// HALBORN

Ocean Protocol H2O System and Action

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **January 20th, 2022 - February 14th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	02/8/2022	Juned Ansari
0.2	Document Update	02/9/2022	Juned Ansari
0.3	Document Update	02/12/2022	Juned Ansari
0.4	Document Update	02/12/2022	Gokberk Gulgun
0.5	Document Update	02/15/2022	Juned Ansari
0.6	Document Update	02/15/2022	Gokberk Gulgun
0.7	Draft Review	02/15/2022	Gabi Urrutia
1.0	Remediation Plan	03/07/2022	Juned Ansari
1.1	Remediation Plan Review	03/07/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	gokberk.gulgun@halborn.com
Juned Ansari	Halborn	Juned.Ansari@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

engaged Halborn to conduct a security audit on their smart contracts beginning on January 20th, 2022 and ending on February 14th, 2022 . The security assessment was scoped to the smart contracts provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided four weeks for the engagement and assigned a full-time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that reflexer-lab and forked smart contract functions operate as intended
- Identify potential security issues with the smart contracts

In summary, Halborn identified some security risks that were mostly addressed by the team.

1.3 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 this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the H2O contract solidity code 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 and purpose.
- Smart contract manual code review 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.
- Manual testing by custom scripts.
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE :
The security assessment was scoped to the following smart contract of reflexer-lab and stablecoin-research:

Core base contracts

- ds-proxy
- ds-roles
- ds-token
- ds-value
- erc20

Core system contracts

- geb
- geb-fsm
- geb-fsm-governance-interface
- geb-proxy-registry
- geb-safe-manager
- geb-uniswap-median
- geb-chainlink-median
- geb-rrfm-calculators
- geb-rrfm-rate-setter
- geb-protocol-token-authority

Actions contracts

- h2o-governance-actions
- geb-proxy-actions
- geb-deploy-pause-proxy-actions
- geb-pause-schedule-proxy-actions

OUT-OF-SCOPE : External libraries, utils, test, mock and economics attacks

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	5	10	8

IMPACT

LIKELIHOOD

(HAL-01)				
	(HAL-04)			
(HAL-07) (HAL-09) (HAL-14)	(HAL-06) (HAL-08) (HAL-10) (HAL-11) (HAL-12) (HAL-15)	(HAL-02) (HAL-03) (HAL-05)		
		(HAL-13)		
(HAL-16) (HAL-17) (HAL-18) (HAL-19) (HAL-20) (HAL-21) (HAL-22) (HAL-23)				

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
AUTHORIZED ACCOUNT CAN REMOVE ITSELF AND ALL OTHER AUTHORIZE ACCOUNT	Medium	RISK ACCEPTED
USE LATESTROUNDDATA INSTEAD OF LATESTANSWER TO RUN MORE VALIDATIONS	Medium	SOLVED - 02/20/2022
UNCHECKED TRANSFER	Medium	RISK ACCEPTED
IMPROPER ACCESS CONTROL POLICY	Medium	FUTURE RELEASE UPDATE
MISSING RE-ENTRANCY PROTECTION	Medium	SOLVED - 02/25/2022
AUTHORIZE ACCOUNT CAN SET INVALID BASE AND MAX REWARDS	Low	RISK ACCEPTED
AUTHORIZE ACCOUNT CAN INCREASE MAX REWARD DELAY TO MAX INT VALUE	Low	RISK ACCEPTED
THE CONTRACT FUNCTION SHOULD APPROVE(0) FIRST	Low	RISK ACCEPTED
ERC20 APPROVE METHOD MISSING RETURN VALUE CHECK	Low	RISK ACCEPTED
MISSING SAFE ENGINE ADDRESS VALIDATION	Low	SOLVED - 02/20/2022
MISSING FACTORY ADDRESS VALIDATION	Low	SOLVED - 03/05/2022
EXTERNAL FUNCTION CALLS WITHIN LOOP	Low	RISK ACCEPTED
IGNORE RETURN VALUES	Low	RISK ACCEPTED
MISSING ZERO-ADDRESS CHECK	Low	RISK ACCEPTED
USE OF DEPRECATED NOW FUNCTION	Low	SOLVED - 02/25/2022
AUCTION CAN BE EXTENDED INDEFINITELY	Informational	ACKNOWLEDGED
SELLER CAN MONOPOLIZE THE AUCTION BID PRICE	Informational	ACKNOWLEDGED

POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	ACKNOWLEDGED
USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS	Informational	ACKNOWLEDGED
CACHE ARRAY LENGTH IN FOR LOOPS CAN SAVE GAS	Informational	ACKNOWLEDGED
UPGRADE AT LEAST PRAGMA 0.8.4	Informational	ACKNOWLEDGED
CONSTANT WITH EXPONENTIATION CAN BE IMPROVED	Informational	SOLVED - 03/05/2022
MISSING SUCCESS CHECK ON LOW LEVEL CALLS	Informational	SOLVED - 02/20/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) AUTHORIZE CAN REMOVE HIMSELF AND ALL OTHER AUTHORIZE ACCOUNT - MEDIUM

Description:

The authorizedAccounts of the contract are the accounts that control all privileged functions. In most of the in-scope reflexer-labs contract, authorizedAccounts can perform privileged actions such as addAuthorization, RemoveAuthorization, ModifyParameters, start , stop (OSM, DSM) etc. . . , the RemoveAuthorization function is used to remove being an authorized account. It is observed that authorizedAccounts can remove himself via the RemoveAuthorization function. As a result, an authorized account could remove authorization before assigning new authorization to another account, which may lead to the contract having no authorized account, eliminating the ability to call privileged functions.

Code Location:

- Missing validation on the number of authorized accounts left in the system after removing an authorized account
- Missing checks to make sure that the addAuthorization function is called before RemoveAuthorization.
- No centralized super owner, any authorized account can remove all known account without a secondary confirmation.

Listing 1: In-Scope Contracts

```
    function addAuthorization(address account) virtual external
↳ isAuthorized {
        authorizedAccounts[account] = 1;
        emit AddAuthorization(account);
    }
    /**
     * @notice Remove auth from an account
```

```
 7       * @param account Account to remove auth from
 8       */
 9      function removeAuthorization(address account) virtual external
↳  isAuthorized {
10          authorizedAccounts[account] = 0;
11          emit RemoveAuthorization(account);
12      }
13      /**
14      * @notice Checks whether msg.sender can call an authed
↳ function
15      **/
16      modifier isAuthorized {
17          require(authorizedAccounts[msg.sender] == 1, "DSM/account-
↳ not-authorized");
18          _;
19      }
```

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

It is recommended that the contract authorized accounts cannot call removeAuthorization without transferring the authorization to another address before by calling addAuthorization. In addition, if a multi-signature wallet is used, calling removeAuthorization function should be confirmed by two or more authorized account. Finally, consider adding validation on the number of authorized accounts left in the system after removing an authorized account, make sure that the authorized accounts count is always equal to one or more than one.

Remediation Plan:

RISK ACCEPTED: The team acknowledged the above issue and claimed that it is the intended behavior of the H2O system and will add the note clarifying the intended functionalities in the README and GitBook documentation.

3.2 (HAL-02) USE LATESTROUNDDATA INSTEAD OF LATESTANSWER TO RUN MORE VALIDATIONS - MEDIUM

Description:

Chainlink contract are calling latestAnswer to get the asset prices. The latestAnswer is deprecated. Freshness of the returned price should be checked, since it affects an account's health (and therefore liquidations). Stale prices that do not reflect the current market price anymore could be used, which would influence the liquidation pricing. This method will return the last value, but you won't be able to check if the data is fresh. On the other hand, calling the method latestRoundData allow you to run some extra validations. Stale prices can put funds in a risk. According to Chainlink's documentation, This function does not error if no answer has been reached but returns 0, causing an incorrect price fed to the Price oracle. (https://docs.chain.link/docs/historical-price-data/#solidity). Furthermore, latestAnswer is deprecated. (https://docs.chain.link/docs/price-feeds-api-reference/))

Code Location:

Chainlink Integration

Listing 2: Chainlink Integration

```
    function read() external view returns (uint256) {
        // The relayer must not be null
        require(address(chainlinkAggregator) != address(0), "
↳ ChainlinkRelayer/null-aggregator");

        // Fetch values from Chainlink
        uint256 medianPrice         = multiply(uint(
↳ chainlinkAggregator.latestAnswer()), 10 ** uint(multiplier));
        uint256 aggregatorTimestamp = chainlinkAggregator.
↳ latestTimestamp();

```

```
        require(both(medianPrice > 0, subtract(now,
↳ aggregatorTimestamp) <= staleThreshold), "ChainlinkRelayer/invalid
↳ -price-feed");
        return medianPrice;
    }
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Implement the following function for checking extra validations. (stale price, incomplete round and return value).

Listing 3

```
(
          roundId,
          rawPrice,
          ,
          updateTime,
          answeredInRound
        ) = _pricefeed().latestRoundData();
        require(rawPrice > 0, "Chainlink price <= 0");
        require(updateTime != 0, "Incomplete round");
        require(answeredInRound >= roundId, "Stale price");
```

References:

Chainlink
OpenZeppelin Oracle Article

Remediation Plan:

SOLVED: The H2O team solved the above issue in the following commits
- b6cc5edc266b78a48a1f1f09bd9d229a73619c50
- 3a3ed51516b53d8ed16b703e07452f8edfb5019b
- eafcca39e12a33db609bc2a371275fd809e68815

As a result, the team added the additional validation and now uses latestRoundData.

3.3 (HAL-03) UNCHECKED TRANSFER - MEDIUM

Description:

In the contracts GebProxyActions.sol, GebProxyAuctionActions.sol, GebProxyIncentivesActions.sol, GebProxyLeverageActions.sol, GebProxySaviourActions.sol, the return values of the external transfer calls are not checked. It should be noted that token does not revert in case of failure and return false. If one of these tokens is used, a deposit would not revert if the transfer fails, and an attacker could deposit tokens for free.

Code Location:

Listing 4: GebProxyActions.sol (Line 149)

```
147     function coinJoin_join(address apt, address safeHandler, uint
 ↳ wad) public {
148         // Gets COIN from the user's wallet
149         CoinJoinLike(apt).systemCoin().transferFrom(msg.sender,
 ↳ address(this), wad);
150
151         _coinJoin_join(apt, safeHandler, wad);
152     }
```

Listing 5: GebProxyAuctionActions.sol (Line 63)

```
61     function claimProxyFunds(address tokenAddress) public {
62         DSTokenLike token = DSTokenLike(tokenAddress);
63         token.transfer(msg.sender, token.balanceOf(address(this)))
 ↳ ;
64     }
```

Listing 6: GebProxyAuctionActions.sol (Line 286)

```
278     function settleAuction(address auctionHouse_, uint auctionId)
 ↳ public {
```

```
279         SurplusAuctionHouseLike auctionHouse =
 ↳ SurplusAuctionHouseLike(auctionHouse_);
280         DSTokenLike stakedToken = DSTokenLike(auctionHouse.
 ↳ stakedToken());
281
282         // Settle auction
283         auctionHouse.settleAuction(auctionId);
284
285         // Sends the staked tokens to the msg.sender
286         stakedToken.transfer(msg.sender, stakedToken.balanceOf(
 ↳ address(this)));
287     }
```

Listing 7: GebProxyIncentivesActions.sol (Line 463)

```
458     function exitAndRemoveLiquidity(address coinJoin, address
 ↳ incentives, address uniswapRouter, uint[2] calldata
 ↳ minTokenAmounts) external returns (uint amountA, uint amountB) {
459         GebIncentivesLike incentivesContract = GebIncentivesLike(
 ↳ incentives);
460         DSTokenLike rewardToken = DSTokenLike(incentivesContract.
 ↳ rewardsToken());
461         DSTokenLike lpToken = DSTokenLike(incentivesContract.
 ↳ stakingToken());
462         incentivesContract.exit();
463         rewardToken.transfer(msg.sender, rewardToken.balanceOf(
 ↳ address(this)));
464         return _removeLiquidityUniswap(uniswapRouter, address(
 ↳ CoinJoinLike(coinJoin).systemCoin()), lpToken.balanceOf(address(
 ↳ this)), msg.sender, minTokenAmounts);
465     }
```

Listing 8: GebProxyIncentivesActions.sol (Lines 355,356)

```
350     function exitMine(address incentives) external {
351         GebIncentivesLike incentivesContract = GebIncentivesLike(
 ↳ incentives);
352         DSTokenLike rewardToken = DSTokenLike(incentivesContract.
 ↳ rewardsToken());
353         DSTokenLike lpToken = DSTokenLike(incentivesContract.
 ↳ stakingToken());
354         incentivesContract.exit();
```

```
355         rewardToken.transfer(msg.sender, rewardToken.balanceOf(
 ↳ address(this)));
356         lpToken.transfer(msg.sender, lpToken.balanceOf(address(
 ↳ this)));
357     }
```

Listing 9: GebProxyIncentivesActions.sol (Line 481)

```
474     function exitRemoveLiquidityRepayDebt(address manager, address
 ↳  coinJoin, uint safe, address incentives, address uniswapRouter,
 ↳ uint[2] calldata minTokenAmounts) external {
475
476         GebIncentivesLike incentivesContract = GebIncentivesLike(
 ↳ incentives);
477         DSTokenLike rewardToken = DSTokenLike(incentivesContract.
 ↳ rewardsToken());
478         DSTokenLike lpToken = DSTokenLike(incentivesContract.
 ↳ stakingToken());
479         DSTokenLike systemCoin = DSTokenLike(CoinJoinLike(coinJoin
 ↳ ).systemCoin());
480         incentivesContract.exit();
481         rewardToken.transfer(msg.sender, rewardToken.balanceOf(
 ↳ address(this)));
482
483         _removeLiquidityUniswap(uniswapRouter, address(systemCoin)
 ↳ , lpToken.balanceOf(address(this)), address(this), minTokenAmounts
 ↳ );
484
485         _repayDebt(manager, coinJoin, safe, systemCoin.balanceOf(
 ↳ address(this)), false);
486         msg.sender.call{value: address(this).balance}("");
487     }
```

Listing 10: GebProxyIncentivesActions.sol (Line 336)

```
319     function generateDebtAndProvideLiquidityStake(
320         address manager,
321         address taxCollector,
322         address coinJoin,
323         address uniswapRouter,
324         address incentives,
325         uint safe,
326         uint wad,
```

```
327         uint[2] calldata minTokenAmounts
328     ) external payable {
329         DSTokenLike systemCoin = DSTokenLike(CoinJoinLike(coinJoin
 ↳ ).systemCoin());
330         _generateDebt(manager, taxCollector, coinJoin, safe, wad,
 ↳ address(this));
331         _provideLiquidityUniswap(coinJoin, uniswapRouter, wad, msg
 ↳ .value, address(this), minTokenAmounts);
332         _stakeInMine(incentives);
333 
334         // sending back any leftover tokens/eth, necessary to
 ↳ manage change from providing liquidity
335         msg.sender.call{value: address(this).balance}("");
336         systemCoin.transfer(msg.sender, systemCoin.balanceOf(
 ↳ address(this)));
337     }
```

Listing 11: GebProxyIncentivesActions.sol (Line 300)

```
284     function generateDebtAndProvideLiquidityUniswap(
285         address manager,
286         address taxCollector,
287         address coinJoin,
288         address uniswapRouter,
289         uint safe,
290         uint wad,
291         uint[2] calldata minTokenAmounts
292     ) external payable {
293         DSTokenLike systemCoin = DSTokenLike(CoinJoinLike(coinJoin
 ↳ ).systemCoin());
294         _generateDebt(manager, taxCollector, coinJoin, safe, wad,
 ↳ address(this));
295 
296         _provideLiquidityUniswap(coinJoin, uniswapRouter, wad, msg
 ↳ .value, msg.sender, minTokenAmounts);
297 
298         // sending back any leftover tokens/eth, necessary to
 ↳ manage change from providing liquidity
299         msg.sender.call{value: address(this).balance}("");
300         systemCoin.transfer(msg.sender, systemCoin.balanceOf(
 ↳ address(this)));
301     }
```

Listing 12: GebProxyIncentivesActions.sol (Line 345)

```
341     function getRewards(address incentives) public {
342         GebIncentivesLike incentivesContract = GebIncentivesLike(
 ↳ incentives);
343         DSTokenLike rewardToken = DSTokenLike(incentivesContract.
 ↳ rewardsToken());
344         incentivesContract.getReward();
345         rewardToken.transfer(msg.sender, rewardToken.balanceOf(
 ↳ address(this)));
346     }
```

Listing 13: GebProxyIncentivesActions.sol (Line 235)

