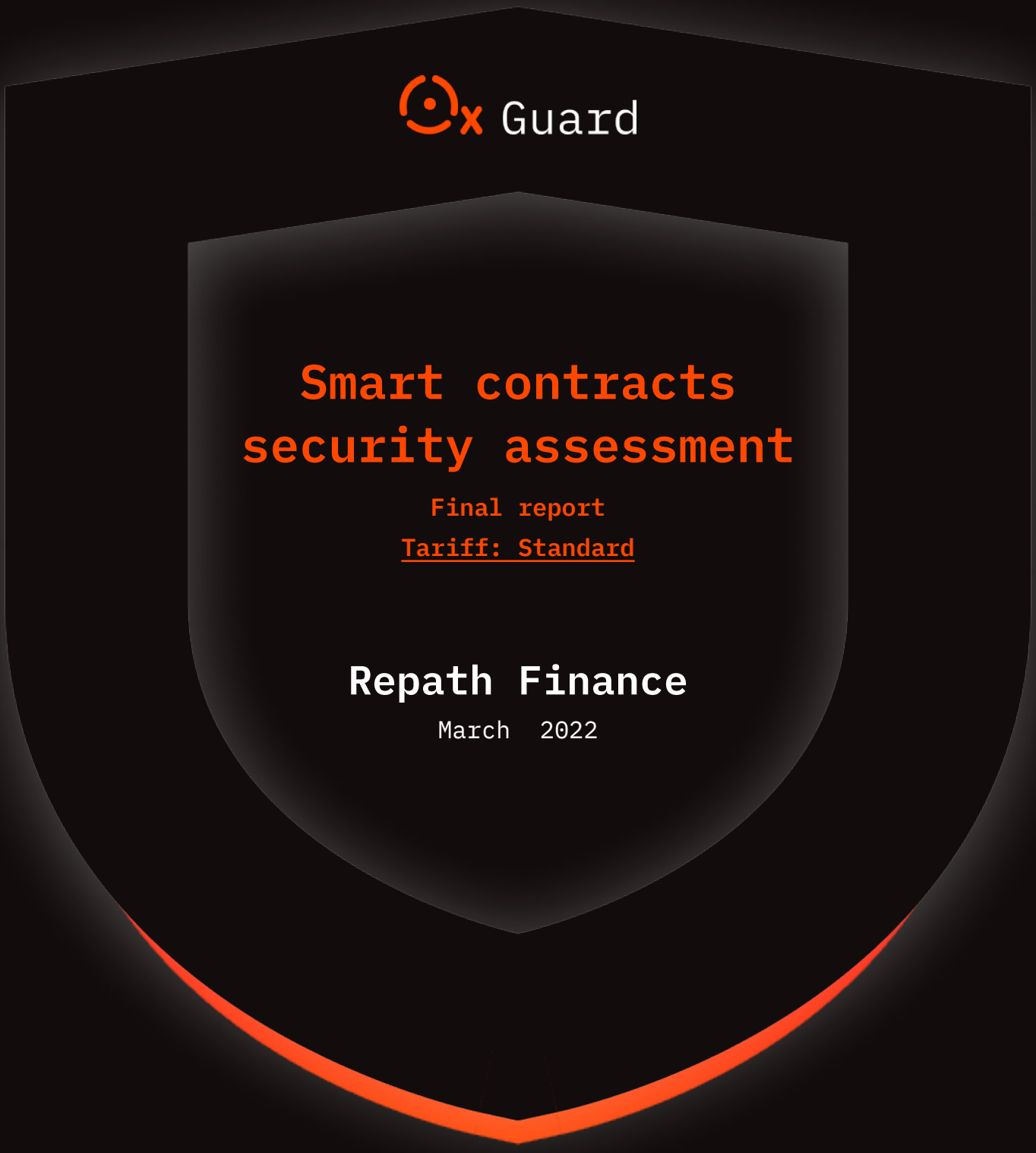

Guard

Smart contracts security assessment

Final report

Tariff: Standard

Repath Finance

March 2022

0xguard.com

hello@0xguard.com

Contents

Introduction

This report has been prepared for the Repath Finance team upon their request.

The audited project is a fork of the Tomb Finance Project. The code is available in the Github repository. The code was checked in 8ef8667 commit.

The purpose of this audit was to ensure that no issues were introduced with the changes to the original code and that known vulnerabilities (e.g. circumventing the protocol's fee system) are fixed prior to deployment.

This project has a problem with the type of ownership in contracts, the Tomb project has the same problem.

Further details about Repath Finance are available at the official website: https://repath.finance/

Name	Repath Finance
Audit date	2022-03-21 - 2022-03-23
Language	Solidity
Platform	Metis

Contracts checked

Name	Address
DevFund	https://github.com/repathfinance/contracts/blob/8ef8667e4a87636a489df7ddbe1669976bf7f8ed/DevFund.sol
GenesisPool	https://github.com/repathfinance/contracts/blob/8ef8667e4a87636a489df7ddbe1669976bf7f8ed/GenesisPool.sol

Masonry	https://github.com/repathfinance/contracts/blob/8ef8667e4a87636a489df7ddbe1669976bf7f8ed/Masonry.sol
Oracle	https://github.com/repathfinance/contracts/blob/8ef8667e4a87636a489df7ddbe1669976bf7f8ed/Oracle.sol
RePATH	https://github.com/repathfinance/contracts/blob/8ef8667e4a87636a489df7ddbe1669976bf7f8ed/RePATH.sol
Rebond	https://github.com/repathfinance/contracts/blob/8ef8667e4a87636a489df7ddbe1669976bf7f8ed/Rebond.sol
Rept	https://github.com/repathfinance/contracts/blob/8ef8667e4a87636a489df7ddbe1669976bf7f8ed/Rept.sol
RewardPool	https://github.com/repathfinance/contracts/blob/8ef8667e4a87636a489df7ddbe1669976bf7f8ed/RewardPool.sol
Timelock	https://github.com/repathfinance/contracts/blob/8ef8667e4a87636a489df7ddbe1669976bf7f8ed/Timelock.sol
Treasury	https://github.com/repathfinance/contracts/blob/8ef8667e4a87636a489df7ddbe1669976bf7f8ed/Treasury.sol
Multiple contracts	

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Comparing the project to the Tomb Finance implementation

Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

No issues were found

Medium severity issues

No issues were found

Low severity issues

1. Contract ownership (Treasury)

The Operator role can change the addresses of funds in the Treasury contract using the function setExtraFunds() function. The daoFund can be withdrawn if the operator account is compromised.

Recommendation: There are a large number of functions with the onlyOperator() modifier, there is a possibility that the operator can be compromised. It is recommended to create multiple roles for different kinds of functions to reduce the operator's problem. It is also recommended to add a time delay to the especially important set functions using the TimelockController. We also recommend that you look through the entire codebase to find functions that are dangerous for you as the owner of the project (mainly set functions), if there are any, then add a call to them via multisig wallet. This will help avoid the issue of owner compromise.

2. Typos (Treasury)

Typos reduce the code's readability.

1) 1171L typo in function name "sereBondDepletionFloorPercent"

2) 944L, 1280L typo in event name "BoughreBonds"

Recommendation: Fix the typos.

3. Few events (Multiple contracts)

Many set functions from contracts are missing events when changing important values in the contact.

Recommendation: Create events for these set functions.

Conclusion

The Repath Finance Project was compared with the Tomb Project. Repath Finance has changed the implementation of GenesisPool and Token contracts. Added a new contract - DevFund.

In the GenesisPool contract, the deposit fees are transferred to the daoFund address.

