



Cyberscope

Audit Report

HYDT Stablecoin

July 2023

Audited by © cyberscope

Table of Contents

| | |
|---|-----------|
| Table of Contents | 1 |
| Review | 3 |
| Audit Updates | 3 |
| Source Files | 4 |
| Overview | 7 |
| HYDT | 7 |
| sHYDT | 7 |
| HYGT | 8 |
| Reserve | 9 |
| ControlResolver | 10 |
| Earn | 11 |
| Farm | 12 |
| Control | 13 |
| Testing deploy | 14 |
| Roles | 15 |
| Governor Roles | 15 |
| Caller Roles | 15 |
| Locker Role | 15 |
| Architecture | 16 |
| Non-Deterministic Concern | 16 |
| Access Control | 17 |
| Blockchain Data Concern | 18 |
| Stablecoin Rebalance Mechanism | 19 |
| Stimulation | 19 |
| Data Source | 19 |
| Price Formula | 19 |
| Mint Stimulation | 20 |
| Redeem Stimulation | 21 |
| Findings Breakdown | 22 |
| Diagnostics | 23 |
| RSSF - Redundant Staking Status Functionality | 24 |
| Description | 24 |
| Recommendation | 24 |
| Team Update | 24 |
| TAP - Transfer Amount Prevalidation | 26 |
| Description | 26 |
| Recommendation | 26 |
| Team Update | 26 |
| WAV - Withdraw Amount Validation | 27 |

| | |
|--|-----------|
| Description | 27 |
| Recommendation | 27 |
| Team Update | 27 |
| MSC - Missing Sanity Check | 28 |
| Description | 28 |
| Recommendation | 28 |
| Team Update | 28 |
| TUU - Time Units Usage | 29 |
| Description | 29 |
| Recommendation | 29 |
| Team Update | 29 |
| EUU - Ether Units Usage | 30 |
| Description | 30 |
| Recommendation | 30 |
| Team Update | 30 |
| L04 - Conformance to Solidity Naming Conventions | 31 |
| Description | 31 |
| Recommendation | 32 |
| Team Update | 32 |
| L09 - Dead Code Elimination | 33 |
| Description | 33 |
| Recommendation | 33 |
| Team Update | 34 |
| L13 - Divide before Multiply Operation | 35 |
| Description | 35 |
| Recommendation | 35 |
| Team Update | 36 |
| L14 - Uninitialized Variables in Local Scope | 37 |
| Description | 37 |
| Recommendation | 37 |
| Team Update | 37 |
| L19 - Stable Compiler Version | 38 |
| Description | 38 |
| Recommendation | 38 |
| Team Update | 38 |
| Functions Analysis | 39 |
| Inheritance Graph | 56 |
| Flow Graph | 57 |
| Summary | 58 |
| Disclaimer | 59 |
| About Cyberscope | 60 |

Review

Repository<https://github.com/cc416-cr/HYDT-Protocol>

Audit Updates

Initial Audit

01 Jul 2023

<https://github.com/cyberscope-io/audits/blob/main/hydt/v1/audit.pdf>**Corrected Phase 2**

28 Jul 2023

<https://github.com/cyberscope-io/audits/blob/main/hydt/v1/audit.pdf>**Corrected Phase 3**

10 Aug 2023

Source Files

| Filename | SHA256 |
|------------------------------|--|
| sHYDT.sol | 031e7731efaa74aaa8670f478516c4e7a23ebb3ef88674e65281e37f1b4c4193 |
| Reserve.sol | 6089fbba77c3a4dea91109d44296af2d6693d1cdc1b0775d6c322e28aa70b285 |
| HYGT.sol | 2b6aef3c23ebc413bde25253714f9f6dfb06b86d909c382037d09d8b6d9e1665 |
| HYDT.sol | 036c33b781392131a5808c6b8f034b802d08f892d02a75d847ef35dbf9dbbfa3 |
| Farm.sol | 1584d2820ed93fac3725e0e81583c9687f7f6f352a786bd66b5e6bd7f2b12bdc |
| Earn.sol | 623c24e2cbc29c5624a5bc59da63bbd3a3cc52dad6871f19d6a0c7f3f5eb26b3 |
| ControlResolver.sol | 87ce2c78365edb3e98637e86cca4883e5963876fa5162165744c33bbfe405825 |
| Control.sol | 0b560c383c6a82af8bf24dd5bdfb787ea51be47e4c798eb9d86fac9de46e1d22 |
| utils/OpsReady.sol | f6b9dac9b33fe7ae3f17b6f227613addfe9cac8a15dcc5bceb9d30d43906852b |
| utils/ERC20Permit.sol | 3494d8963d1ae76bf605b49d12794409ccec7ab5e605746a0fead52db20e60d |
| utils/ERC165.sol | 5ba0f71e926ac7788defe62c2d25c817ad2d294d6527ef0a3d2cedfb2bea50b6 |
| utils/EIP712.sol | af33b3254994206befe97f514dc999e79bb8656519c4696d97084a03b6a813f7 |

| | |
|--|--|
| utils/Context.sol | 6ee66d1e4693ec63d29393cc2c835947 98475ad0eeabcb0951a35780ef003113 |
| utils/AccessControl.sol | f591651e9d07a9652d4a22eec28a8dcd ed4a6a0996a362e3b9932575314c5449 |
| libraries/Strings.sol | eed73914453d64f56137ea7ae4543ffc4f 88b9195bf0fa10935793b98239103e |
| libraries/SignedMath.sol | e7f09613b16cf73d56bb542acda15b6cf 95fddb6f99edc25d1b1f0fb4bd2d459 |
| libraries/SafeETH.sol | a04437f39de4811921c2622269bcee29 78a6b31aa1712723e89af500b792c74d |
| libraries/SafeERC20.sol | 8aed5c25598b9f1105d0d7fc5974d7300 675707610a8a8263f4c091e4be539d7 |
| libraries/Math.sol | edb6dc365e80055e92109eeecbd0fb50 566050cb5903a1366ad52f7667a46a74 |
| libraries/ECDSA.sol | c37622b4ad062dadfaffe16389b875652 8bbde27d5f86e6f530becc3fca8c06a |
| libraries/DataFetcher.sol | 84fc8c9914442f166121f6c8d26d0740d e3f5948195c0c0179bb97132053d1bc |
| libraries/Counters.sol | e10346b263158ef9aa46f5cbf0432b2f52 21fa84f999074a93cc2fa3bd4fbb92 |
| libraries/Address.sol | 719862a65de3111feeaf4b8c44da81004 61d7946672727f34c1ad5d2633e6342 |
| interfaces/Types.sol | 0ca116128e416264e310f04b99a07a38 bc08529aecba484a4160db8f00d3c453 |
| interfaces/IReserve.sol | f7cd18b4dae2098a7cf4867aed671ff227 3109ccee33ecca9e58fd501c1bfc |
| interfaces/IPancakeRouter02.sol | dcbfee6a4cbf91ca36cf0d295d4130de2 eef84426a64b996d01667f55a624e7d |

