

DAPP SECONDARY SECURITY REVIEW REPORT

Customer: Rewards

Date: March 6, 2019

Platform: Ethereum

Language: Solidity / Javascript

This document may contain confidential information about IT systems and intellectual property of the customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the customer or it can be disclosed publicly after all vulnerabilities fixed - upon decision of customer.

Document

Name	DApp Secondary Security Review Report for Rewards
Platform	Ethereum / Solidity / Javascript
Date	28.02.2019
File	Rewards 2019-02-19.zip
MD5 file hash	6f0b1f3eb59f7593eee6a03a9e4b79a5
Date of the secondary review	06.03.2019
File for the secondary review	Rewards 2019-03-06.rar
MD5 hash of the secondary file	1e28ad204a24af5c0929c6b4d8b2b03b

Table of contents

Document.....	2
Table of contents.....	3
Introduction.....	4
Scope.....	4
Executive Summary.....	5
Vulnerabilities tested.....	6
Severity Definitions.....	6
Smart contract AS-IS overview.....	7
Findings.....	12
Conclusion.....	14
Disclaimers.....	15
Appendix A. Evidences.....	16
Appendix B. Automated tools reports.....	17

Introduction

Hacken OÜ (Consultant) was contracted by Rewards (Customer) to conduct a DAPP security review. This report presents the findings of the security assessment of Customer's DAPP, smart contract and its code review conducted between February, 20th 2019 - February 28th, 2019. The secondary review was conducted on March 6th, 2019.

Scope

Scope of the project is smart contract audit and penetration testing of web application.

The scope of the smart contract audit is **RewardsTokenDistribution**, **VestingVault** and **RewardsToken** smart contracts.

We have scanned these smart contracts for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered (the full list includes them but is not limited to them):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Byte array vulnerabilities
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

Executive Summary

According to the assessment, Customer's DApp is well-secure.



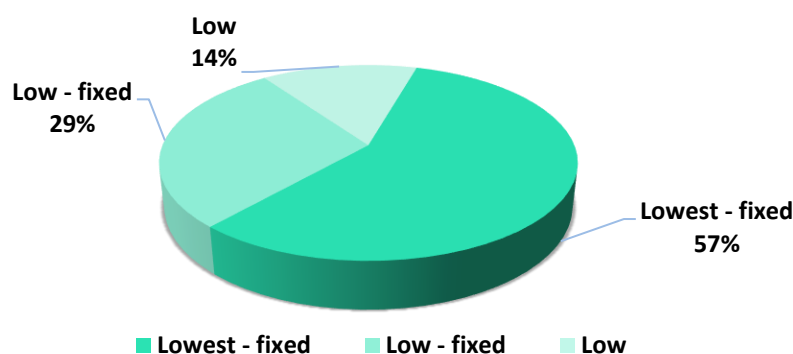
Our team performed analysis of smart contracts code functionality, manual audit and automated checks with Slither and remix IDE (see Appendix B pic 1-10). All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in Findings section. Smart contracts functionality overview is presented in AS-IS section.

Consultants also manually tested web application deployed on test server against applicable web vulnerabilities: XSS and server misconfiguration. We were notified by Customer that DApp will be deployed only locally and all found issues doesn't have serious security impact.

All issues discovered during security review are presented in Findings section.

We found 3 low and 4 lowest vulnerabilities in smart contracts. We also outline 1 best practice recommendation and 1 style guide violation. Most of the issues were fixed after the first audit.

Graph 1. The distribution of vulnerabilities.



Vulnerabilities tested

Vulnerability Name	Issues
Reentrancy	0
Front-Running	0
Integer Overflow and Underflow	0
DoS with (Unexpected) Throw	0
DoS with Block Gas Limit	0
Malicious libraries	0
Compiler version not locked	1 low - fixed
Timestamp Dependence	0
Byte array vulnerabilities	0
Unsafe type inference	0
Check for 0x0 address	1 low - fixed
XSS	1 low - not fixed
Server Misconfiguration	0

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens lose 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
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Smart contract AS-IS overview

RewardsToken contract overview

RewardsToken contract uses **Ownable** contract and **SafeMath** library from OpenZeppelin repository. **SafeMath** library is used for math operations with safety checks that detect errors.

