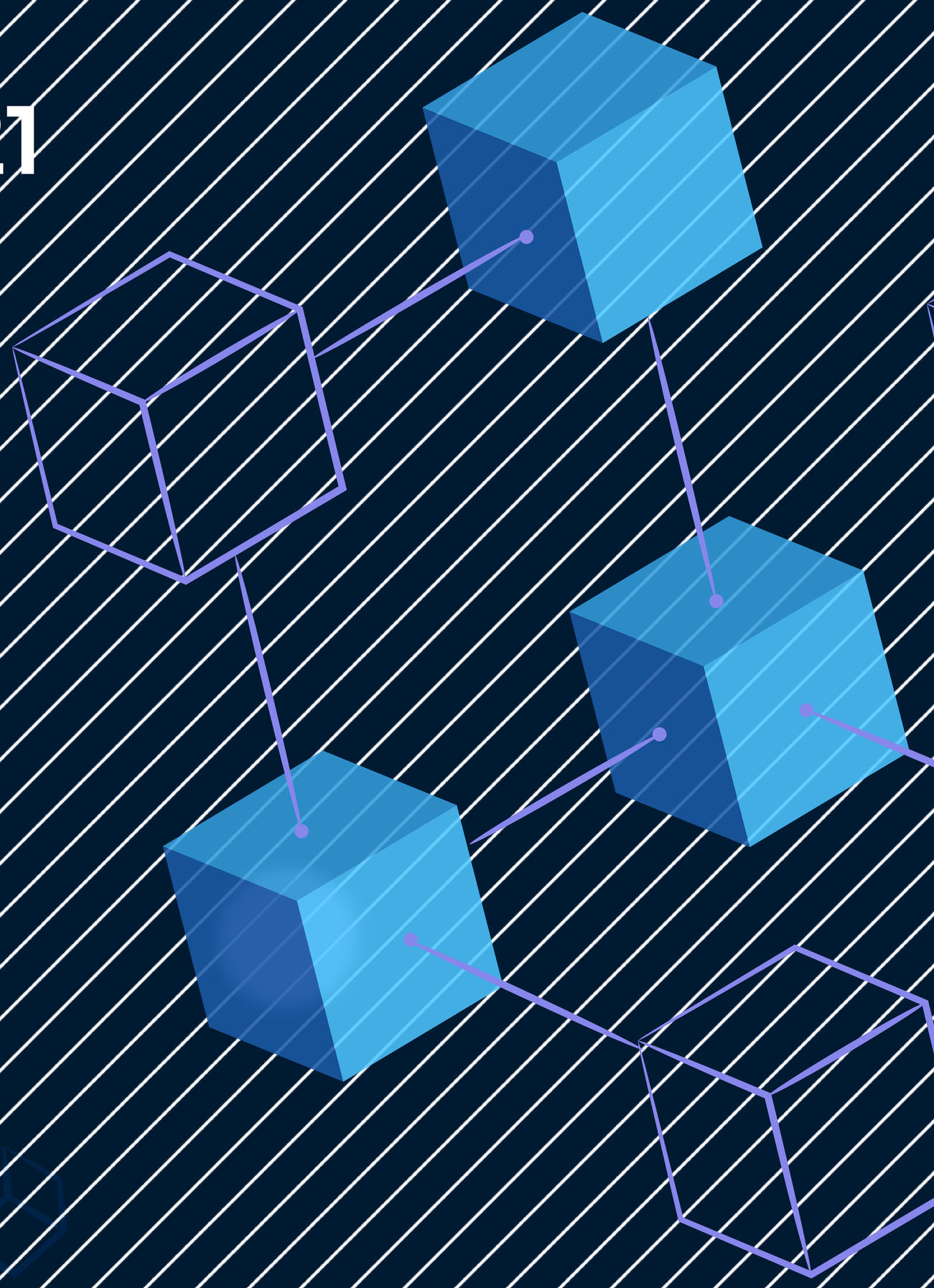
QuillAudits

Audit Report
November, 2021



For

Contents

Contents

Scope of the Audit

The scope of this audit was to analyze and document the Realm Defenders Token smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Token.sol

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	0	0	0
Closed	0	2	1	3

MultiSend.sol

Type	High	Medium	Low	Informational
Open	0	0	0	1
Acknowledged	0	0	0	0
Closed	0	0	0	1

Introduction

During the period of **Nov 14, 2021 to Nov 16, 2021** - QuillAudits Team performed a security audit for **Realm Defenders** smart contracts.

The code for the audit was taken from the following official link:
https://github.com/RealmDefenders/Realm-Defenders/
Commit ID: c29651861373d214ca7a2d19e9519e5c52ee9ca7

Ver	Date	Commit hash
1	Nov 14	https://mumbai.polygonscan.com/address/0x2911 5760c46f5740053d7Dd33405a58F0a8FE531#code
2	Nov 16	https://github.com/RealmDefenders/Realm-Defenders /commit/a1ac02fedce9d6f89414b9eafab5a47dfd8aeae1

Issues Found

A. Contract – Token.sol

High severity issues

No issues were found.