| | |
|---------------------------------------|--|
| interfaces/IPancakePair.sol | c7011f889d18d7ab68cab6707c922fda0 ecd7ac5f3a88f2bd2a78b30fe51b833 |
| interfaces/IPancakeFactory.sol | 9c4d5a4084741a49adba2b8b54c3543d 75f9a84217ca042c8a46b00d3b5eaa8a |
| interfaces/IHYGT.sol | cab0017faa61919d6eec1d4db9f8a1577 7941c448a33377b97f027f3772a4ed2 |
| interfaces/IHYDT.sol | 16dc507b01a4468f18f99ae72b730fe5d 2a4eae016e26db7bc5440ec0124f16f |
| interfaces/IERC20Permit.sol | 58367721c02531647b4a0d9203b12464 0cddd36ea2f23d7a181da5e7a368a773 |
| interfaces/IERC20Metadata.sol | d58377f3fda62337e0b78edbb728ed77 537c6bd306cff66840a060cf876956a9 |
| interfaces/IERC20.sol | a7b8d29448cb54f1a9ba7e2a17e9cc03 b62404f083dabb90ff0a6116cf829357 |
| interfaces/IERC165.sol | 48ab1f2a907c12063745ba591bced76c 1d306b6436d43ce9ce9b5ddd6c515989 |
| interfaces/IControl.sol | 4c6079a27480f3f755f6ea72ccc84ed2a5 6acfafe38ab5bc4c34b84232239feb |
| interfaces/IAccessControl.sol | 7817a30761530bd700985139e6e13411 e84d263d86231c6fca147ea77c5b2b5e |
| extensions/ERC20.sol | a353c7109e40f2b86f43ea47f92857d23 64722be6f86299a69ee60ef4f9b9086 |

Overview

HYDT

The contract implements basic minting and burning functionality for the HYDT token, with role-based access control to restrict certain operations.

sHYDT

The sHYDT contract implements an ERC20 token called "sHYDT" (Staked High Yield Dollar Stable Token).

HYGT

The HYGT contract implements the HYGT (High Yield Dollar Governance Token) token.

The contract initializes with the constructor, which mints an initial supply of HYGT tokens and assigns roles to specific addresses.

The contract includes functions for unlocking vested tokens, such as `unlock`, which allows lockers to mint their tokens based on a predetermined schedule.

It includes functions for minting and burning tokens, which are restricted to addresses with the Caller role.

The contract implements a voting mechanism, where token holders can delegate their votes to other addresses using the `delegate` function.

The `getCurrentVotes` and `getPriorVotes` functions provide information about the current and prior voting balances for an address.

The contract includes internal functions for managing vote delegation and checkpoint updates.

Reserve

The Reserve contract manages the storage and withdrawal of BNB tokens.

The contract includes a fallback function (receive) that accepts BNB transfers. It calculates the total reserve value based on the received BNB balance and emits an In event.

The withdraw function allows an authorized caller to withdraw a specified amount of BNB from the contract to their address. It transfers the BNB using the SafeETH.safeTransferETH function and emits an Out event.

Overall, this contract serves as a reserve that accepts incoming BNB transfers and allows authorized callers to withdraw BNB from the reserve. It also calculates and emits events for the total reserve value based on the received BNB balance.

ControlResolver

The ControlResolver contract checks the execution status based on predefined conditions and interacts with the Control contract to execute specific functions when the conditions are met.

The contract contains a checker function that checks the execution status. Within the checker function, the current price, mint progress count, redeem progress count, last executed mint timestamp, and last executed redeem timestamp are retrieved from the CONTROL contract.

The checker function evaluates conditions for executing the mint or redeems functions based on the price and the time elapsed since the last execution.

If the conditions for executing mint or redeem are met, the execute function of the CONTROL contract is encoded as a function call and returned as the execPayload.

If the conditions are not met, information about the price, mint/redeem last execution time, and progress counts are returned as the execPayload.

Earn

The Earn contract allows users to stake their HYDT (High Yield Dollar Stable Token) and earn rewards in both HYDT and HYG (High Yield Governance Token).

The contract includes events to emit relevant information when users stake, claim payouts, or unstake.

The contract provides functions to update the allocation points for pools, update reward variables, and retrieve pending rewards and payouts for users.

Users can stake their HYDT tokens using the stake function, specifying the amount and stake type. A fee is deducted, and the remaining amount is transferred to the contract and converted to sHYDT tokens.

Users can claim their pending rewards and payouts using the claimPayout function, providing the index of the staking position.

Farm

The Farm contract implements staking LP tokens and earning HYGToken as rewards.

The contract provides functions for adding pools, updating allocation points, and updating reward variables for all pools.

There are functions to get pending HYGToken rewards for a user in each pool, mass update pools to update reward variables for all pools, and update pool-specific reward variables.

The contract includes functions for depositing LP tokens to receive HYGToken rewards, withdrawing LP tokens from a pool, and emergency withdrawing LP tokens without caring about rewards.

Control

The Control contract implements various functionalities related to the control and operation of a decentralized application (DApp). The Control encapsulates the fundamental functionality of the ecosystem, being responsible for executing the mint and redeem mechanisms to rebalance the stablecoin.

It defines several state variables including role constants, time duration variables, addresses of external contracts, initial minting limits, price bounds, and instances of other contracts.

The contract emits various events to notify important state changes.

The contract includes various external and internal functions for controlling the contract's behavior. These functions handle tasks such as updating slippage tolerance, updating the ops ready state, delegating token approvals, getting initial minting information, getting the current HYDT price, performing initial minting, executing operations to maintain the peg, and more.

