

Audit Report for

PWN Finance

February 2024

Version 1.1

Email	Telephone
audits@extropy.io	+44 1865338228

Scope

The code is taken from repo https://github.com/PWNFinance/pwn_dao

The initial audit was at commit 1b457befd35e03c7342ed553eb201a5c412d7721

The final report was taken from branch 'extropy-initial-findings' at commit

838db27c0164ad62c697f55b0390aad996c56020

Contracts in scope

The following contracts were audited

PWN.sol

PWNEpochClock.sol

StakedPWN.sol

VoteEscrowedPWN.so

VoteEscrowedPWNBase.sol

VoteEscrowedPWNPower.sol

VoteEscrowedPWNStake.sol

VoteEscrowedPWNStakeMetadata.sol

VoteEscrowedPWNStorage.sol

BitMaskLib.sol

EpochPowerLib.sol

Error.sol

SlotComputingLib.sol


IStakedPWNSupplyManager.sol

IVotingContract.sol

Using This Report

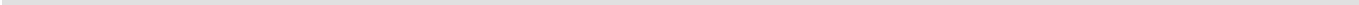
To facilitate the dissemination of the information within this report throughout your organisation, this document has been divided into the following clearly marked and separable sections.

- Executive Summary
- An overview of the assessment from a more technical perspective, including a defined scope and any caveats which may apply
- Technical Findings
- Detailed discussion (including evidence and recommendations) for each individual security issue which was identified

- 
- Audit process and tools used

Document Version Control

Data Classification	Client Confidential
Client Name	PWN Finance
Document Title	PWN DAO Audit
Author	Extropy Audit Team



Executive Summary

Extropy was contracted to conduct an initial code review and vulnerability assessment of the PWN DAO project between 18th January and 27th January 2024.

The final report check was carried out between 2nd and 5th February 2024.

Initial Report Findings

The code is well designed and written, with substantial test coverage.

We had an initial focus on the tokenomics involved but didn't find any issues.

The most important issue we found was a potential denial of service attack arising from repeated malicious addition of small amounts of stake.

The static analysis and formal verification tools results are given in Appendices C and D, of those results most were false positives, the others were low risk issues presented below.

The medium risk issue below involves a view function, whose loop could exceed a block gas limit if a large number of (presumably) small stakes were added.

Of the low risk issues L2 is raised as the nature of the interaction with Aragon DAO is yet to be decided.

Final Report Findings

An additional change was made to add an event when there has been a change to voting contracts reward ratio. This has been checked for correctness.

Most of the issues have been closed after discussion with the development team.

Issue M1 has been closed, as though it is possible, the cost to an attacker is would be too high.

The low risk issues were either closed or judged acceptable.

Issue Count

Stage	Critical	Medium	Low	Informational
Initial Report	0	1	4	1
Final Report	0	0	2	0

Test Coverage

File	% Lines	% Statements	% Branches	% Funcs
script/PWN.s.sol	0.00% (0/11)	0.00% (0/15)	100.00% (0/0)	0.00% (0/1)
src/PWN.sol	100.00% (46/46)	100.00% (55/55)	100.00% (24/24)	100.00% (5/5)
src/PWNEpochClock.sol	100.00% (4/4)	100.00% (7/7)	100.00% (2/2)	100.00% (1/1)
src/StakedPWN.sol	100.00% (55/55)	100.00% (72/72)	85.00% (17/20)	100.00% (10/10)
src/VoteEscrowedPWN.sol	100.00% (6/6)	100.00% (8/8)	100.00% (0/0)	100.00% (3/3)
src/lib/BitMaskLib.sol	0.00% (0/5)	0.00% (0/9)	0.00% (0/2)	0.00% (0/4)
src/lib/EpochPowerLib.sol	0.00% (0/8)	0.00% (0/13)	0.00% (0/2)	0.00% (0/3)
src/lib/SlotComputingLib.sol	0.00% (0/4)	0.00% (0/8)	100.00% (0/0)	0.00% (0/4)
src/vePWN/VoteEscrowedPWNBase.sol	94.12% (32/34)	96.67% (87/90)	100.00% (24/24)	100.00% (20/20)
src/vePWN/VoteEscrowedPWNPower.sol	98.33% (59/60)	98.90% (90/91)	93.75% (15/16)	100.00% (8/8)
src/vePWN/VoteEscrowedPWNStake.sol	100.00% (107/107)	100.00% (152/152)	97.92% (47/48)	100.00% (8/8)
src/vePWN/VoteEscrowedPWNStakeMetadata.sol	89.47% (51/57)	88.10% (74/84)	95.00% (19/20)	80.00% (8/10)
test/Base.t.sol	0.00% (0/2)	0.00% (0/3)	0.00% (0/2)	0.00% (0/1)
test/harness/StakedPWNHarness.sol	100.00% (1/1)	100.00% (1/1)	100.00% (0/0)	100.00% (1/1)
test/harness/VoteEscrowedPWNHarness.sol	100.00% (28/28)	100.00% (40/40)	50.00% (5/10)	100.00% (21/21)
test/integration/Integration.t.sol	0.00% (0/38)	0.00% (0/65)	0.00% (0/14)	0.00% (0/7)
test/unit/PWN.t.sol	0.00% (0/7)	0.00% (0/7)	100.00% (0/0)	0.00% (0/1)
test/unit/PWNEpochClock.t.sol	0.00% (0/3)	0.00% (0/3)	100.00% (0/0)	0.00% (0/1)
test/unit/StakedPWN.t.sol	0.00% (0/6)	0.00% (0/6)	100.00% (0/0)	0.00% (0/3)
test/unit/VoteEscrowedPWN.Power.t.sol	0.00% (0/21)	0.00% (0/28)	0.00% (0/4)	0.00% (0/4)
test/unit/VoteEscrowedPWN.StakeMetadata.t.sol	0.00% (0/5)	0.00% (0/5)	100.00% (0/0)	0.00% (0/1)
test/unit/VoteEscrowedPWNTTest.t.sol	0.00% (0/70)	0.00% (0/106)	0.00% (0/26)	0.00% (0/12)
Total	67.30% (389/578)	67.51% (586/868)	71.50% (153/214)	65.89% (85/129)

Test coverage was on the whole good, but could be improved for
vePWN/VoteEscrowedPWNStakeMetadata.sol.

Technical Findings

The remainder of this document is technical in nature and provides additional detail about the items already discussed, for the purposes of remediation and risk assessment.

