Quantstamp

Sapien - 2

Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

Type	Token Staking
Timeline	2025-06-02 through 2025-06-06
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Source Code	• Sapien-io/sapien-contracts #baa19f7
Auditors	• Paul Clemson Auditing Engineer • Julio Aguilar Auditing Engineer • Tim Sigl Auditing Engineer

Documentation quality	Medium
Test quality	Medium
Total Findings	10 Fixed: 10
High severity findings	4 Fixed: 4
Medium severity findings	2 Fixed: 2
Low severity findings	3 Fixed: 3
Undetermined severity findings	0
Informational findings	1 Fixed: 1

Summary of Findings

The Sapien protocol intends to reward users to participating in AI training activities. The smart contracts of the protocol focus a few different areas. Firstly `SapienVault` allows users to stake and unstake Sapien tokens into a vault to earn a multiplier which will be used offchain to open up more training tasks to the users. The `SapienQA` contract allows the protocol to issue warnings or penalties to misbehaving users in the system. On top of this the `SapienRewards` contract handles allowing users to claim rewards using a signature from the protocol. The protocol's code was well documented and had a solid test suite, however the audit uncovered a number of issues with the core business logic and design of the protocol that should be thought about carefully and adequately resolved before the protocol is launched.

Fix-Review Update 2025-07-01:
During the fix review, the client fixed all of the issues highlighted during the initial audit. However in the process of rectifying these issues and improving the overall design of the system, some additional issues were raised by the audit team and then fixed by the client. However due to this, the codebase has gone through significant changes since the original audit commit hash, so we recommend that the protocol goes through an additional audit before launch to ensure the complete business logic of the protocol is sound and no additional vulnerabilities have arisen.

ID	DESCRIPTION	SEVERITY	STATUS
SAP-1	Cooldown Period Can Be Bypassed Allowing Instant Unstaking	• High	Fixed
SAP-2	Lockup Period for Unstaking Can Be Decreased by Staking Again with Shorter Lockup Period	• High	Fixed
SAP-3	QA Penalty Can Be Avoided by Unstaking Before Penalty Processing	• High	Fixed
SAP-4	`processQAPenalty()` Function Incorrectly Double Counts Tokens in Cooldown	• High	Fixed
SAP-5	Missing Expiration Check when Adding to Existing Stake Allows Timelock Bypass	• Medium	Fixed

ID	DESCRIPTION	SEVERITY	STATUS
SAP-6	Inconsistent Lockup Period Calculation Allows Users to Game the System	● Medium	Fixed
SAP-7	Contracts Could End up without an Admin if Deployer Is Also the Admin	● Low	Fixed
SAP-8	Authorized `emergencyWithdraw()` Function Is Vulnerable to Reentrancy Attack	● Low	Fixed
SAP-9	EIP-712 `QA_DECISION_TYPEHASH` Declares `string` but Encodes `bytes32`	● Low	Fixed
SAP-10	Missing `_disableInitializers()` in `SapienVault`	● Informational	Fixed

Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Disclaimer

Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
   1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Scope

Files Included

Repo: https://github.com/Sapien-io/sapien-contracts(e3905c7) Files: src/*
Repo: `https://github.com/Sapien-io/sapien-contracts`

Included Paths: `src/`

Files Excluded

Repo: https://github.com/Sapien-io/sapien-contracts(e3905c7) Files: none

Operational Considerations

- Admin roles should use a multi-signature wallet to reduce the chances of potential misuse.
- The `treasury` address that receives the total supply of Sapien tokens is trusted to handle these correctly.
- Rewards signatures must be handled well to ensure users get the correct amounts.
- Reward tokens must be added to the `SapienRewards` contract when necessary.
- The QA penalty system must process all outstanding penalties within 48 hours to ensure users cannot avoid penalties by unstaking.
- Before setting a new reward token in `SapienRewards`, the admin has to withdraw all tokens of the current reward token. The admin functions only allow to withdraw the current reward token.
- The token in `SapienRewards` is expected to be a standard ERC-20 token. The protocol does not support tokens with fees on transfers or rebase mechanisms.
- The SapienRewards and SapienVault contracts are upgradeable, which allows for protocol improvements and bug fixes without requiring users to migrate funds to new contracts, enabling the team to respond to changing market conditions and security concerns. However, this flexibility comes with the risk of upgrading to buggy code if not properly tested or an admin compromise could lead to malicious upgrades that endanger user funds.
- The SapienVault contract provides the emergencyWithdraw() function, which offers a critical safety mechanism to protect funds in extreme situations. The risk lies in the case of the admin keys getting compromised.
- Off-chain reward distribution enables flexible rewards based on complex metrics that would be expensive to compute on-chain. The system relies on trusted reward managers who can sign arbitrary reward amounts. Implementing more transparent criteria and on-chain verification would improve trust in the fairness of reward distribution.

Key Actors And Their Capabilities

The protocol has the following authorized roles:

SapienVault

Admin: Can set the addresses of the `treasury` and `multiplier` contracts, can also call `emergencyWithdraw()`. The admin is also given the Pauser role.

Pauser: Is able to pause an unpause the contract.

SapienQA: Can call the `processQAPenalty()` function to issue penalties to stakers. It is expected that this role will be the `SapienQA` smart contract rather than an individual EOA.

SapienQA

Admin: Can set the addresses of the `treasury` and `vaultContract`.

QA Manager: Can call the `processQualityAssessment()` function to issue penalties or warnings to users.

QA Admin: Must be the signer of assessments processed via the `processQualityAssessment()` function.

SapeinRewards

Admin: Can set the address of the `rewardToken`. The admin is also given the Pauser role.

Pauser: Is able to pause an unpause the contract.

Reward Manager: Is responsible for signing the contents of a users reward claims. They can also call the `validateAndGetHashToSign()` function to create signatures.

Reward Admin: Is responsible for adding deposit tokens to the contract via `depositRewards()` or withdrawing tokens via one of the `withdrawRewards()` / `recoverUnaccountedRewards()` functions. They can also call `reconcileBalance()` to match the total claimable rewards amount to the contract reward token balance.

