Security Review Report
NM-0365-PWN-ONLY-TOKENS

NETHERMIND
SECURITY

(Nov 13, 2024)

Contents

1 Executive Summary

This document presents the security review performed by Nethermind Security for PWN smart contracts. The PWN team is implementing an on-chain governance system where holders of `$PWN` tokens can stake their tokens for a predefined time. The owners of the staked tokens are granted voting power and revenue share.

This security review focuses exclusively on the smart contracts listed in Section 2 (*Audited Files*).

The audit was performed using (a) manual analysis of the codebase and (b) creation of test cases. **Along this document, we report** three points of attention, classified as `Informational`. The issues are summarized in Fig. 1.

This document is organized as follows. Section 2 presents the files in the scope. Section 3 summarizes the issues. Section 4 presents the system overview. Section 5 discusses the risk rating methodology. Section 6 details the issues. Section 7 discusses the documentation provided by the client for this audit. Section 8 presents the compilation, tests, and automated tests. Section 9 concludes the document.

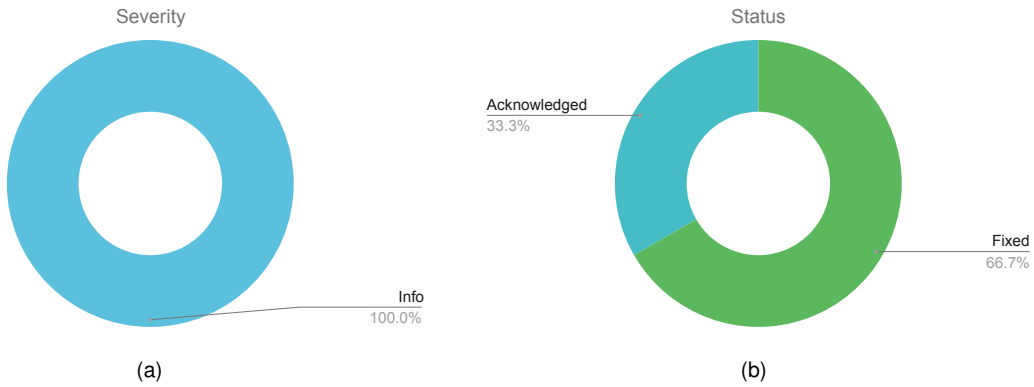

Fig. 1: Distribution of issues: Critical (0), **High** (0), **Medium** (0), **Low** (0), **Undetermined** (0), **Informational** (3), **Best Practices** (0). **Distribution of status: Fixed** (2), **Acknowledged** (1), **Mitigated** (0), **Unresolved** (0)

Summary of the Audit

Audit Type	Security Review
Initial Report	Nov 13, 2024
Response from Client	Regular responses during audit engagement
Final Report	Nov 13, 2024
Repository	pwn_dao
Commit (Audit)	51bf87dc5d5ab54c781e6222485d71c315b6c7e7
Commit (Final)	0e77ede940980c985d97d683081ad66d72e04ab3
Documentation Assessment	High
Test Suite Assessment	High

2 Audited Files

	Contract	LoC	Comments	Ratio	Blank	Total
1	PWNEpochClock.sol	20	19	95%	8	47
2	token/PWN.sol	132	102	77%	52	285
3	token/StakedPWN.sol	48	54	112%	28	130
4	token/VoteEscrowedPWN.sol	40	16	40%	9	65
5	token/vePWN/VoteEscrowedPWNBase.sol	93	46	49%	35	174
6	token/vePWN/VoteEscrowedPWNStake.sol	357	159	44%	77	591
7	token/vePWN/VoteEscrowedPWNPower.sol	147	48	32%	35	223
8	token/vePWN/VoteEscrowedPWNStorage.sol	25	39	156%	17	81
9	token/vePWN/VoteEscrowedPWNStakeMetadata.sol	152	16	10%	26	191
10	lib/BitMaskLib.sol	22	24	109%	9	55
11	lib/SlotComputingLib.sol	15	31	206%	8	54
12	lib/EpochPowerLib.sol	25	18	72%	7	50
13	lib/Error.sol	36	8	22%	9	53
	Total	**1112**	**580**	**52.2%**	**320**	**1999**

3 Summary of Issues

	Finding	Severity	Update
1	Invariant break when increasing stake	Info	Fixed
2	Absence of a mechanism to disable transfers	Info	Acknowledged
3	Inconsistency in conditions of `mergeStakes(...)` function	Info	Fixed

4 System Overview

The PWN ecosystem utilizes three different tokens as shown in the figure below: PWN, StakedPWN, and VoteEscrowedPWN. This section provides an overview of each token contract and its core functionalities.

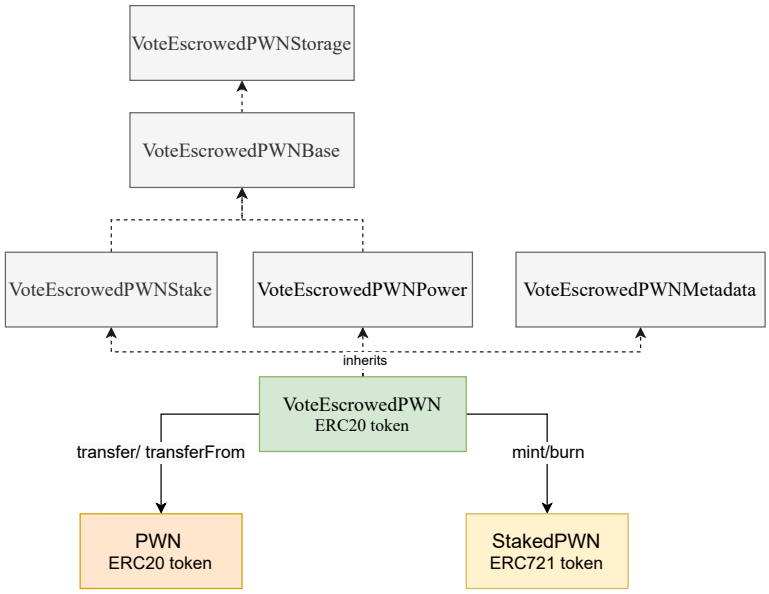

Fig. 2: PWN tokens overview

4.1 PWN token

The PWN token is the main governance token of the PWN DAO and is used as a reward for participants who vote on proposals. The contract owner is the PWN DAO, which holds control over certain functionalities of the token. Initially, the token is non-transferable. The owner can later enable transfers or configure the contract to allow transfers only by specific actors.

The token is designed to have a maximum supply of 100 million tokens, and only the contract owner is authorized to mint new tokens through the `mint(...)` function.

```
function mint(uint256 amount) external onlyOwner
```

In addition to minting, token holders have the ability to burn their tokens. This is done through the `burn(...)` function.

```
function burn(uint256 amount) external
```

4.1.1 Voting Rewards

The contract owner can set a reward percentage for each voting contract via the `setVotingReward(...)` function. This reward is capped at 1% of the total supply of PWN tokens.

```
function setVotingReward(address votingContract, uint256 votingReward) external onlyOwner
```

Once the reward percentage is set, the voting contract can invoke the `assignProposalReward(...)` function to calculate and assign rewards for individual proposals. This function computes the reward amount based on the assigned percentage of the current total supply.

```
function assignProposalReward(uint256 proposalId) external
```

Once a proposal is executed, voters who participated in the vote can claim their rewards through the `claimProposalReward(...)` and `claimProposalRewardBatch(...)` functions to claim individual or batched proposals, respectively.

```
function claimProposalReward(address votingContract, uint256 proposalId) public

function claimProposalRewardBatch(address votingContract, uint256[] calldata proposalIds) external
```

4.2 StakedPWN token

The `StakedPWN` token is an `ERC721` token that represents a stake in the PWN DAO. By default, this token is non-transferable. However, the contract owner can enable transfers using the `enableTransfers()` function:

```
function enableTransfers() external onlyOwner
```

Alternatively, specific addresses can be allowed to transfer `StakedPWN` tokens through the `setTransferAllowlist(...)` function:

```
function setTransferAllowlist(address addr, bool isAllowed) external onlyOwner
```

Additionally, the minting and burning of `StakedPWN` tokens is restricted to the `SupplyManager` address, which points to the `VoteEscrowedPWN` contract.

4.3 VoteEscrowedPWN token

The `VoteEscrowedPWN` token contract handles the main logic for staking `PWN` tokens and voting powers calculation for stakers.

4.3.1 Stake creation

Users can stake their `PWN` tokens through the `createStake(...)` or `createStakeOnBehalfOf(...)` function, the latter allowing them to stake on behalf of another staker. These functions require users to specify the amount of `PWN` tokens to stake and the lockup period (measured in epochs) for the stake. The lockup period must be between 1 and 5 years or exactly 10 years. The tokens are transferred from the user to the contract, and a `StakedPWN` token representing the stake is minted back to the user.

```
function createStake(uint256 amount, uint256 lockUpEpochs) external returns (uint256)

function createStakeOnBehalfOf(address staker, address beneficiary, uint256 amount, uint256 lockUpEpochs)
    public
    returns (uint256 stakeId)
```

Upon creating a stake, the beneficiary receives voting power based on the staked amount and the lockup period. This voting power decreases over time according to a predefined schedule, reaching zero when the lockup period ends.

4.3.2 Stake operations

Stakers can merge two separate stakes into a single one using the `mergeStakes(...)` function. This operation requires that the first stake ends before the second one and that the first stake ends after the current epoch + 1. The function updates the total power in the contract and deletes the two original stakes by burning their corresponding `StakedPWN` tokens. A new stake is then created, with the staker acting as both the owner and beneficiary and the new stake amount being the sum of the merged stakes.

```
function mergeStakes(uint256 stakeId1, address stakeBeneficiary1, uint256 stakeId2, address stakeBeneficiary2)
    external
    returns (uint256 newStakeId)
```

Additionally, stakers can split their stakes into two separate stakes using the `splitStake(...)` function. This operation deletes the original stake and creates two new stakes with the staker as both the owner and beneficiary. The amount is split according to the specified `splitAmount`, and the total power in the contract remains unchanged.

```
function splitStake(uint256 stakeId, address stakeBeneficiary, uint256 splitAmount)
    external
    returns (uint256 newStakeId1, uint256 newStakeId2)
```

The `increaseStake(...)` function allows users to increase the amount or the lockup duration of their stakes. This requires deleting the original stake and burning its corresponding `stakeId`. A new stake is then created with the updated parameters, and the total power in the contract is adjusted to reflect the changes in the amount and lockup epochs.

```
function increaseStake(uint256 stakeId, address stakeBeneficiary, uint256 splitAmount)
    external
    returns (uint256 newStakeId1, uint256 newStakeId2)
```

4.3.3 Withdraw stake

Once the lockup period has expired, users can withdraw their staked `PWN` tokens using the `withdrawStake(...)` function. This process involves deleting the stake, burning the associated `stakeId`, and transferring the staked `PWN` tokens back to the user.

```
    function withdrawStake(uint256 stakeId, address stakeBeneficiary) external
```

5 Risk Rating Methodology

The risk rating methodology used by Nethermind Security follows the principles established by the OWASP Foundation. The severity of each finding is determined by two factors: **Likelihood** and **Impact**.

