AudiTBlock

SURF Reward

v0.8.19+commit.7dd6d404

Low-Risk	Medium-Risk	High-Risk
low-risk code	medium-risk code	high-risk code

SURFReward

Contract Deployed On bscscan.com

0x53f1e15ed3Cea8c1d4Adc4BE2DDE4BA33715a922

Disclaimer

AudiTBlock is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

AudiTBlock is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your ownresearch.

The information provided in this audit is for informational purposes only and should not be considered investment advice. We does not endorse, recommend, support or suggest to invest in any project.

AudiTBlock can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

Tokenomics

- Bscscan.com

Source Code

- AudiTBlock completes audit phases to perform an audit based on the following smart contract:

- https://bscscan.com/address/0x53f1e15ed3Cea8c1d4Adc4BE2DDE4BA33715a922#code

```
 INFO:Detectors:
SURFReward.internalSwap(uint256)
(contracts/contract.sol#380-406) sends eth to arbitrary user
        Dangerous calls:
        - (success,None) = marketingAddress.call{gas:
35000,value: address(this).balance}()
(contracts/contract.sol#404)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#functions-that-send-ether-to-arbitrary-
destinations

INFO:Detectors:
Reentrancy in
SURFReward._transfer(address,address,uint256)
(contracts/contract.sol#313-344):
        External calls:
        - internalSwap(contractTokenBalance)
(contracts/contract.sol#328)
                -
swapRouter.swapExactTokensForETHSupportingFeeOnTran
sferTokens(contractTokenBalance,0,path,address(this),block.t
imestamp) (contracts/contract.sol#391-399)
                - (success,None) = marketingAddress.call{gas:
35000,value: address(this).balance}()
(contracts/contract.sol#404)
        External calls sending eth:
        - internalSwap(contractTokenBalance)
(contracts/contract.sol#328)
                - (success,None) = marketingAddress.call{gas:
35000,value: address(this).balance}()
(contracts/contract.sol#404)
        State variables written after the call(s):
```

Snapshot 1.1

```
SURFReward.takeTaxes(address,bool,bool,uint256)
(contracts/contract.sol#365-377)
	- balance[to] += amountAfterFee
(contracts/contract.sol#339)
	SURFReward.balance (contracts/contract.sol#169) can be
used in cross function reentrancies:
	- SURFReward._transfer(address,address,uint256)
(contracts/contract.sol#313-344)
	- SURFReward.balance (contracts/contract.sol#169)
	- SURFReward.balanceOf(address)
(contracts/contract.sol#159-161)
	- SURFReward.constructor() (contracts/contract.sol#205-
231)
	- SURFReward.takeTaxes(address,bool,bool,uint256)
(contracts/contract.sol#365-377)
	- amountAfterFee =
takeTaxes(from,is_buy(from,to),is_sell(from,to),amount)
(contracts/contract.sol#338)
		- balance[address(this)] += feeAmount
(contracts/contract.sol#372)
	SURFReward.balance (contracts/contract.sol#169) can be
used in cross function reentrancies:
	- SURFReward._transfer(address,address,uint256)
(contracts/contract.sol#313-344)
	- SURFReward.balance (contracts/contract.sol#169)
	- SURFReward.balanceOf(address)
(contracts/contract.sol#159-161)
	- SURFReward.constructor() (contracts/contract.sol#205-
231)
	- SURFReward.takeTaxes(address,bool,bool,uint256)
(contracts/contract.sol#365-377)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#reentrancy-vulnerabilities
```

Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise,
after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed
condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5
Contracts/SURFReward.sol	d01b3c458bd445222081eca174eeee89

Used Code from other Frameworks/Smart Contracts (direct imports)

Dependency / Import Path	Source Sha1 Hash
Contracts/Context, Ownable, IERC20	a32f9e7070451351211ad2c66 48f3824591c21c4

