LIGO

WUΛFMAN CORPORATION

Audit SMAK Farming V2

January 2022

Tezos

Contents

Disclaimer

This report does not provide any warranty or guarantee regarding the absolute bug-freenature of the technology analysed.

This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Wulfman Corporations position is that each company and individual are responsible for their own due diligence and continuous security. Wulfman Corporations goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyse.

Introduction

This audit was commanded to Wulfman Corporation, in quality of main contributor and expert of LigoLANG, by SmartChain

The object of the audit is the analysis of the SMAK Farming V2in order to identify vulnerabilities and contract optimizations in the source code.

The contract targets the Tezos blockchain and was developed in LigoLANG. The auditing methods consist in manual review

The auditing process paid special attention to ensuring that the contract logic is coherent and implements the specification and the best testing schemes.

Overview

Project summary

Project Name	SMAK Farming V2
Publisher	SmartChain
Platform	Tezos
Language	LigoLANG(cameligo flavor)
Codebase	https://github.com/smart-chain-fr/SMAK-Farms
Original commit	89874680827edc379b8b89658b4f72dd97b00498
Contract adress	
Contract url	

Audit summary

Auditer	Wulfman Corporation
Delivery date	January 2022
Scope	Farm contract, Factory & FA contract
Methodology	Manual review
Tezos version	10 Hangzhou
Tezos client version	
LigoLANGversion	0.34.0

Vulnerability summary

Total issues	14
Critical	4
Major	3
Medium	2
Minor	3
Informational	2

Code Quality summary

Total improvements	2
Maintenance	0
Scalability	0
Readability	1
Origination cost	0
Gas cost	1

Vulnerability

Contents

V1. The administrator can steal my tokens

Category	Severity	Location	Status
Trust issue	Critical	FA2.ml	

Description

The transfer entrypoint verify the transfer authorization as such

```
1    let sent_by_owner : bool = (current_sender = from_) or (current_sender =
     ↪  s.administrator) in
2    let sent_by_operator : bool = Set.mem {owner=from_; operator=current_sender;
     ↪  token_id=tr_tokenid} s.operators in
3    if sent_by_owner or sent_by_operator then
4
```

Which mean that for the contract, the administrator is the owner of all the tokens, and such can the administrator can transfer any token to any other address.

Solution

This is the correct check : `let sent_by_owner : bool = (current_sender = from_)` which comply with the TZIP specification

V2. Balance_of is failing

Category	Severity	Location	Status
Undesired failure	Critical	FA2.ml	

Description

The TZIP specification

- If the account does not hold any tokens, the account balance is interpreted as zero.
- If one of the specified token_ids is not defined within the FA2 contract, the entrypoint MUST fail with the error mnemonic "FA2_TOKEN_UNDEFINED".

Instead, the contract is doing this :

```
1    match Map.find_opt (i.owner, i.token_id) s.ledger with
2    | Some e -> { request = i ; balance = e.balance }
3    | None -> (failwith("unknown owner") : balance_of_response)
```

Making the entrypoint fail when the account does not hold any tokens

Solution

To comply with the specification, the `failwith` should be replaced by `{ request = i; balance = e.balance }` An extra test should be added to the `balance_of_response` to check that the token exist, which requires to have a collection of existing token_id which you don't have at the moment since you removed the token metadata

V3. There is no metadata

Category	Severity	Location	Status
	Medium	FA2.ml	

Description

The metadata is suppose to give all the necessary information on the tokens. Without it, tokens are only identify with their token_id. which makes it very inconvenient to understand there different purpose and increase the likely of user to mix them up

Solution

Add the metadata to the FA2 contract, as describe in the TZIP specification

V4. Incorrect error on transfert of unexisting token

Category	Severity	Location	Status
Incorrect contract interaction	Major	FA2.ml	

Description

The TZIP specification

- If the token owner does not hold any tokens of type token_id, the owner's balance is interpreted as zero. No token owner can have a negative balance.
- If one of the specified token_ids is not defined within the FA2 contract, the entrypoint MUST fail with the error mnemonic "FA2_TOKEN_UNDEFINED".

The entrypoint is not checking for the existance of the token, thus the dedicated error is not raised. Thankfully you also didn't respect to first specifiaction, which will make a transfers of zero amount fail with "FA2_INSUFFICIENT_BALANCE". But the end-user will have a complete erroneous message.

Solution

Return `On` when the transfert Check for the token existence before transfering it.

V5. Incorrect error on transfert of unexisting token

Category	Severity	Location	Status
Undesired contract failure	Critical	FA2.ml	

Description

The TZIP specification

- Transfers of zero amount MUST be treated as normal transfers.

The entrypoint is checking that the transfer amount is not zero, and raised a custom failure if it is. This lead to an unexpected failure which revert all transaction

Solution

Remove this check. If the logic is correct, a transfer of zero amount will have no effect on the ledger, but shouldn't trigger a failure

V6. Get your critical FA contract from a trusted source

Category	Severity	Location	Status
	Critical	FA2.ml	

Description

We already raised this point with minimal severity on the previous audit. The level of severity was minimal because then the FA1.2 was correct and happen to be from the Ligo Team. This time, we are raising severity to critical as your implementation contains 3 critical vulnerabilities.