Likelihood measures how likely the finding is to be uncovered and exploited by an attacker. This factor will be one of the following values:

a) **High**: The issue is trivial to exploit and has no specific conditions that need to be met;

b) **Medium**: The issue is moderately complex and may have some conditions that need to be met;

c) **Low**: The issue is very complex and requires very specific conditions to be met.

When defining the likelihood of a finding, other factors are also considered. These can include but are not limited to motive, opportunity, exploit accessibility, ease of discovery, and ease of exploit.

Impact is a measure of the damage that may be caused if an attacker exploits the finding. This factor will be one of the following values:

a) **High**: The issue can cause significant damage, such as loss of funds or the protocol entering an unrecoverable state;

b) **Medium**: The issue can cause moderate damage, such as impacts that only affect a small group of users or only a particular part of the protocol;

c) **Low**: The issue can cause little to no damage, such as bugs that are easily recoverable or cause unexpected interactions that cause minor inconveniences.

When defining the impact of a finding, other factors are also considered. These can include but are not limited to Data/state integrity, loss of availability, financial loss, and reputation damage. After defining the likelihood and impact of an issue, the severity can be determined according to the table below.

		Severity Risk		
Impact	**High**	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Info/Best Practices	Low	Medium
	Undetermined	Undetermined	Undetermined	Undetermined
		Low	**Medium**	**High**
		Likelihood		

To address issues that do not fit a High/Medium/Low severity, Nethermind Security also uses three more finding severities: **Informational**, **Best Practices**, and **Undetermined**.

a) **Informational** findings do not pose any risk to the application, but they carry some information that the audit team intends to pass to the client formally;

b) **Best Practice** findings are used when some piece of code does not conform with smart contract development best practices;

c) **Undetermined** findings are used when we cannot predict the impact or likelihood of the issue.

6 Issues

6.1 [Info] Invariant break when increasing stake

File(s): VoteEscrowedPWNStake.sol

Description: The contract enforces a limit on the maximum amount of tokens that can be staked by using the `uint88` type. As seen in the following code snippet, the maximum stake is limited by the range of `uint88`:

In the `increaseStake(...)` function, the contract checks that the `additionalAmount` being staked does not exceed the `uint88` limit. However, it only considers the `additionalAmount` and does not account for the user's already staked amount. As a result, the total stake (current stake + additional stake) may exceed the `uint88` limit.

```
function increaseStake(...) external returns (uint256 newStakeId){
    // ...
    // @audit only the additional amount is checked
    if (additionalAmount > type(uint88).max) {
        revert Error.InvalidAmount();
    }
    // ...
}
```

Recommendation(s): Update the check to ensure that the total stake, including both the current stake and the `additionalAmount` (`stake.amount + additionalAmount`), does not exceed the `uint88` limit.

Status: Fixed

Update from the client: Fixed by adding a check for the new amount in range of uint88 after stake increase.

Commit: 0aec9d5176fad7161663a89212bcaebbc5d38c95

6.2 [Info] Absence of a mechanism to disable transfers

File(s): PWN.sol

Description: The `PWN` contract has an `enableTransfers()` function which enables transfers of `PWN` tokens.

```
function enableTransfers() external onlyOwner {  //@audit no mechanism to disable transfers
    if (transfersEnabled) {
        revert Error.TransfersAlreadyEnabled();
    }
    transfersEnabled = true;
}
```

However, there is no mechanism to disable transfers if needed. As such, once transfer is enabled it cannot be disabled.

Recommendation(s): Consider implementing a functionality to disable transfers in the contract.

Status: Acknowledged

Update from the client: It is an intended behaviour. PWN and stPWN tokens should have only one-time transfer enabler.

6.3 [Info] Inconsistency in conditions of `mergeStakes(...)` function

File(s): VoteEscrowedPWNStake.sol

Description: In the `mergeStakes(...)` function, there is an inconsistency between the comment and the actual condition implemented in the code. The comment suggests that both lockup ends must be greater than the current epoch (`newInitialEpoch`). However, the implemented condition only checks that `finalEpoch1` is greater than or equal to the current epoch, not validating the lockup end of the second stake (`finalEpoch2`)

```
function mergeStakes(...) external returns (uint256 newStakeId) {
    // ...
    // the first stake lockup end must be greater than or equal to the second stake lockup end
    // both stake lockup ends must be greater than the current epoch
    // @audit The condition only checks if first stake lockup end is greater than the current epoch
    if (finalEpoch1 < finalEpoch2 || finalEpoch1 <= newInitialEpoch) {
        revert Error.LockUpPeriodMismatch();
    }
    // ...
}
```

Recommendation(s): Revise the intended behavior and update either the condition or the comment to ensure consistency.

Status: Fixed

Update from the client: Commit: 3c286fec003c5d2a86586dc1c5ed08ea9d03b201

7 Documentation Evaluation

Software documentation refers to the written or visual information that describes the functionality, architecture, design, and implementation of software. It provides a comprehensive overview of the software system and helps users, developers, and stakeholders understand how the software works, how to use it, and how to maintain it. Software documentation can take different forms, such as user manuals, system manuals, technical specifications, requirements documents, design documents, and code comments. Software documentation is critical in software development, enabling effective communication between developers, testers, users, and other stakeholders. It helps to ensure that everyone involved in the development process has a shared understanding of the software system and its functionality. Moreover, software documentation can improve software maintenance by providing a clear and complete understanding of the software system, making it easier for developers to maintain, modify, and update the software over time. Smart contracts can use various types of software documentation. Some of the most common types include:

- Technical whitepaper: A technical whitepaper is a comprehensive document describing the smart contract's design and technical details. It includes information about the purpose of the contract, its architecture, its components, and how they interact with each other;
- User manual: A user manual is a document that provides information about how to use the smart contract. It includes step-by-step instructions on how to perform various tasks and explains the different features and functionalities of the contract;
- Code documentation: Code documentation is a document that provides details about the code of the smart contract. It includes information about the functions, variables, and classes used in the code, as well as explanations of how they work;
- API documentation: API documentation is a document that provides information about the API (Application Programming Interface) of the smart contract. It includes details about the methods, parameters, and responses that can be used to interact with the contract;
- Testing documentation: Testing documentation is a document that provides information about how the smart contract was tested. It includes details about the test cases that were used, the results of the tests, and any issues that were identified during testing;
- Audit documentation: Audit documentation includes reports, notes, and other materials related to the security audit of the smart contract. This type of documentation is critical in ensuring that the smart contract is secure and free from vulnerabilities.

These types of documentation are essential for smart contract development and maintenance. They help ensure that the contract is properly designed, implemented, and tested, and they provide a reference for developers who need to modify or maintain the contract in the future.

Remarks about PWN documentation

The PWN team has provided detailed and comprehensive documentation that offers a clear overview of the protocol, along with technical specifications for each token and its main functionalities. Additionally, the team has proactively addressed concerns and questions raised by the Nethermind Security team during their regular meetings.

8 Test Suite Evaluation

8.1 Compilation Output

```
> forge build

[] Compiling...
[] Compiling 21 files with Solc 0.8.25
[] Solc 0.8.25 finished in 17.56s
Compiler run successful with warnings:
Warning (2018): Function state mutability can be restricted to view
   --> src/token/StakedPWN.sol:121:5:
    |
121 |     function _beforeTokenTransfer(
    |     ^ (Relevant source part starts here and spans across multiple lines).

Warning (5574): Contract code size is 40802 bytes and exceeds 24576 bytes (a limit introduced in Spurious Dragon). This
↪  contract may not be deployable on Mainnet. Consider enabling the optimizer (with a low "runs" value!), turning off
↪  revert strings, or using libraries.
  --> script/PWN.token.s.sol:14:1:
   |
14 | contract Deploy is Script {
   | ^ (Relevant source part starts here and spans across multiple lines).

Warning (5574): Contract code size is 24664 bytes and exceeds 24576 bytes (a limit introduced in Spurious Dragon). This
↪  contract may not be deployable on Mainnet. Consider enabling the optimizer (with a low "runs" value!), turning off
↪  revert strings, or using libraries.
 --> test/harness/VoteEscrowedPWNHarness.sol:8:1:
  |
8 | contract VoteEscrowedPWNHarness is VoteEscrowedPWN {
  | ^ (Relevant source part starts here and spans across multiple lines).
```

