Savvy - DeFi

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **December 30th, 2022 - January 31st, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	01/18/2023	Josep Bové
0.2	Document Updates	01/31/2023	Josep Bové
0.3	Draft Review	02/02/2023	Gokberk Gulgun
0.4	Draft Review	02/02/2023	Gabi Urrutia
0.5	Document Updates	02/23/2023	Josep Bové
1.0	Remediation Plan	02/21/2023	Josep Bové
1.1	Remediation Plan Review	02/23/2023	Gokberk Gulgun
1.2	Remediation Plan Review	02/23/2023	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

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Savvy engaged Halborn to conduct a security audit on their smart contracts beginning on December 30th, 2022 and ending on January 31st, 2023 . The security assessment was scoped to the smart contracts provided in the savvy GitHub repository. Commit hashes and further details can be found in the Scope section of this report.

1.2 AUDIT SUMMARY

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

The purpose of this audit is to:

- Ensure that 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 Savvy 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 code and can quickly identify items that do not follow the security best practices. The following phases and associated tools were used during 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 hot-spots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment. (Brownie, 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.

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

1.4 SCOPE

1. IN-SCOPE TREE & COMMIT :

Code repositories:

1. Savvy Defi

- Repository: savvy
- Commit ID: f0dd94f3f7cbe6b8ea2b9a9a8b744eaeb4c1be81

Out-of-scope:

- Third-party libraries and dependencies.

2. REMEDIATION COMMITS:

LATEST COMMIT ID : 9843e94c709aef44a4bfafc7e238c24dba3a2fe7

- 65140e9b2859285d98ec1c3945309ab189689d57
- 553366534cceb768b66f6f9985df0a600ee87e89
- fcbe31d40fe606b15ecf79d62fa36e900f8705b1
- a8e3b8f2c9996e7da5a0859eff8e2030761be152
- d34d6e9da09c3d40a01303d15c30f30b6c1a91e4
- a29168360102c53f4b5dd1993457050acf77b1c4
- 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	14	11	10

EXECUTIVE OVERVIEW

LIKELIHOOD

IMPACT

(HAL-02) (HAL-04) (HAL-10)	(HAL-07) (HAL-09)			
	(HAL-06) (HAL-12)	(HAL-01) (HAL-11)		
(HAL-17) (HAL-22) (HAL-23)	(HAL-18) (HAL-19) (HAL-20) (HAL-21)	(HAL-03) (HAL-05) (HAL-08) (HAL-13) (HAL-14)		
	(HAL-15) (HAL-24) (HAL-25)			
(HAL-26) (HAL-27) (HAL-28) (HAL-29) (HAL-30) (HAL-31) (HAL-32) (HAL-33) (HAL-34) (HAL-35)		(HAL-16)		

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - DEBT CAN BE REPAID WITH UNDERLYING TOKEN	Medium	RISK ACCEPTED
HAL02 - INCOMPATIBILITY WITH DEFLATIONARY TOKENS	Medium	SOLVED - 02/20/2023
HAL03 - CHAINLINK ORACLE RETURN VALUES ARE NOT HANDLED PROPERTY	Medium	SOLVED - 02/20/2023
HAL04 - CHAINLINK ORACLE CAN CRASH WITH DECIMALS LONGER THAN 18	Medium	SOLVED - 02/20/2023
HAL05 - VERIFICATION OF NFT OWNER AND FLASH LOANS	Medium	RISK ACCEPTED
HAL06 - OWNER CAN EXTRACT ALL WRAPPED TOKEN FROM WRAPTOKENGATEWAY	Medium	SOLVED - 02/20/2023
HAL07 - FLASHMINT FEE IS NOT ADJUSTED ACCORDING TO BPS	Medium	SOLVED - 02/20/2023
HAL08 - DECREASE REPAY WITH BASE TOKEN LIMITER NOT USED	Medium	SOLVED - 02/20/2023
HAL09 - ALLOWLISTING ALLOWS ADMINS TO BLOCK WITHDRAWALS	Medium	SOLVED - 02/20/2023
HAL10 - ATTACK ON HARVEST BY A MALICIOUS KEEPER	Medium	SOLVED - 02/20/2023
HAL11 - CONFIGURECREDITUNLOCKRATE EFFECTS IMMEDIATELY	Medium	RISK ACCEPTED
HAL12 - SETSAVVY WILL FREEZE DEPOSITED FUNDS	Medium	SOLVED - 02/20/2023
HAL13 - REGISTERTOKEN MISUSE CAN PERMANENTLY DISABLE SAVVYSAGE AND BREAK THE SYSTEM	Medium	SOLVED - 02/20/2023
HAL14 - SAVVYSAGE.SOL CALLS DEPOSITUNDERLYING WITH NO SLIPPAGE BOUNDS	Medium	SOLVED - 02/20/2023
HAL15 - DECREASE ALLOWANCE WHEN ITS ALREADY SET TO NON-ZERO	Low	SOLVED - 02/20/2023

HAL16 - USE DISABLEINITILIAZERS TO PREVENT FRONT-RUNNING ON THE INITIALIZE FUNCTION	Low	SOLVED - 02/20/2023
HAL17 - OWNABLE CONTRACT HAS A SINGLE STEP OWNERSHIP TRANSFER	Low	SOLVED - 02/20/2023
HAL18 - ALLOWANCE GRANTED TO THE OLD ADAPTER SHOULD BE REVOKED WHEN SETTING THE NEW ADAPTER	Low	SOLVED - 02/20/2023
HAL19 - MISSING SANITY CHECK FOR THE NEW ADAPTER	Low	SOLVED - 02/20/2023
HAL20 - USE GRANTROLE INSTEAD OF SETUPROLE	Low	SOLVED - 02/20/2023
HAL21 - UPGRADEABLE CONTRACTS ARE NOT INITIALIZED	Low	SOLVED - 02/20/2023
HAL22 - NO STORAGE GAP FOR UPGRADEABLE CONTRACT	Low	SOLVED - 02/20/2023
HAL23 - LACK OF ALLOWLIST CHECK ON THE OWNER ADDRESS	Low	SOLVED - 02/20/2023
HAL24 - DEPOSITS CAN NOT BE PAUSED ON THE SWAP CONTRACT	Low	SOLVED - 02/20/2023
HAL25 - VAULT LOSS SCENARIO	Low	RISK ACCEPTED
HAL26 - REDUNDANT HARDHAT/CONSOLE.SOL IMPORT	Informational	SOLVED - 02/20/2023
HAL27 - OPTIMAL COMPARISON	Informational	SOLVED - 02/20/2023
HAL28 - OPTIMAL MULTIPLICATIONS AND DIVISIONS BY POWERS OF 2	Informational	SOLVED - 02/20/2023
HAL29 - OPTIMISATION OF IFS AND ZERO ADDRESS	Informational	PARTIALLY SOLVED - 02/20/2023
HAL30 - PACK STRUCTS	Informational	SOLVED - 02/20/2023
HAL31 - USE CALLDATA INSTEAD OF MEMORY FOR FUNCTION ARGUMENTS THAT DO NOT GET MUTATED	Informational	SOLVED - 02/20/2023

