Smart Contract Security Audit

SafeLight

https://safelightcrypto.com

04/19/2021

https://fightagainstrugs.com/

audit@fightagainstrugs.com
https://t.me/FightAgainstRug
https://twitter.com/fightrugs

Table of Contents

Background

FAR was commissioned by SafeLight to perform an audit of their smart contract. The purpose of the

audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Project Information

- **Website**: https://safelightcrypto.com
- **Telegram**: https:/ t.me/officialsafelight
- **Platform**: Binance Smart Chain
- **Contract Address**: 0xa5360c2070FaECFc231fd6bd743FE88382F2991d

SafeLight was created to provide our holders with leading tokenomics within the DeFi space.
Their frictionless yield & liquidity generation protocol makes you earn passive income without having to do anything at all.

They charge a **10%** transaction fee which is split in two ways. **5%** goes into a locked liquidity pool and **5%** is shared amongst our stakeholders. is a community-driven, fair-launched DeFi token which builds exponential value for its holders using three main principles : static rewards, liquidity pool acquisition, and burns.

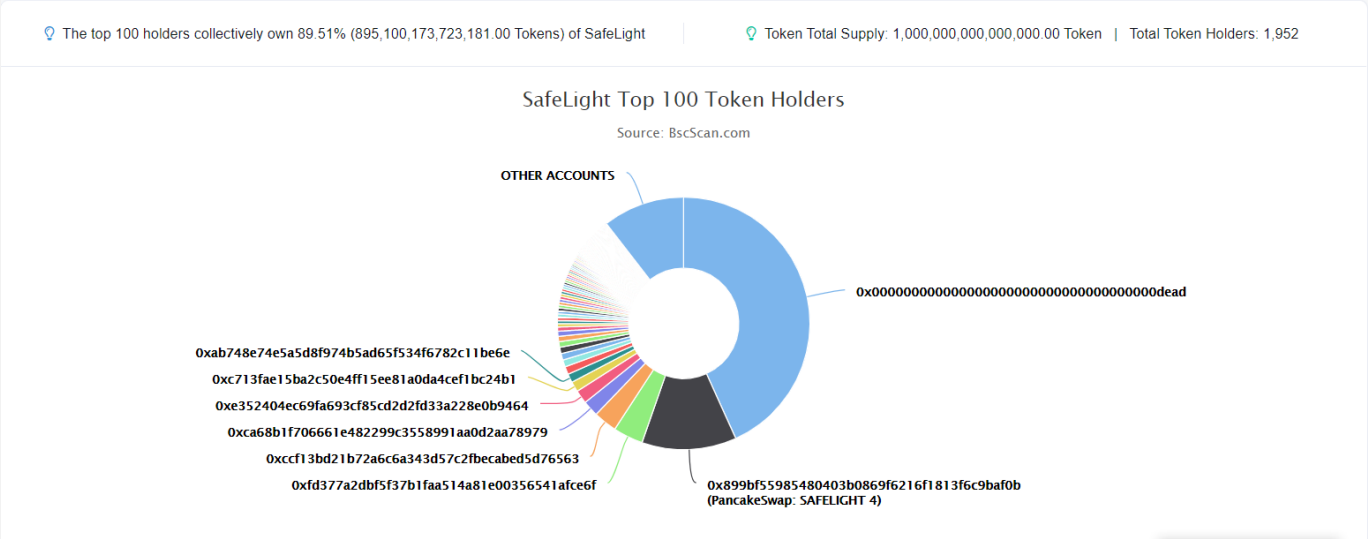
They will also be providing token holders a governance feature that will enable their holders to have control of the platform and a lock up feature that will reward lock ups with access to Ultra rare NFT's in a set. NFT ownership transfers will incur and distribute royalty fees by smart contract