8.2 Tests Output

```
> forge test
[] Compiling...
No files changed, compilation skipped

Ran 1 test for test/unit/PWNEpochClock.t.sol:PWNEpochClock_Constants_Test
[PASS] test_constants() (gas: 5381)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 5.59ms (1.00ms CPU time)

Ran 1 test for test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_HasVetoed_Test
[PASS] test_shouldReturnCorrectValue() (gas: 16638)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 5.77ms (27.63µs CPU time)

Ran 4 tests for test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_CanVeto_Test
[PASS] test_shouldReturnFalse_whenProposalClosed() (gas: 56156)
[PASS] test_shouldReturnFalse_whenVetoed() (gas: 72719)
[PASS] test_shouldReturnFalse_whenZeroPower() (gas: 37392)
[PASS] test_shouldReturnTrue_whenOpen_whenNotVetoed_whenNonZeroPower() (gas: 36192)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 5.86ms (271.33µs CPU time)

Ran 10 tests for test/unit/token/PWN.t.sol:PWN_ClaimProposalRewardBatch_Test
[PASS] test_mintableCapReached() (gas: 192752)
[PASS] test_shouldEmit_VotingRewardClaimed() (gas: 176444)
[PASS] test_shouldFail_whenCallerHasNotVoted() (gas: 127210)
[PASS] test_shouldFail_whenNoRewardAssigned() (gas: 128797)
[PASS] test_shouldFail_whenProposalNotExecuted() (gas: 141040)
[PASS] test_shouldFail_whenVoterAlreadyClaimedReward() (gas: 127920)
[PASS] test_shouldFail_whenZeroVotingContract() (gas: 20519)
[PASS] test_shouldMintRewardToCaller() (gas: 171002)
[PASS] test_shouldStoreThatVoterClaimedReward() (gas: 162098)
[PASS] test_shouldUseProposalSnapshotAsPastVotesTimepoint() (gas: 161094)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 1.72ms (971.42µs CPU time)
```

```
Ran 6 tests for test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_Cancel_Test
[PASS] testFuzz_shouldFail_whenCallerWithoutPermission(address) (runs: 256, : 30844, ~: 30844)
[PASS] test_shouldEmit_ProposalCancelled() (gas: 52629)
[PASS] test_shouldFail_whenCancelled() (gas: 22986)
[PASS] test_shouldFail_whenExecuted() (gas: 22915)
[PASS] test_shouldFail_whenVetoRatioReached() (gas: 31865)
[PASS] test_shouldStoreCancelledProposal() (gas: 52272)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 11.23ms (10.03ms CPU time)

Ran 2 tests for test/unit/PWNEpochClock.t.sol:PWNEpochClock_Constructor_Test
[PASS] testFuzz_shouldFail_whenInitialEpochTimestampIsInTheFuture(uint256,uint256) (runs: 256, : 41365, ~: 41311)
[PASS] testFuzz_shouldSetInitialEpoch(uint256) (runs: 256, : 128535, ~: 128535)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 18.63ms (13.96ms CPU time)

Ran 1 test for test/unit/governance/PWNOptimisticGovernancePluginSetup.t.sol:PWNOptimisticGovernancePluginSetup
↪  _EncodeInstallationParams_Test
[PASS] testFuzz_shouldReturnEncodedParams(uint32,uint64,address,address) (runs: 256, : 24769, ~: 24788)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 21.98ms (16.14ms CPU time)

Ran 7 tests for test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_Execute_Test
[PASS] testFuzz_shouldBeAbleToExecuteByAnyAddress(address) (runs: 256, : 69311, ~: 69311)
[PASS] testFuzz_shouldFail_whenProposalNotEnded(uint256) (runs: 256, : 27505, ~: 27779)
[PASS] test_shouldEmit_ProposalExecuted() (gas: 67619)
[PASS] test_shouldExecuteActions() (gas: 79012)
[PASS] test_shouldFail_whenAlreadyExecuted() (gas: 71209)
[PASS] test_shouldFail_whenProposalVetoed() (gas: 26922)
[PASS] test_shouldStoreExecutedProposal() (gas: 67206)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 25.25ms (19.44ms CPU time)

Ran 8 tests for test/unit/governance/PWNOptimisticGovernancePluginSetup.t.sol:PWNOptimisticGovernancePluginSetup
_PrepareInstallation_Test
[PASS] test_shouldFail_whenNoProposers() (gas: 47222)
[PASS] test_shouldGrantPermission_CANCELLER_PERMISSION_ID_wherePlugin_whoProposers() (gas: 852024)
[PASS] test_shouldGrantPermission_EXECUTE_PERMISSION_ID_whereDAO_whoPlugin() (gas: 839839)
[PASS] test_shouldGrantPermission_PROPOSER_PERMISSION_ID_wherePlugin_whoProposers() (gas: 851906)
[PASS] test_shouldGrantPermission_UPDATE_OPTIMISTIC_GOVERNANCE_SETTINGS_PERMISSION_ID_wherePlugin_whoDAO() (gas:
↪  838609)
[PASS] test_shouldGrantPermission_UPGRADE_PLUGIN_PERMISSION_ID_wherePlugin_whoDAO() (gas: 838673)
[PASS] test_shouldReturnHelpersArrayWithAllowlist() (gas: 1300795)
[PASS] test_shouldReturnNewlyDeployedAndInitializedPluginClone() (gas: 852129)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 3.62ms (2.43ms CPU time)

Ran 1 test for test/unit/token/PWN.t.sol:PWN_Constructor_Test
[PASS] testFuzz_shouldSetInitialParams(address) (runs: 256, : 1479422, ~: 1479422)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 26.26ms (20.53ms CPU time)

Ran 4 tests for test/unit/governance/PWNOptimisticGovernancePluginSetup.t.sol:PWNOptimisticGovernancePluginSetup
_PrepareUninstallation_Test
[PASS] test_shouldFail_whenHelpersArrayLengthIsNotOne() (gas: 19091)
[PASS] test_shouldRevokePermission_EXECUTE_PERMISSION_ID_whereDAO_whoPlugin() (gas: 36116)
[PASS] test_shouldRevokePermission_UPDATE_OPTIMISTIC_GOVERNANCE_SETTINGS_PERMISSION_ID_wherePlugin_whoDAO() (gas:
↪  33225)
[PASS] test_shouldRevokePermission_UPGRADE_PLUGIN_PERMISSION_ID_wherePlugin_whoDAO() (gas: 33225)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 1.64ms (309.71µs CPU time)

Ran 2 tests for test/unit/governance/DAOExecuteAllowlist.t.sol:DAOExecuteAllowlist_Constructor_Test
[PASS] testFuzz_shouldGetExecutePermissionIdFromDAO(bytes32) (runs: 256, : 474488, ~: 474488)
[PASS] testFuzz_shouldStoreConstructorArguments(address) (runs: 256, : 477706, ~: 477706)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 30.89ms (25.25ms CPU time)

Ran 3 tests for test/unit/token/PWN.t.sol:PWN_EnableTransfers_Test
[PASS] testFuzz_shouldFail_whenCallerNotOwner(address) (runs: 256, : 13660, ~: 13660)
[PASS] test_shouldFail_whenTransferAlreadyEnabled() (gas: 11699)
[PASS] test_shouldStorageTransferEnabledFlag() (gas: 35805)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 8.26ms (7.55ms CPU time)

Ran 1 test for test/unit/governance/DAOExecuteAllowlist.t.sol:DAOExecuteAllowlist_IsAllowed_Test
[PASS] testFuzz_shouldReturnStoredValue(address,bytes4) (runs: 256, : 13436, ~: 13436)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 4.42ms (4.26ms CPU time)

Ran 2 tests for test/unit/token/PWN.t.sol:PWN_SetTransferAllowlist_Test
[PASS] testFuzz_shouldFail_whenCallerNotOwner(address) (runs: 256, : 15591, ~: 15591)
[PASS] testFuzz_shouldSetTransferAllowlist(address,bool) (runs: 256, : 25645, ~: 16239)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 17.31ms (17.00ms CPU time)
```

```
Ran 5 tests for test/unit/token/PWN.t.sol:PWN_AssignProposalReward_Test
[PASS] testFuzz_shouldEmit_ProposalRewardAssigned(uint256,uint256) (runs: 256, : 53217, ~: 54162)
[PASS] testFuzz_shouldNotStoreAssignedReward_whenVotingRewardNotSetForCaller(address) (runs: 256, : 16563, ~: 16563)
[PASS] testFuzz_shouldNotUpdateAssignedReward_whenProposalRewardAlreadySet(uint256) (runs: 256, : 18109, ~: 18105)
[PASS] testFuzz_shouldStoreAssignedReward(uint256,uint256) (runs: 256, : 49629, ~: 50595)
[PASS] test_shouldNotMintNewTokens() (gas: 43061)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 46.66ms (40.91ms CPU time)

Ran 2 tests for test/unit/token/PWN.t.sol:PWN_Mint_Test
[PASS] testFuzz_shouldFail_whenCallerNotOwner(address) (runs: 256, : 13711, ~: 13711)
[PASS] testFuzz_shouldFail_whenInitialSupplyReached(uint256,uint256) (runs: 256, : 21257, ~: 21287)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 12.53ms (11.81ms CPU time)

Ran 2 tests for test/unit/PWNEpochClock.t.sol:PWNEpochClock_CurrentEpoch_Test
[PASS] testFuzz_shouldReturnCorrectEpoch_whenPostInitialTimestamp(uint256) (runs: 256, : 15721, ~: 15619)
[PASS] testFuzz_shouldReturnEpochZero_whenPreInitialTimestamp(uint256) (runs: 256, : 14503, ~: 14794)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 49.64ms (45.00ms CPU time)

Ran 4 tests for test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_CanCancel_Test
[PASS] test_shouldReturnFalse_whenCancelled() (gas: 23390)
[PASS] test_shouldReturnFalse_whenExecuted() (gas: 23167)
[PASS] test_shouldReturnFalse_whenVetoRatioReached() (gas: 23643)
[PASS] test_shouldReturnTrue_whenNotExecuted_whenNotCancelled_whenVetoRatioNotReached() (gas: 23589)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 566.29µs (100.96µs CPU time)

Ran 5 tests for test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_CanExecute_Test
[PASS] test_shouldReturnFalse_whenCancelled() (gas: 25699)
[PASS] test_shouldReturnFalse_whenEnded() (gas: 25944)
[PASS] test_shouldReturnFalse_whenExecuted() (gas: 25476)
[PASS] test_shouldReturnFalse_whenVetoed() (gas: 26321)
[PASS] test_shouldReturnTrue_whenNotExecuted_whenNotEnded_whenNotVetoed() (gas: 26224)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 583.63µs (125.46µs CPU time)

Ran 9 tests for test/unit/token/PWN.t.sol:PWN_ClaimProposalReward_Test
[PASS] testFuzz_shouldEmit_VotingRewardClaimed(uint256,uint256,uint256) (runs: 256, : 104355, ~: 110832)
[PASS] testFuzz_shouldFail_whenCallerHasNotVoted(address) (runs: 256, : 20017, ~: 20017)
[PASS] testFuzz_shouldMintRewardToCaller(uint256,uint256,uint256,uint256,uint256) (runs: 256, : 127749, ~: 117313)
[PASS] test_shouldFail_whenNoRewardAssigned() (gas: 15064)