Findings

SAP-1
Cooldown Period Can Be Bypassed Allowing Instant Unstaking

● High ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `3d82c229626f11633838f9aa94265d497a838de9`.

File(s) affected: `SapienVault.sol`

Description: The `initiateUnstake()` function in the `SapienVault` contract sets the cooldown start timestamp only if the user is not already in a cooldown state, as controlled by the condition `if (uint256(userStake.cooldownStart) == 0)`. This design creates an opportunity to bypass the intended cooldown period for the majority of a user's stake. Specifically, a user may initiate unstaking for a small portion of their stake, wait for the cooldown to complete, and then unstake the remaining amount without waiting again, effectively circumventing the cooldown for most of their tokens.

Exploit Scenario:
1. A user stakes 1,000 tokens with a cooldown period of 7 days.
2. After the lockup period expires, the user initiates unstaking for only 1 token, which sets `cooldownStart` to the current block timestamp.
3. The user retains the remaining 999 tokens in active staking, continuing to receive associated benefits.
4. Once the 2-day cooldown completes for the 1 token, the user initiates unstaking for the remaining 999 tokens.
5. Since `cooldownStart` is already set and not updated, the cooldown is considered complete, allowing the user to immediately unstake all remaining tokens.

Recommendation: To enforce the cooldown consistently, consider one of the following mitigations:
- **Update `cooldownStart` on each call to `initiateUnstake()`**, regardless of its current value. This ensures the cooldown period is enforced for each unstake request. Be aware that this approach may unintentionally reset the cooldown timer if users initiate small unstake amounts frequently.
- **Restrict unstaking to the full available amount** at once, thereby preventing partial cooldown circumvention.
- **Implement per-amount cooldown tracking**, associating each unstaked amount with its own cooldown start. This solution is the most accurate but requires structural changes to how stake and cooldowns are stored and managed.

SAP-2
Lockup Period for Unstaking Can Be Decreased by Staking Again with Shorter Lockup Period

● High Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `c926eed4e5d707a63f34736158bd79873b8e0411` and `342795073d5d8abf2c3272f861cbbf63a4974774`.

File(s) affected: `SapienVault.sol`

Description: The `SapienVault` contract recalculates the effective lockup period when a user adds to an existing stake using the `stake()` function. The recalculation uses a **weighted average** between the current effective lockup period and the newly provided lockup period. This mechanism allows users to **reduce** their overall lockup duration by staking additional tokens with a shorter lockup period.

As a result, users can strategically lower their lockup commitment after the initial stake, undermining the intended locking mechanism.

Exploit Scenario:
1. A user initially stakes 1,000 tokens with a 365-day lockup period.
2. Later, the user stakes an additional 10,000 tokens with a 30-day lockup period.
3. The weighted average lockup period is recalculated, heavily influenced by the new larger amount and shorter duration.
4. The new effective lockup period becomes significantly lower than 365 days, allowing the user to unlock all tokens earlier than originally committed.
5. The restaking can be repeated with a 30-day lockup period, further reducing the effective lockup duration.

Recommendation: Disallow new stakes with shorter lockup periods than the existing commitment. Enforce that any additional stake must use a lockup period equal to or greater than the current one.

SAP-3
QA Penalty Can Be Avoided by Unstaking Before Penalty Processing

● High Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `737af6c63b0eae0af9c80425eb8a27afffd14ad8`, `cf36a2fac54e84904c51f8d419c04e215c7b35da` and `24e56c21d8cabbaf5b6236080265d629361e182e`.

File(s) affected: `SapienVault.sol`, `SapienQA.sol`

Description: Typical unstaking within the protocol requires a two day cooldown period to pass before unstaking can be completed. The intention of this is to allow a sufficient window for any QA penalties to be applied before the user can unstake their tokens. However, in the case of the `earlyUnstake()` function, users are able to instantly withdraw their funds by paying a 20 percent penalty. In cases where a user is expecting to be slashed a value greater than 20 percent they can call the `earlyUnstake()` function to immediately withdraw their tokens and avoid the impending larger QA penalty.

Recommendation: Consider enforcing a cooldown period on all unstaking actions to ensure that QA penalties cannot be avoided.

SAP-4

`processQAPenalty()` Function Incorrectly Double Counts Tokens in Cooldown

● High ⓘ Fixed

Update

Marked as "Fixed" by the client.

Addressed in: `56d63be892926ae9381098d467d0c235a1b660ad` .

File(s) affected: `SapienVault.sol`

Description: The `_calculateApplicablePenalty()` function incorrectly calculates available tokens for penalties by adding `userStake.amount + userStake.cooldownAmount` . However, `cooldownAmount` represents tokens that are already included in `amount` but are queued for unstaking. This double counting allows penalties to exceed the user's actual stake balance, potentially leading to accounting errors or failed penalty applications when the contract attempts to transfer more tokens than available.

Recommendation: Consider using only `userStake.amount` as the total available balance for penalties since `cooldownAmount` is a subset of this value, not an additional amount. In cases where the penalty amount is greater then `amount - cooldownAmount` the leftover tokens must be deducted from both `amount` and `cooldownAmount` to ensure the contract's internal accounting remains correct.

SAP-5

Missing Expiration Check when Adding to Existing Stake Allows Timelock Bypass

● Medium ⓘ Fixed

Update

Marked as "Fixed" by the client.

Addressed in: `ffda07756c53963c20a254c11c97f6809d08cfaf` .

File(s) affected: `SapienVault.sol`

Description: The `stake()` and `increaseAmount()` functions allow users to add tokens to stakes without validation that the initial stake's lockup period has not ended. This can allow for some stakes where the new `weightedStartTime + effectiveLockUpPeriod < block.timestamp` meaning the stake is immediately unlocked. This allows users to benefit from an increased multiplier while being able to unlock their tokens at any time.

Recommendation: Consider checking if a user's existing stake has expired before allowing them to add additional tokens via `stake()` or `increaseAmount()` functions. Alternatively, reset the weighted start time to the current timestamp when adding to expired stakes.

SAP-6

Inconsistent Lockup Period Calculation Allows Users to Game the System

● Medium ⓘ Fixed

Update

Marked as "Fixed" by the client.

