audita

CygnusDAO

Initial Screening

Sept 4, 2023

Prepared for: CygnusDAO (cygnusdao.finance)

Presented By Audita Security (audita.io)

Document

Approved by: Reni Dimitrova and Jose Saez Lopez @ Audita.io
Protocol: CygnusDAO
Method: Manual Initial Review by Audita's Team
Client Website: https://www.cygnusdao.finance/

Initial Findings

To be revised, tested and expanded by Audita's experts upon commencement of Audit.

`Ambiguous Oracle Integration`
The system relies on oracle prices, they define the max liquidity users may have per their collateral. However, it is not possible to stop the oracle or activate a fallback mechanism in case it is compromised or manipulated. Prices from the oracle are not aggregated. This lack of aggregation may expose the system to flash loan attacks and manipulations if the prices aren't properly consolidated within the oracle. This may lead to token compromises.
Paths:
https://github.com/CygnusDAO/core/blob/main/contracts/cygnus-core/CygnusCollateralModel.sol#L120
https://github.com/CygnusDAO/core/blob/main/contracts/cygnus-core/CygnusTerminal.sol#L108

`Requirements Violation`
Borrow and repay are possible within the same transaction. It is stated in the comments that borrow and repay should not be possible in one transaction: "Check for borrow amount, if a repay transaction this should be 0, else reverts at the end.", "Avoid borrow and repay in the same transaction". However, the current verification does not accurately verify this scenario. In case the borrowAmount > repayAmount and repayAmount > 0, the function will not fail. Additionally, the _updateBorrow function is designed to process both the repay and borrowing in 1 call which contradicts the requirements.
Paths:
https://github.com/CygnusDAO/core/blob/main/contracts/cygnus-core/CygnusBorrow.sol#L190
https://github.com/CygnusDAO/core/blob/main/contracts/cygnus-core/CygnusBorrowModel.sol#L365

Centralization risk

Admin can change debtRatio, liquidationIncentive, liquidationFee which may lead to the liquidation of the position.
In case the mentioned parameters are changed, the previously safe positions may become liquidatable. Additionally, the users' borrow transactions may be front-runned with the changing of those parameters and users will create borrows when those parameters are different than they expected.
Paths:
https://github.com/CygnusDAO/core/blob/main/contracts/cygnus-core/CygnusCollateralControl.sol#L172,
https://github.com/CygnusDAO/core/blob/main/contracts/cygnus-core/CygnusCollateralControl.sol#L190,
https://github.com/CygnusDAO/core/blob/main/contracts/cygnus-core/CygnusCollateralControl.sol#L208,
https://github.com/CygnusDAO/core/blob/main/contracts/cygnus-core/CygnusCollateralModel.sol#L74

If rebase token used as an underlying token - can be stolen from the contracts

When redeeming the tokens, the shares amount are converted to assets amount using previously stored totalBalance value (totalAssets() function). Due to this, in case the underlying token is a rebase token, the actual totalBalance may differ which may lead to the manipulations and redeeming more or less tokens than expected.
Path:
https://github.com/CygnusDAO/core/blob/main/contracts/cygnus-core/CygnusTerminal.sol#L340

Denial of Service vulnerability

Looping through the unlimited number of orbiters.
The allOrbiters array is not limited, therefore, checkOrbitersPrivate function may fail when limiting it, which may lock initializeOrbiter function.
Path:
https://github.com/CygnusDAO/core/blob/main/contracts/cygnus-core/Hangar18.sol#L233

Unsafe rewarder integration may also lead to DoS

The rewarder gammaId is stated to type(uint256).max and then is updated with the correct value in the chargeVoid function. The function uses a loop to find the corresponding pool. According to this, in case the REWARDER.poolLength() is big enough to exceed the Gas limit, the function will fail and it would be impossible to set the pool.

Path:

https://github.com/CygnusDAO/core/blob/main/contracts/cygnus-core/CygnusCollateralVoid.sol#L275

Missing validation check if the amount transferred from user is bigger than required

Users may transfer more underlying tokens than required when repaying and liquidating, therefore, they may lose those tokens.

Paths:

https://github.com/CygnusDAO/core/blob/main/contracts/cygnus-core/CygnusBorrowModel.sol#L412

https://github.com/CygnusDAO/core/blob/main/contracts/cygnus-core/CygnusBorrow.sol#L227

Best Practice Violation: 2-step admin transferring is not properly implemented

According to best security practices, pending admin should accept the admin role transferring, however, it is not implemented. This may lead to transferring the admin role to the incorrect address.

Path:

https://github.com/CygnusDAO/core/blob/main/contracts/cygnus-core/Hangar18.sol#L578

Disclaimer

This initial screening makes no statements or warranties on the security of the code. This report should not be considered a sufficient assessment on the safety of the code, quality status, or any other contract statements. In order for contracts to be considered as safe as possible, the industry standard requires protocols to undergo a thorough security review.