Token Information

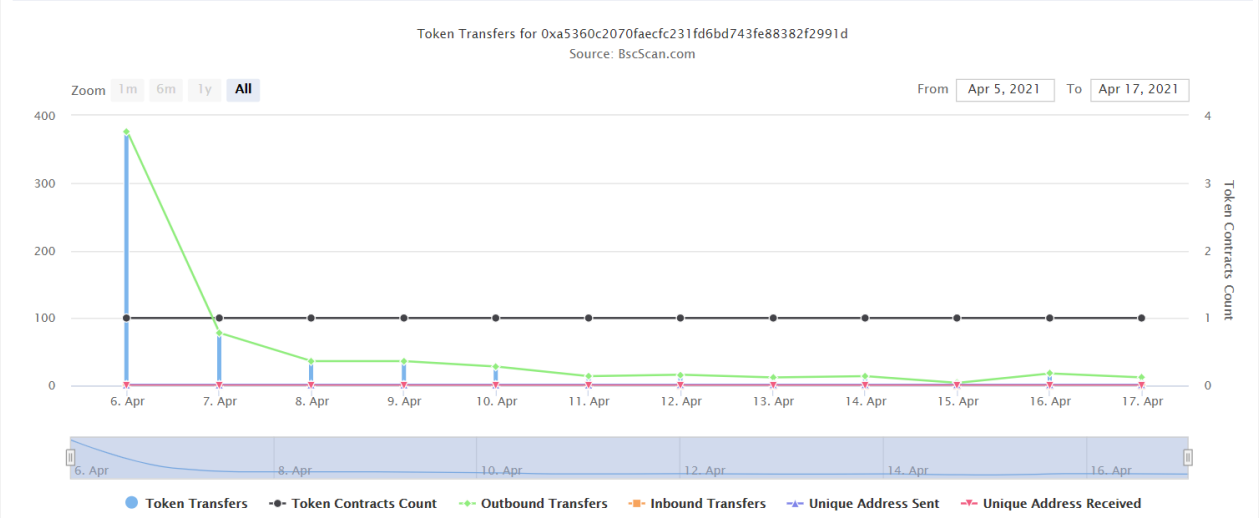
- Name: SAFELIGHT
- Total Supply: 1,000,000,000,000,000
- Circulating Supply: 608,000,000,000,000
- Burned Supply: 432,000,000,000,000
- Holders: 1951 addresses
- Total transactions: 7504
- Contract: 0xa5360c2070FaECFc231fd6bd743FE88382F2991d

- 43.2% burned
- 11.8% added for initial PancakeSwap liquidity
- 2% held in the Dev wallet
- 10% Tax on each transaction
- 0.5% Marketing Wallet

SafeLight Token Distribution:



Contract Interaction Details:



Executive Summary

The contract is based on the famous RFI project with some known liquidity addition mechanisms.
According to our assessment, the customer`s solidity smart contract is **Secured**.

Well Secured	
Secured	✓
Poor Secured	
Insecure	

Automated checks are with smartDec, Mythril, and remix IDE. All issues were performed by our team, which included the analysis of code functionality, manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

We found 0 critical, 0 high, 0 medium, 2 low and 0 very low level issues.

File and Function Level Report

File in Scope:

Contract Name	MD5	Contract Address
SafeLight.sol	**4abcaffcbdc84ba903f9b12c857082d7**	0xa5360c2070FaECFc231fd6bd743FE88382F2991d

- Contract: Renounced
- Inherit: Context
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Return Type	Score
owner	✓	public	**Passed**
onlyOwner	✓	public	**Passed**
renounceOwnership	✓	public	**Passed**
transferOwnership	✓	public	**Passed**
geUnlockTime	✓	public	**Passed**
lock	✓	public	**Passed**
unlock	✓	public	**Passed**

- Contract: **SafeLight**
- Inherit: Context, IERC20, Renounced
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Return Type	Score
name	✓	public	**Passed**
symbol	✓	public	**Passed**
decimals	✓	public	**Passed**
totalSupply	✓	public	**Passed**
balanceOf	✓	public	**Passed**
transfer	✓	public	**Passed**
allowance	✓	public	**Passed**
approve	✓	public	**Passed**
TransferFrom	✓	public	**Passed**
increaseAllowance	✓	public	**Passed**
decreaseAllowance	✓	public	**Passed**
isExcludedFromReward	✓	public	**Passed**
totalFees	✓	public	**Passed**
deliver	✓	public	**Passed**
reflectionFromToken	✓	public	**Passed**
tokenFromReflection	✓	public	**Passed**
excludeFromReward	✓	owner	**Passed**
includeInReward	✓	owner	**Passed**
transferBothExcluded	✓	private	**Passed**
excludeFromFee	✓	owner	**Passed**
includeInFee	✓	owner	**Passed**

