Telescr.in

Audit of smart contract Wula

revision 3 dated 23.08.2021

Telescr.in

Summary

Brief information

Project: Wula
Network: BSC
Compiler version: 0.5.12
Optimization: Enabled
Audit date: 23.08.2021

Information

The contract code was reviewed and analysed for vulnerabilities, logical errors and developer exit scams possibilities. This work was carried out concerning the project source code and documentation provided by the customer.

Customer provided project whitepaper (english version), the audit was made based on Chapter 4 and 5 of provided document.

General conclusion

As a result of the audit, no errors were found that affect the security of users' funds on the contract. No obvious signs of an exit scam were found. This is a set of contracts, that are tied together. (!) Warning were found in section B of the current audit - there is excessive centralization of tokens, tokens are associated with the backend of the project.

Liability disclaimer

The telescr.in team within this audit framework is not responsible for the developers or third parties' actions on the platforms associated with this project (websites, mobile applications, and so on). The audit confirms and guarantees only the smart contract correct functioning in the revision provided by the project developers.

Confirmed by digital signature

Aggregated data

The Contract analysis was performed using the following methods:

- Static analysis
  - Checking the code for common errors leading to the most common vulnerabilities
- Dynamic analysis
  - The Contract launching and carrying out the attacks various kinds to identify vulnerabilities
- Code Review

Received data

Recommendation	Type	Priority	Occurrence probability	Line of error
Setting literal values	notice	low		wula.sol, 710, 725, 726, 728
Using the Dead-address as BlackHole	notice	low		wula.sol
Code style	notice	low	low	wula_stack.sol
Redundant type casting	notice	low	low	wula_stack.sol, 462
Source of Reward values	warning	medium	medium	wula_stack.sol, 708

A. Errors

Not found.

B. Warnings

1. Source of Reward values

It's not clear where the function GetReward gets reward values from. If it's sent from a back-end server, it makes the contract depended on the developers' behaviour

Recommendation: such behaviour must at least explained in comments in source code.

Comments from developers: *Something about the design of the mining reward mechanism makes it impossible to implement it in a fully centralised way. For the back-end statistics, it is all based on the on-chain events of the stak contract to synchronize the status, and based on the latest status, the back-end mining reward logic will decide how much to reward the user. The mining reward logic cannot be implemented in a fully decentralised way. eip1599's part of the mining reward implementation can be used to determine how much reward is given to the user from external signature data. In the meantime, this part is for the wula coin, we are the initiator of the project, we will not at all maliciously construct signature data to modify the user's balance and other operations, where this pledge is only for the wula coin, and the wula operator is ourselves, these are our rewards to our users. So there is absolutely no reason for us to maliciously break it.*

C. Notice

1. Setting literal values

In WULA contract some values set as expressions (e.g. Lines 710, 725, 726, 728)

Recommendation: set as a calculated value to make it more understandable and readable

2. Using the Dead-address as BlackHole

We recommend use the Dead address for readability

3. Code style

We recommend to follow the common code-style for naming contracts, functions and variables

4. Redundant type casting

variable wula is defined as address (L462) and casted each time to IERC20 explicitly Recommendation: define the variable fixed values according to documentation, but this is not covered by code.

D. Remarks

Not found.

Application. Error classification

Priority	
informational	This question is not directly related to functionality but may be important to understand.
low	This question has nothing to do with security, but it can affect some behavior in unexpected ways.
medium	The problem affects some functionality but does not result in an economically significant user funds loss.
high	This issue can result in the user funds loss.
Probability	
low	It is unlikely that the system is in a state in which an error could occur or could be caused by any party.
medium	This problem may likely arise or be caused by some party.
high	It is highly likely that this problem could arise or could be exploited by some parties.

Application. Digital bytecode print

The audit was carried out for the code certain version on the compiler version 0.5.12 with the optimization enabled.

To check the contract bytecode for identity to the one that was analyzed during the audit, you must:

1. Get contract bytecode (in any block explorer)
2. Get SHA1 from bytecode string
3. Compare with reference in this report

Sha1 from bytecode:

6f08456b318dd86081a4914518397975861af0ac

Sha1 from bytecode (non-metadata):

b6c094883ae822b7dc54bc7fd332f8dcc42282f3

Contract address:

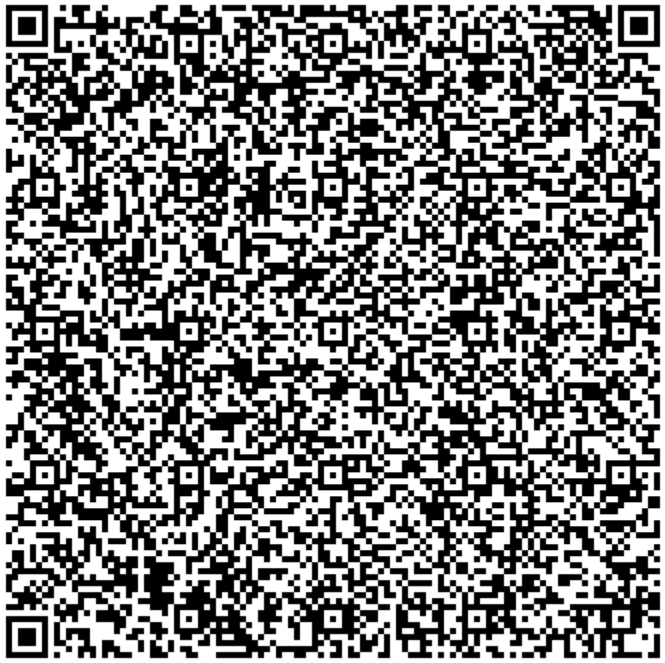
0xa502e153ee236a89842cbd4bc59779cf99d589e1

Check the digital print

Application. Signature of the audit report


```
{
        "address": "0x505ade8cea4db608250e503a5e8d4cb436044d2e",
        "msg": "As a result of the audit, no errors were found that affect the security of users' funds on the contract. No obvious signs of an exit scam were
        found. This is a set of contracts, that are tied together. (!) Warning were found in section B of the current audit - there is excessive
        centralization of tokens, tokens are associated with the backend of the project.  . Sha1 of contract - 6f08456b318dd86081a4914518397975861af0ac. Sha1
        without meta of contract - b6c094883ae822b7dc54bc7fd332f8dcc42282f3Contract address - 0xa502e153ee236a89842cbd4bc59779cf99d589e1",
        "sig": "0x14f92d16034efac7a57b49d1432d00cba1e4431ae0aafc70ec039ad95f0e19b53de687d6d0d0eee456adccb3be5eb12c3df3cb3347ee47a19add09605df278cf1c",
        "version": "3",
        "signer": "MEW"
}
```

Telescr.in



Check the signature