HAL32 - USE ASSEMBLY TO CHECK FOR ADDRESS(0)	Informational	ACKNOWLEDGED
HAL33 - USE 1E18 INSTEAD OF 10**18	Informational	SOLVED - 02/20/2023
HAL34 - ADAPTER'S MAX SLIPPAGE VARIABLE IS NEVER USED	Informational	SOLVED - 02/20/2023
HAL35 - EMIT POOL INFORMATION ON THE ADD POOL FUNCTION	Informational	SOLVED - 02/20/2023

FINDINGS & TECH DETAILS

3.1 (HAL-01) DEBT CAN BE REPAID WITH UNDERLYING TOKEN - MEDIUM

Description:

When an underlyingToken is used as collateral, the amount in terms of the underlyingToken (adjusted for decimals) will always be taken in a 1:1 ratio/price into _totalValue. We believe this design is flawed and can cause systemic failure when one of the underlyingTokens is severely depegged. Because when the underlyingToken is depegged, and the svyToken minted is worth more than the collateral underlyingToken in the market. The arbitrageur can simply mint and dump svyToken as much as they can and choose to abandon the collateral.

Code Location:

Listing 1: contracts/SavvyPositionManager.sol

```
1029  function _totalValue(address owner) internal view returns (
 ↳ uint256) {
1030          uint256 totalValue = 0;
1031
1032          Sets.AddressSet storage depositedTokens = _accounts[owner
 ↳ ].depositedTokens;
1033          for (uint256 i = 0; i < depositedTokens.values.length; i
 ↳ ++) {
1034              address yieldToken = depositedTokens.values[i];
1035              uint256 shares = _accounts[owner].balances[yieldToken
 ↳ ];
1036              (
1037                  address baseToken_,
1038                  uint256 amountBaseTokens
1039              ) = _yieldStrategyManager.convertSharesToBaseTokens(
 ↳ yieldToken, shares);
1040
1041              totalValue += _normalizeBaseTokensToDebt(baseToken_,
 ↳ amountBaseTokens);
1042          }
1043
1044          return totalValue;
```

```
1045     }
```

Scenario:

- One collateral token used by Savvy experiences a severe de-pegging against other collaterals. For this example, we will assume DAI drops to 80 cents vs. USDC & USDT. svyUSD maintains its peg against USDC & USDT.
- Users can buy DAI off the market, deposit it into Savvy, take a loan, and repeat this loop until the minting cap is reached. Users can buy DAI off the market and use it to repay their loans until the repay cap is reached.
- Users can liquidate their current yDAI position (paying off their outstanding debt discounted), buy more DAI with their loan, deposit it into Savvy, take a loan, and repeat until the liquidation cap is reached. These arbitrage opportunities will likely result in one or more of the mint / repay / liquidate caps being met.

Risk Level:

Likelihood - 3
Impact - 4

Recommendation:

Consider using an oracle to query the market price of the `underlyingTokens`, and calculate the `_totalValue` based on the market price.

Remediation Plan:

RISK ACCEPTED: The Savvy team accepted the risk of this issue.

3.2 (HAL-02) INCOMPATIBILITY WITH DEFLATIONARY TOKENS - MEDIUM

Description:

We have discovered that the system is incompatible with deflationary tokens. These are tokens that are designed to reduce in supply over time, often through a process called “burning”, which permanently removes tokens from circulation.

The protocol relies on accurate token balances to function properly, and the changing supply of deflationary tokens causes issues with this. Specifically, we have noticed that transactions involving deflationary tokens are not being recorded correctly, leading to incorrect balances and potentially causing problems for our users.

Code Location:

Listing 2: contracts/SavvyLGE.sol

```
289   function _buy(
290         uint256 deposited,
291         address nftCollectionAddress,
292         uint256 nftId,
293         uint256 vestModeIndex
294     ) internal {
295         uint256 allotmentsPerDepositToken  =
 ↳ _getAllotmentsPerDepositToken(nftCollectionAddress, vestModeIndex)
 ↳ ;
296         uint256 allotments = deposited * allotmentsPerDepositToken
 ↳ ;
297
298         if (nftCollectionAddress != address(0)) {
299             NFTAllocationInfo storage nftAllocationInfo =
 ↳ nftAllocationInfos[nftCollectionAddress][nftId];
300             if (!nftAllocationInfo.activated) {
301                 _activate(nftCollectionAddress, nftId, deposited);
302             } else {
303                 Checker.checkState(nftAllocationInfo.remaining >=
```

```
 ↳ deposited, "insufficient availability for nft");
304                 nftAllocationInfo.remaining -= deposited;
305             }
306         }
307
308         _updateUserBuyInfo(deposited, allotments, vestModeIndex);
309         allotmentSupply += allotments;
310         totalDeposited += deposited;
311
312         emit AllotmentsBought(msg.sender, deposited, allotments);
313     }
```

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

It is mandatory to have this dynamic in mind when on-boarding new tokens to be able to track the token balances correctly.