The changed Token contract is not affected by the vulnerability that was discovered in the Tomb before because it doesn't contain the implementation of transfer with taxes.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability)set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Slither output

```
UniswapV2OracleLibrary.currentBlockTimestamp() (contracts/Oracle.sol#492-494) uses a
weak PRNG: "uint32(block.timestamp % 2 ** 32) (contracts/Oracle.sol#493)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Reentrancy in ReptGenesisRewardPool.deposit(uint256,uint256) (contracts/
GenesisPool.sol#750-774):
External calls:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#758)
- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/GenesisPool.sol#543)
- rept.safeTransfer(_to,_reptBalance) (contracts/GenesisPool.sol#812)
- rept.safeTransfer(_to,_amount) (contracts/GenesisPool.sol#814)
- (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/
GenesisPool.sol#763)
- pool.token.safeTransfer(daoFund,depositFee) (contracts/GenesisPool.sol#766)
External calls sending eth:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#758)
- (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
State variables written after the call(s):
- user.amount = user.amount.add(_amount).sub(depositFee) (contracts/
GenesisPool.sol#767)
- user.rewardDebt = user.amount.mul(pool.accReptPerShare).div(1e18) (contracts/
GenesisPool.sol#772)
Reentrancy in ReptGenesisRewardPool.deposit(uint256,uint256) (contracts/
GenesisPool.sol#750-774):
External calls:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#758)
- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/GenesisPool.sol#543)
- rept.safeTransfer(_to,_reptBalance) (contracts/GenesisPool.sol#812)
- rept.safeTransfer(_to,_amount) (contracts/GenesisPool.sol#814)
- (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/
GenesisPool.sol#763)
```

```
External calls sending eth:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#758)
- (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
State variables written after the call(s):
- user.amount = user.amount.add(_amount) (contracts/GenesisPool.sol#769)
Reentrancy in ReptGenesisRewardPool.withdraw(uint256,uint256) (contracts/
GenesisPool.sol#777-794):
External calls:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#785)
- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/GenesisPool.sol#543)
- rept.safeTransfer(_to,_reptBalance) (contracts/GenesisPool.sol#812)
- rept.safeTransfer(_to,_amount) (contracts/GenesisPool.sol#814)
- (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
External calls sending eth:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#785)
- (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
State variables written after the call(s):
- user.amount = user.amount.sub(_amount) (contracts/GenesisPool.sol#789)
Reentrancy in ReptGenesisRewardPool.withdraw(uint256,uint256) (contracts/
GenesisPool.sol#777-794):
External calls:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#785)
- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/GenesisPool.sol#543)
- rept.safeTransfer(_to,_reptBalance) (contracts/GenesisPool.sol#812)
- rept.safeTransfer(_to,_amount) (contracts/GenesisPool.sol#814)
- (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
- pool.token.safeTransfer(_sender,_amount) (contracts/GenesisPool.sol#790)
External calls sending eth:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#785)
- (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
State variables written after the call(s):
- user.rewardDebt = user.amount.mul(pool.accReptPerShare).div(1e18) (contracts/
GenesisPool.sol#792)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities
```

```
Reentrancy in Masonry.stake(uint256) (contracts/Masonry.sol#795-800):
	External calls:
	- super.stake(amount) (contracts/Masonry.sol#797)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/Masonry.sol#406)
		- share.safeTransferFrom(msg.sender,address(this),amount) (contracts/Masonry.sol#624)
		- (success,returndata) = target.call{value: value}(data) (contracts/Masonry.sol#530)
	External calls sending eth:
	- super.stake(amount) (contracts/Masonry.sol#797)
		- (success,returndata) = target.call{value: value}(data) (contracts/Masonry.sol#530)
	State variables written after the call(s):
	- masons[msg.sender].epochTimerStart = treasury.epoch() (contracts/Masonry.sol#798)
Reentrancy in Masonry.withdraw(uint256) (contracts/Masonry.sol#802-808):
	External calls:
	- claimReward() (contracts/Masonry.sol#805)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/Masonry.sol#406)
		- (success,returndata) = target.call{value: value}(data) (contracts/Masonry.sol#530)
		- rept.safeTransfer(msg.sender,reward) (contracts/Masonry.sol#820)
	- super.withdraw(amount) (contracts/Masonry.sol#806)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/Masonry.sol#406)
		- (success,returndata) = target.call{value: value}(data) (contracts/Masonry.sol#530)
		- share.safeTransfer(msg.sender,amount) (contracts/Masonry.sol#632)
	External calls sending eth:
	- claimReward() (contracts/Masonry.sol#805)
		- (success,returndata) = target.call{value: value}(data) (contracts/Masonry.sol#530)
	- super.withdraw(amount) (contracts/Masonry.sol#806)
		- (success,returndata) = target.call{value: value}(data) (contracts/Masonry.sol#530)
	State variables written after the call(s):
	- super.withdraw(amount) (contracts/Masonry.sol#806)
		- _balances[msg.sender] = masonShare.sub(amount) (contracts/Masonry.sol#631)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities

Reentrancy in RePathRewardPool.deposit(uint256,uint256) (contracts/
RewardPool.sol#738-756):
	External calls:
	- safeRePathTransfer(_sender,_pending) (contracts/RewardPool.sol#746)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
```

```
(contracts/RewardPool.sol#542)
        - repath.safeTransfer(_to,_repathBal) (contracts/RewardPool.sol#794)
        - repath.safeTransfer(_to,_amount) (contracts/RewardPool.sol#796)
        - (success,returndata) = target.call{value: value}(data) (contracts/
RewardPool.sol#119)
        - pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/
RewardPool.sol#751)
        External calls sending eth:
        - safeRePathTransfer(_sender,_pending) (contracts/RewardPool.sol#746)
        - (success,returndata) = target.call{value: value}(data) (contracts/
RewardPool.sol#119)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount) (contracts/RewardPool.sol#752)
        - user.rewardDebt = user.amount.mul(pool.accRePathPerShare).div(1e18) (contracts/
RewardPool.sol#754)
Reentrancy in RePathRewardPool.withdraw(uint256,uint256) (contracts/
RewardPool.sol#759-776):
        External calls:
        - safeRePathTransfer(_sender,_pending) (contracts/RewardPool.sol#767)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/RewardPool.sol#542)
        - repath.safeTransfer(_to,_repathBal) (contracts/RewardPool.sol#794)
        - repath.safeTransfer(_to,_amount) (contracts/RewardPool.sol#796)
        - (success,returndata) = target.call{value: value}(data) (contracts/
RewardPool.sol#119)
        External calls sending eth:
        - safeRePathTransfer(_sender,_pending) (contracts/RewardPool.sol#767)
        - (success,returndata) = target.call{value: value}(data) (contracts/
RewardPool.sol#119)
        State variables written after the call(s):
        - user.amount = user.amount.sub(_amount) (contracts/RewardPool.sol#771)
Reentrancy in RePathRewardPool.withdraw(uint256,uint256) (contracts/
RewardPool.sol#759-776):
        External calls:
        - safeRePathTransfer(_sender,_pending) (contracts/RewardPool.sol#767)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/RewardPool.sol#542)
        - repath.safeTransfer(_to,_repathBal) (contracts/RewardPool.sol#794)
        - repath.safeTransfer(_to,_amount) (contracts/RewardPool.sol#796)
        - (success,returndata) = target.call{value: value}(data) (contracts/
RewardPool.sol#119)
```

```
- pool.token.safeTransfer(_sender,_amount) (contracts/RewardPool.sol#772)
External calls sending eth:
- safeRePathTransfer(_sender,_pending) (contracts/RewardPool.sol#767)
  - (success,returndata) = target.call{value: value}(data) (contracts/
RewardPool.sol#119)
State variables written after the call(s):
- user.rewardDebt = user.amount.mul(pool.accRePathPerShare).div(1e18) (contracts/
RewardPool.sol#774)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities

Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#1344-1384):
External calls:
- _updateReptPrice() (contracts/Treasury.sol#1345)
  - IOracle(reptOracle).update() (contracts/Treasury.sol#1241)
- _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1375)
  - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/Treasury.sol#860)
  - IBasisAsset(rept).mint(address(this),_amount) (contracts/Treasury.sol#1310)
  - (success,returndata) = target.call{value: value}(data) (contracts/Treasury.sol#142)
  - IERC20(rept).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#1315)
  - IERC20(rept).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#1322)
  - IERC20(rept).safeApprove(masonry,0) (contracts/Treasury.sol#1328)
  - IERC20(rept).safeApprove(masonry,_amount) (contracts/Treasury.sol#1329)
  - IMasonry(masonry).allocateSeigniorage(_amount) (contracts/Treasury.sol#1330)
External calls sending eth:
- _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1375)
  - (success,returndata) = target.call{value: value}(data) (contracts/Treasury.sol#142)
State variables written after the call(s):
- seigniorageSaved = seigniorageSaved.add(_savedForBond) (contracts/Treasury.sol#1378)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities

DevFund.distribute() (contracts/DevFund.sol#218-228) ignores return value by
token.transfer(allocations[a].account,balance * allocations[a].points / totalPoints)
(contracts/DevFund.sol#224)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer

RePath.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/
RePATH.sol#855-861) ignores return value by _token.transfer(_to,_amount) (contracts/
RePATH.sol#860)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer

Rept.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/
Rept.sol#1030-1036) ignores return value by _token.transfer(_to,_amount) (contracts/
Rept.sol#1035)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer

Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#1309-1332) ignores return
value by IERC20(rept).transfer(daoFund,_daoFundSharedAmount) (contracts/
Treasury.sol#1315)
Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#1309-1332) ignores return
value by IERC20(rept).transfer(devFund,_devFundSharedAmount) (contracts/
Treasury.sol#1322)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer

ReptGenesisRewardPool.pendingREPT(uint256,address) (contracts/GenesisPool.sol#705-716)
performs a multiplication on the result of a division:
-_reptReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/
GenesisPool.sol#712)
-accReptPerShare = accReptPerShare.add(_reptReward.mul(1e18).div(tokenSupply))
(contracts/GenesisPool.sol#713)
ReptGenesisRewardPool.updatePool(uint256) (contracts/GenesisPool.sol#727-747) performs
a multiplication on the result of a division:
-_reptReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/
GenesisPool.sol#743)
-pool.accReptPerShare =
pool.accReptPerShare.add(_reptReward.mul(1e18).div(tokenSupply)) (contracts/
GenesisPool.sol#744)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

RePathRewardPool.pendingShare(uint256,address) (contracts/RewardPool.sol#693-704)
performs a multiplication on the result of a division:
-_repathReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/
RewardPool.sol#700)
-accRePathPerShare = accRePathPerShare.add(_repathReward.mul(1e18).div(tokenSupply))
(contracts/RewardPool.sol#701)
RePathRewardPool.updatePool(uint256) (contracts/RewardPool.sol#715-735) performs a
```

```
multiplication on the result of a division:
-_repathReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/
RewardPool.sol#731)
-pool.accRePathPerShare =
pool.accRePathPerShare.add(_repathReward.mul(1e18).div(tokenSupply)) (contracts/
RewardPool.sol#732)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

Treasury.allocateSeigniorage() (contracts/Treasury.sol#1344-1384) performs a
multiplication on the result of a division:
-_seigniorage = reptSupply.mul(_percentage).div(1e18) (contracts/Treasury.sol#1367)
-_savedForMasonry = _seigniorage.mul(seigniorageExpansionFloorPercent).div(10000)
(contracts/Treasury.sol#1368)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

ReptGenesisRewardPool.updatePool(uint256) (contracts/GenesisPool.sol#727-747) uses a
dangerous strict equality:
- tokenSupply == 0 (contracts/GenesisPool.sol#733)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities

RePathRewardPool.updatePool(uint256) (contracts/RewardPool.sol#715-735) uses a
dangerous strict equality:
- tokenSupply == 0 (contracts/RewardPool.sol#721)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities

Reentrancy in Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#1254-1281):
External calls:
- IBasisAsset(rept).burnFrom(msg.sender,_reptAmount) (contracts/Treasury.sol#1274)
- IBasisAsset(rebond).mint(msg.sender,_bondAmount) (contracts/Treasury.sol#1275)
State variables written after the call(s):
- epochSupplyContractionLeft = epochSupplyContractionLeft.sub(_reptAmount) (contracts/
Treasury.sol#1277)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1

Treasury.setSupplyTiersEntry(uint8,uint256) (contracts/Treasury.sol#1150-1161) contains
a tautology or contradiction:
```

```
- require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/
Treasury.sol#1151)
Treasury.setMaxExpansionTiersEntry(uint8,uint256) (contracts/Treasury.sol#1163-1169)
contains a tautology or contradiction:
- require(bool,string)(_index >= 0,Index has to be higher than 0) (contracts/
Treasury.sol#1164)
Treasury._calculateMaxSupplyExpansionPercent(uint256) (contracts/
Treasury.sol#1334-1342) contains a tautology or contradiction:
- tierId >= 0 (contracts/Treasury.sol#1335)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-
contradiction

DevFund.distribute().t (contracts/DevFund.sol#219) is a local variable never
initialized
DevFund.distribute().a (contracts/DevFund.sol#223) is a local variable never
initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
local-variables

FixedPoint.mul(FixedPoint.uq112x112,uint256).z (contracts/Oracle.sol#451) is a local
variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
local-variables

Treasury.getReptPrice().price (contracts/Treasury.sol#1002) is a local variable never
initialized
Treasury.allocateSeigniorage()._savedForBond (contracts/Treasury.sol#1356) is a local
variable never initialized
Treasury.getReptUpdatedPrice().price (contracts/Treasury.sol#1010) is a local variable
never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
local-variables

Treasury.getReptPrice() (contracts/Treasury.sol#1001-1007) ignores return value by
IOracle(reptOracle).consult(rept,1e18) (contracts/Treasury.sol#1002-1006)
Treasury.getReptUpdatedPrice() (contracts/Treasury.sol#1009-1015) ignores return value
by IOracle(reptOracle).twap(rept,1e18) (contracts/Treasury.sol#1010-1014)
Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#1254-1281) ignores return
value by IBasisAsset(rebond).mint(msg.sender,_bondAmount) (contracts/Treasury.sol#1275)
Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#1309-1332) ignores return
value by IBasisAsset(rept).mint(address(this),_amount) (contracts/Treasury.sol#1310)
```

```
Treasury.allocateSeigniorage() (contracts/Treasury.sol#1344-1384) ignores return value
by IBasisAsset(rept).mint(address(this),_savedForBond) (contracts/Treasury.sol#1379)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

ReptGenesisRewardPool.setOperator(address) (contracts/GenesisPool.sol#819-821) should
emit an event for:
- operator = _operator (contracts/GenesisPool.sol#820)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
access-control

Masonry.setOperator(address) (contracts/Masonry.sol#730-732) should emit an event for:
- operator = _operator (contracts/Masonry.sol#731)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
access-control

RePathRewardPool.setOperator(address) (contracts/RewardPool.sol#801-803) should emit an
event for:
- operator = _operator (contracts/RewardPool.sol#802)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
access-control

Treasury.setOperator(address) (contracts/Treasury.sol#1128-1130) should emit an event
for:
- operator = _operator (contracts/Treasury.sol#1129)
Treasury.setMasonry(address) (contracts/Treasury.sol#1132-1134) should emit an event
for:
- masonry = _masonry (contracts/Treasury.sol#1133)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
access-control

DevFund.addAllocation(address,uint256) (contracts/DevFund.sol#230-236) should emit an
event for:
- totalPoints += points (contracts/DevFund.sol#235)
DevFund.setAllocationPoints(address,uint256) (contracts/DevFund.sol#249-256) should
emit an event for:
- totalPoints = totalPoints - allocations[a].points + points (contracts/
DevFund.sol#252)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

ReptGenesisRewardPool.add(uint256,IERC20,bool,uint256) (contracts/
```

```
GenesisPool.sol#638-676) should emit an event for:
	- totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/GenesisPool.sol#674)
ReptGenesisRewardPool.set(uint256,uint256) (contracts/GenesisPool.sol#679-688) should
emit an event for:
	- totalAllocPoint = totalAllocPoint.sub(pool.allocPoint).add(_allocPoint) (contracts/
GenesisPool.sol#683-685)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

Masonry.setLockUp(uint256,uint256) (contracts/Masonry.sol#734-738) should emit an event
for:
	- withdrawLockupEpochs = _withdrawLockupEpochs (contracts/Masonry.sol#736)
	- rewardLockupEpochs = _rewardLockupEpochs (contracts/Masonry.sol#737)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

RePathRewardPool.add(uint256,IERC20,bool,uint256) (contracts/RewardPool.sol#626-664)
should emit an event for:
	- totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/RewardPool.sol#662)
RePathRewardPool.set(uint256,uint256) (contracts/RewardPool.sol#667-676) should emit an
event for:
	- totalAllocPoint = totalAllocPoint.sub(pool.allocPoint).add(_allocPoint) (contracts/
RewardPool.sol#671-673)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

Treasury.setReptPriceCeiling(uint256) (contracts/Treasury.sol#1140-1143) should emit an
event for:
	- reptPriceCeiling = _reptPriceCeiling (contracts/Treasury.sol#1142)
Treasury.setMaxSupplyExpansionPercents(uint256) (contracts/Treasury.sol#1145-1148)
should emit an event for:
	- maxSupplyExpansionPercent = _maxSupplyExpansionPercent (contracts/
Treasury.sol#1147)
Treasury.sereBondDepletionFloorPercent(uint256) (contracts/Treasury.sol#1171-1174)
should emit an event for:
	- bondDepletionFloorPercent = _bondDepletionFloorPercent (contracts/
Treasury.sol#1173)
Treasury.setMaxDebtRatioPercent(uint256) (contracts/Treasury.sol#1181-1184) should emit
an event for:
	- maxDebtRatioPercent = _maxDebtRatioPercent (contracts/Treasury.sol#1183)
Treasury.setBootstrap(uint256,uint256) (contracts/Treasury.sol#1186-1191) should emit
```

```
an event for:
	- bootstrapEpochs = _bootstrapEpochs (contracts/Treasury.sol#1189)
	- bootstrapSupplyExpansionPercent = _bootstrapSupplyExpansionPercent (contracts/
Treasury.sol#1190)
Treasury.setExtraFunds(address,uint256,address,uint256) (contracts/
Treasury.sol#1193-1207) should emit an event for:
	- daoFundSharedPercent = _daoFundSharedPercent (contracts/Treasury.sol#1204)
	- devFundSharedPercent = _devFundSharedPercent (contracts/Treasury.sol#1206)
Treasury.setMaxDiscountRate(uint256) (contracts/Treasury.sol#1209-1211) should emit an
event for:
	- maxDiscountRate = _maxDiscountRate (contracts/Treasury.sol#1210)
Treasury.setMaxPremiumRate(uint256) (contracts/Treasury.sol#1213-1215) should emit an
event for:
	- maxPremiumRate = _maxPremiumRate (contracts/Treasury.sol#1214)
Treasury.setDiscountPercent(uint256) (contracts/Treasury.sol#1217-1220) should emit an
event for:
	- discountPercent = _discountPercent (contracts/Treasury.sol#1219)
Treasury.setPremiumThreshold(uint256) (contracts/Treasury.sol#1222-1226) should emit an
event for:
	- premiumThreshold = _premiumThreshold (contracts/Treasury.sol#1225)
Treasury.setPremiumPercent(uint256) (contracts/Treasury.sol#1228-1231) should emit an
event for:
	- premiumPercent = _premiumPercent (contracts/Treasury.sol#1230)
Treasury.setMintingFactorForPayingDebt(uint256) (contracts/Treasury.sol#1233-1236)
should emit an event for:
	- mintingFactorForPayingDebt = _mintingFactorForPayingDebt (contracts/
Treasury.sol#1235)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

DevFund.call(address,uint256,bytes)._to (contracts/DevFund.sol#258) lacks a zero-check
on :
		- (success,result) = _to.call{value: _value}(_data) (contracts/DevFund.sol#259)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

ReptGenesisRewardPool.constructor(address,uint256,address)._daoFund (contracts/
