ORAICHAIN Controller and Vault

Smart Contract Security Audit

Prepared by: **Halborn**
Date of Engagement: December 22, 2020
Visit: Halborn.com

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	12/22/2020	Gabi Urrutia
0.2	Document Edits	12/26/2020	Gabi Urrutia
1.0	Final Version	12/31/2020	Gabi Urrutia

CONTACT	COMPANY	EMAIL
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com

EXECUTIVE SUMMARY

1.1 INTRODUCTION

Oraichain engaged Halborn to conduct a security assessment on their Vault and Controller smart contracts beginning on December 22th, 2020 and ending December 31th, 2020. The security assessment was scoped to the contracts vault_v2.sol and controller_v2 and an audit of the security risk and implications regarding the changes introduced by the development team at Oraichain prior to its production release shortly following the assessments deadline.

Both smart contracts do not import any external libraries. Thus, the contract vault_v2.sol is made up of 19 contracts: Math, SafeMath, IERC20, Address, SafeERC20, Initializable, Context, ERC20, ERC20Detailed, IStrategy, IStrategyV2, IController, IVault, IUpgradeSource, Storage, GovernableInit, ControllableInit, VaultStorage and Vault. On the other hand, the contract controller_v2 is made up of 14 contracts: Address, SafeMath, IERC20, SafeERC20, IController, IStrategy, IVault, Storage, Governable, IRewardPool, IFeeRewardForwarder, IHardRewards, IApiConsumer and Controller. Therefore, the contract works by itself without importing any external contracts, increasing its security.

Overall, the smart contracts code does NOT contain any obvious exploitation vectors that Halborn was able to leverage within the timeframe of testing allotted. The most significant observations made in the security assessment is in regard to the use of multiples and floating pragmas and the use of deprecated OpenZeppelin libraries. It is important to lock the pragma and using the lastest versions OpenZeppelin libraries. In addition, note that pay attention to the latest attacks on farming platforms last October.

Halborn recommends performing further testing to validate extended safety and correctness in context to the whole set of contracts. External threats, such as economic attacks, oracle attacks, and inter-contract functions and calls should be validated for expected logic and state.

1.2 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the smart contract audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture, purpose, and use of Vault and Controller.
- Smart Contract manual code read and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual Assessment of use and safety for the critical solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Scanning of solidity files for vulnerabilities, security hotspots, or bugs. (MythX)
- Static Analysis of security for scoped contract and imported functions. (Slither)
- Smart Contract analysis and automatic exploitation (limited-time)
- Symbolic Execution / EVM bytecode security assessment (limited-time)

1.3 SCOPE

IN-SCOPE:

Code related to Vault_v2 and Controller_v2 smart contracts.

Specific commit of contract: commit

e097479e6417bd93e612d6db3dfe1edae7e76c43

OUT-OF-SCOPE:

Other smart contracts in the repository and economics attacks.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW
0	0	0	3

SECURITY ANALYSIS	RISK LEVEL
MULTIPLES AND FLOATING	Low
OUTDATED LIBRARIES	Low
USE OF TX.ORIGIN	Low
USE OF INLINE ASSEMBLY	Informational
POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational
THE LASTEST ECONOMIC ATTACK ON FARMING PLATFORM	Informational
STATIC ANALYSIS REPORT	Informational
AUTOMATED SECURITY SCAN	Informational

FINDINGS & TECH DETAILS

3.1 MULTIPLES AND FLOATING - LOW

Description:
In both contracts, many different pragmas are used instead of use only one (. The Solidity Compiler only use the pragma which the pragmas are not used. On the other hand, Vault and Controller contracts use floating pragmas ^0.5.0. and ^0.5.5. Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively. At the time of this audit, the current version is already at 0.7 The newer versions provide features that provide checks and accounting, as well as prevent insecure use of code.

Code Location:

Vault_v2.sol Line #7

```
7   pragma solidity ^0.5.0;
8
9   /**
10   * @dev Standard math utilities mis
11   */
```

Line #39 - pragma solidity ^0.5.0

Line #198 - pragma solidity ^0.5.0

Line #277 - pragma solidity ^0.5.5

Line #350 - pragma solidity ^0.5.0

Line #427 - pragma solidity >=0.4.24 <0.6.0

Line #492 - pragma solidity ^0.5.0

Line #524 - pragma solidity ^0.5.0

Line #759 - pragma solidity ^0.5.0

Line #807 - pragma solidity 0.5.16

Line #864 - pragma solidity 0.5.16

Line #898 - pragma solidity 0.5.16

Line #938 - pragma solidity 0.5.16

Line #948 - pragma solidity 0.5.16

Line #985 - pragma solidity 0.5.16

Line #1035 - pragma solidity 0.5.16

Line #1231 - pragma solidity 0.5.16

Controller_v2.sol Line #1

```
pragma solidity ^0.5.5;
UnitTest stub | dependencies | uml
library Address {