Medium severity issues

A.1 Lack of event emissions

Description
The critical settings are completely devoid of event definitions or emissions. This makes it very difficult for users or other interested parties to track important changes that take place in the system.

Hence, the missing event makes it difficult to track off-chain liquidity fee changes. An event should be emitted for significant transactions calling the following functions:

- updateWhaleAmount()
- updateAntiWhale()

Remediation
Consider emitting events after sensitive changes take place to facilitate tracking and notify off-chain clients that may be following the contracts' activity.

As a result, we recommend logging the update of those variables that change when the aforementioned functions are called.

Status: Fixed in version 02

Low severity issues

A.2 Lack of Zero address validation

Description
To favor explicitness, consider adding a check for all functions that are taking address parameters in the entire codebase. Although most of the functions throughout the codebase properly validate function inputs, there are some instances of functions that do not. One example is:
- **Constructor** - does not check for zero address

Remediation
Consider implementing **require** statements where appropriate to validate all user-controlled input, including governance functions, to avoid the potential for erroneous values to result in unexpected behaviors or wasted gas.

Status: Fixed in version 02

Informational issues

A.3 Old Solidity Version

Description
solc frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks. We also recommend avoiding complex pragma statements.

Remediation
Deploy with any of the following Solidity versions:
- 0.5.16 - 0.5.17
- 0.6.11 - 0.6.12
- 0.7.5 - 0.7.6

Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

Status: Fixed in version 02

A.4 Missing docstrings

It is extremely difficult to locate any contracts or functions, as they lack documentation. One consequence of this is that reviewers' understanding of the code's intention is impeded, which is significant because it is necessary to accurately determine both security and correctness.

They are additionally more readable and easier to maintain when wrapped in docstrings. The functions should be documented so that users can understand the purpose or intention of each function, as well as the situations in which it may fail, who is allowed to call it, what values it returns, and what events it emits.

Status: Fixed in version 02

A.5 Typos

Please consider fixing the following typos:

L462: **Destoys** should be **Destroys**

L530: **stardard** should be **standard**

Status: Fixed in version 02

B. Contract – MultiSend.sol

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Informational issues

B.1 Using old Solidity version

Throughout the contract, Solidity 0.8.0 is used and is deployable. However, at the time of writing there are 2 known bugs in version 0.8.0. Consider upgrading your contracts to use the latest version of Solidity, 0.8.9.

Status: Fixed in version 02

Functional test

Token.sol
Address: 0x4923D374f3d84Es448BC2fbd22D920FfAcF0E031b
Evn: Ganache

Function name	Input	Output	TX	Status
updateAntiWhale	false	true	0x22cca6f3f55e1de4b829e010d756199f6e71903d71ebe88801f48e7fa22ca583	Passed
updateWhaleAmount	6000000	true	0x35fe17a0c85049096894cfdff7cbd3910b3c683daefcaa903fc25789ae503fa7	Passed
transfer	"0x61a6562754e5ef7d91f22A435e9d7D1FB84e8B07","6000000"	true	0xc2da6bddde89f99efecf0be148e33b2e725c6e28c39da6276d8bfc1b78ce6ac6	Passed
transfer	0x61a6562754e5ef7d91f22A435e9d7D1FB84e8B07","7000000"	reverted	N/A	Passed
transfer	"0x61a6562754e5ef7d91f22A435e9d7D1FB84e8B07","1000000"	true	0x26ac21c2bdd0bafc54a235427bde961e375e3dae6c621dfedfe22289d0fc2763	Passed
BalanceOf	0x61a6562754e5ef7d91f22A435e9d7D1FB84e8B07	7000000	call0x4bf71f3704d83af4CBBF93BFB6E4D8292aebC7B30x4923D374f3d84E448BC2fbd22D920FfAcF0E031b0x70a0823100000000000000000000000061a6562754e5ef7d91f22a435e9d7d1fb84e8b07	Passed
approve	"0x2Aee1d68836D4B95515d9	true	0x30c9aad42ee040effa4fa1f455dad8f7586d3b	Passed