Solution

In december, we publish the 3 possible implementation of the FA2 specification. They have been tested by the Ligo team and no bug have been found yet. You can find them here : `https://github.com/pewulfman/Tezos-TZIP-implementation`

V7. Vulnerability in Farm Remove : parameters are independant

Category	Severity	Location	Status
Unsafe data	Medium	database/main.ml	

Description

Here the entrypoint is assuming that the user will send as parameters the farm address and it's corresponding lp-address. But there the entrypoint doesn't check this assumption, only that the address was previously registered. If the case that we have (farm address A, lp-address A) and (farm address B, lp-address B) and the user send (farm address A, lp-address B), then the entrypoint will remove (farm address A, lp-address A) from `all_farm` and `all_farm_data` but not (lp-address A, farm address A) from `inverse_farms`. The absence of documentation makes it unclear to me how much this would be an issue.

The fact that the entrypoint is only callable by the admin and not by anyone should prevent should limit adversarial behavior but does not prevent human error.

Solution

The entrypoint should only takes `farm_address` as a parameter and obtain `lp-address` from `all_farm_data` This garantie that we have (farm address A, lp-address A) and not (farm address A, lp-address B)

V8. Failure on no-op in Farm Remove

Category	Severity	Location	Status
Undesirable failure	Major	database/main.ml	

Description

The entrypoint is checking that the input `lp-address` was previously registered in `inverse_farm`, which is equivalent the add entrypoint was previously call with the same `lp address` as a parameters. First of all that doesn't mean that it was call with the same couple of parameters, so that the farm was registered so it is uncear what security does this check provide. But even if it was checking that the farm was previously added, then that make the entypoint fail when the admin intend to remove a farm that is not present. The pattern (suppressing something that doesn't exist) doesn't have any logical or security impact. But on the opposite, failing when we try to do so, may lead to a complex contract to contract interaction to fail unexpectedly (or limit the possibility of interaction)

Solution

Treat the case of the absence of preregistered `lp address` as a normal case.

V9. FA1.2 is not indexable

Category	Severity	Location	Status
Violation of Standard	Major	fa12/fa12.ml	

Description

The TZIP specification define a standard for storing the tokens in an FA1.2 contract. This is define so external wallet and dex are able to index them. The ledger in this contract doesn't comply with the specification wich makes it not recognize by external software as a token.

Solution

Use this following type for the ledger : `big_map %ledger address (pair nat (map address nat))` which in ligo is the type `(address,(nat,(address,nat) map)) big_map` under the label `ledger` in the storage

V10. Non mutable value in storage

Category	Severity	Location	Status
bad design	Informational	farm/partials/types.ml	

Description

By definition, the storage store value that are supposed to be mutable, which by looking the contract on BetterCall.dev, a perspective user is suppose to assume that value in the storage can change. Which can lead them to decrease their trust in your program if you put there some critical value such as `input_token_address` `reward_reserve_address`.

V11. The contract should use initialization time instead of creation time

Category	Severity	Location	Status
bad design	Minor	farm/partials/method.ml	

Description

The contract use the amount of time pass since the creation of the contract in it's internal logic. First of all the value of creation time is taken from the storage, but never written there, so it is directly inputed by the creator when emitting the originate operation, and there can be a long time before the operation is inserted. Second of all, since the contract requires a call to the initialization entrypoint to be operational,there can be a big difference between `creation_time` and the time at which the contract is operational. In my opinion this is not desirable. At least putting it as a initialization parameter if not use `Tezos.now` for a bit more control on the starting time.

V12. Duplicated initialization information

Category	Severity	Location	Status
bad design	Informational	database/partials/method.ml	

Description

Given the two checks for uninitialization in the initialization entrypoint. The information of initialization is stored twice in the storage. One as a boolean `initialized` and one as the size of the `reward_at_week` list. It is a bad design has you can have incoherent value. A better design is two choose one and be consistent

V13. Absence of comment and documentation

Category	Severity	Location	Status
	Minor	everywhere	

Description

There is little to no documentation and comment. This makes it difficult for the developers and reviewers to understand and analyse the code and check if it comply to the specification

Solution

1. Write a specification of the expected behaviour of the contract across invocations (what are the invariants?).
2. Comment the code and explain how it relates to the specification, and what are the invariants and pre/post-conditions of each function.

3. Explicit the invariants between the fields of the storage.

V14. Use of imprecise fix-point arithmetic to represent fractions

Category	Severity	Location	Status
Imprecise calculation	Minor	farm/partials/method	

Description

The reward per-week is calculated from a geometric series, thus every value is a retional number which is projected several times into a natural number, each time losing some precision

Solution

Define a module to represent infinite precision retional number by using the type `t = {p:nat;q:nat}` which is the value of the numerator and denominator. Use multiplication on both side and only project them at the end, when you need to convert them to token values

Code Quality

Contents

Most remark from FarmV1 audit that were not fixed are still applicatble to FarmV2 (usage of `import`, `assert with error`). We found some new possible improvements

Q1. Testing boolean

Category	Impact	Location	Status
Code smell	Gas	farm/partials/method	

Description

In several places, the code check for equality of a boolean value for `true` or `false` which is a bad design and increase the size of the michelson for no reason.

Solution

Use the boolean value directly which is logicaly equivalent to `b = true` or its negation, equivalent to `b = false`

Q2. Usage of boolean instead of Variant value

Category	Impact	Location	Status
Code smell	Readability	farm/partials/method	

Description

Instead of writing :

```
1     let add : bool = true
2     let subtract : bool = false
```

define a sum type : `type op = Add | Sub`. This will protect from missusage of boolean in the code in place of `Add` or `Sub`