GenesisPool.sol#615) lacks a zero-check on :
		- daoFund = _daoFund (contracts/GenesisPool.sol#619)
ReptGenesisRewardPool.setOperator(address)._operator (contracts/GenesisPool.sol#819)
lacks a zero-check on :
```

```
- operator = _operator (contracts/GenesisPool.sol#820)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Masonry.setOperator(address)._operator (contracts/Masonry.sol#730) lacks a zero-check on :
- operator = _operator (contracts/Masonry.sol#731)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

RePath.setTreasuryFund(address)._communityFund (contracts/RePATH.sol#800) lacks a zero-check on :
- communityFund = _communityFund (contracts/RePATH.sol#802)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

RePathRewardPool.setOperator(address)._operator (contracts/RewardPool.sol#801) lacks a zero-check on :
- operator = _operator (contracts/RewardPool.sol#802)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Timelock.constructor(address,uint256).admin_ (contracts/Timelock.sol#244) lacks a zero-check on :
- admin = admin_ (contracts/Timelock.sol#248)
Timelock.setPendingAdmin(address).pendingAdmin_ (contracts/Timelock.sol#271) lacks a zero-check on :
- pendingAdmin = pendingAdmin_ (contracts/Timelock.sol#273)
Timelock.executeTransaction(address,uint256,string,bytes,uint256).target (contracts/Timelock.sol#298) lacks a zero-check on :
- (success,returnData) = target.call.value(value)(callData) (contracts/Timelock.sol#317)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Treasury.initialize(address,address,address,address,address,uint256)._rept (contracts/Treasury.sol#1085) lacks a zero-check on :
- rept = _rept (contracts/Treasury.sol#1092)
Treasury.initialize(address,address,address,address,address,uint256)._rebond (contracts/Treasury.sol#1086) lacks a zero-check on :
- rebond = _rebond (contracts/Treasury.sol#1093)
```

```
Treasury.initialize(address,address,address,address,address,uint256)._repath (contracts/
Treasury.sol#1087) lacks a zero-check on :
	- repath = _repath (contracts/Treasury.sol#1094)
Treasury.initialize(address,address,address,address,address,uint256)._reptOracle
(contracts/Treasury.sol#1088) lacks a zero-check on :
	- reptOracle = _reptOracle (contracts/Treasury.sol#1095)
Treasury.initialize(address,address,address,address,address,uint256)._masonry
(contracts/Treasury.sol#1089) lacks a zero-check on :
	- masonry = _masonry (contracts/Treasury.sol#1096)
Treasury.setOperator(address)._operator (contracts/Treasury.sol#1128) lacks a zero-
check on :
	- operator = _operator (contracts/Treasury.sol#1129)
Treasury.setMasonry(address)._masonry (contracts/Treasury.sol#1132) lacks a zero-check
on :
	- masonry = _masonry (contracts/Treasury.sol#1133)
Treasury.setReptOracle(address)._reptOracle (contracts/Treasury.sol#1136) lacks a zero-
check on :
	- reptOracle = _reptOracle (contracts/Treasury.sol#1137)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

DevFund.distribute() (contracts/DevFund.sol#218-228) has external calls inside a loop:
balance = token.balanceOf(address(this)) (contracts/DevFund.sol#221)
DevFund.distribute() (contracts/DevFund.sol#218-228) has external calls inside a loop:
token.transfer(allocations[a].account,balance * allocations[a].points / totalPoints)
(contracts/DevFund.sol#224)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop

ReptGenesisRewardPool.updatePool(uint256) (contracts/GenesisPool.sol#727-747) has
external calls inside a loop: tokenSupply = pool.token.balanceOf(address(this))
(contracts/GenesisPool.sol#732)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop

RePathRewardPool.updatePool(uint256) (contracts/RewardPool.sol#715-735) has external
calls inside a loop: tokenSupply = pool.token.balanceOf(address(this)) (contracts/
RewardPool.sol#720)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop
```

```
Treasury.getReptCirculatingSupply() (contracts/Treasury.sol#1244-1252) has external
calls inside a loop: balanceExcluded =
balanceExcluded.add(reptErc20.balanceOf(excludedFromTotalSupply[entryId])) (contracts/
Treasury.sol#1249)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop

Variable 'Treasury.getReptPrice().price (contracts/Treasury.sol#1002)' in
Treasury.getReptPrice() (contracts/Treasury.sol#1001-1007) potentially used before
declaration: uint256(price) (contracts/Treasury.sol#1003)
Variable 'Treasury.getReptUpdatedPrice().price (contracts/Treasury.sol#1010)' in
Treasury.getReptUpdatedPrice() (contracts/Treasury.sol#1009-1015) potentially used
before declaration: uint256(price) (contracts/Treasury.sol#1011)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-
declaration-usage-of-local-variables

Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#1344-1384):
External calls:
- _updateReptPrice() (contracts/Treasury.sol#1345)
  - IOracle(reptOracle).update() (contracts/Treasury.sol#1241)
State variables written after the call(s):
- _mse = _calculateMaxSupplyExpansionPercent(reptSupply).mul(1e14) (contracts/
Treasury.sol#1358)
  - maxSupplyExpansionPercent = maxExpansionTiers[tierId] (contracts/Treasury.sol#1337)
- previousEpochReptPrice = getReptPrice() (contracts/Treasury.sol#1346)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2

Reentrancy in ReptGenesisRewardPool.deposit(uint256,uint256) (contracts/
GenesisPool.sol#750-774):
External calls:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#758)
  - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/GenesisPool.sol#543)
  - rept.safeTransfer(_to,_reptBalance) (contracts/GenesisPool.sol#812)
  - rept.safeTransfer(_to,_amount) (contracts/GenesisPool.sol#814)
  - (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
External calls sending eth:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#758)
  - (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
```

```
Event emitted after the call(s):
- RewardPaid(_sender,_pending) (contracts/GenesisPool.sol#759)
Reentrancy in ReptGenesisRewardPool.deposit(uint256,uint256) (contracts/
GenesisPool.sol#750-774):
External calls:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#758)
  - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/GenesisPool.sol#543)
  - rept.safeTransfer(_to,_reptBalance) (contracts/GenesisPool.sol#812)
  - rept.safeTransfer(_to,_amount) (contracts/GenesisPool.sol#814)
  - (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/
GenesisPool.sol#763)
- pool.token.safeTransfer(daoFund,depositFee) (contracts/GenesisPool.sol#766)
External calls sending eth:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#758)
  - (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
Event emitted after the call(s):
- Deposit(_sender,_pid,_amount) (contracts/GenesisPool.sol#773)
Reentrancy in ReptGenesisRewardPool.emergencyWithdraw(uint256) (contracts/
GenesisPool.sol#797-805):
External calls:
- pool.token.safeTransfer(msg.sender,_amount) (contracts/GenesisPool.sol#803)
Event emitted after the call(s):
- EmergencyWithdraw(msg.sender,_pid,_amount) (contracts/GenesisPool.sol#804)
Reentrancy in ReptGenesisRewardPool.withdraw(uint256,uint256) (contracts/
GenesisPool.sol#777-794):
External calls:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#785)
  - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/GenesisPool.sol#543)
  - rept.safeTransfer(_to,_reptBalance) (contracts/GenesisPool.sol#812)
  - rept.safeTransfer(_to,_amount) (contracts/GenesisPool.sol#814)
  - (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
External calls sending eth:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#785)
  - (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
```

```
Event emitted after the call(s):
- RewardPaid(_sender,_pending) (contracts/GenesisPool.sol#786)
Reentrancy in ReptGenesisRewardPool.withdraw(uint256,uint256) (contracts/
GenesisPool.sol#777-794):
External calls:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#785)
  - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/GenesisPool.sol#543)
  - rept.safeTransfer(_to,_reptBalance) (contracts/GenesisPool.sol#812)
  - rept.safeTransfer(_to,_amount) (contracts/GenesisPool.sol#814)
  - (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
- pool.token.safeTransfer(_sender,_amount) (contracts/GenesisPool.sol#790)
External calls sending eth:
- safeReptTransfer(_sender,_pending) (contracts/GenesisPool.sol#785)
  - (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
Event emitted after the call(s):
- Withdraw(_sender,_pid,_amount) (contracts/GenesisPool.sol#793)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

Reentrancy in Masonry.allocateSeigniorage(uint256) (contracts/Masonry.sol#825-842):
External calls:
- rept.safeTransferFrom(msg.sender,address(this),amount) (contracts/Masonry.sol#840)
Event emitted after the call(s):
- RewardAdded(msg.sender,amount) (contracts/Masonry.sol#841)
Reentrancy in Masonry.claimReward() (contracts/Masonry.sol#814-823):
External calls:
- rept.safeTransfer(msg.sender,reward) (contracts/Masonry.sol#820)
Event emitted after the call(s):
- RewardPaid(msg.sender,reward) (contracts/Masonry.sol#821)
Reentrancy in Masonry.stake(uint256) (contracts/Masonry.sol#795-800):
External calls:
- super.stake(amount) (contracts/Masonry.sol#797)
  - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/Masonry.sol#406)
  - share.safeTransferFrom(msg.sender,address(this),amount) (contracts/Masonry.sol#624)
  - (success,returndata) = target.call{value: value}(data) (contracts/Masonry.sol#530)
External calls sending eth:
- super.stake(amount) (contracts/Masonry.sol#797)
```

```
- (success,returndata) = target.call{value: value}(data) (contracts/Masonry.sol#530)
Event emitted after the call(s):
- Staked(msg.sender,amount) (contracts/Masonry.sol#799)
Reentrancy in Masonry.withdraw(uint256) (contracts/Masonry.sol#802-808):
External calls:
- claimReward() (contracts/Masonry.sol#805)
- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/Masonry.sol#406)
- (success,returndata) = target.call{value: value}(data) (contracts/Masonry.sol#530)
- rept.safeTransfer(msg.sender,reward) (contracts/Masonry.sol#820)
- super.withdraw(amount) (contracts/Masonry.sol#806)
- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/Masonry.sol#406)
- (success,returndata) = target.call{value: value}(data) (contracts/Masonry.sol#530)
- share.safeTransfer(msg.sender,amount) (contracts/Masonry.sol#632)
External calls sending eth:
- claimReward() (contracts/Masonry.sol#805)
- (success,returndata) = target.call{value: value}(data) (contracts/Masonry.sol#530)
- super.withdraw(amount) (contracts/Masonry.sol#806)