RewardsToken describes ERC20 token with next parameters:

- **symbol** - **RWD**
- **name** - **Rewards Cash**
- **decimals** - **18**
- **hardCap** - **10 ** (18 + 9)**

RewardsToken has 2 modifiers:

- **canMint** - checks whether **mintingFinished** is not **false**.
- **canTransfer** - checks whether **msg.sender** is owner or frozen is not **false**.

RewardsToken has 14 functions:

- **mint** is a public function - mints a specified amount of tokens to the specified address. Has **onlyOwner** and **canMint** modifiers.
- **finishMinting** is a public function - sets **mintingFinished** to **true**. Has **onlyOwner** modifier.
- **startMinting** is a public function - sets **mintingFinished** to **false**. Has **onlyOwner** modifier.

- **transfer** is a public function - transfers specified amount of tokens to the specified account. Has **canTransfer** modifier.
- **transferFrom** is a public function - transfers specified amount of allowed tokens to the specified account. Has **canTransfer** modifier.
- **approve** is a public function - approves a specified amount of tokens.
- **allowance** is a public view function - returns an amount of tokens that an owner allowed to a spender.
- **increaseApproval** is a public function - increases an amount of approved tokens for the specified account.
- **decreaseApproval** is a public function - decreases an amount of approved tokens for the specified account.
- **balanceOf** is a public view function - returns balance of the specified address.
- **burn** is a public function - burns specified amount of tokens from **msg.sender** balance.
- **revoke** is a public function - revokes specified amount of tokens from the specified address. Has **onlyOwner** modifier.
- **freeze** is a public function - sets **frozen** to **true**. Has **onlyOwner** modifier.
- **unfreeze** is a public function - sets **frozen** to **false**. Has **onlyOwner** modifier.

VestingVault contract overview

VestingVault contract is used to implement token vesting scheme.

VestingVault constructor sets:

- **token** to **_token**
- **locked** to **false**

VestingVault has 1 modifiers:

- **isOpen** - checks whether **locked** is **false**.

VestingVault has 9 functions:

- **returnVestedAddresses** is a public view function - returns an array of **vestedAddresses**.
- **returnGrantInfo** is a public view function - returns information about grant for the specified address.
- **grant** is a public view function - returns an array of **vestedAddresses**. Has **onlyOwner** and **isOpen** modifier.
- **transferableTokens** is a public view function - returns a number of tokens available to transfer at specified time.
- **calculateTransferableTokens** is a private pure function - returns an available amount of tokens to transfer at specified time.
- **claim** is a public function - transfers vested tokens to `msg.sender`.

- **burnRemainingTokens** is a public function - transfers all tokens from contract to the owner and burns them. Has **onlyOwner** modifier.
- **withdraw** is a public function - transfers all tokens from contract to the owner. Has **onlyOwner** modifier.
- **lockVault** is a public function - locks vault.

RewardsTokenDistribution contract overview

RewardsTokenDistribution contract is used for tokens distribution.

RewardsTokenDistribution constructor sets:

- **token** to **_token**
- **vestingVault** to **_vestingVault**
- **finished** to **false**

RewardsTokenDistribution has 1 modifier:

- **isAllowed** - checks whether **finished** is **false**.

RewardsTokenDistribution has 6 functions:

- **allocNormalUser** is a public function - mints specified amount of tokens to the specified address. Has **onlyOwner** and **isAllowed** modifiers.
- **allocNormalUsers** is a public function - mints specified amounts of tokens to the specified addresses. Has **onlyOwner** and **isAllowed** modifiers.