Testing deploy

| Contract Name | Explorer |
|---------------------|---|
| HYDT | https://testnet.bscscan.com/address/0x5917818f8b418dC8c03Ca122485E9f81D717b4D4 |
| HYGT | https://testnet.bscscan.com/address/0x956202e240033EA82F65dFeA7B28b5eF5Dfe5726ress/0x4179f4384160dA284605F6b34316134179E328F9 |
| sHYDT | https://testnet.bscscan.com/address/0x7BE0bC7FB4d083651365971Faf3fB0339624d7a8 |
| Control | https://testnet.bscscan.com/address/0x29959aA684CF13977Ff1D19Fb4313782c9E2CEA9 |
| ControlReso lver | https://testnet.bscscan.com/address/0xe6Ef5546694953f5458De063a9F28A9d807c1d29 |
| Earn | https://testnet.bscscan.com/address/0x18E33d855aFFd477da480e4370267f0cabB881dE |
| Farm | https://testnet.bscscan.com/address/0x097f9A6795150Fc1684Fa4FA1CdE81f23F94c02D |
| Reserve | https://testnet.bscscan.com/address/0x29e07C0e024D8479D25eEFe368C516c722D61a1d |

Roles

The ecosystem roles consist of three roles.

Governor Roles

The Governor role is responsible for configuring ecosystem parameters.

Caller Roles

The Caller role is responsible for calling functions that make changes to the ecosystem.

Locker Role

The Locker role is responsible for unlocking vested tokens.

Architecture

Non-Deterministic Concern

It is crucial to prioritize a deterministic implementation. By ensuring that the system behaves consistently and predictably, we can enhance user trust and confidence. The contract's deterministic performance may be impacted due to its reliance on an external service. As the ecosystem utilizes this external service, it introduces an element of uncertainty that can affect the deterministic behavior of the contract.

One approach to optimize and achieve determinism within the ecosystem is to transfer the stablecoin rebalance functionality to the transfer transaction. By integrating the rebalance process directly into the transfer transaction, the ecosystem gains the advantage of greater efficiency and predictability. This implementation ensures that every transfer of stablecoins is accompanied by an automatic rebalancing action, maintaining the stability of the ecosystem. This optimization leads to a more streamlined and reliable operation, reducing the need for separate rebalancing mechanisms and providing a more coherent and synchronized experience for users. Additionally, this approach enhances the overall robustness of the ecosystem, ensuring smooth and deterministic rebalancing, regardless of external factors or market fluctuations.

Access Control

To streamline the roles and access control functionality within the ecosystem, a recommended approach is to consider moving them to a library contract. Consolidating all the roles and access control logic into a separate library contract offers several advantages. Firstly, it promotes code reusability and modularity, allowing for cleaner and more maintainable contract architecture. Secondly, centralizing these functionalities in a library contract enables easier upgrades and modifications, as changes made to the library contract propagate throughout the ecosystem. Moreover, by decoupling the roles and access control logic from the main contract, potential security risks and complexities can be reduced. This approach enhances code readability, simplifies contract development, and provides a robust foundation for managing roles and access control throughout the ecosystem.

The presence of multiple instances of the same roles is observed in a significant number of contracts. For instance,

```
bytes32 public constant GOVERNOR_ROLE =  
    keccak256(abi.encodePacked("Governor")) ;  
bytes32 public constant CALLER_ROLE =  
    keccak256(abi.encodePacked("Caller")) ;
```

Blockchain Data Concern

In several instances, the contract makes use of the same constant variables across multiple contracts, resulting in duplication of code and potential maintenance challenges. To address this, it would be beneficial to consider moving these common variables to a utility contract. By centralizing these shared variables in a utility contract, we can promote code reusability, enhance maintainability, and reduce redundancy across multiple contracts.

```
address public constant PANCAKE_FACTORY =  
0xcA143Ce32Fe78f1f7019d7d551a6402fC5350c73;  
address public constant WBNB =  
0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;  
address public constant USDT =  
0x55d398326f99059fF775485246999027B3197955;
```

Furthermore, although the contract utilizes a router, it currently manually adds information to the contract instead of retrieving it from the router. This approach may lead to inconsistencies and manual errors.

Stablecoin Rebalance Mechanism

Upon reviewing the rebalance architecture of the stablecoin, we observed that the process relies on both the current price of the stablecoin and the reserve balance of the ecosystem. Furthermore, to prevent significant fluctuations in stablecoin prices, the rebalance functionality is capped with a percentage of the liquidity amount.

Stimulation

The purpose of the simulation was to observe how the rebalance functionality behaves through multiple mints and redeems. After each mint or redeem the price was calculated. Furthermore, the Mint and Redeem formulas were simulated with 0% slippage.

By conducting simulations on the mint and redeem functionalities, we observed that the mint function tends to exhibit more aggressive price changes compared to the redeem function, as indicated in the chart.

Data Source

Data was forked from BSC Blockchain for the simulation, meaning that prices and pair supplies were retrieved directly from the blockchain.

Price Formula

The formula calculates the price of the token HYDT in terms of the BUSD token

1. Calculate the price of 1 HYDT token in terms of WBNB (Wrapped Binance Coin):

```
HYDTBNBprice = 1 HYDT * HYDTpair.WBNBBalance /  
HYDTpair.BUSDBalance
```

2. Calculate the price of 1 HYDT token in terms of BUSD: `price = (1`

```
HYDTBNBprice * BUSDpair.BUSDBalance) / BUSDpair.WBNBBalance
```

```
price = (((1 HYDT * HYDTpair.WBNBBalance) / HYDTpair.BUSDBalance) *  
BUSDpair.BUSDBalance) / BUSDpair.WBNBBalance
```

Mint Stimulation

Illustrates the price change in relation to mint executions.