setTaxFeePercent	✓	owner	**Passed**
setLiquidityFeePercent	✓	owner	**Passed**
setMaxPercent	✓	owner	**Passed**
setSwapAndLiquifyEnabled	✓	owner	**Passed**
_reflectFee	✓	private	**Passed**
_getValues	✓	private	**Passed**
_getTValues	✓	private	**Passed**
_getRValues	✓	private	**Passed**
_getRate	✓	private	**Passed**
_getCurrentSupple	✓	private	**Passed**
_takeLiquidity	✓	private	**Passed**
_calculateTaxFee	✓	private	**Passed**
calculateLiquidityFee	✓	private	**Passed**
removeAllFee	✓	private	**Passed**
restoreAllFEe	✓	private	**Passed**
isExcludedFromFee	✓	public	**Passed**
_approve	✓	private	**Passed**
_transfer	✓	private	**Passed**
swapAndLiquify	✓	private	**Passed**
swapTokenForEth	✓	private	**Passed**
addLiquidity	✓	private	**Passed**
_tokenTransfer	✓	private	**Passed**
_transferStandard	✓	private	**Passed**
_transferToExcluded	✓	private	**Passed**
_transferFromExcluded	✓	private	**Passed**

Issues Checking Status

No.	Issue Description	Checking Status
1	Compiler warnings.	**Passed**
2	Race conditions and Reentrancy. Cross-function race conditions.	**Passed**
3	Possible delays in data delivery.	**Passed**
4	Oracle calls.	**Passed**
5	Front running.	**Passed**
6	Timestamp dependence.	**Passed**
7	Integer Overflow and Underflow.	**Passed**
8	DoS with Revert.	**Passed**
9	DoS with block gas limit.	**Passed**
10	Methods execution permissions.	**Passed**
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	**Passed**
12	The impact of the exchange rate on the logic.	**Passed**
13	Private user data leaks.	**Passed**
14	Malicious Event log.	**Passed**
15	Scoping and Declarations.	**Passed**
16	Uninitialized storage pointers.	**Passed**
17	Arithmetic accuracy.	**Passed**
18	Design Logic.	**Passed**

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Info	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Audit Findings

Critical:
No critical severity vulnerabilities were found.

High:
No high severity vulnerabilities were found.

Medium:
No Medium severity vulnerabilities were found.

Low:
Issue #1. Out of gas:
Issue: ❑The function includeInReward() uses the loop to find and remove addresses from the _isExcluded list. Function will be aborted with OUT_OF_GAS exception if there will be a long excluded addresses list.

```
function includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already excluded"); for
    (uint256 i = 0; i < _excluded.length; i++) {
```

```
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
```

❏ The function _getCurrentSupply also uses the loop for evaluating total supply. It also could be aborted with OUT_OF_GAS exception if there will be a long excluded addresses list

```
function _getCurrentSupply() private view returns(uint256, uint256) {
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal,
_tTotal);
        rSupply = rSupply.sub(_rOwned[_excluded[i]]); tSupply
        = tSupply.sub(_tOwned[_excluded[i]]);
    }
    if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
    return (rSupply, tSupply);
  }
```