Remediation Plan:

SOLVED: The Savvy team solved this issue by adding pre- / post-balance check.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.3 (HAL-03) CHAINLINK ORACLE RETURN VALUES ARE NOT HANDLED PROPERTY - MEDIUM

Description:

Chainlink oracle return values are not handled properly, the priceFeed will return the following variables:
- roundId
- answer
- startedAt
- updatedAt
- answeredInRound

These return values are meant to be used to do some extra checks before updating the price. By just receiving the price, you can get stale prices and incomplete rounds.

Code Location:

Listing 3: contracts/SavvyPriceFeed.sol

```
65
66      function _getChainlinkTokenPrice(
67          address priceFeed_
68      ) internal view returns (uint256) {
69          AggregatorV3Interface priceFeed = AggregatorV3Interface(
↳ priceFeed_);
70          (,int price,,,) = priceFeed.latestRoundData();
71          uint256 tokenPrice = 0;
72          if (price > 0) {
73              tokenPrice = SafeCast.toUint256(price);
74          }
75
76          uint8 decimals = priceFeed.decimals();
77          uint8 additionDecimals = 18 - decimals;
78          return tokenPrice * 10**additionDecimals;
79      }
```

Scenario:

_getChainlinkTokenPrice calls out to a Chainlink oracle receiving the latestRoundData(). If there is a problem with Chainlink starting a new round and finding consensus on the new value for the oracle (e.g. Chainlink nodes abandon the oracle, chain congestion, vulnerability/attacks on the chainlink system) consumers of this contract may continue using outdated stale or incorrect data (if oracles are unable to submit no new round is started).

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

It is recommended to use this code to get all the values and sanitize the input.

Listing 4: contracts/SavvyPriceFeed.sol

```
65  (uint80 roundID ,answer,, uint256 timestamp, uint80
↳ answeredInRound) = AggregatorV3Interface(chainLinkAggregatorMap[
↳ underlying]).latestRoundData();
66
67 require(answer > 0, ""Chainlink price <= 0"");
68 require(answeredInRound >= roundID, ""Stale price"");
69 require(timestamp != 0, ""Round not complete"");"
```

Remediation Plan:

SOLVED: The Savvy team solved this issue. They added the recommended checks to check for stale prices and incomplete rounds.

Commit ID: 08ae4cde0e606121cbdbca9831c9cb430cfe3155

3.4 (HAL-04) CHAINLINK ORACLE CAN CRASH WITH DECIMALS LONGER THAN 18 - MEDIUM

Description:

If the priceFeed from chainlink returns 18 decimals or more, it will make the function crash or fail the calculation as the code is multiplying the token price by 18 minus the number of decimals of the priceFeed.

Code Location:

Listing 5: contracts/SavvyPriceFeed.sol

```
65   function _getChainlinkTokenPrice(
66         address priceFeed_
67     ) internal view returns (uint256) {
68         AggregatorV3Interface priceFeed = AggregatorV3Interface(
↳ priceFeed_);
69         (
70             uint80 roundID,
71             int price,,
72             uint256 timestamp,
73             uint80 answeredInRound
74         ) = priceFeed.latestRoundData();
75
76         require(price > 0, "Chainlink price <= 0");
77         require(answeredInRound >= roundID, "Stale price");
78         require(timestamp != 0, "Round not complete");
79
80         uint256 tokenPrice = uint256(price);
81
82         uint8 decimals = priceFeed.decimals();
83         uint8 additionDecimals = 18 - decimals;
84         return tokenPrice * 10**additionDecimals;
85     }
```

Proof Of Concept:

`_getChainlinkTokenPrice` gets the latest round information at that point, if the price feed returns a number of `decimals >= 18` they will be set to `0`. That will make the transaction revert or return a `0 - token price`.

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

It is mandatory to not use any token with more than 18 decimals to avoid breaking the feed.

Remediation Plan:

SOLVED: The Savvy team solved this issue by adding the decimal check.

Commit ID: 08ae4cde0e606121cbdbca9831c9cb430cfe3155

3.5 (HAL-05) VERIFICATION OF NFT OWNER AND FLASH LOANS - MEDIUM

Description:

Suppose that the NFT is borrowed (via a flash loan), then there is temporary access to the NFT and the user can interact with the buy function. If an attacker discovers a NFT that has not claimed all the tokens that they can, he will be able to do a flashloan then call the buy function, getting the tokens and then returning the NFT.

Code Location:

Listing 6: contracts/SavvyLGE.sol

```
202     function buy(
203         uint256 amount,
204         address nftCollectionAddress,
205         uint256 nftId,
206         uint256 vestModeIndex
207     ) public override whenNotPaused lgeNotEnded nonReentrant {
208         Checker.checkState(block.timestamp >= lgeStartTimestamp, "
 ↳ lge has not begun");
209         Checker.checkArgument(amount != 0, "amount is invalid");
210         Checker.checkArgument(vestModeIndex < vestModes.length, "
 ↳ invalid vest mode index");
211         if (nftCollectionAddress != address(0)) {
212             Checker.checkArgument(nftCollectionInfos[
 ↳ nftCollectionAddress].limit != 0, "this NFT is not eligible for
 ↳ boost");
213             Checker.checkArgument(IERC721(nftCollectionAddress).
 ↳ ownerOf(nftId) == msg.sender, "buyer is not the owner of this NFT"
 ↳ );
214         }
215         amount = TokenUtils.safeTransferFrom(address(depositToken)
 ↳ , msg.sender, depositTokenWallet, amount);
216         _buy(amount, nftCollectionAddress, nftId, vestModeIndex);
217     }
218
```

Proof Of Concept:

Listing 7: contracts/SavvyLGE.sol

```
289   function exploit(address _nftAddress, uint256 _nftId, uint256
 ↳ _vestMode) {
290         // Get fake flashloan
291         uint256 flashloanAmount = 10;
292         vm.deal(attacker, flashLoanAmount);
293
294         savvyLGE.buy(10, _nftAddress, _nftId, _vestMode);
295
296         // sell tokens
297
298         // Return fake flashloan
299         vm.deal(attacker, (payable)(address.this).balance -
 ↳ flashLoanAmount);
300   }
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Keep in mind that this can be abused if you create any new functionality that could be profitable for the exploiter.

Remediation Plan:

RISK ACCEPTED: The Savvy team accepted the risk of this issue.

3.6 (HAL-06) OWNER CAN EXTRACT ALL WRAPPED TOKEN FROM WRAPTOKENGATEWAY - MEDIUM

Description:

The owner of the WrapTokenGateway contract can extract all existing WrapToken from the contract via the refreshAllowance method.
These capabilities must be limited in favor of decentralization, in order to avoid loss of reputation or user confidence.

Code Location:

Listing 8: contracts/WrapTokenGateway.sol

