6
3.3 ID-03: Bad modifier naming . . . 7
3.3.1 Description . . . 7
3.3.2 Recommendation . . . 7
3.4 ID-04: Bad gas optimization in function `_validateTimesAndAmounts()` . . . 8
3.4.1 Description . . . 8
3.4.2 Recommendation . . . 8
3.5 ID-05: Bad gas optimization in function `_validateAccountsAndAmounts()` 9
3.5.1 Description . . . 9
3.5.2 Recommendation . . . 9
3.6 ID-06: Inconsistency unit of parameter `amount` . . . 10
3.6.1 Description . . . 10
3.6.2 Recommendation . . . 10
3.7 ID-07: Bad naming convention . . . 11
3.7.1 Description . . . 11
3.7.2 Recommendation . . . 11
3.8 ID-08: Bad function naming . . . 12
3.8.1 Description . . . 12
3.9 ID-09: Bad gas optimization in function `_getUsedTokenByStatus()` and `_getAllowanceByStatus()` . . . 13
3.9.1 Description . . . 13
3.9.2 Recommendation . . . 13
3.10 ID-10: Logic issue of function `_createCampaign()` . . . 15
3.10.1 Description . . . 15
3.10.2 Recommendation . . . 15
3.11 ID-11: Bad modifier naming . . . 16
3.11.1 Description . . . 16
3.12 ID-12: Function `_getAllowanceByStatus()` calculate wrong `activeToken` . 17
3.12.1 Description . . . 17
3.13 ID-13: Unbalance issue of campaign creation and deletion . . . 18

3.13.1 Description 18
3.14 ID-14: Proper usage of `public` and `external` access 19
3.14.1 Description 19

4 Appendix **20**
4.1 Severity Definitions 20
4.2 Finding Categories 20

1 Introduction

Trufy (Consultant) was contracted by IVIRSE (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between September 7th, 2022 – September 14th, 2022.

1.1 Project Summary

- Project Name: IVIRSE
- Language: Solidity
- Codebase: https://github.com/IVIRSE/IVIRSE
- Commit: fc22999e16afe7ae1581f95f8c9b358bc0dcc773
- Audit method: Static Analysis, Manual Review
- Scope:
  - contracts/community/CampaignManagement.sol

1.2 Vulnerability Summary

Severity	# of Findings
Critical	0
Medium	1
Low	2
Informational	11

2 Findings

ID	Title	Type	Severity
ID-01	Suggested modification for project structure	Coding Style	Informational
ID-02	Lack of contract information in the error message of `require()`	Coding Style	Informational
ID-03	Bad modifier naming	Coding Style	Informational
ID-04	Bad gas optimization in function `_validateTimesAndAmounts()`	Gas Optimization	Informational
ID-05	Bad gas optimization in function `_validateAccountsAndAmounts()`	Gas Optimization	Informational
ID-06	Inconsistency unit of parameter `amount`	Inconsistency	Informational
ID-07	Bad naming convention	Coding Style	Informational
ID-08	Bad function naming	Coding Style	Informational
ID-09	Bad gas optimization in function `_getUsedTokenByStatus()` and `_getAllowanceByStatus()`	Gas Optimization	Informational
ID-10	Logic issue of function `_createCampaign()`	Logical Issue	Low
ID-11	Bad modifier naming	Coding Style	Informational
ID-12	Function `_getAllowanceByStatus()` calculate wrong `activeToken`	Logical Issue	Medium
ID-13	Unbalance issue of campaign creation and deletion	Logical Issue	Low
ID-14	Proper usage of `public` and `external` access	Gas optimization	Informational

3 Detailed Results

3.1 ID-01: Suggested modification for project structure

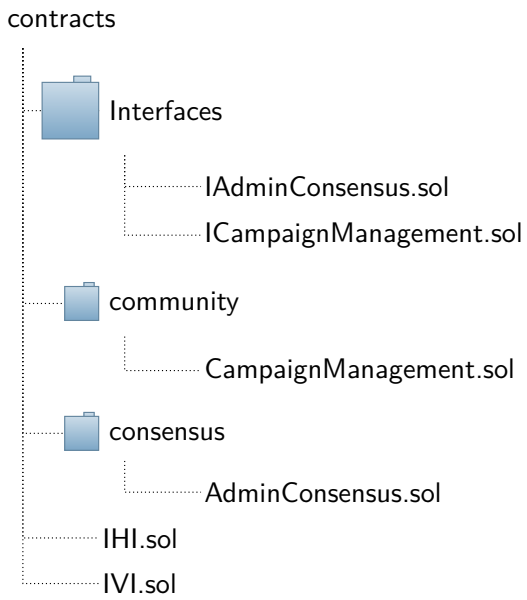
Type	Severity	Location
Coding Style	Informational	

3.1.1 Description

All interfaces should be located in one directory.