```
211     function lockETHGenerateDebtProvideLiquidityStake(
212         address manager,
213         address taxCollector,
214         address ethJoin,
215         address coinJoin,
216         address uniswapRouter,
217         address incentives,
218         uint safe,
219         uint deltaWad,
220         uint liquidityWad,
221         uint[2] memory minTokenAmounts
222     ) public payable {
223         DSTokenLike systemCoin = DSTokenLike(CoinJoinLike(coinJoin
 ↳ ).systemCoin());
224
225         _lockETH(manager, ethJoin, safe, subtract(msg.value,
 ↳ liquidityWad));
226
227         _generateDebt(manager, taxCollector, coinJoin, safe,
 ↳ deltaWad, address(this));
228
229         _provideLiquidityUniswap(coinJoin, uniswapRouter, deltaWad
 ↳ , liquidityWad, address(this), minTokenAmounts);
230
231         _stakeInMine(incentives);
232
233         // sending back any leftover tokens/eth, necessary to
 ↳ manage change from providing liquidity
234         msg.sender.call{value: address(this).balance}("");
```

```
235         systemCoin.transfer(msg.sender, systemCoin.balanceOf(
 ↳ address(this)));
236     }
```

Listing 14: GebProxyIncentivesActions.sol (Line 158)

```
137     function lockETHGenerateDebtProvideLiquidityUniswap(
138         address manager,
139         address taxCollector,
140         address ethJoin,
141         address coinJoin,
142         address uniswapRouter,
143         uint safe,
144         uint deltaWad,
145         uint liquidityWad,
146         uint[2] calldata minTokenAmounts
147     ) external payable {
148         DSTokenLike systemCoin = DSTokenLike(CoinJoinLike(coinJoin
 ↳ ).systemCoin());
149 
150         _lockETH(manager, ethJoin, safe, subtract(msg.value,
 ↳ liquidityWad));
151 
152         _generateDebt(manager, taxCollector, coinJoin, safe,
 ↳ deltaWad, address(this));
153 
154         _provideLiquidityUniswap(coinJoin, uniswapRouter, deltaWad
 ↳ , liquidityWad, msg.sender, minTokenAmounts);
155 
156         // sending back any leftover tokens/eth, necessary to
 ↳ manage change from providing liquidity
157         msg.sender.call{value: address(this).balance}("");
158         systemCoin.transfer(msg.sender, systemCoin.balanceOf(
 ↳ address(this)));
159     }
```

Listing 15: GebProxyIncentivesActions.sol (Line 372)

```
362     function migrateCampaign(address _oldIncentives, address
 ↳ _newIncentives) external {
363         GebIncentivesLike incentives = GebIncentivesLike(
 ↳ _oldIncentives);
```

```
364         GebIncentivesLike newIncentives = GebIncentivesLike(
 ↳ _newIncentives);
365         require(incentives.stakingToken() == newIncentives.
 ↳ stakingToken(), "geb-incentives/mismatched-staking-tokens");
366         DSTokenLike rewardToken = DSTokenLike(incentives.
 ↳ rewardsToken());
367         DSTokenLike lpToken = DSTokenLike(incentives.stakingToken
 ↳ ());
368         incentives.exit();
369
370         _stakeInMine(_newIncentives);
371
372         rewardToken.transfer(msg.sender, rewardToken.balanceOf(
 ↳ address(this)));
373     }
```

Listing 16: GebProxyIncentivesActions.sol (Line 197)

```
172     function openLockETHGenerateDebtProvideLiquidityStake(
173         address manager,
174         address taxCollector,
175         address ethJoin,
176         address coinJoin,
177         address uniswapRouter,
178         address incentives,
179         bytes32 collateralType,
180         uint256 deltaWad,
181         uint256 liquidityWad,
182         uint256[2] calldata minTokenAmounts
183     ) external payable returns (uint safe) {
184         safe = openSAFE(manager, collateralType, address(this));
185         DSTokenLike systemCoin = DSTokenLike(CoinJoinLike(coinJoin
 ↳ ).systemCoin());
186
187         _lockETH(manager, ethJoin, safe, subtract(msg.value,
 ↳ liquidityWad));
188
189         _generateDebt(manager, taxCollector, coinJoin, safe,
 ↳ deltaWad, address(this));
190
191         _provideLiquidityUniswap(coinJoin, uniswapRouter, deltaWad
 ↳ , liquidityWad, address(this), minTokenAmounts);
192
193         _stakeInMine(incentives);
```

```
194
195         // sending back any leftover tokens/eth, necessary to
 ↳ manage change from providing liquidity
196         msg.sender.call{value: address(this).balance}("");
197         systemCoin.transfer(msg.sender, systemCoin.balanceOf(
 ↳ address(this)));
198     }
```

Listing 17: GebProxyIncentivesActions.sol (Line 124)

```
102     function openLockETHGenerateDebtProvideLiquidityUniswap(
103         address manager,
104         address taxCollector,
105         address ethJoin,
106         address coinJoin,
107         address uniswapRouter,
108         bytes32 collateralType,
109         uint deltaWad,
110         uint liquidityWad,
111         uint[2] calldata minTokenAmounts
112     ) external payable returns (uint safe) {
113         safe = openSAFE(manager, collateralType, address(this));
114         DSTokenLike systemCoin = DSTokenLike(CoinJoinLike(coinJoin
 ↳ ).systemCoin());
115
116         _lockETH(manager, ethJoin, safe, subtract(msg.value,
 ↳ liquidityWad));
117
118         _generateDebt(manager, taxCollector, coinJoin, safe,
 ↳ deltaWad, address(this));
119
120         _provideLiquidityUniswap(coinJoin, uniswapRouter, deltaWad
 ↳ , liquidityWad, msg.sender, minTokenAmounts);
121
122         // sending back any leftover tokens/eth, necessary to
 ↳ manage change from providing liquidity
123         msg.sender.call{value: address(this).balance}("");
124         systemCoin.transfer(msg.sender, systemCoin.balanceOf(
 ↳ address(this)));
125     }
```

Listing 18: GebProxyIncentivesActions.sol (Lines 266,273)

```
258     function provideLiquidityStake(
259         address coinJoin,
260         address uniswapRouter,
261         address incentives,
262         uint wad,
263         uint[2] memory minTokenAmounts
264     ) public payable {
265         DSTokenLike systemCoin = DSTokenLike(CoinJoinLike(coinJoin
 ↳ ).systemCoin());
266         systemCoin.transferFrom(msg.sender, address(this), wad);
267         _provideLiquidityUniswap(coinJoin, uniswapRouter, wad, msg
 ↳ .value, address(this), minTokenAmounts);
268
269         _stakeInMine(incentives);
270
271         // sending back any leftover tokens/eth, necessary to
 ↳ manage change from providing liquidity
272         msg.sender.call{value: address(this).balance}("");
273         systemCoin.transfer(msg.sender, systemCoin.balanceOf(
 ↳ address(this)));
274     }
```

Listing 19: GebProxyIncentivesActions.sol (Lines 245,250)

```
243     function provideLiquidityUniswap(address coinJoin, address
 ↳ uniswapRouter, uint wad, uint[2] calldata minTokenAmounts)
 ↳ external payable {
244         DSTokenLike systemCoin = DSTokenLike(CoinJoinLike(coinJoin
 ↳ ).systemCoin());
245         systemCoin.transferFrom(msg.sender, address(this), wad);
246         _provideLiquidityUniswap(coinJoin, uniswapRouter, wad, msg
 ↳ .value, msg.sender, minTokenAmounts);
247
248         // sending back any leftover tokens/eth, necessary to
 ↳ manage change from providing liquidity
249         msg.sender.call{value: address(this).balance}("");
250         systemCoin.transfer(msg.sender, systemCoin.balanceOf(
 ↳ address(this)));
251     }
```

Listing 20: GebProxyIncentivesActions.sol (Line 419)

```
418     function removeLiquidityUniswap(address uniswapRouter, address
 ↳  systemCoin, uint value, uint[2] calldata minTokenAmounts)
 ↳ external returns (uint amountA, uint amountB) {
419         DSTokenLike(getWethPair(uniswapRouter, systemCoin)).
 ↳ transferFrom(msg.sender, address(this), value);
420         return _removeLiquidityUniswap(uniswapRouter, systemCoin,
 ↳ value, msg.sender, minTokenAmounts);
421     }
```

Listing 21: GebProxyIncentivesActions.sol (Line 307)

```
306     function stakeInMine(address incentives, uint wad) external {
307         DSTokenLike(GebIncentivesLike(incentives).stakingToken()).
 ↳ transferFrom(msg.sender, address(this), wad);
308         _stakeInMine(incentives);
309     }
```

Listing 22: GebProxyIncentivesActions.sol (Line 394)

```
388     function withdrawAndHarvest(address incentives, uint value)
 ↳ external {
389         GebIncentivesLike incentivesContract = GebIncentivesLike(
 ↳ incentives);
390         DSTokenLike rewardToken = DSTokenLike(incentivesContract.
 ↳ rewardsToken());
391         DSTokenLike lpToken = DSTokenLike(incentivesContract.
 ↳ stakingToken());
392         incentivesContract.withdraw(value);
393         getRewards(incentives);
394         lpToken.transfer(msg.sender, lpToken.balanceOf(address(
 ↳ this)));
395     }
```

Listing 23: GebProxyIncentivesActions.sol (Line 382)

```
378     function withdrawFromMine(address incentives, uint value)
 ↳ external {
379         GebIncentivesLike incentivesContract = GebIncentivesLike(
 ↳ incentives);
```

```
380         DSTokenLike lpToken = DSTokenLike(incentivesContract.
 ↳ stakingToken());
381         incentivesContract.withdraw(value);
382         lpToken.transfer(msg.sender, lpToken.balanceOf(address(
 ↳ this)));
383     }
```

Listing 24: GebProxyIncentivesActions.sol (Line 409)

```
403     function withdrawHarvestRemoveLiquidity(address incentives,
 ↳ address uniswapRouter, address systemCoin, uint value, uint[2]
 ↳ memory minTokenAmounts) public returns (uint amountA, uint amountB
 ↳ ) {
404         GebIncentivesLike incentivesContract = GebIncentivesLike(
 ↳ incentives);
405         DSTokenLike rewardToken = DSTokenLike(incentivesContract.
 ↳ rewardsToken());
406         DSTokenLike lpToken = DSTokenLike(incentivesContract.
 ↳ stakingToken());
407         incentivesContract.withdraw(value);
408         getRewards(incentives);
409         rewardToken.transfer(msg.sender, rewardToken.balanceOf(
 ↳ address(this)));
410         return _removeLiquidityUniswap(uniswapRouter, systemCoin,
 ↳ lpToken.balanceOf(address(this)), msg.sender, minTokenAmounts);
411     }
```

Listing 25: GebProxyIncentivesActions.sol (Line 449)

```
441     function withdrawRemoveLiquidityRepayDebt(address manager,
 ↳ address coinJoin, uint safe, address incentives, uint value,
 ↳ address uniswapRouter, uint[2] calldata minTokenAmounts) external
 ↳ {
442         GebIncentivesLike incentivesContract = GebIncentivesLike(
 ↳ incentives);
443         DSTokenLike rewardToken = DSTokenLike(incentivesContract.
 ↳ rewardsToken());
444         DSTokenLike systemCoin = DSTokenLike(CoinJoinLike(coinJoin
 ↳ ).systemCoin());
445         incentivesContract.withdraw(value);
446
447         _removeLiquidityUniswap(uniswapRouter, address(systemCoin)
 ↳ , value, address(this), minTokenAmounts);
```

```
448         _repayDebt(manager, coinJoin, safe, systemCoin.balanceOf(
 ↳ address(this)), false);
449         rewardToken.transfer(msg.sender, rewardToken.balanceOf(
 ↳ address(this)));
450         msg.sender.call{value: address(this).balance}("");
451     }
```

Listing 26: GebProxyLeverageActions.sol (Line 44)

```
38     function uniswapV2Call(address _sender, uint _amount0, uint
 ↳ _amount1, bytes calldata _data) external {
39         require(_sender == proxy, "invalid sender");
40         require(msg.sender == uniswapPair, "invalid uniswap pair")
 ↳ ;
41
42         // transfer coins
43         (address _tokenBorrow,,,,,,,address _proxy) = abi.decode(
 ↳ _data, (address, uint, address, bool, bool, bytes, address,
 ↳ address));
44         DSTokenLike(_tokenBorrow).transfer(proxy, (_amount0 > 0) ?
 ↳  _amount0 : _amount1);
45
46         // call proxy
47         (bool success,) = proxy.call(abi.encodeWithSignature("
 ↳ execute(address,bytes)", _proxy, msg.data));
48         require(success, "");
49     }
```

Listing 27: GebProxyLeverageActions.sol (Line 151)

```
113     function uniswapV2Call(address _sender, uint /* _amount0 */,
 ↳ uint /* _amount1 */, bytes calldata _data) external {
114         require(_sender == address(this), "only this contract may
 ↳ initiate");
115         DSAuth(address(this)).setAuthority(DSAuthority(address(0))
 ↳ );
116
117         // decode data
118         (
119             address _tokenBorrow,
120             uint _amount,
121             address _tokenPay,
122             bool _isBorrowingEth,
```

```
123             bool _isPayingEth,
124             bytes memory _userData,
125             address weth,
126             // address proxy
127         ) = abi.decode(_data, (address, uint, address, bool, bool,
 ↳  bytes, address, address));
128
129         // unwrap WETH if necessary
130         if (_isBorrowingEth) {
131             WethLike(weth).withdraw(_amount);
132         }
133
134         // compute the amount of _tokenPay that needs to be repaid
135         // address pairAddress = permissionedPairAddress; // gas
 ↳ efficiency
136         uint pairBalanceTokenBorrow = DSTokenLike(_tokenBorrow).
 ↳ balanceOf(FlashSwapProxy(msg.sender).uniswapPair());
137         uint pairBalanceTokenPay = DSTokenLike(_tokenPay).
 ↳ balanceOf(FlashSwapProxy(msg.sender).uniswapPair());
138         uint amountToRepay = ((1000 * pairBalanceTokenPay *
 ↳ _amount) / (997 * pairBalanceTokenBorrow)) + 1;
139
140         // do whatever the user wants
141         if (_isBorrowingEth)
142             flashLeverageCallback(_amount, amountToRepay,
 ↳ _userData);
143         else
144             flashDeleverageCallback(_amount, amountToRepay,
 ↳ _userData);
145
146         // payback loan
147         // wrap ETH if necessary
148         if (_isPayingEth) {
149             WethLike(weth).deposit{value: amountToRepay}();
150         }
151         DSTokenLike(_tokenPay).transfer(FlashSwapProxy(msg.sender)
 ↳ .uniswapPair(), amountToRepay);
152     }
```

Listing 28: GebProxySaviourActions.sol (Line 45)

```
44     function transferTokenFromAndApprove(address token, address
 ↳ target, uint256 amount) internal {
```

```
45          DSTokenLike(token).transferFrom(msg.sender, address(this),
↳  amount);
46          DSTokenLike(token).approve(target, 0);
47          DSTokenLike(token).approve(target, amount);
48      }
```

Listing 29: GebProxySaviourActions.sol (Line 59)

```
55      function transferTokensToCaller(address[] memory tokens)
↳ public {
56          for (uint i = 0; i < tokens.length; i++) {
57              uint256 selfBalance = DSTokenLike(tokens[i]).balanceOf
↳ (address(this));
58              if (selfBalance > 0) {
59                DSTokenLike(tokens[i]).transfer(msg.sender,
↳ selfBalance);
60              }
61          }
62      }
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

It is recommended to use SafeERC20, or ensure that the transfer return value is checked.

Remediation Plan:

RISK ACCEPTED: The H2O team accepts the risk of this issue, claiming that all tokens in the system, i.e., the OCEAN collateral token, the stable coin and the governance token will be used in production revert on failure. Moreover, the team will add WARNING documentation to warn downstream users to use the H2O system only with tokens that revert to the failure.

3.4 (HAL-04) IMPROPER ACCESS CONTROL POLICY - MEDIUM

Description:

Implementing a valid Access Control policy in smart contracts is essential to maintain security and decentralize permissions on a token. Moreover, access Control gives the features to mint/burn tokens and pause contracts. For instance, Ownership is the most common form of Access Control. In other words, the owner of a contract (the account that deployed it by default) can do some administrative tasks on it. Nevertheless, different authorization levels are required to keep the principle of least privilege, also known as least authority. Briefly, any process, user, or program can only access the necessary resources or information. Otherwise, the ownership role is beneficial in simple systems, but more complex projects require more roles using Role-based access control.