Addressed in: `b4c59bc4e68926746413e842fdaed3e6cd386934` and `cf36a2fac54e84904c51f8d419c04e215c7b35da` .

File(s) affected: `SapienVault.sol`

Description: The `SapienVault` contract exhibits an inconsistency in the `effectiveLockupPeriod` calculation methodology. When users stake multiple times, the contract calculates a new weighted start time and a new weighted effective lockup period. Also when users increase their stake amount, the contract calculates a new weighted start time. However, the `increaseLockup()` function bypasses this weighted calculation for a new lockup period, and instead uses a direct addition of the remaining lockup time and the additional lockup period.

This inconsistency allows users to game the system by strategically choosing whether to add new stakes or increase lockup on existing stakes to get the shortest lockup period, as the mathematical approach differs between these operations.

Recommendation: Modify the `increaseLockup()` function to utilize the same weighted calculation approach as employed in the staking operation.

SAP-7 Contracts Could End up without an Admin if Deployer Is Also the Admin

• Low ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `7ce34d478bb85fc43161e1af537aeb65f6927000`.

File(s) affected: `SapienVault.sol`, `SapienRewards.sol`

Description: In both `SapienVault` and `SapienRewards`, the `initialize()` function grants the `DEFAULT_ADMIN_ROLE` to the specified `admin` address and then unconditionally revokes the same role from `msg.sender`. If the deployer and the admin are the **same address**, this logic leads to a scenario where **no account holds the** `DEFAULT_ADMIN_ROLE`, effectively rendering the contract **administratively locked** and unmanageable.

Recommendation: Update `initialize()` in both `SapienVault` and `SapienRewards` to match the logic used in `SapienQA`, ensuring that the admin role is only revoked from the deployer if it differs from the admin:

```
if (msg.sender != admin) {
    _revokeRole(DEFAULT_ADMIN_ROLE, msg.sender);
}
```

SAP-8 Authorized `emergencyWithdraw()` Function Is Vulnerable to Reentrancy Attack

• Low ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `7179ca51941cf5cbe03c1e3bab547db0ec347d73`.

File(s) affected: `SapienVault.sol`

Description: The `emergencyWithdraw()` function performs ether transfer via a `.call()` without reentrancy protection. While this function is access-controlled to admin only, reentrancy vulnerabilities in emergency functions can be particularly dangerous as they may be called during compromised states. An attacker who gains admin access could potentially exploit reentrancy to drain funds beyond intended amounts.

Recommendation: Consider applying the `nonReentrant` modifier to the `emergencyWithdraw()` function to prevent potential reentrancy attacks.

SAP-9 EIP-712 `QA_DECISION_TYPEHASH` Declares `string` but Encodes `bytes32`

• Low ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `b38e69ebe7171ec897cb65285703bce77c3e593d`.

File(s) affected: `SapienQA.sol`

Description: The `QA_DECISION_TYPEHASH` in `SapienQA` is defined as:

```
keccak256("QADecision(address userAddress,uint8 actionType,uint256 penaltyAmount,bytes32
decisionId,string reason)")
```

However, when encoding the struct for signature verification, the `reason` field is encoded as `keccak256(bytes(reason))`, which is a `bytes32`, not a `string`. This means the actual encoded struct does not match the declared type hash, potentially breaking EIP-712 signature compatibility and causing signature verification to fail or be non-standard.

Recommendation: Update the `QA_DECISION_TYPEHASH` to use `bytes32 reason` instead of `string reason` to match the actual encoding.

SAP-10 Missing `_disableInitializers()` in `SapienVault`

• Informational ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `6bc9fdf9cce877c7aea265a9fe56a2c0f4b45c05`.

File(s) affected: `SapienVault.sol`

Description: The `SapienVault` contract is missing a constructor to call `_disableInitializers()`. With the current implementation, although the access through a proxy is not exposed, the deployed contract can be initialized which could open up unexpected behavior.

Recommendation: We recommend adding a constructor that calls `_disableInitializers()`.

Auditor Suggestions

S1 Miscellaneous Improvements

Mitigated

Update
Marked as "Fixed" by the client.
Addressed in: `b97530a427b4b94b1076c1b3bcd1e6a4ae7da165`.

Update
The majority of these suggestions were either implemented or the code referred to has been removed or refactored.

File(s) affected: `*`

Description: The following is a list of potential suggestions to improve the codebase overall:

1. Make the `Muliplier` contract a library instead of a contract. This would allow the `SapienVault` to use the `Multiplier` functions without needing to deploy a separate contract, reducing gas costs and complexity.
2. Remove `isValidLockupPeriod()` function from `Multiplier` contract. The exact same function also exists on the `SapienVault` contract.
3. Add validation to `Mulitplier.interpolate()` function to ensure `x2 > x1`. Optionally, also add check if `y2 > y1` and `x1 <= x <= x2`.
4. Fix misleading `T1_FACTOR` - `T5_FACTOR` constants. They are not basis points but amount of tokens.
5. Remove unused errors from `IMultiplier` interface.
6. Use a different address than the `admin` address for the `PAUSER_ROLE` in the `SapienVault` contract. Roles only make sense if they are assigned to different addresses.
7. Remove parameters from `MultiplierUpdated` event, they are always `MultiplierUpdated(0, 0)`.
8. The function `SapienVault.stake()` checks if the `msg.sender` is `address(0)`. The sender cannot be `address(0)`.
9. Re-evaluate if the storage savings by using lower uint types in the `UserStake` struct are worth the additional complexity of casting and overflow checks. It adds a significant amount of code in this contract.
10. Consider using a constant for the magic number for the additional amount in the `_validateIncreaseAmount()` function.
11. Avoid double validation and validate user input in the beginning of external or public functions. For example, in `increaseAmount()` the amount is first validated in `_validateIncreaseAmount()` and again in `_calculateWeightedStartTime()`.
12. Streamline EIP-712 implementation in `SapienRewards` to also use OpenZeppelin's `EIP712` library as done in the `SapienQA` contract. This would reduce code duplication and potential inconsistencies in signature handling.
13. Reconsider if own accounting of reward tokens in the `SapienRewards` contract through the `availableRewards` variable is necessary. Using `balanceOf(address(this))` would remove a lot of business logic but would also remove the ability to distinguish between tokens that are transferred to the contract via the `depositRewards()` function and directly via the token's transfer function.
14. The functions `_validateStakeInputs()` and `_validateIncreaseAmount()` use the same hardcoded maximum stake value `(10_000_000 * Const.TOKEN_DECIMALS)`. For clarity and maintainability, consider defining a constant such as `MAXIMUM_STAKE_AMOUNT`.
15. The `calculateMultiplier()` function returns zero if the lockup period is outside the valid range or if the staked amount is too low. Since these cases should already be prevented, consider reverting the transaction instead to avoid silent failures.
16. In `_calculateWeightedValues()`, the checks `if (amount == 0)`, `if (newTotalAmount == 0),` and the lockup period bounds check are redundant, as these are already validated upstream. Removing them can slightly reduce gas usage.
17. The functions `_calculateWeightedStartTimeValue()` and `_calculateWeightedLockupPeriod()` contain overflow checks that are no longer necessary in Solidity >= 0.8.0. Consider removing these checks to reduce gas costs.
18. The `_verifySignature()` function checks for replay using a decision ID but does not enforce expiration. It is best practice to include a timestamp or expiry field in signed messages to prevent indefinite reuse.
19. The `_resetUserStake()` function manually sets each struct field to zero, which could be replaced with `delete userStakes[msg.sender]`
20. The `getUserStakingSummary()` function could check `userStake.hasStake` and revert if false as all the values will be zero.
21. The `QAPenaltyPartial` event is emitted by both `SapienQA` and `SapienVault`, one of these can be removed to avoid this redundancy.