3.1.2 Recommendation

Recommended directory tree:



3.2 ID-02: Lack of contract information in the error message of `require()`

Type	Severity	Location
Coding Style	**Informational**	Lines 67, 81, 87, etc.

3.2.1 Description

It is better to include more details about the contract where the issue was thrown in the error message.

For example, in lines 67 – 69:

```
67 require(
68     totalCampaignConsensus > adminsLength.div(2),
69     "Not enough consensus!"
70 );
```

3.2.2 Recommendation

Adding contract name to the error message:

```
67 require(
68     totalCampaignConsensus > adminsLength.div(2),
69     "CampaignManagement: Not enough consensus!"
70 );
```

3.3 ID-03: Bad modifier naming

Type	Severity	Location
Coding Style	Informational	Line 102

3.3.1 Description

Modifier `isExist()` can be bypassed if `campaignName` has **not existed** yet and vice versa.

3.3.2 Recommendation

It is better to change name of modifier `isExist()` to `ifNotExist()`.

3.4 ID-04: Bad gas optimization in function `_validateTimesAndAmounts()`

Type	Severity	Location
Gas Optimization	Informational	Line 256

3.4.1 Description

The function `_validateTimesAndAmounts()` should require `numberOfTime == numberOfAmount` first, then require `numberOfTime > 0` instead of require `numberOfTime > 0 && numberOfAmount > 0`.

3.4.2 Recommendation

```
256 require(
257     numberOfTime == numberOfAmount,
258     "CampaignManagement: Times and accounts not match!"
259 );
260 require(
261     numberOfTime > 0,
262     "CampaignManagement: Times can't be zero!"
263 );
```

3.5 ID-05: Bad gas optimization in function `_validateAccountsAndAmounts()`

Type	Severity	Location
Gas Optimization	Informational	Line 278

3.5.1 Description

The function `_validateAccountsAndAmounts()` should require `numberOfAccount == numberOfAmount` first, then require `numberOfAccount > 0`, instead of require `numberOfAccount > 0 && numberOfAmount > 0`.

3.5.2 Recommendation

```
278 require(
279     numberOfAccount == numberOfAmount,
280     "CampaignManagement: Amounts and times not match!"
281 );
282 require(
283     numberOfAccount > 0,
284     "CampaignManagement: Accounts can't be zero!"
285 );
```

3.6 ID-06: Inconsistency unit of parameter `amount`

Type	Severity	Location
Inconsistency	**Informational**	Lines 113,258,280,297

3.6.1 Description

Parameters `amounts` represents the amount of tokens. In different functions, these parameters are inconsistent on unit.

For example, in the constructor, parameter `amounts_` is not multiplied by decimals 10^{18}. But in function `_createCampaign()`, parameter `_amounts` is in the form of multiplied by decimals 10^{18}.

This inconsistency may lead to confusion in programming.

3.6.2 Recommendation

Parameter `amount` should be in the form of multiplied by decimals, in order to be able to represent the amount of tokens as a floating point number.

3.7 ID-07: Bad naming convention

Type	Severity	Location
Coding Style	Informational	

3.7.1 Description

In `constructor`:

```
113 constructor(
114     IERC20 token_,
115     uint256[] memory times_,
116     uint256[] memory amounts_
117 )
```

Parameter `times_` and `amounts_` have suffix _.

In function `_createCampaign()`:

```
294 function _createCampaign(
295 string memory_campaignName,
296 address[] memory_accounts,
297 uint256[] memory_amounts,
298     uint256 releaseTime,
299 bool_isUpdate
300 )
```

Parameter `_campaignName` has prefix _, but parameter `releaseTime` has no prefix _.

3.7.2 Recommendation

Name of parameters passed into functions should be unified one of the two conventions:

- Neither add underscore character _ to prefix or suffix of parameters.
- Either add underscore character _ to prefix or suffix of parameters.

3.8 ID-08: Bad function naming

Type	Severity	Location
Coding Style	Informational	Lines 230, 391

3.8.1 Description

Name of function `getTokenCanUse()` in line 230 should be changed to `getUseableTokenAmount`.

Name of function `_getTokenMustNotUsed()` in line 391 should be changed to `_getUnuseableTokenAmount`.

3.9 ID-09: Bad gas optimization in function `_getUsedTokenByStatus()` and `_getAllowanceByStatus()`

Type	Severity	Location
Gas Optimization	Informational	Lines 403, 423

3.9.1 Description

The declaration of parameters `participants` and `participantLength` may be redundant when `if condition` is not passed.