[PASS] test_shouldFail_whenProposalNotExecuted() (gas: 35695)
[PASS] test_shouldFail_whenVoterAlreadyClaimedReward() (gas: 22570)
[PASS] test_shouldFail_whenZeroVotingContract() (gas: 13558)
[PASS] test_shouldStoreThatVoterClaimedReward() (gas: 83626)
[PASS] test_shouldUseProposalSnapshotAsPastVotesTimepoint() (gas: 87354)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 47.29ms (46.95ms CPU time)

Ran 1 test for test/unit/token/PWN.t.sol:PWN_Constants_Test
[PASS] test_constants() (gas: 17868)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 338.33µs (30.33µs CPU time)

Ran 4 tests for test/integration/Integration.t.sol:Integration_vePWN_Power_Test
[PASS] test_pastAndCurrentPowerIsImmutable_whenIncrease() (gas: 416625)
[PASS] test_pastAndCurrentPowerIsImmutable_whenMerge() (gas: 491512)
[PASS] test_pastAndCurrentPowerIsImmutable_whenSplit() (gas: 369176)
[PASS] test_pastAndCurrentPowerIsImmutable_whenWithdraw() (gas: 335613)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 4.55ms (3.92ms CPU time)

Ran 1 test for test/fork/PWN.governance.fork.t.sol:PWNGovernance_ForkTest
[FAIL: vm.envAddress: environment variable "OSX_PLUGIN_REPO_FACTORY" not found] setUp() (gas: 0)
Suite result: FAILED. 0 passed; 1 failed; 0 skipped; finished in 609.25µs (0.00ns CPU time)

Ran 2 tests for test/unit/token/PWN.t.sol:PWN_Allowance_Test
[PASS] test_shouldFailToCall_decreaseAllowance() (gas: 11048)
[PASS] test_shouldFailToCall_increaseAllowance() (gas: 11016)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 352.75µs (24.17µs CPU time)

Ran 1 test for test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_MinVetoRatio_Test
[PASS] test_shouldReturnStoredMinVetoRatio() (gas: 14601)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 461.50µs (12.25µs CPU time)

Ran 1 test for
↪  test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_TotalVotingPower_Test
[PASS] testFuzz_shouldCallVotingTokenPastTotalSupply(uint256,uint256) (runs: 256, : 20030, ~: 20030)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 4.83ms (4.39ms CPU time)
```

```
Ran 3 tests for test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_Initialize_Test
[PASS] testFuzz_shouldEmit_MembershipContractAnnounced(address) (runs: 256, : 332444, ~: 332444)
[PASS] testFuzz_shouldEmit_OptimisticGovernanceSettingsUpdated(uint32,uint64) (runs: 256, : 343534, ~: 343397)
[PASS] testFuzz_shouldStoreProperties(address,uint32,uint64,address,address) (runs: 256, : 338781, ~: 338637)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 63.10ms (62.23ms CPU time)

Ran 5 tests for test/unit/token/PWN.t.sol:PWN_SetVotingReward_Test
[PASS] testFuzz_shouldEmit_VotingRewardSet(address,uint256) (runs: 256, : 51151, ~: 51498)
[PASS] testFuzz_shouldFail_whenCallerNotOwner(address) (runs: 256, : 18181, ~: 18181)
[PASS] testFuzz_shouldStoreVotingReward(address,uint256) (runs: 256, : 50295, ~: 50537)
[PASS] test_shouldFail_whenVotingContractZeroAddress() (gas: 15839)
[PASS] test_shouldFail_whenVotingRewardBiggerThanMax(uint256) (runs: 256, : 25355, ~: 25329)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 34.68ms (34.35ms CPU time)

Ran 2 tests for test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_GetProposal_Test
[PASS] testFuzz_shouldReturnProposal(bool,bool,uint64,uint64,uint64,uint256,uint256,uint256) (runs: 256, : 162076, ~:
↪  162068)
[PASS] test_shouldReturnCorrectOpenStatus() (gas: 73610)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 48.03ms (47.54ms CPU time)

Ran 2 tests for test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_IsMember_Test
[PASS] testFuzz_shouldReturnTrue_whenVotingPowerGreaterThanZero(uint256) (runs: 256, : 24802, ~: 24799)
[PASS] test_shouldReturnFalse_whenVotingPowerEqualZero() (gas: 21231)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 5.48ms (4.72ms CPU time)

Ran 1 test for
↪  test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_IsMinVetoRatioReached_Test
[PASS] test_shouldReturnTrue_whenMinVetoPowerReached() (gas: 29017)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 494.42µs (44.38µs CPU time)

Ran 1 test for test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_MinDuration_Test
[PASS] test_shouldReturnStoredMinDuration() (gas: 14744)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 449.67µs (11.33µs CPU time)

Ran 1 test for
↪  test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_GetVotingToken_Test
[PASS] testFuzz_shouldReturnVotingToken(address) (runs: 256, : 11396, ~: 11396)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 3.38ms (2.92ms CPU time)

Ran 5 tests for
↪  test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_UpdateGovernanceSettings_Test
[PASS] testFuzz_shouldEmit_OptimisticGovernanceSettingsUpdated(uint256,uint256) (runs: 256, : 41539, ~: 41805)
[PASS] testFuzz_shouldStoreNewSettings(uint256,uint256) (runs: 256, : 41223, ~: 41511)
[PASS] test_shouldFail_whenCallerWithoutPermission(address) (runs: 256, : 30980, ~: 30980)
[PASS] test_shouldFail_whenMinDurationOutOfBounds() (gas: 42234)
[PASS] test_shouldFail_whenMinVetoRatioOutOfBounds() (gas: 41401)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 45.58ms (45.07ms CPU time)

Ran 6 tests for test/unit/token/PWN.t.sol:PWN_TransferCallback_Test
[PASS] testFuzz_shouldAllowTransfer_whenTransfersDisabled_whenAddrInAllowlist_whenCallerOperator(address) (runs: 256, :
↪  80074, ~: 80060)
[PASS] testFuzz_shouldAllowTransfer_whenTransfersDisabled_whenAddrInAllowlist_whenCallerOwner(address) (runs: 256, :
↪  56097, ~: 56080)
[PASS] testFuzz_shouldAllowTransfer_whenTransfersEnabled(address) (runs: 256, : 52536, ~: 52520)
[PASS] testFuzz_shouldBlockTransfer_whenTransfersDisabled_whenAddrNotInAllowlist(address) (runs: 256, : 45223, ~: 45223)
[PASS] test_shouldAllowBurn() (gas: 22223)
[PASS] test_shouldAllowMint() (gas: 34427)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 63.26ms (67.57ms CPU time)

Ran 7 tests for test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_Veto_Test
[PASS] testFuzz_shouldBeAbleToVetoDuringVotingPeriod(uint256) (runs: 256, : 80854, ~: 81168)
[PASS] testFuzz_shouldFail_whenProposalEnded(uint256) (runs: 256, : 28510, ~: 28531)
[PASS] testFuzz_shouldFail_whenProposalNotStarted(uint256) (runs: 256, : 123477, ~: 123523)
[PASS] testFuzz_shouldStoreVeto(uint256) (runs: 256, : 83358, ~: 83080)
[PASS] test_shouldEmit_VetoCast() (gas: 78643)
[PASS] test_shouldFail_whenVoterAlreadyVetoed() (gas: 78439)
[PASS] test_shouldFail_whenVoterHasNoPower() (gas: 32282)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 36.39ms (35.88ms CPU time)

Ran 1 test for
↪  test/unit/governance/PWNOptimisticGovernancePluginSetup.t.sol:PWNOptimisticGovernancePluginSetup_Constructor_Test
[PASS] test_shouldDeployOptimisticGovernancePluginBase() (gas: 4404599)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 606.96µs (123.46µs CPU time)
```

```
Ran 2 tests for test/unit/token/PWN.t.sol:PWN_Burn_Test
[PASS] testFuzz_shouldBurnCallersTokens(uint256,uint256) (runs: 256, : 158547, ~: 158710)
[PASS] testFuzz_shouldNotDecreseOwnerMintedAmount(uint256,uint256) (runs: 256, : 158502, ~: 158345)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 105.44ms (104.54ms CPU time)

Ran 1 test for test/unit/governance/PWNOptimisticGovernancePluginSetup.t.sol:PWNOptimisticGovernancePluginSetup
_DecodeInstallationParams_Test
[PASS] testFuzz_shouldReturnDecodedParams(uint32,uint64,address,address) (runs: 256, : 27080, ~: 27118)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 14.66ms (14.17ms CPU time)

Ran 11 tests for test/unit/governance/PWNTokenGovernancePlugin.t.sol:PWNTokenGovernancePlugin_CreateProposal_Test
[PASS] testFuzz_shouldFail_whenCallerWithoutMinPower(uint256) (runs: 256, : 34079, ~: 33781)
[PASS] testFuzz_shouldFail_whenInvalidStartOrEndDate(uint256) (runs: 256, : 65896, ~: 66341)
[PASS] testFuzz_shouldIncreaseAndReturnProposalId(uint256) (runs: 256, : 147226, ~: 146982)
[PASS] testFuzz_shouldStoreProposal(uint256,uint256,uint256,uint256,uint256) (runs: 256, : 484230, ~: 484765)
[PASS] test_shouldAssignProposalReward() (gas: 138990)
[PASS] test_shouldCreateProposal_whenNoPower_whenMinPowerZero_whenNoVoting() (gas: 107828)
[PASS] test_shouldEmit_ProposalCreated() (gas: 94149)
[PASS] test_shouldFail_whenNoPower_whenMinPowerZero_whenVoting() (gas: 113039)
[PASS] test_shouldFail_whenNoVotingPower() (gas: 33178)
[PASS] test_shouldGetCurrentEpochFromEpochClock() (gas: 113987)
[PASS] test_shouldVote_whenVoteOptionProvided() (gas: 373595)
Suite result: ok. 11 passed; 0 failed; 0 skipped; finished in 140.87ms (140.34ms CPU time)

Ran 2 tests for test/unit/EpochPowerLib.t.sol:EpochPowerLib_Test
[PASS] testFuzz_shouldReturnCorrectPower(bytes32,uint256,int104) (runs: 256, : 9169, ~: 8933)
[PASS] testFuzz_shouldUpdateCorrectPower(bytes32,uint256,int104,int104) (runs: 256, : 11147, ~: 10589)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 15.86ms (15.81ms CPU time)

Ran 2 tests for test/unit/governance/DAOExecuteAllowlist.t.sol:DAOExecuteAllowlist_SetAllowlist_Test
[PASS] testFuzz_shouldFail_whenCallerIsNotDAO(address) (runs: 256, : 14913, ~: 14913)
[PASS] testFuzz_shouldStoreAllowlistValue(address,bytes4) (runs: 256, : 26498, ~: 26478)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 24.81ms (24.64ms CPU time)