Medium Risk

M1 : Staker power calculation Denial Of Service attack

Risk:	Medium
Affects	VoteEscrowedPWNPower.sol

Description

Malicious user might create an amount of stakes that is going to make stakerPowerAt impossible to execute.

stakerPowerAt is a view method and you don't pay gas when calling a view function. This doesn't change the fact it still has operations to do, which have costs, and thus are subject to the upper-bound gas limit as well as a time limit from blockchain nodes.

`stakerPowerAt` execution is essential for the Voting to work.

Recommendation

Calculate gas usage of the `stakerPowerAt` method. Based on the results limit of stakes per staker.

Outcome - *Closed*

Although an attacker could cause this loop to fail by adding many small stakes, the development team thought this unlikely due to the gas cost to the attacker.

We have therefore closed this issue.

Low Risk Issues

L1 : Using `ERC721::_mint()` can have unintended consequences

Risk:	Low
Affects	StakedPWN.sol

Description

If `PWN` is held and staked by a contract that can't handle `ERC721`, the user won't be able to transfer the `stPWN` token.

Recommendation

Use `ERC721::_safeMint()`

Outcome - *Accepted*

The dev team see this as acceptable, given the alternative could introduce reentrancy possibilities.

L2 : Token is not pausable.

Risk:	Low
Affects	PWN.sol

Description

If Aragon DAO gets hacked attacker would mint maximum amount of tokens. Would make sense to pause the token until situation is sorted.

Recommendation

Make `PWN` token pausable by the deployer. The deployer can upgrade the contract so there is trust in the deployer anyway.

Outcome - *Closed*

Having the token pausable would not solve the issue if the DAO was hacked, therefore this issue is closed.

L3 : Unable to withdraw ERC20s & ERC721s from the PWN token contracts.

Risk:	Low
Affects	PWN.sol
Affects	VoteEscrowedPWNBase.sol

Description

Users very often by a mistake or an accident send their token to tokens contract address. Malicious users send tokens to contact address to promote scam / phishing projects.

Recommendation

Add a method to withdraw any ERC20 / ERC721 tokens from PWN token contracts.

Outcome - *Accepted*

The users will mainly interact via the UI, therefore it is unlikely that tokens will be sent to the contract in error, the dev team sees this as an acceptable.

L4 : Return values from functions are not checked.

Risk:	Low
Affects	PWN.sol
Affects	VoteEscrowedPWNStake.sol

Description

For example PWN.sol line 187
proposalParameters is not checked but is then used on line 204

```
( // get proposal data
,, IVotingContract.ProposalParameters memory proposalParameters,
IVotingContract.Tally memory tally,,
) = IVotingContract(votingContract).getProposal(proposalId);

...

uint256 voterVotes = IVotingContract(votingContract).getVotingToken()
```

```
.getPastVotes(voter, proposalParameters.snapshotEpoch);  
uint256 totalVotes = tally.abstain + tally.yes + tally.no;
```

In VoteEscrowedPWNStake.sol

functions : createStake, increaseStake, withdrawStake

For example :

```
// transfer pwn token  
pwnToken.transferFrom(staker, address(this), amount);  
  
// emit event  
emit StakeCreated(stakeId, staker, amount, lockUpEpochs);
```

Recommendation

Check return values as handle appropriately.

Outcome - *Closed*

The return values are seen as acceptable values.

Informational Issues

I1 : Index events for easier searching.

Risk:	Informational
Affects	PWN.sol - event ProposalRewardAssigned
	VoteEscrowedPWNStake.sol - event StakeCreated
	VoteEscrowedPWNStake.sol - event StakeIncreased
	VoteEscrowedPWNStake.sol - event StakeWithdrawn

Description

Index event fields make the field more quickly accessible to off-chain tools that parse events. However, note that each index field costs extra gas during emission, so it's not necessarily best to index the maximum allowed per event (three fields). Each event should use three indexed fields if there are three or more fields, and gas usage is not particularly of concern for the events in question. If there are fewer than three fields, all of the fields should be indexed.

Recommendation

Add third indexed event to affected events.

Outcome - *Closed*

There would be no real advantage in indexing the other parameters here.

Appendix A

General Audit Goals

We audit the code in accordance with the following criteria:

Sound Architecture

This audit includes assessments of the overall architecture and design choices. Given the subjective nature of these assessments, it will be up to the development team to determine whether any changes should be made.

Smart Contract Best Practices

This audit will evaluate whether the codebase follows the current established best practices for smart contract development.

Code Correctness

This audit will evaluate whether the code does what it is intended to do.

Code Quality

This audit will evaluate whether the code has been written in a way that ensures readability and maintainability.

Security

This audit will look for any exploitable security vulnerabilities, or other potential threats to the users.

Although we have commented on the application design, issues of cryptoeconomics, game theory and suitability for business purposes as they relate to this project are beyond the scope of this audit.

Appendix B

Tools Used

Static Analysis

Aderyn : <https://github.com/Cyfrin/aderyn>

Slither : <https://github.com/crytic/slither>

Formal Verification

Certora : <https://certora.com>

Appendix C

Formal Verification with Certora

[PWN Certora JobID: 341444/e6527923f187435898d08969492e6ba8](#)

Results for envfreeFuncsStaticCheck:

Rule name	Verified	Time (sec)	Description	Local vars
name()	Not violated	0		no local variables
MINTABLE_TOTAL_SUPPLY()	Not violated	0		no local variables
balanceOf(address)	Not violated	0		no local variables
MAX_VOTING_REWARD()	Not violated	0		no local variables
allowance(address,address)	Not violated	0		no local variables
symbol()	Not violated	0		no local variables
owner()	Not violated	0		no local variables
totalSupply()	Not violated	0		no local variables
decimals()	Not violated	0		no local variables
IMMUTABLE_PERIOD()	Not violated	0		no local variables
pendingOwner()	Not violated	0		no local variables
mintedSupply()	Not violated	0		no local variables
VOTING_REWARD_DENOMINATOR()	Not violated	0		no local variables

Results for totalSupplyIsSumOfBalances_preserve:

Rule name	Verified	Time (sec)	Description	Local vars
ERC20.approve(address,uint256)	Not violated	24		no local variables
ERC20.transferFrom(address,address,uint256)	Not violated	1		no local variables
PWN.renounceOwnership()	Not violated	2		no local variables
PWN.claimProposalReward(address,uint256)	Not violated	11		no local variables
ERC20.decreaseAllowance(address,uint256)	Not violated	4		no local variables
PWN.acceptOwnership()	Not violated	2		no local variables
PWN.transferFrom(address,address,uint256)	Not violated	1		no local variables
)				
ERC20.increaseAllowance(address,uint256)	Not violated	4		no local variables
PWN.decreaseAllowance(address,uint256)	Not violated	1		no local variables
PWN.approve(address,uint256)	Not violated	1		no local variables
PWN.transfer(address,uint256)	Not violated	7		no local variables
ERC20.transfer(address,uint256)	Not violated	4		no local variables
PWN.mint(uint256)	Not violated	1		no local variables
PWN.assignProposalReward(address,uint256)	Not violated	3		no local variables
)				
PWN.transferOwnership(address)	Not violated	0		no local variables
PWN.increaseAllowance(address,uint256)	Not violated	2		no local variables

Rule name	Verified	Time (sec)	Description	Local vars
PWN.setVotingReward(address,uint256)	Not violated	6		no local variables
PWN.burn(uint256)	Not violated	1		no local variables

Results for totalSupplyIsSumOfBalances:

Rule name	Verified	Time (sec)	Description	Local vars
totalSupplyIsSumOfBalances_preserve	Not violated	39		no local variables
totalSupplyIsSumOfBalances_instate	Not violated	0		no local variables

Results for zeroAddressNoBalance_preserve:

Rule name	Verified	Time (sec)	Description	Local vars
PWN.acceptOwnership()	Not violated	8		no local variables
PWN.assignProposalReward(address,uint256)	Not violated	3		no local variables
PWN.setVotingReward(address,uint256)	Not violated	3		no local variables
ERC20.transferFrom(address,address,uint256)	Not violated	1		no local variables
PWN.burn(uint256)	Not violated	5		no local variables
PWN.decreaseAllowance(address,uint256)	Not violated	4		no local variables
PWN.increaseAllowance(address,uint256)	Not violated	2		no local variables
PWN.transferOwnership(address)	Not violated	8		no local variables

Rule name	Verified	Time (sec)	Description	Local vars
ERC20.approve(address,uint256)	Not violated	32		no local variables
ERC20.decreaseAllowance(address,uint256)	Not violated	1		no local variables
PWN.transfer(address,uint256)	Not violated	3		no local variables
PWN.mint(uint256)	Not violated	1		no local variables
ERC20.increaseAllowance(address,uint256)	Not violated	2		no local variables
PWN.approve(address,uint256)	Not violated	2		no local variables
PWN.claimProposalReward(address,uint256)	Not violated	6		no local variables
PWN.transferFrom(address,address,uint256)	Not violated	4		no local variables
)				
PWN.renounceOwnership()	Not violated	2		no local variables
ERC20.transfer(address,uint256)	Not violated	2		no local variables

Results for zeroAddressNoBalance:

Rule name	Verified	Time (sec)	Description	Local vars
zeroAddressNoBalance_preserve	Not violated	46		no local variables
zeroAddressNoBalance_instate	Not violated	0		no local variables

Results for mintedSupply_preserve:

Rule name	Verified	Time (sec)	Description	Local vars
ERC20.decreaseAllowance(address,uint256)	Not violated	0		no local variables

Rule name	Verified	Time (sec)	Description	Local vars
PWN.renounceOwnership()	Not violated	0		no local variables
PWN.approve(address,uint256)	Not violated	0		no local variables
PWN.increaseAllowance(address,uint256)	Not violated	0		no local variables
PWN.transferOwnership(address)	Not violated	0		no local variables
ERC20.transfer(address,uint256)	Not violated	0		no local variables
ERC20.increaseAllowance(address,uint256)	Not violated	0		no local variables
PWN.decreaseAllowance(address,uint256)	Not violated	0		no local variables
PWN.acceptOwnership()	Not violated	0		no local variables
PWN.mint(uint256)	Sanity check	2		no local variables
	failed			
ERC20.approve(address,uint256)	Not violated	0		no local variables
PWN.setVotingReward(address,uint256)	Not violated	0		no local variables
PWN.burn(uint256)	Not violated	0		no local variables
PWN.transfer(address,uint256)	Not violated	0		no local variables
ERC20.transferFrom(address,address,uint256)	Not violated	0		no local variables
PWN.claimProposalReward(address,uint256)	Not violated	0		no local variables
PWN.assignProposalReward(address,uint256)	Not violated	0		no local variables
)				

Rule name	Verified	Time (sec)	Description	Local vars
PWN.transferFrom(address,address,uint256	Not violated	0		no local variables
)				

Results for mintedSupply:

Rule name	Verified	Time (sec)	Description	Local vars
mintedSupply_preserve	Not violated	32		no local variables
mintedSupply_instate	Not violated	0		no local variables

Results for onlyAuthorizedCanTransfer:

Rule name	Verified	Time (sec)	Description	Local vars
PWN.votingRewards(address)	Sanity check	5		no local variables
	failed			
PWN.balanceOf(address)	Sanity check	3		no local variables
	failed			
PWN.name()	Sanity check	3		no local variables
	failed			
PWN.approve(address,uint256)	Sanity check	9		no local variables
	failed			
ERC20.transfer(address,uint256)	Sanity check	14		no local variables
	failed			
ERC20.approve(address,uint256)	Sanity check	108		no local variables
	failed			
PWN.assignProposalReward(address,uint256	Sanity check	2		no local variables
)	failed			

Rule name	Verified	Time (sec)	Description	Local vars
PWN.mintedSupply()	Sanity check	9		no local variables
	failed			
PWN.owner()	Sanity check	19		no local variables
	failed			
PWN.MINTABLE_TOTAL_SUPPLY()	Sanity check	3		no local variables
	failed			
PWN.increaseAllowance(address,uint256)	Sanity check	11		no local variables
	failed			
ERC20.decreaseAllowance(address,uint256)	Sanity check	9		no local variables
	failed			
PWN.pendingOwner()	Sanity check	23		no local variables
	failed			
PWN.renounceOwnership()	Sanity check	15		no local variables
	failed			
PWN.symbol()	Sanity check	4		no local variables
	failed			
PWN.MAX_VOTING_REWARD()	Sanity check	13		no local variables
	failed			
ERC20.name()	Sanity check	8		no local variables
	failed			
PWN.decreaseAllowance(address,uint256)	Sanity check	7		no local variables
	failed			