Snapshot 2.0

```
 SURFReward.changeBuyFee(uint32) (contracts/contract.sol#352-354) should emit an event for:
	- buyfee = _buyfee (contracts/contract.sol#353)
SURFReward.changeSellFee(uint32) (contracts/contract.sol#356-358) should emit an event for:
	- sellfee = _sellfee (contracts/contract.sol#357)
SURFReward.changeTransferFee(uint32) (contracts/contract.sol#360-362) should emit an event for:
	- transferfee = _transferfee (contracts/contract.sol#361)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
Reentrancy in SURFReward._transfer(address,address,uint256) (contracts/contract.sol#313-344):
	External calls:
	- internalSwap(contractTokenBalance) (contracts/contract.sol#328)
		- swapRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(contractTokenBalance,0,path,address(this),block.timestamp) (contracts/contract.sol#391-399)
		- (success,None) = marketingAddress.call{gas: 35000,value: address(this).balance}() (contracts/contract.sol#404)
	External calls sending eth:
	- internalSwap(contractTokenBalance) (contracts/contract.sol#328)
		- (success,None) = marketingAddress.call{gas: 35000,value: address(this).balance}() (contracts/contract.sol#404)
	Event emitted after the call(s):
	- Transfer(from,address(this),feeAmount) (contracts/contract.sol#373)
		- amountAfterFee = takeTaxes(from,is_buy(from,to),is_sell(from,to),amount) (contracts/contract.sol#338)
	- Transfer(from,to,amountAfterFee) (contracts/contract.sol#339)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Context._msgData() (contracts/contract.sol#19-22) is never used and should be removed
SURFReward.is_transfer(address,address) (contracts/contract.sol#279-282) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

Snapshot 2.1

INFO: Detectors

Pragma version=0.8.19 (contracts/contract.sol#8) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

INFO: Detectors:
The redundant expression "this (contracts/contract.sol#20)" inContext (contracts/contract.sol#11-23)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

INFO: Detectors:
Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/contract.sol#86) is too similar to IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/contract.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

INFO: Detectors:
SURFReward.swapRouter (contracts/contract.sol#181) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable

Results: analyzed (8 contracts with 85 detectors), 13 result(s) found

SOLIDITY UNIT TESTING

Progress: Starting
Contract: **SURFReward**

✓ Check winning proposal
✓ Check winning proposal with return value
✓ Before all
✓ Check success
✓ Check success2
✓ Check sender and value

PASS ✓ ✓ **Tested**

Result for tests Passed:

0Time Taken: 0.23s

Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, AudiTBlock experts found **0 medium Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, AudiTBlock experts found **0 High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, AudiTBlock experts found **1 Medium issue** in the code of the smart contract.

LOW ISSUES

During the audit, AudiTBlock experts found **3 Low issues** in the code of the smart contract.

INFORMATIONAL ISSUES

During the audit, AuditBlock experts found **1 Informational issues** in the code of the smart contract.

SWC Attacks

ID	Title		Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✔
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✔
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✔
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✔
SWC-127	Arbitrary Jump with Function TypeVariable	CWE-695: Use of Low-Level Functionality	✔
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✔
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✔
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✔

ID	Title		Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✔
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✔
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✔
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✔
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✔
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✔
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✔
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✔
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✔
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✔

Owner privileges

Verify Claims

- Status: Tested Owner verification ✓
- Status: Tested 1 and Function verified ✓
- Status: Tested 2 and interfaces verified ✓
- Status: Tested 3 and Erc20 verified ✓
- Status: Tested 4 and context verified ✓
- Status: verified ✓

Executive Summary

Two (2) independent AuditBlock experts performed an unbiased and isolated audit of the smart contract. The final debriefs

The overall code quality is good and not overloaded with unnecessary functions, these is greatly

benefiting the security of the contract. It correctly implemented widely used and reviewed contracts he main goal of the audit was to verify the claims regarding the security of the smart contract and the claims inside the scope of work.

During the audit, no issues were found after the manual and automated security testing.

Deployed On Bscscan.com

VERIFIED ✔

https://bscscan.com/address/0x53f1e15ed3Cea8c1d4Adc4BE2DDE4BA33715a922#code