```

Line #54 - pragma solidity ^0.5.0

Line #213 - pragma solidity ^0.5.0

Line #292 - pragma solidity ^0.5.0

Line #369 - pragma solidity 0.5.16

Line #403 - pragma solidity 0.5.16

Line #484 - pragma solidity 0.5.16

Line #521 - pragma solidity 0.5.16

Line #587 - pragma solidity 0.5.16

Recommendation:

Consider using only one pragma in each smart contract and lock the pragma version to avoid vulnerabilities in the following compiler deployment.

3.2 OUTDATED LIBRARIES - LOW

Description:

OpenZeppelin is a set of testing Smart Contracts libraries to be reused. Using OpenZeppelin libraries, the risk of smart contracts is highly reduced. Otherwise, OpenZeppelin usually update the Smart Contracts templates to add new functionality or fix vulnerabilities found by the community. The versions of OpenZeppelin used in Vault_v2 and Controller_v2 are already deprecated. For instance, ERC20Detailed contract was removed and merged with ERC20 contract.

The different versions of pragma used in both smart contracts of OpenZeppelin and current libraries can be seen in the following table:

Controller_v2		
Library	**Used version**	**Current version**
Address	^0.5.5	>=0.6.2 <0.8.0
SafeMath	^0.5.5	>=0.6.0 <0.8.0
IERC20	^0.5.5	>=0.6.0 <0.8.0
SafeERC20	^0.5.5	>=0.6.0 <0.8.0
Vault_v2		

Library	Used version	Current version
Math	^0.5.0	>=0.6.0 <0.8.0
SafeMath	^0.5.0	>=0.6.0 <0.8.0
IERC20	^0.5.0	>=0.6.0 <0.8.0
Address	^0.5.0	>=0.6.0 <0.8.0
SafeERC20	^0.5.0	>=0.6.0 <0.8.0
Initializable	>=0.4.24 <0.6.0	>=0.4.24 <0.8.0
Context	^0.5.0	>=0.6.0 <0.8.0
ERC20	^0.5.0	>=0.6.0 <0.8.0
ERC20Detailed	^0.5.0	Removed and Merged with ERC20

Recommendation:

When possible, use the most updated OpenZeppelin libraries to avoid malfunctions or vulnerabilities already fixed in OpenZeppelin new versions.

3.3 USE OF TX.ORIGIN - LOW

Description:

"tx.origin" is useful only in very exceptional cases. If it is use for authentication, then it makes no impact, because any contract you call can act on your behalf. So it is recommended to Never use tx.origin for authorization.

Here in defense() function of vault_v2.sol contract which is callable from external has this require() condition which should be fix to filter out non required address to call this method.

Code Location:

Vault_v2.sol Line #1270

```
1268    function defense() private {
1269        require(
1270            (msg.sender == tx.origin) || // If it is a normal
1271            // then the requirement will pass
1272            !IController(controller()).greyList(msg.sender), /
1273            "V:4"  // make sure that it is not on our greyList
1274        );
1275    }
```

Recommendation:

When possible, do not tx.origin for authentication except for exceptional cases. Furthermore, tx.origin will be probably deprecated in the following versions of Solidity.

3.4 USE OF INLINE ASSEMBLY - INFORMATIONAL

Description:

a low level. This discards several important safety features in Solidity.

Code Location:

Vault_v2.sol Line #307

```
307         assembly {codehash := extcodehash(account)}
308         return (codehash != accountHash && codehash != 0x0);
309     }
```

Vault_v2.sol Line #481

```
481         assembly {cs := extcodesize
482         (address) }
483         return cs == 0;
484     }
```

Vault_v2.sol Line #1009

```
1009        assembly {
1010            sstore(slot, newStorage)
1011        }
```

Vault_v2.sol Line #1023

```
1023        assembly {
1024            str := sload(slot)
1025        }
```

Vault_v2.sol Line #1199

```
1199        assembly {
1200            sstore(slot, _value)
```

Vault_v2.sol Line #1214

```
1214        assembly {
1215            str := sload(slot)
1216        }
1217    }
1218
```

Vault_v2.sol Line #1221

```
1219    function getUint256(bytes32 slot) private view returns (uint256 str) {
1220        // solhint-disable-next-line no-inline-assembly
1221        assembly {
1222            str := sload(slot)
1223        }
1224    }
```

Controller_v2.sol Line #11

```
11          assembly {codehash := extcodehash(account)}
12          return (codehash != accountHash && codehash != 0x0);
13      }
```

Recommendation:

When possible, do not use inline assembly because it is a manner to access to the EVM (Ethereum Virtual Machine) at a low level. An attacker could bypass many important safety features of Solidity.