	c7e4603D24CD86Ae669","7000000"		4f12721ad717286d29ac62fb8f	
allowance	"0x61a6562754e5ef7d91f22A435e9d7D1FB84e8B07","0x2Aee1d68836D4B95515d9c7e4603D24CD86Ae669"	7000000	call0x2Aee1d68836D4B95515d9c7e4603D24CD86Ae6690x4923D374f3d84E448BC2fbd22D920FfAcF0E031b0xdd62ed3e0000000000000000000000061a6562754e5ef7d91f22a435e9d7d1fb84e8b07000000000000000000000000002aee1d68836d4b95515d9c7e4603d24cd86ae669	Passed
transferFrom	"0x61a6562754e5ef7d91f22A435e9d7D1FB84e8B07","0x2Aee1d68836D4B95515d9c7e4603D24CD86Ae669", 7000000	reverted	N/A	Passed
transferFrom	"0x61a6562754e5ef7d91f22A435e9d7D1FB84e8B07","0x2Aee1d68836D4B95515d9c7e4603D24CD86Ae669", 6000000	true	0x975a1881f0f89026a54427035ecacbe60227c637cba467704005d1bf466eb887	Passed
BalanceOf	0x2Aee1d68836D4B95515d9c7e4603D24CD86Ae669	6000000	call0x61a6562754e5ef7d91f22A435e9d7D1FB84e8B070x4923D374f3d84E448BC2fbd22D920FfAcF0E031b0x70a08231000000000000000000000000002aee1d68836d4b95515d9c7e4603d24cd86ae669	Passed
allowance	"0x61a6562754e5ef7d91f22A435e9d7D1FB84	1000000		Passed

	e8B07","0x2Aee1d68836D4B95515d9c7e4603D24CD86Ae669"			
decreaseAllowance	"0x2Aee1d68836D4B95515d9c7e4603D24CD86Ae669","500000"	true	0xf640aa9015a23b825d7ae7f1c5f7887589a27fdf80ac225c942bd95b0c0ea60e	Passed
allowance	"0x61a6562754e5ef7d91f22A435e9d7D1FB84e8B07","0x2Aee1d68836D4B95515d9c7e4603D24CD86Ae669"	500000	call0x61a6562754e5ef7d91f22A435e9d7D1FB84e8B070x4923D374f3d84E448BC2fbd22D920FfAcF0E031b0xdd62ed3e0000000000000000000000061a6562754e5ef7d91f22a435e9d7d1fb84e8b0700000000000000000000000000002aee1d68836d4b95515d9c7e4603d24cd86ae669	Passed
increaseAllowance	"0x2Aee1d68836D4B95515d9c7e4603D24CD86Ae669","500000"	true	0xf5e9331609b737ce29fcae4c5d39591dd6afe5f2fb701a9f534be8d9f1ee3a8f	Passed
allowance	"0x61a6562754e5ef7d91f22A435e9d7D1FB84e8B07","0x2Aee1d68836D4B95515d9c7e4603D24CD86Ae669"	1000000		Passed
burn	6000000 from 0x2Aee1d68836D4B95515d9c7e4603D24CD86Ae669	true	0x4e5f21661851f800d535b9a2ea3c14db212f8c3212648f431c5c79d8166d7497	Passed
BalanceOf	0x2Aee1d68836D4B95515d9c	0	call0x2Aee1d68836D4B95515d9c7e4603D24	Passed

	7e4603D24CD86Ae669		CD86Ae6690x4923D374f3d84E448BC2fbd22D920FfAcF0E031b0x70a0823100000000000000000000002aee1d68836d4b95515d9c7e4603d24cd86ae669	

multiSend.sol
Address: 0x755141d640562Ee206355Bf31cF37e90b8101CE3
Evn: Ganache

Function name	Input	Output	TX	Status
multiSend	"0x4923D374f3d84E448BC2fbd22D920FfAcF0E031b",["0x2Aee1d6883 6D4B95515d9c7e4603D24CD86Ae669","0x4bf71f3704d83af4CBBF93BFB6E4D8292aebC7B3"],["1000000","1000000"],"2000000"	true	0xf8cf8af851bece77d16128810ce6a7bff949781c046f97bd90a89e8a44774627	Passed