Rule name	Verified	Time (sec)	Description	Local vars
ERC20.increaseAllowance(address,uint256)	Sanity check	24		no local variables
	failed			
PWN.mint(uint256)	Sanity check	6		no local variables
	failed			
PWN.totalSupply()	Sanity check	3		no local variables
	failed			
PWN.claimProposalReward(address,uint256)	Sanity check	7		no local variables
	failed			
PWN.proposalRewards(address,uint256)	Sanity check	1		no local variables
	failed			
PWN.acceptOwnership()	Sanity check	2		no local variables
	failed			
PWN.VOTING_REWARD_DENOMINATOR()	Sanity check	1		no local variables
	failed			
PWN.setVotingReward(address,uint256)	Sanity check	1		no local variables
	failed			
ERC20.decimals()	Sanity check	5		no local variables
	failed			
PWN.transferFrom(address,address,uint256	Sanity check	2		no local variables
)	failed			
ERC20.transferFrom(address,address,uint2	Sanity check	2		no local variables
56)	failed			

Rule name	Verified	Time (sec)	Description	Local vars
PWN.allowance(address,address)	Sanity check	3		no local variables
	failed			
PWN.transfer(address,uint256)	Sanity check	7		no local variables
	failed			
PWN.transferOwnership(address)	Sanity check	3		no local variables
	failed			
PWN.IMMUTABLE_PERIOD()	Sanity check	12		no local variables
	failed			
PWN.decimals()	Sanity check	1		no local variables
	failed			
ERC20.balanceOf(address)	Sanity check	33		no local variables
	failed			
PWN.burn(uint256)	Sanity check	0		no local variables
	failed			
ERC20.symbol()	Sanity check	5		no local variables
	failed			
ERC20.allowance(address,address)	Sanity check	22		no local variables
	failed			
ERC20.totalSupply()	Sanity check	13		no local variables
	failed			

Results for noChangeToTotalSupply:

Rule name	Verified	Time (sec)	Description	Local va
PWN.IMMUTABLE_PERIOD()	Sanity check	7		no local v
	failed			
PWN.claimProposalReward(address,uint256)	Violated	47	Assert message:	ERC20=
				PWN=PV
				args=byt initializec
				e.block.c
				e.block.d
				e.block.g
				e.block.n
				e.block.ti
				e.msg.se (0x2716)
				e.msg.va
				e.tx.origi
				f.contrac
				f.isFallba to unknow
				f.isPayak unknown
				f.isPure= unknown
				f.isView= unknown
				f.number
				f.selector
				totalSupp
				totalSupp
PWN.totalSupply()	Sanity check	10		no local v
	failed			

Rule name	Verified	Time (sec)	Description	Local va
ERC20.symbol()	Sanity check	9		no local v
	failed			
PWN.transferFrom(address,address,uint256	Sanity check	11		no local v
)	failed			
PWN.votingRewards(address)	Sanity check	1		no local v
	failed			
PWN.setVotingReward(address,uint256)	Sanity check	9		no local v
	failed			
PWN.transferOwnership(address)	Sanity check	4		no local v
	failed			
ERC20.transfer(address,uint256)	Sanity check	3		no local v
	failed			
PWN.allowance(address,address)	Sanity check	5		no local v
	failed			
PWN.acceptOwnership()	Sanity check	1		no local v
	failed			
ERC20.approve(address,uint256)	Sanity check	126		no local v
	failed			
ERC20.decimals()	Sanity check	14		no local v
	failed			
PWN.balanceOf(address)	Sanity check	4		no local v
	failed			

Rule name	Verified	Time (sec)	Description	Local va
PWN.assignProposalReward(address,uint256	Sanity check	12		no local v
)	failed			
PWN.MINTABLE_TOTAL_SUPPLY()	Sanity check	8		no local v
	failed			
PWN.decimals()	Sanity check	2		no local v
	failed			
PWN.mintedSupply()	Sanity check	6		no local v
	failed			
ERC20.balanceOf(address)	Sanity check	11		no local v
	failed			
ERC20.totalSupply()	Sanity check	2		no local v
	failed			
PWN.transfer(address,uint256)	Sanity check	3		no local v
	failed			
PWN.renounceOwnership()	Sanity check	13		no local v
	failed			
PWN.mint(uint256)	Sanity check	48		no local v
	failed			
PWN.decreaseAllowance(address,uint256)	Sanity check	16		no local v
	failed			
PWN.MAX_VOTING_REWARD()	Sanity check	7		no local v
	failed			

Rule name	Verified	Time (sec)	Description	Local va
PWN.proposalRewards(address,uint256)	Sanity check	26		no local v
	failed			
PWN.pendingOwner()	Sanity check	14		no local v
	failed			
PWN.VOTING_REWARD_DENOMINATOR()	Sanity check	5		no local v
	failed			
PWN.approve(address,uint256)	Sanity check	2		no local v
	failed			
PWN.symbol()	Sanity check	2		no local v
	failed			
PWN.name()	Sanity check	4		no local v
	failed			
ERC20.name()	Sanity check	18		no local v
	failed			
ERC20.allowance(address,address)	Sanity check	2		no local v
	failed			
ERC20.decreaseAllowance(address,uint256)	Sanity check	1		no local v
	failed			
PWN.increaseAllowance(address,uint256)	Sanity check	2		no local v
	failed			
PWN.burn(uint256)	Sanity check	1		no local v
	failed			

Rule name	Verified	Time (sec)	Description	Local va
PWN.owner()	Sanity check	18		no local v
	failed			
ERC20.transferFrom(address,address,uint2	Sanity check	24		no local v
56)	failed			
ERC20.increaseAllowance(address,uint256)	Sanity check	2		no local v
	failed			

Results for all:

Rule name	Verified	Time (sec)	Description	Local vars
envfreeFuncsStaticCheck	Not violated	0		no local variables
zeroAddressNoBalance_instate	Not violated	0		no local variables
totalSupplyIsSumOfBalances_instate	Not violated	0		no local variables
mintedSupply_instate	Not violated	0		no local variables
burn	Violated	0	Assert message: "Expected burn to revert"	ERC20=ERC20
				(0xbfffffffffffffffffffff
				PWN=PWN
				(0x7fffffffffffffffffffff
				amount=10
				e.block.coinbase=
				e.block.difficulty=!
				e.block.gaslimit=8
				e.block.number=6
				e.msg.value=0