3.9.2 Recommendation

The declaration of parameters `participants` and `participantLength` should be in the scope of `if` to optimize gas. These functions can be rewritten as follows:

```
403 function _getUsedTokenByStatus(CampaignStatus status)
404         private
405         view
406         returns (uint256 activeToken)
407     {
408         uint256 campaignLength =_campaignNames.length;
409         for (uint256 i = 0; i < campaignLength; i++) {
410             string memory name = _campaignNames[i];
411             Campaign memory campaign =_campaigns[name];
412             if (campaign.status == status) {
413                 Participant[] memory participants = campaign.
                        participants;
414                 uint256 participantLength = participants.length;
415                 for (uint256 j = 0; j < participantLength; j++) {
416                     Participant memory joiner = participants[j];
417                     activeToken += joiner.amount;
418                 }
419             }
420         }
421     }
```

```
423 function _getAllowanceByStatus(CampaignStatus status)
424         private
425         view
426         returns (uint256 activeToken)
427     {
428         uint256 campaignLength =_campaignNames.length;
429         for (uint256 i = 0; i < campaignLength; i++) {
430             string memory name = _campaignNames[i];
431             Campaign memory campaign =_campaigns[name];
432             if (campaign.status == status) {
433                 Participant[] memory participants = campaign.
                        participants;
434                 uint256 participantLength = participants.length;
```

```
435                 for (uint256 j = 0; j < participantLength; j++) {
436                     Participant memory joiner = participants[j];
437                     activeToken += _token.allowance(
438                         address(this),
439                         joiner.account
440                     );
441                 }
442             }
443         }
444     }
```

3.10 ID-10: Logic issue of function `_createCampaign()`

Type	Severity	Location
Logical Issue	Low	Line 294

3.10.1 Description

In line 302:

```
302 _campaignNames.push(_campaignName);
```

Value `campaignName` is pushed into array `_campaignNames`. Therefore, function `_getTokenCanUse()` will iterate over this `campaignName` uselessly.

3.10.2 Recommendation

Despite `campaignName` doesn't have data in mapping `_campaigns`, this is not optimal for gas and may lead to some logical issues in the future. It is better to move line 302:

```
302 _campaignNames.push(_campaignName);
```

right below line 309

```
309 require(tokenCanUse >= totalAmount, "Not enough erc20 token");
```

3.11 ID-11: Bad modifier naming

Type	Severity	Location
Coding Style	Informational	Line 87

3.11.1 Description

`AdminConsentStatus` has three values: *NoAction*, *Reject*, *Accept*. It is more logical to change the name of modifer `confirmedRelease` to `notRejectedRelease`. It is compatible with modifer `notConfirmedRelease`.

3.12 ID-12: Function `_getAllowanceByStatus()` calculate wrong `activeToken`

Type	Severity	Location
Logical Issue	Medium	Line 423

3.12.1 Description

Function `_getAllowanceByStatus(status)` calculates the total allowance amount of all participants in all campaign that has `CampaignStatus` = `status`.

In function `_getAllowanceByStatus(status)` (line 423), `_token.allowance()` will return token amount which an *account* can spend over all campaigns. By the following code

```
437 activeToken += _token.allowance(address(this), joiner.account);
```

if an *account* joins more than one campaign, his `activeToken` will be plus more than one time of `_token.allowance()`.

As a result, function `_getAllowanceByStatus(status)` does not perform as expected.

3.13 ID-13: Unbalance issue of campaign creation and deletion

Type	Severity	Location
Logical Issue	Low	Line 129

3.13.1 Description

Any malicious admin can call function `createCampaign()` to create a useless campaign and decrease the amount of `tokenCanUse`. But to *Delete* these campaigns, we need at least half of the admins to vote *Reject* to *Delete* spam campaign and restore `tokenCanUse`.

3.14 ID-14: Proper usage of `public` and `external` access

Type	Severity	Location
Gas Optimization	Informational	

3.14.1 Description

The public functions:

- `createCampaign()` in line 129
- `adminAcceptRelease()` in line 141
- `adminRejectRelease()` in line 153
- `release()` in line 165
- `deleteCampaign()` in line 195
- `getDatas()` in line 209
- `getCampaigns()` in line 213
- `getCampaign()` in line 217

are only called from external, it is better to change from `public` to `external` for gas optimization.

4 Appendix

4.1 Severity Definitions

Critical

This level vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.

Medium

This level vulnerabilities are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to critical-risk severity.

Low

This level vulnerabilities should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution.

Informational

This level vulnerabilities can be ignored. They are code style violations and informational statements in the code. They may not affect the smart contract execution.

4.2 Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete`.