- **allocVestedUser** is a public function - allocates specified amount of tokens to the specified address. Has **onlyOwner** and **isAllowed** modifiers.
- **transferBackTokenOwnership** is a public function - transfers an ownership of **token** contract to the owner. Has **onlyOwner** modifier.
- **transferBackVestingVaultOwnership** is a public function - transfers an ownership of **vestingVault** contract to the owner. Has **onlyOwner** modifier.
- **finalize** is a public function - calls **finishMinting** function and finishes token distribution. Has **onlyOwner** modifier.

Findings

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

1. `mint` function in `RewardsToken` contract doesn't check for the `0x0` address. Require statement should be added (see Appendix A pic 1 for evidence).

Fixed: require statement was added.

2. Compiler version is not locked. Consider locking compiler version with the latest one (see Appendix A pic 2 for evidence).

```
pragma solidity ^0.4.24; // bad: compiles w 0.4.24 and above
pragma solidity 0.5.4; // good: compiles w 0.5.4 only
```

Fixed: Compiler version was updated and locked.

3. Self XSS on `/dist/index.html#/distribution`. Address output values are not HTML encoded within the DApp. XSS payload can be put to any of `Recipient Address` textbox and malicious script will be executed (see Appendix A pic 3 for evidence). You should HTML encode all data on the page.

Lowest / Code style / Best Practice

Best Practice

4. Contracts differ from the latest OpenZeppelin contracts.

Lowest

5. `symbol`, `name` and `hardCap` state variables in `RewardsToken` contract could be declared as `constant` to save more gas.

Fixed: constant modifier was added.

6. `burn` function could be rewritten without `burner` variable. It will save a small amount of gas.

Fixed: function was rewritten.

7. `canTransfer` require description should be rewritten to be more precise.

Fixed: require description was rewritten.

8. Comment inside the `withdraw` function should be rewritten.

Fixed: comment was rewritten.

Code style

9. Line length must be no more than 120 but it is higher in:

- `RewardsTokenDistribution` on line 81
- `VestingVault` on lines 34, 66, 81 and 151

Fixed: line length was changed.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. The contract's high-level description of functionality was presented in As-is overview section of the report.

Consultants performed all necessary security checks against web part of deployed application.

Security report contains all found security vulnerabilities and other issues in the reviewed smart contract code and web application.

Overall quality of the system is good and it doesn't have any significant security issues.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

The audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on blockchain platform. The platform, its programming language, and other software related to the smart contract can have own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

Appendix A. Evidences

Pic 1. Mint function:

```

57 function mint(address _to, uint256 _amount) public onlyOwner canMint returns (bool) {
58     require(totalSupply.add(_amount) <= hardCap);
59
60     totalSupply = totalSupply.add(_amount);
61     balances[_to] = balances[_to].add(_amount);
62     emit Transfer(address(0), _to, _amount);
63     return true;
64 }

```

Pic 2. Compiler version not locked:

```

1 pragma solidity ^0.4.24;

```

Pic 3. Self XSS in Recipient Address:

Scheduled Vesting

Recipient Address:


Total Vesting Amount: RWRDs Add List >>

Vesting Time:

+ Add Another Schedule

Vested Allocation List

Start Allocate Stop Allocate

No	Address	Amount	Type	Show Details	allocated	Remove
1		RWRDs	Default	Show Details	✖	remove