22. The `SapienVault` contract uses `SafeCast` correctly for downcasting values, however it unnecessarily upcasts values to uint256, which is unnecessary in Solidity `>= 0.8.0`. These upcasts can be removed for clarity.
23. The if `(penalty > 0)` condition in `earlyUnstake()` is unnecessary because both the amount and penalty percentage are non-zero. Consider removing the check for a minor gas improvement.
24. Precision loss when calculating penalties in `earlyUnstake()` can lead to zero penalty amounts with very small amounts of tokens. Consider implementing some reasonable minimum unstake amount to eliminate the chance of this.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Files

Repo: `https://github.com/Sapien-io/sapien-contracts`

- `cef...482 ./src/Multiplier.sol`
- `740...e87 ./src/SapienQA.sol`
- `8bd...30a ./src/SapienRewards.sol`
- `254...79d ./src/SapienToken.sol`
- `503...4bf ./src/SapienVault.sol`
- `711...f78 ./src/interfaces/IMultiplier.sol`
- `60f...f6b ./src/interfaces/ISapienQA.sol`
- `951...340 ./src/interfaces/ISapienRewards.sol`
- `b1b...8b7 ./src/interfaces/ISapienToken.sol`
- `ed4...2bc ./src/interfaces/ISapienVault.sol`
- `e55...bf7 ./src/utils/Common.sol`
- `8d8...184 ./src/utils/Constants.sol`
- `8b2...8a7 ./src/utils/SafeCast.sol`