```
53     function refreshAllowance(address savvy) external onlyOwner {
54         require (ISavvyPositionManager(savvy).supportInterface(
↳ type(ISavvyAdminActions).interfaceId), "not SavvyPositionManager
↳ address");
55         WAVAX.approve(savvy, type(uint256).max);
56     }
```

Proof Of Concept:

Listing 9: contracts/SavvyLGE.sol

```
289   function exploit(address attackerAddress) {
290         vm.prank(owner);
291         wrapTokenGateway.refreshAllowance(attacker);
292         WAVAX.withdraw(WAVAX.balanceOf(addressWrapTokenGateway()))
 ↳ ;
293   }
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

It is recommended a multisignature wallet / governance wallet or any other method to make sure that the owner's address does not get compromised.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They changed the refreshAllowance logic to and added a removeAllowance method that can help when changing the signer.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.7 (HAL-07) FLASHMINT FEE IS NOT ADJUSTED ACCORDING TO BPS - MEDIUM

Description:

The `setFlashFee` does not use the BPS constant when setting the `newFee`, this will lead to miss calculating every number that depends on `flashMintFee` after `setFlashFee` it is called.

Code Location:

Listing 10: contracts/SavvySyntheticToken.sol

```
100   function setFlashFee(uint256 newFee) public onlyAdmin {
101     Checker.checkArgument(newFee <= BPS, "invalid fee");
102     flashMintFee = newFee;
103     emit SetFlashMintFee(flashMintFee);
104   }
```

Proof Of Concept:

Listing 11: contracts/SavvyLGE.sol

```
289   function exploit() {
290         uint256 newFee = 10;
291         uint256 BPS = 10000;
292         vm.prank(admin);
293         savvySyntheticToken.setFlashFee(10);
294         // Here I compare the result with the BPS and the number
 ↳ that we get in the contract
295         vm.assertEq(savvySyntheticToken.flashMintFee, newFee/BPS);
296   }
```

Risk Level:

Likelihood - 2
Impact - 5

Recommendation:

It is a must to add the BPS to the calculation of the fee in order to have the calculations working.

Listing 12: contracts/SavvySyntheticToken.sol

```
100   function setFlashFee(uint256 newFee) public onlyAdmin {
101     Checker.checkArgument(newFee <= BPS, "invalid fee");
102     flashMintFee = newFee/BPS;
103     emit SetFlashMintFee(flashMintFee);
104   }
```

Remediation Plan:

SOLVED: The Savvy team solved this issue. They added the BPS rate to the calculation, making the result correct.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.8 (HAL-08) DECREASE REPAY WITH BASE TOKEN LIMITER NOT USED - MEDIUM

Description:

During the code review, It has been noticed that decreaseRepayWithBaseTokenLimiter function is not used. On the other hand, the limiter is decreased after limit check on the repayWithBaseToken function. Furthermore, in the repayWithCollateral token does not decrease limiter.

Code Location:

Listing 13: contracts/YieldStrategyManager.sol

```
401   function repayWithBaseToken(
402         address baseToken,
403         uint256 amount,
404         int256 debt
405     ) external onlySavvyPositionManager returns (uint256, uint256)
 ↳  {
406         // Determine the maximum amount of base tokens that can be
 ↳  repaid.
407         //
408         // It is implied that this value is greater than zero
 ↳ because `debt` is greater than zero so a noop is not possible
409         // beyond this point. Casting the debt to an unsigned
 ↳ integer is also safe because `debt` is greater than zero.
410         uint256 maximumAmount = _normalizeDebtTokensToUnderlying(
 ↳ baseToken, uint256(debt));
411
412         // Limit the number of base tokens to repay up to the
 ↳ maximum allowed.
413         uint256 actualAmount = amount > maximumAmount ?
 ↳ maximumAmount : amount;
414
415         // Check to make sure that the base token repay limit has
 ↳ not been breached.
```

```
416         uint256 _currentRepayWithBaseTokenLimit = _repayLimiters[
 ↳ baseToken].get();
417         if (actualAmount > _currentRepayWithBaseTokenLimit) {
418           revert RepayLimitExceeded(baseToken, actualAmount,
 ↳ _currentRepayWithBaseTokenLimit);
419         }
420
421         uint256 credit = _normalizeBaseTokensToDebt(baseToken,
 ↳ actualAmount);
422
423          // Decrease the amount of the base token which is
 ↳ globally available to be repaid.
424         _repayLimiters[baseToken].decrease(actualAmount);
425
426         return (credit, actualAmount);
427     }
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

It is recommended to delete the function or call it from repayWithBaseToken and repayWithCollateral instead of decreasing it there.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They changed the code to call decreaseRepayWithBaseTokenLimiter from the functions instead of writing the logic inside.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.9 (HAL-09) ALLOWLISTING ALLOWS ADMINS TO BLOCK WITHDRAWALS - MEDIUM

Description:

Allowlisting allows admins to block withdrawals. After some time of operation, the admins turn off allowlisting, allowing full integration by any smart contract. A small DeFi platform started integrating with SavvyDefi. Users using contract-based wallets and solutions like Gnosis Safe start using SavvyDefi. Then, due to a perceived risk, the SavvyDefi admins turn the allowlist back on. The small platform and the end users are now blocked from withdrawing funds, or interacting at all with the contracts in any way, and are at the mercy of the admins.

Code Location:

Listing 14: contracts/SavvyPositionManager.sol