Appendix B. Automated tools reports

Pic 1. Slither automated report part 1:

```
max@ubuntu:~/Desktop/Rewards 2019-03-04/Smart Contract/contracts$ slither RewardsTokenDistribution.sol
INFO:Detectors:
Reentrancy in VestingVault.claim (VestingVault.sol#184-205):
  External calls:
    - require(bool,string)(token.balanceOf(address(this)) >= transferable,Contract Balance is insufficient) (VestingVault.sol#194-196)
  State variables written after the call(s):
    - grants (VestingVault.sol#197-201)
Reference: https://github.com/trailofbits/slither/wiki/Vulnerabilities-Description#reentrancy-vulnerabilities-1
INFO:Detectors:
RewardsTokenDistribution.allocNormalUser (RewardsTokenDistribution.sol#50-56) does not use the value returned by external calls:
  - token.mint(_to,_value) (RewardsTokenDistribution.sol#52)
Reference: https://github.com/trailofbits/slither/wiki/Vulnerabilities-Description#unused-return
INFO:Detectors:
RewardsTokenDistribution.allocVestedUser (RewardsTokenDistribution.sol#67-75) does not use the value returned by external calls:
  - token.mint(address(vestingVault),_value) (RewardsTokenDistribution.sol#71)
Reference: https://github.com/trailofbits/slither/wiki/Vulnerabilities-Description#unused-return
INFO:Detectors:
RewardsTokenDistribution.allocNormalUsers (RewardsTokenDistribution.sol#79-89) does not use the value returned by external calls:
  - token.mint(_holders[i],_amounts[i]) (RewardsTokenDistribution.sol#84)
Reference: https://github.com/trailofbits/slither/wiki/Vulnerabilities-Description#unused-return
INFO:Detectors:
RewardsTokenDistribution.finalize (RewardsTokenDistribution.sol#108-110) does not use the value returned by external calls:
  - token.finishMinting() (RewardsTokenDistribution.sol)
Reference: https://github.com/trailofbits/slither/wiki/Vulnerabilities-Description#unused-return
INFO:Detectors:
VestingVault.claim (VestingVault.sol#184-205) does not use the value returned by external calls:
  - token.transfer(beneficiary,transferable) (VestingVault.sol#203-204)
Reference: https://github.com/trailofbits/slither/wiki/Vulnerabilities-Description#unused-return
INFO:Detectors:
VestingVault.withdraw (VestingVault.sol#219-229) does not use the value returned by external calls:
  - token.transfer(owner,amount) (VestingVault.sol#226-227)
Reference: https://github.com/trailofbits/slither/wiki/Vulnerabilities-Description#unused-return
INFO:Detectors:
VestingVault.grant (VestingVault.sol#84-126) uses a dangerous strict equality:
  - require(bool,string)(grants[_to].value == 0,Already added to vesting vault)
Reference: https://github.com/trailofbits/slither/wiki/Vulnerabilities-Description#dangerous-strict-equalities
```