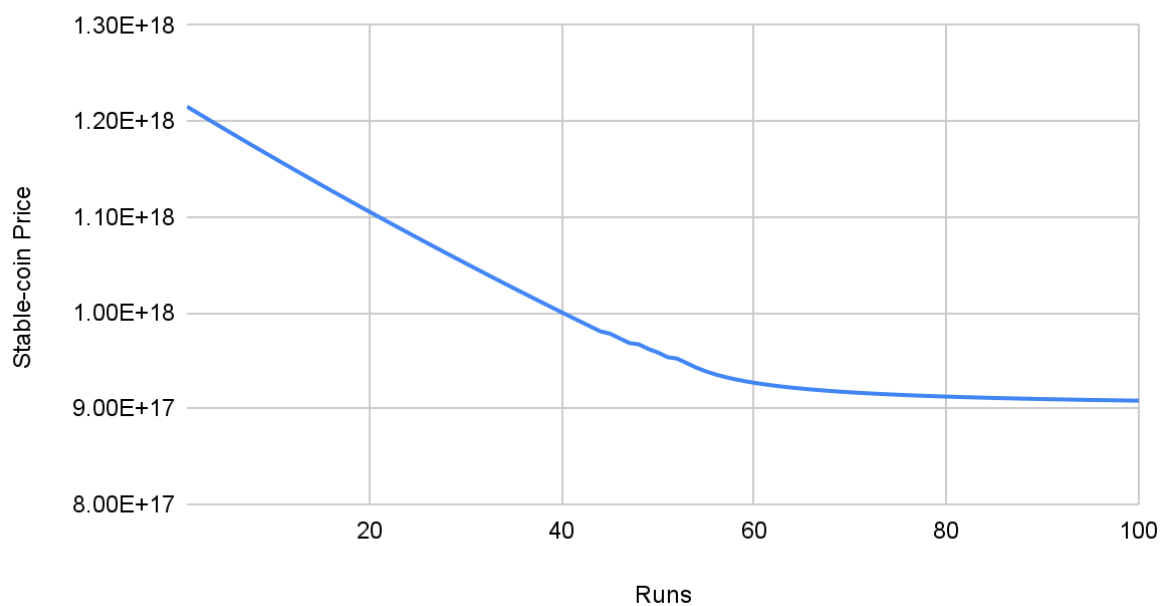
The mint amount was calculated with the following formula:

$$\text{Mint amount} = (\text{Current price} - 0.9)^2 * \text{Reservebalance} * 0.04$$

Mint stimulation starting price: 1.20 \$

Mint stimulation ending price: 0.9 \$

Mint Stimulation



Redeem Stimulation

Illustrates the price change in relation to multiple redeem executions.

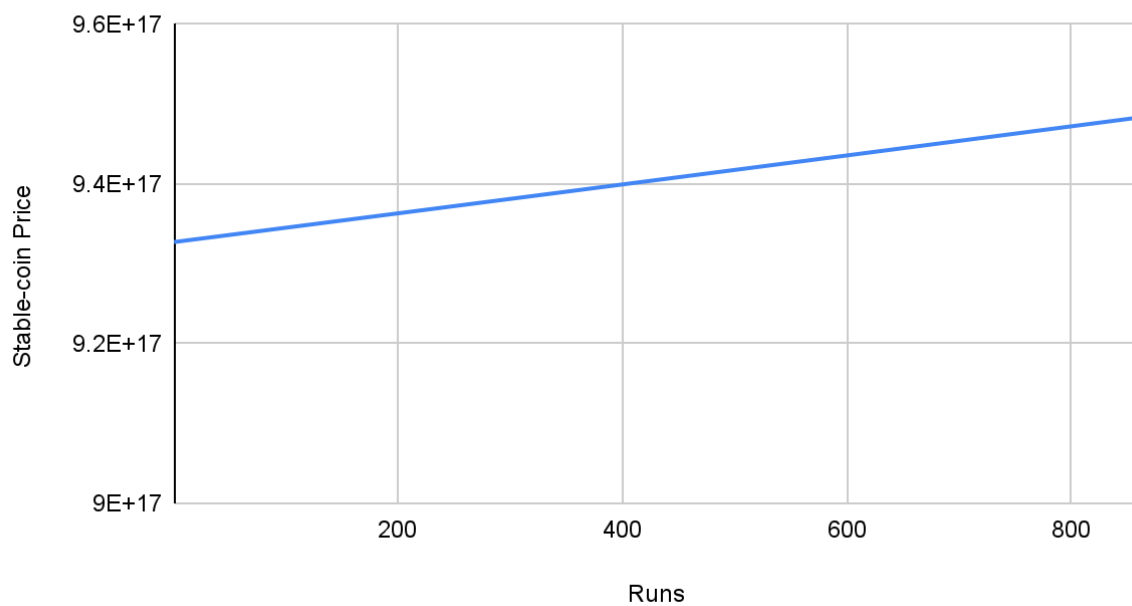
The redeem amount was calculated with the following formula :

$$\text{Redeem amount} = (1.1 - \text{Current price})^2 * \text{Reservebalance} * 0.004$$

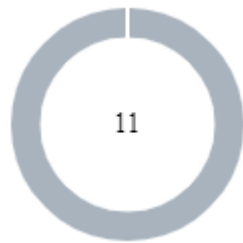
Redeem stimulation starting price: 0.93 \$

Redeem stimulation ending price: 0.95 \$

Redeem Stimulation



Findings Breakdown



| | |
|-----------------------|----|
| ● Critical | 0 |
| ● Medium | 0 |
| ● Minor / Informative | 11 |

| Severity | Unresolved | Acknowledged | Resolved | Other |
|-----------------------|------------|--------------|----------|-------|
| ● Critical | 0 | 0 | 0 | 0 |
| ● Medium | 0 | 0 | 0 | 0 |
| ● Minor / Informative | 5 | 5 | 0 | 1 |

Diagnostics

● Critical ● Medium ● Minor / Informative

| Severity | Code | Description | Status |
|----------|------|--|--------------|
| ● | RSSF | Redundant Staking Status Functionality | Acknowledged |
| ● | TAP | Transfer Amount Prevalidation | Acknowledged |
| ● | WAV | Withdraw Amount Validation | Acknowledged |
| ● | MSC | Missing Sanity Check | SemiResolved |
| ● | TUU | Time Units Usage | Acknowledged |
| ● | EUU | Ether Units Usage | Acknowledged |
| ● | L04 | Conformance to Solidity Naming Conventions | Acknowledged |
| ● | L09 | Dead Code Elimination | Acknowledged |
| ● | L13 | Divide before Multiply Operation | Acknowledged |
| ● | L14 | Uninitialized Variables in Local Scope | Acknowledged |
| ● | L19 | Stable Compiler Version | Acknowledged |

RSSF - Redundant Staking Status Functionality

| | |
|-------------|---------------------|
| Criticality | Minor / Informative |
| Location | Earn.sol#L417 |
| Status | Acknowledged |

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract is utilizing the `staking.status` variable in the contract implementation. The staking status variable is used only to determine whether the staking element exists. There are variables that can be reused to determine whether the staking element exists. Hence, the `staking.status` is redundant.

```
function stake(uint256 amount, uint8 stakeType) external {  
    ...  
    staking.status = true;  
  
    function claimPayout(uint256 index) external {  
        require(staking.status, "Earn: invalid staking");  
    }  
}
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it. It is recommended to reuse existing variables to determine if a staking element exists in the ecosystem. For instance, The contract could utilize the `staking.lastClaimTime`

```
require(staking.lastClaimTime>0);
```

Team Update

"We are choosing to keep the redundant variables such as status which will be a tradeoff between readability/accessibility and storage. "

TAP - Transfer Amount Prevalidation

| | |
|-------------|-----------------------------------|
| Criticality | Minor / Informative |
| Location | Earn.sol#L383 Control.sol#L284 |
| Status | Acknowledged |

Description

The current implementation of the contract does not prevalidate whether sufficient tokens are available to perform transactions, which can lead to transaction failures or unexpected behavior. For example, if a user attempts to transfer tokens that are not available, the transaction will fail and potentially leave the contract in an inconsistent state.

```
SafeERC20.safeTransferFrom(HYDT, _msgSender(), TREASURY, fee);  
SafeETH.safeTransferETH(address(RESERVE), msg.value);
```

Recommendation

It is recommended to implement prevalidation checks before any transaction. To ensure that there are sufficient tokens available for any transaction. By performing prevalidation checks, the contract will be more reliable and less prone to errors or vulnerabilities.

Team Update

"Choosing to rely on external reverts is detrimental to gas costs when a failing transaction is sent but will be better for transactions that have correct arguments."

WAV - Withdraw Amount Validation

| | |
|-------------|---------------------|
| Criticality | Minor / Informative |
| Location | Reserve.sol#L74 |
| Status | Acknowledged |

Description

The contract is missing withdraw amount validation in the `withdraw` function. The absence of validation may lead to revert of the method.

```
function withdraw(uint256 amount) external
onlyRole(CALLER_ROLE) {
    SafeETH.safeTransferETH(_msgSender(), amount);
    uint256 totalReserveBNB = address(this).balance;
    uint256 totalReserve = DataFetcher.quote(PANCAKE_FACTORY,
totalReserveBNB, WBNB, USDT);

    emit Out(_msgSender(), amount, totalReserveBNB,
totalReserve);
}
```

Recommendation

It is recommended to implement a sufficient withdraw amount validation mechanisms. A recommended approach could be to verify that the withdrawal amount is sufficient to perform the transaction.

Team Update

"Choosing to rely on external reverts is detrimental to gas costs when a failing transaction is sent but will be better for transactions that have correct arguments."

MSC - Missing Sanity Check

| | |
|-------------|-----------------------|
| Criticality | Minor / Informative |
| Location | Earn.sol#L268,294,383 |
| Status | SemiResolved |

Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

The arguments `user`, and `index` are not properly sanitized.

```
function getPending(uint256 pid, address user) public view
returns (uint256) { ... }
function getPending(address user, uint256 index) public view
returns (uint256) { ... }
...
```

Recommendation

The team is advised to properly check the variables according to the required specifications.

- The variable `index` should be a valid index of the pool indexes.
- The variable `user` should be a valid address and it must be a valid user.

Team Update

"Added "require" statements in all mentioned write functions. All read functions mentioned were not changed because of a few reasons. First reason being that all write functions that utilise the aforementioned read functions already have the "require" statements, and secondly to avoid conflicting with the front end read function must not have any "require" statements."

TUU - Time Units Usage

| | |
|--------------------|---|
| Criticality | Minor / Informative |
| Location | ControlResolver.sol#L15,16 HYGT.sol#L19Farm.sol#L19,31 Earn.sol#L20,21,136,139 Control.sol#L25,26,27 |
| Status | Acknowledged |

Description

The contract is using arbitrary numbers to form time-related values. As a result, it decreases the readability of the codebase and prevents the compiler to optimize the source code.

```
uint128 private constant FOUR_HOURS_TIME = 14400;  
uint128 private constant FIVE_MINUTES_TIME = 300;  
  
uint128 private constant ONE_MONTH_TIME = 2592000;  
  
uint128 private constant ONE_DAY_TIME = 86400;  
lockPeriods = [7776000, 15552000, 31536000];  
HYGTPerSecond = 0.6666666666666667 * 1e18;  
  
uint256 averageBlockTime = 3;
```

Recommendation

It is a good practice to use the time units reserved keywords like `seconds`, `minutes`, `hours`, `days` and `weeks` to process time-related calculations.

It's important to note that these time units are simply a shorthand notation for representing time in seconds, and do not have any effect on the actual passage of time or the execution of the contract. The time units are simply a convenience for expressing time in a more human-readable form.

Team Update

"Will be kept as is."

EUU - Ether Units Usage

| | |
|-------------|------------------------|
| Criticality | Minor / Informative |
| Location | Control.sol#L52,53,249 |
| Status | Acknowledged |

Description

The contract is using arbitrary numbers to form ether-related values. As a result, it decreases the readability of the codebase and prevents the compiler to optimize the source code.

```
uint256 private constant PRICE_UPPER_BOUND = 1.02 * 1e18;  
uint256 private constant PRICE_LOWER_BOUND = 0.98 * 1e18;  
  
uint256 amountIn = 1 * 1e18;
```

Recommendation

It is a good practice to use the ether units reserved keyword `ethers` to process ether-related calculations.

It's important to note that these ether unit is simply a shorthand notation for representing ether, and do not have any effect on the actual passage of ether or the execution of the contract. The ether unit id simply a convenience for expressing ethers in a more human-readable form. For instance,

```
uint128 private constant PRICE_UPPER_BOUND = 1.02 ether;  
uint128 private constant PRICE_LOWER_BOUND = 0.98 * ether;
```

Team Update

"Will be kept as is."

L04 - Conformance to Solidity Naming Conventions

| | |
|--------------------|--|
| Criticality | Minor / Informative |
| Location | sHYDT.sol#L11 HYGT.sol#L27 Farm.sol#L22,25 Earn.sol#L24,26,30,42 ControlResolver.sol#L23 Control.sol#L56,58 |
| Status | Acknowledged |

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.


```
contract sHYDT is ERC20, IHYDT, AccessControl {  
  
    /* ===== STATE VARIABLES ===== */  
  
    bytes32 public constant CALLER_ROLE =  
    keccak256(abi.encodePacked("Caller"));  
  
    ...  
    function burnFrom(address from, uint256 amount) external  
    override onlyRole(CALLER_ROLE) returns (bool) {  
        address spender = _msgSender();  
        _spendAllowance(from, spender, amount);  
        _burn(from, amount);  
        return true;  
    }  
}  
  
...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation

<https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention>.

Team Update

"Will be kept as is."