Automated Tests - Token.sol

Slither

```
INFO:Detectors:
Contract locking ether found:
        Contract Token (Token.sol#489-563) has payable functions:
         - Token.constructor(string,string,uint8,uint256,bool,uint256,address) (Token.sol#502-520)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether
INFO:Detectors:
Token.constructor(string,string,uint8,uint256,bool,uint256,address).name (Token.sol#503) shadows:
        - Token.name() (Token.sol#535-537) (function)
Token.constructor(string,string,uint8,uint256,bool,uint256,address).symbol (Token.sol#504) shadows:
        - Token.symbol() (Token.sol#542-544) (function)
Token.constructor(string,string,uint8,uint256,bool,uint256,address).decimals (Token.sol#505) shadows:
        - Token.decimals() (Token.sol#549-551) (function)
Token.constructor(string,string,uint8,uint256,bool,uint256,address).totalSupply (Token.sol#506) shadows:
        - ERC20.totalSupply() (Token.sol#246-248) (function)
        - IERC20.totalSupply() (Token.sol#11) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Token.constructor(string,string,uint8,uint256,bool,uint256,address).tokenOwnerAddress (Token.sol#509) lacks a zero-check on :
                - tokenOwner = tokenOwnerAddress (Token.sol#514)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
ERC20._burnFrom(address,uint256) (Token.sol#467-474) is never used and should be removed
SafeMath.div(uint256,uint256) (Token.sol#175-182) is never used and should be removed
SafeMath.mod(uint256,uint256) (Token.sol#195-198) is never used and should be removed
SafeMath.mul(uint256,uint256) (Token.sol#150-162) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.5.0 (Token.sol#1) allows old versions
Pragma version^0.5.0 (Token.sol#93) allows old versions
Pragma version^0.5.0 (Token.sol#203) allows old versions
Pragma version^0.5.0 (Token.sol#479) allows old versions
solc-0.5.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter Token.updateWhaleAmount(uint256)._amount (Token.sol#553) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
totalSupply() should be declared external:
        - ERC20.totalSupply() (Token.sol#246-248)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (Token.sol#253-255)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (Token.sol#265-268)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (Token.sol#273-279)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (Token.sol#288-291)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (Token.sol#305-317)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (Token.sol#331-341)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (Token.sol#357-367)
burn(uint256) should be declared external:
        - Token.burn(uint256) (Token.sol#526-528)
name() should be declared external:
        - Token.name() (Token.sol#535-537)
symbol() should be declared external:
        - Token.symbol() (Token.sol#542-544)
decimals() should be declared external:
        - Token.decimals() (Token.sol#549-551)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Mythril

```
[audarbhan@audarbhan contracts % myth a --rpc 127.0.0.1:8545 -a 0x4923D374f3d84E448BC2fbd22D920FfAcF0E031b
The analysis was completed successfully. No issues were detected.
```

SOLHINT LINTER

```
Token.sol
    1:1  error    Compiler version ^0.5.0 does not satisfy the ^0.5.8 semver requirement  compiler-version
   93:1  error    Compiler version ^0.5.0 does not satisfy the ^0.5.8 semver requirement  compiler-version
  159:9  warning  Error message for require is too long                                    reason-string
  203:1  error    Compiler version ^0.5.0 does not satisfy the ^0.5.8 semver requirement  compiler-version
  388:9  warning  Error message for require is too long                                    reason-string
  389:9  warning  Error message for require is too long                                    reason-string
  429:9  warning  Error message for require is too long                                    reason-string
  454:9  warning  Error message for require is too long                                    reason-string
  455:9  warning  Error message for require is too long                                    reason-string
  479:1  error    Compiler version ^0.5.0 does not satisfy the ^0.5.8 semver requirement  compiler-version

✖ 10 problems (4 errors, 6 warnings)
```

Results

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Automated Tests - MultiSend.sol

Slither

```
INFO:Detectors:
SafeERC20.safeApprove(IERC20,address,uint256) (MultiSend.sol#163-169) is never used and should be removed
SafeMath.add(uint256,uint256) (MultiSend.sol#89-91) is never used and should be removed
SafeMath.div(uint256,uint256) (MultiSend.sol#103-105) is never used and should be removed
SafeMath.div(uint256,uint256,string) (MultiSend.sol#122-131) is never used and should be removed
SafeMath.mod(uint256,uint256) (MultiSend.sol#107-109) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (MultiSend.sol#133-142) is never used and should be removed
SafeMath.mul(uint256,uint256) (MultiSend.sol#98-100) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (MultiSend.sol#111-120) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (MultiSend.sol#51-57) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (MultiSend.sol#75-80) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (MultiSend.sol#82-87) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (MultiSend.sol#66-73) is never used and should be removed
SafeMath.trySub(uint256,uint256) (MultiSend.sol#59-64) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.0 (MultiSend.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable MultiSend.ERC20Interface (MultiSend.sol#176) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

Mythril

```
enderphan@enderphan contracts % myth a --rpc 127.0.0.1:8545 -a 0x755141d640562Ee206355Bf31cF37e90b8101CE3
The analysis was completed successfully. No issues were detected.
```

SOLHINT LINTER

```
MultiSend.sol
    3:1  error    Compiler version 0.8.0 does not satisfy the ^0.5.8 semver requirement  compiler-version
  151:9  warning  Provide an error message for require                                   reason-string
  160:9  warning  Provide an error message for require                                   reason-string
  168:9  warning  Provide an error message for require                                   reason-string
  176:5  warning  Variable name must be in mixedCase                                     var-name-mixedcase

✖ 5 problems (1 error, 4 warnings)
```

Results

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the **Realm Defenders** platform. We performed our audit according to the procedure described above.

The audit discovered a number of issues of medium, low, and informational severity. In the end, the Auditee resolved all outstanding issues.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the **Realm Defenders**. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the **Realm Defenders** Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
November, 2021

For

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com