Pic 2. Slither automated report part 2:

```
INFO:Detectors:
Reentrancy in VestingVault.claim (VestingVault.sol#184-205):
  External calls:
    - require(bool,string)(token.balanceOf(address(this)) >= transferable,Contract Balance is insufficient) (VestingVault.sol#194-196)
  External calls sending eth:
  State variables written after the call(s):
    - totalVestedTokens (VestingVault.sol#201-203)
Reference: https://github.com/trailofbits/slither/wiki/Vulnerabilities-Description#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in RewardsTokenDistribution.finalize (RewardsTokenDistribution.sol#108-110):
  External calls:
    - token.finishMinting() (RewardsTokenDistribution.sol)
  External calls sending eth:
  State variables written after the call(s):
    - finished (RewardsTokenDistribution.sol)
Reference: https://github.com/trailofbits/slither/wiki/Vulnerabilities-Description#reentrancy-vulnerabilities-2
INFO:Detectors:
RewardsTokenDistribution.allocNormalUsers has external calls inside a loop:
  - token.mint(_holders[i],_amounts[i]) (RewardsTokenDistribution.sol#84)
Reference: https://github.com/trailofbits/slither/wiki/Vulnerabilities-Description/\_edit#calls-inside-a-loop
INFO:Detectors:
VestingVault.returnGrantInfo.grant (local variable @ VestingVault.sol#67) shadows:
  - VestingVault.grant (function @ VestingVault.sol#84-126)
Reference: https://github.com/trailofbits/slither/wiki/Vulnerabilities-Description#local-variable-shadowing
INFO:Detectors:
Detected issues with version pragma in RewardsTokenDistribution.sol:
  - pragma solidity^0.5.4 (RewardsToken.sol#1): it allows old versions
  - pragma solidity^0.5.4 (RewardsTokenDistribution.sol#1): it allows old versions
  - pragma solidity^0.5.4 (VestingVault.sol#1): it allows old versions
  - pragma solidity^0.5.4 (lib/Ownable.sol#1): it allows old versions
  - pragma solidity^0.5.4 (lib/SafeMath.sol#1): it allows old versions
Reference: https://github.com/trailofbits/slither/wiki/Vulnerabilities-Description#incorrect-version-of-solidity
INFO:Detectors:
Parameter '_to' of RewardsToken.mint (RewardsToken.sol#57) is not in mixedCase
Parameter '_amount' of RewardsToken.mint (RewardsToken.sol#57-58) is not in mixedCase
Parameter '_to' of RewardsToken.transfer (RewardsToken.sol#95) is not in mixedCase
Parameter '_value' of RewardsToken.transfer (RewardsToken.sol#95) is not in mixedCase
Parameter '_from' of RewardsToken.transferFrom (RewardsToken.sol#112) is not in mixedCase
Parameter '_to' of RewardsToken.transferFrom (RewardsToken.sol#112) is not in mixedCase
Parameter '_value' of RewardsToken.transferFrom (RewardsToken.sol#112-113) is not in mixedCase
Parameter '_spender' of RewardsToken.approve (RewardsToken.sol#135) is not in mixedCase
Parameter '_value' of RewardsToken.approve (RewardsToken.sol#135-136) is not in mixedCase
Parameter '_owner' of RewardsToken.allowance (RewardsToken.sol#150) is not in mixedCase
Parameter '_spender' of RewardsToken.allowance (RewardsToken.sol#150) is not in mixedCase
Parameter '_spender' of RewardsToken.increaseApproval (RewardsToken.sol#156-157) is not in mixedCase
```