- (success,returndata) = target.call{value: value}(data) (contracts/Masonry.sol#530)
Event emitted after the call(s):
- Withdrawn(msg.sender,amount) (contracts/Masonry.sol#807)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

Reentrancy in RePathRewardPool.deposit(uint256,uint256) (contracts/
RewardPool.sol#738-756):
External calls:
- safeRePathTransfer(_sender,_pending) (contracts/RewardPool.sol#746)
- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/RewardPool.sol#542)
- repath.safeTransfer(_to,_repathBal) (contracts/RewardPool.sol#794)
- repath.safeTransfer(_to,_amount) (contracts/RewardPool.sol#796)
- (success,returndata) = target.call{value: value}(data) (contracts/
RewardPool.sol#119)
External calls sending eth:
- safeRePathTransfer(_sender,_pending) (contracts/RewardPool.sol#746)
- (success,returndata) = target.call{value: value}(data) (contracts/
RewardPool.sol#119)
Event emitted after the call(s):
- RewardPaid(_sender,_pending) (contracts/RewardPool.sol#747)
```

```
Reentrancy in RePathRewardPool.deposit(uint256,uint256) (contracts/
RewardPool.sol#738-756):
	External calls:
	- safeRePathTransfer(_sender,_pending) (contracts/RewardPool.sol#746)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/RewardPool.sol#542)
		- repath.safeTransfer(_to,_repathBal) (contracts/RewardPool.sol#794)
		- repath.safeTransfer(_to,_amount) (contracts/RewardPool.sol#796)
		- (success,returndata) = target.call{value: value}(data) (contracts/
RewardPool.sol#119)
	- pool.token.safeTransferFrom(_sender,address(this),_amount) (contracts/
RewardPool.sol#751)
	External calls sending eth:
	- safeRePathTransfer(_sender,_pending) (contracts/RewardPool.sol#746)
		- (success,returndata) = target.call{value: value}(data) (contracts/
RewardPool.sol#119)
	Event emitted after the call(s):
	- Deposit(_sender,_pid,_amount) (contracts/RewardPool.sol#755)
Reentrancy in RePathRewardPool.emergencyWithdraw(uint256) (contracts/
RewardPool.sol#779-787):
	External calls:
	- pool.token.safeTransfer(msg.sender,_amount) (contracts/RewardPool.sol#785)
	Event emitted after the call(s):
	- EmergencyWithdraw(msg.sender,_pid,_amount) (contracts/RewardPool.sol#786)
Reentrancy in RePathRewardPool.withdraw(uint256,uint256) (contracts/
RewardPool.sol#759-776):
	External calls:
	- safeRePathTransfer(_sender,_pending) (contracts/RewardPool.sol#767)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/RewardPool.sol#542)
		- repath.safeTransfer(_to,_repathBal) (contracts/RewardPool.sol#794)
		- repath.safeTransfer(_to,_amount) (contracts/RewardPool.sol#796)
		- (success,returndata) = target.call{value: value}(data) (contracts/
RewardPool.sol#119)
	External calls sending eth:
	- safeRePathTransfer(_sender,_pending) (contracts/RewardPool.sol#767)
		- (success,returndata) = target.call{value: value}(data) (contracts/
RewardPool.sol#119)
	Event emitted after the call(s):
	- RewardPaid(_sender,_pending) (contracts/RewardPool.sol#768)
Reentrancy in RePathRewardPool.withdraw(uint256,uint256) (contracts/
RewardPool.sol#759-776):
```

```
External calls:
- safeRePathTransfer(_sender,_pending) (contracts/RewardPool.sol#767)
  - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/RewardPool.sol#542)
  - repath.safeTransfer(_to,_repathBal) (contracts/RewardPool.sol#794)
  - repath.safeTransfer(_to,_amount) (contracts/RewardPool.sol#796)
  - (success,returndata) = target.call{value: value}(data) (contracts/
RewardPool.sol#119)
- pool.token.safeTransfer(_sender,_amount) (contracts/RewardPool.sol#772)
External calls sending eth:
- safeRePathTransfer(_sender,_pending) (contracts/RewardPool.sol#767)
  - (success,returndata) = target.call{value: value}(data) (contracts/
RewardPool.sol#119)
Event emitted after the call(s):
- Withdraw(_sender,_pid,_amount) (contracts/RewardPool.sol#775)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

Reentrancy in Timelock.executeTransaction(address,uint256,string,bytes,uint256)
(contracts/Timelock.sol#298-323):
External calls:
- (success,returnData) = target.call.value(value)(callData) (contracts/
Timelock.sol#317)
Event emitted after the call(s):
- ExecuteTransaction(txHash,target,value,signature,data,eta) (contracts/
Timelock.sol#320)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

Reentrancy in Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#1309-1332):
External calls:
- IBasisAsset(rept).mint(address(this),_amount) (contracts/Treasury.sol#1310)
- IERC20(rept).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#1315)
Event emitted after the call(s):
- DaoFundFunded(now,_daoFundSharedAmount) (contracts/Treasury.sol#1316)
Reentrancy in Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#1309-1332):
External calls:
- IBasisAsset(rept).mint(address(this),_amount) (contracts/Treasury.sol#1310)
- IERC20(rept).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#1315)
- IERC20(rept).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#1322)
Event emitted after the call(s):
```

```
- DevFundFunded(now,_devFundSharedAmount) (contracts/Treasury.sol#1323)
Reentrancy in Treasury._sendToMasonry(uint256) (contracts/Treasury.sol#1309-1332):
External calls:
- IBasisAsset(rept).mint(address(this),_amount) (contracts/Treasury.sol#1310)
- IERC20(rept).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#1315)
- IERC20(rept).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#1322)
- IERC20(rept).safeApprove(masonry,0) (contracts/Treasury.sol#1328)
- IERC20(rept).safeApprove(masonry,_amount) (contracts/Treasury.sol#1329)
- IMasonry(masonry).allocateSeigniorage(_amount) (contracts/Treasury.sol#1330)
Event emitted after the call(s):
- MasonryFunded(now,_amount) (contracts/Treasury.sol#1331)
Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#1344-1384):
External calls:
- _updateReptPrice() (contracts/Treasury.sol#1345)
  - IOracle(reptOracle).update() (contracts/Treasury.sol#1241)
- _sendToMasonry(reptSupply.mul(bootstrapSupplyExpansionPercent).div(10000))
(contracts/Treasury.sol#1350)
  - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/Treasury.sol#860)
  - IBasisAsset(rept).mint(address(this),_amount) (contracts/Treasury.sol#1310)
  - IERC20(rept).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#1315)
  - (success,returndata) = target.call{value: value}(data) (contracts/Treasury.sol#142)
  - IERC20(rept).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#1322)
  - IERC20(rept).safeApprove(masonry,0) (contracts/Treasury.sol#1328)
  - IERC20(rept).safeApprove(masonry,_amount) (contracts/Treasury.sol#1329)
  - IMasonry(masonry).allocateSeigniorage(_amount) (contracts/Treasury.sol#1330)
External calls sending eth:
- _sendToMasonry(reptSupply.mul(bootstrapSupplyExpansionPercent).div(10000))
(contracts/Treasury.sol#1350)
  - (success,returndata) = target.call{value: value}(data) (contracts/Treasury.sol#142)
Event emitted after the call(s):
- DaoFundFunded(now,_daoFundSharedAmount) (contracts/Treasury.sol#1316)
  - _sendToMasonry(reptSupply.mul(bootstrapSupplyExpansionPercent).div(10000))
(contracts/Treasury.sol#1350)
- DevFundFunded(now,_devFundSharedAmount) (contracts/Treasury.sol#1323)
  - _sendToMasonry(reptSupply.mul(bootstrapSupplyExpansionPercent).div(10000))
(contracts/Treasury.sol#1350)
- MasonryFunded(now,_amount) (contracts/Treasury.sol#1331)
  - _sendToMasonry(reptSupply.mul(bootstrapSupplyExpansionPercent).div(10000))
(contracts/Treasury.sol#1350)
Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#1344-1384):
```

```
External calls:
- _updateReptPrice() (contracts/Treasury.sol#1345)
  - IOracle(reptOracle).update() (contracts/Treasury.sol#1241)
- _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1375)
  - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/Treasury.sol#860)
  - IBasisAsset(rept).mint(address(this),_amount) (contracts/Treasury.sol#1310)
  - IERC20(rept).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#1315)
  - (success,returndata) = target.call{value: value}(data) (contracts/Treasury.sol#142)
  - IERC20(rept).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#1322)
  - IERC20(rept).safeApprove(masonry,0) (contracts/Treasury.sol#1328)
  - IERC20(rept).safeApprove(masonry,_amount) (contracts/Treasury.sol#1329)
  - IMasonry(masonry).allocateSeigniorage(_amount) (contracts/Treasury.sol#1330)
External calls sending eth:
- _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1375)
  - (success,returndata) = target.call{value: value}(data) (contracts/Treasury.sol#142)
Event emitted after the call(s):
- DaoFundFunded(now,_daoFundSharedAmount) (contracts/Treasury.sol#1316)
  - _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1375)
- DevFundFunded(now,_devFundSharedAmount) (contracts/Treasury.sol#1323)
  - _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1375)
- MasonryFunded(now,_amount) (contracts/Treasury.sol#1331)
  - _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1375)
Reentrancy in Treasury.allocateSeigniorage() (contracts/Treasury.sol#1344-1384):
External calls:
- _updateReptPrice() (contracts/Treasury.sol#1345)
  - IOracle(reptOracle).update() (contracts/Treasury.sol#1241)
- _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1375)
  - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(contracts/Treasury.sol#860)
  - IBasisAsset(rept).mint(address(this),_amount) (contracts/Treasury.sol#1310)
  - (success,returndata) = target.call{value: value}(data) (contracts/Treasury.sol#142)
  - IERC20(rept).transfer(daoFund,_daoFundSharedAmount) (contracts/Treasury.sol#1315)
  - IERC20(rept).transfer(devFund,_devFundSharedAmount) (contracts/Treasury.sol#1322)
  - IERC20(rept).safeApprove(masonry,0) (contracts/Treasury.sol#1328)
  - IERC20(rept).safeApprove(masonry,_amount) (contracts/Treasury.sol#1329)
  - IMasonry(masonry).allocateSeigniorage(_amount) (contracts/Treasury.sol#1330)
- IBasisAsset(rept).mint(address(this),_savedForBond) (contracts/Treasury.sol#1379)
External calls sending eth:
- _sendToMasonry(_savedForMasonry) (contracts/Treasury.sol#1375)
  - (success,returndata) = target.call{value: value}(data) (contracts/Treasury.sol#142)
```

```
Event emitted after the call(s):
- TreasuryFunded(now,_savedForBond) (contracts/Treasury.sol#1380)
Reentrancy in Treasury.buyBonds(uint256,uint256) (contracts/Treasury.sol#1254-1281):
External calls:
- IBasisAsset(rept).burnFrom(msg.sender,_reptAmount) (contracts/Treasury.sol#1274)
- IBasisAsset(rebond).mint(msg.sender,_bondAmount) (contracts/Treasury.sol#1275)
- _updateReptPrice() (contracts/Treasury.sol#1278)
  - IOracle(reptOracle).update() (contracts/Treasury.sol#1241)
Event emitted after the call(s):
- BoughreBonds(msg.sender,_reptAmount,_bondAmount) (contracts/Treasury.sol#1280)
Reentrancy in Treasury.redeemBonds(uint256,uint256) (contracts/Treasury.sol#1283-1307):
External calls:
- IBasisAsset(rebond).burnFrom(msg.sender,_bondAmount) (contracts/Treasury.sol#1301)
- IERC20(rept).safeTransfer(msg.sender,_reptAmount) (contracts/Treasury.sol#1302)
- _updateReptPrice() (contracts/Treasury.sol#1304)
  - IOracle(reptOracle).update() (contracts/Treasury.sol#1241)
Event emitted after the call(s):
- RedeemedBonds(msg.sender,_reptAmount,_bondAmount) (contracts/Treasury.sol#1306)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

ReptGenesisRewardPool.constructor(address,uint256,address) (contracts/
GenesisPool.sol#612-623) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(block.timestamp < _poolStartTime,late) (contracts/
GenesisPool.sol#617)
ReptGenesisRewardPool.checkPoolDuplicate(IERC20) (contracts/GenesisPool.sol#630-635)
uses timestamp for comparisons
Dangerous comparisons:
- pid < length (contracts/GenesisPool.sol#632)
- require(bool,string)(poolInfo[pid].token != _token,ReptGenesisPool: existing pool?)
(contracts/GenesisPool.sol#633)
ReptGenesisRewardPool.add(uint256,IERC20,bool,uint256) (contracts/
GenesisPool.sol#638-676) uses timestamp for comparisons
Dangerous comparisons:
- block.timestamp < poolStartTime (contracts/GenesisPool.sol#648)
- _lastRewardTime == 0 (contracts/GenesisPool.sol#650)
- _lastRewardTime < poolStartTime (contracts/GenesisPool.sol#653)
- _lastRewardTime == 0 || _lastRewardTime < block.timestamp (contracts/
GenesisPool.sol#659)
- _isStarted = (_lastRewardTime <= poolStartTime) || (_lastRewardTime <=
```

```
block.timestamp) (contracts/GenesisPool.sol#663-665)
ReptGenesisRewardPool.getGeneratedReward(uint256,uint256) (contracts/
GenesisPool.sol#691-702) uses timestamp for comparisons
Dangerous comparisons:
- _fromTime >= _toTime (contracts/GenesisPool.sol#692)
- _toTime >= poolEndTime (contracts/GenesisPool.sol#693)
- _toTime <= poolStartTime (contracts/GenesisPool.sol#698)
ReptGenesisRewardPool.pendingREPT(uint256,address) (contracts/GenesisPool.sol#705-716)
uses timestamp for comparisons
Dangerous comparisons:
- block.timestamp > pool.lastRewardTime && tokenSupply != 0 (contracts/
GenesisPool.sol#710)
ReptGenesisRewardPool.massUpdatePools() (contracts/GenesisPool.sol#719-724) uses
timestamp for comparisons
Dangerous comparisons:
- pid < length (contracts/GenesisPool.sol#721)
ReptGenesisRewardPool.updatePool(uint256) (contracts/GenesisPool.sol#727-747) uses
timestamp for comparisons
Dangerous comparisons:
- block.timestamp <= pool.lastRewardTime (contracts/GenesisPool.sol#729)
ReptGenesisRewardPool.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/
GenesisPool.sol#823-834) uses timestamp for comparisons
Dangerous comparisons:
- block.timestamp < poolEndTime + 7776000 (contracts/GenesisPool.sol#824)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

UniswapV2OracleLibrary.currentCumulativePrices(address) (contracts/Oracle.sol#497-521)
uses timestamp for comparisons
Dangerous comparisons:
- blockTimestampLast != blockTimestamp (contracts/Oracle.sol#512)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

RePath.unclaimedTreasuryFund() (contracts/RePATH.sol#811-816) uses timestamp for
comparisons
Dangerous comparisons:
- _now > endTime (contracts/RePATH.sol#813)
- communityFundLastClaimed >= _now (contracts/RePATH.sol#814)
RePath.unclaimedDevFund() (contracts/RePATH.sol#818-823) uses timestamp for comparisons
Dangerous comparisons:
```

```
- _now > endTime (contracts/RePATH.sol#820)
- devFundLastClaimed >= _now (contracts/RePATH.sol#821)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

RePathRewardPool.constructor(address,uint256) (contracts/RewardPool.sol#602-611) uses
timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(block.timestamp < _poolStartTime,late) (contracts/
RewardPool.sol#606)
RePathRewardPool.checkPoolDuplicate(IERC20) (contracts/RewardPool.sol#618-623) uses
timestamp for comparisons
Dangerous comparisons:
- pid < length (contracts/RewardPool.sol#620)
- require(bool,string)(poolInfo[pid].token != _token,RePathRewardPool: existing pool?)
(contracts/RewardPool.sol#621)
RePathRewardPool.add(uint256,IERC20,bool,uint256) (contracts/RewardPool.sol#626-664)
uses timestamp for comparisons
Dangerous comparisons:
- block.timestamp < poolStartTime (contracts/RewardPool.sol#636)
- _lastRewardTime == 0 (contracts/RewardPool.sol#638)
- _lastRewardTime < poolStartTime (contracts/RewardPool.sol#641)
- _lastRewardTime == 0 || _lastRewardTime < block.timestamp (contracts/
RewardPool.sol#647)
- _isStarted = (_lastRewardTime <= poolStartTime) || (_lastRewardTime <=
block.timestamp) (contracts/RewardPool.sol#651-653)
RePathRewardPool.getGeneratedReward(uint256,uint256) (contracts/RewardPool.sol#679-690)
uses timestamp for comparisons
Dangerous comparisons:
- _fromTime >= _toTime (contracts/RewardPool.sol#680)
- _toTime >= poolEndTime (contracts/RewardPool.sol#681)
- _toTime <= poolStartTime (contracts/RewardPool.sol#686)
RePathRewardPool.pendingShare(uint256,address) (contracts/RewardPool.sol#693-704) uses
timestamp for comparisons
Dangerous comparisons:
- block.timestamp > pool.lastRewardTime && tokenSupply != 0 (contracts/
RewardPool.sol#698)
RePathRewardPool.massUpdatePools() (contracts/RewardPool.sol#707-712) uses timestamp
for comparisons
Dangerous comparisons:
- pid < length (contracts/RewardPool.sol#709)
```

```
RePathRewardPool.updatePool(uint256) (contracts/RewardPool.sol#715-735) uses timestamp
for comparisons
Dangerous comparisons:
- block.timestamp <= pool.lastRewardTime (contracts/RewardPool.sol#717)
RePathRewardPool.governanceRecoverUnsupported(IERC20,uint256,address) (contracts/
RewardPool.sol#805-816) uses timestamp for comparisons
Dangerous comparisons:
- block.timestamp < poolEndTime + 7776000 (contracts/RewardPool.sol#806)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

Timelock.queueTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#278-287) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(eta >=
getBlockTimestamp().add(delay),Timelock::queueTransaction: Estimated execution block
must satisfy delay.) (contracts/Timelock.sol#280)
Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#298-323) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(getBlockTimestamp() >= eta,Timelock::executeTransaction:
Transaction hasn't surpassed time lock.) (contracts/Timelock.sol#303)
- require(bool,string)(getBlockTimestamp() <=
eta.add(GRACE_PERIOD),Timelock::executeTransaction: Transaction is stale.) (contracts/
Timelock.sol#304)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

Address.isContract(address) (contracts/GenesisPool.sol#26-35) uses assembly
- INLINE ASM (contracts/GenesisPool.sol#33)
Address._verifyCallResult(bool,bytes,string) (contracts/GenesisPool.sol#171-188) uses
assembly
- INLINE ASM (contracts/GenesisPool.sol#180-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address.isContract(address) (contracts/Masonry.sol#437-446) uses assembly
- INLINE ASM (contracts/Masonry.sol#444)
Address._verifyCallResult(bool,bytes,string) (contracts/Masonry.sol#582-599) uses
assembly
- INLINE ASM (contracts/Masonry.sol#591-594)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Address.isContract(address) (contracts/RewardPool.sol#26-35) uses assembly
 - INLINE ASM (contracts/RewardPool.sol#33)
Address._verifyCallResult(bool,bytes,string) (contracts/RewardPool.sol#171-188) uses
assembly
 - INLINE ASM (contracts/RewardPool.sol#180-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address.isContract(address) (contracts/Treasury.sol#49-58) uses assembly
 - INLINE ASM (contracts/Treasury.sol#56)
Address._verifyCallResult(bool,bytes,string) (contracts/Treasury.sol#194-211) uses
assembly
 - INLINE ASM (contracts/Treasury.sol#203-206)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
 - Version used: ['0.6.12', '>=0.6.0<0.8.0', '^0.6.0']
 - >=0.6.0<0.8.0 (contracts/DevFund.sol#5)
 - >=0.6.0<0.8.0 (contracts/DevFund.sol#31)
 - ^0.6.0 (contracts/DevFund.sol#99)
 - 0.6.12 (contracts/DevFund.sol#108)
 - 0.6.12 (contracts/DevFund.sol#187)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
pragma-directives-are-used

Different versions of Solidity is used:
 - Version used: ['0.6.12', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0']
 - >=0.6.2<0.8.0 (contracts/GenesisPool.sol#3)
 - >=0.6.0<0.8.0 (contracts/GenesisPool.sol#191)
 - >=0.6.0<0.8.0 (contracts/GenesisPool.sol#267)
 - >=0.6.0<0.8.0 (contracts/GenesisPool.sol#480)
 - 0.6.12 (contracts/GenesisPool.sol#551)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
pragma-directives-are-used

Different versions of Solidity is used:
 - Version used: ['0.6.12', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0', '^0.6.0']
 - >=0.6.0<0.8.0 (contracts/Masonry.sol#3)
 - ^0.6.0 (contracts/Masonry.sol#216)
 - 0.6.12 (contracts/Masonry.sol#232)
 - 0.6.12 (contracts/Masonry.sol#256)
```

```
- >=0.6.0<0.8.0 (contracts/Masonry.sol#270)
- >=0.6.2<0.8.0 (contracts/Masonry.sol#414)
- 0.6.12 (contracts/Masonry.sol#602)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Different versions of Solidity is used:
- Version used: ['0.6.12', '>=0.6.0<0.8.0', '^0.6.0']
- >=0.6.0<0.8.0 (contracts/Oracle.sol#2)
- >=0.6.0<0.8.0 (contracts/Oracle.sol#215)
- >=0.6.0<0.8.0 (contracts/Oracle.sol#238)
- ^0.6.0 (contracts/Oracle.sol#304)
- ^0.6.0 (contracts/Oracle.sol#394)
- ^0.6.0 (contracts/Oracle.sol#412)
- ^0.6.0 (contracts/Oracle.sol#485)
- 0.6.12 (contracts/Oracle.sol#525)
- ^0.6.0 (contracts/Oracle.sol#561)
- 0.6.12 (contracts/Oracle.sol#643)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Different versions of Solidity is used:
- Version used: ['0.6.12', '>=0.6.0<0.8.0']
- >=0.6.0<0.8.0 (contracts/Rebond.sol#3)
- >=0.6.0<0.8.0 (contracts/Rebond.sol#26)
- >=0.6.0<0.8.0 (contracts/Rebond.sol#92)
- 0.6.12 (contracts/Rebond.sol#305)
- >=0.6.0<0.8.0 (contracts/Rebond.sol#341)
- >=0.6.0<0.8.0 (contracts/Rebond.sol#417)
- >=0.6.0<0.8.0 (contracts/Rebond.sol#718)
- 0.6.12 (contracts/Rebond.sol#757)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Different versions of Solidity is used:
- Version used: ['0.6.12', '>=0.6.0<0.8.0']
- >=0.6.0<0.8.0 (contracts/RePATH.sol#3)
- >=0.6.0<0.8.0 (contracts/RePATH.sol#26)
- >=0.6.0<0.8.0 (contracts/RePATH.sol#92)
- >=0.6.0<0.8.0 (contracts/RePATH.sol#305)
- >=0.6.0<0.8.0 (contracts/RePATH.sol#381)
```

```
- >=0.6.0<0.8.0 (contracts/RePATH.sol#682)
- 0.6.12 (contracts/RePATH.sol#720)
- 0.6.12 (contracts/RePATH.sol#756)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Different versions of Solidity is used:
- Version used: ['0.6.12', '>=0.6.0<0.8.0']
- >=0.6.0<0.8.0 (contracts/Rept.sol#3)
- >=0.6.0<0.8.0 (contracts/Rept.sol#26)
- >=0.6.0<0.8.0 (contracts/Rept.sol#92)
- >=0.6.0<0.8.0 (contracts/Rept.sol#305)
- 0.6.12 (contracts/Rept.sol#335)
- 0.6.12 (contracts/Rept.sol#493)
- >=0.6.0<0.8.0 (contracts/Rept.sol#503)
- >=0.6.0<0.8.0 (contracts/Rept.sol#579)
- >=0.6.0<0.8.0 (contracts/Rept.sol#880)
- 0.6.12 (contracts/Rept.sol#918)
- 0.6.12 (contracts/Rept.sol#954)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Different versions of Solidity is used:
- Version used: ['0.6.12', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0']
- >=0.6.2<0.8.0 (contracts/RewardPool.sol#3)
- >=0.6.0<0.8.0 (contracts/RewardPool.sol#191)
- >=0.6.0<0.8.0 (contracts/RewardPool.sol#404)
- >=0.6.0<0.8.0 (contracts/RewardPool.sol#480)
- 0.6.12 (contracts/RewardPool.sol#550)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Different versions of Solidity is used:
- Version used: ['0.6.12', '>=0.6.0<0.8.0']
- >=0.6.0<0.8.0 (contracts/Timelock.sol#5)
- 0.6.12 (contracts/Timelock.sol#220)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Different versions of Solidity is used:
- Version used: ['0.6.12', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0', '^0.6.0']
```

```
 - >=0.6.0<0.8.0 (contracts/Treasury.sol#3)
 - >=0.6.2<0.8.0 (contracts/Treasury.sol#26)
 - >=0.6.0<0.8.0 (contracts/Treasury.sol#214)
 - >=0.6.0<0.8.0 (contracts/Treasury.sol#427)
 - >=0.6.0<0.8.0 (contracts/Treasury.sol#457)
 - ^0.6.0 (contracts/Treasury.sol#523)
 - 0.6.12 (contracts/Treasury.sol#539)
 - 0.6.12 (contracts/Treasury.sol#573)
 - ^0.6.0 (contracts/Treasury.sol#583)
 - 0.6.12 (contracts/Treasury.sol#601)
 - >=0.6.0<0.8.0 (contracts/Treasury.sol#625)
 - >=0.6.0<0.8.0 (contracts/Treasury.sol#701)
 - 0.6.12 (contracts/Treasury.sol#762)
 - >=0.6.0<0.8.0 (contracts/Treasury.sol#798)
 - 0.6.12 (contracts/Treasury.sol#868)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
pragma-directives-are-used

DevFund.constructor(address[],uint256[]) (contracts/DevFund.sol#204-212) has costly
operations inside a loop:
 - totalPoints += points[a] (contracts/DevFund.sol#210)
DevFund.removeAllocation(address) (contracts/DevFund.sol#238-247) has costly operations
inside a loop:
 - totalPoints -= allocations[a].points (contracts/DevFund.sol#241)
DevFund.removeAllocation(address) (contracts/DevFund.sol#238-247) has costly operations
inside a loop:
 - allocations.pop() (contracts/DevFund.sol#243)
DevFund.setAllocationPoints(address,uint256) (contracts/DevFund.sol#249-256) has costly
operations inside a loop:
 - totalPoints = totalPoints - allocations[a].points + points (contracts/
DevFund.sol#252)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-
operations-inside-a-loop

ReptGenesisRewardPool.updatePool(uint256) (contracts/GenesisPool.sol#727-747) has
costly operations inside a loop:
 - totalAllocPoint = totalAllocPoint.add(pool.allocPoint) (contracts/
GenesisPool.sol#739)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-
operations-inside-a-loop
```

```
RePathRewardPool.updatePool(uint256) (contracts/RewardPool.sol#715-735) has costly
operations inside a loop:
- totalAllocPoint = totalAllocPoint.add(pool.allocPoint) (contracts/
RewardPool.sol#727)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-
operations-inside-a-loop

Context._msgData() (contracts/DevFund.sol#22-25) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Address.functionCall(address,bytes) (contracts/GenesisPool.sol#79-81) is never used and
should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/
GenesisPool.sol#104-106) is never used and should be removed
Address.functionDelegateCall(address,bytes) (contracts/GenesisPool.sol#153-155) is
never used and should be removed
Address.functionDelegateCall(address,bytes,string) (contracts/GenesisPool.sol#163-169)
is never used and should be removed
Address.functionStaticCall(address,bytes) (contracts/GenesisPool.sol#129-131) is never
used and should be removed
Address.functionStaticCall(address,bytes,string) (contracts/GenesisPool.sol#139-145) is
never used and should be removed
Address.sendValue(address,uint256) (contracts/GenesisPool.sol#53-59) is never used and
should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (contracts/GenesisPool.sol#511-520) is
never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (contracts/
GenesisPool.sol#527-530) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (contracts/
GenesisPool.sol#522-525) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/GenesisPool.sol#454-457) is never used
and should be removed
SafeMath.mod(uint256,uint256) (contracts/GenesisPool.sol#416-419) is never used and
should be removed
SafeMath.mod(uint256,uint256,string) (contracts/GenesisPool.sol#474-477) is never used
and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/GenesisPool.sol#434-437) is never used
and should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/GenesisPool.sol#288-292) is never used and
should be removed
SafeMath.tryDiv(uint256,uint256) (contracts/GenesisPool.sol#324-327) is never used and
```

```
should be removed
SafeMath.tryMod(uint256,uint256) (contracts/GenesisPool.sol#334-337) is never used and
should be removed
SafeMath.tryMul(uint256,uint256) (contracts/GenesisPool.sol#309-317) is never used and
should be removed
SafeMath.trySub(uint256,uint256) (contracts/GenesisPool.sol#299-302) is never used and
should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Babylonian.sqrt(uint256) (contracts/Oracle.sol#397-409) is never used and should be
removed
FixedPoint.decode(FixedPoint.uq112x112) (contracts/Oracle.sol#464-466) is never used
and should be removed
FixedPoint.div(FixedPoint.uq112x112,uint112) (contracts/Oracle.sol#443-446) is never
used and should be removed
FixedPoint.encode(uint112) (contracts/Oracle.sol#433-435) is never used and should be
removed
FixedPoint.encode144(uint144) (contracts/Oracle.sol#438-440) is never used and should
be removed
FixedPoint.reciprocal(FixedPoint.uq112x112) (contracts/Oracle.sol#474-477) is never
used and should be removed
FixedPoint.sqrt(FixedPoint.uq112x112) (contracts/Oracle.sol#480-482) is never used and
should be removed
SafeMath.div(uint256,uint256) (contracts/Oracle.sol#134-137) is never used and should
be removed
SafeMath.mul(uint256,uint256) (contracts/Oracle.sol#115-120) is never used and should
be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

ERC20._setupDecimals(uint8) (contracts/Rebond.sol#697-699) is never used and should be
removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Math.average(uint256,uint256) (contracts/Rept.sol#329-332) is never used and should be
removed
Math.max(uint256,uint256) (contracts/Rept.sol#314-316) is never used and should be
removed
Math.min(uint256,uint256) (contracts/Rept.sol#321-323) is never used and should be
removed
SafeMath8.add(uint8,uint8) (contracts/Rept.sol#361-366) is never used and should be
removed
```

```
SafeMath8.div(uint8,uint8) (contracts/Rept.sol#435-437) is never used and should be
removed
SafeMath8.div(uint8,uint8,string) (contracts/Rept.sol#451-457) is never used and should
be removed
SafeMath8.mod(uint8,uint8) (contracts/Rept.sol#471-473) is never used and should be
removed
SafeMath8.mod(uint8,uint8,string) (contracts/Rept.sol#487-490) is never used and should
be removed
SafeMath8.mul(uint8,uint8) (contracts/Rept.sol#409-421) is never used and should be
removed
SafeMath8.sub(uint8,uint8) (contracts/Rept.sol#378-380) is never used and should be
removed
SafeMath8.sub(uint8,uint8,string) (contracts/Rept.sol#392-397) is never used and should
be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

SafeMath.sub(uint256,uint256) (contracts/Timelock.sol#103-106) is never used and should
be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (contracts/
Treasury.sol#817-819) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>=0.6.0<0.8.0 (contracts/DevFund.sol#5) is too complex
Pragma version>=0.6.0<0.8.0 (contracts/DevFund.sol#31) is too complex

Low level call in DevFund.call(address,uint256,bytes) (contracts/DevFund.sol#258-262):
- (success,result) = _to.call{value: _value}(_data) (contracts/DevFund.sol#259)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Low level call in Address.sendValue(address,uint256) (contracts/GenesisPool.sol#53-59):
- (success) = recipient.call{value: amount}() (contracts/GenesisPool.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(contracts/GenesisPool.sol#114-121):
- (success,returndata) = target.call{value: value}(data) (contracts/
GenesisPool.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/
GenesisPool.sol#139-145):
- (success,returndata) = target.staticcall(data) (contracts/GenesisPool.sol#143)
```

```
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/
GenesisPool.sol#163-169):
	- (success,returndata) = target.delegatecall(data) (contracts/GenesisPool.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Low level call in Address.sendValue(address,uint256) (contracts/Masonry.sol#464-470):
	- (success) = recipient.call{value: amount}() (contracts/Masonry.sol#468)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(contracts/Masonry.sol#525-532):
	- (success,returndata) = target.call{value: value}(data) (contracts/Masonry.sol#530)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/
Masonry.sol#550-556):
	- (success,returndata) = target.staticcall(data) (contracts/Masonry.sol#554)
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/
Masonry.sol#574-580):
	- (success,returndata) = target.delegatecall(data) (contracts/Masonry.sol#578)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Low level call in Address.sendValue(address,uint256) (contracts/RewardPool.sol#53-59):
	- (success) = recipient.call{value: amount}() (contracts/RewardPool.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(contracts/RewardPool.sol#114-121):
	- (success,returndata) = target.call{value: value}(data) (contracts/
RewardPool.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/
RewardPool.sol#139-145):
	- (success,returndata) = target.staticcall(data) (contracts/RewardPool.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/
RewardPool.sol#163-169):
	- (success,returndata) = target.delegatecall(data) (contracts/RewardPool.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Low level call in Timelock.executeTransaction(address,uint256,string,bytes,uint256)
(contracts/Timelock.sol#298-323):