In most scope contracts, The authorizedAccounts of the contract are the accounts that control all privileged functions. Everything is managed and controlled by the Authorized accounts, with no other access control. If this account is compromised, then all functionalities would be controlled by an attacker, such as removing all other authorized accounts, changing price source to a malicious address, restarting feed and setting it to 0, starting and stopping the DSM OSM, etc.

Code Location:

Listing 30: In-Scope Contracts

```
1     constructor (address priceSource_) public {
2         authorizedAccounts[msg.sender] = 1;
```

Listing 31: In-Scope Contracts

```
1     constructor (address priceSource_, uint256 deviation) public {
2         require(deviation > 0 && deviation < WAD, "DSM/invalid-
```

```
↳ deviation");
3
4         authorizedAccounts[msg.sender] = 1;
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

It's recommended to use role-based access control based on the principle of least privilege to lock permissioned functions using different roles. For example, use authorized account to manage permission only, where it could use other roles to manage other critical functionalities such as DSM OSM role to start, stop DSM OSM, etc.

References:

Least Privilege
OpenZeppelin Access Control

Remediation Plan:

PENDING: The team will fix the issue in a future release adding the below points.
- In case externally owned accounts (EOAs) are the authorized accounts, the team will implement RBAC according to the software engineering principle of separation of concerns by authorizing only a single proxy contract and implementing RBAC in the proxy contract configuration. For example, the DSRoles proxy contract implements permissions at an address method signature level.
- In other cases, if a smart contract is the authorized account; the team will only authorize smart contracts for which the code is known and trusted in advance.

3.5 (HAL-05) MISSING RE-ENTRANCY PROTECTION - MEDIUM

Description:

It was identified that some in-scope contracts of H2O branch are missing nonReentrant guard. In these function, write of persistent state and external calls following an external call is identified, making it vulnerable to a Reentrancy attack.

- DebtCeilingProposal.sol contract function executeProposal missing nonReentrant guard.
- GlobalAuctionParamsProposal.sol contract function executeProposal missing nonReentrant guard.
- GlobalSettlementProposal.sol contract function executeProposal missing nonReentrant guard.

To protect against cross-function reentrancy attacks, it may be necessary to use a mutex. By using this lock, an attacker can no longer exploit the function with a recursive call. OpenZeppelin has its own mutex implementation called ReentrancyGuard which provides a modifier to any function called “nonReentrant” that guards the function with a mutex against the Reentrancy attacks.

Code Location:

- state change executed = true following an external call pause.executeTransaction

Listing 32: DebtCeilingProposal.sol (Lines 70,72)

```
67     function executeProposal() public { // exec
68         require(!executed, "proposal-already-executed");
69
70         pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
```

```
71
72          executed = true;
73      }
```

Listing 33: GlobalAuctionParamsProposal.sol (Lines 86,94,96)

```
76      function executeProposal() public {
77          require(!executed, "proposal-already-executed");
78
79          bytes memory signature =
80              abi.encodeWithSignature(
81                  "modifyParameters(address,bytes32,uint256)",
82                  accountingEngine,
83                  bytes32("initialDebtAuctionMintedTokens"),
84                  initialDebtMintedTokens
85          );
86          pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
87
88          signature = abi.encodeWithSignature(
89                  "modifyParameters(address,bytes32,uint256)",
90                  accountingEngine,
91                  bytes32("debtAuctionBidSize"),
92                  debtAuctionBidSize
93          );
94          pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
95
96          executed = true;
97      }
```

Listing 34: GlobalSettlementProposal.sol (Lines 63,65)

```
57      function executeProposal() public {
58          require(!executed, "proposal-already-executed");
59
60          bytes memory signature =
61                  abi.encodeWithSignature("shutdownSystem(address)",
↳  globalSettlement);
62
63          pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
64
```

```
65          executed = true;
66      }
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Change the code to follow the checks-effects-interactions pattern and use ReentrancyGuard through the nonReentrant modifier.

Remediation Plan:

SOLVED: The H2O team solved the above issue in the commit 384b866ebf808831d3f5980f4f4f62dc17d9365d. As a result, the team added the nonReentrant modifier.

3.6 (HAL-06) AUTHORIZE ACCOUNT CAN SET INVALID BASE AND MAX REWARDS - LOW

Description:

In the scope contract FSMWrapper.sol, OSM.sol and DSM.sol, it is observed that the authorize accounts can set max and base rewards to an invalid value such as 0. It is possible, as no minimum and maximum value validation on the state variable baseUpdateCallerReward and maxUpdateCallerReward is performed.

Code Location:

- Authorize accounts calling modifyParameters with byte32 value of baseUpdateCallerReward as a parameter and val as 0 sets baseUpdateCallerReward to 0. After this, call modifyParameters with byte32 value of maxUpdateCallerReward as a parameter and val as 0 sets maxUpdateCallerReward to 0.

Listing 35: FSMWrapper.sol, OSM.sol and DSM.sol

```
    function modifyParameters(bytes32 parameter, uint256 val)
↳ external isAuthorized {
        if (parameter == "baseUpdateCallerReward") {
          require(val <= maxUpdateCallerReward, "FSMWrapper/
↳ invalid-base-caller-reward");
          baseUpdateCallerReward = val;
        }
        else if (parameter == "maxUpdateCallerReward") {
          require(val >= baseUpdateCallerReward, "FSMWrapper/
↳ invalid-max-caller-reward");
          maxUpdateCallerReward = val;
        }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

Consider declaring a state variable that can store baseUpdateCallerReward and maxUpdateCallerReward reward's minimum and maximum value. Add the require check to make sure supplied val for baseUpdateCallerReward and maxUpdateCallerReward parameter lies within the min and max value.

Remediation Plan:

RISK ACCEPTED: The H2O team accepts the risk of this issue, claiming that for the above issue, in case the delay is set incorrectly, you can use the same function call to set it to a correct value. Additionally, the team can add an extra layer by including a multisig system.

3.7 (HAL-07) AUTHORIZE ACCOUNT CAN INCREASE MAX REWARD DELAY TO MAX INT VALUE - LOW

Description:

In the scope contract FSMWrapper.sol, OSM.sol and DSM.sol, it is observed that the authorize accounts can set max reward increase delay to a vast number, as val is a type of uint256 a max value 2^256-1 could be supplied as an argument which ends up setting the state variable maxRewardIncreaseDelay to max INT value, this sets max reward not to increase for years.

Code Location:

- Authorize accounts calling modifyParameters with byte32 value of maxRewardIncreaseDelay as a parameter and val as 2^256-1 sets maxRewardIncreaseDelay to max INT value.

Listing 36: FSMWrapper.sol, OSM.sol and DSM.sol (Lines 14-17)

```
1     function modifyParameters(bytes32 parameter, uint256 val)
↳ external isAuthorized {
2         if (parameter == "baseUpdateCallerReward") {
3           require(val <= maxUpdateCallerReward, "FSMWrapper/
↳ invalid-base-caller-reward");
4           baseUpdateCallerReward = val;
5         }
6         else if (parameter == "maxUpdateCallerReward") {
7           require(val >= baseUpdateCallerReward, "FSMWrapper/
↳ invalid-max-caller-reward");
8           maxUpdateCallerReward = val;
9         }
10        else if (parameter == "perSecondCallerRewardIncrease") {
11          require(val >= RAY, "FSMWrapper/invalid-caller-reward-
↳ increase");
12          perSecondCallerRewardIncrease = val;
13        }
```

```
14          else if (parameter == "maxRewardIncreaseDelay") {
15            require(val > 0, "FSMWrapper/invalid-max-increase-delay"
↳ );
16            maxRewardIncreaseDelay = val;
17          }
18          else if (parameter == "reimburseDelay") {
19            reimburseDelay = val;
20          }
21          else revert("FSMWrapper/modify-unrecognized-param");
22          emit ModifyParameters(
23            parameter,
24            val
25          );
26      }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to set a maximum value of `maxRewardIncreaseDelay` and add a `require` check that validates `val` is always less than or equal to max value of `maxRewardIncreaseDelay` such as `require( 0 < val <= maxDelayLimit , "FSMWrapper/invalid-max-increase-delay");`.

Remediation Plan:

RISK ACCEPTED: The H2O team accepts the risk of this issue, claiming that for the above issue, in case the delay is set incorrectly, you can use the same function call to set it to a correct value. Additionally, the team can add an extra layer by including a multisig system.

3.8 (HAL-08) THE CONTRACT FUNCTION SHOULD APPROVE(0) FIRST - LOW

Description:

Some tokens (like USDT L199) do not work when changing the allowance from an existing non-zero allowance value. They must first be approved by zero, and then the actual allowance must be approved.

Listing 37

```
IERC20(token).approve(address(operator), 0);
IERC20(token).approve(address(operator), amount);
```

Code Location:

Listing 38: GebProxyIncentivesActions.sol (Line 81)

```
    function _removeLiquidityUniswap(address uniswapRouter,
↳ address systemCoin, uint value, address to, uint[2] memory
↳ minTokenAmounts) internal returns (uint amountA, uint amountB) {
        DSTokenLike(getWethPair(uniswapRouter, systemCoin)).
↳ approve(uniswapRouter, value);
        return IUniswapV2Router02(uniswapRouter).
↳ removeLiquidityETH(
            systemCoin,
            value,
            minTokenAmounts[0],
            minTokenAmounts[1],
            to,
            block.timestamp
        );
    }
```

Listing 39: GebProxyIncentivesActions.sol (Line 70)

```
    function _stakeInMine(address incentives) internal {
        DSTokenLike lpToken = DSTokenLike(GebIncentivesLike(
↳ incentives).stakingToken());
```

```
70          lpToken.approve(incentives, uint(0 - 1));
71          GebIncentivesLike(incentives).stake(lpToken.balanceOf(
↳ address(this)));
72      }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

Approve with a zero amount first before setting the actual amount.

Remediation Plan:

RISK ACCEPTED: The H2O team accepts the risk of this issue, claiming that the issue will not be triggered as Uniswap LP tokens will be used in production. Moreover, the team will add documentation to warn downstream users using the H2O system only with tokens that do not exhibit this behavior in their approve functions.

3.9 (HAL-09) ERC20 APPROVE METHOD MISSING RETURN VALUE CHECK - LOW

Description:

The following contract functions perform an ERC20.approve() call, but does not check the success return value. Some tokens do not revert if the approval failed and return false instead of true.

Code Location:

Listing 40: GebProxyActions.sol (Line 192)

```
187     function increaseBidSize(address auctionHouse, uint auctionId,
 ↳  uint bidSize) public {
188         SurplusAuctionHouseLike surplusAuctionHouse =
 ↳ SurplusAuctionHouseLike(auctionHouse);
189         DSTokenLike protocolToken = DSTokenLike(
 ↳ surplusAuctionHouse.protocolToken());
190
191         require(protocolToken.transferFrom(msg.sender, address(
 ↳ this), bidSize), "geb-proxy-auction-actions/transfer-from-failed")
 ↳ ;
192         protocolToken.approve(address(surplusAuctionHouse),
 ↳ bidSize);
193         // Restarts auction if inactive
194         (, uint amountToSell,, uint48 bidExpiry, uint48
 ↳ auctionDeadline) = surplusAuctionHouse.bids(auctionId);
195         if (auctionDeadline < now && bidExpiry == 0 &&
 ↳ auctionDeadline > 0) {
196             surplusAuctionHouse.restartAuction(auctionId);
197         }
198         // Bid
199         surplusAuctionHouse.increaseBidSize(auctionId,
 ↳ amountToSell, bidSize);
200     }
```

Listing 41: GebProxyActions.sol (Line 177)

```
169     function startAndIncreaseBidSize(address
 ↳ accountingEngineAddress, uint bidSize) public {
170         AccountingEngineLike accountingEngine =
 ↳ AccountingEngineLike(accountingEngineAddress);
171         SurplusAuctionHouseLike surplusAuctionHouse =
 ↳ SurplusAuctionHouseLike(accountingEngine.surplusAuctionHouse());
172         DSTokenLike protocolToken = DSTokenLike(
 ↳ surplusAuctionHouse.protocolToken());
173
174         // Starts auction
175         uint auctionId = accountingEngine.auctionSurplus();
176         require(protocolToken.transferFrom(msg.sender, address(
 ↳ this), bidSize), "geb-proxy-auction-actions/transfer-from-failed")
 ↳ ;
177         protocolToken.approve(address(surplusAuctionHouse),
 ↳ bidSize);
178         (, uint amountToSell,,,) = surplusAuctionHouse.bids(
 ↳ auctionId);
179         // Bids
180         surplusAuctionHouse.increaseBidSize(auctionId,
 ↳ amountToSell, bidSize);
181     }
```

Listing 42: GebProxyIncentivesActions.sol (Line 55)

```
54     function _provideLiquidityUniswap(address coinJoin, address
 ↳ uniswapRouter, uint tokenWad, uint ethWad, address to, uint[2]
 ↳ memory minTokenAmounts) internal {
55         CoinJoinLike(coinJoin).systemCoin().approve(uniswapRouter,
 ↳  tokenWad);
56         IUniswapV2Router02(uniswapRouter).addLiquidityETH{value:
 ↳ ethWad}(
57             address(CoinJoinLike(coinJoin).systemCoin()),
58             tokenWad,
59             minTokenAmounts[0],
60             minTokenAmounts[1],
61             to,
62             block.timestamp
63         );
64     }
```

Listing 43: GebProxyIncentivesActions.sol (Line 81)

```
80      function _removeLiquidityUniswap(address uniswapRouter,
↳ address systemCoin, uint value, address to, uint[2] memory
↳ minTokenAmounts) internal returns (uint amountA, uint amountB) {
81          DSTokenLike(getWethPair(uniswapRouter, systemCoin)).
↳ approve(uniswapRouter, value);
82          return IUniswapV2Router02(uniswapRouter).
↳ removeLiquidityETH(
83              systemCoin,
84              value,
85              minTokenAmounts[0],
86              minTokenAmounts[1],
87              to,
88              block.timestamp
89          );
90      }
```

Listing 44: GebProxyIncentivesActions.sol (Line 70)

```
68      function _stakeInMine(address incentives) internal {
69          DSTokenLike lpToken = DSTokenLike(GebIncentivesLike(
↳ incentives).stakingToken());
70          lpToken.approve(incentives, uint(0 - 1));
71          GebIncentivesLike(incentives).stake(lpToken.balanceOf(
↳ address(this)));
72      }
```

Listing 45: CollateralLike Abstract Interface (Lines 23-25)

```
22 abstract contract CollateralLike {
23      function approve(address, uint) virtual public;
24      function transfer(address, uint) virtual public;
25      function transferFrom(address, address, uint) virtual public;
26 }
```

Listing 46: DSTokenLike Abstract Interface (Line 73)

```
71 abstract contract DSTokenLike {
72      function balanceOf(address) virtual public view returns (uint)
↳ ;
73      function approve(address, uint) virtual public;
```

```
74     function transfer(address, uint) virtual public returns (bool)
↳ ;
75     function transferFrom(address, address, uint) virtual public
↳ returns (bool);
76 }
```

Listing 47: WethLike Abstract Interface (Lines 80-82)

```
78 abstract contract WethLike {
79     function balanceOf(address) virtual public view returns (uint)
↳ ;
80     function approve(address, uint) virtual public;
81     function transfer(address, uint) virtual public;
82     function transferFrom(address, address, uint) virtual public;
83     function deposit() virtual public payable;
84     function withdraw(uint) virtual public;
85 }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It's recommend to using OpenZeppelin's SafeERC20 versions with the safeIncreaseAllowance & safeDecreaseAllowance function that handles the return value check as well as non-standard-compliant tokens.

Reference:

OpenZeppelin SafeERC20

Remediation Plan:

RISK ACCEPTED: The H2O team accepts the risk of this issue, claiming that all tokens to be used in production will be reverted in the event of a

failure. Moreover, the team will add documentation to warn downstream users to use the H2O system only with tokens that will revert the `ERC20.approve()`.

3.10 (HAL-10) MISSING SAFE ENGINE ADDRESS VALIDATION - LOW

Description:

It was noted that safeEngine_ address is not validated in SurplusAuctionHouse.sol contract. Lack of address validation has been found when assigning supplied address values to state variables directly. safeEngine_ address should be !=0 or add logic to check if the provided address is a valid Safe Engine Address.