Pic 3. Slither automated report part 3:

```

Parameter '_addedValue' of RewardsToken.increaseApproval (RewardsToken.sol#157) is not in mixedCase
Parameter '_spender' of RewardsToken.decreaseApproval (RewardsToken.sol#171-172) is not in mixedCase
Parameter '_subtractedValue' of RewardsToken.decreaseApproval (RewardsToken.sol#172) is not in mixedCase
Parameter '_owner' of RewardsToken.balanceOf (RewardsToken.sol#191) is not in mixedCase
Parameter '_burnAmount' of RewardsToken.burn (RewardsToken.sol#197-198) is not in mixedCase
Parameter '_from' of RewardsToken.revoke (RewardsToken.sol#209) is not in mixedCase
Parameter '_value' of RewardsToken.revoke (RewardsToken.sol#209) is not in mixedCase
Constant 'RewardsToken.hardCap' (RewardsToken.sol#18) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter '_token' of RewardsTokenDistribution. (RewardsTokenDistribution.sol#35) is not in mixedCase
Parameter '_vestingVault' of RewardsTokenDistribution. (RewardsTokenDistribution.sol#36-37) is not in mixedCase
Parameter '_to' of RewardsTokenDistribution.allocNormalUser (RewardsTokenDistribution.sol#50-51) is not in mixedCase
Parameter '_value' of RewardsTokenDistribution.allocNormalUser (RewardsTokenDistribution.sol#51) is not in mixedCase
Parameter '_to' of RewardsTokenDistribution.allocVestedUser (RewardsTokenDistribution.sol#67) is not in mixedCase
Parameter '_value' of RewardsTokenDistribution.allocVestedUser (RewardsTokenDistribution.sol#67) is not in mixedCase
Parameter '_start' of RewardsTokenDistribution.allocVestedUser (RewardsTokenDistribution.sol#67-68) is not in mixedCase
Parameter '_duration' of RewardsTokenDistribution.allocVestedUser (RewardsTokenDistribution.sol#68) is not in mixedCase
Parameter '_cliff' of RewardsTokenDistribution.allocVestedUser (RewardsTokenDistribution.sol#68) is not in mixedCase
Parameter '_scheduleTimes' of RewardsTokenDistribution.allocVestedUser (RewardsTokenDistribution.sol#68) is not in mixedCase
Parameter '_scheduleValues' of RewardsTokenDistribution.allocVestedUser (RewardsTokenDistribution.sol#68-69) is not in mixedCase
Parameter '_level' of RewardsTokenDistribution.allocVestedUser (RewardsTokenDistribution.sol#69) is not in mixedCase
Parameter '_holders' of RewardsTokenDistribution.allocNormalUsers (RewardsTokenDistribution.sol#79-80) is not in mixedCase
Parameter '_amounts' of RewardsTokenDistribution.allocNormalUsers (RewardsTokenDistribution.sol#80) is not in mixedCase
Parameter '_token' of VestingVault. (VestingVault.sol#46) is not in mixedCase
Parameter '_user' of VestingVault.returnGrantInfo (VestingVault.sol#63) is not in mixedCase
Parameter '_to' of VestingVault.grant (VestingVault.sol#84) is not in mixedCase
Parameter '_value' of VestingVault.grant (VestingVault.sol#84-85) is not in mixedCase
Parameter '_start' of VestingVault.grant (VestingVault.sol#85) is not in mixedCase
Parameter '_duration' of VestingVault.grant (VestingVault.sol#85) is not in mixedCase
Parameter '_cliff' of VestingVault.grant (VestingVault.sol#85) is not in mixedCase
Parameter '_scheduleTimes' of VestingVault.grant (VestingVault.sol#85) is not in mixedCase
Parameter '_scheduleValues' of VestingVault.grant (VestingVault.sol#85-86) is not in mixedCase
Parameter '_level' of VestingVault.grant (VestingVault.sol#86) is not in mixedCase
Parameter '_holder' of VestingVault.transferableTokens (VestingVault.sol#133) is not in mixedCase
Parameter '_time' of VestingVault.transferableTokens (VestingVault.sol#133-134) is not in mixedCase
Parameter '_grant' of VestingVault.calculateTransferableTokens (VestingVault.sol#147-149) is not in mixedCase
Parameter '_time' of VestingVault.calculateTransferableTokens (VestingVault.sol#149) is not in mixedCase
Parameter '_a' of SafeMath.mul (lib/SafeMath.sol#13) is not in mixedCase
Parameter '_b' of SafeMath.mul (lib/SafeMath.sol#13) is not in mixedCase
Parameter '_a' of SafeMath.div (lib/SafeMath.sol#30) is not in mixedCase
Parameter '_b' of SafeMath.div (lib/SafeMath.sol#30) is not in mixedCase
Parameter '_a' of SafeMath.sub (lib/SafeMath.sol#41) is not in mixedCase
Parameter '_b' of SafeMath.sub (lib/SafeMath.sol#41-42) is not in mixedCase
Parameter '_a' of SafeMath.add (lib/SafeMath.sol#51-52) is not in mixedCase
Parameter '_b' of SafeMath.add (lib/SafeMath.sol#52) is not in mixedCase
Reference: https://github.com/trailofbits/slither/wiki/Vulnerabilities-Description#conformance-to-solidity-naming-conventions