3.5 POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In public functions, array arguments are immediately copied array to memory, while external functions can read directly from calldata. Reading calldata is cheaper than memory allocation. Public functions need to write the arguments to memory because public functions may be memory. Thus, function expects its arguments being in memory when the compiler generates the code for an internal function. In Vault and Controlles contracts, many functions are never directly called by another function in the same contract.

Code Location:

Controller_v2.sol Line #514

```
514    function isController(address account) public view returns (bool) {
515        return account == controller;
516    }
```

Controller_v2.sol Line #505

```
505    function setController(address _controller) public onlyGovernance {
506        require(_controller != address(0), "new controller shouldn't be empty");
507        controller = _controller;
508    }
509
```

Controller_v2.sol Line #500

```
500     function setGovernance(address _governance) public onlyGovernance {
501         require(_governance != address(0), "new governance shouldn't be empty");
502         governance = _governance;
503     }
```

Controller_v2.sol Line #538

```
538     function setStorage(address _store) public onlyGovernance {
539         require(_store != address(0), "new storage shouldn't be empty");
540         store = Storage(_store);
541     }
```

Vault_v2.sol Line #1014

```
1014    function setStorage(address _store) public {
1015        require(Storage(_storage()).isGovernance(msg.sender), "Gvn:2");
1016        require(_store != address(0), "Gvn:3");
1017        _setStorage(_store);
1018    }
```

Vault_v2.sol Line #774

```
774     function name() public view returns (string memory) {
775         return _name;
776     }
```

Vault_v2.sol Line #658

```
658     function decreaseAllowance(address spender, uint256 subtractedValue) public returns (bool) {
659         _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20:2"));
660         return true;
661     }
```

Vault_v2.sol Line #969

```
969     function setController(address _controller) public onlyGovernance {
970         require(_controller != address(0), "new controller shouldn't be empty");
971         controller = _controller;
972     }
```

Vault_v2.sol Line #1435

```
1435    function depositFor(uint256 amount, address holder) public {
1436        defense();
1437        _deposit(amount, msg.sender, holder);
1438    }
```

Vault_v2.sol Line #621

```
621     function transferFrom(address sender, address recipient, uint256 amount) public returns (bool) {
622         _transfer(sender, recipient, amount);
623         _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20:1"));
624         return true;
625     }
```

Vault_v2.sol Line #639

```
639     function increaseAllowance(address spender, uint256 addedValue) public returns (bool) {
640         _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
641         return true;
642     }
```

Vault_v2.sol Line #1390

```
1390    function rebalance() public {
1391        onlyControllerOrGovernance();
1392        withdrawAll();
1393        invest();
1394    }
```

Vault_v2.sol Line #604

```
604     function approve(address spender, uint256 amount) public returns (bool) {
605         _approve(_msgSender(), spender, amount);
606         return true;
607     }
```

Vault_v2.sol Line #585

```
585     function transfer(address recipient, uint256 amount) public returns (bool) {
586         _transfer(_msgSender(), recipient, amount);
587         return true;
588     }
```

Vault_v2.sol Line #782

```
782     function symbol() public view returns (string memory) {
783         return _symbol;
784     }
```

Vault_v2.sol Line #964

```
964     function setGovernance(address _governance) public onlyGovernance {
965         require(_governance != address(0), "new governance shouldn't be empty");
966         governance = _governance;
967     }
```

Recommendation:

Consider as much as possible declaring external variables instead of public variables. As for best practices, you should use external if you expect that the function will only ever be called externally and use public if you need to call the function internally. In that case, both functions are not called by another function in the same contract, so marking both function as external can save gas.

3.6 THE LASTEST ECONOMIC ATTACK ON FARMING PLATFORMS - INFORMATIONAL

Description:

Impermanent Loss, arbitrage and slippage are market effects which affect the assets inside pools. The assets, such as USDT and USDC, inside the vaults are located into shared pools.

On October 26, an attacker stole funds from the USDT and USDC vaults of Harvest Finance. The attacker repeatedly exploited an arbitrage and impermanent loss that influences the value of individual assets inside the pool.

The value of asset invested are calculated in real -time. This value is used by the vaults to calculate the number of shares to be issued to the user depositing the funds. In addition, the value of the assets was used by the attacker when funds are removed from the vaults and it calculates how much payout the user will be receive.

Reference: https://medium.com/harvest-finance/harvest-flashloan-economic-attack-post-mortem-3cf900d65217

3.7 STATIC ANALYSIS REPORT - INFORMATIONAL

Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped contract. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their ABI and binary formats, Slither was run on Controller and Vault contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire codebase.

Results:

Vault_v2

```
ziion@eltitourruts-virtual-machine:~/Desktop/yai-protocol-feat-orai-bridge$ slither contracts/vaults/vault_v2.sol
Compilation warnings/errors on contracts/vaults/vault_v2.sol:
contracts/vaults/vault_v2.sol:994:5: Warning: Function state mutability can be restricted to view
    function onlyGovernance() internal {
    ^ (Relevant source part starts here and spans across multiple lines).
contracts/vaults/vault_v2.sol:1048:5: Warning: Function state mutability can be restricted to view
    function onlyControllerOrGovernance() public {
    ^ (Relevant source part starts here and spans across multiple lines).
contracts/vaults/vault_v2.sol:1263:5: Warning: Function state mutability can be restricted to view
    function whenStrategyDefined() private {
    ^ (Relevant source part starts here and spans across multiple lines).
contracts/vaults/vault_v2.sol:1268:5: Warning: Function state mutability can be restricted to view
    function defense() private {
    ^ (Relevant source part starts here and spans across multiple lines).

INFO:Detectors:
ERC20.______gap (contracts/vaults/vault_v2.sol#754) shadows:
        - Initializable.______gap (contracts/vaults/vault_v2.sol#487)
ERC20Detailed.______gap (contracts/vaults/vault_v2.sol#802) shadows:
        - Initializable.______gap (contracts/vaults/vault_v2.sol#487)
VaultStorage.______gap (contracts/vaults/vault_v2.sol#1226) shadows:
        - Initializable.______gap (contracts/vaults/vault_v2.sol#487)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
INFO:Detectors:
Vault.getPricePerFullShare() (contracts/vaults/vault_v2.sol#1331-1335) uses a dangerous strict equality:
        - totalSupply() == 0 (contracts/vaults/vault_v2.sol#1332-1334)
Vault.underlyingBalanceWithInvestmentForHolder(address) (contracts/vaults/vault_v2.sol#1339-1344) uses a dangerous strict equality:
        - totalSupply() == 0 (contracts/vaults/vault_v2.sol#1340)
Vault.withdraw(uint256) (contracts/vaults/vault_v2.sol#1446-1492) uses a dangerous strict equality:
        - numberOfShares == totalSupply (contracts/vaults/vault_v2.sol#1460)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
```

```
INFO:Detectors:
ControllableInit.initialize(address)._storage (contracts/vaults/vault_v2.sol#1044) shadows:
        - GovernableInit._storage() (contracts/vaults/vault_v2.sol#1020-1026) (function)
Vault.initializeVault(address,address,uint256,uint256)._storage (contracts/vaults/vault_v2.sol#1244) shadows:
        - GovernableInit._storage() (contracts/vaults/vault_v2.sol#1020-1026) (function)
Vault.withdraw(uint256).totalSupply (contracts/vaults/vault_v2.sol#1449) shadows:
        - ERC20.totalSupply() (contracts/vaults/vault_v2.sol#566-568) (function)
        - IERC20.totalSupply() (contracts/vaults/vault_v2.sol#208) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Vault.doHardWork() (contracts/vaults/vault_v2.sol#1286-1311) uses timestamp for comparisons
        Dangerous comparisons:
        - futureStrategy() != address(0) && futureStrategy() != strategy() && strategyUpdateTime() <= block.timestamp (contracts/vaults/vault_v2.sol#1289)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (contracts/vaults/vault_v2.sol#300-309) uses assembly
        - INLINE ASM (contracts/vaults/vault_v2.sol#307)
Initializable.isConstructor() (contracts/vaults/vault_v2.sol#474-484) uses assembly
        - INLINE ASM (contracts/vaults/vault_v2.sol#481-482)
GovernableInit._setStorage(address) (contracts/vaults/vault_v2.sol#1006-1012) uses assembly
        - INLINE ASM (contracts/vaults/vault_v2.sol#1009-1011)
GovernableInit._storage() (contracts/vaults/vault_v2.sol#1020-1026) uses assembly
        - INLINE ASM (contracts/vaults/vault_v2.sol#1023-1025)
VaultStorage.setAddress(bytes32,address) (contracts/vaults/vault_v2.sol#1190-1195) uses assembly
        - INLINE ASM (contracts/vaults/vault_v2.sol#1192-1194)
VaultStorage.setUint256(bytes32,uint256) (contracts/vaults/vault_v2.sol#1197-1202) uses assembly
        - INLINE ASM (contracts/vaults/vault_v2.sol#1199-1201)
VaultStorage.getAddress(bytes32) (contracts/vaults/vault_v2.sol#1212-1217) uses assembly
        - INLINE ASM (contracts/vaults/vault_v2.sol#1214-1216)
VaultStorage.getUint256(bytes32) (contracts/vaults/vault_v2.sol#1219-1224) uses assembly
        - INLINE ASM (contracts/vaults/vault_v2.sol#1221-1223)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
Different versions of Solidity is used in :
        - Version used: ['0.5.16', '>=0.4.24<0.6.0', '^0.5.0', '^0.5.16', '^0.5.5']
        - ^0.5.16 (contracts/vaults/vault_v2.sol#7)
        - ^0.5.0 (contracts/vaults/vault_v2.sol#39)
        - ^0.5.0 (contracts/vaults/vault_v2.sol#198)
        - ^0.5.5 (contracts/vaults/vault_v2.sol#277)
        - ^0.5.0 (contracts/vaults/vault_v2.sol#350)
        - >=0.4.24<0.6.0 (contracts/vaults/vault_v2.sol#427)
        - ^0.5.0 (contracts/vaults/vault_v2.sol#492)
        - ^0.5.0 (contracts/vaults/vault_v2.sol#524)
        - ^0.5.0 (contracts/vaults/vault_v2.sol#759)
        - 0.5.16 (contracts/vaults/vault_v2.sol#807)
        - 0.5.16 (contracts/vaults/vault_v2.sol#864)
        - 0.5.16 (contracts/vaults/vault_v2.sol#898)
        - 0.5.16 (contracts/vaults/vault_v2.sol#938)
        - 0.5.16 (contracts/vaults/vault_v2.sol#948)
        - 0.5.16 (contracts/vaults/vault_v2.sol#985)
        - 0.5.16 (contracts/vaults/vault_v2.sol#1035)
        - 0.5.16 (contracts/vaults/vault_v2.sol#1231)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version^0.5.0 (contracts/vaults/vault_v2.sol#39) allows old versions
Pragma version^0.5.0 (contracts/vaults/vault_v2.sol#198) allows old versions
Pragma version^0.5.5 (contracts/vaults/vault_v2.sol#277) is known to contain severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
Pragma version^0.5.0 (contracts/vaults/vault_v2.sol#350) allows old versions
Pragma version>=0.4.24<0.6.0 (contracts/vaults/vault_v2.sol#427) allows old versions
Pragma version^0.5.0 (contracts/vaults/vault_v2.sol#492) allows old versions
Pragma version^0.5.0 (contracts/vaults/vault_v2.sol#524) allows old versions
Pragma version^0.5.0 (contracts/vaults/vault_v2.sol#759) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (contracts/vaults/vault_v2.sol#339-345):
        - (success) = recipient.call.value(amount)() (contracts/vaults/vault_v2.sol#343)
Low level call in SafeERC20.callOptionalReturn(IERC20,bytes) (contracts/vaults/vault_v2.sol#403-422):
        - (success,returndata) = address(token).call(data) (contracts/vaults/vault_v2.sol#415)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Variable Initializable.______gap (contracts/vaults/vault_v2.sol#487) is not in mixedCase
Variable ERC20.______gap (contracts/vaults/vault_v2.sol#754) is not in mixedCase
Variable ERC20Detailed.______gap (contracts/vaults/vault_v2.sol#802) is not in mixedCase
Parameter Storage.setGovernance(address)._governance (contracts/vaults/vault_v2.sol#964) is not in mixedCase
Parameter Storage.setController(address)._controller (contracts/vaults/vault_v2.sol#969) is not in mixedCase
Parameter GovernableInit.initialize(address)._store (contracts/vaults/vault_v2.sol#1002) is not in mixedCase
Parameter GovernableInit.setStorage(address)._store (contracts/vaults/vault_v2.sol#1014) is not in mixedCase
Parameter ControllableInit.initialize(address)._storage (contracts/vaults/vault_v2.sol#1044) is not in mixedCase
Parameter VaultStorage.initialize(address,uint256,uint256,uint256)._underlying (contracts/vaults/vault_v2.sol#1095) is not in mixedCase
Parameter VaultStorage.initialize(address,uint256,uint256,uint256)._toInvestNumerator (contracts/vaults/vault_v2.sol#1096) is not in mixedCase
Parameter VaultStorage.initialize(address,uint256,uint256,uint256)._toInvestDenominator (contracts/vaults/vault_v2.sol#1097) is not in mixedCase
Parameter VaultStorage.initialize(address,uint256,uint256,uint256)._underlyingUnit (contracts/vaults/vault_v2.sol#1098) is not in mixedCase
Function VaultStorage._sharePriceCheckpoint() (contracts/vaults/vault_v2.sol#1138-1140) is not in mixedCase
Function VaultStorage._allowSharePriceDecrease() (contracts/vaults/vault_v2.sol#1146-1148) is not in mixedCase
Parameter VaultStorage.setBoolean(bytes32,bool)._value (contracts/vaults/vault_v2.sol#1204) is not in mixedCase
Variable VaultStorage.______gap (contracts/vaults/vault_v2.sol#1226) is not in mixedCase
Parameter Vault.initializeVault(address,address,uint256,uint256)._storage (contracts/vaults/vault_v2.sol#1244) is not in mixedCase
Parameter Vault.initializeVault(address,address,uint256,uint256)._underlying (contracts/vaults/vault_v2.sol#1245) is not in mixedCase
Parameter Vault.initializeVault(address,address,uint256,uint256)._toInvestNumerator (contracts/vaults/vault_v2.sol#1246) is not in mixedCase
Parameter Vault.initializeVault(address,address,uint256,uint256)._toInvestDenominator (contracts/vaults/vault_v2.sol#1247) is not in mixedCase
Parameter Vault.announceStrategyUpdate(address,uint256)._strategy (contracts/vaults/vault_v2.sol#1348) is not in mixedCase
Parameter Vault.setStrategy(address)._strategy (contracts/vaults/vault_v2.sol#1367) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformity-to-solidity-naming-conventions
INFO:Detectors:
VaultStorage.______gap (contracts/vaults/vault_v2.sol#1226) is never used in Vault (contracts/vaults/vault_v2.sol#1234-1509)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
ERC20Detailed._decimals (contracts/vaults/vault_v2.sol#769) should be constant
ERC20Detailed._name (contracts/vaults/vault_v2.sol#767) should be constant
ERC20Detailed._symbol (contracts/vaults/vault_v2.sol#768) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