Code Location:

Listing 48: SurplusAuctionHouse.sol (Lines 119,121)

```
119     constructor(address safeEngine_, address protocolToken_)
 ↳ public {
120         authorizedAccounts[msg.sender] = 1;
121         safeEngine = SAFEEngineLike(safeEngine_);
122         protocolToken = TokenLike(protocolToken_);
123         contractEnabled = 1;
124         emit AddAuthorization(msg.sender);
125     }
```

Listing 49: SAFEEngine.sol

```
180     constructor() public {
181         authorizedAccounts[msg.sender] = 1;
182         safeDebtCeiling = uint256(-1);
183         contractEnabled = 1;
184         emit AddAuthorization(msg.sender);
185         emit ModifyParameters("safeDebtCeiling", uint256(-1));
186     }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended to add proper address validation when assigning a supplied input to a variable. Consider adding a logic to validate provided safeEngine_ address is a valid safeEngine.

Remediation Plan:

SOLVED: The H2O team solved the above issue in the commit 9c3225a836a570df867a8116eeac6f678b322a13. As a result, the team added the zero address check to safeEngine_ throughout the in-scope H2O system.

3.11 (HAL-11) MISSING FACTORY ADDRESS VALIDATION - LOW

Description:

It was noted that factory_ address is not validated in GebProxyRegistry.sol contract. Lack of address validation has been found when assigning supplied address values to state variables directly. factory_ address should be !=0 or add logic to check if the provided address is a valid Factory Address.

Code Location:

Listing 50: GebProxyRegistry.sol (Line 13)

```
13     constructor(address factory_) public {
14         factory = DSProxyFactory(factory_);
15     }
```

Listing 51: proxy.sol (Lines 101,102)

```
96 contract DSProxyFactory {
97     event Created(address indexed sender, address indexed owner,
↳ address proxy, address cache);
98     mapping(address=>bool) public isProxy;
99     DSProxyCache public cache;
100
101    constructor() public {
102        cache = new DSProxyCache();
103    }
104
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended to add proper address validation when assigning a supplied input to a variable. Consider adding a logic to validate provided `factory_` address is a valid `factory`.

Remediation Plan:

SOLVED: The H2O team solved the above issue in the commit 48863e0b91e152c6900fb6d61d31566fa9819bb1. As a result, the team added the zero address check to `factory_`.

Listing 52: Updated Code

```
    constructor(address factory_) public {
        require(factory_ != address(0), "GebProxyRegistry/proxy-
↳ factory-address-invalid");
        factory = DSProxyFactory(factory_);
    }
```

3.12 (HAL-12) EXTERNAL FUNCTION CALLS WITHIN LOOP - LOW

Description:

External calls inside a loop increase Gas usage or might lead to a denial-of-service attack. In GebProxySaviourActions.sol contract functions discovered, there is a for loop on variable i that iterates up to the tokens.length array length. Where, in AccessManagementProposal.sol, CollateralStabilityFeeProposal.sol, LiquidationCRatioProposal.sol, LiquidationPenaltyProposal.sol, MultiDebtCeilingProposal.sol, SafetyCRatioProposal.sol, SecondaryTaxReceiversProposal.sol contract functions discovered there is a for loop on variable i that iterates up to the gebModules.length, and collateralTypes.length array length and these loop has external calls inside a loop. If this integer is evaluated at huge numbers, this can cause a DoS.

Code Location:

Listing 53: GebProxySaviourActions.sol (Lines 57,59)

```
55     function transferTokensToCaller(address[] memory tokens)
↳ public {
56         for (uint i = 0; i < tokens.length; i++) {
57             uint256 selfBalance = DSTokenLike(tokens[i]).balanceOf
↳ (address(this));
58             if (selfBalance > 0) {
59               DSTokenLike(tokens[i]).transfer(msg.sender,
↳ selfBalance);
60             }
61         }
62     }
```

Listing 54: AccessManagementProposal.sol (Line 84)

```
73     function executeProposal() public {
74         require(!executed, "proposal-already-executed");
75
```

```
76          for (uint256 i = 0; i < gebModules.length; i++) {
77              bytes memory signature =
78                  abi.encodeWithSignature(
79                      (grantAccess[i] != 0) ? "addAuthorization(
↳ address,address)" : "removeAuthorization(address,address)",
80                      gebModules[i],
81                      addresses[i]
82              );
83
84              pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
85          }
86
87          executed = true;
88      }
```

Listing 55: AccessManagementProposal.sol (Line 69)

```
57      function scheduleProposal() public {
58          require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
59          earliestExecutionTime = now + PauseLike(pause).delay();
60
61          for (uint256 i = 0; i < gebModules.length; i++) {
62              bytes memory signature =
63                  abi.encodeWithSignature(
64                      (grantAccess[i] != 0) ? "addAuthorization(
↳ address,address)" : "removeAuthorization(address,address)",
65                      gebModules[i],
66                      addresses[i]
67              );
68
69              pause.scheduleTransaction(target, codeHash, signature,
↳  earliestExecutionTime);
70          }
71      }
```

Listing 56: CollateralStabilityFeeProposal.sol (Line 19)

```
16      function deploy(address taxCollector, bytes32[] calldata
↳ collateralTypes, uint256[] calldata stabilityFees) external {
17
18          for (uint i = 0; i < collateralTypes.length; i++)
```

```
            ConfigLike(taxCollector).modifyParameters(
↳ collateralTypes[i], "stabilityFee", stabilityFees[i]);
    }
```

Listing 57: LiquidationCRatioProposal.sol (Line 84)

```
    function executeProposal() public {
        require(!executed, "proposal-already-executed");

        for (uint256 i = 0; i < collateralTypes.length; i++) {
            bytes memory signature =
                abi.encodeWithSignature(
                    "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
                    oracleRelayer,
                    collateralTypes[i],
                    bytes32("liquidationCRatio"),
                    liquidationCRatios[i]
            );
            pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
        }

        executed = true;
    }
```

Listing 58: LiquidationCRatioProposal.sol (Line 68)

```
    function scheduleProposal() public {
        require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
        earliestExecutionTime = now + PauseLike(pause).delay();

        for (uint256 i = 0; i < collateralTypes.length; i++) {
            bytes memory signature =
                abi.encodeWithSignature(
                    "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
                    oracleRelayer,
                    collateralTypes[i],
                    bytes32("liquidationCRatio"),
                    liquidationCRatios[i]
            );
```

```
68              pause.scheduleTransaction(target, codeHash, signature,
↳  earliestExecutionTime);
69          }
70      }
```

Listing 59: LiquidationPenaltyProposal.sol (Line 84)

```
72      function executeProposal() public {
73          require(!executed, "proposal-already-executed");
74
75          for (uint256 i = 0; i < collateralTypes.length; i++) {
76              bytes memory signature =
77                  abi.encodeWithSignature(
78                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
79                      liquidationEngine,
80                      collateralTypes[i],
81                      bytes32("liquidationPenalty"),
82                      liquidationPenalties[i]
83              );
84              pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
85          }
86
87          executed = true;
88      }
```

Listing 60: LiquidationPenaltyProposal.sol (Line 68)

```
55      function scheduleProposal() public {
56          require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
57          earliestExecutionTime = now + PauseLike(pause).delay();
58
59          for (uint256 i = 0; i < collateralTypes.length; i++) {
60              bytes memory signature =
61                  abi.encodeWithSignature(
62                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
63                      liquidationEngine,
64                      collateralTypes[i],
65                      bytes32("liquidationPenalty"),
66                      liquidationPenalties[i]
```

```
67              );
68              pause.scheduleTransaction(target, codeHash, signature,
↳  earliestExecutionTime);
69          }
70      }
```

Listing 61: MultiDebtCeilingProposal.sol (Line 85)

```
73      function executeProposal() public {
74          require(!executed, "proposal-already-executed");
75
76          for (uint256 i = 0; i < collateralTypes.length; i++) {
77              bytes memory signature =
78                  abi.encodeWithSignature(
79                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
80                      safeEngine,
81                      collateralTypes[i],
82                      bytes32("debtCeiling"),
83                      debtCeilings[i]
84              );
85              pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
86          }
87
88          executed = true;
89      }
```

Listing 62: MultiDebtCeilingProposal.sol (Line 69)

```
56      function scheduleProposal() public {
57          require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
58          earliestExecutionTime = now + PauseLike(pause).delay();
59
60          for (uint256 i = 0; i < collateralTypes.length; i++) {
61              bytes memory signature =
62                  abi.encodeWithSignature(
63                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
64                      safeEngine,
65                      collateralTypes[i],
66                      bytes32("debtCeiling"),
```

```
67                      debtCeilings[i]
68              );
69              pause.scheduleTransaction(target, codeHash, signature,
↳  earliestExecutionTime);
70          }
71      }
```

Listing 63: SafetyCRatioProposal.sol (Line 84)

```
72      function executeProposal() public {
73          require(!executed, "proposal-already-executed");
74
75          for (uint256 i = 0; i < collateralTypes.length; i++) {
76              bytes memory signature =
77                  abi.encodeWithSignature(
78                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
79                      oracleRelayer,
80                      collateralTypes[i],
81                      bytes32("safetyCRatio"),
82                      safetyCRatios[i]
83              );
84              pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
85          }
86
87          executed = true;
88      }
```

Listing 64: SafetyCRatioProposal.sol (Line 68)

```
55      function scheduleProposal() public {
56          require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
57          earliestExecutionTime = now + PauseLike(pause).delay();
58
59          for (uint256 i = 0; i < collateralTypes.length; i++) {
60              bytes memory signature =
61                  abi.encodeWithSignature(
62                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
63                      oracleRelayer,
64                      collateralTypes[i],
```

```
65                      bytes32("safetyCRatio"),
66                      safetyCRatios[i]
67              );
68              pause.scheduleTransaction(target, codeHash, signature,
↳  earliestExecutionTime);
69          }
70      }
```

Listing 65: SecondaryTaxReceiversProposal.sol (Line 24)

```
14      function deploy(
15          address taxCollector,
16          bytes32[] calldata collateralTypes,
17          uint256[] calldata positions,
18          uint256[] calldata percentages,
19          address[] calldata secondaryReceivers)
20          external
21      {
22
23          for (uint i = 0; i < collateralTypes.length; i++)
24              ConfigLike(taxCollector).modifyParameters(
↳ collateralTypes[i], positions[i], percentages[i],
↳ secondaryReceivers[i]);
25      }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended to set the max length to which a for loop can iterate. If possible, use pull over push strategy for external calls.

Reference:

External Calls Recommendation

Remediation Plan:

RISK ACCEPTED: The H2O team accepts the risk of this issue, claiming that for V1 of the system, the identified array lengths will not exceed more than a handful of values. However, in the case of `collateralTypes`, the length of the array is fixed at `1` for the lifetime of the V1 system. Furthermore, the team claims that the contracts in which these issues occur are optional and may not be seen in the deployment for the lifetime of V1. However, the team notified that in the V2 system, the team would address these issues.

3.13 (HAL-13) IGNORE RETURN VALUES - LOW

Description:

The return value of an external call is not stored in a local or state variable. In contract GebProxyIncentivesActions.sol, there is an instance where external method is being called, and return value is being ignored.

Code Location:

Listing 66: GebProxyIncentivesActions.sol (Lines 56-63)

```
54     function _provideLiquidityUniswap(address coinJoin, address
↳ uniswapRouter, uint tokenWad, uint ethWad, address to, uint[2]
↳ memory minTokenAmounts) internal {
55         CoinJoinLike(coinJoin).systemCoin().approve(uniswapRouter,
↳  tokenWad);
56         IUniswapV2Router02(uniswapRouter).addLiquidityETH{value:
↳ ethWad}(
57             address(CoinJoinLike(coinJoin).systemCoin()),
58             tokenWad,
59             minTokenAmounts[0],
60             minTokenAmounts[1],
61             to,
62             block.timestamp
63         );
64     }
```

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

Add return value check to avoid unexpected crash of the contract. Return value check will help in handling the exceptions better way.

Remediation Plan:

RISK ACCEPTED: The H2O team accepts the risk of this issue, claiming that the implementation of addLiquidityETH throws on failure unless the codebase is used with an alternative implementation of Uniswap that doesn't exhibit this behavior. Furthermore, the team claims that only addLiquidityETH implementations that revert on failure are used with this codebase. Moreover, the team will add documentation to WARN downstream users of this assumption.

3.14 (HAL-14) MISSING ZERO-ADDRESS CHECK - LOW

Description:

There are multiple instances found where Address validation is missing. Lack of zero address validation has been found when assigning user supplied address values to state variables directly.

- In contract AccessManagementProposal.sol:
  - constructor lacks a zero address check on _target.
- In contract CollateralStabilityFeeProposal.sol:
  - constructor lacks a zero address check on _pause.
- In contract DebtCeilingProposal.sol:
  - constructor lacks a zero address check on _target, _cdpEngine.
- In contract GlobalAuctionParamsProposal.sol:
  - constructor lacks a zero address check on _target, _accountingEngine.
- In contract GlobalSettlementProposal.sol:
  - constructor lacks a zero address check on _target, _globalSettlement.
- In contract GlobalStabilityFeeProposal.sol:
  - constructor lacks a zero address check on _pause.
- In contract LiquidationCRatioProposal.sol:
  - constructor lacks a zero address check on _target, _oracleRelayer.
- In contract LiquidationPenaltyProposal.sol:
  - constructor lacks a zero address check on _target, _liquidationEngine.

- In contract MultiDebtCeilingProposal.sol:
  - constructor lacks a zero address check on _target, _safeEngine.
- In contract NewCollateralProposal.sol:
  - constructor lacks a zero address check on pause_.
- In contract Proposal.sol:
  - constructor lacks a zero address check on target_.
- In contract SafetyCRatioProposal.sol:
  - constructor lacks a zero address check on _target, _oracleRelayer.
- In contract SecondaryTaxReceiversProposal.sol:
  - constructor lacks a zero address check on _pause.
- In contract GebProxyLeverageActions.sol:
  - constructor lacks a zero address check on _pair.
  - uniswapV2Call lacks a zero address check on _proxy.

Code Location:

Zero Address Validation missing before address assignment to this state variable.

Listing 67: AccessManagementProposal.sol

```
1  target = _target (#47)
```

Listing 68: CollateralStabilityFeeProposal.sol

```
1  pause = _pause (#43)
```

Listing 69: DebtCeilingProposal.sol

```
1  target = _target (#44)
2  cdpEngine = _cdpEngine (#45)
```

Listing 70: GlobalAuctionParamsProposal.sol

```
1 target = _target (#45)
2 accountingEngine = _accountingEngine (#46)
```

Listing 71: GlobalSettlementProposal.sol

```
1 target = _target (#39)
2 globalSettlement = _globalSettlement (#40)
```

Listing 72: GlobalStabilityFeeProposal.sol

```
1 pause = _pause (#37)
```

Listing 73: LiquidationCRatioProposal.sol

```
1 target = _target (#46)
2 oracleRelayer = _oracleRelayer (#47)
```

Listing 74: LiquidationPenaltyProposal.sol

```
1 target = _target (#46)
2 liquidationEngine = _liquidationEngine (#47)
```

Listing 75: MultiDebtCeilingProposal.sol

```
1 target = _target (#47)
2 safeEngine = _safeEngine (#48)
```

Listing 76: NewCollateralProposal.sol

```
1 pause = pause_ (#76)
```

Listing 77: Proposal.sol

```
1 target = target_ (#30)
```

Listing 78: SafetyCRatioProposal.sol

```
1 target = _target (#46)
2 oracleRelayer = _oracleRelayer (#47)
```

Listing 79: SecondaryTaxReceiversProposal.sol

```
1 pause = _pause (#60)
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Although administrative restrictions are imposed to this function due to the role-based access controls (RBAC), it is recommended to add proper address validation when assigning user supplied input to a variable. This could be as simple as using the following statement:

Listing 80

```
1 require(address_input != 0, "Address is zero")
```

Remediation Plan:

RISK ACCEPTED: The H2O team accepts the risk of this issue, claiming that these state variables are declared and assigned in scripts that mitigate human error compared to calling these functions manually; as a result, they do not appear to be a problem in practice.