```

Pic 4. Slither automated report part 4:

```

INFO:Detectors:
RewardsToken.mint (RewardsToken.sol#57-69) should be declared external
RewardsToken.finishMinting (RewardsToken.sol#72-80) should be declared external
RewardsToken.startMinting (RewardsToken.sol#83-91) should be declared external
RewardsToken.transfer (RewardsToken.sol#95-107) should be declared external
RewardsToken.transferFrom (RewardsToken.sol#112-125) should be declared external
RewardsToken.approve (RewardsToken.sol#135-141) should be declared external
RewardsToken.allowance (RewardsToken.sol#149-151) should be declared external
RewardsToken.increaseApproval (RewardsToken.sol#156-163) should be declared external
RewardsToken.decreaseApproval (RewardsToken.sol#171-183) should be declared external
RewardsToken.balanceOf (RewardsToken.sol#191-194) should be declared external
RewardsToken.burn (RewardsToken.sol#197-206) should be declared external
RewardsToken.revoke (RewardsToken.sol#209-231) should be declared external
RewardsToken.freeze (RewardsToken.sol#232-236) should be declared external
RewardsToken.unfreeze (RewardsToken.sol) should be declared external
RewardsTokenDistribution.allocNormalUser (RewardsTokenDistribution.sol#50-56) should be declared external
RewardsTokenDistribution.allocVestedUser (RewardsTokenDistribution.sol#67-75) should be declared external
RewardsTokenDistribution.allocNormalUsers (RewardsTokenDistribution.sol#79-89) should be declared external
RewardsTokenDistribution.transferBackTokenOwnership (RewardsTokenDistribution.sol#92-96) should be declared external
RewardsTokenDistribution.transferBackVestingVaultOwnership (RewardsTokenDistribution.sol#99-105) should be declared external
RewardsTokenDistribution.finalize (RewardsTokenDistribution.sol#108-110) should be declared external
VestingVault.returnVestedAddresses (VestingVault.sol#55-60) should be declared external
VestingVault.returnGrantInfo (VestingVault.sol#63-73) should be declared external
VestingVault.grant (VestingVault.sol#84-126) should be declared external
VestingVault.transferableTokens (VestingVault.sol#131-141) should be declared external
VestingVault.claim (VestingVault.sol#184-205) should be declared external
VestingVault.burnRemainingTokens (VestingVault.sol#208-216) should be declared external
VestingVault.withdraw (VestingVault.sol#219-229) should be declared external
VestingVault.lockVault (VestingVault.sol) should be declared external
Ownable.transferOwnership (lib/Ownable.sol#37-43) should be declared external
Reference: https://github.com/trailofbits/slither/wiki/Vulnerabilities-Description#public-function-that-could-be-declared-as-external
INFO:Slither:RewardsTokenDistribution.sol analyzed (5 contracts), 98 result(s) found
max@ubuntu:~/Desktop/Rewards 2019-03-04/Smart Contract/contracts$

```

Pic 5. Remix IDE automated report part 1:

Static Analysis raised 42 warning(s) that requires your attention. Click here to show the warning(s).	✕
Ownable	✕
RewardsToken	✕
RewardsTokenDistribution	✕
SafeMath	✕
VestingVault	✕

Pic 6. Remix IDE automated report part 2:

Potential Violation of Checks-Effects-Interaction pattern in RewardsTokenDistribution.allocNormalUser(address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis. more	✕
Potential Violation of Checks-Effects-Interaction pattern in RewardsTokenDistribution.allocVestedUser(address,uint256,uint256,uint256,uint256,uint256,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis. more	✕
Potential Violation of Checks-Effects-Interaction pattern in RewardsTokenDistribution.allocNormalUsers(address,uint256[]): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis. more	✕
Potential Violation of Checks-Effects-Interaction pattern in RewardsTokenDistribution.finalize(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis. more	✕
Potential Violation of Checks-Effects-Interaction pattern in VestingVault.claim(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis. more	✕
Potential Violation of Checks-Effects-Interaction pattern in VestingVault.burnRemainingTokens(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis. more	✕
Potential Violation of Checks-Effects-Interaction pattern in VestingVault.withdraw(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis. more	✕
browser/VestingVault.sol:186:65:use of "now": "now" does not mean current time. Now is an alias for block.timestamp. Block.timestamp can be influenced by miners to a certain degree, be careful. more	✕
Gas requirement of function RewardsToken.burn(uint256) high: infinite. If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)	✕

Pic 9. Remix IDE automated report part 5:



Pic 10. Remix IDE automated report part 6:

