Wildfire Token Security Audit

Wildfire token security audit, conducted by the Callisto Network Security Department during May 2021.

Wildfire Token Security Audit Report

Are Your Funds Safe?

Audit request

Wildfire (WDF) Token smart contract security audit report performed by Callisto Security Audit Department.

- Telegram: https://t.me/WildfireOfficial
- Website: https://wildfire-coin.com/

Source code

https://github.com/Wildfire-new/Wildfirebeb20/blob/main/wildfire.sol

Disclosure policy

… Do you want us to publish the report as it is or to notify you privately in case of finding critical mistakes? … Yes.

… provide your conditions for publishing the report or leave only standard disclosure policy link … Yes.

Standard disclosure policy.

Contact information

Wildfire-NEW@protonmail.com

Platform

BSC.

1. In scope

https://bscscan.com/address/0x6218352Cc511A86F154fe03F69B0aa062e01aeA9#contracts

1.1 Excluded

The correctness of the mathematical calculations was not verified during the audit due to the lack of complete documentation of what the contract should do and under what conditions.

2. Findings

In total, **4 issues** were reported including:

- 2 medium severity issues.
- 1 low severity issue.
- 1 owner privilege.

2.1 Known vulnerabilities of ERC-20 and BEP-20 token

Severity: low.

Description:

1. 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.

Recommendation

Add the following code to the `transfer(_to address, ...)` function:

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

2. ERC20 is a widely used standard across the Ethereum ecosystem. It is reasonable to assume that ERC20 tokens could be “accidentally” deposited into this contract even though it is not intentional.

Every user on the entire Ethereum ecosystem can send ERC20 tokens to this contract and he will have no ability to extract it back unless there is a special “ERC20-rescue” function implemented in your contract. It is advised to implement this function.

Example: here is BAT contract address. As you can see the contract itself holds $497,000 worth of different ERC20 tokens – all these tokens are permanently “stuck” inside the contract and therefore uselessly lost.

Recommendation

A simple “ERC20-rescue” function can solve the problem.

```
function rescueERC20(address _token, uint256 _amount) external onlyOwner {
    IERC20(_token).transfer(owner(), _amount);
  }
```

2.2 Function includeInReward() waste a Gas and has a risk of `OUT_OF_GAS` exception.

Severity: medium.

Description:

The function `includeInReward()` (lines 400-411) use the loop `for()` to find and remove address from the `_excludedFromReward` list. It requires a lot of Gas if many addresses are excluded. In the case of the long excluded list, the function will be aborted with `OUT_OF_GAS` exception.

Recommendation

Use EnumerableSet instead of address array (line 204). So you could remove record using function remove().

2.3 Function _getCurrentSupply() has a risk of `OUT_OF_GAS` exception

Severity: medium.

Description:

The function `_getCurrentSupply()` (lines 361-371) use the loop `for()` (lines 364-368) to calculate `reflectedSupply` and `tokenSupply` values. In the case of the big length of `_excludedFromReward` list, this function will be aborted with `OUT_OF_GAS` exception. Since this function used in most other functions, the performance of the entire contract will be broken.
The risk becomes a higher due Gas cost of storage reading that may grow in future node updates. For example, after applying the EIP-1884 the cost of the SLOAD (0x54) operation changes from 200 to 800 gas. And in EIP-2929 discussing to increase the gas cost of `SLOAD (0x54)` to 2100.
This means that even if with current `_excludedFromReward` list length contract works now, it may stop works in the future when the cost of storage reading will change.

Recommendation

1. Try to avoid long loops.
2. Add limitation of `_excludedFromReward` list growing into function `excludeFromReward()` (lines 391-398):

```
require(_excludedFromReward.length < 50, "Excluded list too long");
```

2.4 Owner privileges

Severity: owner privileges.

Description:

- Owner can include (line 400) and exclude (line 391) any account to/from reward.
- Owner can include (line 387) and exclude (line 382) any account to/from fees.

3. Conclusion

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

Appendix

Smart Contract Audits by Callisto Network.

Miscellaneous

Why Audit Smart Contracts?

Our Most Popular Audit Reports.

Trust the Blockchain, Audit the Smart Contracts.

Follow Callisto's Security Department on Twitter to get our latest news and updates!

Published on **May 26, 2021**

‹ Previous post

Next post ›

Callisto Network LTD

71-75 Shelton Street
London, Greater London
United Kingdom, WC2H 9JQ

Join Our Community

Resources

FAQ
Timeline
Airdrop
Community Guidelines

Callisto

Partners
Our GitHub repositories
Media Kit
Contact us
Want to sell your CLO coins OTC?