Rule name	Verified	Time (sec)	Description	Local vars from=0xffff
				fromBalanceBefore
				lastReverted=true
				other=0x4300
				otherBalanceBefore
				sumOfBalances=0
				totalSupplyBefore
mint	Violated	0	Assert message:	ERC20=ERC20
				(0xbfffffffffffffffffffff
				PWN=PWN
				(0x7fffffffffffffffffffff
				amount=10
				e.block.coinbase=
				e.block.difficulty=
				e.block.gaslimit=1
				e.block.number=1
				e.msg.value=0
				lastReverted=unir
				other=0x4300
				otherBalanceBefore
				sumOfBalances=0
				to=0xc100
				toBalanceBefore=
				totalSupplyBefore
acceptOwnership	Sanity check	2		no local variables
	failed			
approve	Sanity check	1		no local variables
	failed			
transferOwnership	Sanity check	2		no local variables
	failed			

Rule name	Verified	Time	Description	Local vars
transfer	Sanity check	5 (sec)		no local variables
	failed			
renounceOwnership	Sanity check	2		no local variables
	failed			
transferFrom	Sanity check	4		no local variables
	failed			
totalSupplyIsSumOfBalances	Not violated	58		no local variables
zeroAddressNoBalance	Not violated	65		no local variables
mintedSupply	Not violated	57		no local variables
onlyAuthorizedCanTransfer	Sanity check	134		no local variables
	failed			
noChangeToTotalSupply	Violated	145	Assert message:	no local variables

Failures summary:

Failed on burn:

Assert message: "Expected burn to revert"

Failed on mint:

Assert message:

Failed on acceptOwnership:

Failed on approve:

Failed on transferOwnership:

Failed on transfer:

Failed on renounceOwnership:

Failed on transferFrom:

Failed on onlyAuthorizedCanTransfer:

Failed on noChangeToTotalSupply:

Assert message:

Violated for:

PWN.claimProposalReward(address,uint256)

Duration 279694, RuleCheckResultsStats: numTotal = 13; numVerified = 3; numViolated = 10; numTimeout = 0; numError = 0

Done 4m

Done 4m

Event reporter: all events were sent without errors

Appendix D

Findings from Slither

INFO:Detectors:

PWN.assignProposalReward(address,uint256) (src/PWN.sol#139-169) ignores return value by (executed,proposalParameters) =

IVotingContract(votingContract).getProposal(proposalId) (src/PWN.sol#152-154)

PWN.c(address,uint256) (src/PWN.sol#176-212) ignores return value by (proposalParameters,tally) = IVotingContract(votingContract).getProposal(proposalId) (src/PWN.sol#186-188)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return>

INFO:Detectors:

PWNEpochClock.constructor(uint256) (src/PWNEpochClock.sol#22-28) uses timestamp for comparisons

Dangerous comparisons:

- initialEpochTimestamp > block.timestamp (src/PWNEpochClock.sol#24)

PWNEpochClock.currentEpoch() (src/PWNEpochClock.sol#37-45) uses timestamp for comparisons

Dangerous comparisons:

- block.timestamp < INITIAL_EPOCH_TIMESTAMP (src/PWNEpochClock.sol#39)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp>

INFO:Detectors:

Variable PWNEpochClock.INITIAL_EPOCH_TIMESTAMP (src/PWNEpochClock.sol#18) is not in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>

INFO:Detectors:

VoteEscrowedPWNStake.createStake(uint256,uint256)
(src/vePWN/VoteEscrowedPWNStake.sol#101-132) ignores return value by
pwnToken.transferFrom(staker,address(this),amount)
(src/vePWN/VoteEscrowedPWNStake.sol#128)

VoteEscrowedPWNStake.increaseStake(uint256,uint256,uint256)
(src/vePWN/VoteEscrowedPWNStake.sol#240-310) ignores return value by
pwnToken.transferFrom(staker,address(this),additionalAmount)
(src/vePWN/VoteEscrowedPWNStake.sol#303)

VoteEscrowedPWNStake.withdrawStake(uint256)
(src/vePWN/VoteEscrowedPWNStake.sol#315-336) ignores return value by
pwnToken.transfer(staker,staking.amount) (src/vePWN/VoteEscrowedPWNStake.sol#332)

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

INFO:Detectors:

StakedPWN._ownedTokensInEpochs (src/StakedPWN.sol#37) is never initialized. It is used in:

- StakedPWN.ownedTokensInEpochs(address) (src/StakedPWN.sol#109-111)
- StakedPWN.ownedTokenIdsAt(address,uint16) (src/StakedPWN.sol#116-142)
- StakedPWN._addIdToOwner(address,uint256,uint16) (src/StakedPWN.sol#182-200)
- StakedPWN._removeIdFromOwner(address,uint256,uint16) (src/StakedPWN.sol#202-214)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables>

INFO:Detectors:

Reentrancy in VoteEscrowedPWNStake.increaseStake(uint256,uint256,uint256)
(src/vePWN/VoteEscrowedPWNStake.sol#240-310):

External calls:

- _deleteStake(stakeId) (src/vePWN/VoteEscrowedPWNStake.sol#296)
- stakedPWN.burn(stakeId) (src/vePWN/VoteEscrowedPWNStake.sol#359)
- newStakeId = _createStake(staker,newInitialEpoch,newAmount,newLockUpEpochs)
(src/vePWN/VoteEscrowedPWNStake.sol#299)
- stakedPWN.mint(staker,newStakeId) (src/vePWN/VoteEscrowedPWNStake.sol#354)

State variables written after the call(s):

- newStakeId = _createStake(staker,newInitialEpoch,newAmount,newLockUpEpochs)
(src/vePWN/VoteEscrowedPWNStake.sol#299)
- stake.initialEpoch = initialEpoch (src/vePWN/VoteEscrowedPWNStake.sol#350)
- stake.amount = amount (src/vePWN/VoteEscrowedPWNStake.sol#351)
- stake.lockUpEpochs = lockUpEpochs (src/vePWN/VoteEscrowedPWNStake.sol#352)

VoteEscrowedPWNStorage.stakes (src/vePWN/VoteEscrowedPWNStorage.sol#57) can be used in cross function reentrancies:

- VoteEscrowedPWNStake._createStake(address,uint16,uint104,uint8)
(src/vePWN/VoteEscrowedPWNStake.sol#344-355)
- VoteEscrowedPWNStake.increaseStake(uint256,uint256,uint256)
(src/vePWN/VoteEscrowedPWNStake.sol#240-310)
- VoteEscrowedPWNStake.mergeStakes(uint256,uint256)
(src/vePWN/VoteEscrowedPWNStake.sol#186-229)
- VoteEscrowedPWNStake.splitStake(uint256,uint256)
(src/vePWN/VoteEscrowedPWNStake.sol#140-178)
- VoteEscrowedPWNStorage.stakes (src/vePWN/VoteEscrowedPWNStorage.sol#57)
- VoteEscrowedPWNStake.withdrawStake(uint256)
(src/vePWN/VoteEscrowedPWNStake.sol#315-336)

Reentrancy in VoteEscrowedPWNStake.mergeStakes(uint256,uint256)
(src/vePWN/VoteEscrowedPWNStake.sol#186-229):

External calls:

- _deleteStake(stakeId1) (src/vePWN/VoteEscrowedPWNStake.sol#220)
- stakedPWN.burn(stakeId) (src/vePWN/VoteEscrowedPWNStake.sol#359)
- _deleteStake(stakeId2) (src/vePWN/VoteEscrowedPWNStake.sol#221)
- stakedPWN.burn(stakeId) (src/vePWN/VoteEscrowedPWNStake.sol#359)
- newStakeId = _createStake(staker,newInitialEpoch,newAmount,newLockUpEpochs)
(src/vePWN/VoteEscrowedPWNStake.sol#225)
- stakedPWN.mint(staker,newStakeId) (src/vePWN/VoteEscrowedPWNStake.sol#354)

State variables written after the call(s):

- newStakeId = _createStake(staker,newInitialEpoch,newAmount,newLockUpEpochs)
(src/vePWN/VoteEscrowedPWNStake.sol#225)
- stake.initialEpoch = initialEpoch (src/vePWN/VoteEscrowedPWNStake.sol#350)

- stake.amount = amount (src/vePWN/VoteEscrowedPWNStake.sol#351)
- stake.lockUpEpochs = lockUpEpochs (src/vePWN/VoteEscrowedPWNStake.sol#352)

VoteEscrowedPWNStorage.stakes (src/vePWN/VoteEscrowedPWNStorage.sol#57) can be used in cross function reentrancies:

- VoteEscrowedPWNStake._createStake(address,uint16,uint104,uint8) (src/vePWN/VoteEscrowedPWNStake.sol#344-355)
- VoteEscrowedPWNStake.increaseStake(uint256,uint256,uint256) (src/vePWN/VoteEscrowedPWNStake.sol#240-310)
- VoteEscrowedPWNStake.mergeStakes(uint256,uint256) (src/vePWN/VoteEscrowedPWNStake.sol#186-229)
- VoteEscrowedPWNStake.splitStake(uint256,uint256) (src/vePWN/VoteEscrowedPWNStake.sol#140-178)
- VoteEscrowedPWNStorage.stakes (src/vePWN/VoteEscrowedPWNStorage.sol#57)
- VoteEscrowedPWNStake.withdrawStake(uint256) (src/vePWN/VoteEscrowedPWNStake.sol#315-336)

Reentrancy in VoteEscrowedPWNStake.splitStake(uint256,uint256) (src/vePWN/VoteEscrowedPWNStake.sol#140-178):

External calls:

- _deleteStake(stakeId) (src/vePWN/VoteEscrowedPWNStake.sol#168)
- stakedPWN.burn(stakeId) (src/vePWN/VoteEscrowedPWNStake.sol#359)
- newStakeId1 = _createStake(staker,originalInitialEpoch,originalAmount - uint104(splitAmount),originalLockUpEpochs) (src/vePWN/VoteEscrowedPWNStake.sol#171-173)
- stakedPWN.mint(staker,newStakeId) (src/vePWN/VoteEscrowedPWNStake.sol#354)

State variables written after the call(s):

- newStakeId1 = _createStake(staker,originalInitialEpoch,originalAmount - uint104(splitAmount),originalLockUpEpochs) (src/vePWN/VoteEscrowedPWNStake.sol#171-173)
- stake.initialEpoch = initialEpoch (src/vePWN/VoteEscrowedPWNStake.sol#350)
- stake.amount = amount (src/vePWN/VoteEscrowedPWNStake.sol#351)
- stake.lockUpEpochs = lockUpEpochs (src/vePWN/VoteEscrowedPWNStake.sol#352)

VoteEscrowedPWNStorage.stakes (src/vePWN/VoteEscrowedPWNStorage.sol#57) can be used in cross function reentrancies:

- `VoteEscrowedPWNStake._createStake(address,uint16,uint104,uint8)`
(src/vePWN/VoteEscrowedPWNStake.sol#344-355)
- `VoteEscrowedPWNStake.increaseStake(uint256,uint256,uint256)`
(src/vePWN/VoteEscrowedPWNStake.sol#240-310)
- `VoteEscrowedPWNStake.mergeStakes(uint256,uint256)`
(src/vePWN/VoteEscrowedPWNStake.sol#186-229)
- `VoteEscrowedPWNStake.splitStake(uint256,uint256)`
(src/vePWN/VoteEscrowedPWNStake.sol#140-178)
- `VoteEscrowedPWNStorage.stakes` (src/vePWN/VoteEscrowedPWNStorage.sol#57)
- `VoteEscrowedPWNStake.withdrawStake(uint256)`
(src/vePWN/VoteEscrowedPWNStake.sol#315-336)

Reentrancy in `VoteEscrowedPWNStake.splitStake(uint256,uint256)`
(src/vePWN/VoteEscrowedPWNStake.sol#140-178):

External calls:

- `_deleteStake(stakeId)` (src/vePWN/VoteEscrowedPWNStake.sol#168)
- `stakedPWN.burn(stakeId)` (src/vePWN/VoteEscrowedPWNStake.sol#359)
- `newStakeId1 = _createStake(staker,originalInitialEpoch,originalAmount - uint104(splitAmount),originalLockUpEpochs)`
(src/vePWN/VoteEscrowedPWNStake.sol#171-173)
- `stakedPWN.mint(staker,newStakeId)` (src/vePWN/VoteEscrowedPWNStake.sol#354)
- `newStakeId2 = _createStake(staker,originalInitialEpoch,uint104(splitAmount),originalLockUpEpochs)`
(src/vePWN/VoteEscrowedPWNStake.sol#174)
- `stakedPWN.mint(staker,newStakeId)` (src/vePWN/VoteEscrowedPWNStake.sol#354)

State variables written after the call(s):

- `newStakeId2 = _createStake(staker,originalInitialEpoch,uint104(splitAmount),originalLockUpEpochs)`
(src/vePWN/VoteEscrowedPWNStake.sol#174)
- `newStakeId = ++ lastStakeId` (src/vePWN/VoteEscrowedPWNStake.sol#348)

`VoteEscrowedPWNStorage.lastStakeId` (src/vePWN/VoteEscrowedPWNStorage.sol#39)
can be used in cross function reentrancies:

- `VoteEscrowedPWNStake._createStake(address,uint16,uint104,uint8)`
(src/vePWN/VoteEscrowedPWNStake.sol#344-355)

- `VoteEscrowedPWNStorage.lastStakeId`
(src/vePWN/VoteEscrowedPWNStorage.sol#39)
- `newStakeId2 =`
`_createStake(staker,originalInitialEpoch,uint104(splitAmount),originalLockUpEpochs)`
(src/vePWN/VoteEscrowedPWNStake.sol#174)
- `stake.initialEpoch = initialEpoch` (src/vePWN/VoteEscrowedPWNStake.sol#350)
- `stake.amount = amount` (src/vePWN/VoteEscrowedPWNStake.sol#351)
- `stake.lockUpEpochs = lockUpEpochs` (src/vePWN/VoteEscrowedPWNStake.sol#352)

`VoteEscrowedPWNStorage.stakes` (src/vePWN/VoteEscrowedPWNStorage.sol#57) can be used in cross function reentrancies:

- `VoteEscrowedPWNStake._createStake(address,uint16,uint104,uint8)`
(src/vePWN/VoteEscrowedPWNStake.sol#344-355)
- `VoteEscrowedPWNStake.increaseStake(uint256,uint256,uint256)`
(src/vePWN/VoteEscrowedPWNStake.sol#240-310)
- `VoteEscrowedPWNStake.mergeStakes(uint256,uint256)`
(src/vePWN/VoteEscrowedPWNStake.sol#186-229)
- `VoteEscrowedPWNStake.splitStake(uint256,uint256)`
(src/vePWN/VoteEscrowedPWNStake.sol#140-178)
- `VoteEscrowedPWNStorage.stakes` (src/vePWN/VoteEscrowedPWNStorage.sol#57)
- `VoteEscrowedPWNStake.withdrawStake(uint256)`
(src/vePWN/VoteEscrowedPWNStake.sol#315-336)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1>

INFO:Detectors:

`VoteEscrowedPWNStakeMetadata._computeAttributes(uint256).currentPowerChangeIndex` (src/vePWN/VoteEscrowedPWNStakeMetadata.sol#126) is a local variable never initialized

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables>

INFO:Detectors:

`PWN.assignProposalReward(address,uint256)` (src/PWN.sol#139-169) ignores return value by `(executed,proposalParameters) =`

`IVotingContract(votingContract).getProposal(proposalId)` (src/PWN.sol#152-154)

PWN.claimProposalReward(address,uint256) (src/PWN.sol#176-212) ignores return value by (proposalParameters,tally) = IVotingContract(votingContract).getProposal(proposalId) (src/PWN.sol#186-188)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return>

INFO:Detectors:

VoteEscrowedPWNPower.stakerPowerAt(address,uint256)
(src/vePWN/VoteEscrowedPWNPower.sol#91-107) has external calls inside a loop:
stakeIds = stakedPWN.ownedTokenIdsAt(staker,_epoch)
(src/vePWN/VoteEscrowedPWNPower.sol#93)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop>

INFO:Detectors:

Reentrancy in VoteEscrowedPWNStake.createStake(uint256,uint256)
(src/vePWN/VoteEscrowedPWNStake.sol#101-132):

External calls:

- stakeId = _createStake(staker,initialEpoch,uint104(amount),uint8(lockUpEpochs))
(src/vePWN/VoteEscrowedPWNStake.sol#125)
- stakedPWN.mint(staker,newStakeId) (src/vePWN/VoteEscrowedPWNStake.sol#354)
- pwnToken.transferFrom(staker,address(this),amount)
(src/vePWN/VoteEscrowedPWNStake.sol#128)

Event emitted after the call(s):

- StakeCreated(stakeId,staker,amount,lockUpEpochs)
(src/vePWN/VoteEscrowedPWNStake.sol#131)

Reentrancy in VoteEscrowedPWNStake.increaseStake(uint256,uint256,uint256)
(src/vePWN/VoteEscrowedPWNStake.sol#240-310):

External calls:

- _deleteStake(stakeId) (src/vePWN/VoteEscrowedPWNStake.sol#296)
- stakedPWN.burn(stakeId) (src/vePWN/VoteEscrowedPWNStake.sol#359)
- newStakeId = _createStake(staker,newInitialEpoch,newAmount,newLockUpEpochs)
(src/vePWN/VoteEscrowedPWNStake.sol#299)
- stakedPWN.mint(staker,newStakeId) (src/vePWN/VoteEscrowedPWNStake.sol#354)

- `pwnToken.transferFrom(staker,address(this),additionalAmount)`
(src/vePWN/VoteEscrowedPWNStake.sol#303)

Event emitted after the call(s):

- `StakeIncreased(stakeId,staker,additionalAmount,newAmount,additionalEpochs,newLockUpEpochs,newStakeId)` (src/vePWN/VoteEscrowedPWNStake.sol#307-309)

Reentrancy in `VoteEscrowedPWNStake.mergeStakes(uint256,uint256)`
(src/vePWN/VoteEscrowedPWNStake.sol#186-229):

External calls:

- `_deleteStake(stakeId1)` (src/vePWN/VoteEscrowedPWNStake.sol#220)
- `stakedPWN.burn(stakeId)` (src/vePWN/VoteEscrowedPWNStake.sol#359)
- `_deleteStake(stakeId2)` (src/vePWN/VoteEscrowedPWNStake.sol#221)
- `stakedPWN.burn(stakeId)` (src/vePWN/VoteEscrowedPWNStake.sol#359)
- `newStakeId = _createStake(staker,newInitialEpoch,newAmount,newLockUpEpochs)`
(src/vePWN/VoteEscrowedPWNStake.sol#225)
- `stakedPWN.mint(staker,newStakeId)` (src/vePWN/VoteEscrowedPWNStake.sol#354)

Event emitted after the call(s):

- `StakeMerged(stakeId1,stakeId2,staker,newAmount,newLockUpEpochs,newStakeId)`
(src/vePWN/VoteEscrowedPWNStake.sol#228)

Reentrancy in `VoteEscrowedPWNStake.splitStake(uint256,uint256)`
(src/vePWN/VoteEscrowedPWNStake.sol#140-178):

External calls:

- `_deleteStake(stakeId)` (src/vePWN/VoteEscrowedPWNStake.sol#168)
- `stakedPWN.burn(stakeId)` (src/vePWN/VoteEscrowedPWNStake.sol#359)
- `newStakeId1 = _createStake(staker,originalInitialEpoch,originalAmount - uint104(splitAmount),originalLockUpEpochs)`
(src/vePWN/VoteEscrowedPWNStake.sol#171-173)
- `stakedPWN.mint(staker,newStakeId)` (src/vePWN/VoteEscrowedPWNStake.sol#354)
- `newStakeId2 = _createStake(staker,originalInitialEpoch,uint104(splitAmount),originalLockUpEpochs)`
(src/vePWN/VoteEscrowedPWNStake.sol#174)
- `stakedPWN.mint(staker,newStakeId)` (src/vePWN/VoteEscrowedPWNStake.sol#354)

Event emitted after the call(s):

- StakeSplit(stakeId,staker,originalAmount - uint104(splitAmount),splitAmount,newStakeId1,newStakeId2) (src/vePWN/VoteEscrowedPWNStake.sol#177)

Reentrancy in VoteEscrowedPWNStake.withdrawStake(uint256) (src/vePWN/VoteEscrowedPWNStake.sol#315-336):

External calls:

- _deleteStake(stakeId) (src/vePWN/VoteEscrowedPWNStake.sol#329)
- stakedPWN.burn(stakeId) (src/vePWN/VoteEscrowedPWNStake.sol#359)
- pwnToken.transfer(staker,stake.amount) (src/vePWN/VoteEscrowedPWNStake.sol#332)

Event emitted after the call(s):

- StakeWithdrawn(stakeId,staker,stake.amount) (src/vePWN/VoteEscrowedPWNStake.sol#335)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3>

INFO:Detectors:

PWNEpochClock.constructor(uint256) (src/PWNEpochClock.sol#22-28) uses timestamp for comparisons

Dangerous comparisons:

- initialEpochTimestamp > block.timestamp (src/PWNEpochClock.sol#24)

PWNEpochClock.currentEpoch() (src/PWNEpochClock.sol#37-45) uses timestamp for comparisons

Dangerous comparisons:

- block.timestamp < INITIAL_EPOCH_TIMESTAMP (src/PWNEpochClock.sol#39)

VoteEscrowedPWNStakeMetadata._computeAttributes(uint256) (src/vePWN/VoteEscrowedPWNStakeMetadata.sol#106-156) uses timestamp for comparisons

Dangerous comparisons:

- `attributes.powerChanges[i].timestamp <= block.timestamp`
(src/vePWN/VoteEscrowedPWNStakeMetadata.sol#135)
- `block.timestamp < attributes.powerChanges[0].timestamp`
(src/vePWN/VoteEscrowedPWNStakeMetadata.sol#149)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp>

INFO:Detectors:

`VoteEscrowedPWNStake.increaseStake(uint256,uint256,uint256)`
(src/vePWN/VoteEscrowedPWNStake.sol#240-310) has a high cyclomatic complexity (12).

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity>

INFO:Detectors:

Variable `PWNEpochClock.INITIAL_EPOCH_TIMESTAMP` (src/PWNEpochClock.sol#18) is not in mixedCase

Parameter `VoteEscrowedPWN.initialize(address,address,address,address)._pwnToken`
(src/VoteEscrowedPWN.sol#33) is not in mixedCase

Parameter `VoteEscrowedPWN.initialize(address,address,address,address)._stakedPWN`
(src/VoteEscrowedPWN.sol#34) is not in mixedCase

Parameter `VoteEscrowedPWN.initialize(address,address,address,address)._epochClock`
(src/VoteEscrowedPWN.sol#35) is not in mixedCase

Parameter `VoteEscrowedPWN.initialize(address,address,address,address)._owner`
(src/VoteEscrowedPWN.sol#36) is not in mixedCase

Function `VoteEscrowedPWNBase.CLOCK_MODE()`
(src/vePWN/VoteEscrowedPWNBase.sol#118-120) is not in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>

INFO:Detectors:

Variable `VoteEscrowedPWNStake.mergeStakes(uint256,uint256).finalEpoch1`
(src/vePWN/VoteEscrowedPWNStake.sol#190) is too similar to
`VoteEscrowedPWNStake.mergeStakes(uint256,uint256).finalEpoch2`
(src/vePWN/VoteEscrowedPWNStake.sol#191)

Variable `VoteEscrowedPWNStake.splitStake(uint256,uint256).newStakeId1` (src/vePWN/VoteEscrowedPWNStake.sol#142) is too similar to `VoteEscrowedPWNStake.splitStake(uint256,uint256).newStakeId2` (src/vePWN/VoteEscrowedPWNStake.sol#142)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar>

INFO:Detectors:

`StakedPWN._ownedTokensInEpochs` (src/StakedPWN.sol#37) is never initialized. It is used in:

- `StakedPWN.ownedTokensInEpochs(address)` (src/StakedPWN.sol#109-111)
- `StakedPWN.ownedTokenIdsAt(address,uint16)` (src/StakedPWN.sol#116-142)
- `StakedPWN._addIdToOwner(address,uint256,uint16)` (src/StakedPWN.sol#182-200)
- `StakedPWN._removeIdFromOwner(address,uint256,uint16)` (src/StakedPWN.sol#202-214)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables>

INFO:Detectors:

`PWNEpochClock.constructor(uint256)` (src/PWNEpochClock.sol#22-28) uses timestamp for comparisons

Dangerous comparisons:

- `initialEpochTimestamp > block.timestamp` (src/PWNEpochClock.sol#24)

`PWNEpochClock.currentEpoch()` (src/PWNEpochClock.sol#37-45) uses timestamp for comparisons

Dangerous comparisons:

- `block.timestamp < INITIAL_EPOCH_TIMESTAMP` (src/PWNEpochClock.sol#39)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp>

INFO:Detectors:

Variable `PWNEpochClock.INITIAL_EPOCH_TIMESTAMP` (src/PWNEpochClock.sol#18) is not in `mixedCase`

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>

INFO:Slither:src analyzed (64 contracts with 93 detectors), 38 result(s) found