Test Suite Results

The test suite can be run by issuing the following commands

```
forge install
forge test
```

```
Ran 8 tests for test/unit/SapienVault_CooldownBugTest.t.sol:SapienVaultCooldownBugTest
[PASS] test_Vault_CooldownLogic_CannotIncreaseAmountDuringCooldown() (gas: 211340)
[PASS] test_Vault_CooldownLogic_CannotIncreaseLockupDuringCooldown() (gas: 186671)
[PASS] test_Vault_CooldownLogic_CooldownReadyAfterPeriod() (gas: 241665)
[PASS] test_Vault_CooldownLogic_InstantUnstakeExcludesStakesInCooldown() (gas: 208832)
[PASS] test_Vault_CooldownLogic_InstantUnstakeWorksOnLockedStake() (gas: 246791)
[PASS] test_Vault_CooldownLogic_MultiplePartialCooldowns() (gas: 230000)
[PASS] test_Vault_CooldownLogic_PartialUnstakeCorrectlyTracked() (gas: 216993)
[PASS] test_Vault_CooldownLogic_PartialUnstakeFromCooldown() (gas: 246395)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 2.65ms (1.94ms CPU time)

Ran 8 tests for test/unit/SapienToken_Integration.t.sol:SapienTokenIntegrationTest
[PASS] test_Token_BatchTransfers_GasEfficiency() (gas: 172942)
[PASS] test_Token_ManySmallTransfers() (gas: 2949650)
[PASS] test_Token_MaxApproval() (gas: 42353)
[PASS] test_Token_Permit_IntegrationWithTransfers() (gas: 108824)
[PASS] test_Token_SelfTransfer() (gas: 22197)
[PASS] test_Token_TokenDistribution_Simulation() (gas: 203881)
[PASS] test_Token_TotalSupplyInvariant() (gas: 109105)
[PASS] test_Token_ZeroValueTransfers() (gas: 40883)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 3.39ms (2.61ms CPU time)

Ran 6 tests for test/unit/SapienVault_Scenarios.t.sol:SapienVaultScenariosTest
[PASS] test_Vault_Scenario_EmergencyLiquidator() (gas: 241296)
[PASS] test_Vault_Scenario_LockupExtensionBenefits() (gas: 284695)
[PASS] test_Vault_Scenario_MultiUserInteraction() (gas: 431047)
[PASS] test_Vault_Scenario_ProgressiveStaker() (gas: 312691)
[PASS] test_Vault_Scenario_StrategicRebalancer() (gas: 338143)
[PASS] test_Vault_Scenario_WeightedAverageStaking() (gas: 290826)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 6.78ms (2.95ms CPU time)

Ran 5 tests for test/unit/SapienRewards_Scenarios.t.sol:SapienRewardsScenariosTest
[PASS] test_Rewards_Scenario_CompleteRewardsLifecycle() (gas: 518917)
[PASS] test_Rewards_Scenario_EmergencyRecoveryWorkflow() (gas: 309218)
[PASS] test_Rewards_Scenario_HighVolumeOperations() (gas: 1714567)
[PASS] test_Rewards_Scenario_MultiManagerCoordination() (gas: 423932)
[PASS] test_Rewards_Scenario_StressTestConcurrentOperations() (gas: 2276307)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 10.40ms (15.60ms CPU time)

Ran 12 tests for test/unit/SapienToken.t.sol:SapienTokenTest
[PASS] test_Token_Approve_Success() (gas: 45386)
[PASS] test_Token_Constructor_RevertZeroAddress() (gas: 88789)
[PASS] test_Token_Constructor_Success() (gas: 32441)
[PASS] test_Token_Fuzz_Approve(uint256) (runs: 256, μ: 42301, ~: 42612)
[PASS] test_Token_Fuzz_Transfer(uint256) (runs: 256, μ: 52981, ~: 52788)
[PASS] test_Token_MaxSupply_Immutable() (gas: 12728)
[PASS] test_Token_Permit_ExpiredDeadline() (gas: 21468)
[PASS] test_Token_Permit_Success() (gas: 110643)
[PASS] test_Token_TransferFrom_InsufficientAllowance() (gas: 20755)
[PASS] test_Token_TransferFrom_Success() (gas: 66386)
[PASS] test_Token_Transfer_InsufficientBalance() (gas: 18202)
[PASS] test_Token_Transfer_Success() (gas: 52640)
Suite result: ok. 12 passed; 0 failed; 0 skipped; finished in 17.90ms (22.99ms CPU time)

Ran 16 tests for test/unit/Multiplier.t.sol:MultiplierTest
[PASS] testFuzz_Multiplier_CalculateMultiplier_InvalidInputs(uint256,uint256) (runs: 256, μ: 11952, ~:
13189)
[PASS] testFuzz_Multiplier_CalculateMultiplier_ValidInputs(uint256,uint256) (runs: 256, μ: 22205, ~:
22392)
[PASS] test_Multiplier_CalculateMultiplier_AmountTiers() (gas: 32346)
[PASS] test_Multiplier_CalculateMultiplier_BoundaryAmounts() (gas: 49165)
[PASS] test_Multiplier_CalculateMultiplier_ConsistentWithMatrix() (gas: 172218)
[PASS] test_Multiplier_CalculateMultiplier_ExactDiscretePeriods() (gas: 22862)
[PASS] test_Multiplier_CalculateMultiplier_InvalidAmounts() (gas: 13138)
[PASS] test_Multiplier_CalculateMultiplier_InvalidLockupPeriods() (gas: 15030)
[PASS] test_Multiplier_CalculateMultiplier_LargeAmounts() (gas: 15669)
[PASS] test_Multiplier_CalculateMultiplier_Matrix() (gas: 81690)
[PASS] test_Multiplier_CalculateMultiplier_MonotonicIncrease() (gas: 30562)
[PASS] test_Multiplier_DurationMultiplier_InterpolationBetween180And365Days() (gas: 13514)
[PASS] test_Multiplier_DurationMultiplier_InterpolationBetween30And90Days() (gas: 17989)
```

```
[PASS] test_Multiplier_DurationMultiplier_InterpolationBetween90And180Days() (gas: 12797)
[PASS] test_Multiplier_GetAmountTierFactor_AllTiers() (gas: 32900)
[PASS] test_Multiplier_IsValidLockupPeriod() (gas: 23505)
Suite result: ok. 16 passed; 0 failed; 0 skipped; finished in 21.64ms (24.96ms CPU time)

Ran 14 tests for test/unit/SapienVault_WeightedCalculations.t.sol:SapienVaultWeightedCalculationsTest
[PASS] test_Vault_Consistency_OrderIndependence() (gas: 389181)
[PASS] test_Vault_CustomError_AmountMustBePositive() (gas: 36457)
[PASS] test_Vault_CustomError_InvalidLockupPeriod() (gas: 77575)
[PASS] test_Vault_CustomError_LockupWeightCalculationOverflow() (gas: 3705)
[PASS] test_Vault_CustomError_WeightedCalculationOverflow() (gas: 3673)
[PASS] test_Vault_EdgeCase_MultipleSmallStakes() (gas: 337944)
[PASS] test_Vault_EdgeCase_VeryLargeAmounts() (gas: 223514)
[PASS] test_Vault_GasEfficiency_CustomErrorsVsRequire() (gas: 171490)
[PASS] test_Vault_Precision_RoundingUp() (gas: 219105)
[PASS] test_Vault_WeightedLockup_DifferentAmounts() (gas: 216965)
[PASS] test_Vault_WeightedLockup_EqualAmounts() (gas: 215735)
[PASS] test_Vault_WeightedLockup_MaximumCapping() (gas: 184824)
[PASS] test_Vault_WeightedStartTime_DifferentAmounts() (gas: 216038)
[PASS] test_Vault_WeightedStartTime_EqualAmounts() (gas: 215723)
Suite result: ok. 14 passed; 0 failed; 0 skipped; finished in 21.38ms (5.66ms CPU time)

Ran 4 tests for test/unit/SapienVault_EndToEnd.t.sol:SapienVaultEndToEndTest
[PASS] test_EndToEnd_CompleteStakingJourney() (gas: 1677686)
[PASS] test_StakingPattern_EarlyExitOptimizer() (gas: 288650)