Ran 5 tests for test/unit/governance/PWNTokenGovernancePluginSetup.t.sol:PWNTokenGovernancePluginSetup
_PrepareInstallation_Test
[PASS] test_shouldGrantPermission_EXECUTE_PERMISSION_ID_whereDAO_whoPlugin() (gas: 385072)
[PASS] test_shouldGrantPermission_UPDATE_TOKEN_GOVERNANCE_SETTINGS_PERMISSION_ID_wherePlugin_whoDAO() (gas: 383997)
[PASS] test_shouldGrantPermission_UPGRADE_PLUGIN_PERMISSION_ID_wherePlugin_whoDAO() (gas: 383974)
[PASS] test_shouldReturnEmptyHelpersArray() (gas: 382359)
[PASS] test_shouldReturnNewlyDeployedAndInitializedPluginClone() (gas: 399759)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 1.85ms (1.22ms CPU time)

Ran 4 tests for test/unit/governance/PWNTokenGovernancePluginSetup.t.sol:PWNTokenGovernancePluginSetup
_PrepareUninstallation_Test
[PASS] test_shouldFail_whenHelpersArrayLengthIsNotZero() (gas: 43238)
[PASS] test_shouldRevokePermission_EXECUTE_PERMISSION_ID_whereDAO_whoPlugin() (gas: 31400)
[PASS] test_shouldRevokePermission_UPDATE_TOKEN_GOVERNANCE_SETTINGS_PERMISSION_ID_wherePlugin_whoDAO() (gas: 30722)
[PASS] test_shouldRevokePermission_UPGRADE_PLUGIN_PERMISSION_ID_wherePlugin_whoDAO() (gas: 30611)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 2.47ms (1.52ms CPU time)

Ran 4 tests for test/unit/token/VoteEscrowedPWN.Base.t.sol:VoteEscrowedPWN_Base_IERC20_Test
[PASS] testFuzz_shouldReturnStakerPower_forBalanceOf(address,uint256) (runs: 256, : 83444, ~: 83833)
[PASS] testFuzz_shouldReturnTotalPower_forTotalSupply(uint256) (runs: 256, : 60757, ~: 61146)
[PASS] test_shouldHaveTransferDisabled() (gas: 20526)
[PASS] test_shouldReturnCorrectMetadata() (gas: 10188)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 12.23ms (11.22ms CPU time)

Ran 8 tests for test/unit/governance/PWNTokenGovernancePlugin.t.sol:PWNTokenGovernancePlugin_Execute_Test
[PASS] testFuzz_shouldBeAbleToExecuteByAnyAddress(address) (runs: 256, : 75592, ~: 75592)
[PASS] testFuzz_shouldFail_whenMinParticipationNotReached(uint256) (runs: 256, : 37884, ~: 37582)
[PASS] testFuzz_shouldFail_whenProposalNotEnded(uint256) (runs: 256, : 29717, ~: 29987)
[PASS] test_shouldEmit_ProposalExecuted() (gas: 73844)
[PASS] test_shouldExecuteActions() (gas: 85252)
[PASS] test_shouldFail_whenAlreadyExecuted() (gas: 14870)
[PASS] test_shouldFail_whenSupportThresholdNotReached() (gas: 28368)
[PASS] test_shouldStoreExecutedProposal() (gas: 73148)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 26.54ms (25.97ms CPU time)

Ran 3 tests for test/unit/token/StakedPWN.t.sol:StakedPWN_Burn_Test
[PASS] testFuzz_shouldBurnStakedPWNToken(address,uint256) (runs: 256, : 14807, ~: 14803)
[PASS] testFuzz_shouldFail_whenCallerNotSupplyManager(address) (runs: 256, : 11125, ~: 11125)
[PASS] test_shouldBurnStakedPWNToken_whenTransfersDisabled(address,uint256) (runs: 256, : 13647, ~: 13647)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 30.69ms (30.45ms CPU time)
```

```
Ran 1 test for test/unit/governance/PWNTokenGovernancePluginSetup.t.sol:PWNTokenGovernancePluginSetup_Constructor_Test
[PASS] test_shouldDeployTokenGovernancePluginBase() (gas: 3834470)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 1.53ms (242.17µs CPU time)

Ran 4 tests for test/unit/token/VoteEscrowedPWN.Base.t.sol:VoteEscrowedPWN_Base_Votes_Test
[PASS] testFuzz_shouldReturnStakerPower_forGetPastVotes(address,uint256,uint16) (runs: 256, : 75245, ~: 76878)
[PASS] testFuzz_shouldReturnStakerPower_forGetVotes(address,uint256) (runs: 256, : 83876, ~: 83876)
[PASS] testFuzz_shouldReturnTotalPower_forGetPastTotalSupply(uint256,uint16) (runs: 256, : 52916, ~: 54083)
[PASS] test_shouldHaveDisabledDelegation() (gas: 13609)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 31.26ms (30.23ms CPU time)

Ran 1 test for test/unit/governance/PWNTokenGovernancePluginSetup.t.sol:PWNTokenGovernancePluginSetup
_DecodeInstallationParams_Test
[PASS] testFuzz_shouldReturnDecodedParams(uint32,uint32,uint64,uint256,address,address,address) (runs: 256, : 37336, ~:
↪  37570)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 16.21ms (14.80ms CPU time)

Ran 1 test for test/unit/token/StakedPWN.t.sol:StakedPWN_Constructor_Test
[PASS] testFuzz_shouldSetInitialParams(address,address,address) (runs: 256, : 1314228, ~: 1314228)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 22.48ms (22.24ms CPU time)

Ran 1 test for
↪  test/unit/governance/PWNTokenGovernancePluginSetup.t.sol:PWNTokenGovernancePluginSetup_EncodeInstallationParams_Test
[PASS] testFuzz_shouldReturnEncodedParams(uint32,uint32,uint64,uint256,address,address,address) (runs: 256, : 37060, ~:
↪  37216)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 6.34ms (5.63ms CPU time)

Ran 3 tests for test/unit/token/StakedPWN.t.sol:StakedPWN_EnableTransfer_Test
[PASS] testFuzz_shouldFail_whenCallerNotOwner(address) (runs: 256, : 13704, ~: 13704)
[PASS] test_shouldEnableTransfers() (gas: 35445)
[PASS] test_shouldFail_whenTransfersEnabled() (gas: 11542)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 5.56ms (4.94ms CPU time)

Ran 3 tests for test/unit/governance/PWNTokenGovernancePlugin.t.sol:PWNTokenGovernancePlugin_Initialize_Test
[PASS] testFuzz_shouldEmit_MembershipContractAnnounced(address) (runs: 256, : 382713, ~: 382713)
[PASS] testFuzz_shouldEmit_TokenGovernanceSettingsUpdated(uint32,uint32,uint64,uint256) (runs: 256, : 398682, ~: 398766)
[PASS] testFuzz_shouldStoreProperties(address,uint32,uint32,uint64,uint256,address,address,address) (runs: 256, :
↪  393228, ~: 393423)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 62.71ms (61.53ms CPU time)

Ran 3 tests for test/unit/token/VoteEscrowedPWN.Power.t.sol:StakedPWN_BeneficiaryOfStakesAt_Test
[PASS] testFuzz_shouldReturnEmpty_whenEpochBeforeFirstStake(uint256) (runs: 256, : 108243, ~: 108300)
[PASS] testFuzz_shouldReturnEmpty_whenNoStakes(uint16) (runs: 256, : 10829, ~: 10829)
[PASS] test_shouldReturnListOfStakesForBeneficiary() (gas: 268908)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 23.09ms (22.10ms CPU time)

Ran 1 test for test/unit/token/StakedPWN.t.sol:StakedPWN_Metadata_Test
[PASS] testFuzz_shouldReturnMetadataURI(uint256,string) (runs: 256, : 15915, ~: 15680)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 8.92ms (8.69ms CPU time)

Ran 1 test for test/unit/token/VoteEscrowedPWN.Stake.t.sol:VoteEscrowedPWN_Stake_GetStakes_Test
[PASS] test_shouldReturnStakeData() (gas: 563992)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 740.75µs (267.38µs CPU time)

Ran 3 tests for test/unit/token/StakedPWN.t.sol:StakedPWN_Mint_Test
[PASS] testFuzz_shouldFail_whenCallerNotSupplyManager(address) (runs: 256, : 12134, ~: 12134)
[PASS] testFuzz_shouldMintStakedPWNToken(address,uint256) (runs: 256, : 69151, ~: 69151)
[PASS] testFuzz_shouldMintStakedPWNToken_whenTransfersDisabled(address,uint256) (runs: 256, : 65139, ~: 65139)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 22.81ms (22.56ms CPU time)

Ran 7 tests for test/unit/token/VoteEscrowedPWN.StakeMetadata.t.sol:VoteEscrowedPWN_StakeMetadata_MakeFields_Test
[PASS] testFuzz_makeStakedAmount(uint256,uint256,address) (runs: 256, : 41212, ~: 40641)
[PASS] test_makeApiUriWith() (gas: 70166)
[PASS] test_makeDescription() (gas: 7707)
[PASS] test_makeExternalUrl() (gas: 68098)
[PASS] test_makeMultiplier() (gas: 32633)
[PASS] test_makeName() (gas: 21575)
[PASS] test_makePowerChanges() (gas: 23110)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 44.15ms (43.52ms CPU time)

Ran 2 tests for test/unit/token/StakedPWN.t.sol:StakedPWN_SetTransferAllowlist_Test
[PASS] testFuzz_shouldFail_whenCallerNotOwner(address) (runs: 256, : 15513, ~: 15513)
[PASS] testFuzz_shouldSetTransferAllowlist(address,bool) (runs: 256, : 25943, ~: 16226)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 27.55ms (27.31ms CPU time)
```

```
Ran 8 tests for test/unit/governance/PWNTokenGovernancePlugin.t.sol:PWNTokenGovernancePlugin_Vote_Test
[PASS] testFuzz_shouldFail_whenProposalEnded(uint256) (runs: 256, : 53334, ~: 53355)
[PASS] testFuzz_shouldFail_whenProposalNotStarted(uint256) (runs: 256, : 50796, ~: 50895)
[PASS] testFuzz_shouldStoreVoteOption(uint8) (runs: 256, : 78310, ~: 78345)
[PASS] testFuzz_shouldUpdateTally(uint8,uint256) (runs: 256, : 76632, ~: 76752)
[PASS] test_shouldEmit_VoteCast() (gas: 99927)
[PASS] test_shouldFail_whenAlreadyVoted() (gas: 99726)
[PASS] test_shouldFail_whenNoVotingPower() (gas: 52923)
[PASS] test_shouldFail_whenVoteOptionNone() (gas: 24592)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 61.32ms (60.74ms CPU time)

Ran 6 tests for test/unit/token/VoteEscrowedPWNTest.t.sol:VoteEscrowedPWN_Exposed_Test
[PASS] testFuzz_epochsToNextPowerChange_whenLessThanFiveYears_whenDivisibleByYear(uint8) (runs: 256, : 10343, ~: 10517)
[PASS] testFuzz_epochsToNextPowerChange_whenLessThanFiveYears_whenNotDivisibleByYear(uint8) (runs: 256, : 13524, ~:
↪  13760)
[PASS] testFuzz_epochsToNextPowerChange_whenMoreThanFiveYears(uint8) (runs: 256, : 10741, ~: 10958)
[PASS] testFuzz_powerChangeMultipliers_decreasePower(uint256,uint8) (runs: 256, : 13140, ~: 13158)
[PASS] testFuzz_powerChangeMultipliers_initialPower(uint256,uint8) (runs: 256, : 12966, ~: 12873)
[PASS] test_powerChangeMultipliers_powerChangesShouldSumToZero(uint8) (runs: 256, : 21833, ~: 25832)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 64.06ms (63.06ms CPU time)