```
164     function _withdrawYieldToken(
165         address owner,
166         address yieldToken,
167         uint256 shares,
168         address recipient
169     ) internal returns (uint256) {
170         _onlyAllowlisted();
171         Checker.checkArgument(recipient != address(0), "zero
 ↳ recipient address");
172         _yieldStrategyManager.checkSupportedYieldToken(yieldToken)
 ↳ ;
173 
174         uint256 amountYieldTokens = _withdraw(yieldToken, owner,
 ↳ shares, recipient);
175         TokenUtils.safeTransfer(yieldToken, recipient,
 ↳ amountYieldTokens);
176 
177         return amountYieldTokens;
178     }
```

Proof Of Concept:

Listing 15: contracts/SavvyPositionManager.sol

```
289 function exploit() {
290     vm.prank(victim);
291     uint256 tokenAmount = yieldToken.balanceOf(victim);
292     savvyPositionManager.depositBaseToken(address(yieldToken),
 ↳ tokenAmount, victim, tokenAmount);
293     vm.prank(owner);
294     allowlist.remove(victim);
295     vm.expectRevert(bytes("Unauthorized"));
296     savvyPositionManager.withdrawYieldToken(address(yieldToken),
 ↳ tokenAmount, victim);
297 }
```

Risk Level:

Likelihood - 2
Impact - 5

Recommendation:

It is recommended to a multisignature wallet or any other method to make sure that the owner's address doesn't get compromised.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They removed the allowlist check on the withdraw, meaning that the users will be able to remove the funds.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.10 (HAL-10) ATTACK ON HARVEST BY A MALICIOUS KEEPER - MEDIUM

Description:

Keepers are highly trusted entities that are intended to be either immutable smart contracts, upgradeable smart contracts with trusted governance, or EOAs or multisigs with trust assumptions similar to governance addresses. Whales can perform an attack on harvests, but it is usually infeasible. However, if a keeper account is compromised or acts maliciously, it can easily steal almost an entire harvest by flash loaning tokens, making a large deposit in the SavvyDefi, triggering a harvest, withdrawing the deposit, and repaying the flash loan in one transaction. To perform this attack on a yield token, a keeper would need to take out a large flash loan, buy a large amount of the yield token, deposit it in SavvyDefi, trigger a harvest, withdraw the yield tokens, and sell them before repaying the flash loan.

Code Location:

Listing 16: contracts/SavvyPositionManager.sol

```
681     function harvest(address yieldToken, uint256 minimumAmountOut
 ↳ ) external override lock {
682         _onlyKeeper();
683 
684         (
685             address baseToken_,
686             uint256 amountBaseTokens,
687             uint256 feeAmount,
688             uint256 distributeAmount,
689             uint256 credit
690         ) = _yieldStrategyManager.harvest(
691             yieldToken,
692             minimumAmountOut,
693             protocolFee
694         );
695 
696         // Transfer the tokens to the fee receiver and savvySwap.
```

```
697         TokenUtils.safeTransfer(baseToken_, protocolFeeReceiver,
 ↳ feeAmount);
698         TokenUtils.safeTransfer(baseToken_, savvySwap,
 ↳ distributeAmount);
699
700         // Inform the savvySwap that it has received tokens.
701         IERC20TokenReceiver(savvySwap).onERC20Received(baseToken_,
 ↳  distributeAmount);
702
703         emit Harvest(yieldToken, minimumAmountOut,
 ↳ amountBaseTokens, credit);
704     }
```

Proof Of Concept:

Listing 17: contracts/SavvyLGE.sol

```
289   function exploit() {
290     uint256 tokenAmount = yieldToken.balanceOf(
 ↳ savvyPositionManager);
291
292     savvyPositionManager.depositBaseToken(address(yieldToken),
 ↳ tokenAmount, attacker, tokenAmount);
293     savvyPositionManager.harvest(address(yieldToken), tokenAmount)
 ↳ ;
294     savvyPositionManager.withdrawYieldToken(address(yieldToken),
 ↳ tokenAmount, attacker);
295   }
```

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

It is recommended to have the keeper's access well-secured to avoid misuse of the trust.

Remediation Plan:

RISK ACCEPTED: The Savvy team accepted the risk of this issue.

3.11 (HAL-11) CONFIGURE CREDITUNLOCKRATE EFFECTS IMMEDIATELY - MEDIUM

Description:

If the configureCreditUnlockRate function is called to increase the rate, it will take effect immediately. However, the proportion of distributedCredit and pendingCredit will remain unchanged. As a consequence, every call to this function, harvest, repay* is going to revert.

Code Location:

Listing 18: contracts/YieldStrategyManager.sol

```
296     function configureCreditUnlockRate(address yieldToken, uint256
 ↳  blocks) external onlySavvyPositionManager {
297         Checker.checkArgument(blocks > 0, "zero blocks");
298         _checkSupportedYieldToken(yieldToken);
299         _yieldTokens[yieldToken].creditUnlockRate =
 ↳ FIXED_POINT_SCALAR / blocks;
300     }
```

Proof of Concept:

Listing 19

```
1   function exploit() {
2     vm.startPrank(address(savvyPositionManager));
3     yieldStrategyManager.configureCreditUnlockRate(yieldToken,
↳ numBlocks);
4     vm.expectRevert();
5     yieldStrategyManager.harvest(yieldToken, minimumAmount, 0);
6   }
```

Risk Level:

Likelihood - 3
Impact - 4

Recommendation:

It is recommended to call configureCreditUnlockRate whenever the proportion of distributedCredit and pendingCredit is updated.

Remediation Plan:

RISK ACCEPTED: The Savvy team accepted the risk of this issue.

3.12 (HAL-12) SETSAVVY WILL FREEZE DEPOSITED FUNDS - MEDIUM

Description:

Currently, setSavvy doesn't check whether there are any open positions left with the old savvy before switching to the new one. As this requires several checks, the probability of operational mistake isn't low, and it's prudent to introduce the main controls directly to the code to minimize it. In the case if the system goes on with new Savvy before realizing that there are some funds left in the old one, tedious and error-prone manual recovery will be needed. There is also going to be corresponding reputational damage. Setting the severity to medium as while the function is admin only, the impact is up to massive user fund freeze, i.e., this is system breaking with external assumptions.

Code Location:

Listing 20: contracts/SavvySage.sol