[PASS] test_StakingPattern_LiquidityManager() (gas: 291018)
[PASS] test_StakingPattern_ProgressiveBuilder() (gas: 312629)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 21.48ms (3.69ms CPU time)

Ran 26 tests for test/unit/SapienQA.t.sol:SapienQATest
[PASS] test_QA_AccessControl_AdminFunctions() (gas: 58845)
[PASS] test_QA_AccessControl_AdminFunctions_ZeroAddressValidation() (gas: 18574)
[PASS] test_QA_AccessControl_AdminTransferAndRoleRevocation() (gas: 84241)
[PASS] test_QA_AccessControl_MultipleQAManagers() (gas: 535522)
[PASS] test_QA_AccessControl_ProcessQualityAssessment_UnauthorizedCaller() (gas: 213141)
[PASS] test_QA_AccessControl_ProcessQualityAssessment_ValidQAManager() (gas: 487366)
[PASS] test_QA_AccessControl_QAManagerRole() (gas: 34311)
[PASS] test_QA_AccessControl_RoleManagement() (gas: 104904)
[PASS] test_QA_AccessControl_SignatureValidation_InvalidSignatureLength() (gas: 180433)
[PASS] test_QA_AccessControl_SignatureValidation_UnauthorizedSigner() (gas: 193146)
[PASS] test_QA_AccessControl_ViewFunctionsPublicAccess() (gas: 99825)
[PASS] test_QA_BasicFunctionality() (gas: 34114)
[PASS] test_QA_ConstructorValidation() (gas: 131109)
[PASS] test_QA_EndToEndScenario() (gas: 1004174)
[PASS] test_QA_GetUserQARecordCount() (gas: 594821)
[PASS] test_QA_PenaltyProcessingFailure() (gas: 425779)
[PASS] test_QA_PenaltyProcessingNoStake() (gas: 272707)
[PASS] test_QA_PenaltyProcessingStringError() (gas: 411187)
[PASS] test_QA_PenaltyProcessingVaultPausedError() (gas: 432523)
[PASS] test_QA_ProcessQualityAssessmentInvalidSignature() (gas: 27009)
[PASS] test_QA_ProcessQualityAssessmentMinorPenalty() (gas: 506434)
[PASS] test_QA_ProcessQualityAssessmentPenalty() (gas: 506543)
[PASS] test_QA_ProcessQualityAssessmentValidationErrors() (gas: 63297)
[PASS] test_QA_ProcessQualityAssessmentWarning() (gas: 422875)
[PASS] test_QA_ReplayAttackPrevention() (gas: 244543)
[PASS] test_QA_VaultIntegration() (gas: 223940)
Suite result: ok. 26 passed; 0 failed; 0 skipped; finished in 21.73ms (10.15ms CPU time)

Ran 91 tests for test/unit/SapienVault.t.sol:SapienVaultBasicTest
[PASS] test_Vault_CalculateMultiplier() (gas: 43174)
[PASS] test_Vault_CompleteUnstakingFlow() (gas: 199884)
[PASS] test_Vault_ComprehensiveEdgeCases() (gas: 249018)
[PASS] test_Vault_DustAttackPrevention() (gas: 177628)
[PASS] test_Vault_EarlyWithdrawal_PenaltyValidation() (gas: 200726)
[PASS] test_Vault_EdgeCase_JustUnderTenMillionLimit() (gas: 181815)
[PASS] test_Vault_EdgeCase_MaximumValidStakeAmount() (gas: 181881)
[PASS] test_Vault_EmergencyWithdrawERC20() (gas: 125504)
[PASS] test_Vault_EmergencyWithdrawETH() (gas: 92309)
[PASS] test_Vault_EmergencyWithdrawFullERC20Balance() (gas: 101690)
[PASS] test_Vault_EmergencyWithdrawFullETHBalance() (gas: 91485)
[PASS] test_Vault_EmergencyWithdrawScenario() (gas: 1308452)
[PASS] test_Vault_EmergencyWithdrawWithETHAndERC20() (gas: 167492)
```

```
[PASS] test_Vault_GetTotalStaked() (gas: 239269)
[PASS] test_Vault_GetUserStakingSummary() (gas: 251750)
[PASS] test_Vault_GrantPauserRoleToOther() (gas: 58239)
[PASS] test_Vault_IncreaseAmount() (gas: 216173)
[PASS] test_Vault_IncreaseLockup() (gas: 202689)
[PASS] test_Vault_IncreaseLockupCapAt365Days() (gas: 196785)
[PASS] test_Vault_Initialization() (gas: 38470)
[PASS] test_Vault_InitiateUnstake_CooldownAmountOverflow_Theoretical() (gas: 201600)
[PASS] test_Vault_InitiateUnstake_MultipleCallsAccumulateCooldown() (gas: 225569)
[PASS] test_Vault_InstantUnstake() (gas: 203928)
[PASS] test_Vault_InstantUnstakePartial() (gas: 245552)
[PASS] test_Vault_InterpolatedMultipliers() (gas: 214313)
[PASS] test_Vault_LockupPeriodCap() (gas: 219701)
[PASS] test_Vault_MultiplierContract_InvalidLockupPeriod() (gas: 39987)
[PASS] test_Vault_PartialUnstaking() (gas: 220349)
[PASS] test_Vault_PauseUnpause() (gas: 185182)
[PASS] test_Vault_PauserRoleGrantedToAdmin() (gas: 33984)
[PASS] test_Vault_PrecisionRounding_CalculateWeightedStartTime() (gas: 206268)
[PASS] test_Vault_PrecisionRounding_Lockup() (gas: 221637)
[PASS] test_Vault_PrecisionRounding_StartTime() (gas: 222340)
[PASS] test_Vault_RevertCannotIncreaseAmountInCooldown() (gas: 220592)
[PASS] test_Vault_RevertCannotIncreaseStakeInCooldown() (gas: 221328)
[PASS] test_Vault_RevertEarlyUnstake_AmountExceedsAvailableBalance() (gas: 183222)
[PASS] test_Vault_RevertEarlyUnstake_AmountExceedsAvailableBalance_WithCooldown() (gas: 193473)
[PASS] test_Vault_RevertEarlyUnstake_NoStakeFound() (gas: 26637)
[PASS] test_Vault_RevertEmergencyWithdrawInsufficientETH() (gas: 86089)
[PASS] test_Vault_RevertEmergencyWithdrawNotPaused() (gas: 26334)
[PASS] test_Vault_RevertEmergencyWithdrawUnauthorized() (gas: 54204)
[PASS] test_Vault_RevertEmergencyWithdrawZeroAddress() (gas: 50505)
[PASS] test_Vault_RevertIncreaseAmountDuringCooldown() (gas: 207020)
[PASS] test_Vault_RevertIncreaseAmountNoExistingStake() (gas: 58232)
[PASS] test_Vault_RevertIncreaseLockupBelowMinimum() (gas: 175843)
[PASS] test_Vault_RevertIncreaseLockupNoExistingStake() (gas: 26573)
[PASS] test_Vault_RevertInitiateUnstakeBeforeLockExpiry() (gas: 178047)
[PASS] test_Vault_RevertInitiateUnstake_CooldownAmountOverflow() (gas: 200662)
[PASS] test_Vault_RevertInitiateUnstake_NoStakeFound() (gas: 26678)
[PASS] test_Vault_RevertInitiateUnstake_NoStakeFound_AfterFullUnstake() (gas: 193392)
[PASS] test_Vault_RevertInstantUnstakeAfterLockExpiry() (gas: 179451)
[PASS] test_Vault_RevertPauseNotPauser() (gas: 21017)
[PASS] test_Vault_RevertPauseUnauthorized() (gas: 18719)
[PASS] test_Vault_RevertSetMultiplierContractNotAdmin() (gas: 23797)
[PASS] test_Vault_RevertSetMultiplierContractZeroAddress() (gas: 19565)
[PASS] test_Vault_RevertSetRewardSafeNotAdmin() (gas: 23729)
[PASS] test_Vault_RevertSetRewardSafeZeroAddress() (gas: 19455)
[PASS] test_Vault_RevertStakeAmountTooLarge_ActualUint128InIncreaseAmount() (gas: 316609)
[PASS] test_Vault_RevertStakeAmountTooLarge_ActualUint128Overflow() (gas: 5198)
[PASS] test_Vault_RevertStakeAmountTooLarge_CombineStakeValidation() (gas: 215003)
[PASS] test_Vault_RevertStakeAmountTooLarge_ExceedsUint128Max() (gas: 207998)
[PASS] test_Vault_RevertStakeAmountTooLarge_ExcessiveIncreaseAmount() (gas: 208904)