proptest: Aborting shrinking after the PROPTEST_MAX_SHRINK_ITERS environment variable or ProptestConfig.max_shrink_iters
↪  iterations (set 0 to a large(r) value to shrink more; current configuration: 0 iterations)
proptest: Saving this and future failures in cache/fuzz/failures
proptest: If this test was run on a CI system, you may wish to add the following line to your copy of the file. (You may
↪  need to create it.)
cc 8031e7107744ca2fe3345769962df020b78565c6121642c159429324e286a47c

Ran 6 tests for test/unit/token/StakedPWN.t.sol:StakedPWN_Transfer_Test
[PASS] testFuzz_shouldAllowTransfer_whenTransfersDisabled_whenAddrInAllowlist_whenCallerOperator(address) (runs: 256, :
↪  75617, ~: 75602)
[PASS] testFuzz_shouldAllowTransfer_whenTransfersDisabled_whenAddrInAllowlist_whenCallerOwner(address) (runs: 256, :
↪  52711, ~: 52695)
[PASS] testFuzz_shouldAllowTransfer_whenTransfersEnabled(address) (runs: 256, : 50746, ~: 50730)
[PASS] test_shouldAllowBurn_whenTransferDisabled() (gas: 14833)
[PASS] test_shouldAllowMint_whenTransferDisabled() (gas: 66169)
[FAIL: Error != expected error: ERC721: invalid token ID != TransfersDisabled(); counterexample:
↪  calldata=0x436f798100000000000000000000000000000000000000000000000000000000000000000000000000000000
↪  args=[0x0000000000000000000000000000000000000000]]
↪  test_shouldFail_whenTransfersNotEnabled_whenAddrNotInAllowlist(address) (runs: 105, : 18334, ~: 18334)
Suite result: FAILED. 5 passed; 1 failed; 0 skipped; finished in 47.74ms (47.45ms CPU time)

Ran 2 tests for test/unit/token/VoteEscrowedPWN.Base.t.sol:VoteEscrowedPWN_Base_Clock_Test
[PASS] test_shouldCallEpochClock() (gas: 18230)
[PASS] test_shouldReturnCorrectClockMode() (gas: 7074)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 960.29µs (31.62µs CPU time)

Ran 1 test for test/unit/token/VoteEscrowedPWNTest.t.sol:VoteEscrowedPWN_Helpers_Test
[PASS] testFuzzHelper_storeStake(uint256,uint16,uint8,uint104) (runs: 256, : 26179, ~: 26176)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 17.37ms (16.91ms CPU time)

Ran 22 tests for test/UseCases.t.sol:UseCases
[PASS] testUseCase_createStake_10Year() (gas: 2572949)
[PASS] testUseCase_createStake_1Year() (gas: 1853011)
[PASS] testUseCase_createStake_2Year() (gas: 1987410)
[PASS] testUseCase_createStake_3Year() (gas: 2142688)
[PASS] testUseCase_createStake_4Year() (gas: 2277201)
[PASS] testUseCase_createStake_5Year() (gas: 2435156)
[PASS] testUseCase_getEpochPower_10Stake() (gas: 2455855)
[PASS] testUseCase_getEpochPower_1Stake() (gas: 413968)
[PASS] testUseCase_getEpochPower_2Stake() (gas: 629467)
[PASS] testUseCase_getEpochPower_3Stake() (gas: 845032)
[PASS] testUseCase_getEpochPower_4Stake() (gas: 1060646)
[PASS] testUseCase_getEpochPower_5Stake() (gas: 1276237)
[PASS] testUseCase_getEpochPower_6Stake() (gas: 1513803)
[PASS] testUseCase_getEpochPower_7Stake() (gas: 1751684)
[PASS] testUseCase_getEpochPower_8Stake() (gas: 1989562)
[PASS] testUseCase_getEpochPower_9Stake() (gas: 2227465)
[PASS] testUseCase_transferStake_10Year() (gas: 2723170)
[PASS] testUseCase_transferStake_1Year() (gas: 2003297)
[PASS] testUseCase_transferStake_2Year() (gas: 2137672)
[PASS] testUseCase_transferStake_3Year() (gas: 2292865)
[PASS] testUseCase_transferStake_4Year() (gas: 2427486)
[PASS] testUseCase_transferStake_5Year() (gas: 2585354)
Suite result: ok. 22 passed; 0 failed; 0 skipped; finished in 19.41ms (18.82ms CPU time)
```

```
Ran 3 tests for test/unit/token/VoteEscrowedPWN.Power.t.sol:VoteEscrowedPWN_Power_StakePowers_Test
[PASS] testFuzz_shouldReturnCorrectEpochPowers(uint256,uint256,uint256) (runs: 257, : 155595, ~: 189632)
[PASS] test_shouldFail_whenInvalidAmount() (gas: 26412)
[PASS] test_shouldFail_whenInvalidLockUpEpochs() (gas: 17746)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 54.97ms (53.93ms CPU time)

Ran 4 tests for test/unit/token/VoteEscrowedPWN.Stake.t.sol:VoteEscrowedPWN_Stake_GetStake_Test
[PASS] test_shouldReturnRemainingEpochsHigherThanLockup_whenStakeHaveNotStarted() (gas: 352139)
[PASS] test_shouldReturnStakeData() (gas: 351616)
[PASS] test_shouldReturnZeroMultiplier_whenStakeExpired() (gas: 352006)
[PASS] test_shouldReturnZeroMultiplier_whenStakeHaveNotStarted() (gas: 351942)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 1.02ms (555.96µs CPU time)

Ran 12 tests for test/unit/token/VoteEscrowedPWN.Stake.t.sol:VoteEscrowedPWN_Stake_WithdrawStake_Test
[PASS] testFuzz_shouldEmit_StakePowerDelegated(address) (runs: 256, : 236569, ~: 236554)
[PASS] testFuzz_shouldFail_whenCallerIsNotStakeOwner(address) (runs: 256, : 26836, ~: 26836)
[PASS] testFuzz_shouldFail_whenStillLock(uint256) (runs: 256, : 383927, ~: 383971)
[PASS] testFuzz_shouldPass_whenUnlocked(uint256) (runs: 257, : 449800, ~: 449830)
[PASS] testFuzz_shouldUpdateStakesOfBeneficiary_whenOwnerIsNotBeneficiary(address) (runs: 257, : 242610, ~: 242596)
[PASS] test_shouldBurnStakedPWNToken() (gas: 91684)
[PASS] test_shouldEmit_StakeWithdrawn() (gas: 92748)
[PASS] test_shouldFail_whenIncorrectStakeBeneficiary() (gas: 31029)
[PASS] test_shouldFail_whenStakeDoesNotExist() (gas: 25806)
[PASS] test_shouldNotDeleteStakeData() (gas: 93965)
[PASS] test_shouldTransferPWNTokenToStaker() (gas: 93512)
[PASS] test_shouldUpdateStakesOfBeneficiary_whenOwnerIsBeneficiary() (gas: 93999)
Suite result: ok. 12 passed; 0 failed; 0 skipped; finished in 229.57ms (229.51ms CPU time)

Ran 4 tests for test/unit/token/VoteEscrowedPWN.Stake.t.sol:VoteEscrowedPWN_Stake_CreateStakeOnBehalfOf_Test
[PASS] test_shouldAddStakeToBeneficiary() (gas: 229995)
[PASS] test_shouldEmit_StakeCreated() (gas: 224679)
[PASS] test_shouldMintStakedPWNTokenToStaker() (gas: 224669)
[PASS] test_shouldTransferPWNTokensFromCaller() (gas: 224045)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 693.63µs (164.38µs CPU time)

Ran 9 tests for test/unit/token/VoteEscrowedPWN.Stake.t.sol:VoteEscrowedPWN_Stake_DelegateStakePower_Test
[PASS] testFuzz_shouldAddStakeToNewBeneficiary_whenFirstStake(address) (runs: 256, : 273438, ~: 273423)
[PASS] testFuzz_shouldAddStakeToNewBeneficiary_whenNotSameEpoch(address) (runs: 256, : 449518, ~: 449505)
[PASS] testFuzz_shouldAddStakeToNewBeneficiary_whenSameEpoch(address) (runs: 256, : 340227, ~: 340213)
[PASS] testFuzz_shouldEmit_StakePowerDelegated(address,address) (runs: 257, : 323528, ~: 323495)
[PASS] testFuzz_shouldFail_whenCallerNotStakeOwner(address) (runs: 257, : 188702, ~: 188687)
[PASS] testFuzz_shouldFail_whenProvidedCurrentBeneficiaryNotStakeBeneficiary(address) (runs: 257, : 194846, ~: 194828)
[PASS] testFuzz_shouldRemoveStakeFromCurrentBeneficiary_whenNotSameEpoch(address) (runs: 257, : 331986, ~: 331969)
[PASS] testFuzz_shouldRemoveStakeFromCurrentBeneficiary_whenSameEpoch(address) (runs: 257, : 276429, ~: 276413)
[PASS] test_shouldFail_whenNewBeneficiarySameAsCurrentBeneficiary() (gas: 184494)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 383.62ms (382.45ms CPU time)

Ran 4 tests for
↪  test/unit/token/VoteEscrowedPWN.StakeMetadata.t.sol:VoteEscrowedPWN_StakeMetadata_ComputeAttributes_Test
[PASS] testFuzz_shouldComputeLockUpDuration(uint256) (runs: 257, : 290891, ~: 309737)
[PASS] testFuzz_shouldComputeOwner(address) (runs: 257, : 58925, ~: 58925)
[PASS] testFuzz_shouldComputePowerChangesAndFieldsDerivedFromIt(uint256,uint256,uint256) (runs: 257, : 487641, ~:
↪  576878)
[PASS] testFuzz_shouldComputeStakedAmount(uint256) (runs: 257, : 310870, ~: 310644)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 359.91ms (334.79ms CPU time)

Ran 1 test for test/unit/token/VoteEscrowedPWN.Power.t.sol:VoteEscrowedPWN_Power_TotalPowers_Test
[PASS] testFuzz_shouldReturnTotalPowersForEpochs(uint256) (runs: 257, : 1196353, ~: 1123345)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 427.81ms (427.35ms CPU time)

Ran 6 tests for test/unit/token/VoteEscrowedPWN.Power.t.sol:VoteEscrowedPWN_Power_CalculateTotalPower_Test
[PASS] testFuzz_shouldCalculateTotalPowers(uint256,uint256,uint256) (runs: 256, : 1217937, ~: 1152130)
[PASS] testFuzz_shouldEmit_TotalPowerCalculated(uint256,uint256) (runs: 257, : 309405, ~: 232399)
[PASS] testFuzz_shouldFail_whenEpochDoesNotEnded(uint256) (runs: 257, : 19730, ~: 19703)
[PASS] testFuzz_shouldFail_whenTotalPowerAlreadyCalculated(uint256,uint256) (runs: 257, : 20321, ~: 20408)
[PASS] testFuzz_shouldStoreNewLastCalculatedTotalPower(uint256,uint256) (runs: 257, : 284879, ~: 200360)
[PASS] test_calculateTotalPower_shouldUseCurrentEpoch() (gas: 1162210)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 558.11ms (705.04ms CPU time)

Ran 2 tests for test/unit/token/VoteEscrowedPWN.Power.t.sol:VoteEscrowedPWN_Power_TotalPowerAt_Test
[PASS] testFuzz_shouldReturnComputedPower_whenEpochIsNotYetCalculated(uint256,uint256,uint256) (runs: 513, : 1220938,
↪  ~: 1201594)
[PASS] testFuzz_shouldReturnStoredPower_whenEpochIsCalculated(uint256,uint256,uint256) (runs: 513, : 1233179, ~:
↪  1203752)
```

```
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 763.58ms (1.39s CPU time)

Ran 10 tests for test/unit/token/VoteEscrowedPWN.Stake.t.sol:VoteEscrowedPWN_Stake_CreateStake_Test
[PASS] testFuzz_shouldStoreTotalPowerChanges(uint256,uint256) (runs: 257, : 1740831, ~: 1885493)
[PASS] testFuzz_shouldUpdateTotalPowerChanges(uint256,uint256,uint256,uint256) (runs: 257, : 2029412, ~: 2066844)
[PASS] test_shouldAddStakeToBeneficiary() (gas: 225676)