```
235   function setSavvy(address _savvy) external override onlyAdmin {
236         sources[savvy] = false;
237         sources[_savvy] = true;
238
239         if (savvy != address(0)) {
240             for (uint256 i = 0; i < registeredBaseTokens.length; i
 ↳ ++) {
241                 TokenUtils.safeApprove(registeredBaseTokens[i],
 ↳ savvy, 0);
242             }
243             TokenUtils.safeApprove(debtToken, savvy, 0);
244         }
245
246         savvy = _savvy;
247         yieldStrategyManager = ISavvyPositionManager(savvy).
 ↳ yieldStrategyManager();
248         for (uint256 i = 0; i < registeredBaseTokens.length; i++)
 ↳ {
```

```
249             TokenUtils.safeApprove(registeredBaseTokens[i], savvy,
 ↳  type(uint256).max);
250         }
251         TokenUtils.safeApprove(debtToken, savvy, type(uint256).max
 ↳ );
252
253         emit SetSavvy(savvy);
254     }
```

Proof Of Concept:

Savvy implementation change can happen while open deposits are remaining with the current contract. As there looks to be no process to transfer them in the code, such SavvySage funds will be frozen with old savvy:

Listing 21: contracts/SavvySage.sol

```
235   function setSavvy(address _savvy) external override onlyAdmin {
236         vm.startPrank(admin);
237         savvySage.setSavvy(address(newSavvy));
238         // Here we can not access the funds of the old savvy from
 ↳ the sage and they are stucked
239     }
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

It is recommended to require that all exposure to the old savvy is closed, for example, both `getAvailableFlow` and `getTotalCredit` is zero.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They added a check to see if there are any positions on the old savvy swap before making the change.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.13 (HAL-13) REGISTERTOKEN MISUSE CAN PERMANENTLY DISABLE SAVVYSAGE AND BREAK THE SYSTEM - MEDIUM

Description:

SavvySage's refreshStrategies() is the only way to actualize _yieldTokens array. The function requires registeredUnderlyings array to match current Savvy's _supportedUnderlyingTokens. At the same time, registeredUnderlyings can only be increased via registeredBaseTokens(): there is no way to reduce, remove or reconstruct the array. This way, if registerToken() was mistakenly called extra time or savvyst was switched with setSavvy to a new one with less supported assets, then the strategy refresh becomes impossible and the SavvySage be blocked as it cannot be properly used without synchronization with Savvy. The redeployment of the contract doesn't provide an easy fix, as it holds the accounting data that needs to be recreated (flowAvailable, currentExchanged mappings).

Code Location:

Listing 22: contracts/SavvyLGE.sol

```
367    function refreshStrategies() public override {
368        address[] memory supportedYieldTokens =
 ↳ yieldStrategyManager
369            .getSupportedYieldTokens();
370        address[] memory supportedBaseTokens =
 ↳ yieldStrategyManager
371            .getSupportedBaseTokens();
372
373        Checker.checkState(registeredBaseTokens.length ==
 ↳ supportedBaseTokens.length, "invalid base tokens information");
374
375        // clear current strats
376        for (uint256 j = 0; j < registeredBaseTokens.length; j++)
 ↳ {
377            delete _yieldTokens[registeredBaseTokens[j]];
378        }
```

```
379
380         uint256 numYTokens = supportedYieldTokens.length;
381         for (uint256 i = 0; i < numYTokens; i++) {
382             address yieldToken = supportedYieldTokens[i];
383
384             ISavvyPositionManager.YieldTokenParams memory params =
 ↳  yieldStrategyManager
385                 .getYieldTokenParameters(yieldToken);
386             if (params.enabled) {
387                 _yieldTokens[params.baseToken].push(yieldToken);
388             }
389         }
390         emit RefreshStrategies();
391     }
```

Proof Of Concept:

Listing 23: contracts/SavvyLGE.sol

```
289     function exploit() {
290         vm.startPrank(admin);
291         savvySage.registerToken(baseToken, address(savvySwap));
292         savvySage.registerToken(baseToken, address(savvySwap));
293         savvySage.registerToken(baseToken, address(savvySwap));
294         vm.stopPrank();
295         vm.expectRevert("invalid base tokens information");
296         savvySage.refreshStrategies();
297
298     }
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

It is a must careful when calling registerToken to not do any extra call or add a function to remove registeredBaseTokens from the array.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They added a remove base token function to be able to make the system functional again.

Commit ID: 65140e9b2859285d98ec1c3945309ab189689d57

3.14 (HAL-14) SAVVYSAGE CALLS DEPOSITUNDERLYING WITH NO SLIPPAGE BOUNDS - MEDIUM

Description:

If the buffer (SavvySage) is called during and unfavorable time, then a large portion of deposited funds may be lost due to slippage because deposit is called with 0 as the minimum out allowing any level of slippage

Code Location:

Listing 24: contracts/SavvyLGE.sol

```
486     function _savvyDeposit(address token, uint256 amount) internal
 ↳  {
487         ISavvyPositionManager(savvy).depositBaseToken(
488             token,
489             amount,
490             address(this),
491             0
492         );
493     }
```

Proof Of Concept:

Imagine that the market is having a huge volatility currently that you call the deposit function, It will call the depositBaseToken function with a 0 in the slippage parameter. Due to these conditions, your amount deposited on the position manager will be very different from the amount that you introduced as an argument when you were calling _savvyDeposit and there is no way to control that from the user side.

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

It is recommended to slippage / minimum out as a user's parameter.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They added a minOut variable that enables the user to set slippage bonds to avoid the loss of funds.

Commit ID: 9843e94c709aef44a4bfafc7e238c24dba3a2fe7

3.15 (HAL-15) DECREASE ALLOWANCE WHEN ITS ALREADY SET TO NON-ZERO - LOW

Description:

We have noticed that when attempting to decrease the allowance for a given token using the decreaseAllowance function, it only works if the allowance is already set to zero. If the allowance is set to a non-zero value, the function does not properly decrease the allowance. Non-standard tokens like USDT will revert the transaction when a contract or a user tries to approve an allowance when the spender allowance is already set to a non-zero value. For that reason, the previous allowance should be decreased before increasing allowance in the related function.

Code Location:

Listing 25: contracts/YieldStrategyManager.sol