[PASS] test_Vault_RevertStakeAmountTooLarge_ExcessiveStakeValidation() (gas: 70403)
[PASS] test_Vault_RevertStakeAmountTooLarge_ForceUint128Overflow() (gas: 183302)
[PASS] test_Vault_RevertStakeAmountTooLarge_InitiateUnstakeOverflow() (gas: 200638)
[PASS] test_Vault_RevertStakeAmountTooLarge_NearUint128Max() (gas: 5478)
[PASS] test_Vault_RevertStakeAmountTooLarge_TheoreticalUint128Overflow() (gas: 5219)
[PASS] test_Vault_RevertStakeAmountTooLarge_WeightedCalculationActualOverflow() (gas: 212949)
[PASS] test_Vault_RevertStakeAmountTooLarge_WeightedCalculationOverflow() (gas: 222247)
[PASS] test_Vault_RevertStakeAmountTooLarge_WeightedLockupCalculationOverflow() (gas: 70426)
[PASS] test_Vault_RevertStakeBelowMinimum() (gas: 56521)
[PASS] test_Vault_RevertStakeInvalidLockPeriod() (gas: 56641)
[PASS] test_Vault_RevertStakeWhenPaused() (gas: 78450)
[PASS] test_Vault_RevertStakeZeroAddress() (gas: 22497)
[PASS] test_Vault_RevertUnpauseNotPauser() (gas: 51069)
[PASS] test_Vault_RevertUnstakeBeforeCooldown() (gas: 187807)
[PASS] test_Vault_RevertUnstakeExceedsAmount() (gas: 179656)
[PASS] test_Vault_RevertUnstake_AmountExceedsCooldownAmount() (gas: 195777)
[PASS] test_Vault_RevertUnstake_NoStakeFound() (gas: 26614)
[PASS] test_Vault_RevertUpdateTreasuryUnauthorized() (gas: 21399)
[PASS] test_Vault_RevertUpdateTreasuryZeroAddress() (gas: 19501)
[PASS] test_Vault_RevertValidateIncreaseAmount_InvalidAmount() (gas: 181638)
[PASS] test_Vault_RevertValidateIncreaseAmount_StakeAmountTooLarge() (gas: 213325)
[PASS] test_Vault_SetMultiplierContract() (gas: 30439)
[PASS] test_Vault_SetRewardTreasurySafe() (gas: 31435)
```

```
[PASS] test_Vault_StakeAllLockPeriods() (gas: 591685)
[PASS] test_Vault_StakeMinimumAmount() (gas: 182974)
[PASS] test_Vault_StakeMultipleTimesAddsToSingleStake() (gas: 215575)
[PASS] test_Vault_UpdateSapienTreasury() (gas: 31451)
[PASS] test_Vault_WeightedCalculationOverflow_NewTotalAmount() (gas: 204962)
[PASS] test_Vault_WeightedCalculation_OverflowProtection() (gas: 205569)
Suite result: ok. 91 passed; 0 failed; 0 skipped; finished in 21.58ms (20.88ms CPU time)

Ran 4 tests for test/unit/SapienRewards_EndToEnd.t.sol:SapienRewardsEndToEndTest
[PASS] test_EndToEnd_CompleteUserJourney() (gas: 12615992)
[PASS] test_EndToEnd_EdgeCases() (gas: 218854)
[PASS] test_EndToEnd_ErrorConditions() (gas: 292917)
[PASS] test_EndToEnd_MultiManagerCoordination() (gas: 248908)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 40.95ms (42.33ms CPU time)

Ran 51 tests for test/unit/SapienRewards.t.sol:SapienRewardsTest
[PASS] test_Rewards_CannotInitializeTwice() (gas: 26205)
[PASS] test_Rewards_ClaimExactlyAllAvailableRewards() (gas: 138843)
[PASS] test_Rewards_ClaimRewardRevertsOnDoubleSpend() (gas: 183953)
[PASS] test_Rewards_ClaimRewardRevertsOnECDSARecoveryError() (gas: 105529)
[PASS] test_Rewards_ClaimRewardRevertsOnExceedingMaxAmount() (gas: 131855)
[PASS] test_Rewards_ClaimRewardRevertsOnInvalidSignature() (gas: 115815)
[PASS] test_Rewards_ClaimRewardRevertsOnInvalidSignatureFormat() (gas: 108711)
[PASS] test_Rewards_ClaimRewardRevertsOnMalformedSignature() (gas: 104780)
[PASS] test_Rewards_ClaimRewardRevertsWhenPaused() (gas: 137436)
[PASS] test_Rewards_ClaimRewardWithValidSignature() (gas: 183028)
[PASS] test_Rewards_DepositRevertsOnZeroAmount() (gas: 19105)
[PASS] test_Rewards_DepositRewards() (gas: 94429)
[PASS] test_Rewards_DomainSeparatorRecalculationOnChainFork() (gas: 21672)
[PASS] test_Rewards_FuzzClaimReward(uint256,bytes32) (runs: 256, μ: 141131, ~: 141160)
[PASS] test_Rewards_FuzzDepositAndWithdraw(uint256,uint256) (runs: 256, μ: 101156, ~: 101156)
[PASS] test_Rewards_GetAvailableRewards() (gas: 89336)
[PASS] test_Rewards_GetDomainSeparator() (gas: 18213)
[PASS] test_Rewards_GetOrderRedeemedStatus() (gas: 138747)
[PASS] test_Rewards_GetRewardTokenBalances() (gas: 106167)
[PASS] test_Rewards_Initialize() (gas: 2839002)
[PASS] test_Rewards_InitializeRevertsOnZeroAddresses() (gas: 2835337)
[PASS] test_Rewards_MultipleUsersCanClaimDifferentOrders() (gas: 211404)
[PASS] test_Rewards_NonAdminCannotSetRewardToken() (gas: 909843)
[PASS] test_Rewards_NonPauserCannotPause() (gas: 19630)
[PASS] test_Rewards_NonPauserCannotUnpause() (gas: 49657)
[PASS] test_Rewards_NonRewardSafeCannotDeposit() (gas: 20060)
[PASS] test_Rewards_OnlyAdminCanSetRewardToken() (gas: 919433)
[PASS] test_Rewards_OnlyPauserCanPause() (gas: 43987)
[PASS] test_Rewards_OnlyPauserCanUnpause() (gas: 36085)
[PASS] test_Rewards_OnlyRewardSafeCanDeposit() (gas: 87033)
[PASS] test_Rewards_ReconcileBalance() (gas: 107008)
[PASS] test_Rewards_ReconcileBalanceDoesNothingWhenBalancesMatch() (gas: 92870)
[PASS] test_Rewards_ReconcileBalanceOnlyByRewardSafe() (gas: 19695)
[PASS] test_Rewards_RecoverTokensOnlyByRewardSafe() (gas: 56017)
[PASS] test_Rewards_RecoverTokensRevertsOnInsufficientUnaccounted() (gas: 90132)
[PASS] test_Rewards_RecoverUnaccountedTokens() (gas: 112434)
[PASS] test_Rewards_RewardManagerCannotClaim() (gas: 120792)
[PASS] test_Rewards_SetRewardTokenResetsAvailableRewards() (gas: 974982)
[PASS] test_Rewards_SetRewardTokenRevertsOnZeroAddress() (gas: 19113)
[PASS] test_Rewards_ValidateAndGetHashToSign() (gas: 108143)
[PASS] test_Rewards_ValidateAndGetHashToSignRevertsForNonManager() (gas: 98374)
[PASS] test_Rewards_ValidateAndGetHashToSignRevertsWhenPaused() (gas: 122694)
[PASS] test_Rewards_ValidateRewardParameters() (gas: 94029)
[PASS] test_Rewards_ValidateRewardParametersRevertsOnAlreadyRedeemed() (gas: 172878)
[PASS] test_Rewards_ValidateRewardParametersRevertsOnInsufficientRewards() (gas: 18869)
[PASS] test_Rewards_ValidateRewardParametersRevertsOnZeroAmount() (gas: 16842)
[PASS] test_Rewards_ValidateRewardParametersRevertsOnZeroOrderId() (gas: 16655)
[PASS] test_Rewards_VersionIsCorrect() (gas: 15317)
[PASS] test_Rewards_WithdrawRevertsOnInsufficientBalance() (gas: 88318)
[PASS] test_Rewards_WithdrawRevertsOnZeroAmount() (gas: 19214)
[PASS] test_Rewards_WithdrawRewards() (gas: 100566)
Suite result: ok. 51 passed; 0 failed; 0 skipped; finished in 99.95ms (104.43ms CPU time)

Ran 12 test suites in 168.71ms (289.84ms CPU time): 245 tests passed, 0 failed, 0 skipped (245 total
tests)
```