3.15 (HAL-15) USE OF DEPRECATED NOW FUNCTION - LOW

Description:

In the entire smart contracts, Instead of block.timestamp, now is used, which is deprecated. block.timestamp is way more explicit in showing the intent, while now relates to the timestamp of the block controlled by the miner. Reference

Code Location:

Listing 81: LiquidationPenaltyProposal.sol (Line 68)

```
55     function scheduleProposal() public {
56         require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
57         earliestExecutionTime = now + PauseLike(pause).delay();
58
59         for (uint256 i = 0; i < collateralTypes.length; i++) {
60             bytes memory signature =
61                 abi.encodeWithSignature(
62                     "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
63                     liquidationEngine,
64                     collateralTypes[i],
65                     bytes32("liquidationPenalty"),
66                     liquidationPenalties[i]
67             );
68             pause.scheduleTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
69         }
70     }
```

Listing 82: MultiDebtCeilingProposal.sol (Line 85)

```
73     function executeProposal() public {
74         require(!executed, "proposal-already-executed");
75
```

```
76          for (uint256 i = 0; i < collateralTypes.length; i++) {
77              bytes memory signature =
78                  abi.encodeWithSignature(
79                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
80                      safeEngine,
81                      collateralTypes[i],
82                      bytes32("debtCeiling"),
83                      debtCeilings[i]
84              );
85              pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
86          }
87
88          executed = true;
89      }
```

Listing 83: MultiDebtCeilingProposal.sol (Line 69)

```
56      function scheduleProposal() public {
57          require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
58          earliestExecutionTime = now + PauseLike(pause).delay();
59
60          for (uint256 i = 0; i < collateralTypes.length; i++) {
61              bytes memory signature =
62                  abi.encodeWithSignature(
63                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
64                      safeEngine,
65                      collateralTypes[i],
66                      bytes32("debtCeiling"),
67                      debtCeilings[i]
68              );
69              pause.scheduleTransaction(target, codeHash, signature,
↳  earliestExecutionTime);
70          }
71      }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended to replace now with block.timestamp.

Remediation Plan:

SOLVED: The H2O team solved the above issue in the commit 54690e489294a144e56659b117ced4ac11de4400. As a result, the team replaces now with block.timestamp throughout the in-scope H2O system.

3.16 (HAL-16) AUCTION CAN BE EXTENDED INDEFINITELY - INFORMATIONAL

Description:

In the contract, SurplusAuctionHouse.sol observed that the function restartAuction lacks access control; thus, anyone can make an external call to this function. Furthermore, calling the restartAuction function just before the auction expires (no bid is issued) allows the auction to be extended by two days. An external call to restartAuction only requires a valid auction id, the auction id can easily be obtainable via the startAuction function emit event. This can be done repeatedly to extend the auction forever for any auction.

Code Location:

- Default Auction length is of two days

Listing 84: SurplusAuctionHouse.sol (Line 298)

```
297     // Total length of the auction
298     uint48   public   totalAuctionLength = 2 days;
```

- Active auction's id can be obtained via emit event of startAuction function

Listing 85: SurplusAuctionHouse.sol (Line 386)

```
386         emit StartAuction(id, auctionsStarted, amountToSell,
 ↳ initialBid, bids[id].auctionDeadline);
```

- Condition bids[id].auctionDeadline < now will always be true if restartAuction function is called just before an auction expires,

as a result, `bids[id].auctionDeadline = addUint48(uint48(now), totalAuctionLength);` increase the auction length again by 2 days.

Listing 86: SurplusAuctionHouse.sol (Lines 392,393,395)

```
392     function restartAuction(uint256 id) external {
393         require(bids[id].auctionDeadline < now, "
 ↳ RecyclingSurplusAuctionHouse/not-finished");
394         require(bids[id].bidExpiry == 0, "
 ↳ RecyclingSurplusAuctionHouse/bid-already-placed");
395         bids[id].auctionDeadline = addUint48(uint48(now),
 ↳ totalAuctionLength);
396         emit RestartAuction(id, bids[id].auctionDeadline);
397     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider adding an access control modifier or a privilege `require` check to the `restartAuction` function. In addition, add a counter to limit such repeated auctions to restart or set a maximum duration until an auction can continue if no bid is placed in the worst case.

Remediation Plan:

ACKNOWLEDGED: The team claims the above issue is intended behavior. Moreover, the team will add the intended behavior note in the README documentation. Furthermore, the team added that, similar to MakerDAO and RAI, the H2O protocol is optimized to maximize the final bid amount rather than minimize the auction time. Therefore, auctions can be extended if there is no bid as this does not affect the outcome of other auctions or the operation of other parts of the protocol.

3.17 (HAL-17) SELLER CAN MONOPOLIZE THE AUCTION BID PRICE - INFORMATIONAL

Description:

In the contract, SurplusAuctionHouse.sol observed no logic to stop a seller from being a buyer. Following HAL-16, a seller can restart if no bid is placed before the auction closes, resulting in an extension of the auction time by two days. Furthermore, If a valid buyer comes and sets a new bid, then just before the bid expires, a seller can raise the bid again via an external call to increaseBidSize and extend the bid duration by the following three hours. This scenario can continue, as the contract does not prevent the seller from becoming the buyer and controlling the auction's bid price.

Code Location:

- Default extension to bid expiry after a new bid is submitted

Listing 87: SurplusAuctionHouse.sol (Line 296)

```
296     uint48   public   bidDuration = 3 hours;
```

- No logic to stop seller from being a buyer now if a valid buyer comes and places a new bid, then just before bid expires, a seller can raise the bid again and extend the bid duration by next three hours. The scenario can go on until the auction expires.

Listing 88: SurplusAuctionHouse.sol (Lines 414,416,421)

```
404     function increaseBidSize(uint256 id, uint256 amountToBuy,
 ↳ uint256 bid) external {
405         require(contractEnabled == 1, "
 ↳ RecyclingSurplusAuctionHouse/contract-not-enabled");
```

```
406         require(bids[id].highBidder != address(0), "
 ↳ RecyclingSurplusAuctionHouse/high-bidder-not-set");
407         require(bids[id].bidExpiry > now || bids[id].bidExpiry ==
 ↳ 0, "RecyclingSurplusAuctionHouse/bid-already-expired");
408         require(bids[id].auctionDeadline > now, "
 ↳ RecyclingSurplusAuctionHouse/auction-already-expired");
409
410         require(amountToBuy == bids[id].amountToSell, "
 ↳ RecyclingSurplusAuctionHouse/amounts-not-matching");
411         require(bid > bids[id].bidAmount, "
 ↳ RecyclingSurplusAuctionHouse/bid-not-higher");
412         require(multiply(bid, ONE) >= multiply(bidIncrease, bids[
 ↳ id].bidAmount), "RecyclingSurplusAuctionHouse/insufficient-
 ↳ increase");
413
414         if (msg.sender != bids[id].highBidder) {
415             protocolToken.move(msg.sender, bids[id].highBidder,
 ↳ bids[id].bidAmount);
416             bids[id].highBidder = msg.sender;
417         }
418         protocolToken.move(msg.sender, address(this), bid - bids[
 ↳ id].bidAmount);
419
420         bids[id].bidAmount = bid;
421         bids[id].bidExpiry = addUint48(uint48(now), bidDuration);
422
423         emit IncreaseBidSize(id, msg.sender, amountToBuy, bid,
 ↳ bids[id].bidExpiry);
424     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider adding logic to validate seller and buyer addresses. Add a logic to manage the fees if the seller and buyer are the same.

Remediation Plan:

ACKNOWLEDGED: The team claims the above issue is intended behavior. Moreover, the team will add the intended behavior note in the README documentation. Furthermore, the team added that, the protocol intends to maximize the amount of capital raised through the auction mechanism rather than minimize the auction duration or restrict participation; therefore, limiting any means of increasing the bid price is undesirable.

3.18 (HAL-18) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In public functions, array arguments are immediately copied 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 called internally. Internal calls are passed internally by pointers to memory. Thus, the function expects its arguments being located in memory when the compiler generates the code for an internal function.
Furthermore, methods do not necessarily have to be public if they are only called within the contract-in such case they should be marked internal.

Code Location:

Below are smart contracts and their corresponding functions affected:
AccessManagementProposal.sol:
executeProposal() scheduleProposal()

CollateralStabilityFeeProposal.sol:
executeProposal()

ConfigLike.sol:
addAuthorization(address) initializeCollateralType(bytes32) modifyParameters(bytes32,bytes32,address) modifyParameters(bytes32,bytes32,uint256) modifyParameters(bytes32,uint256) modifyParameters(bytes32,uint256,uint256,address)

DebtCeilingProposal.sol:
executeProposal() scheduleProposal()

DeployCollateralAuctionThottler.sol:
execute(address,address,address)

DeployDebtPopperRewards.sol:
execute(address,address)

DeployDSMandWrapper.sol:
execute(address,address,address,bytes32,uint256)

DeployGlobalSettlement.sol:
execute(address)

DeployIncreasingDiscountCollateralHouse.sol:
execute(address,LiquidationEngineLike,bytes32,address)

DeployOSMandWrapper.sol:
execute(address,address,address)

DeployPIRateSetter.sol:
execute(address)

DeploySingleSpotDebtCeilingSetter.sol:
execute(address,address,address,bytes32,address)

DeployUniswapMedianizer.sol:
execute(address,address,address,address,address,uint256,uint256)

DeployUniswapTWAP.sol:
execute(address)

GlobalAuctionParamsProposal.sol:
executeProposal() scheduleProposal()

GlobalSettlementProposal.sol:
executeProposal() scheduleProposal()

GlobalStabilityFeeProposal.sol:
executeProposal()

LiquidationCRatioProposal.sol:
executeProposal() scheduleProposal()

LiquidationPenaltyProposal.sol:
executeProposal() scheduleProposal()

MerkleDistributionManagement.sol:

deployDistributor(address,bytes32,uint256,bool) dropDistributorAuth(address,uint256) getBackTokensFromDistributor(address,uint256,uint256) sendTokensToCustom(address,address,uint256) sendTokensToDistributor(address,uint256)

MultiDebtCeilingProposal.sol:
executeProposal() scheduleProposal()

NewCollateralProposal.sol:
executeProposal()

OldRateSetterLike.sol:
treasury()

OldTwapLike.sol:
treasury()

OracleRelayerLike.sol:
updateCollateralPrice(bytes32)

PauseLike.sol:
delay() executeTransaction(address,bytes32,bytes,uint256) scheduleTransaction(address,bytes32,bytes,uint256)

Proposal.sol:
executeProposal() newProposal(address,uint256,bytes) scheduleProposal()

ProposalFactory.sol:
newProposal(address,uint256,bytes)

SafetyCRatioProposal.sol:
executeProposal() scheduleProposal()

SecondaryTaxReceiversProposal.sol:
executeProposal()

SwapGlobalSettlement.sol:
execute(address,address)

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider, as much as possible, declaring external variables instead of public variables. As for best practice, you should use external if you expect that the function will only be called externally and use public if you need to call the function internally. To sum up, all can access to public functions, external functions only can be accessed externally, and internal functions can only be called within the contract.

Remediation Plan:

ACKNOWLEDGED: The H2O team acknowledged this issue.

3.19 (HAL-19) USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS - INFORMATIONAL

Description:

In the loop below, the variable `i` is incremented using `i++`. It is known that, in loops, using `++i` costs less gas per iteration than `i++`.

Code Location:

Listing 89: AccessManagementProposal.sol (Line 84)

```
73     function executeProposal() public {
74         require(!executed, "proposal-already-executed");
75
76         for (uint256 i = 0; i < gebModules.length; i++) {
77             bytes memory signature =
78                 abi.encodeWithSignature(
79                     (grantAccess[i] != 0) ? "addAuthorization(
↳ address,address)" : "removeAuthorization(address,address)",
80                     gebModules[i],
81                     addresses[i]
82             );
83
84             pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
85         }
86
87         executed = true;
88     }
```

Listing 90: AccessManagementProposal.sol (Line 69)

```
57     function scheduleProposal() public {
58         require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
59         earliestExecutionTime = now + PauseLike(pause).delay();
60
```

```
61          for (uint256 i = 0; i < gebModules.length; i++) {
62              bytes memory signature =
63                  abi.encodeWithSignature(
64                      (grantAccess[i] != 0) ? "addAuthorization(
↳ address,address)" : "removeAuthorization(address,address)",
65                      gebModules[i],
66                      addresses[i]
67              );
68
69              pause.scheduleTransaction(target, codeHash, signature,
↳  earliestExecutionTime);
70          }
71      }
```

Listing 91: CollateralStabilityFeeProposal.sol (Line 19)

```
16      function deploy(address taxCollector, bytes32[] calldata
↳ collateralTypes, uint256[] calldata stabilityFees) external {
17
18          for (uint i = 0; i < collateralTypes.length; i++)
19              ConfigLike(taxCollector).modifyParameters(
↳ collateralTypes[i], "stabilityFee", stabilityFees[i]);
20      }
```

Listing 92: LiquidationCRatioProposal.sol (Line 84)

```
72      function executeProposal() public {
73          require(!executed, "proposal-already-executed");
74
75          for (uint256 i = 0; i < collateralTypes.length; i++) {
76              bytes memory signature =
77                  abi.encodeWithSignature(
78                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
79                      oracleRelayer,
80                      collateralTypes[i],
81                      bytes32("liquidationCRatio"),
82                      liquidationCRatios[i]
83              );
84              pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
85          }
86
```

```
87          executed = true;
88      }
```

Listing 93: LiquidationCRatioProposal.sol (Line 68)

```
55      function scheduleProposal() public {
56          require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
57          earliestExecutionTime = now + PauseLike(pause).delay();
58
59          for (uint256 i = 0; i < collateralTypes.length; i++) {
60              bytes memory signature =
61                  abi.encodeWithSignature(
62                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
63                      oracleRelayer,
64                      collateralTypes[i],
65                      bytes32("liquidationCRatio"),
66                      liquidationCRatios[i]
67              );
68              pause.scheduleTransaction(target, codeHash, signature,
↳  earliestExecutionTime);
69          }
70      }
```

Listing 94: LiquidationPenaltyProposal.sol (Line 84)

```
72      function executeProposal() public {
73          require(!executed, "proposal-already-executed");
74
75          for (uint256 i = 0; i < collateralTypes.length; i++) {
76              bytes memory signature =
77                  abi.encodeWithSignature(
78                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
79                      liquidationEngine,
80                      collateralTypes[i],
81                      bytes32("liquidationPenalty"),
82                      liquidationPenalties[i]
83              );
84              pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
85          }
```

```
86
87          executed = true;
88      }
```

Listing 95: LiquidationPenaltyProposal.sol (Line 68)

```
55      function scheduleProposal() public {
56          require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
57          earliestExecutionTime = now + PauseLike(pause).delay();
58
59          for (uint256 i = 0; i < collateralTypes.length; i++) {
60              bytes memory signature =
61                  abi.encodeWithSignature(
62                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
63                      liquidationEngine,
64                      collateralTypes[i],
65                      bytes32("liquidationPenalty"),
66                      liquidationPenalties[i]
67              );
68              pause.scheduleTransaction(target, codeHash, signature,
↳  earliestExecutionTime);
69          }
70      }
```

Listing 96: MultiDebtCeilingProposal.sol (Line 85)

```
73      function executeProposal() public {
74          require(!executed, "proposal-already-executed");
75
76          for (uint256 i = 0; i < collateralTypes.length; i++) {
77              bytes memory signature =
78                  abi.encodeWithSignature(
79                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
80                      safeEngine,
81                      collateralTypes[i],
82                      bytes32("debtCeiling"),
83                      debtCeilings[i]
84              );
85              pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
```

```
86          }
87
88          executed = true;
89      }
```

Listing 97: MultiDebtCeilingProposal.sol (Line 69)

```
56      function scheduleProposal() public {
57          require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
58          earliestExecutionTime = now + PauseLike(pause).delay();
59
60          for (uint256 i = 0; i < collateralTypes.length; i++) {
61              bytes memory signature =
62                  abi.encodeWithSignature(
63                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
64                      safeEngine,
65                      collateralTypes[i],
66                      bytes32("debtCeiling"),
67                      debtCeilings[i]
68              );
69              pause.scheduleTransaction(target, codeHash, signature,
↳  earliestExecutionTime);
70          }
71      }
```

Listing 98: SafetyCRatioProposal.sol (Line 84)

```
72      function executeProposal() public {
73          require(!executed, "proposal-already-executed");
74
75          for (uint256 i = 0; i < collateralTypes.length; i++) {
76              bytes memory signature =
77                  abi.encodeWithSignature(
78                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
79                      oracleRelayer,
80                      collateralTypes[i],
81                      bytes32("safetyCRatio"),
82                      safetyCRatios[i]
83              );
```

```
84              pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
85          }
86
87          executed = true;
88      }
```

Listing 99: SafetyCRatioProposal.sol (Line 68)

