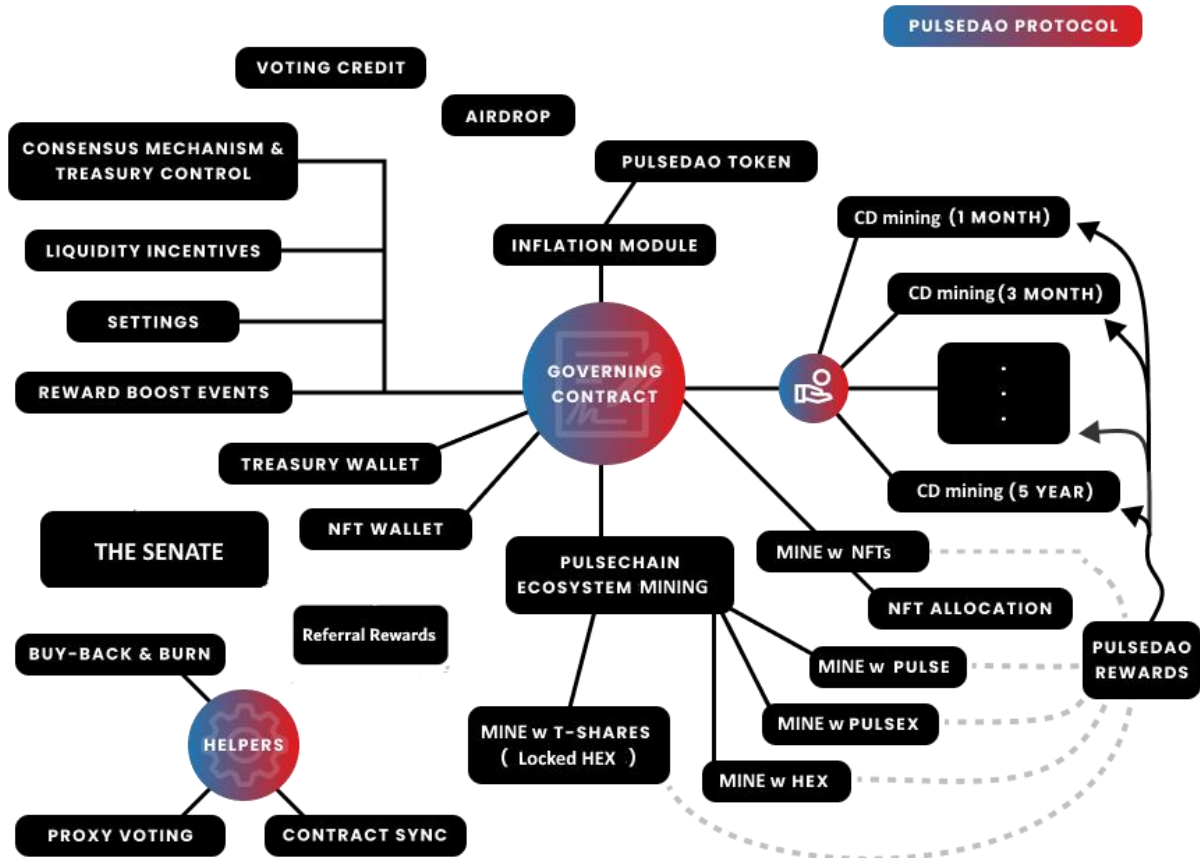




Protocol Security Review

Protocol Information

PulseDAO denotes a comprehensive protocol constituted by multiple smart contracts. These contracts collectively create a completely decentralized protocol, managed securely by the community without any need for a central authority.



Protocol Security Review

An independent smart contract security researcher has undertaken an in-depth review of the protocol.

This review was carried out with the intention of an exhaustive examination of the entire protocol code, scrutinized on a line-by-line basis, rather than focusing on creating an elaborate audit report.

Outlined below is a chat format summary of the comprehensive review, where the security researcher highlighted potential security challenges and vulnerabilities, and our developers proposed and implemented potential solutions. Parts of conversations have been omitted.

Official Audit Report

The vaults holding third-party tokens such as Pulse, HEX, PulseX, and others, have undergone inspection by professional security auditing firms. These firms have generated a report which can be accessed and reviewed here:

<https://pulseDAO.pro/contracts>

Security Analyst, [9/25/2023 9:47 AM]

Hi, mark, here Emanuele for the peer review, I will write you here from now on to have a faster response.

About the tool you send me I will take a look, but since chat gpt is a language model and not a logic model I don't think will work well with not basic contracts

developer,[9/25/2023 10:00 AM]

Ok, great.

Not for the audit itself but it quickly sums up what the contract does and I think it does that rather well (if that helps at all)

Security Analyst, [9/25/2023 10:34 AM]

I gave it a try with one of my project and uni v3, it is usefull for taking notes and because it puts all in 1 place tools like graphing the functions and slither (a static analyzer), I tried the ai but it just paraphrase the code/describe the code. And with uniswap contract it just broke. I might consider to use it because it put together some tools I have to use anyway, and having them in the same place save some times. (I case of private repos I will instead still use the tools installed locally since you don't know what they do with the contracts you upload)

```
constructor(  
    IERC20 _stakeToken,  
    IMasterChef _masterchef,  
    uint256 _poolId,  
    IERC20 _token  
) {  
    stakeToken = _stakeToken;  
    masterchef = _masterchef;  
    poolID = _poolId;  
    token = _token;  
  
    poolPayout[0].amount = 100;  
    poolPayout[0].minServe = 864000;  
  
    poolPayout[1].amount = 300;  
    poolPayout[1].minServe = 2592000;  
  
    poolPayout[2].amount = 500;  
    poolPayout[2].minServe = 5184000;  
  
    poolPayout[3].amount = 1000;  
    poolPayout[3].minServe = 8640000;  
  
    poolPayout[4].amount = 2500;  
    poolPayout[4].minServe = 20736000;  
  
    poolPayout[5].amount = 10000;  
    poolPayout[5].minServe = 31536000;  
}
```

Security Analyst, [9/25/2023 10:48 PM]

There was one thing that caught my attention while I skimmed the contracts. Is not in the 5 core contract but I checked it now before you will publish the contracts. So in every contract under pulse-ecosystem in the constructor you try to give values to object in a mapping without giving any index, and is an illegal operation, you can't even compile the contract because of that. Is it because you are waiting to deploy the pools and after that you will manually add those address manually?

developer,[9/25/2023 10:58 PM]

Yeah I will deploy pools and enter addresses manually and then I can easily copy/paste on every contract that has this format

If I use addresses in constructors it gets tiresome having to enter them one by one and in some cases I think I even got error that there are too many constructor arguments

So I enter the address and then compile it

Security Analyst, [9/25/2023 10:59 PM]
gotcha

Security Analyst, [9/26/2023 6:24 PM]
IDTX is the interface of what contract? I thought it was XPD but the interface have some functions not implemented in XPD: decreaseAllowance, increaseAllowance, transferStuckTokens

developer,[9/26/2023 7:41 PM]
It's for XPD.

Previously the name was DTX and didn't want to change everywhere

I removed the transferStuckTokens just recently

developer,[9/26/2023 7:42 PM]
Previously it was solidity 0.8.0 and now changed to 0.8.20, perhaps they updated the ERC20... Will update interface

developer,[9/26/2023 7:52 PM]
yeah this was the reason

Security Analyst, [9/26/2023 11:59 PM]
gotcha, I recommend you to put it as a comment just to be more clear on the interface e XPD contract.

Talking about the XPD contract, to make it more clean I would have made a custom ERC20 to remove the unused and inaccessible second name/symbol, but no security concerns.

You have the reentrancy guard inherited but unused so you can safely remove it, and also remove the import of the IERC20 since also that is unused.

developer,[9/27/2023 12:03 AM]
ok removed reentrancy and removed import

developer,[9/28/2023 12:32 AM]
Does the logic in governor.sol for rebalancePools seem alright?

Basically I want the rewards to be
20% for pool1
30% for pool2
50% for pool3

75% for pool4
115% for pool5
150% for pool6

Not really percentages, but like relative weights (eg. the allocation should be x7.5 larger for pool6 compared to pool1 for each token staked)

It has to be rebalanced based on the tokens in the pool to make sure the ratios stay correct

Security Analyst, [9/28/2023 12:44 AM]

I'm currently checking masterchef, I do first the base contract and after the one that are build on top of them so I can also check if they interact in the right way. When I will do the governor I will keep in mind to also check that the percentage will be calculated in the right way

developer,[9/28/2023 12:46 AM]

Ok. Ill check it tomorrow also, already had to fix something lol

Security Analyst, [9/28/2023 12:46 AM]

about masterchef, the contracts you add as trustedContracts how will call the burn function on the masterchef? Do they follow any requirments? and which contracts will be added?

Security Analyst, [9/28/2023 12:46 AM]

let me know when you change code so I don't work on old one XD

developer,[9/28/2023 12:46 AM]

the contracts in /mining/ (which are the same)

developer,[9/28/2023 12:47 AM]

and the votingCredit.sol

developer,[9/28/2023 12:47 AM]

just those ones

developer,[9/28/2023 12:47 AM]

they are added as trusted contract at beginning and thats it

Security Analyst, [9/28/2023 12:48 AM]

gotcha, I will then keep a look also on those, because I have to check there is not the possibility of infinetly loop the publishToekns and burn functions through those contracts

Security Analyst, [9/28/2023 12:49 AM]

because if a user do so can push totalPublished to uint256 limit (even if would be a more than a lot gas espensive) and make the contract useless because some function will fail because the overflow

Security Analyst, [9/28/2023 12:49 AM]

can you tell me the chain where you deploy? so in case like these I can check if transaction gas cost make it feasible to such attacks

developer,[9/28/2023 12:55 AM]

how would you do that though?

They need the credit

and credit is given as reward on every block

developer,[9/28/2023 12:57 AM]

totalPublished is kept so we can keep "virtual supply" (and stop the minting in time)... Because token contract will only allow 21B tokens

and here when the virtual supply would exceed 21B, it sets reward to 0

"virtual" is because the credit is given, but not yet minted

Security Analyst, [9/28/2023 12:58 AM]

it depends on the implementations inside the trusted contract that I still have to see.

but in publishTokens you give away credit for the token in a ration of 1:1

in the burn, you burn token for credit (given to the trusted contract) in ration of 1:1.

So in case the trusted contract give you back those credits you can call the 2 functions in loop and just increase the 2 variable

developer,[9/28/2023 12:59 AM]

it will be somewhere where it's relatively cheap like pulse (pulse is 0.002\$ for native "eth" tx)

```
function publishTokens(address _to, uint256 _amount) external {
    require(credit[msg.sender] >= _amount, "Insufficient credit");
    credit[msg.sender] = credit[msg.sender] - _amount;
    totalPublished+=_amount;
    dtx.mint(_to, _amount);
}

function burn(address _from, uint256 _amount) external returns (bool) {
    require(trustedContract[msg.sender], "only trusted contracts");
    require(dtx.burnToken(_from, _amount), "burn failed");
    credit[msg.sender] = credit[msg.sender] + _amount;
    totalPrincipalBurned+= _amount;
    return true;
}
```

Security Analyst, [9/28/2023 12:59 AM]

for example if as a trusted contract I have a contract that after burn give back full credit to the user, the user can loop publish -> burn -> transfercredit(from the trusted contract) and repeat

Security Analyst, [9/28/2023 1:00 AM]

but as I said, have to check the contract you will add to see if there will be anyway to keep full credits and do the loop

developer,[9/28/2023 1:00 AM]

the credit is given to the contract though not the user

developer,[9/28/2023 1:00 AM]
he can't transfer

Security Analyst, [9/28/2023 1:01 AM]
yea yea, for now is not a problem but still to check the contract you will add XD

Security Analyst, [9/28/2023 1:01 AM]
I just noted it down to keep in mind while looking at those trusted contracts

developer,[9/28/2023 1:01 AM]
It's a substitute for transferring tokens and using balance

Security Analyst, [9/28/2023 1:02 AM]



Security Analyst, [9/29/2023 12:20 AM]
so, since I see that you are still editing the contracts I will give a good look to all of them, and after that I will give a final revision to all, doing so I will also know what the other part of the protocol do in detail.

Meanwhile I also finished masterchef and here are the list of things I found:

- 1) Withdraw and EmergencyWithdraw are of no use since you edited the contract and you can remove them.
- 2) fairMintSenate can call by anyone, and can also be called multiple times giving each time the "fee" to the senators
- 3) fairMintSenate in case of very high number of senators the loop could go above gas limit, but I seems you plan to keep it around 100 so there is no problem
- 4) in renounceRewards you should add a massupdate otherwise the rewards from the last block up to that moment are lost, also you should emit the event UpdateEmissions
- 5) in stopPublishing you remove from totalAllocaPoint the current value inside the poolInfo[_pid] that you just setted to 0, so you alway remove 0, you have to save the value in another variable before putting the alloc to 0, and remove that from the end, otherwise first reduce the total allocation and after set to 0 the all of the pool
- 6) governorator fee go to the devAddress that is msg.sender, so idk to what you will change that value and idk if is what you imagined, see next point
- 7) feeAddress is never used (so also useless update function) for fee, so idk if you want the fee on this or dev one or what
- 8) there is an unlikely scenario where updatePool will break, in case of the supply be almost max and nobody still called the renounceRewards, since updatePull mint the fee to the dev address the mint could go over the max_supply and fail the transaction, failing all the transaction that call the update, so in case also the renounceRewards.
- 9) in publishTokens and transferCredit there is not an update pool, so the credit of the pool are not updated, the functions that update the pool accrediting the credits are only: set, startPublishing, stopPublishing, updateEmissionRate, massUpdatePools, updatePool. So if you don't call any of these functions on the trusted contracts before moving/claming credits you will not have them added the once got in the meanwhile, (have to check the contracts that will be trusted to see if they call update or not)

Security Analyst, [9/29/2023 12:22 AM]

also, you use the safemath library but since you don't use any extra functions like tryAdd, trySub etc, nor the base sub, add, mul with the custom error message you can remove that library and just use the default overflow check built in the 0.8.* solidity compiler

Security Analyst, [9/29/2023 12:25 AM]

I think you can also remove the safeErc since you don't use any of the added extra functions, you just call mint, burnToken, transferOwnership and totalSupply

Security Analyst, [9/29/2023 12:27 AM]

Also IERC20 and reentrancyguard(same as the xpd token) are not used

Security Analyst, [9/29/2023 12:29 AM]

about the unused depositFeeBp I still would recommend to remove it and edit the frontend

developer,[9/29/2023 1:07 AM]

- 1.) agree, removed now
- 2.) Yes it should be able to be called perpetually. It allows to mint 0.01% of all tokens credited to each senator (anyone can call, but it gets credited to senators)
- 3.) yeah i know
- 4.) will change
- 5.) oopsie, good discover. changed the lines so its subtracted before its removed
- 6.) yeah governor fee goes to devAddress which is governing contract(didn't change name when initially it was meant for dev)
- 7.) feeAddress is used for some external contracts so leaving that(they update their treasury to feeAddress)
- 8.) fixed by setting governorFee to 0 on renounceRewards before it calls mass update pools
- 9.) the contract transfers credit in case it is changed (the mining one)

and removed all the bloat as you recommended

Security Analyst, [9/29/2023 1:09 AM]

For the 2 isn't a problem that can be called multiple times? For example I can call it 100 times in a single transaction and give 100x rewards to a senate and if I'm a senate is good for me

developer,[9/29/2023 1:11 AM]

it's 0.01% - - fairTokensPublishedToSenate

Keeps track of tokens already published, so that it's always up to 0.01%, or is this not good

Security Analyst, [9/29/2023 1:12 AM]

The 9 you didn't understand, I will try to explain better tomorrow

developer,[9/29/2023 1:12 AM]

ill keep safemath because else i have to be removing and replacing .add with +

developer,[9/29/2023 1:12 AM]

or is it a big difference

Security Analyst, [9/29/2023 1:12 AM]

But if I credit the senate the 0.01% 100 times it end up being a 1%

Security Analyst, [9/29/2023 1:13 AM]



Security Analyst, [9/29/2023 1:13 AM]
Since gas are not a concern is not such big deal

developer,[9/29/2023 1:13 AM]
lol, don't want to make a mistake D:

Security Analyst, [9/29/2023 1:14 AM]
Oke oke

developer,[9/29/2023 1:14 AM]
fairTokensPublishedToSenate increases with every call so if you call immedietly after just calling you'd
get 0 i think

developer,[9/29/2023 1:14 AM]
likely wrong though

Security Analyst, [9/29/2023 1:15 AM]
Will check it better tomorrow and I will let you know

Security Analyst, [9/29/2023 1:15 AM]
I might read it wrong because late hours

developer,[9/29/2023 1:19 AM]
it's wrong though i think

Need to keep track of totalCreditRewardsAtLastFairMint and give 0.01% of that

developer,[9/29/2023 1:43 AM]
fixed it pretty sure

Security Analyst, [9/29/2023 7:17 AM]
I will let you know

Security Analyst, [9/30/2023 6:27 PM]
so about the 9 point. So, every pull start with 0 credits, and with the updatePool, the credit get in the
block timespan are added (accredited). So the credit are updated up to the time you called the last
updatePool on that specific pull (including mass pool update). So the contracts in the pool, have to
call updatePool of their own pool before doing a credit transaction so they receive the credit they
earned in the meanwhile with the token distribution

Security Analyst, [9/30/2023 6:31 PM]
in standard mastercheft it is down every deposit/withdrawn becaues it also change the the user
shares in a way you need to compute them everytime, I will not go in detail. In your case since there
are no problem with share inside a pool you don't have the need of updating the pool before every
change of deposited amount, but you still have to update it to get credited the credit, you could call
if from the interested contract before every transfer/use of the credit, or call it only when you need
to get the pending credit because the one the contract have are less than the amount they have to
transfer/use. (I would still go for the approach of calling it everytime because more clean, and less

complex, also in case of problem better to have the most amount of credit already added to the balance, so if you lose the pending you lose just few of them and not almost all)

Security Analyst, [9/30/2023 6:31 PM]
if you have any question or doubt feel free to ask

Security Analyst, [9/30/2023 6:49 PM]

I checked again the fairMintSenate, and this are the result, you kinda tried in the first version of keeping count of the old one with fairTokensPublishedToSenate (so I was kinda wrong), but it was still implemented wrongly because, when you were removing the fair token from above you were removing it from the amount of a "single share", while when you were adding it in the end you were counting it for every senate, doing so you were making it a lot larger making it impossible to claim the rewards ever again XD.

moreover calculating it like that on the amount of the single share was not keeping in count the decrease/increase of the number of senate fucking a bit up the rewards, let me explain as well as I can, if you have 100 senate and claim, but after you go to 1000 senate the fee to single senate would already be a lot lower (smaller slice of the cake), but you weren't adjusting the removed amount based on the slice resize but just removing the amount as if the senate were still 100

Security Analyst, [9/30/2023 6:49 PM]
anyway the way you fixed it solved both of this problem anyway XD

Security Analyst, [9/30/2023 7:09 PM]
this isn't a repetition of logic? If I get it correctly the first one is for the whitelist of the incoming transfer, and the second one for the outgoing transfer

Security Analyst, [9/30/2023 7:09 PM]
mapping(address => bool) public trustedSender; //Pools with shorter lockup duration(trustedSender(contracts) can transfer into this pool)
mapping(address => bool) public trustedPool; //Pools with longer lockup duration(can transfer from this pool into trustedPool(contracts))

Security Analyst, [9/30/2023 7:09 PM]
but isn't just the outgoing list enough?

developer,[9/30/2023 9:53 PM]
Trusted pools are ABOVE (longer duration) and trusted senders are BELOW(shorter duration)

It's needed.

Because you can only go(hopStake) from lower stake duration into higher, but not vice versa. From 1 month stake, you can go into any with longer duration(3month,6month,1yr,3yr,5year)

but you can't go(hop) from a longer time deposit (eg from 6month into 3month)

So when i launch contracts, i add addresses for each option
For example in 1month contract there will be 0 trusted senders because it's the lowest option, and all other contracts will be trusted pools(because they are all longer and we can extend stake)

developer,[9/30/2023 9:55 PM]

example:

1Month Contract:
Trusted Senders: /
Trusted Pools: 3month, 6month,1yr, 3yr, 5yr

3 Month ContractL
Trusted senders: 1month
Trusted pools: 6month, 1yr, 3yr, 5yr

...

developer,[9/30/2023 11:53 PM]
Its in the governor.sol
(users get paid for calling the contract function, which rebalances rewards and runs the massUpdatePools to accredit pools)

It's under assumption that people would call it periodically so it would be somewhat accurate, but if it was not called you could get less than you deserve

Security Analyst, [10/1/2023 1:14 AM]
yea, I figured it out a bit after reading the contract XD

Security Analyst, [10/1/2023 1:15 AM]
wasn't the mining and the voting credit that will have the pools there?

Security Analyst, [10/1/2023 1:16 AM]
can you explain me a bit better who will be the trusted cotntracts and which contract interacts with masterchef for what? So to have the big picture of how the contracts I still have to look works?

developer,[10/1/2023 8:02 PM]
trusted contracts are:
-all mining pool
-the voting credit contract

developer,[10/1/2023 8:03 PM]
-mining pools burn as deposit and mint the rewards
-voting credit contract is used to burn the tokens and convert them to users voting credit. It has the function to publish its tokens to the governing contract

<instructions for launching protocol; redacted>

developer,[10/1/2023 10:10 PM]
this is the process how i deploy it all:

Security Analyst, [10/1/2023 10:22 PM]
Ohhhhh this is so useful, thx

developer,[10/5/2023 2:17 PM]
Changed the vaults in /pulse-ecosystem/

if pool rewards were updated directly through masterchef (and not through harvest() function inside the vault), it would not have accounted the rewards

So instead now keeping score of "lastCredit" for this contract to get amount of rewards accumulated

Security Analyst, [10/5/2023 2:58 PM]

Didn't understood it. Anyway I'm watching the mining pool these days. Sorry but was busy a bit this week and hurted my eye. I should write you about the mining one tomorrow or the day after

Security Analyst, [10/7/2023 1:03 PM]

I finished to check mining/1Month and there are no concerning vulnerabilities expect you don't update the masterchef pull, beside that there are a ton of code that should have been cleaned and some minor improvements (no need to change much the code if already deployed but I write them so you will keep them in mind for next contracts you write).

about the update pool, in the mining contract you don't call updatePool on masterchef and that don't make the masterchef add the credit from the pending onces to the contract. To fix that you have to call updatePoll every time you "transfer" or "consume" the credit of this contract. (note that is not a security vulnerability but if you don't add them until user update by their own the pool, all function that move credits will revert for insufficient funds).

you have to add the updatePool in these function:

- 1) In the withdraw function before the if that have the 2 publishToekn
- 2) in hopSStakeToAnotherPool before the transferCredit
- 3) in votingCredit before the publishTokens
- 4) in migrateStake before the transferCredit

Contract Events:

- 1) I would add the stakeId in Deposit, GiftDeposit
- 2) In trustedSender and TrustedPool the contractAddress should be indexed (indexed should be the params that you use to filter the events when you seach a specific one, if you wish I can explain where to use indexed in a more detailed way if you wish)
- 3) In stakeApproval owner should be indexed for sure, while spender could also be putted as indexed
- 4) in StakeAllowanceReevoke the owner should be indexed
- 5) in TransferStakeFrom _from and _to should be indexed
- 6) in setDelegate both userDelegating and delegatee should be indexed
- 7) all the setFunctions (enableDisableStakeTransferFrom, setCallFee, setMigrationPool, modifyPartialWithdrawals, modifyPartialTransfers, enableDisableStakeTransfer, allowTxOrigin) need to emit an event since they update some important values and bring changes to the protocol (PS as already said you could merge all this function togther having only one settings object and changing in it the value that you want to change)
- 8) setMasterChefAddress and setMinimumGiftDeposit should have an event (same reason as previous point)

Code improvement:

- 1) where you do: get the voting for, do the if and in case call updateVotingAddDiff, you should put all the logic of checking userVote + the if inside the function updateVotingAddDiff (ps: you anyway pass the user to the function so you already have it to check the mapping)
- 2) the code of: currentshares = 0, and the if else to calculate current share should be putted in a private function that return the amount (you use that same code in 4 different places)

- 3) the code that check if the share are 0 and in that case delete the stake could be putted in a single function (or directly within the `_removeStake` renaming it to a better name), in the case where you subtract the `currentAmount` just transfer that amount before calling this function so it is already subbed from the total.
- 4) in `voteForProposal` and `setDelegate` you could rewrite this `(allowOrigin && asProxy) ? _wallet = tx.origin : _wallet = msg.sender;` to this `_wallet = (allowOrigin && asProxy) ? tx.origin : msg.sender;` is a bit cleaner
- 5) in `voteForProposal` and `delegateeVote` you could put the code from `votingFor` to `userVote[wallet]` in a private function and remove the code repetition

Security Analyst, [10/7/2023 1:03 PM]

Staff to point out that could work in a way you didn't intended:

- 1) the stake allowance to transfer it become invalid if the user do anything to his stake (like partial withdrawn, restake etc)
- 2) in the `withdraw` function the fee are calculated based on the number fo days and not time, you should do the multiplication of 786 before the division of 86400 if you want to have fee based on time and without the step based on the number for days (ps: is for both if branches)
- 3) `calculateHarvestDTXRewards` and `calculaaePendingDTXRewards` calculate the rewards only on the pending token and not on the accredited token to the contract, moreover the amount of pending is based on last time that the `updatePool` was called and change a lot based on how is called (for example if you fix how I told you and add it before each transfer of creadit, how it should be, those function return a lot less), btw I think your intention is to return the credit + the pending.

Going on with minor improvments:

- 1) you can remove the import of `IERC20` (unused)
- 2) you can remove the you can remove the import of `Ownable` since you made your version of it in the contract.
- 3) you could remove `safemath` like the other contracts.
- 4) all contract settings should be putted together in an object so you optimize space, and also you group app all the function to update those variables in one and you will also have only 1 emit event to write the updates.
- 5) I would add an second `addAndExtendStake` function without the `recipientAddr` that call the `addAndExtedStake` passing `msg.sender` as recipient (so user can call that without having to put as input their own address)
- 6) add `nonReentrant` in the `withdraw` function
- 7) inside `HopStakeToAnotherPool` in the `else` block you should separate the requirment every condition concatenated with `&&` in a separate `require`, doing so the code is way more clean and you can put the right error message for each condition, for example the condition of `block.timestamp > _lastDepositedTime.add(withdrawFeePeriod)` could make the transaction fail but user see the error of can only hop into pre-set Pools that is not the real transaction error.
- 8) you could add `nonReentrant` to transfer `StakeToAnotherWallet`
- 9) when you do the `require` of `shares <= user.shares` could be moved up before calculations in `transferStakeeToAnotherWallet`
- 10) you can substitute `i++` with `++i` (more gas efficient) in: `delegateeVotem`, `migrateAllStakes`, `setTrustedSender`, `setTrustedPool`, `getUserTotalShares`

Security Analyst, [10/7/2023 1:03 PM]

that should be all

Security Analyst, [10/7/2023 1:04 PM]

I think that next contract I will check will be the `consensus.sol`, if you prefer otherwise let me know

Security Analyst, [10/7/2023 6:30 PM]
why in consensus contract there are some function commented? Are supposed to be removed or was an error and you didn't removed the comments?

Security Analyst, [10/7/2023 6:31 PM]
pls let me know asap

developer,[10/7/2023 6:40 PM]
maybe havent removed comments

developer,[10/7/2023 6:40 PM]
consensus, then senate
and then /pulse-ecosystem (vaults)

seems fine


developer,[10/7/2023 6:41 PM]
which line speciifcally

Security Analyst, [10/7/2023 6:41 PM]
consensus.sol have voteTreasuryTransferProposalY, voteTreasuryTransferProposalN and vetoTreasuryTransferProposal commented out

Security Analyst, [10/7/2023 6:42 PM]
150-213

developer,[10/7/2023 6:42 PM]
oops. lol. That was a mistake

Security Analyst, [10/7/2023 6:42 PM]
bruh

Security Analyst, [10/7/2023 6:42 PM]


developer,[10/7/2023 6:44 PM]
ive changed it now

Security Analyst, [10/7/2023 6:44 PM]

developer,[10/8/2023 1:25 PM]
Checked for the mining. Thanks for the extensive report.
It's already been tested quite thoroughly and the critical mistakes have been fixed before so happy to see it seems to be okay now.

Since there is nothing critical and i already launched it, i will leave it as is for now.

In regards to updatePool... I think it should all work in general(because when calculating amount the balanceOf() function adds the pending amount).....

And there will be sufficient credit most of the time. And if there is not, it can simply be updated manually.

Calculating penalties based on days is okay, and the calculateHarvestDTXRewards is not really being used so should be fine as well

Security Analyst, [10/8/2023 1:29 PM]

gotcha, keep in mind you will have to call a lot the update at the start

Code improvements:

- 1) you can remove IERC20
- 2) in voteTreasuryTransferProposalN/Y you can put the 2 requirements as function modifier
- 3) in approveTreasuryTransfer, killTreasurytransferProposal senateVetoTreasury you could put the require of .valid in a modifier
- 4) in changeGovernor, vetoGovernor, vetoGovernor2, senateVeto, senateVetoTreasury you could put the proposalID % 2 == 1 in a modifier
- 5) in vetoGovernor I recommend to split the requirement conditions in 2 separate ones with their errors (not recommended use && in require)
- 6) in vetoGovernor and governor2 the part withupdate should be inside the if since is only used if you actually invalidate the governor
- 7) in senateVeto and senateVetoTreasury the require to be senateContract can made as modifier
- 8) in tokenCasterPreVote would be better put the multiplication before the divisions, moreover to make it more easily readable would have been better to put all the value in 100 base and divide by 1e20

Problems:

- 1) in Initiate TreasuryTransferProposal you give to the treasury object the consensus id from length +1 that correspond the consensus of the vote against and after in the other function you get that id for the vote for while +1 for the vote against (that would go on the next proposal)
- 2) in vetoTreasuryTransferProposal (that I assume that you should call it after proposal end time), the require check that you are still in the vote phase, without change a user can just sacrifice a slightly higher value after proposal creation and instantly invalid it.
- 3) in vetoTreasuryTransferProposal you check if proposalID is != but that id is for treasuryProposal not consensus, so you prevent the first proposal to be veto-ed

percentage wrong calculations (or maybe not as intended):

- 1) in approvetreasuryTransfer (line 237) you commented one third but you are checking if the against are 1/3 compared to the yes that is 25% of the total votes
- 2) in change governor(line 329) you wrote in the comment ,Atleast 33% of voters required, but in the code is required 20% instead
- 3) in changeGovernor (line 334) you wrote 2/3 agreement -> 1/3 against, but the vote are checked against the yes, so on the totals is 25% not 33%
- 4) in vetoGovernor (line 360) is 20% against the yes that is 16.7% of the totals

Events:

- 1) EnforceDelay and RemoveDelay are unused
- 2) in TreasuryProposal proposalID should be indexed imo
- 3) in TreasuryEnfore proposalID should be indexed

developer,[10/9/2023 10:11 PM]

Removed the IERC20(1), split the requirement conditions into 2 separate ones(5), moved the withUpdate part inside the if statement (6)

I will leave the rest, because not critical

Going onto problems....

- 1.) you are correct, removed the +1
- 2.) this is correct, if at any time the votes against exceed votes in favor, it can be rejected
- 3.) removed the require for != 0

percentage wrong calculations (or maybe not as intended):

- 1.) Yeah that's okay. If a third of votes in favor is against, it can be rejected
- 2.) changed it to 15% now and changed the comment as well... Because a lot of people just stake and forget i think lower threshold like this is needed
- 3.) if there is a third (of votes in favor) voting against, they can reject, so i think it's okay
- 4.) if more than 20% of votes (20% of whatever is in favor) is against, it can reject. think this is correct

removed the events and added index for the other 2

Security Analyst, [10/9/2023 10:16 PM]

about the veto I was just saying that I can create a bot that immediately after a proposal is deployed I can just veto it with sametoken +1

developer,[10/9/2023 10:40 PM]

ahh, you are right

Added this to fix it

Security Analyst, [10/10/2023 5:19 PM]

I'm sorry, I forgot to tell you that in consensus contract, in senateVetoTreasury you pass a treasury ID not a consensus id so the proposalID % 2 == 1 is wrong and have to be remove. Hope you didn't already deployed the new version

Security Analyst, [10/10/2023 5:20 PM]

moreover, in changeGovernor you use tx.origin, while in trasuryEnfore you use msg.sender

Security Analyst, [10/10/2023 5:22 PM]

about the contract that you modify I would recommend to publish them the days before because last week I have in mind to re-check everything and having all the big picture of all the contracts I can check the interactions among them a lot better and find problematics there

developer,[10/10/2023 5:28 PM]

You are correct, removed it

developer,[10/10/2023 5:29 PM]

Changed msgsender to tx.origin in senateVetoTreasury as well..... should be tx.origin (because it's called from the senate contract)

Security Analyst, [10/10/2023 5:31 PM]

I'm almost finished with the senate contract, after that as core contract remain only the token vault. After that, until the 18/10 I was thinking about to do some other "miscellaneous" contracts, if it is fine for you, in detail I wanted to give a look especially to the credit contract and governor, so can you tell me after those what do you think I should take a look in order of importance? And in the last week, with fresh mind recheck all the contracts focusing on the integrations between each others

Security Analyst, [10/10/2023 5:31 PM]
yea, expected that

developer,[10/10/2023 7:51 PM]
Yeah, seems fine.

the token vault is most important because people deposit their tokens there (the plsVault is equivalent to the tokenVault, it just uses native token instead of ERC20, so there is a few lines of difference) . Though i think the attack surface is rather small.

votingCredit is a few lines basically, shouldn't be much of a problem IMO. And governor is basically used because the contract size would be too big to have it all in one (so a decision process for determining bonuses or whatever is in one contract and decision process for something else in another contract. And they call to governor which actually updates the values)

Security Analyst, [10/11/2023 12:40 AM]
I will start the token vault tomorrow

Security Analyst, [10/11/2023 1:20 AM]
about the senate:

Problems:

- 1) lastVotingCreditGrant is unused
- 2) min senate is setted to 20, but in the expellSenator error message there is written 25, I recommend to remove the number and just leave something like, already minimum senate member reached!.
- 3) in expellSenator and selfReplaceSenator the for loop could be costly, I recommend to add a second function parameter of type: uint256 senatorId, write a require of senators[senatorId] == _senator removing the need of a for for the research of the index.
- 4) in selfReplaceSenator the require of !isSenator[_newSenator] check only if is not currently a senator but not if it was already added and removed in the past, I would change it with !addedSenator (if added senator is false, there is no way that the address is senator, so no need to check both)
- 5) in grantVoteCredit the totalCreditRewards of masterchef are based on the last time updatePool was called, so will be not up to date but reflect the amount of token credited with the last update pool, I recommend to add a mass update before calling the totalCreditRewards
- 6) grantVoteCredit should be called before each add and remove of a senate to prevent wrong calculations (if there is a user added at time 0, and one added at time 20 and I call grantVotingCredit at time 25, both user1 and user2 would get same amount of credit)
- 7) vote are managed with an array, and since user would end up having a lot of votes during the lifespan of the protocol each time they vote the gas cost increase due to the forloop in both vote and remove vote. I recommend to implement a second data structure parrallel to the already existing array made by a mapping of: voter(address) -> proposalId(uint256) -> bool.

The array will be used to check if a user have 0 vote active before selfReplace, and also for giving out the list of votes, the mapping will be used to prevent to check the entire array to see if a vote is present or not.

AddVote) remove the for with the require and just put
require(senVoteMap[msg.sender][proposalId], "already voting")
after the push of the proposal id also put true in the mapping ->
senVoteMap[msg.sender][proposalId] = true.

RemoveVote) add require(!senVoteMap[msg.sender][proposalId], "non existing vote") at the start, before the emit RemoveVote add senVoteMap[msg.sender][proposalId] = false.

8) in removeVote add a function parameter with the index of the position of the vote in the array, check it with the require and remove the array like in expellSenato and selfReplaceSenator, an alternative is to store the index in the mapping of the votes, but also need to change that value each time the proposal is moved from last position to the deleted one

9) in initializeSenators you only push the address in the array but not change the value to true in isSenator and addedSenator

10) in multiCall the allVotes is returned as a monodimensional array, there is no way of creating nested arrays or dynamic multidimensional array in solidity, moreover where you putted (senators.length) you were setting the lenghts of the second dimension that should have been the number of votes, but anyway every value you put there that is greater than 0 make the code output the same, for optimization you could put 1, but since is just a view function there is no point in that XD, (PS, you have to put a massupdate also here before the totalCreditRewards())

Security Analyst, [10/11/2023 1:20 AM]

Code Improvments:

- 1) all the place you reuse more times a require could be turned in a modifier for more clarity
- 2) in expellSenator I would make the check for at least 50% with /2 instead of *50/100 like in expandSenate and other functions.
- 3) this code is used in: addSenator, expandSenate and initializeSenators(if fixed) and can be putted in a private function called something like _addSenator,
senators.push(_newSenator);
isSenator[_newSenator] = true;
addedSenator[_newSenator] = true;
- 4) in the selfReplaceSenator the line inside the for where you make the addedSenator = true could be put outside near the setting of isSenator.
- 5) didn't understood the comment at line 120, could be a old comment that should be removed
- 6) you should replace i++ with ++i in lines: 74, 95, 114, 126, 138, 178, 197

Events:

- 1) in AddSenator senator should be indexed
- 2) in RemoveSenator senator should be indexed
- 3) in AddVote, both voter and proposalId should be indexed
- 4) in RemoveVote, both voter and proposalId should be indexed
- 5) you should create an event for updating the min/max of senate, every time you change an important value in the contract you should emit an event

Security Analyst, [10/11/2023 1:22 AM]

I might not have described the problems in the best way, for anything you don't understand just ask me and I will explain it in a more clear and detailed way

Security Analyst, [10/11/2023 12:18 PM]

what would be the treasury contract? Not the trasury wallet but the tresury one

developer,[10/11/2023 12:20 PM]
i think in consensus (is where treasury can be managed thru)

if that is what you mean

Security Analyst, [10/11/2023 12:21 PM]
so, in token vault you have a treasury that is the masterchef feeAddress and a treasurywallet that is the treasury wallet on the governorator

Security Analyst, [10/11/2023 12:22 PM]
but don't think that the senate will be set as feeAddress on masterchef

developer,[10/11/2023 12:22 PM]
ahh i see

in masterchef the treasury is the governing contract

Security Analyst, [10/11/2023 12:23 PM]
I'm even more confused now XD

developer,[10/11/2023 12:23 PM]
penalties for the token(XPD) go to the "treasury" in masterchef

Security Analyst, [10/11/2023 12:23 PM]
so, in the mining one you pay the token to the governorator contract?

developer,[10/11/2023 12:23 PM]
and fees(deposit fee) and funding fee go to the treasuryWallet

developer,[10/11/2023 12:23 PM]
yes

Security Analyst, [10/11/2023 12:24 PM]
so in mining you pay the governorator, while in the token vault you pay the treasury wallet?

developer,[10/11/2023 12:24 PM]
yeah exactly

Security Analyst, [10/11/2023 12:24 PM]
okee

developer,[10/11/2023 12:25 PM]
from the governing contract the tokens(XPD) can be deposited into treasury wallet or burned
the treasuryWallet has the process for withdrawing tokens to any address (whereas governorator can only burn or to send them out, they must be sent to treasury first)

Security Analyst, [10/11/2023 12:25 PM]
I will have for sure to check governorator and treasurywallet contract to see if tokens don't get stuck in them

Security Analyst, [10/11/2023 12:26 PM]



Security Analyst, [10/11/2023 12:27 PM]

Is good practice to call the contracts in the same way everywhere to avoid confusion, keep it in mind for next time

developer,[10/12/2023 9:33 AM]

Changed 1,2,3,4

5.) i didn't change because it does not matter that much IMO. You can always massUpdatePools and then call grantVotingCredit to get the remaining credit

6.) I'll leave it as is. It's a public function, the community should make sure the credits are given and functions called (else it will not be 100% correct, but no big deal)

7.) there will not be many votes, as Senate is mainly to just vote against token voting if needed... so i'll leave it as is

8.) Fixed and done similarly as in expellSenate and SelfReplaceSenator

9.) Fixed

10.) if it doesn't make a difference ill jsut leave it

Improvements;

1.) Will leave as is

2.) changed

3.) modified as suggested

4.) i think irrelevant after for loop has been removed for the change

5.) removed

6.) updated

changed all the events accordingly

Security Analyst, [10/12/2023 11:04 AM]

about 6 depends on how frequent that function is called and could have a drastically wrong calculation since if I join 1 year later than other senate and no one called that function I get same rewards as them for the time I wasn't a senate, but up to you decide of changing it or not (I still think is fairly important to change)

Security Analyst, [10/12/2023 11:06 AM]

what does the contract sync do?

developer,[10/12/2023 11:11 AM]

it updates parameters

developer,[10/12/2023 11:12 AM]

it doesn't do anything it just updates owners in all contracts and so on

Security Analyst, [10/16/2023 1:03 AM]

still watching the token vault (almost finished), wanted to ask this, the masterchef owner(from ownable) and the masterchef feeAddress are both the same address of the governing contract right?

developer,[10/16/2023 1:06 AM]

yeah

Security Analyst, [10/16/2023 1:06 AM]
good

Security Analyst, [10/16/2023 1:06 AM]
in a while I will give you the vault report

Security Analyst, [10/16/2023 1:07 AM]
I took quite a lot because there was some math stuff to check and took some time for those

Security Analyst, [10/16/2023 1:37 AM]
so, about the vault contract.

I found 2 very important issue that in some way are also connected.

Funding fee:

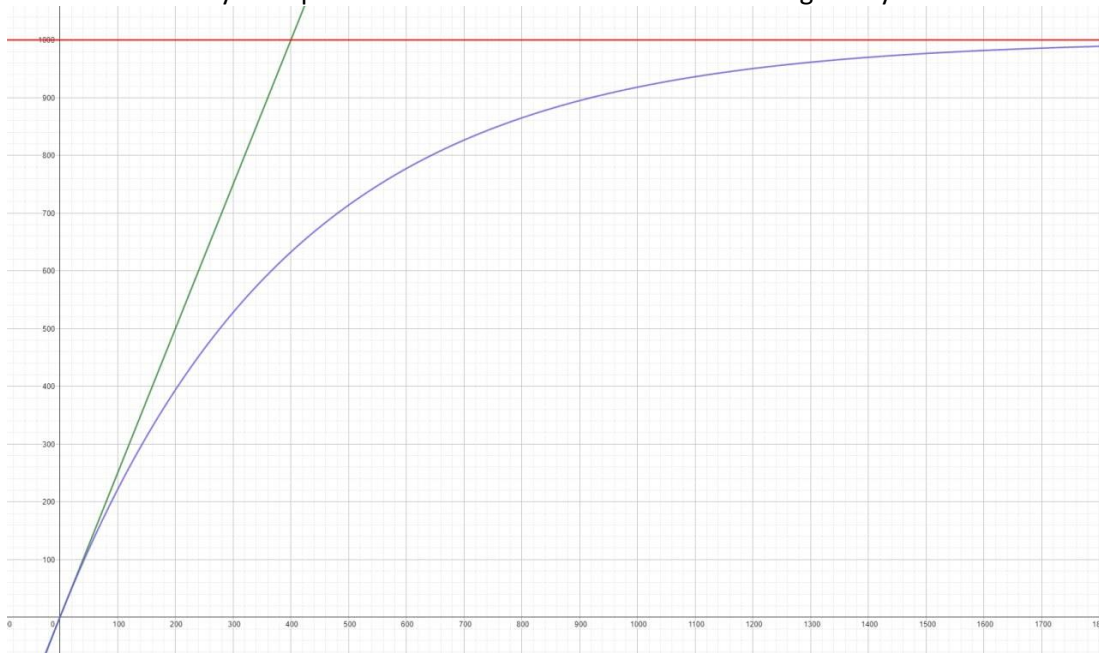
The problem is that you multiply the fee for the time that is wrong, you should use the formula of compound interest, in your case $\text{capital} \times (1 - \text{fundingFee})^{\text{hours}}$. The problem of not using the compound formula is that you can exceed 100% of fee making the pay fee reverting always and stuck tokens forever. While with the compounding formula you apply the fee everytime at the new capital so it can never go below 0.

Example: 0.25% fee, 2400 hours (100 days)
Your comission calculation: 600% comission
Compounded: 99.75% comission.

Since there is no fraction in solidity and there is no easy way to compute the compound interest I would reccomend, (if you want to keep the funding fee), to leave it as you did with the multiplication but cap it at like at max 20% (normal interest and compound interest increase similarly for small values)

Security Analyst, [10/16/2023 1:38 AM]

here there is a graphic example, red line is 100% of fee, x axis are the hours, the green one is how you implemented while the blue should be the right way



Security Analyst, [10/16/2023 1:38 AM]

if you take a look up to 20% they are kinda near the 2 functions, more you go from 20% more inaccurate it is

Security Analyst, [10/16/2023 1:49 AM]

Second main problem:

you store the user debt not for a single share but multiplied by the amount, the problem is when you decrease the ammount with the payfee function, this can easily make the debt greater than the value you calculate from amount * shares etc. My first idea was, in function like withdraw and selfHarvest to pay the fee after calculating the credits, so you use the full amount but that doesn't work because in collectComission and similar function you still call payFee reducing the amount. You have 2 options:

1) in withdraw put payfee after getting the userTokens, in selfHarvest, you would put it after the `_toWithdraw+=` but before the debt part, and in collectCommission and collectCommission auto you recompute the debt dividing by the total amount before the payFee and multiply for the new amount.

2) imo way more simple and cleaner, you store the the dept not multiplied by the staked amount (because of that you also remove the `/1e12`), and change the formula where it is used, for example:

```
uint256 currentAmount = userTokens * accDtxPerShare / 1e12 - user.debt;
```

Become:

```
uint256 currentAmount = userTokens * (accDtxPerShare - user.debt) / 1e12;
```

Security Analyst, [10/16/2023 2:15 AM]

now for the rest in the old fashion way:

Problems:

1) maxFundingFee it is actually 0.25% instead of 0.025% as commented

- 2) in harvest you do $\text{accumulatedRewards} = \text{lastCredit} - \text{currentCredit}$, but you should invert them, the current credit are higher than the last saved since masterchef emit credits
- 3) in case of depositing while the contract have accredited 0 credits the user get 0 shares, I recommend before the first deposit to have at least already accredited 10 credit($10 * 1e18$) to the contract to have the share with better precision
- 4) in withdraw there is the code that accredit some points to both the user than the one in the ref, but no points if there is no ref, keep in mind that user can just put a random address just to get the points for themselves (idk if is already considered or not)
- 5) in collectCommission you putted $i++$ instead of $++j$ in the inner loop (watch out next time you do nested loop, double check always)
- 6) in updateSettings you should put a require that the new value is less or equal to 10_000 that would be 100%

Code Improvements:

- 1) remove openzeppelin Address import, unused
- 2) remove IVoting.sol import, unused
- 3) UserSettingUpdate event can be removed, unused
- 4) in the constructor where you have all the pools minserve time you could add as comment the time in days
- 5) in the deposit function you could put the 2 line of $\text{amount} = \text{amount} - \text{deposit fee}$ and the following safeTransfer inside a if that have $\text{depositFee} != 0$ (no need to spend gas for transferring 0 tokens XD)
- 6) harvest and selfHarvest have similar names even if they are not related at all, I would consider to change the harvest name with something like update or whatever
- 7) in withdraw you can skip the publish token to treasure if penalty is = 0 (in case of the bigger pool)
- 8) add nonReentrant to selfHarvest, emergencyWithdraw
- 9) in selfHarvest you use the toWithdraw variable name in the opposite way of how you used it in withdraw, recommend to use same code and save variable name for same stuff to avoid confusion
- 10) substitute $i++$ with $++i$ in the following lines: 205, 257, 266, 346
- 11) in withdrawStuckTokens you can remove the: `IGovernor(IMasterChef(masterchef).owner()).treasuryWallet()` and replace it with: `treasuryWallet`
- 12) I don't see any use of having the `calculateTotalPendingDTXRewards`
- 13) in virtualAccDtxPerShare I would add parenthesis to avoid reading mistake of the formula, so from: `return (accDtxPerShare + _pending * 1e12 / stakeToken.balanceOf(address(this)));` to return `(accDtxPerShare + (_pending * 1e12 / stakeToken.balanceOf(address(this))));`

Events:

- 1) in Deposit event referral should be indexed,
- 2) in selfHarvest harvestInto should be indexed
- 3) in CollectedFee from should be indexed
- 4) you should add an event for collecting fee
- 5) you should add an event for updateFees
- 6) you should add an event for UpdateTreasury
- 7) you should add an event for setMasterChefAddress
- 8) you should add an event for setPoolpayout (emit it also in the constructor)
- 9) you should add an event for updateSettings

Question:

Are the referral stuff read from other contracts?

developer,[10/16/2023 1:44 PM]

Yeah the referral stuff is read from this contract
<https://github.com/decentralizeX/PulseDAO/blob/main/contracts/referralRewards.sol>

developer,[10/18/2023 11:10 AM]

First main problem;
for the funding fee i added the IF condition if it exceeds 20% if that is okay;
if(commission > user.amount * 2 / 10) {
 commission = user.amount * 2 / 10;
}

developer,[10/18/2023 11:10 AM]

Second main problem;
What if i just recalculated user debt in the payFee function like i have done now in 424 line

developer,[10/18/2023 11:10 AM]

problems;
1.) dividing it by 1million instead of 100k now to get 0.025%
2.) fixed. oopsie that was a big one

3.) i think this is okay? The amount he deposits = shares. The debt will be 0, as it should be, or not?
IDK could be wrong

4.) Yeah i know
5.) changed
6.) added

Improvements;
done 1,2,3

leaving 4

changed 5

leaving 6 because else i would have to be changing on frontends and what not

in 7 added (if currentAmount > 0) to send to treasury

in 8 added nonReentrant

noted 9 but will leave as is since functionally doesn't matter

done 10

done 11

i think 12 was used on frontend (i know could go directly from chef, but ill leave it since the cost is negligible for deployment)

left 13 because its ok

added indexed where necessary, didn't add new events because it will be a rare occasion to change these things through "decentralizedVoting" and it would be known anyways(would have to vote to change the governing contract, in order to call the function)

developer,[10/18/2023 11:30 AM]

hmm or maybe the second main problem wouldn't be good as i said... because the tokens are sent to treasury (and this should change the accDtxPerShare...)

Security Analyst, [10/18/2023 11:33 AM]

In around 30 Min I should be able to check what you changed and recheck point 3 (I might get confused with the share pattern)

Security Analyst, [10/18/2023 11:33 AM]

So write any concerns and I will check then after

Security Analyst, [10/18/2023 11:44 AM]

yea seems, good, anyway I will check the code you changed when I recheck everything

Security Analyst, [10/18/2023 11:45 AM]

that is completely wrong because you are resetting all rewards since you are using the new accDtxPerShare and not old value

developer,[10/18/2023 11:53 AM]

yeah it would not work

Security Analyst, [10/18/2023 11:54 AM]

yea sorry for point 3, it actually work fine, I just had a moment of confusion since your contract is a mix of traditinal stacking one with a bit of autovault that use shares, so I misstook the debt for share

Security Analyst, [10/18/2023 11:54 AM]

if you want I can send you back the edited contracts with the solutions I proposed

developer,[10/18/2023 11:57 AM]

Ok.

solution 1.) seems more simple though or no?

But i think the problem is when it sends tokens to treasury, it reduces the contracts balance (and that would also change accDtxPerShare if it were updated)

developer,[10/18/2023 11:58 AM]

in collectCommission would have to call harvest() after payFee and then recalculate the debt

developer,[10/18/2023 11:58 AM]

i think

Security Analyst, [10/18/2023 11:58 AM]

I think I will check both a bit more in detail and check which one is less error prone

Security Analyst, [10/18/2023 11:58 AM]

and go for that one

Security Analyst, [10/18/2023 11:59 AM]
I will do it within today okay?

developer,[10/18/2023 12:00 PM]
ok.

i think collect commission would be like this then, but this might increase the fees quite significantly

```
function collectCommission(address[] calldata _beneficiary, uint256[][] calldata _stakeID) external
    nonReentrant {
    harvest();
    for(uint256 i = 0; i < _beneficiary.length; ++i) {
        for(uint256 j = 0; j < _stakeID[i].length; ++j {
        UserInfo storage user = userInfo[_beneficiary[i]][_stakeID[i][j]];
        payFee(user, _beneficiary[i]);
        harvest();
        user.debt = user.amount * accDtxPerShare / 1e12;
        }
        }
    }
```

Security Analyst, [10/18/2023 12:01 PM]
harvest isn't the problem

Security Analyst, [10/18/2023 12:02 PM]
also the "older" accDtxPerShare is the value that determine the rewards, so if you just recalculate the debt with new accDtxPerShare you just reset rewards

Security Analyst, [10/18/2023 12:02 PM]
anyway leave it as it is, I will fix it and send you the contract probably by this evening and tell you what and why I did it

Security Analyst, [10/18/2023 8:44 PM]
about the secondary contract I started to watch I finished the governor and this are the result:

First of all there is a lot of unused variable that I think you putted to read from other contract, but the not used inside the contract are: creditContract, nftStakingContract, nftAllocationContract, nftWallet, senateContract, senateContract, nftStakingPoolID, plsVault, plsxVault, incVault, hexVault, tshareVault, tokenDistributionContractExtraPenalty

Moreover those are putted as immutable, but since you write them in the code you can put them as constant.

Problem:

- 1) no way to update masterchef address, so if you change masterchef you also have to redeploy governor
- 2) isInflationStatic is never used and is not possible to update its value
- 3) allocationPercentages, their sum don't add up to 100%, not that is big problem tho. the problem is in another point I would explain after

4) in rebalancePools the rebalance in the sense of getting the pool give the right rewards work, but the sum of the allocation points of those is not constant, since that number represent the % that the pools get of the total emission rate the not consistency of that value will change a lot the % of rewards that go to the pools against the % that goes to for example vaults etc. So can happen that at start 70% of emission go to the pools, but with a rebalance that number go down to like 30% of total emission.

To fix it just write down a sum that all those pools should have as allocation, could be something like 10k, and after you calculate the value as you did, you sum them and do $*10000 / \text{sum}$, so you decrease/increase them until their sum is 10k that would be always constant.

5) stakeRolloverBonus work on a set of fixed pools, in case of adding a pool/changing them you have to change governor

6) idk yet if is a problem or not, but basic contract can add pools but not change the allocation points, that could be done only through the farmContract that I still have to read and idk if I will do so.

7) in transferToReferralContract the require check that 2 months have passed not that they still have to pass, check the >

8) in changeGovernorForSecurityPriorMintingBegins you just change the owner on masterchef but you forgot to update the other 2 "governor address copy" that are feeAddress and devAddress

9) SUPER IMPORTANT, there is no way in the governor to setTrustedContract of the masterchef, that lead to not be able to have any contract that can call burn on masterchef

Code Improvement:

- 1) IGovernor, IVoting, IRewardBoost, IVault import are unused and can be removed<
- 2) all immutable variable if not setted in constructor can be putted as constant, more gas efficient
- 3) treasuryWallet and nftWallet can be putted as constant
- 4) put mintingPhase, depositFee and fundigRate explicitly as public
- 5) since you already pass to the constructor the addresses of the pools why you don't set them there also without manually hardcoding the addresses?
- 6) no need to change but.... you use 3 variable for the same address inside the masterchef contract....
- 7) in treasuryRequest why is treasury wallet payable? you just call a function on it not transferring chain tokens.

Events:

- 1) TransferOwner unused
- 2) EnforceGovernor should have _newGovernor as indexed and could also include old owner as an extra indexed parameter
- 3) In GiveRolloverBonus recipient and poolInto should be indexed
- 4) setActivateFibonaccening, setThresholdFibonaccening, updateDelayBeforeEnforce, updateCostToVote, updateRolloverBonus, updateVault should have an event

Question:

Is havest intended to give fee to users? Is like a you pay gas for rebalancing and you get the fee? in stakeRolloverBonus, when you call it from the mining contract that contract read from the governor the bonus value and pass it to the funcion of the governor, so you like read a value to governor to give it back to him XD, why you don't just read it inside governor?

developer,[10/18/2023 11:14 PM]

Question answer;

yeah harvest gives user reward for rebalancing (so that it stays accurate and they keep calling)

I guess it would have been better the way you said it, but works either way and since mining contracts are already deployed won't be changing

Security Analyst, [10/19/2023 12:14 AM]

I did this version, check the difference. I will recommend you to deploy on testnet and see if it works as expected and don't have revert problem for oversights errors, anyway I will recheck the code when I recheck everything

Security Analyst, [10/19/2023 12:16 AM]

PS also fixed that you accidentally removed a ")" in the for where there is the ++j

developer,[10/19/2023 1:19 AM]

okay changed it as such... will deploy tomorrow and test

the plsVault is near identical to the token vault, only difference being it transfers native token (payable instead of transferring ERC20 and address(this).balance instead of ERC20 token balance <https://www.jstoolset.com/diff/7278b82959009cf2>)

but... My problem is . How does deposit work in this instance

because harvest() has to be called BEFORE the tokens(native balance in this case) are transferred. How do you do this in a payable function?

```
function deposit(uint256 _amount, address referral) external payable nonReentrant {
    require(msg.value == _amount && _amount > 0, "invalid amount");
    _harvest(_amount);

    if(referredBy[msg.sender] == address(0) && referral != msg.sender) {
        referredBy[msg.sender] = referral;
    }

    stakes.pop();
}

function _harvest(uint256 _etherReceived) private {
    IMasterChef(masterchef).updatePool(poolIID);
    uint256 _currentCredit = IMasterChef(masterchef).credit(address(this));
    uint256 _accumulatedRewards = _currentCredit - lastCredit;
    lastCredit = _currentCredit;
    accDtxPerShare += _accumulatedRewards * 1e12 / (address(this).balance - _etherReceived);
}
```

developer,[10/19/2023 10:34 AM]

solved it like this, created private harvest function that deducts the Ether transferred from the balance... think should be it

developer,[10/19/2023 11:03 AM]

Changed the contracts to constants, they are there to have all addresses at one place

problems;

1.) Yeah it's as intended. To change the masterchef we need to change the governor

- 2.) removed it because it's not used anywhere else (old code)
- 3,4) I THINK this is solved in the farms contract (rebalancePools() in farms changes the other pool allocations to match the percentages). So first you would rebalance in governor, and then rebalance in farms as well. I think this works, not sure?
- 5.) Yeah to add new pool, we would have to do it through new governor
- 6.) Yeah that's fine, pool is added in basic, and the allocation is managed through farms
- 7.) oopsie , Corrected to <
- 8.) It's okay. Because i have to change the address to my own to make the changes, and then transfer all of them again to the new governor
- 9.) It's okay, the trustedContracts are set manually and then the ownership is transferred to governor

code improvements;

- 1.) removed
- 2.) changed
- 3.) changed
- 4.) changed
- 5.) changed
- 6.) yeah idk lol.. Won't change
- 7.) Don't know , removed

events;

- 1.) removed
- 2.) indexed _newGovernor. (old owner is this contract)
- 3.) added
- 4.) they have an event in the other contracts,so will leave as is

Security Analyst, [10/19/2023 1:44 PM]

I will check it after and I will let you know

Security Analyst, [10/19/2023 1:45 PM]

yea seems good, if you want I will check that contract than I will recheck vaults ones

Security Analyst, [10/19/2023 1:48 PM]

about 3/4 I will keep it in mind if I manage to also see farm contract, but don't think I have time to do so before the 25

Security Analyst, [10/19/2023 1:49 PM]

after the 25 you need other stuff to be reviewed or everything have to be done for the 25?

developer,[10/19/2023 2:01 PM]

it's till 29th-30th actually

in general would need everything yeah, but maybe i can extend it for a week more if necessary

Security Analyst, [10/19/2023 2:05 PM]

if you want with 1 extra week I should be able to give also a good check with all secondary contract as I did with governor

Security Analyst, [10/19/2023 2:20 PM]

let me know what you prefer because if you want me to give a fast check also to the secondary contract I postpone the check of contracts interactions

Security Analyst, [10/19/2023 8:05 PM]

About votingCredit:

Problems:

1) in constructor you set the creditingContract of airdropContract and airdropContractLocked to true, but you use the internal variable and not the args one, so you are actually settings the crediting contract of zero addresss to true

Improvements:

- 1) Context import can be removed
- 2) aridropContract and airdropContractLocked can be putted as immutable
- 3) in both modifyCreditingContract and modifyDeductingContract you can put the ++ and — before the variable to less gas cost

Events:

- 1) in SetCreditingContract _contract should be indexed
- 2) in SetDeductingContract _contract should be indexed
- 3) in BurnCredit burnFrom and forId should be indexed

Security Analyst, [10/19/2023 8:05 PM]

also checked treasuryWallet and is all fine

Security Analyst, [10/19/2023 8:23 PM]

about this? I need to know to understand how to schedule my next days and work

developer,[10/19/2023 8:25 PM]

Until 30th it's for sure

developer,[10/19/2023 8:26 PM]

There is time until end of month. Think more is needed?

developer,[10/19/2023 9:23 PM]

aw shiet good find, this would have caused big problems

won't be changing contract becasue it's already launched, will just manually add these two as crediting contract

developer,[10/19/2023 9:29 PM]

Ahh nevermind it would have worked anyways because it uses .airdropVotingCredit() and not addCredit() in the actual airdropContract

developer,[10/19/2023 9:36 PM]

made the changes as recommended, will be relaunching that one

Security Analyst, [10/19/2023 9:39 PM]

for giving a check on all secondary is a bit tight but I will try to end before that, in case I will just left out trivial contracts

Security Analyst, [10/19/2023 10:59 PM]

can you make me a list of secondary contract I should check?

developer,[10/19/2023 11:15 PM]
referralRewards > rewardBoost > farms > basicSettings

Security Analyst, [10/19/2023 11:20 PM]
Already doing basicSettings

developer,[10/19/2023 10:34 AM]
solved it like this, created private harvest function that deducts the Ether transferred from the balance... think should be it

developer,[10/19/2023 11:03 AM]
Changed the contracts to constants, they are there to have all addresses at one place

problems;

- 1.) Yeah it's as intended. To change the masterchef we need to change the governor
- 2.) removed it because it's not used anywhere else (old code)
- 3,4) I THINK this is solved in the farms contract (rebalancePools() in farms changes the other pool allocations to match the percentages). So first you would rebalance in governor, and then rebalance in farms as well. I think this works, not sure?
- 5.) Yeah to add new pool, we would have to do it through new governor
- 6.) Yeah that's fine, pool is added in basic, and the allocation is managed through farms
- 7.) oopsie , Corrected to <
- 8.) It's okay. Because i have to change the address to my own to make the changes, and then transfer all of them again to the new governor
- 9.) It's okay, the trustedContracts are set manually and then the ownership is transferred to governor

code improvements;

- 1.) removed
- 2.) changed
- 3.) changed
- 4.) changed
- 5.) changed
- 6.) yeah idk lol.. Won't change
- 7.) Don't know , removed

events;

- 1.) removed
- 2.) indexed _newGovernor. (old owner is this contract)
- 3.) added
- 4.) they have an event in the other contracts,so will leave as is

Security Analyst, [10/19/2023 1:44 PM]
I will check it after and I will let you know

Security Analyst, [10/19/2023 1:45 PM]
yea seems good, if you want I will check that contract than I will recheck vaults ones

Security Analyst, [10/19/2023 1:48 PM]
about 3/4 I will keep it in mind if I manage to also see farm contract, but don't think I have time to do so before the 25

Security Analyst, [10/19/2023 1:49 PM]
after the 25 you need other stuff to be reviewed or everything have to be done for the 25?

developer,[10/19/2023 2:01 PM]
it's till 29th-30th actually

in general would need everything yeah, but maybe i can extend it for a week more if necessary

Security Analyst, [10/19/2023 2:05 PM]
if you want with 1 extra week I should be able to give also a good check with all secondary contract
as I did with governor

Security Analyst, [10/19/2023 2:20 PM]
let me know what you prefer because if you want me to give a fast check also to the secondary
contract I postpone the check of contracts interactions

Security Analyst, [10/19/2023 8:05 PM]
About votingCredit:

Problems:

1) in constructor you set the creditingContract of airddropContract and airdropContractLocked to true, but you use the internal variable and not the args one, so you are actually settings the crediting contract of zero addresss to true

Improvements:

- 1) Context import can be removed
- 2) aridropContract and airdropContractLocked can be putted as immutable
- 3) in both modifyCreditingContract and modifyDeductingContract you can put the ++ and — before the variable to less gas cost

Events:

- 1) in SetCreditingContract _contract should be indexed
- 2) in SetDeductingContract _contract should be indexed
- 3) in BurnCredit burnFrom and forId should be indexed

Security Analyst, [10/19/2023 8:05 PM]
also checked treasuryWallet and is all fine

Security Analyst, [10/19/2023 8:23 PM]
about this? I need to know to understand how to schedule my next days and work

developer,[10/19/2023 8:25 PM]
Until 30th it's for sure

developer,[10/19/2023 8:26 PM]
There is time until end of month. Think more is needed?

developer,[10/19/2023 9:23 PM]
aw shiet good find, this would have caused big problems

won't be changing contract because it's already launched, will just manually add these two as crediting contract

developer, [10/19/2023 9:29 PM]

Ahh nevermind it would have worked anyways because it uses `.airdropVotingCredit()` and not `addCredit()` in the actual `airdropContract`

developer, [10/19/2023 9:36 PM]

made the changes as recommended, will be relaunching that one

Security Analyst, [10/19/2023 9:39 PM]

for giving a check on all secondary is a bit tight but I will try to end before that, in case I will just left out trivial contracts

Security Analyst, [10/19/2023 10:59 PM]

can you make me a list of secondary contract I should check?

developer, [10/19/2023 11:15 PM]

`referralRewards > rewardBoost > farms > basicSettings`

Security Analyst, [10/19/2023 11:20 PM]

Already doing `basicSettings`

Security Analyst, [10/20/2023 1:20 AM]

About the `basicSettings` contract:

Problems:

- 1) in `initiateNewPool` there is missing the `deductCredit` call
- 2) if you add a new pool the `executeSetCallFee` will only update the fee on the 6 base one and not in the new ones

Improvements:

- 1) you can remove the `Context` import
- 2) in `initiateSetMinDeposit` you can bring outside the `deductCredit` from the 2 if branch since you deduct same amount
- 3) some of the require and common code could be putted in a function modifier to simplify a bit the code
- 4) some execute function have `<=` in the require and some `<` and order of summed variable not the same

Events:

- 1) in `ProposeMinDeposit`, `DelayBeforeEnforce`, `InitiateSetCallFee`, `InitiateRolloverBonus`, `ProposeNewPool` and `ProposeSetMinThresholdFibonaccening` `proposalId` and `enforcer` should be indexed
- 2) in `InitiateSetCallFee` and `InitiateRolloverBonus` I would call the `depositingTokens` `valueSacrificedForVote` like in the others, or vice versa
- 3) in `InitiateRolloverBonus` also `forPool` should be indexed
- 4) in `AddVotes` and `EnforceProposal` `_type`, `proposalId` should be indexed aswell
- 5) `updateNewPoolProposalThreshold` should implement an event

extra:

1) line 447 you wrote can 2 times in the comment XD

Security Analyst, [10/20/2023 10:56 AM]

also there is some places where you put == true, and you can remove it,
lines: 129, 194, 266, 276, 399, 495, 513

Security Analyst, [10/20/2023 11:37 AM]

about referralRewards:

Problems:

- 1) in redeemRewards everything is based only on plsVault problems so if you update that the change effect every pool, not a big deal since everyone should have same values.
- 2) you have the withdrawTokens function callable only from governor, but governor don't have a way to call such function

Improvements:

- 1) ++i in lines: 74, 95

events:

- 1) ClaimReferralReward should have claimInto indexed

developer,[10/21/2023 12:14 PM]

Problems;

- 1.) added the deductCredit call
- 2.) that's ok

Improvements;

- 1.) removed
- 2.) DONE
- 3.) mehh, leaving as is
- 4.) changed all to <

Events;

- 1.) indexed
- 2.) i think would be better too, but then i would have to change everything on the frontends and what not so keeping it as is added 3. and 4.
- 5.) not needed imo because to do it, we need to change the governing contract which is like a big deal

developer,[10/21/2023 12:17 PM]

problems;

- 1.) yeah they are all the same so had to pick one, will leave as is
- 2.) In case we wanted to withdraw the tokens, it would happen thru launching new governor yes

added the rest

developer,[10/23/2023 11:00 AM]

did deploy tokenVault.sol on testnet.

anything in particular i should be testing?

Security Analyst, [10/23/2023 11:21 AM]

I would take 2 accounts and do 2 deposit in different time, harvest some fee and after withdraw with both, maybe one emergency idk, just check that nothing revert and the balance are right after the withdraw, so starting balance + some extra credits

Security Analyst, [10/23/2023 11:48 AM]

About the rewardBoost

Problems:

- 1) isn't better burning the tokens at the fibonaccening start? Doing so you reduce the supply before the "print" event, so you are more far to reach supply cap
- 2) you don't set the credit contract in the constructor so you have to call the sync before using the contract
- 3) since the fibonaccening end occur when someone call the end function the event can last a bit more blocks than the duration, in case of a proposal with very low amount of blocks and a high rewardsPerBlock, like 1 block and 5/100 of the virtual supply as rewardPerBlock a higher amount than the limit can be minted. I recommend to put a minimum number of block in the proposeFibonaccening, like at least 100 blocks or whatever.
- 4) about the number of block calculated for 30 days. there is the /10.1 and the *2592 that confuse me, because in case the 2592 are the block for each seconds (I don't think so), why you divide by 10? Instead, if you divide by 10 because there is a block each 6 seconds why you multiply it by 2592?
- 5) in the leverPullFibonaccening, where you have the if that check the emission is <= of 1618e16, you do it after updating the emission with the proposal value that I don't think was what you wanted, since I think you want to check it against the normal emission and not the event emission you can compare it to the remember reward in the governor or just move that code before setting the new inflation value.
- 6) in setMasterchef you miss-use the IMasterchef interface, moreover token is already an address so no need to cast it as address. Instead you should use the IDTX interface and write:
IDTX(token).owner();
- 7) in calculateUpcomingRewardPerBlock in the branch where the grandFibonaccening expired you get the value of the virtualSupply, that value comprehend the 1e18 for the decimals and you don't take count to them, so the final return value is in the order of 1e36, instead of the expected 1e18

Improvements:

- 1) remove the context and Iconsensus import, unused
- 2) remove goldenRatio, unused, but if you keep to read it outside put it as constant
- 3) remove lastCallFibonaccening and targetBlock, unused
- 4) you can remove the == true in the requires at line 158 and 141.

Events:

- 1) in ProposeFibonaccening, EndFibonaccening, CanceFibonaccening the proposalId should be indexed
- 2) rebalanceInflation is never used
- 3) in AddVotes and EnforceProposal _type and proposalId should be indexed
- 4) ChangeGovernor is never used

developer,[10/24/2023 12:33 AM]

- 1.) changed to burn at start
- 2.) yeah it has to be synced because the contract doesn't exist yet (this one is launched first)
- 3.) set minimum 100 blocks

4.) set amountOfBlocks to 256633
5.) You are right. Moved this to the endFibonacci function
6.) you are correct, changed
7.) i divided by 1e18, if that's okay
uint256 supplyToPrint = initialSupply * _factor / 1e18 / 100000;

done improvements

modified events
and updated events

Security Analyst, [10/24/2023 12:42 AM]
let me check how you changed everything

Security Analyst, [10/24/2023 12:43 AM]
about the 5 you did it wrong

Security Analyst, [10/24/2023 12:44 AM]
because the condition to reach that code is that the grand fibonacci is expired, but exactly that code
set it to expired XD

Security Analyst, [10/24/2023 12:44 AM]
moreover you need that value before calling calculateUpcomingRewardPerBlock()

Security Analyst, [10/24/2023 12:45 AM]
put that if code block to like 154 and you will be fine

Security Analyst, [10/24/2023 12:46 AM]
the rest seems good 👍

Security Analyst, [10/24/2023 12:46 AM]
Tomorrow I finish the farm one, after that I recheck everything and if I end some time extra before
the 30, I will just do a fast look at the remainings contracts

developer,[10/24/2023 1:41 AM]
added it to 154 line but like this

I think this is more correct... So if on endFibonacci we would be below that threshold, we should
already be using the other formula for calculating reward

developer,[10/24/2023 2:20 AM]
do /token-distribution/ also because it's quite important

i think this one should be okay, but just to make sure. They are quite small and very similar

Security Analyst, [10/24/2023 11:51 PM]
about farms:

Problems:

1) at start every pool have 0 allocation expect the hex one right? because you only instantiate that
one

- 2) in the constructor you set the hex pool to 9000 that is greater than the maxPulseEcoAllocation (6000), tho I think is intended
- 3) in changeMaxAllocations you have lp that is unused as argument
- 4) rebalancePools is completely broken as function, the math is incorrect, in case of protocol pools allocation sum is over 1000 it cause all function to revert, it set the new allocation to the pulse-ecosystem pools without updating the poolAllocation mapping inside the contract and more important don't rebalance the mining one (I told you this problem before in another contract but you said that those were balanced here, but actually only ecosystem are balanced in this contract and not the mining one)
- 5) in initiateFarmProposal the argument depositFee is unused, moreover is always set to 0 in the struct and never read, so you can safely also remove it from the struct and events
- 6) in the proposeGovernorTransfer you could use timestamp 0 for transaction that you want that can be done instantly (otherwise you can put in the proposal that the given timestamp have to be > of the current + delay + enforce delay)
- 7) in executeGovernorTransfer I think that in case of burn token, they shouldn't be burned only if they are available and otherwise just consume the proposal but or burn all the token if they are less than the amount, otherwise revert the execute so it can be runned in a second moment and still burn the tokens
- 8) the proposeGovTax have a require that can be setted to 0 the amount, in some situation protocols put fee to 0, so feels a bit strange that can't be zero.
- 9) in rebalanceIfPulseAllocationExceedsMaximum the formulas are wrong and the function don't work as it should, moreover similars problems to the other rebalancing function
- 10) in reduceMaxPulseAllocation the reduction is based on assumption that at least one user call it regularly, and since the users don't have any benefits on calling it they will never do so, so you will be the only one calling it. So is bad to have written in a way that require a weekly manual trigger (even if not a security concern), in that case you could improve it a little getting the last timestamp and see how many weeks passed and with a for do the reduction the amount of week passed. When setting the lastReducePulseAllocation you just use the old lastReducePulseAllocation + 7day * weeks passed

Improvements:

- 1) you can remove Context import
- 2) you use only type 0,2,4,5 for the proposal, in the others you used to do 1,2,3,4,5,6,etc
- 3) ++i at line: 111, 118, 126, 208, 433, 439
- 4) in updateFarm you can move the code from line 200 to 202 inside the if since they are not used in the else branch
- 5) you can remove the == true at lines: 269, 279, 348, 358, 405, 414.
- 6) on line 364 in the executeGovTax the comment is wrong and should be removed
- 7) in reduceMaxPulseAllocation in the require of time you should put the value as 7days and I recommend to sum it to the lastReducePulseAllocation to make easier to read.

Events:

- 1) in InitiateFarmProposal proposalID and poolid should be indexed
- 2) you can remove ProposeRewardReduction
- 3) in ProposeGovernorTransfer you should put proposalID indexed
- 4) in ProposeGovTax you should put proposalID as indexed
- 5) in ProposeVault you should put ProposalID and type as indexed
- 6) in AddVotes you should put type and proposalID as indexed
- 7) in EnforceProposal you should put type and ProposalID as indexed
- 8) in InitiateFarmProposal remove the depositFee argument

Optional (you can just skip it): in this and similar contracts with proposal you could add an extra require to stop votes after start time + delay + enforcedelay to prevent user to waste token voting when the proposal is already enforced or can already be enforced

developer, [10/25/2023 12:40 AM]

About the 4.)

(pools 0-5 are mining pools, and above 5 are contracts; vaults)

They are supposed to be percentages

and it can't be more than 9000 (90%) in total so it should be okay?

As you said in the governing contract, the total allocation number is not fixed, so the allocations for pools 0-5 change, and especially their total allocation changes

So we rebalancePools in governor first and have the numbers... And then call rebalancePools in the farms contract (which gives the proper allocation that the pools(>5) should have)

Think the logic is correct, math could be wrong though idk?

First loop gets total allocation for pools 0-5

_percentageAllocatedToPulseEcosystem gets total percentage that should be given to pools >5 ... I guess we could keep a variable also, instead of running loop

So then we calculate what percentage is given to miners, in order to calculate what will be our new total allocation

and then it calculates and updates the actual allocation number for each pool >5

Security Analyst, [10/25/2023 12:44 AM]

About the distributionExtraPenalties:

Problems:

1) in setMerkle use msg.sender instead of origin. Origin could be dangerous if you interact with a malicious contract that use your origin to call this function (even if rare you will never know)

Improvements:

- 1) remove IERC20 import
- 2) put the claim_period_days in days and remove the * 86400 in the endAirdrop
- 3) put directPayout as constant
- 4) in claimAirdrop at the start add a requirement that the proof.length is > 0
- 5) in claimAirDrop, use do "_claimAmount * payout[claimInto] / 10000" 3 times, just calculate one and put it in a variable

Events:

1) in redeemCredit put user and withdrawInto as indexed

Security Analyst, [10/25/2023 12:47 AM]

the total allocation of mining one as I said is not a constant value, so can be 100 as 1567464. And in no contract you adjust it, so if you have the other ecosystem fixed to 9000 it is like nothing compared to 1567464.

Moreover the governor ONLY balance the mining pools between themselves so the total is variable that lead to have a greater or smaller % respect to the rest of the ecosystem.

About the math that yea, is completely wrong, just put some dummy values and you can see how it can also reach negative values as allocations.

Security Analyst, [10/25/2023 12:48 AM]

the best idea as I proposed is to have a fixed total allocation for mining and use it when you set them in the governor, and for the ecosystem just fix the math and you should be done

Security Analyst, [10/25/2023 12:49 AM]

up to you if you want to fix those function and I will check again or if you prefer that I write you a draft of the fixing code that you will check afterwards

Security Analyst, [10/25/2023 12:49 AM]

in case let me know and I will probably do it tomorrow

Security Analyst, [10/25/2023 12:54 AM]

about this I highly recommend to test the contract with the script that generate the proof to check if also the script that generate the proff and the merkle tree work correctly

Security Analyst, [10/25/2023 12:56 AM]

Tomorrow I will also finish the distributionToContributors and let you know

developer,[10/25/2023 12:56 AM]

Yeah because it's not a constant, we use percentages in farms?

It calculates how much is total for pools >5 ... Let's say it's 30% (3000)

So if it's 30% for non-mining pools, then it's 70% for mining pools

And if mining pool has for example 50000 allocation, and we want that to be 70%, the new total allocation is 71429

And now that we have new total allocation, for each non-mining pool we give it proper allocation Say pool 6 is 10%, so it get's 7142... And so on for others?

Or which part of this is incorrect?

developer,[10/25/2023 12:56 AM]

yeah i tested this

Security Analyst, [10/25/2023 1:00 AM]

didn't understood well this message, but in the contract you don't calculate what is written here, and moreover you don't set the allocation of mining contracts, and the non mining one have a cap

written to the contract, so for example if the mining one have a totAllocation of 100000000 but you want it to be 50% - 50%, sadly the ecosystem cap at 9000 so is not possible

Security Analyst, [10/25/2023 1:02 AM]

what I was saying is just chose a fixed total allocation for the mining like 10000 if you want something near a 50-50, and you will not that with the time the mining % will increase because with the time the 9k of the system decrease, but that will not get unbalanced between the 2 parts by the "rebalancing" that rebalance the 2 part singularly and not between them

developer,[10/25/2023 1:02 AM]

but the 9000 is the percentage(90%) not the allocation

Security Analyst, [10/25/2023 1:03 AM]

them you suppose a total of 10000 if you say that those 9000 are to be equal to 90%. But since the mining totAllocation can reach any value if for example is 100000, than 9000 is not 90% of 109000

Security Analyst, [10/25/2023 1:04 AM]

you are thinking them as percentage, but for being percentage the total HAVE to be fixed at 10k

Security Analyst, [10/25/2023 1:05 AM]

did you get what I meant?

Security Analyst, [10/25/2023 1:09 AM]

the fix options are 2:

easy one - put the mining fixed at 1000, and fix the farm maths. This with the decrease of the alloc to the ecosystem to get so slightly less decrease over what you intended since after a while will be 1000 over 9.8k instead of 10k

hard one - fix the math in farm rebalancing also in the correct way the mining pools, tho don't know what problem could cause the governor rebalance (have to check those). But to implement this in the right way is a lot more code

developer,[10/25/2023 1:14 AM]

it is fixed at 9000 (with maxPulseEcoTotalAllocation , and this number should decrease over time with reduceMaxPulseAllocation())

Security Analyst, [10/25/2023 1:15 AM]

Yea, only ecosystem part is fixed, that doesn't include the mining part that is not fixed

developer,[10/25/2023 1:17 AM]

the mining can reach any allocation but it doesn't matter, because the farms calculate for the percentages... so if the mining is 10,000 or 650000 it will give a percentage of that

Security Analyst, [10/25/2023 1:17 AM]

How does that?

Security Analyst, [10/25/2023 1:18 AM]

PS, if mining allocation is bigger than 10k the rebalancePools function revert

developer,[10/25/2023 1:18 AM]

first rebalancePools() in governor (would have to do that manually)

and then to get correct allocations for non-mining pools, call `rebalancePools()` in farms

developer,[10/25/2023 1:18 AM]
And it can't get over 9000 anyways

developer,[10/25/2023 1:19 AM]
`require(_pulseEcoCount <= maxPulseEcoTotalAllocation, "exceeds maximum allowed allocation for pulse ecosystem");`
on line 215

Security Analyst, [10/25/2023 1:20 AM]
Rebalance pools and governor can generate the mining total allocation to be whatever number, in `rebalancePools(farms)` lines 122, you do 1000 - farming allocation total, and that overflow

Security Analyst, [10/25/2023 1:20 AM]
Only the non mining part can't get over 9000....

Security Analyst, [10/25/2023 1:21 AM]
Just open an excel and try both function with some random number and you will get what I mean

Security Analyst, [10/25/2023 1:22 AM]
If you want I make you an example tomorrow with the code references

developer,[10/25/2023 1:24 AM]
the `_percentageAllocatedToPulseEcosystem` won't get over 9000

perhaps the variable is named improperly
`mapping(uint256 => uint256) public poolAllocation;`

this is actually for percentages

Security Analyst, [10/25/2023 1:25 AM]
I get now you you tried to do, tomorrow I will recheck so I can tell you more exactly were the function fails

developer,[10/25/2023 1:27 AM]
ok

developer,[10/25/2023 2:30 AM]
I am testing it on testnet and i think it's working correctly

I've choosen a poor name for `poolAllocation` in `farms.sol`

it's not actual pool allocation(like in masterchef) but `poolAllocationPercentage` ... and their total does not exceed 9000 (90%)

And during `rebalancePools`, it simply gives appropriate share of rewards the pool should get. Whatever the allocations will be set in governor, the farms contract rebalances for non-mining pools(above > 5) for how much it should be

developer,[10/25/2023 2:47 AM]

changed the variable name and also added a variable to keep track of the total (instead of looping and counting them)

will go through the rest of the problems tomorrow

developer, [10/25/2023 12:07 PM]

- 1.) Yes
- 2.) Yes it's intended
- 3.) removed
- 4.) ... check above
- 5.) i will leave it as is, because else i need to change the event listener, the scripts, the frontends, etc... I think the additional fee cost is negligible for this
- 6.) I think it doesn't matter? As long as timestamp is lower than current block.timestamp (or to be more exact, block.timestamp at call of execute) it can be triggered instantly, which is okay imo?
- 7.) added require statement to make sure there is sufficient balance
- 8.) Correct, removed requirement it must be higher than 0.
- 9.) Think it should work (similarly to 4)?
- 10.) Yeah you are right, but i will leave it as is. If users wish to call it they can, fees are low. It should be in the interest of the token holders to give less rewards to others

Improvements;

- done 1.)
- 2.) yeah, leaving as is.. (because i've been changing it over time)
- 3.) done
- 4.) done
- 5.) done
- 6.) removed
- 7.) done (block.timestamp - 7 days)

Events;

Indexed them, left the deposit fee

Security Analyst, [10/25/2023 5:03 PM]

yea, that name variable throw me off, and made me think you wanted to implement it as I had in mind, that is the traditional way where you have a fixed amount and fix the allocations accordingly. In the considering that percentage in base 10000 it works correctly the math. Only concern, that is why this is not the classic approach to implement such thing, is that you have no control about how they grow/shrink. So even if shouldn't be the case might be the possibility of certain pattern of rebalancing will lead to make the mining total allocation growth exponentially (possible overflow) or decrease in a way that the rounding while rebalancing the ecosystem (creates rounding inaccuracy)

Security Analyst, [10/25/2023 5:05 PM]

I also rechecked the rebalance if pulse allocation exceeds maximum, if I get it correctly you want that when called all allocation percentage summed reach the max amount right? In that case the formula is wrong, and in line 440 you should do $\text{poolAllocationPerc}[i] * \text{maxPulseEcoTotalAllocation} / \text{_totalAllocation}$ (tell me if sounds good to you)

Security Analyst, [10/25/2023 5:09 PM]

yea, 6 you can leave it as it is, just pointed out since I wasn't sure if you wanted it that way

developer, [10/25/2023 5:11 PM]

If the current total percentageAllocatedToPulseEcosystem exceeded by 3% for example, then i want to reduce all of the poolAllocationPercentages[*] by 3%

Security Analyst, [10/25/2023 5:12 PM]

but if for example the initial allocation was 9000 as the max, and after the max wen't down to 6000 you want all the percentage to sum to 6000 or what?

developer,[10/25/2023 5:13 PM]

yeah

Security Analyst, [10/25/2023 5:13 PM]

than if i'm not wrong also this time, the formula could work with small values, but don't with a bit more of difference

Security Analyst, [10/25/2023 5:15 PM]

if I follow this example with the current formula the end total allocation will be 4500 not 6000

Security Analyst, [10/25/2023 5:16 PM]

it kinda increse the distance to the new max by how far it was when you call the rebalance

developer,[10/25/2023 5:19 PM]

I don't know 😊 could be wrong formula

What i wanted to do was check by how much we exceed the maximum and if it's by 1%, simply reduce all by 1%

Security Analyst, [10/25/2023 5:20 PM]

if you just want to reduce them keeping the ratio between them but reducing the total sum to the new max you can just use this

Security Analyst, [10/25/2023 5:26 PM]

Security Analyst, [10/25/2023 5:26 PM]

I did this example with only one pull, what the formula actually do is this: if when called it the total is like 5% greater than the max amount, the new total will be 5% less than the new max amount

Security Analyst, [10/25/2023 5:27 PM]

to give you an extreme example, if you call it when the newMax is half the sum of the allocation, the new allocation perc of everyone will be 0 since will do the newMax - 100%

developer,[10/25/2023 5:29 PM]

what is _totalAllocation then

developer,[10/25/2023 5:29 PM]

i updated code

Security Analyst, [10/25/2023 5:29 PM]

the sum of the current total allocation, is the one you calculate in the first half of the code

developer,[10/25/2023 5:29 PM]

keeping it as percentageAllocatedToPulseEcosystem

Security Analyst, [10/25/2023 5:29 PM]

yea, I could have used old name XD

Security Analyst, [10/25/2023 5:31 PM]

used the name I saw here, it was before your last commits, so idk the new name

developer,[10/25/2023 5:39 PM]

```
poolAllocationPercentage[i] = poolAllocationPercentage[i] * maxPulseEcoTotalAllocation /  
percentageAllocatedToPulseEcosystem;
```

this is correct i think to get the ratio

$\text{maxPulseEcoTotalAllocation (new allocation) / percentageAllocated..(old allocation)}$

yeah... basically how you said it

developer,[10/25/2023 5:40 PM]

Your formula is correct but the `_totalAllocation` is now named `percentageAllocatedToPulseEcosystem`

developer,[10/25/2023 5:41 PM]

Security Analyst, [10/25/2023 5:41 PM]

yea seems perfect

Security Analyst, [10/25/2023 5:42 PM]

you could also remove the `exceed` and directly put `percentageAllocation = maxPulseTotalAllocation`

developer,[10/25/2023 6:06 PM]

true, updated

Security Analyst, [10/25/2023 11:38 PM]

about the `distributionToContributors`:

over the same stuff of the other similar contract there is these:

- 1) you could put the interface in a separate file and import it (up to you)
- 2) use the `_` to separate every 3 zeros where you have the big number with like 8 zeros
- 3) reading the `setMerkle` you use here, now I understand that in the other contract you don't call `setMerkle` by yourself but with this, so actually `msg.sender` is wrong, so you can revert to origin, or, that is better, check that the `msg.sender` is the `distributionToContributors` contract or passing it in the constructor or checking the governor

Security Analyst, [10/25/2023 11:39 PM]

In case there is other contract you would like me to check if there is time remaining after rechecking the one done up to now let them written down so I will maybe have time to look at them if they are few

Security Analyst, [10/25/2023 11:59 PM]

about the reked of the XPD:

- 1) you could remove both reentrancyGuard and I governor import (just for that no need to redeploy, you can leave them as they are)
- 2) for now both rebrandSymbol and rebrandName are not implemented in the governor contract (I know you will do in the future, but I just remember it)

developer,[10/26/2023 12:09 AM]

Yeah the tx.origin is fine because it scalled from that one so its okay.

I will leave both of these 2 distribution contracts as they are as long as there is no crucial mistake (or else would have to relaunch masterchef)

Leaving XPD as is, unless there is something crucial as well

Security Analyst, [10/26/2023 12:10 AM]

sure 👍

developer,[10/26/2023 12:10 AM]

This one should be done; <https://github.com/decentralizeX/PulseDAO/blob/main/contracts/pulse-ecosystem/hexSharesVault.sol>

it's basically very similar, but instead of depositing tokens, it checks how many t-shares the user has staked in another contract

Security Analyst, [10/26/2023 12:11 AM]

you mean you edited it?

developer,[10/26/2023 12:38 AM]

It's different. it's similar as token vault, but instead of depositing tokens it checks balance for T-shares from another contract.

It's slightly different(but similar)

I didn't include it as critical because there is no risk of loss of funds here(it only reads from other contract). But still best be checked.

When i tried on testnet it was reverting at the very beginning when trying to selfHarvest/withdraw, not sure why... But now works normally

developer,[10/26/2023 12:42 AM]

Because we only read from the other contract and don't hold any of users collateral.... One problem is user could "deposit"/start earning.. Then in the other contract withdraw, and deposit with new address and then "deposit" to our contract

How i solved it is i added that user must wait X amount of time before harvesting rewards

That contract from which we are checking balance also has penalties if user just withdraws, so i think it's not viable anyways

Security Analyst, [10/26/2023 1:00 AM]

I will check it when I will check all the ecosystem ones

developer,[10/26/2023 1:15 AM]

You already checked the tokenVault where you've made fixes for the debt system (and plsVault is just for native coin instead of ERC20)

Security Analyst, [10/26/2023 1:16 AM]

about token would I sent you the contract, and now when I recheck them I check also the other similar version if they are fine too

developer,[10/26/2023 1:16 AM]

ook

Security Analyst, [10/26/2023 3:50 PM]

masterchef don't have any problems, I will anyway the things that can be improved (no need to do those not that important), and some stuff to remember.

To remember:

1) You need to update both feeAddress and devAddress to the governor before transferring the ownership to it because after transferring the ownership those can't be updated if not proposing a new go or getting back the ownership with changeGovernorForSecurityPriorMintingBegins

not implemented in gov contract (I know they are for the future but take a look didn't forget to implement anything):

- 1) startPublishing
- 2) stopPublishing
- 3) setTrustedContract
- 4) tokenChangeOwnership
- 5) rewardSenators

Improvements:

- 1) remove reentrancyGuard import
- 2) put dtx as immutable
- 3) in TrustedContract event put contractAddress as indexed
- 4) in TransferCredit put from and to indexed
- 5) ++var in lines: 97, 123, 173
- 6) --var in lines: 173
- 7) in line 123, 97, uint256 instead of uint (style recommendation)
- 8) in setGovernorFee the amount ≥ 0 is useless, a uint will always be ≥ 0
- 9) in updateEmissionRate if maxSupply reached is true you can skip the branch since emission will be already to 0, moreover in the emit event in case the maxSupplyReached is true you still pass the new emission even if is skipped and kept to 0, you could just put maxSupplyReached as require and remove the unused if
- 10) in pendingDtx in the if you could also add allocation $\neq 0$

Security Analyst, [10/27/2023 11:16 PM]

so about the mining contracts there is kinda a bit problem, you adapted the autocake of masterchef so in the first check haven't check much about the implementation of the cake, but I give a better check this time. The problem is that in the autocake when you deposit tokens they are added to the total, while you burn them.

So in the cake you have the shares that represent exactly the deposited amount, but in your since the deposit amount is 0, you get a share of the current pending rewards. That leads to a lot of problematics, for example. If there are 50 tokens and 1 user with 50 shares in that instant the share of that user is valued 50, but if in the same instant another user deposit the same amount of the first user, the 50 rewards stay the same but the shares increase, and the user get the same amount of shares and so the first user see his shares go from a total value to 50 to a new value of 25 in zero time.

This create secondari problems in other function. For example I can deposit a big quantity and instantly convert to the creditContract for vote power, doing so I empty the credit of the contract and other users remain with no tokens and probably less than the amount they burned to get the mining started.

So there are other part broken because of this meccanics over the not stable rewards.

I came up with a easy fix and it is to emulate the original autecake implementation, keeping track of deposit and withdrawn virtually and create a virtual total balance that would be the sum of the credit of masterchef + the "deposited/withdrawn" tokens

To do so you have to add a variable: uint256 virtual supply, add a field named amount to the UserInfo (that will represent the deposited withdrawn tokens)

In deposit you will have to set the amount field to the amount deposited, and for calculating the shares you will have to keep the same calculation: $currentShares = \frac{amount \cdot totalShares}{pool}$; but convert the balanceOf to another function or change that directly, that add also the virtual supply to the total, so virtual supply + credits + pending. after getting the value from the balanceOf modified you will have to increase the virtual supply by the amount deposited.

In deposit the dtx at last user action have to be setted to 0 since you burn them and start gettign rewards from that point on.

For giftDeposit same as deposit.

In addAndExtendStake: same as deposit, in dtxAtLastUserAction you have to calculate same way with the modified balanceOf and after getting the value you remove the amount (the new one, so old + the deposited right now) from the total

In withdraw you calculate the amount with obviously the modified balanceOf and you remove the amount of token deposited (in the percentage of shares removed, so if you withdraw 50% of shares you remove 50% of the amount, also subtruct it from the total saved in the struct. To get that value jsut do $amount \cdot share / user.shares$) from that value, that will be the current amount, for the dtxAtLastUserAction same as the other once, In the end decrease the virtualy supply by the amount withdrawn

In hopStakeToAnotherPool you adjust it as a withdraw, and you will pass the amount withdrawn to the hopDeposit,

In hopeDeposit you will have to add a second amount, when you calculate the currentShares you will use the sum of those amount, after the balanceOf remember to increase the virtual supply by the second amount(that only represent the burned initial tokens), is important to divide the amount of credit and the virtual amount, because if you sum them and consider both as burned amount you will lose the credit gained so far.

in stakeRollover if you want give the bonus only as % of the amount gained so far and not taking in cosideration also the initial burning you will have to remove the amount from the currentAmount

In voting credit you will have to always remove the amount deposited from the currentAmount and also decrease the virtual supply
Also in migrateStake you will have to pass both the virtual deposited amount than the credit gained so far

Security Analyst, [10/27/2023 11:16 PM]

For the getPricePerFullShare or you leave it that includes the virtual supply otherwise will lose meaning because everyone will have a different value and a that point you can remove the function completely.

Hope to have explained it decently, anyway just look at the original contract and where you see safeTransfer you increase/decrease the virtual supply and remember to adjust the current amount everywhere you calculate it

Security Analyst, [10/27/2023 11:16 PM]

Immediately after you finish it let me know and I will check it asap

Security Analyst, [10/27/2023 11:30 PM]

For the other stuff:

Function not implemented in gov contract:

- 1) enableDisableStakeTransferFrom
- 2) setTrustedSender
- 3) setTrustedPool
- 4) setMigrationPool
- 5) modifyPartialWithdrawals
- 6) modifyPartialTransfers
- 7) enableDisableStakeTransfer
- 8) setMasterChefAddress
- 9) allowTxOrigin
- 10) setMinimumGiftDeposit

Problems:

- 1) in the votingCredit you don't check for partialWithdraw and also that the mandatoryTimeToServe have passed + no penalties for early withdraw

Improvements:

- 1) remove the IERC20 and Ownable import
- 2) put as constant withdrawFeePeriod and gracePeriod
- 3) in voteForProposal put the _wallet = outside the ternary operator (line 498, 554)
- 4) ++var in lines: 524, 699, 722, 726 (also --var), 738, 742 (also --var), 837

Events:

- 1) in RemoveVotes put proposalId as indexed, same for AddVotes
- 2) in TrustedSender and Trusted Pool put contractAddress as indexed
- 3) in StakeApproval and StakeAllowanceRevoke put owner and spender as indexed
- 4) in TransferStakeFrom put from and to indexed
- 5) in SetDelegate put userDelegating and delegatee as indexed

6) missing update variables events: enableDisableStakeTransferFrom, setCallFee, setMigrationPool, modifyPartialWithdrawals, modifyPartialTransfers, enableDisableStakeTransfer, setMasterChefAddress, allowTxOrigin, setMinimumGiftDeposit

developer,[10/27/2023 11:34 PM]

Hmh... but when i burn the tokens, the credit gets added(in masterchef) and is reflected in the function balanceOf()

That's why i thought there was no fundamental difference to cake.

Think so it's incorrect anyways?

Security Analyst, [10/27/2023 11:35 PM]

I might have confused with the burn of the tokens, let me check

Security Analyst, [10/27/2023 11:37 PM]

yea my bad, confused it with the xpd burn

Security Analyst, [10/27/2023 11:37 PM]

than ignore everything and only check the second part I sent you

Security Analyst, [10/27/2023 11:38 PM]

next time call that function convertToCredit

developer,[10/27/2023 11:38 PM]

the second part going from this
"For the other stuff:"

Security Analyst, [10/27/2023 11:38 PM]

yea

developer,[10/27/2023 11:38 PM]

k

Security Analyst, [10/27/2023 11:42 PM]

when you are always used to see the burn function as the standard one with no side effects you can easily do this error, this is why naming is important, and since you in a way convert the tokens to something else and in that function you actually burn the token in the meaning of burning, you should use a non misleading name as convert, transfer and so on. Anyway kinda my bad, but since the time constrain in a way don't check everytime what actually the function do and base what I remember of previous read and names. Anyway since you remember them well those problems are found easily so we can move faster to next stuff to check

Security Analyst, [10/27/2023 11:48 PM]

if the addCredit you designed to not do those check than no need to republish the contract just for the events and the improvements, is fine as it is

Security Analyst, [10/28/2023 12:17 AM]

about the other mining contracts, in the 6 month you can remove the public from constructor (is ignore since versions >= 0.7)

in 1 and 5 years the part of withdraFee the number are very slithly off, you can ignore them or just adjust by a little decreasing by little the max fee % or increasing the multiply for each day, like in the 5 years you can add 200 to the multiply by day making it more near the others

Security Analyst, [10/28/2023 1:17 AM]

also finished to recheck the consensus one and haven't found any error, I skip minor improvements because they will be almost identical to the one I told you at first check, and since no problem no need to redeploy

Security Analyst, [10/28/2023 7:57 PM]

about the senate contract:

Not implemented in gov contract:

- 1) addSenator
- 2) setSenatorCount

Problems:

- 1) in selfReplaceSenator in the first require you have to check if msg.sender is a senator not the one you will add after (that also guarantee that the sender is a senator so you could remove the next require that check if sender is a senator)
- 2) in vetoProposal, if it is a treasury proposal you have 2 errors, when you call the contract getting the treasuryProposal you pass the consensusproposalid instead that the treasuryproposalid, you also call the name of the variable as treasuryProposalId, when it actually is the consensusProposalId attached to the treasuryProposal object.
And in the next line you compare it to the treasuryProposalId and not the consensusProposalId as it should be done
- 3) in the multicall in the allCredits you only calculate the current one, since you also calculate the rewards you could add them to that amount since if grantVotingCredit would have been called that would have been the amount

Other:

- 1) haven't understood the comments at line 102/103, also why +1? if you just vote on the consensusProposal?

Security Analyst, [10/28/2023 10:24 PM]

in tokenVault there are no problems.

function not implemented in gov:

- setMasterChefAddress, setPoolPayout, updateSettings

Security Analyst, [10/28/2023 11:02 PM]

I checked the pls vault and is good, usually is not used anymore the .transfer because it forward a limited amount of gas, but it works fine

Security Analyst, [10/28/2023 11:05 PM]

do I have to check also the hexSharesVault? I haven't started it because it is quite different from the tokenVault, let me know if you want I also do that.

developer,[10/28/2023 11:13 PM]

Yeah

developer,[10/28/2023 11:14 PM]

i tried it on testnet and it was reverting when trying to harvest at beginning for some reason... But then it worked later so IDK

Security Analyst, [10/28/2023 11:14 PM]

I will do it probably tomorrow than

Security Analyst, [10/28/2023 11:14 PM]

I will keep that in mind

Security Analyst, [10/28/2023 11:16 PM]

in last days I work a lot extra time, and I should be able to finish before 31/10 end of the day, maybe if I manage to dedicate even extra time I could finish for the 30

developer,[10/28/2023 11:16 PM]

maybe we can extend for a few days over the end of month

Security Analyst, [10/28/2023 11:17 PM]

I have other work to do next month, so it's better also for me to finish before november

Security Analyst, [10/28/2023 11:37 PM]

about the treasury wallet no problems,

about the votingCredit no problems, but if you modify it I would:

- 1) add a deductCredit event and use it in deductCredit
- 2) in the addCredit emit the event of addcredit anyway
- 3) in deductCredit I would change a bit the code and first consume all the userCredit and after burn only the remaining one, so users don't end up having some credit around that are never spent because the amount is always greater (you could don't care and leave the job to manually add the difference between deducting to the user)

function not implemented in gov: modifyCreditingContract, modifyDeductingContract

Security Analyst, [10/29/2023 2:01 AM]

about the rewardsBoost, in case you edit it you can remove the tokensForBurn since you use that value inside the same function just use it in memory not as a state.

Moreover there is some stuff concerning me (could be as intended tho).

So, the masterchef starting emission rate is 850 token for each block, and in this contract each fibonaccening that value is reduced by 1.618 token, that means that you will trigger the expired fibonaccening after "only" 524 fibonaccening events (kinda too many to have, I don't think you do 1 each week, and if you really do 1 for each week it means that after 10 years you expire with the fibonaccening), anyway since the high amount I think you will reach max supply and put the emission to 0 before reaching the fib expiration.

Moreover the expired fibonaccening will be triggered when the emission rate will reach 2.168 token for each block, and after that will use the different model to calculate the new emission rate. In the new model supponing to have 0 after grand fibonaccening (kinda indifferent since in calculation very so little) the emission rate will be about 51.8 tokens each 10 billion of virtual supply, and since max supply will be of 27 billion I don't see it hard to reach at least 10 billion by the time there will be the new reward per block model. But in that case imo is a bit strange jump from a 3.786 token per block to a whopping 51.8 token per block

Security Analyst, [10/29/2023 2:02 AM]

moreover I would add some more precision to the factor value in calculateUpcomingRewardPerBlock, because with no some extra zeros the rounding inside the loop start to take a big weight. For example this are the first 30 value with and without rounding

Security Analyst, [10/29/2023 2:03 AM]

Security Analyst, [10/29/2023 2:03 AM]

and more you go on more the rounding chop off a bigger amount

Security Analyst, [10/29/2023 2:21 AM]

about the referralRewards I just wanted to ask if add vault is intended to be called by anyone and not using the "decentralized voting" require. The rest of the contract has no problems

developer,[10/29/2023 2:25 AM]

Yeah... if it exists in chef, but hasn't been added, anyone can do it

developer,[10/29/2023 2:26 AM]

haven't checked the rest yet, will do so tomorrow

Security Analyst, [10/29/2023 2:26 AM]

sure

Security Analyst, [10/29/2023 7:35 PM]

about the hexShares I found out that you haven't added every modification you did on the other one and some other stuff.

Problems:

- 1) in stakeHexShares you don't check if the nrOfStakes go over the maximum amount, but you do so only in the recalculate
- 2) in withdraw and harvest you have the require about the afePeriod that will be always true since you add the safePeriod to the current timestamp instead of the user.lastaction

Improvements:

- 1) as did in the other vaults you could publish the tokens only if they are >0 in withdraw and selfHarvest
- 2) as the vault you could add nonReentrant in the selfHarvest
- 3) as the vault you could add the require of being <= 10_000 in the updateSettings

Not implemented in gov:

- 1) setMaxStakes
- 2) setSafePeriod

Events:

- 1) when you update safePeriod and maxStakes you should fire a event

About the revert:

when you adjust the thing about safe period that one will make you revert, but in your case I think it was a problem or roundings when you save the debt since you divide by 1e12, so could happen that when you try to harvest there are some tokens ending in 178 in decimanls and you try to get out a

total of 180 (in the decimals of the token not token amount). But that would not be a problem as soon there is more than 1 people stacking since you will not try to claim all the rewards in the contract. I checked the math and seems good.

If you want me to check it in more detail I need you to tell me the exact sequence of how to make that revert happens (every function you called with what value and in what order) and I will have to deploy locally all the contracts needed and try to debug it. That I think will take me quite a lot of time

Security Analyst, [10/29/2023 9:19 PM]
about the governor contract:

- 1) you can't set a mapping value where you declare variables, so you have to move line 45-50 inside the constructor, also on those you can use acPool1-6 since they will already have the address when the constructor will be runned.
- 2) minDelay and maxDelay are not used in the fibonnaccening contract since in that contract you use hardcoded values, moreover there you use as maxDelay 90 days and not 31 as written in this contract

Security Analyst, [10/29/2023 11:37 PM]
about farms:

improvements:

- 1) to all votes add a require that don't let you vote after proposal "expiration", you did it in senate and other but not in farm and basicSettings
- 2) in updateFarm _pulseEcoCount and _poolLength are from old code and they are of no use now, you can remove them
- 3) at line 300 in the comment the tax is a % on top of the emission and is not included, so if you have a emission of 100, and a tax fee of 3, will be minted 3 as tax, but still emitted 100, so the total emission if you sum credit + minted is 103. Just inaccurate the comment

not implemented in gov yet:
1) changeMaxAllocations

Security Analyst, [10/30/2023 12:24 AM]
about the basicSettings:

don't have problems,

not implemented in gov: updateNewPoolProposalThreshold

as the farms you could add the require that stop votes after the delay + delayBeforeEnforce, if you do so you could also change this small stuff:

- 1) there are some == true at line: 128, 194, 266, 276, 399, 494, 512
- 2) you could add the event for the updateNewPoolProposalThreshold

PS: if you add something to masterchef with new pool that pull, since will not be in trusted contract, will not be able to call burnToken on masterchef. (need to change gov for that)

Security Analyst, [10/30/2023 12:25 AM]

With how much I progressed today I'm sure I will finish tomorrow. If i'm right I still have left to check the tokens distribution contracts, or I'm missing something? In case let me know if there is anything I haven't done and I will do it tomorrow after those 2

developer,[10/30/2023 1:42 AM]
Maybe in /NFT
the NFTmining.sol and allocationNFT.sol

NFTmining is similar tot oken vaults, but for NFT
the allocationNFT is rather simple i think

But these 2 contracts will not be used at beginning atleast perhaps never(because no NFTs)

Security Analyst, [10/30/2023 5:02 PM]
both distribution contract are good to go, and about the improvments are the same as I told you
time ago

developer,[10/30/2023 9:26 PM]
As far as fuction not implemented in governing contract;
If we are to change the rules, we need to change the governing contract and these functions are so
that we don't have to change everything else as well.... So in new governor we would just
setTrustedSender for example or whatever was needed

In votingCredit it's alright if user can redeem credits from a stake that has not yet been served

Since the contracts are already launched and there is nothing significant, i will leave it as is

Security Analyst, [10/30/2023 9:35 PM]
the not implemented in gov is just to let you know, because maybe reading them you remember you
forgot to implement something

developer,[10/30/2023 9:58 PM]
problems;
1.) fixed
2.) fixed as suggested
3.) it's only used on front-end, so it is calculated there and should be okay i think?

I don't remember about the comment, removed it.

And added the function to massVote

Security Analyst, [10/30/2023 9:59 PM]
if you add the extra amount there sure is fine

developer,[10/30/2023 10:18 PM]
1.) added
2.)added
3.) done like this, i think should be correct

developer,[10/30/2023 11:21 PM]
changed tokens for burn

I added global variable rewardReduceBy so we can change the number by how much it reduces..
and i changed it from 1618 to 2 * the amount... so the halve the inflation it would be 130 events,
every 11 days on average (in 4 years)

Little less frequent, albeit still.... And if the events are too often, it can be changed later on

i also added so once supply reaches > 15B tokens, the "expiredGrand" is set to true and we go to the
second formula

The second formula is supposed to set the inflation at 1.618% annually (and then after each event
reduce it by 1.618% from 1.618%)

Security Analyst, [10/30/2023 11:24 PM]
didn't understood point 2 and 3

Security Analyst, [10/30/2023 11:25 PM]
about last one since I don't know the block frequency I can't verify, but since I wrote you the value
that is emitted for 10B, you can just take that number that if I remember will should be around 50,
and multiply by the block in a year, and if you reach 1.6% of the 10B than is right

Security Analyst, [10/30/2023 11:25 PM]
the reduction is right

developer,[10/30/2023 11:52 PM]
Yeah i think the formula is correct (10.1S block time)

Security Analyst, [10/31/2023 12:08 AM]
yea is right

Security Analyst, [10/31/2023 12:50 AM]
So, I gave a fast look about the nft contracts, they were a lot messy and not in proper state of
evaluation, but I gave them a look anyway, but for sure not a definite look.

About the allocationExample:
even if I think is just for testing that loop for decrease the amount isn't the best decision, even more
because there are collection of 10k+ nfts.

- About the allocationNFT.sol
- 1) you can remove the context.sol
 - 2) since the non usage of Ownable functions you could remove the owable and put the owner as you
do in the other contracts
 - 3) you can remove the masterchef import, remove the variable from the contract and constructor,
never used.
 - 4) about the pendingContract timestamp, or you change name respect that is the timestamp + the 3
day delay (if you do so the time for approving is 3 days + 7 days), or you move the 3 days thing inside
the require before the re-proposal
 - 5) ProposalStructure is unused, so you can also remove the array.
 - 6) SetAllocationContract, SetPendingContract, UpdateVotes should have contractAddress indexed

- 7) notifyVote should have _contrat and enforcer indexed
- 8) why in getAllocation you don't move the allocationContract[_allocationContract inside a require?
- 9) I don't see the point of requiring that the alloc is >= of 1e18, ps you can also put that as require since in the NFTmining you revert in case the alloc is 0
- 10) I haven't understood much the notifyVote function, especially because it deduct a lot of tokens. Moreover it check only if the address is a contract, you should or check if the pending is valid and so call it after the propose of it, or check that the contract passed is a compatible contract in some way, or calling the nftAllocation with some true value or checking the compatibility with an interface.
- 11) in proposeAllocationContract you should in some way check if the contract passed is a valid contract as in the point before.
- 12) in proposeAllocationContract you should check the address before because I could populate that to block the proposal or the real contract or something like that, but implementing a check of the validity of the contract resolve any problems, and you could remove completely this check
- 13) since you use the consensus contract a lot you should put it as a variable with the syncfunction as you do with other contracts.
- 14) the treshold is the total balance of token but not weighted, meanwhile the weightedVote are weighted, this means that if everyone only stake in the 1month, and everyone vote for the proposal the threshold will not be met because you will reach 20% even if everyone voted for yes
- 15) rejectAllocationContract can be called even after the address is approved and setted, I would add in the proposal that the address is not already added, and in the approve just invalidate the pending
- 16) there are setAllocationContract, setApproveThreshold, setRejectThreshold, setRejectionPeriod that are not in gov and some of them should emit an updateEvent

Security Analyst, [10/31/2023 12:50 AM]
about the NFTmining

- 1) you can remove the following imports, Address, IERC721Receiver, IDTX, IVoting
- 2) userSettings is not used
- 3) there are some problems that you fixed in tokenVault but not here like in harverst that the 2 variable are inverted, in withdraw that you send the penalties only if they are >0
- 4) ++i at lines: 207, 239, 375
- 5) in selfHarvest and selfHarvestCustom I don't get why the totalWithdraw, just do as in the tokenVault and do += on the toWithdraw (have to replace totalwithdraw with this after in the code)
- 6) in selfHarvestCustom you are missing the line where you decrease the lastCredit, you can check selfHarvest
- 7) in rebalanceNFT if the new all is 0 you lose ALL the rewards, even when it was valid, and they go to treasury. I would give them to the user since they earned for some time before the allocation went to 0, but that is just my opinion
- 8) in the rebalance in the else if allocation is different, when you reset the user debt you just remove the profit, instead you should compute the new debt with the new allocation and remove the profit from that.
- 9) you can add nonReentrat to emergencyWithdraw
- 10) setPoolPayout and updateSettings, setMasterChefAddress not in gov
- 11) updateSettings, setMasterChefAddress don't have the event
- 12) withdrawStuckTokens in the other vaults don't have the decentralizedVoting modifier
- 13) in publicBalanceOf you should implement it as in tokenVault adding the credit to the pending amount

about the NFTmining I haven't check the events about indexing and other stuff, about those just take the tokenVault and update this contract to match the modification you did on the token vault, after done that the countract should be fine

Security Analyst, [10/31/2023 12:53 AM]

Let me know if you have any major necessity or problem with the contract I re-checked before the end of tomorrow, if everything is okay for you I will give you the info for the last payment tomorrow in the evening. Even after the payment if you have doubt about some modification you have to do to fix some code or some small problem you are free to ask, and will be out of charge

developer,[10/31/2023 1:05 AM]

- 1.) added that in stake as well
- 2.) fixed that

1improvements;

- 1.) added that
- 2.) won't add nonReentrant because we are calling "familiar" contracts only
- 3.)added require

Won't add event because in order to change we have to change the governing contract which can't happen "out of the blue" anyways

Think it's not necessary, seems to be working now. Will try it again afresh on testnet

developer,[10/31/2023 1:11 AM]

- 1.) moved into constructor
- 2.) Yeah that's okay

developer,[10/31/2023 1:29 AM]

- 1.) added require
- 2.) removed
- 3.) ok

developer,[10/31/2023 1:37 AM]

Added that require that was added to farms as well

- 1.) removed
- 2.) Not necessary

And yeah that's okay (that they can't pull)...

Security Analyst, [10/31/2023 1:38 AM]

Tomorrow I will give a fast check on the commits on GitHub that you are doing

developer,[10/31/2023 11:42 AM]

- 1.) removed
- 2.) replaced
- 3.) removed
- 4.) added = block.timestamp + 3 days;
- 5.) removed
- 6.) done
- 7.) done
- 8.) what if we just want to check the allocation and if it's 0 it's 0, no need to revert here imo
- 9.) removed $\geq 1e18$

10.) added check for compatibility, but else i think it's okay. In a sense that first you trigger NotifyVote, which notifies other users to start submitting votes (costs tokens so you don't spam random contracts)...and only once there are sufficient votes, can the proposeAllocationContract be triggered

11.) added contract check

12.) Checked before if it is a contract, so i guess it's okay now?

13.) This will rarely, if ever be used anyways so i will leave it as is

14.) I believe this is okay. The threshold should be set with that in mind (is currently 10% for approve and 5% for reject)

15.) I instead just added .isValid = false when enforcing contract, should be okay i think?

16.) Won't be emitting, as would need to update governor for it anyways

1.) removed

2.) userSettings? Can't find any such word in the file

3.) inverted last credit and current, add if penalties above 0

4.) done ++i

5.) (i removed selfHarvestCustom because it's useless)

Fixed and replaced with _toWithdraw

6.) removed selfHarvestCustom anyways

7.) It's a rare event that it would happen and in case it does it would be the responsibility of the user to harvest rewards prior and if someone else has to do it for them, they get penalized and lose the tokens

8.) recomputed the debt and detracted profit as recommended

9.) added

10.) ok

11.) not needed

12.) removed the modifier. Initially added it because i didn't want the NFT transferred out, but the interface is different so it wouldn't work anyways

13.) Already seems to be like that? Or did i change it before you took a look at it, don't know

Security Analyst, [10/31/2023 5:26 PM]

Seems good

Security Analyst, [10/31/2023 5:27 PM]

this

Security Analyst, [10/31/2023 5:30 PM]

yea I was seeig the older version, this why also the UserSettings was missing