	- (success,returnData) = target.call.value(value)(callData) (contracts/
Timelock.sol#317)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls
```

```
Low level call in Address.sendValue(address,uint256) (contracts/Treasury.sol#76-82):
	- (success) = recipient.call{value: amount}() (contracts/Treasury.sol#80)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(contracts/Treasury.sol#137-144):
	- (success,returndata) = target.call{value: value}(data) (contracts/Treasury.sol#142)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/
Treasury.sol#162-168):
	- (success,returndata) = target.staticcall(data) (contracts/Treasury.sol#166)
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/
Treasury.sol#186-192):
	- (success,returndata) = target.delegatecall(data) (contracts/Treasury.sol#190)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Redundant expression "this (contracts/DevFund.sol#23)" inContext (contracts/
DevFund.sol#17-26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements

Redundant expression "this (contracts/Oracle.sol#233)" inContext (contracts/
Oracle.sol#227-236)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements

Redundant expression "this (contracts/Rebond.sol#21)" inContext (contracts/
Rebond.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements

Redundant expression "this (contracts/RePATH.sol#21)" inContext (contracts/
RePATH.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements

Redundant expression "this (contracts/Rept.sol#21)" inContext (contracts/
Rept.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements

Redundant expression "this (contracts/Treasury.sol#21)" inContext (contracts/
Treasury.sol#15-24)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements

Variable UniswapV2OracleLibrary.currentCumulativePrices(address).price0Cumulative
(contracts/Oracle.sol#501) is too similar to
UniswapV2OracleLibrary.currentCumulativePrices(address).price1Cumulative (contracts/
Oracle.sol#502)
Variable Oracle.price0Average (contracts/Oracle.sol#662) is too similar to
Oracle.price1Average (contracts/Oracle.sol#663)
Variable Oracle.update().price0Cumulative (contracts/Oracle.sol#687) is too similar to
Oracle.update().price1Cumulative (contracts/Oracle.sol#687)
Variable Oracle.twap(address,uint256).price0Cumulative (contracts/Oracle.sol#718) is
too similar to Oracle.twap(address,uint256).price1Cumulative (contracts/Oracle.sol#718)
Variable Oracle.twap(address,uint256).price0Cumulative (contracts/Oracle.sol#718) is
too similar to Oracle.update().price1Cumulative (contracts/Oracle.sol#687)
Variable Oracle.price0CumulativeLast (contracts/Oracle.sol#660) is too similar to
Oracle.price1CumulativeLast (contracts/Oracle.sol#661)
Variable Oracle.update().price0Cumulative (contracts/Oracle.sol#687) is too similar to
Oracle.twap(address,uint256).price1Cumulative (contracts/Oracle.sol#718)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
are-too-similar

Variable Treasury.setExtraFunds(address,uint256,address,uint256)._daoFundSharedPercent
(contracts/Treasury.sol#1195) is too similar to
Treasury.setExtraFunds(address,uint256,address,uint256)._devFundSharedPercent
(contracts/Treasury.sol#1197)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
are-too-similar

Treasury.initialize(address,address,address,address,address,uint256) (contracts/
Treasury.sol#1084-1126) uses literals with too many digits:
- supplyTiers = (0,50000000000000000000000,100000000000000000000000,150000000000000000
00000000,200000000000000000000000000,500000000000000000000000000,1000000000000000000000000000,
2000000000000000000000000000,5000000000000000000000000000) (contracts/Treasury.sol#1103)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-
digits

ReptGenesisRewardPool.depositFeeBP (contracts/GenesisPool.sol#598) should be constant
ReptGenesisRewardPool.reptPerSecond (contracts/GenesisPool.sol#594) should be constant
ReptGenesisRewardPool.runningTime (contracts/GenesisPool.sol#595) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
```

```
variables-that-could-be-declared-constant

Rept.reptOracle (contracts/Rept.sol#971) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-constant

RePathRewardPool.rePathPerSecond (contracts/RewardPool.sol#593) should be constant
RePathRewardPool.runningTime (contracts/RewardPool.sol#594) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-constant

renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (contracts/DevFund.sol#81-84)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (contracts/DevFund.sol#90-94)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external

set(uint256,uint256) should be declared external:
- ReptGenesisRewardPool.set(uint256,uint256) (contracts/GenesisPool.sol#679-688)
deposit(uint256,uint256) should be declared external:
- ReptGenesisRewardPool.deposit(uint256,uint256) (contracts/GenesisPool.sol#750-774)
withdraw(uint256,uint256) should be declared external:
- ReptGenesisRewardPool.withdraw(uint256,uint256) (contracts/GenesisPool.sol#777-794)
emergencyWithdraw(uint256) should be declared external:
- ReptGenesisRewardPool.emergencyWithdraw(uint256) (contracts/GenesisPool.sol#797-805)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external

initialize(IERC20,IERC20,ITreasury) should be declared external:
- Masonry.initialize(IERC20,IERC20,ITreasury) (contracts/Masonry.sol#710-728)
rewardPerShare() should be declared external:
- Masonry.rewardPerShare() (contracts/Masonry.sol#782-784)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external

isOperator() should be declared external:
- Operator.isOperator() (contracts/Oracle.sol#546-548)
transferOperator(address) should be declared external:
- Operator.transferOperator(address) (contracts/Oracle.sol#550-552)
getCurrentEpoch() should be declared external:
```

```
- Epoch.getCurrentEpoch() (contracts/Oracle.sol#611-613)
getPeriod() should be declared external:
- Epoch.getPeriod() (contracts/Oracle.sol#615-617)
getStartTime() should be declared external:
- Epoch.getStartTime() (contracts/Oracle.sol#619-621)
getLastEpochTime() should be declared external:
- Epoch.getLastEpochTime() (contracts/Oracle.sol#623-625)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external

operator() should be declared external:
- Operator.operator() (contracts/Rebond.sol#317-319)
name() should be declared external:
- ERC20.name() (contracts/Rebond.sol#474-476)
symbol() should be declared external:
- ERC20.symbol() (contracts/Rebond.sol#482-484)
decimals() should be declared external:
- ERC20.decimals() (contracts/Rebond.sol#499-501)
totalSupply() should be declared external:
- ERC20.totalSupply() (contracts/Rebond.sol#506-508)
transfer(address,uint256) should be declared external:
- ERC20.transfer(address,uint256) (contracts/Rebond.sol#525-528)
approve(address,uint256) should be declared external:
- ERC20.approve(address,uint256) (contracts/Rebond.sol#544-547)
transferFrom(address,address,uint256) should be declared external:
- ERC20.transferFrom(address,address,uint256) (contracts/Rebond.sol#562-566)
increaseAllowance(address,uint256) should be declared external:
- ERC20.increaseAllowance(address,uint256) (contracts/Rebond.sol#580-583)
decreaseAllowance(address,uint256) should be declared external:
- ERC20.decreaseAllowance(address,uint256) (contracts/Rebond.sol#599-602)
mint(address,uint256) should be declared external:
- ReBond.mint(address,uint256) (contracts/Rebond.sol#771-777)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external

balanceOf(address) should be declared external:
- ERC20.balanceOf(address) (contracts/RePATH.sol#477-479)
burnFrom(address,uint256) should be declared external:
- ERC20Burnable.burnFrom(address,uint256) (contracts/RePATH.sol#712-717)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
```

```
transferFrom(address,address,uint256) should be declared external:
 - ERC20.transferFrom(address,address,uint256) (contracts/Rept.sol#724-728)
 - Rept.transferFrom(address,address,uint256) (contracts/Rept.sol#1005-1013)
mint(address,uint256) should be declared external:
 - Rept.mint(address,uint256) (contracts/Rept.sol#989-995)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external

set(uint256,uint256) should be declared external:
 - RePathRewardPool.set(uint256,uint256) (contracts/RewardPool.sol#667-676)
deposit(uint256,uint256) should be declared external:
 - RePathRewardPool.deposit(uint256,uint256) (contracts/RewardPool.sol#738-756)
withdraw(uint256,uint256) should be declared external:
 - RePathRewardPool.withdraw(uint256,uint256) (contracts/RewardPool.sol#759-776)
emergencyWithdraw(uint256) should be declared external:
 - RePathRewardPool.emergencyWithdraw(uint256) (contracts/RewardPool.sol#779-787)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external

setDelay(uint256) should be declared external:
 - Timelock.setDelay(uint256) (contracts/Timelock.sol#254-261)
acceptAdmin() should be declared external:
 - Timelock.acceptAdmin() (contracts/Timelock.sol#263-269)
setPendingAdmin(address) should be declared external:
 - Timelock.setPendingAdmin(address) (contracts/Timelock.sol#271-276)
queueTransaction(address,uint256,string,bytes,uint256) should be declared external:
 - Timelock.queueTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#278-287)
cancelTransaction(address,uint256,string,bytes,uint256) should be declared external:
 - Timelock.cancelTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#289-296)
executeTransaction(address,uint256,string,bytes,uint256) should be declared external:
 - Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#298-323)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external

isInitialized() should be declared external:
 - Treasury.isInitialized() (contracts/Treasury.sol#991-993)
getReptUpdatedPrice() should be declared external:
```

```
- Treasury.getReptUpdatedPrice() (contracts/Treasury.sol#1009-1015)
getReserve() should be declared external:
- Treasury.getReserve() (contracts/Treasury.sol#1018-1020)
getBurnableReptLeft() should be declared external:
- Treasury.getBurnableReptLeft() (contracts/Treasury.sol#1022-1034)
getRedeemableBonds() should be declared external:
- Treasury.getRedeemableBonds() (contracts/Treasury.sol#1036-1045)
initialize(address,address,address,address,address,uint256) should be declared
external:
- Treasury.initialize(address,address,address,address,address,uint256) (contracts/
Treasury.sol#1084-1126)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
. analyzed (78 contracts with 77 detectors), 452 result(s) found
```

0x Guard