L09 - Dead Code Elimination

| | |
|-------------|--|
| Criticality | Minor / Informative |
| Location | utils/AccessControl.sol#L205,214 libraries/DataFetcher.sol#L44,60 |
| Status | Acknowledged |

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function _setupRole(bytes32 role, address account) internal
virtual {
    _grantRole(role, account);
}

function _setRoleAdmin(bytes32 role, bytes32 adminRole)
internal virtual {
    bytes32 previousAdminRole = getRoleAdmin(role);
    _roles[role].adminRole = adminRole;
    emit RoleAdminChanged(role, previousAdminRole,
adminRole);
}

...
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

Team Update

"Will be kept as is."

L13 - Divide before Multiply Operation

| | |
|--------------------|---|
| Criticality | Minor / Informative |
| Location | HYGT.sol#L183,184,187 Farm.sol#L193,194,221,222 Earn.sol#L230,231,306,307,369,373,374,430,431 Control.sol#L234,235,330,331,337,371,405,406,410,412,450 |
| Status | Acknowledged |

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of precision.

```
uint256 numberOfIntervals = (block.timestamp - lastUnlockTime)
/ ONE_MONTH_TIME
uint256 unlockAmount = (lock.totalAmount * numberOfIntervals) /
lock.totalIntervals
numberOfIntervals = numberOfIntervals > maxIntervals ?
maxIntervals : numberOfIntervals

uint256 HYGTReward = (numberOfBlocks * HYGTPerBlock *
pool.allocPoint) / totalAllocPoint
accHYGTPerShare += ((HYGTReward * 1e12) / lpSupply)

uint256 HYGTReward = (numberOfSeconds * HYGTPerSecond *
pool.allocPoint) / totalAllocPoint
pool.accHYGTPerShare += (HYGTReward * 1e12) / stakeSupply

uint256 secondValue = ((0.0025 * 1e4) * amountLiquidity) / 1e4
uint256 reduction = (amountRedeem * 1e18) / baseValue
uint256 amountRedeem = firstValue < secondValue ? firstValue :
secondValue
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses

to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

Team Update

"Will be kept as is. (Changed where deemed necessary, HYG.T.sol#L187 previously L194)."

L14 - Uninitialized Variables in Local Scope

| | |
|--------------------|--|
| Criticality | Minor / Informative |
| Location | Farm.sol#L133 Earn.sol#L168,281,295,318,350,364,397 Control.sol#L345,419,422 |
| Status | Acknowledged |

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
PoolInfo memory pool
uint256 totalPending
uint256 pending
uint256 totalDailyPayout
uint256 totalPayout
uint256 payout
Staking memory staking
bool check
uint256 amountBurnHYDT
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

Team Update

"Will be kept as is."

L19 - Stable Compiler Version

| | |
|--------------------|--|
| Criticality | Minor / Informative |
| Location | utils/AccessControl.sol#L4 interfaces/IAccessControl.sol#L4 |
| Status | Acknowledged |

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.0;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

Team Update

"Will be kept as is."

Functions Analysis

| Contract | Type | Bases | | |
|----------------|----------------|---|------------|----------------------|
| | Function Name | Visibility | Mutability | Modifiers |
| | | | | |
| sHYDT | Implementation | ERC20, IHYDT, AccessContr ol | | |
| | | Public | ✓ | ERC20 |
| | initialize | External | ✓ | - |
| | mint | External | ✓ | onlyRole |
| | burn | External | ✓ | - |
| | burnFrom | External | ✓ | onlyRole |
| | | | | |
| Reserve | Implementation | AccessContr ol | | |
| | | Public | ✓ | - |
| | initialize | External | ✓ | - |
| | | External | Payable | - |
| | withdraw | External | ✓ | onlyRole |
| | | | | |
| HYGT | Implementation | IHYGT, AccessContr ol, ERC20Permi t | | |
| | | Public | ✓ | ERC20 ERC20Permit |

| | | | | |
|-------------|----------------------|---|---|----------------------|
| | initialize | External | ✓ | - |
| | maxTotalSupply | External | | - |
| | getCurrentVotes | External | | - |
| | getPriorVotes | External | | - |
| | unlock | External | ✓ | onlyRole |
| | mint | External | ✓ | onlyRole |
| | burn | External | ✓ | - |
| | burnFrom | External | ✓ | onlyRole |
| | delegate | External | ✓ | - |
| | _delegate | Internal | ✓ | |
| | _beforeTokenTransfer | Internal | ✓ | |
| | _moveDelegates | Internal | ✓ | |
| | _writeCheckpoint | Internal | ✓ | |
| | | | | |
| HYDT | Implementation | IHYDT, AccessContr ol, ERC20Permi t | | |
| | | Public | ✓ | ERC20 ERC20Permit |
| | initialize | External | ✓ | - |
| | mint | External | ✓ | onlyRole |
| | burn | External | ✓ | - |
| | burnFrom | External | ✓ | onlyRole |
| | | | | |
| Farm | Implementation | AccessContr ol | | |

| | | | | |
|-------------|----------------------------|-------------------|---|----------|
| | | Public | ✓ | - |
| | initialize | External | ✓ | - |
| | poolLength | External | | - |
| | addPool | External | ✓ | onlyRole |
| | _addPool | Internal | ✓ | |
| | updateAllocation | External | ✓ | onlyRole |
| | massUpdatePools | Public | ✓ | - |
| | updatePool | Public | ✓ | - |
| | getPendingBatch | External | | - |
| | getPending | Public | | - |
| | deposit | External | ✓ | - |
| | withdraw | External | ✓ | - |
| | withdrawRewards | External | ✓ | - |
| | emergencyWithdraw | Public | ✓ | - |
| | | | | |
| Earn | Implementation | AccessContr ol | | |
| | | Public | ✓ | - |
| | initialize | External | ✓ | - |
| | poolLength | External | | - |
| | allPoolInfo | External | | - |
| | _addPool | Internal | ✓ | |
| | updateAllocationWithUpdate | Public | ✓ | onlyRole |
| | massUpdatePools | Public | ✓ | - |

| | | | | |
|------------------------|-------------------------|-------------------------|---------|----------|
| | updatePool | Public | ✓ | - |
| | _binarySearchShare | Internal | | |
| | getPendingBatch | External | | - |
| | getPendingType | External | | - |
| | getPending | Public | | - |
| | getDailyPayoutBatch | External | | - |