```
303     function setTokenAdapter(address yieldToken, address adapter)
 ↳ external onlySavvyPositionManager {
304         address oldAdapter = _yieldTokens[yieldToken].adapter;
305         Checker.checkState(yieldToken == ITokenAdapter(adapter).
 ↳ token(), "invalid yield token address");
306         Checker.checkState(ITokenAdapter(oldAdapter).baseToken()
 ↳ == ITokenAdapter(adapter).baseToken(), "invalid base token address
 ↳ ");
307         _checkSupportedYieldToken(yieldToken);
308
309
310         TokenUtils.safeApprove(yieldToken, oldAdapter, 0);
311         TokenUtils.safeApprove(ITokenAdapter(oldAdapter).baseToken
 ↳ (), oldAdapter, 0);
312
313         TokenUtils.safeApprove(yieldToken, adapter, type(uint256).
 ↳ max);
314         TokenUtils.safeApprove(ITokenAdapter(adapter).baseToken(),
 ↳  adapter, type(uint256).max);
315
```

```
316             _yieldTokens[yieldToken].adapter = adapter;
317         }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to do a study of the tokens that you will use to avoid non-standard tokens like USDT.

Remediation Plan:

SOLVED: The Savvy team solved this issue by setting allowance to zero first.

Commit ID: 839f3858f7726fba6a3570bb0ecca5d5c1e3383e

3.16 (HAL-16) USE DISABLEINITIALIZERS TO PREVENT FRONT-RUNNING ON THE INITIALIZE FUNCTION - LOW

Description:

Use disableInitializers to prevent front-running on the initialize function, as it would make you deploy the smart contract again if someone initializes it before you.

Code Location:

All the smart contracts.

Risk Level:

Likelihood - 3
Impact - 1

Recommendation:

It is recommended to disable initializers to prevent front-running on the initialize function.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They disabled the initializers, so the possibility of doing front-run is not enabled.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.17 (HAL-17) OWNABLE CONTRACT HAS A SINGLE STEP OWNERSHIP TRANSFER - LOW

Description:

The transferOwnership function transfers contract ownership in a single step. If the owner of a contract were set to an address not controlled by the SavvyDefi Team, the contract would be impossible to recover.

Code Location:

All the smart contracts.

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to a two-step process in which an owner proposes an ownership transfer and the proposed new owner accepts it.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They changed the process to a two-step verification to be able to have control over the flow.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.18 (HAL-18) ALLOWANCE GRANTED TO THE OLD ADAPTER SHOULD BE REVOKED WHEN SETTING THE NEW ADAPTER - LOW

Description:

Allowance granted to the old adapter should be revoked when setting the new adapter to avoid unexpected behaviors and abuses.

Code Location:

Listing 26: contracts/YieldStrategyManager.sol

```
303     function setTokenAdapter(address yieldToken, address adapter)
 ↳ external onlySavvyPositionManager {
304         address oldAdapter = _yieldTokens[yieldToken].adapter;
305         Checker.checkState(yieldToken == ITokenAdapter(adapter).
 ↳ token(), "invalid yield token address");
306         Checker.checkState(ITokenAdapter(oldAdapter).baseToken()
 ↳ == ITokenAdapter(adapter).baseToken(), "invalid base token address
 ↳ ");
307         _checkSupportedYieldToken(yieldToken);
308
309
310         TokenUtils.safeApprove(yieldToken, oldAdapter, 0);
311         TokenUtils.safeApprove(ITokenAdapter(oldAdapter).baseToken
 ↳ (), oldAdapter, 0);
312
313         TokenUtils.safeApprove(yieldToken, adapter, type(uint256).
 ↳ max);
314         TokenUtils.safeApprove(ITokenAdapter(adapter).baseToken(),
 ↳  adapter, type(uint256).max);
315
316         _yieldTokens[yieldToken].adapter = adapter;
317     }
318
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended to revoke the allowance to the old adapter when setting the new one.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They remove the allowance when setting the new adopter.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.19 (HAL-19) MISSING SANITY CHECK FOR THE NEW ADAPTER - LOW

Description:

The new adapter should be checked for a matching underlyingToken to avoid setting an incompatible adapter. If, for some reason, you set an adapter with another underlyingToken it will not be compatible and break the protocol.

Code Location:

Listing 27: contracts/YieldStrategyManager.sol

```
299   function setTokenAdapter(address yieldToken, address adapter)
 ↳ external onlySavvyPositionManager {
300         address oldAdapter = _yieldTokens[yieldToken].adapter;
301         Checker.checkState(yieldToken == ITokenAdapter(adapter).
 ↳ token(), "invalid yield token address");
302         Checker.checkState(ITokenAdapter(oldAdapter).baseToken()
 ↳ == ITokenAdapter(adapter).baseToken(), "invalid base token address
 ↳ ");
303         _checkSupportedYieldToken(yieldToken);
304
305
306         TokenUtils.safeApprove(yieldToken, oldAdapter, 0);
307         TokenUtils.safeApprove(ITokenAdapter(oldAdapter).baseToken
 ↳ (), oldAdapter, 0);
308
309         TokenUtils.safeApprove(yieldToken, adapter, type(uint256).
 ↳ max);
310         TokenUtils.safeApprove(ITokenAdapter(adapter).baseToken(),
 ↳  adapter, type(uint256).max);
311
312         _yieldTokens[yieldToken].adapter = adapter;
313     }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended to add a check to see if the token adapter underlying asset is the same one as the old one.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They added the sanity check to see if the underlying collateral is the same as the new adapter.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.20 (HAL-20) USE GRANTROLE INSTEAD OF SETUPROLE - LOW

Description:

According to OpenZeppelin documentation, _setupRole() is only to be called from the constructor while _grantRole() is to be called anywhere else, doing otherwise is circumventing the admin system.