Code Coverage

The code coverage for the in scope contracts can be viewed by running the following command: `forge coverage --no-match-coverage "script/|test/"`

```
╭------------------------+-------------------+-------------------+------------------+------------------╮
| File                   | % Lines           | % Statements      | % Branches       | % Funcs          |
+======================================================================================================+
| src/Multiplier.sol     | 100.00% (41/41)   | 100.00% (50/50)   | 100.00% (23/23)  | 100.00% (5/5)    |
|------------------------+-------------------+-------------------+------------------+------------------|
| src/SapienQA.sol       | 96.84% (92/95)    | 96.59% (85/88)    | 92.86% (26/28)   | 95.65% (22/23)   |
|------------------------+-------------------+-------------------+------------------+------------------|
| src/SapienRewards.sol  | 100.00% (112/112) | 100.00% (108/108) | 100.00% (24/24)  | 100.00% (27/27)  |
|------------------------+-------------------+-------------------+------------------+------------------|
| src/SapienToken.sol    | 100.00% (5/5)     | 100.00% (4/4)     | 100.00% (1/1)    | 100.00% (2/2)    |
|------------------------+-------------------+-------------------+------------------+------------------|
| src/SapienVault.sol    | 94.60% (298/315)  | 92.16% (329/357)  | 65.22% (45/69)   | 95.35% (41/43)   |
|------------------------+-------------------+-------------------+------------------+------------------|
| src/utils/SafeCast.sol | 60.00% (9/15)     | 60.00% (6/10)     | 40.00% (4/10)    | 60.00% (3/5)     |
|------------------------+-------------------+-------------------+------------------+------------------|
| Total                  | 95.54% (557/583)  | 94.33% (582/617)  | 79.35% (123/155) | 95.24% (100/105) |
╰------------------------+-------------------+-------------------+------------------+------------------╯
```

Changelog

- 2025-06-06 - Initial report
- 2025-07-02 - Final report

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication or other making available of the report to you by Quantstamp.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. 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. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. 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.

Quantstamp