| | getPayoutBatch | External | | - |
| | getPayoutType | External | | - |
| | getPayout | Public | | - |
| | stake | External | ✓ | - |
| | claimPayout | External | ✓ | - |
| | | | | |
| ControlResolver | Implementation | Context | | |
| | | Public | ✓ | - |
| | initialize | External | ✓ | - |
| | checker | External | | - |
| | | | | |
| Control | Implementation | AccessControl, OpsReady | | |
| | | Public | ✓ | - |
| | initialize | External | ✓ | - |
| | | External | Payable | - |
| | updateSlippageTolerance | External | ✓ | onlyRole |
| | updateOpsReadyState | External | ✓ | onlyRole |

| | | | | |
|--------------------|------------------------------|-----------------------------------|---------|----------|
| | delegateApprove | External | ✓ | onlyRole |
| | _delegateApprove | Internal | ✓ | |
| | getInitialMints | External | | - |
| | getDailyInitialMints | External | | - |
| | _getNextDailyInitialMintTime | Internal | | |
| | getCurrentPrice | Public | | - |
| | initialMint | External | Payable | - |
| | execute | External | ✓ | onlyRole |
| | _mint | Internal | ✓ | |
| | _redeem | Internal | ✓ | |
| | | | | |
| OpsReady | Implementation | | | |
| | _transfer | Internal | ✓ | |
| | _getFeeDetails | Internal | | |
| | | | | |
| ERC20Permit | Implementation | ERC20, IERC20Permit, EIP712 | | |
| | | Public | ✓ | EIP712 |
| | permit | Public | ✓ | - |
| | nonces | Public | | - |
| | DOMAIN_SEPARATOR | External | | - |
| | _useNonce | Internal | ✓ | |
| | | | | |
| ERC165 | Implementation | IERC165 | | |

| | | | | |
|----------------------|-----------------------|---------------------------------------|---|----------|
| | supportsInterface | Public | | - |
| | | | | |
| EIP712 | Implementation | | | |
| | | Public | ✓ | - |
| | _domainSeparatorV4 | Internal | | |
| | _buildDomainSeparator | Private | | |
| | _hashTypedDataV4 | Internal | | |
| | | | | |
| Context | Implementation | | | |
| | _msgSender | Internal | | |
| | _msgData | Internal | | |
| | | | | |
| AccessControl | Implementation | Context, IAccessControl, ERC165 | | |
| | supportsInterface | Public | | - |
| | hasRole | Public | | - |
| | _checkRole | Internal | | |
| | _checkRole | Internal | | |
| | getRoleAdmin | Public | | - |
| | grantRole | Public | ✓ | onlyRole |
| | revokeRole | Public | ✓ | onlyRole |
| | renounceRole | Public | ✓ | - |
| | _setupRole | Internal | ✓ | |
| | _setRoleAdmin | Internal | ✓ | |

| | | | | |
|-------------------|------------------|----------|---|--|
| | _grantRole | Internal | ✓ | |
| | _revokeRole | Internal | ✓ | |
| | | | | |
| Strings | Library | | | |
| | toString | Internal | | |
| | toString | Internal | | |
| | toHexString | Internal | | |
| | toHexString | Internal | | |
| | toHexString | Internal | | |
| | equal | Internal | | |
| | | | | |
| SignedMath | Library | | | |
| | max | Internal | | |
| | min | Internal | | |
| | average | Internal | | |
| | abs | Internal | | |
| | | | | |
| SafeETH | Library | | | |
| | safeTransferETH | Internal | ✓ | |
| | | | | |
| SafeERC20 | Library | | | |
| | safeTransfer | Internal | ✓ | |
| | safeTransferFrom | Internal | ✓ | |

| | | | | |
|--------------|-----------------------|----------|---|--|
| | safeApprove | Internal | ✓ | |
| | safeIncreaseAllowance | Internal | ✓ | |
| | safeDecreaseAllowance | Internal | ✓ | |
| | safePermit | Internal | ✓ | |
| | _callOptionalReturn | Private | ✓ | |
| | | | | |
| Math | Library | | | |
| | max | Internal | | |
| | min | Internal | | |
| | average | Internal | | |
| | ceilDiv | Internal | | |
| | mulDiv | Internal | | |
| | mulDiv | Internal | | |
| | sqrt | Internal | | |
| | sqrt | Internal | | |
| | log2 | Internal | | |
| | log2 | Internal | | |
| | log10 | Internal | | |
| | log10 | Internal | | |
| | log256 | Internal | | |
| | log256 | Internal | | |
| | | | | |
| ECDSA | Library | | | |

| | | | | |
|--------------------|------------------------|----------|---|--|
| | _throwError | Private | | |
| | tryRecover | Internal | | |
| | recover | Internal | | |
| | tryRecover | Internal | | |
| | recover | Internal | | |
| | tryRecover | Internal | | |
| | recover | Internal | | |
| | toEthSignedMessageHash | Internal | | |
| | toEthSignedMessageHash | Internal | | |
| | toTypedDataHash | Internal | | |
| | | | | |
| DataFetcher | Library | | | |
| | pairFor | Internal | | |
| | getReserves | Internal | | |
| | quote | Internal | | |
| | quoteBatch | Internal | | |
| | quoteRouted | Internal | | |
| | | | | |
| Counters | Library | | | |
| | current | Internal | | |
| | increment | Internal | ✓ | |
| | decrement | Internal | ✓ | |
| | reset | Internal | ✓ | |

| | | | | |
|----------------|----------------------------|----------|---|---|
| | | | | |
| Address | Library | | | |
| | isContract | Internal | | |
| | sendValue | Internal | ✓ | |
| | functionCall | Internal | ✓ | |
| | functionCall | Internal | ✓ | |
| | functionCallWithValue | Internal | ✓ | |
| | functionCallWithValue | Internal | ✓ | |
| | functionStaticCall | Internal | | |
| | functionStaticCall | Internal | | |
| | functionDelegateCall | Internal | ✓ | |
| | functionDelegateCall | Internal | ✓ | |
| | verifyCallResultFromTarget | Internal | | |
| | verifyCallResult | Internal | | |
| | _revert | Private | | |
| | | | | |
| IOps | Interface | | | |
| | createTask | External | ✓ | - |
| | cancelTask | External | ✓ | - |
| | getFeeDetails | External | | - |
| | gelato | External | | - |
| | taskTreasury | External | | - |
| | | | | |

| | | | | |
|--------------------------------|------------------------------|----------|---------|---|
| ITaskTreasuryUpgradable | Interface | | | |
| | depositFunds | External | Payable | - |
| | withdrawFunds | External | ✓ | - |
| | | | | |
| IOpsProxyFactory | Interface | | | |