```
55      function scheduleProposal() public {
56          require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
57          earliestExecutionTime = now + PauseLike(pause).delay();
58
59          for (uint256 i = 0; i < collateralTypes.length; i++) {
60              bytes memory signature =
61                  abi.encodeWithSignature(
62                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
63                      oracleRelayer,
64                      collateralTypes[i],
65                      bytes32("safetyCRatio"),
66                      safetyCRatios[i]
67              );
68              pause.scheduleTransaction(target, codeHash, signature,
↳  earliestExecutionTime);
69          }
70      }
```

Listing 100: SecondaryTaxReceiversProposal.sol (Line 24)

```
14      function deploy(
15          address taxCollector,
16          bytes32[] calldata collateralTypes,
17          uint256[] calldata positions,
18          uint256[] calldata percentages,
19          address[] calldata secondaryReceivers)
20          external
21      {
22
23          for (uint i = 0; i < collateralTypes.length; i++)
24              ConfigLike(taxCollector).modifyParameters(
↳ collateralTypes[i], positions[i], percentages[i],
```

```
↳ secondaryReceivers[i]);
25      }
```

Proof of Concept:

For example, based in the following test contract:

Listing 101: Test.sol

```
1  //SPDX-License-Identifier: MIT
2  pragma solidity 0.8.9;
3
4  contract test {
5      function postiincrement(uint256 iterations) public {
6          for (uint256 i = 0; i < iterations; i++) {
7          }
8      }
9      function preiincrement(uint256 iterations) public {
10         for (uint256 i = 0; i < iterations; ++i) {
11         }
12     }
13 }
```

We can see the difference in the gas costs:

```
>>> test_contract.postiincrement(1)
Transaction sent: 0x1ecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 44
  test.postiincrement confirmed   Block: 13622335   Gas used: 21620 (0.32%)

<Transaction '0x1ecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b'>
>>> test_contract.preiincrement(1)
Transaction sent: 0x205f09a4d2268de4c1a40f35bb2ec2847bf2ab8d584909b42c71a022b047614a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 45
  test.preiincrement confirmed   Block: 13622336   Gas used: 21593 (0.32%)

<Transaction '0x205f09a4d2268de4c1a40f35bb2ec2847bf2ab8d584909b42c71a022b047614a'>
>>> test_contract.postiincrement(10)
Transaction sent: 0x98c04430526a59b1cf947c114b62666a4417165947d31bf300cd6ae68328033
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 46
  test.postiincrement confirmed   Block: 13622337   Gas used: 22673 (0.34%)

<Transaction '0x98c04430526a59b1cf947c114b62666a4417165947d31bf300cd6ae68328033'>
>>> test_contract.preiincrement(10)
Transaction sent: 0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20e1de05
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 47
  test.preiincrement confirmed   Block: 13622338   Gas used: 22601 (0.34%)

<Transaction '0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20e1de05'>
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use `++i` instead of `i++` to increment the value of an `uint` variable inside a loop. This is not applicable outside of loops.

Remediation Plan:

ACKNOWLEDGED: The H2O team acknowledged this issue.

3.20 (HAL-20) CACHE ARRAY LENGTH IN FOR LOOPS CAN SAVE GAS - INFORMATIONAL

Description:

Reading array length at each iteration of the loop takes 6 gas (3 for mload and 3 to place memory_offset) in the stack. Caching the array length in the stack saves around 3 gas per iteration.

Code Location:

Listing 102: AccessManagementProposal.sol (Line 84)

```
73     function executeProposal() public {
74         require(!executed, "proposal-already-executed");
75
76         for (uint256 i = 0; i < gebModules.length; i++) {
77             bytes memory signature =
78                 abi.encodeWithSignature(
79                     (grantAccess[i] != 0) ? "addAuthorization(
↳ address,address)" : "removeAuthorization(address,address)",
80                     gebModules[i],
81                     addresses[i]
82             );
83
84             pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
85         }
86
87         executed = true;
88     }
```

Listing 103: AccessManagementProposal.sol (Line 69)

```
57     function scheduleProposal() public {
58         require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
59         earliestExecutionTime = now + PauseLike(pause).delay();
```

```

        for (uint256 i = 0; i < gebModules.length; i++) {
            bytes memory signature =
                abi.encodeWithSignature(
                    (grantAccess[i] != 0) ? "addAuthorization(address,address)" : "removeAuthorization(address,address)",
                    gebModules[i],
                    addresses[i]
            );

            pause.scheduleTransaction(target, codeHash, signature, earliestExecutionTime);
        }
    }
```

Listing 104: CollateralStabilityFeeProposal.sol (Line 19)

```
    function deploy(address taxCollector, bytes32[] calldata collateralTypes, uint256[] calldata stabilityFees) external {

        for (uint i = 0; i < collateralTypes.length; i++)
            ConfigLike(taxCollector).modifyParameters(collateralTypes[i], "stabilityFee", stabilityFees[i]);
    }
```

Listing 105: LiquidationCRatioProposal.sol (Line 84)

```
    function executeProposal() public {
        require(!executed, "proposal-already-executed");

        for (uint256 i = 0; i < collateralTypes.length; i++) {
            bytes memory signature =
                abi.encodeWithSignature(
                    "modifyParameters(address,bytes32,bytes32,uint256)",
                    oracleRelayer,
                    collateralTypes[i],
                    bytes32("liquidationCRatio"),
                    liquidationCRatios[i]
            );
            pause.executeTransaction(target, codeHash, signature, earliestExecutionTime);
        }
```

```
86
87          executed = true;
88      }
```

Listing 106: LiquidationCRatioProposal.sol (Line 68)

```
55      function scheduleProposal() public {
56          require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
57          earliestExecutionTime = now + PauseLike(pause).delay();
58
59          for (uint256 i = 0; i < collateralTypes.length; i++) {
60              bytes memory signature =
61                  abi.encodeWithSignature(
62                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
63                      oracleRelayer,
64                      collateralTypes[i],
65                      bytes32("liquidationCRatio"),
66                      liquidationCRatios[i]
67              );
68              pause.scheduleTransaction(target, codeHash, signature,
↳  earliestExecutionTime);
69          }
70      }
```

Listing 107: LiquidationPenaltyProposal.sol (Line 84)

```
72      function executeProposal() public {
73          require(!executed, "proposal-already-executed");
74
75          for (uint256 i = 0; i < collateralTypes.length; i++) {
76              bytes memory signature =
77                  abi.encodeWithSignature(
78                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
79                      liquidationEngine,
80                      collateralTypes[i],
81                      bytes32("liquidationPenalty"),
82                      liquidationPenalties[i]
83              );
84              pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
```

```
85          }
86
87          executed = true;
88      }
```

Listing 108: LiquidationPenaltyProposal.sol (Line 68)

```
55      function scheduleProposal() public {
56          require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
57          earliestExecutionTime = now + PauseLike(pause).delay();
58
59          for (uint256 i = 0; i < collateralTypes.length; i++) {
60              bytes memory signature =
61                  abi.encodeWithSignature(
62                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
63                      liquidationEngine,
64                      collateralTypes[i],
65                      bytes32("liquidationPenalty"),
66                      liquidationPenalties[i]
67              );
68              pause.scheduleTransaction(target, codeHash, signature,
↳  earliestExecutionTime);
69          }
70      }
```

Listing 109: MultiDebtCeilingProposal.sol (Line 85)

```
73      function executeProposal() public {
74          require(!executed, "proposal-already-executed");
75
76          for (uint256 i = 0; i < collateralTypes.length; i++) {
77              bytes memory signature =
78                  abi.encodeWithSignature(
79                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
80                      safeEngine,
81                      collateralTypes[i],
82                      bytes32("debtCeiling"),
83                      debtCeilings[i]
84              );
```

```
85              pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
86          }
87
88          executed = true;
89      }
```

Listing 110: MultiDebtCeilingProposal.sol (Line 69)

```
56      function scheduleProposal() public {
57          require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
58          earliestExecutionTime = now + PauseLike(pause).delay();
59
60          for (uint256 i = 0; i < collateralTypes.length; i++) {
61              bytes memory signature =
62                  abi.encodeWithSignature(
63                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
64                      safeEngine,
65                      collateralTypes[i],
66                      bytes32("debtCeiling"),
67                      debtCeilings[i]
68              );
69              pause.scheduleTransaction(target, codeHash, signature,
↳  earliestExecutionTime);
70          }
71      }
```

Listing 111: SafetyCRatioProposal.sol (Line 84)

```
72      function executeProposal() public {
73          require(!executed, "proposal-already-executed");
74
75          for (uint256 i = 0; i < collateralTypes.length; i++) {
76              bytes memory signature =
77                  abi.encodeWithSignature(
78                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
79                      oracleRelayer,
80                      collateralTypes[i],
81                      bytes32("safetyCRatio"),
82                      safetyCRatios[i]
```

```
83              );
84              pause.executeTransaction(target, codeHash, signature,
↳ earliestExecutionTime);
85          }
86
87          executed = true;
88      }
```

Listing 112: SafetyCRatioProposal.sol (Line 68)

```
55      function scheduleProposal() public {
56          require(earliestExecutionTime == 0, "proposal-already-
↳ scheduled");
57          earliestExecutionTime = now + PauseLike(pause).delay();
58
59          for (uint256 i = 0; i < collateralTypes.length; i++) {
60              bytes memory signature =
61                  abi.encodeWithSignature(
62                      "modifyParameters(address,bytes32,bytes32,
↳ uint256)",
63                      oracleRelayer,
64                      collateralTypes[i],
65                      bytes32("safetyCRatio"),
66                      safetyCRatios[i]
67              );
68              pause.scheduleTransaction(target, codeHash, signature,
↳  earliestExecutionTime);
69          }
70      }
```

Listing 113: SecondaryTaxReceiversProposal.sol (Line 24)

```
14      function deploy(
15          address taxCollector,
16          bytes32[] calldata collateralTypes,
17          uint256[] calldata positions,
18          uint256[] calldata percentages,
19          address[] calldata secondaryReceivers)
20          external
21      {
22
23          for (uint i = 0; i < collateralTypes.length; i++)
```

```
24              ConfigLike(taxCollector).modifyParameters(
↳ collateralTypes[i], positions[i], percentages[i],
↳ secondaryReceivers[i]);
25      }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider the cache array length. The sample code can be seen from the below.

Listing 114: SecondaryTaxReceiversProposal.sol (Line 24)

```
14      function deploy(
15          address taxCollector,
16          bytes32[] calldata collateralTypes,
17          uint256[] calldata positions,
18          uint256[] calldata percentages,
19          address[] calldata secondaryReceivers)
20          external
21      {
22          uint collateralTypeLength = collateralTypes.length;
23          for (uint i = 0; i < collateralTypeLength; i++)
24              ConfigLike(taxCollector).modifyParameters(
↳ collateralTypes[i], positions[i], percentages[i],
↳ secondaryReceivers[i]);
25      }
```

Remediation Plan:

ACKNOWLEDGED: The H2O team acknowledged this issue.

3.21 (HAL-21) UPGRADE AT LEAST PRAGMA 0.8.4 - INFORMATIONAL

Description:

Gas optimizations and additional safety checks are available for free when using newer compiler versions and the optimizer.

- Safemath by default from 0.8.0 (can be more gas efficient than library based safemath.)
- Low level inliner : from 0.8.2, leads to cheaper runtime gas. Especially relevant when the contract has small functions. For example, OpenZeppelin libraries typically have a lot of small helper functions and if they are not inlined, they cost an additional 20 to 40 gas because of 2 extra jump instructions and additional stack operations needed for function calls.
- Optimizer improvements in packed structs: Before 0.8.3, storing packed structs, in some cases, used an additional storage read operation. After EIP-2929, if the slot was already cold, this means unnecessary stack operations and extra deploy time costs. However, if the slot was already warm, this means additional cost of 100 gas alongside the same unnecessary stack operations and extra deploy time costs.
- Custom errors from 0.8.4, leads to cheaper deploy time cost and run time cost. Note: the run time cost is only relevant when the revert condition is met. In short, replace revert strings by custom errors.

Code Location:

All Contracts

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Upgrade **pragma** to at least 0.8.4.

Remediation Plan:

ACKNOWLEDGED: The H2O team acknowledged this issue.

3.22 (HAL-22) CONSTANT WITH EXPONENTIATION CAN BE IMPROVED - INFORMATIONAL

Description:

In the EVM, the exponentiation costs 10 gas, using the 1eX implementation can improve the gas usage.

Example Code Location:

Location

Listing 115: PRawPerSecondCalculator.sol (Lines 80,81)

```
80     uint256 internal constant TWENTY_SEVEN_DECIMAL_NUMBER = 10 **
↳ 27;
81     uint256 internal constant EIGHTEEN_DECIMAL_NUMBER     = 10 **
↳ 18;
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended that to change 10**X implementation with 1eX.

Remediation Plan:

SOLVED: The H2O team solved the above issue in the commit b49ec9ba27ad92e10f1a244078c343c2393773a0. As a result, the team replaced exponentiation with scientific notation.

Listing 116: Updated Code

```
    uint256 internal constant TWENTY_SEVEN_DECIMAL_NUMBER = 1e27;
    uint256 internal constant EIGHTEEN_DECIMAL_NUMBER     = 1e18;
```

3.23 (HAL-23) MISSING SUCCESS CHECK ON LOW LEVEL CALLS - INFORMATIONAL

Description:

In Solidity, the use of low-level calls could cause unexpected behavior if the call fails or an attacker provokes the call to fail. Then, it is a good practice not to use low-level calls to avoid a potentially exploitable behaviour. If the return value of a message call is not checked, the called contract may throw an exception.

###Code Location

Listing 117: GebProxyIncentivesActions.sol (Lines 123,157,196 ,234,249,272,299,335,450,486)

```
        // sending back any leftover tokens/eth, necessary to manage change from providing liquidity
        msg.sender.call{value: address(this).balance}("");
        systemCoin.transfer(msg.sender, systemCoin.balanceOf(address(this)));
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to check the return value of msg.sender.call() and update the logic similar to the below code.

Listing 118

```
        (bool success, ) = msg.sender.call{value: address(this).balance}("");
        if(success)
```

```
3         systemCoin.transfer(msg.sender, systemCoin.balanceOf(
↳ address(this)));
```

Remediation Plan:

SOLVED: The H2O team solved the above issue in the commit 593a0388dd719aa475d4c85a38c7fd6c05b56a8c. As a result, the team added the return value check that reverts if the return of funds fails.

Listing 119: Updated Code

```
1         ( bool success , ) = msg.sender.call{value: address(this).
↳ balance}("");
2         if (!success)
3             revert("GebProxyIncentivesActions/failed-to-return-eth
↳ ");
4         systemCoin.transfer(msg.sender, systemCoin.balanceOf(
↳ address(this)));
```

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

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. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Results:

```
StabilityFeeTreasuryLike is re-used:
        - lib/geb-fsm/lib/geb-treasury-reimbursement/src/reimbursement/NoSetupNoAuthIncreasingTreasuryReimbursement.sol#5-9
        - lib/geb-fsm/lib/geb-treasury-reimbursement/src/reimbursement/NoSetupIncreasingTreasuryReimbursement.sol#5-9
        - src/DeployOSMandWrapper.sol#10-13
        - lib/geb-fsm/lib/geb-treasury-reimbursement/src/reimbursement/NoSetupIncreasingTreasuryReimbursement.sol#5-9
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#name-reused
Function not found safeEngine
Impossible to generate IR for SwapGlobalSettlement.execute

ESMThresholdSetter is re-used:
        - lib/geb-esm-threshold-setter/src/ESMThresholdSetter.sol#12-120
        - lib/geb-deploy/lib/esm/src/ESM.sol#18-20
GlobalSettlementLike is re-used:
        - src/DeployGlobalSettlement.sol#90-96
        - lib/geb-deploy/lib/esm/src/ESM.sol#29-31
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#name-reused
Function not found setPerBlockAllowance
Impossible to generate IR for DeployDebtPopperRewards.execute

StabilityFeeTreasuryLike is re-used:
        - src/DeployDebtPopperRewards.sol#5-8
        - lib/geb-fsm/lib/geb-treasury-reimbursement/src/reimbursement/MandatoryFixedTreasuryReimbursement.sol#5-9
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#name-reused

GlobalSettlementProposal.scheduleProposal() (src/GlobalSettlementProposal.sol#47-55) uses a dangerous strict equality:
        - require(bool,string)(earliestExecutionTime == 0,proposal-already-scheduled) (src/GlobalSettlementProposal.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in GlobalSettlementProposal.executeProposal() (src/GlobalSettlementProposal.sol#57-66):
        External calls:
        - pause.executeTransaction(target,codeHash,signature,earliestExecutionTime) (src/GlobalSettlementProposal.sol#63)
        State variables written after the call(s):
        - executed = true (src/GlobalSettlementProposal.sol#65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