[PASS] test_shouldEmit_StakeCreated() (gas: 220213)
[PASS] test_shouldFail_whenInvalidAmount() (gas: 17899)
[PASS] test_shouldFail_whenInvalidLockUpEpochs() (gas: 12483)
[PASS] test_shouldIncreaseStakeId() (gas: 212205)
[PASS] test_shouldMintStakedPWNToken() (gas: 220406)
[PASS] test_shouldStoreStakeData() (gas: 217534)
[PASS] test_shouldTransferPWNTokens() (gas: 219760)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 1.02s (1.70s CPU time)

Ran 1 test for test/unit/token/VoteEscrowedPWN.Power.t.sol:VoteEscrowedPWN_Power_StakerPowers_Test
[PASS] testFuzz_shouldReturnStakerPowersForEpochs(uint256,uint256) (runs: 257, : 7108415, ~: 6949092)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 1.33s (1.33s CPU time)

Ran 7 tests for
↪  test/unit/governance/PWNOptimisticGovernancePlugin.t.sol:PWNOptimisticGovernancePlugin_CreateProposal_Test
[PASS] testFuzz_shouldFail_whenCallerWithoutPermission(address) (runs: 256, : 31982, ~: 31982)
[PASS] testFuzz_shouldFail_whenInvalidStartOrEndDate(uint256) (runs: 256, : 71015, ~: 71561)
[PASS] testFuzz_shouldIncreaseAndReturnProposalId(uint256) (runs: 256, : 143477, ~: 143229)
[PASS] testFuzz_shouldStoreProposal(uint256,uint256,uint256,uint256,uint256) (runs: 256, : 480900, ~: 481733)
[PASS] test_shouldEmit_ProposalCreated() (gas: 91254)
[PASS] test_shouldFail_whenNoVotingPower() (gas: 35765)
[PASS] test_shouldGetCurrentEpochFromEpochClock() (gas: 111155)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 1.93s (149.14ms CPU time)

Ran 5 tests for test/integration/Integration.t.sol:Integration_vePWN_Stake_Test
[PASS] testFuzz_createStake(uint256,uint256) (runs: 512, : 422663, ~: 413085)
[PASS] testFuzz_increaseStake(uint256,uint256,uint256,uint256,uint256) (runs: 512, : 645836, ~: 647322)
[PASS] testFuzz_mergeStakes(uint256,uint256,uint256,uint256,uint256,uint256) (runs: 512, : 725258, ~: 731250)
[PASS] testFuzz_splitStake(uint256,uint256,uint256,uint256) (runs: 512, : 565315, ~: 560541)
[PASS] testFuzz_withdrawStake(uint256,uint256) (runs: 512, : 471396, ~: 457900)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 1.90s (993.47ms CPU time)

Ran 15 tests for test/unit/token/VoteEscrowedPWN.Stake.t.sol:VoteEscrowedPWN_Stake_MergeStakes_Test
[PASS] testFuzz_shouldClearRemainingTotalPowerChanges_whenDifferentPowerChangeEpochs(uint256,uint256,uint256,uint256,
uint256,uint256) (runs: 513, : 796864, ~: 798225)
[PASS] testFuzz_shouldCreateNewStake(uint256,uint256,uint256) (runs: 257, : 660377, ~: 711327)
[PASS] testFuzz_shouldEmit_StakeMerged(uint256,uint256,uint256) (runs: 257, : 651913, ~: 711504)
[PASS] testFuzz_shouldFail_whenBothStakesLockupEnded(uint8) (runs: 257, : 263790, ~: 254108)
[PASS] testFuzz_shouldFail_whenFirstLockUpSmallerThanSecond(uint256) (runs: 257, : 562791, ~: 590416)
[PASS] testFuzz_shouldUpdateStakesOfBeneficiary_whenOwnerIsNotBeneficiary(address,address) (runs: 257, : 694208, ~:
↪  694176)
[PASS] testFuzz_shouldUpdateTotalPowerChanges(uint256,uint256,uint256,uint256,uint256,uint256) (runs: 513, : 2056959,
↪  ~: 2044808)
[PASS] test_shouldBurnOriginalStakedPWNTokens() (gas: 513264)
[PASS] test_shouldEmit_StakePowerDelegated() (gas: 518126)
[PASS] test_shouldFail_whenCallerNotFirstStakeOwner() (gas: 448749)
[PASS] test_shouldFail_whenCallerNotSecondStakeOwner() (gas: 507896)
[PASS] test_shouldFail_whenIncorrectFirstStakeBeneficiary() (gas: 504536)
[PASS] test_shouldFail_whenIncorrectSecondStakeBeneficiary() (gas: 509546)
[PASS] test_shouldNotDeleteOriginalStakes() (gas: 514536)
[PASS] test_shouldUpdateStakesOfBeneficiary_whenOwnerIsBeneficiary() (gas: 517463)
Suite result: ok. 15 passed; 0 failed; 0 skipped; finished in 1.69s (2.11s CPU time)

Ran 17 tests for test/unit/token/VoteEscrowedPWN.Stake.t.sol:VoteEscrowedPWN_Stake_IncreaseStake_Test
[PASS] testFuzz_shouldClearRemainingTotalPowerChanges_whenNonZeroAdditionalEpochs(uint16,uint8,uint256,uint256) (runs:
↪  512, : 534657, ~: 498172)
[PASS] testFuzz_shouldEmit_StakePowerDelegated(address) (runs: 257, : 474887, ~: 474872)
[PASS] testFuzz_shouldFail_whenCallerNotStakeOwner(address) (runs: 257, : 103551, ~: 103551)
[PASS] testFuzz_shouldFail_whenIncorrectAdditionalAmount(uint256) (runs: 257, : 237587, ~: 237580)
[PASS] testFuzz_shouldFail_whenIncorrectdAdditionalEpochs(uint256) (runs: 257, : 262433, ~: 262436)
[PASS] testFuzz_shouldStoreNewStakeData(uint8,uint256,uint256) (runs: 257, : 532930, ~: 489480)
[PASS] testFuzz_shouldTransferAdditionalPWNTokens(uint256) (runs: 257, : 438383, ~: 438240)
[PASS] testFuzz_shouldUpdateStakesOfBeneficiary_whenOwnerIsNotBeneficiary(address) (runs: 257, : 478649, ~: 478633)
[PASS] testFuzz_shouldUpdateTotalPowerChanges(uint16,uint8,uint256,uint256) (runs: 513, : 1779307, ~: 1781077)
[PASS] test_shouldBurnOldStakedPWNToken() (gas: 430875)
[PASS] test_shouldEmit_StakeIncreased() (gas: 435941)
[PASS] test_shouldFail_whenIncorrectStakeBeneficiary() (gas: 298100)
[PASS] test_shouldFail_whenNothingToIncrease() (gas: 223862)
```

```
[PASS] test_shouldMintNewStakedPWNToken() (gas: 431751)
[PASS] test_shouldNotDeleteStakeData() (gas: 431607)
[PASS] test_shouldReturnNewStakedPWNTokenIds() (gas: 430813)
[PASS] test_shouldUpdateStakesOfBeneficiary_whenOwnerIsBeneficiary() (gas: 435435)
Suite result: ok. 17 passed; 0 failed; 0 skipped; finished in 1.77s (2.01s CPU time)

Ran 12 tests for test/unit/token/VoteEscrowedPWN.Stake.t.sol:VoteEscrowedPWN_Stake_SplitStake_Test
[PASS] testFuzz_shouldEmit_StakePowerDelegated(address) (runs: 256, : 369440, ~: 369423)
[PASS] testFuzz_shouldFail_whenCallerNotStakeOwner(address) (runs: 256, : 24089, ~: 24089)
[PASS] testFuzz_shouldFail_whenInvalidSplitAmount(uint256) (runs: 256, : 37712, ~: 38029)
[PASS] testFuzz_shouldUpdateStakesOfBeneficiary_whenOwnerIsNotBeneficiary(address) (runs: 256, : 370972, ~: 370957)
[PASS] test_shouldBurnOriginalStakedPWNToken() (gas: 193785)
[PASS] test_shouldEmit_StakeSplit() (gas: 196138)
[PASS] test_shouldFail_whenIncorrectStakeBeneficiary() (gas: 32756)
[PASS] test_shouldMintNewStakedPWNTokens() (gas: 196970)
[PASS] test_shouldNotDeleteOriginalStake() (gas: 194579)
[PASS] test_shouldReturnNewStakedPWNTokenIds() (gas: 192609)
[PASS] test_shouldStoreNewStakes(uint256) (runs: 256, : 200289, ~: 199988)
[PASS] test_shouldUpdateStakesOfBeneficiary_whenOwnerIsBeneficiary() (gas: 198132)
Suite result: ok. 12 passed; 0 failed; 0 skipped; finished in 1.84s (182.41ms CPU time)

Ran 9 tests for test/unit/governance/DAOExecuteAllowlist.t.sol:DAOExecuteAllowlist_IsGranted_Test
[PASS] testFuzz_shouldReturnTrue_whenDataLengthIsLessThan4(bytes) (runs: 256, : 16198, ~: 16214)
[PASS] testFuzz_shouldReturnTrue_whenPermissionIsNotExecute(bytes32) (runs: 256, : 15492, ~: 15492)
[PASS] testFuzz_shouldReturnTrue_whenSelectorIsNotExecute(bytes4) (runs: 256, : 16303, ~: 16303)
[PASS] testFuzz_shouldReturnTrue_whenWhereIsNotDAO(address) (runs: 256, : 15547, ~: 15547)
[PASS] test_shouldReturnFalse_whenAnyActionNotAllowed() (gas: 217111)
[PASS] test_shouldReturnFalse_whenSelectorNotAllowed() (gas: 97540)
[PASS] test_shouldReturnTrue_whenAllActionsAllowed() (gas: 234638)
[PASS] test_shouldReturnTrue_whenAnySelectorAllowed() (gas: 115070)
[PASS] test_shouldReturnTrue_whenSelectorAllowed() (gas: 117430)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 2.01s (187.42ms CPU time)

Ran 6 tests for test/unit/token/VoteEscrowedPWN.Power.t.sol:VoteEscrowedPWN_Power_StakerPowerAt_Test
[PASS] testFuzz_shouldFail_whenEpochTooBig(uint256) (runs: 256, : 16852, ~: 16587)
[PASS] testFuzz_shouldReturnCurrentStakePower(uint256,uint256) (runs: 512, : 322933, ~: 352149)
[PASS] testFuzz_shouldReturnSumOfAllStakesPower(uint256,uint256) (runs: 513, : 6218532, ~: 6188104)
[PASS] testFuzz_shouldReturnZero_whenEpochAfterExpiration(uint256) (runs: 257, : 184556, ~: 184738)
[PASS] testFuzz_shouldReturnZero_whenEpochBeforeInitialEpoch(uint256) (runs: 257, : 182825, ~: 182874)
[PASS] test_shouldReturnZero_whenEpochIsZero() (gas: 12671)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 2.00s (2.06s CPU time)

Ran 3 tests for test/integration/Integration.t.sol:Integration_vePWN_PowerTransfer_Test
[PASS] testFuzz_transferStake_whenAfterFinalEpoch(uint256,uint256) (runs: 256, : 654252, ~: 643033)
[PASS] testFuzz_transferStake_whenInInitialEpoch(uint256,uint256) (runs: 256, : 494036, ~: 503306)
[PASS] testFuzz_transferStake_whenNotInInitialEpoch(uint256,uint256) (runs: 512, : 560120, ~: 553483)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 2.31s (444.01ms CPU time)

Ran 86 test suites in 2.31s (22.10s CPU time): 370 tests passed, 2 failed, 0 skipped (372 total tests)

Failing tests:
Encountered 1 failing test in test/fork/PWN.governance.fork.t.sol:PWNGovernance_ForkTest
[FAIL: vm.envAddress: environment variable "OSX_PLUGIN_REPO_FACTORY" not found] setUp() (gas: 0)

Encountered 1 failing test in test/unit/token/StakedPWN.t.sol:StakedPWN_Transfer_Test
[FAIL: Error != expected error: ERC721: invalid token ID != TransfersDisabled(); counterexample:
 ↪  calldata=0x436f798100000000000000000000000000000000000000000000000000000000000000000000
 ↪  args=[0x0000000000000000000000000000000000000000]]
 ↪  test_shouldFail_whenTransfersNotEnabled_whenAddrNotInAllowlist(address) (runs: 105, : 18334, ~: 18334)
Encountered a total of 2 failing tests, 370 tests succeeded
```