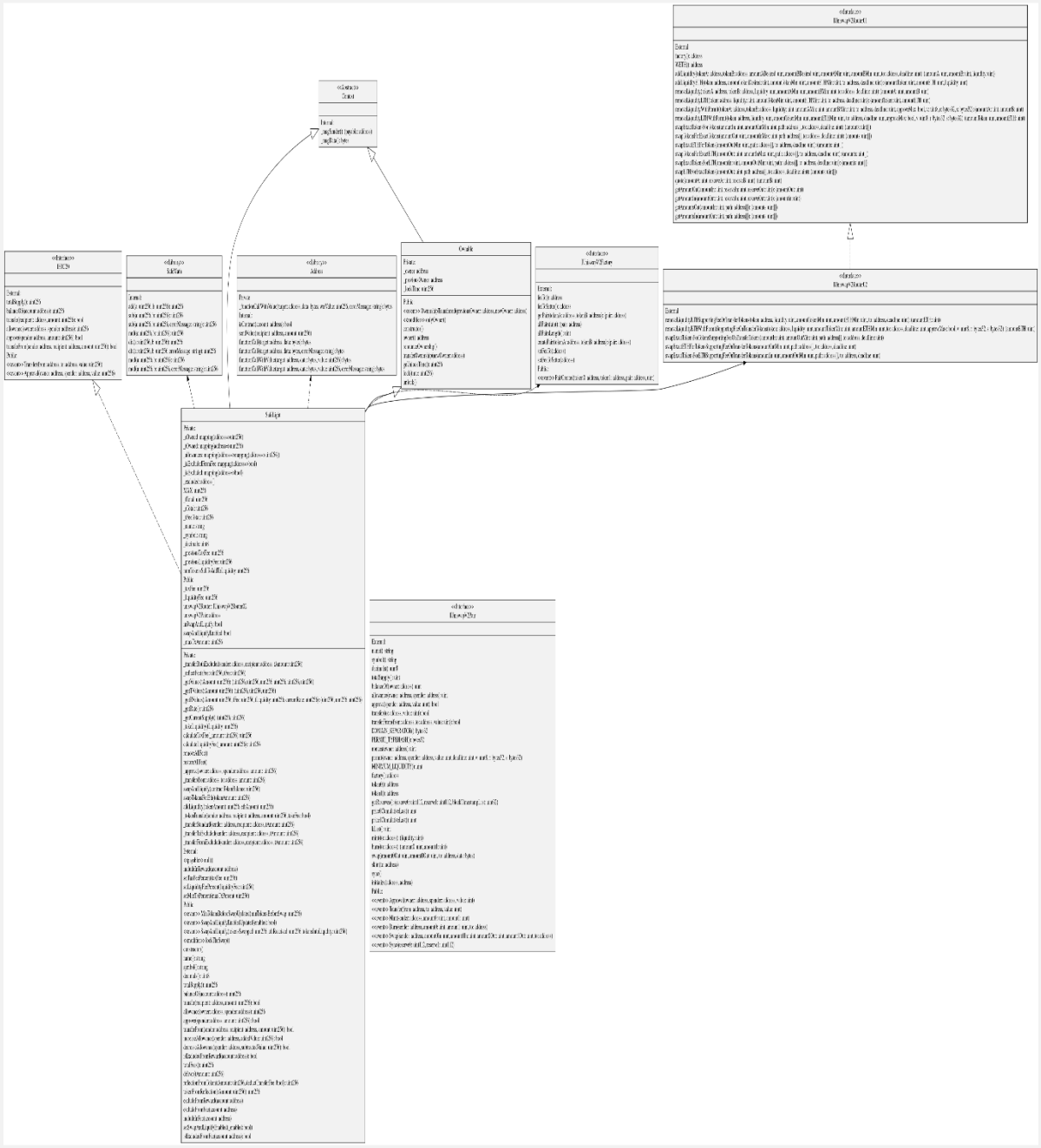
Recommendation: Use EnumerableSet instead of array or do not use long arrays.

Issue #2 : Optimization Error

Description:
public functions that are never called by the contract should be declared external to save gas.
Recommendation:
Consider using “external” attribute for these functions: SafeLight.isExcludedFromReward, SafeLight.totalFees, SafeLight.deliver, SafeLight.reflectionFromToken, SafeLight.excludeFromReward, SafeLight.changeLimit, SafeLight.isExcludedFromFee, SafeLight.excludeFromFee, SafeLight.includeInFee, SafeLight.setSwapAndLiquifyEnabled

Very Low:
No very Low severity vulnerabilities were found.

UML:



Conclusion

We were given contract files. And we have used all possible tests based on the given object.

The contracts are written systematically. We found no critical issues. So, it is good to go for production.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan Everything.

Security state of the reviewed contract is “secured”.

✓ No mint function.
✓ No volatile code.
✓ No high severity issues were found.
✓ Contract Ownership Renounced.

⚠ Liquidity is locked by a third-party service called dxsale.

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and FightAgainstRugs and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (FightAgainstRugs) owe no duty of care towards you or any other person, nor does FightAgainstRugs make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and FightAgainstRug hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, FightAgainstRugs hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against FightAgainstRugs, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

https://fightagainstrugs.com/

audit@fightagainstrugs.com

https://t.me/FightAgainstRug
https://twitter.com/fightrugs