GlobalAuctionParamsProposal.scheduleProposal() (src/GlobalAuctionParamsProposal.sol#54-74) uses a dangerous strict equality:
        - require(bool,string)(earliestExecutionTime == 0,proposal-already-scheduled) (src/GlobalAuctionParamsProposal.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in GlobalAuctionParamsProposal.executeProposal() (src/GlobalAuctionParamsProposal.sol#76-97):
        External calls:
        - pause.executeTransaction(target,codeHash,signature,earliestExecutionTime) (src/GlobalAuctionParamsProposal.sol#86)
        - pause.executeTransaction(target,codeHash,signature,earliestExecutionTime) (src/GlobalAuctionParamsProposal.sol#94)
        State variables written after the call(s):
        - executed = true (src/GlobalAuctionParamsProposal.sol#96)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

```
GlobalSettlementProposal.constructor(address,address,address)._target (src/GlobalSettlementProposal.sol#36) lacks a zero-check on :
                - target = _target (src/GlobalSettlementProposal.sol#39)
GlobalSettlementProposal.constructor(address,address,address)._globalSettlement (src/GlobalSettlementProposal.sol#36) lacks a zero-check on :
                - globalSettlement = _globalSettlement (src/GlobalSettlementProposal.sol#40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

GlobalSettlementProposal.scheduleProposal() (src/GlobalSettlementProposal.sol#47-55) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(earliestExecutionTime == 0,proposal-already-scheduled) (src/GlobalSettlementProposal.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Proposal.constructor(address,uint256,bytes).target_ (src/Proposal.sol#29) lacks a zero-check on :
                - target = target_ (src/Proposal.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

GlobalAuctionParamsProposal.constructor(address,address,address,uint256,uint256)._target (src/GlobalAuctionParamsProposal.sol#40) lacks a zero-check on :
                - target = _target (src/GlobalAuctionParamsProposal.sol#45)
GlobalAuctionParamsProposal.constructor(address,address,address,uint256,uint256)._accountingEngine (src/GlobalAuctionParamsProposal.sol#40) lacks a zero-check
 on :
                - accountingEngine = _accountingEngine (src/GlobalAuctionParamsProposal.sol#46)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

GlobalAuctionParamsProposal.scheduleProposal() (src/GlobalAuctionParamsProposal.sol#54-74) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(earliestExecutionTime == 0,proposal-already-scheduled) (src/GlobalAuctionParamsProposal.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
SafetyCRatioProposal.constructor(address,address,address,bytes32[],uint256[])._target (src/SafetyCRatioProposal.sol#40) lacks a zero-check on :
                - target = _target (src/SafetyCRatioProposal.sol#46)
SafetyCRatioProposal.constructor(address,address,address,bytes32[],uint256[])._oracleRelayer (src/SafetyCRatioProposal.sol#40) lacks a zero-check on :
                - oracleRelayer = _oracleRelayer (src/SafetyCRatioProposal.sol#47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

SafetyCRatioProposal.scheduleProposal() (src/SafetyCRatioProposal.sol#55-70) has external calls inside a loop: pause.scheduleTransaction(target,codeHash,signa
ture,earliestExecutionTime) (src/SafetyCRatioProposal.sol#68)
SafetyCRatioProposal.executeProposal() (src/SafetyCRatioProposal.sol#72-88) has external calls inside a loop: pause.executeTransaction(target,codeHash,signatu
re,earliestExecutionTime) (src/SafetyCRatioProposal.sol#84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

SafetyCRatioProposal.scheduleProposal() (src/SafetyCRatioProposal.sol#55-70) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(earliestExecutionTime == 0,proposal-already-scheduled) (src/SafetyCRatioProposal.sol#56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

LiquidationPenaltyProposal.constructor(address,address,address,bytes32[],uint256[])._target (src/LiquidationPenaltyProposal.sol#40) lacks a zero-check on :
                - target = _target (src/LiquidationPenaltyProposal.sol#46)
LiquidationPenaltyProposal.constructor(address,address,address,bytes32[],uint256[])._liquidationEngine (src/LiquidationPenaltyProposal.sol#40) lacks a zero-ch
eck on :
                - liquidationEngine = _liquidationEngine (src/LiquidationPenaltyProposal.sol#47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

LiquidationPenaltyProposal.scheduleProposal() (src/LiquidationPenaltyProposal.sol#55-70) has external calls inside a loop: pause.scheduleTransaction(target,co
deHash,signature,earliestExecutionTime) (src/LiquidationPenaltyProposal.sol#68)
LiquidationPenaltyProposal.executeProposal() (src/LiquidationPenaltyProposal.sol#72-88) has external calls inside a loop: pause.executeTransaction(target,code
Hash,signature,earliestExecutionTime) (src/LiquidationPenaltyProposal.sol#84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

delay() should be declared external:
        - PauseLike.delay() (src/GlobalSettlementProposal.sol#17)
scheduleTransaction(address,bytes32,bytes,uint256) should be declared external:
        - PauseLike.scheduleTransaction(address,bytes32,bytes,uint256) (src/GlobalSettlementProposal.sol#18)
executeTransaction(address,bytes32,bytes,uint256) should be declared external:
        - PauseLike.executeTransaction(address,bytes32,bytes,uint256) (src/GlobalSettlementProposal.sol#19)
scheduleProposal() should be declared external:
        - GlobalSettlementProposal.scheduleProposal() (src/GlobalSettlementProposal.sol#47-55)
executeProposal() should be declared external:
        - GlobalSettlementProposal.executeProposal() (src/GlobalSettlementProposal.sol#57-66)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

executeProposal() should be declared external:
        - Proposal.executeProposal() (src/Proposal.sol#35-39)
newProposal(address,uint256,bytes) should be declared external:
        - ProposalFactory.newProposal(address,uint256,bytes) (src/Proposal.sol#43-45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

delay() should be declared external:
        - PauseLike.delay() (src/GlobalAuctionParamsProposal.sol#17)
scheduleTransaction(address,bytes32,bytes,uint256) should be declared external:
        - PauseLike.scheduleTransaction(address,bytes32,bytes,uint256) (src/GlobalAuctionParamsProposal.sol#18)
executeTransaction(address,bytes32,bytes,uint256) should be declared external:
        - PauseLike.executeTransaction(address,bytes32,bytes,uint256) (src/GlobalAuctionParamsProposal.sol#19)
scheduleProposal() should be declared external:
        - GlobalAuctionParamsProposal.scheduleProposal() (src/GlobalAuctionParamsProposal.sol#54-74)
executeProposal() should be declared external:
        - GlobalAuctionParamsProposal.executeProposal() (src/GlobalAuctionParamsProposal.sol#76-97)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
GebMath.rpower(uint256,uint256,uint256) (geb-treasury-reimbursement/src/math/GebMath.sol#39-61) performs a multiplication on the result of a division:
        -x = xxRound_rpower_asm_0 / base (geb-treasury-reimbursement/src/math/GebMath.sol#50)
        -zx_rpower_asm_0 = z * x (geb-treasury-reimbursement/src/math/GebMath.sol#52)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in SingleDebtFloorAdjuster.manualRecomputeCollateralDebtFloor() (src/SingleDebtFloorAdjuster.sol#256-261):
        External calls:
        - nextCollateralFloor = getNextCollateralFloor() (src/SingleDebtFloorAdjuster.sol#258)
                - redemptionPrice = oracleRelayer.redemptionPrice() (src/SingleDebtFloorAdjuster.sol#280)
        State variables written after the call(s):
        - lastManualUpdateTime = now (src/SingleDebtFloorAdjuster.sol#259)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SingleDebtFloorAdjuster.modifyParameters(bytes32,uint256) (src/SingleDebtFloorAdjuster.sol#175-217) contains a tautology or contradiction:
        - require(bool,string)(val >= 0,SingleDebtFloorAdjuster/invalid-call-gap-length) (src/SingleDebtFloorAdjuster.sol#193)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction

IncreasingTreasuryReimbursement.rewardCaller(address,uint256).revertReason (geb-treasury-reimbursement/src/reimbursement/IncreasingTreasuryReimbursement.sol#1
51) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

SingleDebtFloorAdjuster.constructor(address,address,address,address,address,bytes32,uint256,uint256,uint256,uint256,uint256,uint256,uint256) (src/SingleDebtFl
oorAdjuster.sol#83-130) ignores return value by oracleRelayer.redemptionPrice() (src/SingleDebtFloorAdjuster.sol#120)
SingleDebtFloorAdjuster.modifyParameters(bytes32,address) (src/SingleDebtFloorAdjuster.sol#150-169) ignores return value by oracleRelayer.redemptionPrice() (s
rc/SingleDebtFloorAdjuster.sol#157)
SingleDebtFloorAdjuster.modifyParameters(bytes32,address) (src/SingleDebtFloorAdjuster.sol#150-169) ignores return value by gasPriceOracle.read() (src/SingleD
ebtFloorAdjuster.sol#161)
SingleDebtFloorAdjuster.modifyParameters(bytes32,address) (src/SingleDebtFloorAdjuster.sol#150-169) ignores return value by ethPriceOracle.read() (src/SingleD
ebtFloorAdjuster.sol#165)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Variable 'IncreasingTreasuryReimbursement.rewardCaller(address,uint256).revertReason (geb-treasury-reimbursement/src/reimbursement/IncreasingTreasuryReimburse
ment.sol#151)' in IncreasingTreasuryReimbursement.rewardCaller(address,uint256) (geb-treasury-reimbursement/src/reimbursement/IncreasingTreasuryReimbursement.
sol#143-154) potentially used before declaration: FailRewardCaller(revertReason,finalFeeReceiver,reward) (geb-treasury-reimbursement/src/reimbursement/Increas
ingTreasuryReimbursement.sol#152)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in SingleDebtFloorAdjuster.constructor(address,address,address,address,address,bytes32,uint256,uint256,uint256,uint256,uint256,uint256,uint256) (sr
c/SingleDebtFloorAdjuster.sol#83-130):
        External calls:
        - oracleRelayer.redemptionPrice() (src/SingleDebtFloorAdjuster.sol#120)
        Event emitted after the call(s):
        - AddManualSetter(msg.sender) (src/SingleDebtFloorAdjuster.sol#122)
        - ModifyParameters(oracleRelayer,oracleRelayer_) (src/SingleDebtFloorAdjuster.sol#123)
        - ModifyParameters(gasPriceOracle,gasPriceOracle_) (src/SingleDebtFloorAdjuster.sol#124)
        - ModifyParameters(ethPriceOracle,ethPriceOracle_) (src/SingleDebtFloorAdjuster.sol#125)
        - ModifyParameters(gasAmountForLiquidation,gasAmountForLiquidation) (src/SingleDebtFloorAdjuster.sol#126)
        - ModifyParameters(updateDelay,updateDelay) (src/SingleDebtFloorAdjuster.sol#127)
        - ModifyParameters(maxDebtFloor,maxDebtFloor) (src/SingleDebtFloorAdjuster.sol#128)
        - ModifyParameters(minDebtFloor,minDebtFloor) (src/SingleDebtFloorAdjuster.sol#129)
Reentrancy in SingleDebtFloorAdjuster.manualRecomputeCollateralDebtFloor() (src/SingleDebtFloorAdjuster.sol#256-261):
        External calls:
        - nextCollateralFloor = getNextCollateralFloor() (src/SingleDebtFloorAdjuster.sol#258)
                - redemptionPrice = oracleRelayer.redemptionPrice() (src/SingleDebtFloorAdjuster.sol#280)
        - setFloor(nextCollateralFloor) (src/SingleDebtFloorAdjuster.sol#260)
                - safeEngine.modifyParameters(collateralName,debtFloor,nextDebtFloor) (src/SingleDebtFloorAdjuster.sol#226)
        Event emitted after the call(s):
        - UpdateFloor(nextDebtFloor) (src/SingleDebtFloorAdjuster.sol#227)
                - setFloor(nextCollateralFloor) (src/SingleDebtFloorAdjuster.sol#260)
Reentrancy in SingleDebtFloorAdjuster.modifyParameters(bytes32,address) (src/SingleDebtFloorAdjuster.sol#150-169):
        External calls:
        - oracleRelayer.redemptionPrice() (src/SingleDebtFloorAdjuster.sol#157)
        Event emitted after the call(s):
```

```
IncreasingTreasuryReimbursement.getCallerReward(uint256,uint256) (geb-treasury-reimbursement/src/reimbursement/IncreasingTreasuryReimbursement.sol#108-137) us
es timestamp for comparisons
        Dangerous comparisons:
        - either(timeOfLastUpdate >= now,nullRewards) (geb-treasury-reimbursement/src/reimbursement/IncreasingTreasuryReimbursement.sol#111)
SingleDebtFloorAdjuster.modifyParameters(bytes32,uint256) (src/SingleDebtFloorAdjuster.sol#175-217) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(val > now,SingleDebtFloorAdjuster/invalid-update-time) (src/SingleDebtFloorAdjuster.sol#205)
SingleDebtFloorAdjuster.recomputeCollateralDebtFloor(address) (src/SingleDebtFloorAdjuster.sol#235-252) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(lastUpdateTime < now,SingleDebtFloorAdjuster/update-time-in-the-future) (src/SingleDebtFloorAdjuster.sol#237)
        - require(bool,string)(either(subtract(now,lastUpdateTime) >= updateDelay,lastUpdateTime == 0),SingleDebtFloorAdjuster/wait-more) (src/SingleDebtFloor
Adjuster.sol#239)
SingleDebtFloorAdjuster.manualRecomputeCollateralDebtFloor() (src/SingleDebtFloorAdjuster.sol#256-261) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now > lastManualUpdateTime,SingleDebtFloorAdjuster/cannot-update-twice-same-block) (src/SingleDebtFloorAdjuster.sol#257)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

ray(uint256) should be declared external:
        - GebMath.ray(uint256) (geb-treasury-reimbursement/src/math/GebMath.sol#7-9)
rad(uint256) should be declared external:
        - GebMath.rad(uint256) (geb-treasury-reimbursement/src/math/GebMath.sol#10-12)
addition(uint256,uint256) should be declared external:
        - GebMath.addition(uint256,uint256) (geb-treasury-reimbursement/src/math/GebMath.sol#16-19)
rdivide(uint256,uint256) should be declared external:
        - GebMath.rdivide(uint256,uint256) (geb-treasury-reimbursement/src/math/GebMath.sol#30-32)
wdivide(uint256,uint256) should be declared external:
        - GebMath.wdivide(uint256,uint256) (geb-treasury-reimbursement/src/math/GebMath.sol#33-35)
wmultiply(uint256,uint256) should be declared external:
        - GebMath.wmultiply(uint256,uint256) (geb-treasury-reimbursement/src/math/GebMath.sol#36-38)
collateralTypes(bytes32) should be declared external:
        - SAFEEngineLike.collateralTypes(bytes32) (src/SingleDebtFloorAdjuster.sol#11-16)
```

```
openLockTokenCollateralGenerateDebtAndProtectSAFE(address,address,address,address,bytes32,uint256,uint256,bool,address,address) should be declared external:
        - GebProxyActions.openLockTokenCollateralGenerateDebtAndProtectSAFE(address,address,address,address,bytes32,uint256,uint256,bool,address,address) (src
/GebProxyActions.sol#991-1006)
openLockGNTGenerateDebtAndProtectSAFE(address,address,address,address,bytes32,uint256,uint256,address,address) should be declared external:
        - GebProxyActions.openLockGNTGenerateDebtAndProtectSAFE(address,address,address,address,bytes32,uint256,uint256,address,address) (src/GebProxyActions.
sol#1027-1048)
repayAllDebtAndFreeETH(address,address,address,uint256,uint256) should be declared external:
        - GebProxyActions.repayAllDebtAndFreeETH(address,address,address,uint256,uint256) (src/GebProxyActions.sol#1050-1079)
repayAllDebtAndFreeTokenCollateral(address,address,address,uint256,uint256) should be declared external:
        - GebProxyActions.repayAllDebtAndFreeTokenCollateral(address,address,address,uint256,uint256) (src/GebProxyActions.sol#1106-1132)
freeTokenCollateral(address,address,address,uint256) should be declared external:
        - GebProxyActionsGlobalSettlement.freeTokenCollateral(address,address,address,uint256) (src/GebProxyActions.sol#1180-1189)