| | getProxyOf | External | | - |
| | | | | |
| IReserve | Interface | | | |
| | withdraw | External | ✓ | - |
| | | | | |
| IPancakeRouter01 | Interface | | | |
| | factory | External | | - |
| | WETH | External | | - |
| | addLiquidity | External | ✓ | - |
| | addLiquidityETH | External | Payable | - |
| | removeLiquidity | External | ✓ | - |
| | removeLiquidityETH | External | ✓ | - |
| | removeLiquidityWithPermit | External | ✓ | - |
| | removeLiquidityETHWithPermit | External | ✓ | - |
| | swapExactTokensForTokens | External | ✓ | - |
| | swapTokensForExactTokens | External | ✓ | - |
| | swapExactETHForTokens | External | Payable | - |
| | swapTokensForExactETH | External | ✓ | - |

| | | | | |
|-------------------------|---|------------------|---------|---|
| | swapExactTokensForETH | External | ✓ | - |
| | swapETHForExactTokens | External | Payable | - |
| | quote | External | | - |
| | getAmountOut | External | | - |
| | getAmountIn | External | | - |
| | getAmountsOut | External | | - |
| | getAmountsIn | External | | - |
| | | | | |
| IPancakeRouter02 | Interface | IPancakeRouter01 | | |
| | removeLiquidityETHSupportingFeeOnTransferTokens | External | ✓ | - |
| | removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External | ✓ | - |
| | swapExactTokensForTokensSupportingFeeOnTransferTokens | External | ✓ | - |
| | swapExactETHForTokensSupportingFeeOnTransferTokens | External | Payable | - |
| | swapExactTokensForETHSupportingFeeOnTransferTokens | External | ✓ | - |
| | | | | |
| IPancakePair | Interface | | | |
| | name | External | | - |
| | symbol | External | | - |
| | decimals | External | | - |
| | totalSupply | External | | - |
| | balanceOf | External | | - |
| | allowance | External | | - |

| | | | | |
|------------------------|----------------------|----------|---|---|
| | approve | External | ✓ | - |
| | transfer | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | DOMAIN_SEPARATOR | External | | - |
| | PERMIT_TYPEHASH | External | | - |
| | nonces | External | | - |
| | permit | External | ✓ | - |
| | MINIMUM_LIQUIDITY | External | | - |
| | factory | External | | - |
| | token0 | External | | - |
| | token1 | External | | - |
| | getReserves | External | | - |
| | price0CumulativeLast | External | | - |
| | price1CumulativeLast | External | | - |
| | kLast | External | | - |
| | mint | External | ✓ | - |
| | burn | External | ✓ | - |
| | swap | External | ✓ | - |
| | skim | External | ✓ | - |
| | sync | External | ✓ | - |
| | initialize | External | ✓ | - |
| | | | | |
| IPancakeFactory | Interface | | | |

| | | | | |
|--------------|-----------------|----------|---|---|
| | feeTo | External | | - |
| | feeToSetter | External | | - |
| | getPair | External | | - |
| | allPairs | External | | - |
| | allPairsLength | External | | - |
| | createPair | External | ✓ | - |
| | setFeeTo | External | ✓ | - |
| | setFeeToSetter | External | ✓ | - |
| | | | | |
| IHYGT | Interface | IERC20 | | |
| | maxTotalSupply | External | | - |
| | mint | External | ✓ | - |
| | burn | External | ✓ | - |
| | burnFrom | External | ✓ | - |
| | delegate | External | ✓ | - |
| | getCurrentVotes | External | | - |
| | getPriorVotes | External | | - |
| | | | | |
| IHYDT | Interface | IERC20 | | |
| | mint | External | ✓ | - |
| | burn | External | ✓ | - |
| | burnFrom | External | ✓ | - |
| | | | | |

| | | | | |
|-----------------------|-------------------|----------|---|---|
| IERC20Permit | Interface | | | |
| | permit | External | ✓ | - |
| | nonces | External | | - |
| | DOMAIN_SEPARATOR | External | | - |
| | | | | |
| IERC20Metadata | Interface | IERC20 | | |
| | name | External | | - |
| | symbol | External | | - |
| | decimals | External | | - |
| | | | | |
| IERC20 | Interface | | | |
| | totalSupply | External | | - |
| | balanceOf | External | | - |
| | transfer | External | ✓ | - |
| | allowance | External | | - |
| | approve | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | | | | |
| IERC165 | Interface | | | |
| | supportsInterface | External | | - |
| | | | | |
| IControl | Interface | | | |
| | mintProgressCount | External | | - |

| | | | | |
|-----------------------|----------------------|---|---------|---|
| | redeemProgressCount | External | | - |
| | lastExecutedMint | External | | - |
| | lastExecutedRedeem | External | | - |
| | delegateApprove | External | ✓ | - |
| | getDailyInitialMints | External | | - |
| | getInitialMints | External | | - |
| | initialMint | External | Payable | - |
| | getCurrentPrice | External | | - |
| | execute | External | ✓ | - |
| | | | | |
| IAccessControl | Interface | | | |
| | hasRole | External | | - |
| | getRoleAdmin | External | | - |
| | grantRole | External | ✓ | - |
| | revokeRole | External | ✓ | - |
| | renounceRole | External | ✓ | - |
| | | | | |
| ERC20 | Implementation | Context, IERC20, IERC20Meta data | | |
| | | Public | ✓ | - |
| | name | Public | | - |
| | symbol | Public | | - |
| | decimals | Public | | - |

| | | | | |
|--|----------------------|----------|---|---|
| | totalSupply | Public | | - |
| | balanceOf | Public | | - |
| | transfer | Public | ✓ | - |
| | allowance | Public | | - |
| | approve | Public | ✓ | - |
| | transferFrom | Public | ✓ | - |
| | increaseAllowance | Public | ✓ | - |
| | decreaseAllowance | Public | ✓ | - |
| | _transfer | Internal | ✓ | |
| | _mint | Internal | ✓ | |
| | _burn | Internal | ✓ | |
| | _approve | Internal | ✓ | |
| | _spendAllowance | Internal | ✓ | |
| | _beforeTokenTransfer | Internal | ✓ | |
| | _afterTokenTransfer | Internal | ✓ | |

Inheritance Graph



Flow Graph



Summary

HYDT Stablecoin contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security. Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.



The Cyberscope team

<https://www.cyberscope.io>