```
INFO:Detectors:
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (contracts/vaults/vault_v2.sol#585-588)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (contracts/vaults/vault_v2.sol#593-595)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (contracts/vaults/vault_v2.sol#604-607)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (contracts/vaults/vault_v2.sol#621-625)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (contracts/vaults/vault_v2.sol#639-642)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (contracts/vaults/vault_v2.sol#658-661)
name() should be declared external:
        - ERC20Detailed.name() (contracts/vaults/vault_v2.sol#774-776)
symbol() should be declared external:
        - ERC20Detailed.symbol() (contracts/vaults/vault_v2.sol#782-784)
decimals() should be declared external:
        - ERC20Detailed.decimals() (contracts/vaults/vault_v2.sol#798-800)
setGovernance(address) should be declared external:
        - Storage.setGovernance(address) (contracts/vaults/vault_v2.sol#964-967)
setController(address) should be declared external:
        - Storage.setController(address) (contracts/vaults/vault_v2.sol#969-972)
isController(address) should be declared external:
        - Storage.isController(address) (contracts/vaults/vault_v2.sol#978-980)
setStorage(address) should be declared external:
        - GovernableInit.setStorage(address) (contracts/vaults/vault_v2.sol#1014-1018)
governance() should be declared external:
        - GovernableInit.governance() (contracts/vaults/vault_v2.sol#1028-1030)
doHardWork() should be declared external:
        - Vault.doHardWork() (contracts/vaults/vault_v2.sol#1286-1311)
rebalance() should be declared external:
        - Vault.rebalance() (contracts/vaults/vault_v2.sol#1390-1394)
depositFor(uint256,address) should be declared external:
        - Vault.depositFor(uint256,address) (contracts/vaults/vault_v2.sol#1435-1438)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:contracts/vaults/vault_v2.sol analyzed (19 contracts with 46 detectors), 72 result(s) found
```

Results:

Controller_v2