prepareCoinsForRedeeming(address,address,uint256) should be declared external:
        - GebProxyActionsGlobalSettlement.prepareCoinsForRedeeming(address,address,uint256) (src/GebProxyActions.sol#1191-1203)
redeemETH(address,address,bytes32,uint256) should be declared external:
        - GebProxyActionsGlobalSettlement.redeemETH(address,address,bytes32,uint256) (src/GebProxyActions.sol#1205-1219)
redeemTokenCollateral(address,address,bytes32,uint256) should be declared external:
        - GebProxyActionsGlobalSettlement.redeemTokenCollateral(address,address,bytes32,uint256) (src/GebProxyActions.sol#1221-1231)
deposit(address,address,uint256) should be declared external:
        - GebProxyActionsCoinSavingsAccount.deposit(address,address,uint256) (src/GebProxyActions.sol#1236-1252)
withdraw(address,address,uint256) should be declared external:
        - GebProxyActionsCoinSavingsAccount.withdraw(address,address,uint256) (src/GebProxyActions.sol#1254-1278)
withdrawAll(address,address) should be declared external:
        - GebProxyActionsCoinSavingsAccount.withdrawAll(address,address) (src/GebProxyActions.sol#1280-1297)
openLockETHLeverage(address,address,address,address,address,address,address,address,bytes32,uint256) should be declared external:
        - GebProxyLeverageActions.openLockETHLeverage(address,address,address,address,address,address,bytes32,uint256) (src/GebProxyLeverageActions.so
l#164-189)
lockETHLeverage(address,address,address,address,address,address,address,address,bytes32,uint256,uint256) should be declared external:
        - GebProxyLeverageActions.lockETHLeverage(address,address,address,address,address,address,address,address,bytes32,uint256,uint256) (src/GebProxyLeverageAction
s.sol#202-227)
flashDeleverageFreeETH(address,address,address,address,address,address,address,address,bytes32,uint256,uint256) should be declared external:
        - GebProxyLeverageActions.flashDeleverageFreeETH(address,address,address,address,address,address,address,address,bytes32,uint256,uint256) (src/GebProxyLeverag
```

```
BasicActions.freeETH(address,address,uint256,uint256) (src/GebProxyActions.sol#618-634) sends eth to arbitrary user
        Dangerous calls:
        - msg.sender.transfer(wad) (src/GebProxyActions.sol#633)
BasicActions.exitETH(address,address,uint256,uint256) (src/GebProxyActions.sol#642-656) sends eth to arbitrary user
        Dangerous calls:
        - msg.sender.transfer(wad) (src/GebProxyActions.sol#655)
BasicActions.repayDebtAndFreeETH(address,address,address,uint256,uint256,uint256) (src/GebProxyActions.sol#732-743) sends eth to arbitrary user
        Dangerous calls:
        - msg.sender.transfer(collateralWad) (src/GebProxyActions.sol#742)
GebProxyActions.repayAllDebtAndFreeETH(address,address,address,uint256,uint256) (src/GebProxyActions.sol#1050-1079) sends eth to arbitrary user
        Dangerous calls:
        - msg.sender.transfer(collateralWad) (src/GebProxyActions.sol#1078)
GebProxyActionsGlobalSettlement.freeETH(address,address,address,uint256) (src/GebProxyActions.sol#1165-1178) sends eth to arbitrary user
        Dangerous calls:
        - msg.sender.transfer(wad) (src/GebProxyActions.sol#1177)
GebProxyActionsGlobalSettlement.redeemETH(address,address,bytes32,uint256) (src/GebProxyActions.sol#1205-1219) sends eth to arbitrary user
        Dangerous calls:
        - msg.sender.transfer(collateralWad) (src/GebProxyActions.sol#1218)
GebProxyIncentivesActions.openLockETHGenerateDebtProvideLiquidityStake(address,address,address,address,address,address,bytes32,uint256,uint256,uint256[2]) (sr
c/GebProxyIncentivesActions.sol#172-198) sends eth to arbitrary user
        Dangerous calls:
        - msg.sender.call{value: address(this).balance}() (src/GebProxyIncentivesActions.sol#196)
GebProxyIncentivesActions.lockETHGenerateDebtProvideLiquidityStake(address,address,address,address,address,address,uint256,uint256,uint256,uint256[2]) (src/Ge
bProxyIncentivesActions.sol#211-236) sends eth to arbitrary user
        Dangerous calls:
        - msg.sender.call{value: address(this).balance}() (src/GebProxyIncentivesActions.sol#234)
emCoin.transferFrom(msg.sender,address(this),wad) (src/GebProxyIncentivesActions.sol#245)
GebProxyIncentivesActions.provideLiquidityUniswap(address,address,uint256,uint256[2]) (src/GebProxyIncentivesActions.sol#243-251) ignores return value by syst
emCoin.transfer(msg.sender,systemCoin.balanceOf(address(this))) (src/GebProxyIncentivesActions.sol#250)
GebProxyIncentivesActions.provideLiquidityStake(address,address,address,uint256,uint256[2]) (src/GebProxyIncentivesActions.sol#258-274) ignores return value b
y systemCoin.transferFrom(msg.sender,address(this),wad) (src/GebProxyIncentivesActions.sol#266)
GebProxyIncentivesActions.provideLiquidityStake(address,address,address,uint256,uint256[2]) (src/GebProxyIncentivesActions.sol#258-274) ignores return value b
y systemCoin.transfer(msg.sender,systemCoin.balanceOf(address(this))) (src/GebProxyIncentivesActions.sol#273)
GebProxyIncentivesActions.generateDebtAndProvideLiquidityUniswap(address,address,address,address,uint256,uint256,uint256[2]) (src/GebProxyIncentivesActions.so
l#284-301) ignores return value by systemCoin.transfer(msg.sender,systemCoin.balanceOf(address(this))) (src/GebProxyIncentivesActions.sol#300)
GebProxyIncentivesActions.stakeInMine(address,uint256) (src/GebProxyIncentivesActions.sol#306-309) ignores return value by DSTokenLike(GebIncentivesLike(incen
tives).stakingToken()).transferFrom(msg.sender,address(this),wad) (src/GebProxyIncentivesActions.sol#307)
GebProxyIncentivesActions.generateDebtAndProvideLiquidityStake(address,address,address,address,address,uint256,uint256,uint256[2]) (src/GebProxyIncentivesActi
ons.sol#319-337) ignores return value by systemCoin.transfer(msg.sender,systemCoin.balanceOf(address(this))) (src/GebProxyIncentivesActions.sol#336)
GebProxyIncentivesActions.getRewards(address) (src/GebProxyIncentivesActions.sol#341-346) ignores return value by rewardToken.transfer(msg.sender,rewardToken.
balanceOf(address(this))) (src/GebProxyIncentivesActions.sol#345)
GebProxyIncentivesActions.exitMine(address) (src/GebProxyIncentivesActions.sol#350-357) ignores return value by rewardToken.transfer(msg.sender,rewardToken.ba
lanceOf(address(this))) (src/GebProxyIncentivesActions.sol#356)
GebProxyIncentivesActions.exitMine(address) (src/GebProxyIncentivesActions.sol#350-357) ignores return value by lpToken.transfer(msg.sender,lpToken.balanceOf(
address(this))) (src/GebProxyIncentivesActions.sol#356)
GebProxyIncentivesActions.migrateCampaign(address,address) (src/GebProxyIncentivesActions.sol#362-373) ignores return value by rewardToken.transfer(msg.sender
,rewardToken.balanceOf(address(this))) (src/GebProxyIncentivesActions.sol#372)
GebProxyIncentivesActions.withdrawFromMine(address,uint256) (src/GebProxyIncentivesActions.sol#378-383) ignores return value by lpToken.transfer(msg.sender,lp
Token.balanceOf(address(this))) (src/GebProxyIncentivesActions.sol#382)
GebProxyIncentivesActions.withdrawAndHarvest(address,uint256) (src/GebProxyIncentivesActions.sol#388-395) ignores return value by lpToken.transfer(msg.sender,
lpToken.balanceOf(address(this))) (src/GebProxyIncentivesActions.sol#394)
GebProxyIncentivesActions.withdrawHarvestRemoveLiquidity(address,address,address,uint256,uint256[2]) (src/GebProxyIncentivesActions.sol#403-411) ignores retur
n value by rewardToken.transfer(msg.sender,rewardToken.balanceOf(address(this))) (src/GebProxyIncentivesActions.sol#409)
GebProxyIncentivesActions.removeLiquidityUniswap(address,address,uint256,uint256[2]) (src/GebProxyIncentivesActions.sol#418-421) ignores return value by DSTok
enLike(getWethPair(uniswapRouter,systemCoin)).transferFrom(msg.sender,address(this),value) (src/GebProxyIncentivesActions.sol#419)
GebProxyIncentivesActions.withdrawRemoveLiquidityRepayDebt(address,address,uint256,address,uint256,address,uint256[2]) (src/GebProxyIncentivesActions.sol#441-
451) ignores return value by rewardToken.transfer(msg.sender,rewardToken.balanceOf(address(this))) (src/GebProxyIncentivesActions.sol#449)
GebProxyIncentivesActions.exitAndRemoveLiquidity(address,address,address,uint256[2]) (src/GebProxyIncentivesActions.sol#458-465) ignores return value by rewar
dToken.transfer(msg.sender,rewardToken.balanceOf(address(this))) (src/GebProxyIncentivesActions.sol#463)
```

```
DSAuth.setOwner(address).owner_ (ds-auth/auth.sol#37) lacks a zero-check on :
                - owner = owner_ (ds-auth/auth.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

DSAuth.setOwner(address).owner_ (ds-auth/auth.sol#37) lacks a zero-check on :
                - owner = owner_ (ds-auth/auth.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

DSAuth.setOwner(address).owner_ (ds-auth/auth.sol#37) lacks a zero-check on :
                - owner = owner_ (ds-auth/auth.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

GebProxySaviourActions.transferTokensToCaller(address[]) (src/GebProxySaviourActions.sol#55-62) has external calls inside a loop: selfBalance = DSTokenLike(to
kens[i]).balanceOf(address(this)) (src/GebProxySaviourActions.sol#57)
GebProxySaviourActions.transferTokensToCaller(address[]) (src/GebProxySaviourActions.sol#55-62) has external calls inside a loop: DSTokenLike(tokens[i]).trans
fer(msg.sender,selfBalance) (src/GebProxySaviourActions.sol#59)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

DSAuth.setOwner(address).owner_ (ds-auth/auth.sol#37) lacks a zero-check on :
                - owner = owner_ (ds-auth/auth.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

GebProxyDebtAuctionActions.decreaseSoldAmount(address,address,uint256,uint256) (src/GebProxyAuctionActions.sol#114-133) uses timestamp for comparisons
        Dangerous comparisons:
        - both(both(auctionDeadline < now,bidExpiry == 0),auctionDeadline > 0) (src/GebProxyAuctionActions.sol#128)
GebProxySurplusAuctionActions.increaseBidSize(address,uint256,uint256) (src/GebProxyAuctionActions.sol#187-200) uses timestamp for comparisons
        Dangerous comparisons:
        - auctionDeadline < now && bidExpiry == 0 && auctionDeadline > 0 (src/GebProxyAuctionActions.sol#195)
GebProxyStakedTokenAuctionActions.increaseBidSize(address,address,uint256,uint256) (src/GebProxyAuctionActions.sol#254-273) uses timestamp for comparisons
        Dangerous comparisons:
        - both(both(auctionDeadline < now,bidExpiry == 0),auctionDeadline > 0) (src/GebProxyAuctionActions.sol#268)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Low level call in GebProxyIncentivesActions.openLockETHGenerateDebtProvideLiquidityUniswap(address,address,address,address,address,bytes32,uint256,uint256,uin
t256[2]) (src/GebProxyIncentivesActions.sol#102-125):
        - msg.sender.call{value: address(this).balance}() (src/GebProxyIncentivesActions.sol#123)
Low level call in GebProxyIncentivesActions.lockETHGenerateDebtProvideLiquidityUniswap(address,address,address,address,address,uint256,uint256,uint256,uint256
[2]) (src/GebProxyIncentivesActions.sol#137-159):
        - msg.sender.call{value: address(this).balance}() (src/GebProxyIncentivesActions.sol#157)
Low level call in GebProxyIncentivesActions.openLockETHGenerateDebtProvideLiquidityStake(address,address,address,address,address,address,bytes32,uint256,uint2
56,uint256[2]) (src/GebProxyIncentivesActions.sol#172-198):
        - msg.sender.call{value: address(this).balance}() (src/GebProxyIncentivesActions.sol#196)
Low level call in GebProxyIncentivesActions.lockETHGenerateDebtProvideLiquidityStake(address,address,address,address,address,address,uint256,uint256,uint256,u
int256[2]) (src/GebProxyIncentivesActions.sol#211-236):
        - msg.sender.call{value: address(this).balance}() (src/GebProxyIncentivesActions.sol#234)
Low level call in GebProxyIncentivesActions.provideLiquidityUniswap(address,address,uint256,uint256[2]) (src/GebProxyIncentivesActions.sol#243-251):
        - msg.sender.call{value: address(this).balance}() (src/GebProxyIncentivesActions.sol#249)
Low level call in GebProxyIncentivesActions.provideLiquidityStake(address,address,address,uint256,uint256[2]) (src/GebProxyIncentivesActions.sol#258-274):
        - msg.sender.call{value: address(this).balance}() (src/GebProxyIncentivesActions.sol#272)
Low level call in GebProxyIncentivesActions.generateDebtAndProvideLiquidityUniswap(address,address,address,address,uint256,uint256,uint256[2]) (src/GebProxyIn
centivesActions.sol#284-301):
        - msg.sender.call{value: address(this).balance}() (src/GebProxyIncentivesActions.sol#299)
Low level call in GebProxyIncentivesActions.generateDebtAndProvideLiquidityStake(address,address,address,address,address,uint256,uint256,uint256[2]) (src/GebP
roxyIncentivesActions.sol#319-337):
        - msg.sender.call{value: address(this).balance}() (src/GebProxyIncentivesActions.sol#335)
Low level call in GebProxyIncentivesActions.withdrawRemoveLiquidityRepayDebt(address,address,uint256,address,uint256,address,uint256[2]) (src/GebProxyIncentiv
esActions.sol#441-451):
        - msg.sender.call{value: address(this).balance}() (src/GebProxyIncentivesActions.sol#450)
Low level call in GebProxyIncentivesActions.exitRemoveLiquidityRepayDebt(address,address,uint256,address,address,uint256[2]) (src/GebProxyIncentivesActions.so
l#474-487):
        - msg.sender.call{value: address(this).balance}() (src/GebProxyIncentivesActions.sol#486)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

According to the test results, some findings found by these tools were considered as false positives, while some of these findings were real security concerns. All relevant findings were reviewed by the auditors and relevant findings addressed in the report as security concerns.

4.2 AUTOMATED SECURITY SCAN

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. Only security-related findings are shown below.

Results:

Report for src/multi/MultiStabilityFeeTreasury.sol

Line	SWC Title	Severity	Short Description
422	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
425	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
426	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.

Report for src/multi/MultiIncreasingDiscountCollateralAuctionHouse.sol

Line	SWC Title	Severity	Short Description
307	(SWC-123) Requirement Violation	Low	Requirement violation.

Report for src/single/GlobalSettlement.sol

Line	SWC Title	Severity	Short Description
295	(SWC-107) Reentrancy	Low	Read of persistent state following external call
295	(SWC-113) DoS with Failed Call	Low	Multiple calls are executed in the same transaction.
297	(SWC-107) Reentrancy	Low	Read of persistent state following external call
300	(SWC-107) Reentrancy	Low	Read of persistent state following external call
300	(SWC-107) Reentrancy	Low	Write to persistent state following external call
301	(SWC-107) Reentrancy	Low	Read of persistent state following external call
301	(SWC-107) Reentrancy	Low	Write to persistent state following external call
302	(SWC-107) Reentrancy	Low	Read of persistent state following external call

Report for src/single/CoinSavingsAccount.sol

Line	SWC Title	Severity	Short Description
111	(SWC-116) Timestamp Dependence	Low	A control flow decision is made based on The block.timestamp environment variable.
113	(SWC-116) Timestamp Dependence	Low	A control flow decision is made based on The block.timestamp environment variable.
119	(SWC-116) Timestamp Dependence	Low	A control flow decision is made based on The block.timestamp environment variable.
137	(SWC-116) Timestamp Dependence	Low	A control flow decision is made based on The block.timestamp environment variable.

All relevant valid findings were found in the manual code review.

THANK YOU FOR CHOOSING

// HALBORN