8.2.1 Slither

All the relevant issues raised by Slither have been incorporated into the issues described in this report.

8.2.2 AuditAgent

All the relevant issues raised by the AuditAgent have been incorporated into this report. The AuditAgent is an AI-powered smart contract auditing tool that analyses code, detects vulnerabilities, and provides actionable fixes. It accelerates the security analysis process, complementing human expertise with advanced AI models to deliver efficient and comprehensive smart contract audits. Available at https://app.auditagent.nethermind.io.

9 About Nethermind

Nethermind is a Blockchain Research and Software Engineering company. Our work touches every part of the web3 ecosystem - from layer 1 and layer 2 engineering, cryptography research, and security to application-layer protocol development. We offer strategic support to our institutional and enterprise partners across the blockchain, digital assets, and DeFi sectors, guiding them through all stages of the research and development process, from initial concepts to successful implementation.

We offer security audits of projects built on EVM-compatible chains and Starknet. We are active builders of the Starknet ecosystem, delivering a node implementation, a block explorer, a Solidity-to-Cairo transpiler, and formal verification tooling. Nethermind also provides strategic support to our institutional and enterprise partners in blockchain, digital assets, and decentralized finance (DeFi). In the next paragraphs, we introduce the company in more detail.

Blockchain Security: At Nethermind, we believe security is vital to the health and longevity of the entire Web3 ecosystem. We provide security services related to Smart Contract Audits, Formal Verification, and Real-Time Monitoring. Our Security Team comprises blockchain security experts in each field, often collaborating to produce comprehensive and robust security solutions. The team has a strong academic background, can apply state-of-the-art techniques, and is experienced in analyzing cutting-edge Solidity and Cairo smart contracts, such as ArgentX and StarkGate (the bridge connecting Ethereum and StarkNet). Most team members hold a Ph.D. degree and actively participate in the research community, accounting for 240+ articles published and 1,450+ citations in Google Scholar. The security team adopts customer-oriented and interactive processes where clients are involved in all stages of the work.

Blockchain Core Development: Our core engineering team, consisting of over 20 developers, maintains, improves, and upgrades our flagship product - the Nethermind Ethereum Execution Client. The client has been successfully operating for several years, supporting both the Ethereum Mainnet and its testnets, and now accounts for nearly a quarter of all synced Mainnet nodes. Our unwavering commitment to Ethereum's growth and stability extends to sidechains and layer 2 solutions. Notably, we were the sole execution layer client to facilitate Gnosis Chain's Merge, transitioning from Aura to Proof of Stake (PoS), and we are actively developing a full-node client to bolster Starknet's decentralization efforts. Our core team equips partners with tools for seamless node set-up, using generated docker-compose scripts tailored to their chosen execution client and preferred configurations for various network types.

DevOps and Infrastructure Management: Our infrastructure team ensures our partners' systems operate securely, reliably, and efficiently. We provide infrastructure design, deployment, monitoring, maintenance, and troubleshooting support, allowing you to focus on your core business operations. Boasting extensive expertise in Blockchain as a Service, private blockchain implementations, and node management, our infrastructure and DevOps engineers are proficient with major cloud solution providers and can host applications in-house or on clients' premises. Our global in-house SRE teams offer 24/7 monitoring and alerts for both infrastructure and application levels. We manage over 5,000 public and private validators and maintain nodes on major public blockchains such as Polygon, Gnosis, Solana, Cosmos, Near, Avalanche, Polkadot, Aptos, and StarkWare L2. Sedge is an open-source tool developed by our infrastructure experts, designed to simplify the complex process of setting up a proof-of-stake (PoS) network or chain validator. Sedge generates docker-compose scripts for the entire validator set-up based on the chosen client, making the process easier and quicker while following best practices to avoid downtime and being slashed.

Cryptography Research: At Nethermind, our Cryptography Research team is dedicated to continuous internal research while fostering close collaboration with external partners. The team has expertise across a wide range of domains, including cryptography protocols, consensus design, decentralized identity, verifiable credentials, Sybil resistance, oracles, and credentials, distributed validator technology (DVT), and Zero-knowledge proofs. This diverse skill set, combined with strong collaboration between our engineering teams, enables us to deliver cutting-edge solutions to our partners and clients.

Smart Contract Development & DeFi Research: Our smart contract development and DeFi research team comprises 40+ world-class engineers who collaborate closely with partners to identify needs and work on value-adding projects. The team specializes in Solidity and Cairo development, architecture design, and DeFi solutions, including DEXs, AMMs, structured products, derivatives, and money market protocols, as well as ERC20, 721, and 1155 token design. Our research and data analytics focuses on three key areas: technical due diligence, market research, and DeFi research. Utilizing a data-driven approach, we offer in-depth insights and outlooks on various industry themes.

Our suite of L2 tooling: Warp is Starknet's approach to EVM compatibility. It allows developers to take their Solidity smart contracts and transpile them to Cairo, Starknet's smart contract language. In the short time since its inception, the project has accomplished many achievements, including successfully transpiling Uniswap v3 onto Starknet using Warp.

- **Voyager** is a user-friendly Starknet block explorer that offers comprehensive insights into the Starknet network. With its intuitive interface and powerful features, Voyager allows users to easily search for and examine transactions, addresses, and contract details. As an essential tool for navigating the Starknet ecosystem, Voyager is the go-to solution for users seeking in-depth information and analysis;
- **Horus** is an open-source formal verification tool for StarkNet smart contracts. It simplifies the process of formally verifying Starknet smart contracts, allowing developers to express various assertions about the behavior of their code using a simple assertion language;
- **Juno** is a full-node client implementation for Starknet, drawing on the expertise gained from developing the Nethermind Client. Written in Golang and open-sourced from the outset, Juno verifies the validity of the data received from Starknet by comparing it to proofs retrieved from Ethereum, thus maintaining the integrity and security of the entire ecosystem.

Learn more about us at nethermind.io.

General Advisory to Clients

As auditors, we recommend that any changes or updates made to the audited codebase undergo a re-audit or security review to address potential vulnerabilities or risks introduced by the modifications. By conducting a re-audit or security review of the modified codebase, you can significantly enhance the overall security of your system and reduce the likelihood of exploitation. However, we do not possess the authority or right to impose obligations or restrictions on our clients regarding codebase updates, modifications, or subsequent audits. Accordingly, the decision to seek a re-audit or security review lies solely with you.

Disclaimer

This report is based on the scope of materials and documentation provided by you to Nethermind in order that Nethermind could conduct the security review outlined in **1. Executive Summary** and **2. Audited Files**. The results set out in this report may not be complete nor inclusive of all vulnerabilities. Nethermind has provided the review and this report on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. This report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on this report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, Nethermind disclaims any liability in connection with this report, its content, and any related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. Nethermind does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and Nethermind will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.