```
INFO:Detectors:
Reentrancy in Controller.requestFutureStrategy(address,address) (contracts/controller/controller_v2.sol#742-755):
        External calls:
        - IApiConsumer(_requestConsumer).requestData(data,this.addFutureStrategyAndTime.selector,address(this)) (contracts/controller/controller_v2.sol#748-752)
        State variables written after the call(s):
        - isRequestFutureVault[_vault] = true (contracts/controller/controller_v2.sol#753)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
Controller.requestFutureStrategy(address,address) (contracts/controller/controller_v2.sol#742-755) ignores return value by IApiConsumer(_requestConsumer).requestData(data,this.addFutu
reStrategyAndTime.selector,address(this)) (contracts/controller/controller_v2.sol#748-752)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in Controller.addFutureStrategyAndTime(bytes) (contracts/controller/controller_v2.sol#731-740):
        External calls:
        - IVault(_vault).announceStrategyUpdate(_strategy,time) (contracts/controller/controller_v2.sol#737)
        State variables written after the call(s):
        - isRequestFutureVault[_vault] = false (contracts/controller/controller_v2.sol#738)
Reentrancy in Controller.cancelRequestFutureStragety(address) (contracts/controller/controller_v2.sol#757-762):
        External calls:
        - IVault(_vault).finalizeStrategyUpdate() (contracts/controller/controller_v2.sol#760)
        State variables written after the call(s):
        - isRequestFutureVault[_vault] = false (contracts/controller/controller_v2.sol#761)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Controller.doHardWork(address) (contracts/controller/controller_v2.sol#775-789):
        External calls:
        - IVault(_vault).doHardWork() (contracts/controller/controller_v2.sol#777)
        - hardRewards.rewardMe(msg.sender,_vault) (contracts/controller/controller_v2.sol#780)
        Event emitted after the call(s):
        - SharePriceChangeLog(_vault,IVault(_vault).strategy(),oldSharePrice,IVault(_vault).getPricePerFullShare(),block.timestamp) (contracts/controller/controller_v2.sol#782-788)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (contracts/controller/controller_v2.sol#24-33) uses assembly
        - INLINE ASM (contracts/controller/controller_v2.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
Different versions of Solidity is used in :
        - Version used: ['0.5.16', '^0.5.0']
        - 0.5.16 (contracts/controller/controller_v2.sol#1)
        - ^0.5.0 (contracts/controller/controller_v2.sol#74)
        - ^0.5.0 (contracts/controller/controller_v2.sol#233)
        - ^0.5.0 (contracts/controller/controller_v2.sol#312)
        - 0.5.16 (contracts/controller/controller_v2.sol#389)
        - 0.5.16 (contracts/controller/controller_v2.sol#423)
        - 0.5.16 (contracts/controller/controller_v2.sol#454)
        - 0.5.16 (contracts/controller/controller_v2.sol#504)
        - 0.5.16 (contracts/controller/controller_v2.sol#541)
        - 0.5.16 (contracts/controller/controller_v2.sol#607)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version^0.5.0 (contracts/controller/controller_v2.sol#74) allows old versions
Pragma version^0.5.0 (contracts/controller/controller_v2.sol#233) allows old versions
Pragma version^0.5.0 (contracts/controller/controller_v2.sol#312) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (contracts/controller/controller_v2.sol#63-69):
        - (success) = recipient.call.value(amount)() (contracts/controller/controller_v2.sol#67)
Low level call in SafeERC20.callOptionalReturn(IERC20,bytes) (contracts/controller/controller_v2.sol#365-384):
        - (success,returndata) = address(token).call(data) (contracts/controller/controller_v2.sol#377)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Parameter Storage.setGovernance(address)._governance (contracts/controller/controller_v2.sol#520) is not in mixedCase
Parameter Storage.setController(address)._controller (contracts/controller/controller_v2.sol#525) is not in mixedCase
Parameter Governable.setStorage(address)._store (contracts/controller/controller_v2.sol#558) is not in mixedCase
Parameter Controller.setOracleAddress(address)._address (contracts/controller/controller_v2.sol#695) is not in mixedCase
Parameter Controller.addHardWorker(address)._worker (contracts/controller/controller_v2.sol#699) is not in mixedCase
Parameter Controller.removeHardWorker(address)._worker (contracts/controller/controller_v2.sol#704) is not in mixedCase
Parameter Controller.hasVault(address)._vault (contracts/controller/controller_v2.sol#709) is not in mixedCase
Parameter Controller.addToGreyList(address)._target (contracts/controller/controller_v2.sol#714) is not in mixedCase
Parameter Controller.removeFromGreyList(address)._target (contracts/controller/controller_v2.sol#718) is not in mixedCase
Parameter Controller.setFeeRewardForwarder(address)._feeRewardForwarder (contracts/controller/controller_v2.sol#722) is not in mixedCase
Parameter Controller.requestFutureStrategy(address,address)._vault (contracts/controller/controller_v2.sol#742) is not in mixedCase
Parameter Controller.requestFutureStrategy(address,address)._requestConsumer (contracts/controller/controller_v2.sol#742) is not in mixedCase
Parameter Controller.cancelRequestFutureStragety(address)._vault (contracts/controller/controller_v2.sol#757) is not in mixedCase
Parameter Controller.addVaultAndStrategy(address,address)._vault (contracts/controller/controller_v2.sol#764) is not in mixedCase
Parameter Controller.addVaultAndStrategy(address,address)._strategy (contracts/controller/controller_v2.sol#764) is not in mixedCase
Parameter Controller.doHardWork(address)._vault (contracts/controller/controller_v2.sol#775) is not in mixedCase
Parameter Controller.rebalance(address)._vault (contracts/controller/controller_v2.sol#791) is not in mixedCase
Parameter Controller.setHardRewards(address)._hardRewards (contracts/controller/controller_v2.sol#795) is not in mixedCase
Parameter Controller.salvage(address,uint256)._token (contracts/controller/controller_v2.sol#800) is not in mixedCase
Parameter Controller.salvage(address,uint256)._amount (contracts/controller/controller_v2.sol#800) is not in mixedCase
Constant Controller.profitSharingNumerator (contracts/controller/controller_v2.sol#665) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Controller.profitSharingDenominator (contracts/controller/controller_v2.sol#666) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformity-to-solidity-naming-conventions
INFO:Detectors:
setGovernance(address) should be declared external:
        - Storage.setGovernance(address) (contracts/controller/controller_v2.sol#520-523)
setController(address) should be declared external:
        - Storage.setController(address) (contracts/controller/controller_v2.sol#525-528)
isController(address) should be declared external:
        - Storage.isController(address) (contracts/controller/controller_v2.sol#534-536)
setStorage(address) should be declared external:
        - Governable.setStorage(address) (contracts/controller/controller_v2.sol#558-561)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:contracts/controller/controller_v2.sol analyzed (14 contracts with 46 detectors), 38 result(s) found
```

3.8 AUTOMATED SECURITY SCAN - INFORMATIONAL

Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruit on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the testers machine and sent the compiled results to the analyzers to locate any vulnerabilities. Security Detections are only in scope, and the analysis was pointed towards issues with vault and controller.

Results

Vault_v2

MythX detected 0 High findings, 17 Medium, and 19 Low.

DETECTED ISSUES

0 High | 17 Medium | 19 Low

ID	SEVERITY	NAME	FILE	LOCATION
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 585 C: 4
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 593 C: 4
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 604 C: 4
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 621 C: 4
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 639 C: 4
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 658 C: 4
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 774 C: 4
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 782 C: 4
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 798 C: 4
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 964 C: 4
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 969 C: 4
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 978 C: 4
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 1014 C: 4
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 1286 C: 4
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 1390 C: 4
SWC-000	Medium	Function could be marked as external.	vault_v2.sol	L: 1435 C: 4
SWC-103	Low	A floating pragma is set.	vault_v2.sol	L: 7 C: 0
SWC-103	Low	A floating pragma is set.	vault_v2.sol	L: 39 C: 0
SWC-103	Low	A floating pragma is set.	vault_v2.sol	L: 198 C: 0
SWC-103	Low	A floating pragma is set.	vault_v2.sol	L: 277 C: 0
SWC-103	Low	A floating pragma is set.	vault_v2.sol	L: 350 C: 0
SWC-103	Low	A floating pragma is set.	vault_v2.sol	L: 427 C: 0
SWC-103	Low	A floating pragma is set.	vault_v2.sol	L: 492 C: 0
SWC-103	Low	A floating pragma is set.	vault_v2.sol	L: 524 C: 0
SWC-103	Low	A floating pragma is set.	vault_v2.sol	L: 759 C: 0
SWC-115	Low	Use of "tx.origin" as a part of authorization control.	vault_v2.sol	L: 1270 C: 27
SWC-115	Low	Use of tx.origin as a part of authorization control.	vault_v2.sol	L: 1270 C: 12
SWC-115	Low	Use of tx.origin as a part of authorization control.	vault_v2.sol	L: 1269 C: 8
SWC-115	Low	Use of tx.origin as a part of authorization control.	vault_v2.sol	L: 736 C: 8
SWC-115	Low	Use of tx.origin as a part of authorization control.	vault_v2.sol	L: 1496 C: 8

Controller_v2

MythX detected 0 High findings, 0 Medium, and 0 Low.

FINDINGS & TECH DETAILS

Main Source File	Contracts/Controller/Controller_v2.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	0

THANK YOU FOR CHOOSING

HALBORN