Code Location:

Listing 28: contracts/SavvyLGE.sol

```
126   function setAllowedMinter(address minter, bool state) external
 ↳ onlyAdmin {
127     Checker.checkArgument(minter != address(0), "zero minter
 ↳ address");
128     if (state) {
129       _grantRole(MINTER_ROLE, minter);
130     } else {
131       _revokeRole(MINTER_ROLE, minter);
132     }
133   }
134
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended to use grantRole instead of setupRole.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They changed the code from grantrole to setuprole.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.21 (HAL-21) UPGRADEABLE CONTRACTS ARE NOT INITIALIZED - LOW

Description:

In the project, the ReentrancyGuardUpgradeable and AccessControlUpgradeable upgradeable contracts are not initialized.

Code Location:

Listing 29

```
./SavvySwap.sol:27:contract SavvySwap is ISavvySwap,
↳ ReentrancyGuardUpgradeable, AccessControlUpgradeable {}
./SavvyActionBatcher.sol:13:contract SavvyActionBatcher is
↳ ISavvyActionBatcher, OwnableUpgradeable {}
./YieldStrategyManager.sol:16:contract YieldStrategyManager is
↳ IYieldStrategyManager, Mutex, OwnableUpgradeable {}
./SavvySage.sol:27:contract SavvySage is ISavvySage,
↳ AccessControlUpgradeable {
./WrapTokenGateway.sol:17:contract WrapTokenGateway is
↳ IWrapTokenGateway, OwnableUpgradeable {}
./InfoAggregator.sol:25:contract InfoAggregator is IInfoAggregator
↳ , OwnableUpgradeable {}
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

Consider initializing the all upgradeable contracts in the initializer of the contract.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They initialized the contracts.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.22 (HAL-22) NO STORAGE GAP FOR UPGRADEABLE CONTRACT - LOW

Description:

For upgradeable contracts, there must be a storage gap to “allow developers to freely add new state variables in the future without compromising the storage compatibility with existing deployments” (quote OpenZeppelin). Otherwise, it may be very difficult to write new implementation code. Without storage gap, the variable in the child contract might be overwritten by the upgraded base contract if new variables are added to the base contract. This could have unintended and serious consequences on the child contracts, potentially causing loss of user funds or cause the contract to malfunction completely. However, none of these contracts contain a storage gap.

Refer to the bottom part of this article: writing-upgradeable

Code Location:

Listing 30

```
./SavvySwap.sol:27:contract SavvySwap is ISavvySwap,
↳ ReentrancyGuardUpgradeable, AccessControlUpgradeable {}
./SavvyActionBatcher.sol:13:contract SavvyActionBatcher is
↳ ISavvyActionBatcher, OwnableUpgradeable {}
./YieldStrategyManager.sol:16:contract YieldStrategyManager is
↳ IYieldStrategyManager, Mutex, OwnableUpgradeable {}
./SavvySage.sol:27:contract SavvySage is ISavvySage,
↳ AccessControlUpgradeable {
./WrapTokenGateway.sol:17:contract WrapTokenGateway is
↳ IWrapTokenGateway, OwnableUpgradeable {}
./InfoAggregator.sol:25:contract InfoAggregator is IInfoAggregator
↳ , OwnableUpgradeable {}
```

Recommendation:

Consider adding a storage gap at the end of the upgradeable abstract contract.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They added gaps of size 100 to fix this issue.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.23 (HAL-23) LACK OF ALLOWLIST CHECK ON THE OWNER ADDRESS - LOW

Description:

The current system lacks a crucial security measure in the form of an allowlist check on the owner's address. This means that any address can potentially act as the owner and change the smart contract, regardless of whether they are authorized to do so. The deposit function does not check If the owner is allow-listed on the contract. With the following scenario, a depositor could not withdraw funds from the contract.

- Depositor call deposit function with non-allow-listed address (as an owner).
- Depositor is trying to call withdraw function.
- Depositor cannot withdraw due to owner address is not white-listed.

Code Location:

SavvySwap.sol#L211

Listing 31

```
  function deposit(uint256 amount, address owner) external
↳ override nonReentrant {
    _onlyAllowlisted();
    amount = TokenUtils.safeTransferFrom(syntheticToken, msg.
↳ sender, address(this), amount);
    _updateAccount(
      UpdateAccountParams({
        owner: owner,
        unswappedDelta: SafeCast.toInt256(amount),
        swappedDelta: 0
      })
    );
    emit Deposit(msg.sender, owner, amount);
  }
```

Recommendation:

Consider checking If the owner’s address is white-listed.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They changed the code to see if the owner is allowlisted instead of the msg.sender.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.24 (HAL-24) DEPOSITS CAN NOT BE PAUSED ON THE SWAP CONTRACT - LOW

Description:

It has been reported that deposits into the swap contract cannot be paused, even when necessary. This is a critical issue as there may be situations where it is necessary to temporarily halt deposits for maintenance or security reasons. The inability to pause deposits on the swap contract can lead to potential security risks, as well as increased workload for the team responsible for managing the contract.

Code Location:

/contracts/SavvySwap.sol#L206

Listing 32

```
1   function deposit(uint256 amount, address owner) external
↳ override nonReentrant {
2     _onlyAllowlisted();
3     amount = TokenUtils.safeTransferFrom(syntheticToken, msg.
↳ sender, address(this), amount);
4     _updateAccount(
5       UpdateAccountParams({
6         owner: owner,
7         unswappedDelta: SafeCast.toInt256(amount),
8         swappedDelta: 0
9       })
10    );
11    emit Deposit(msg.sender, owner, amount);
12  }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to implement a mechanism that allows deposits to be paused in the swap contract.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They added a way to stop deposits.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.25 (HAL-25) VAULT LOSS SCENARIO - LOW

Description:

If a yield-bearing asset experiences a loss in the underlying collateral, the max loss function will get activated and automatic changes will be triggered that can create a total amount of funds lost from the vault.

We will assume a 10% loss of DAI from the ysvyDAI vault that is unrecoverable. After the transaction that causes the loss is committed to the chain, the maxloss() is triggered and the following happens:

The following ysvyDAI Savvy functions are automatically disabled:
- deposit()
- depositUnderlying()
- withdrawUnderlying()
- withdrawUnderlyingFrom()
- liquidate()
- harvest()

The following ysvyDAI Savvy functions are still usable:
- withdraw()
- withdrawFrom()
- repay()
- mint()
- burn()

Code Location:

Listing 33: contracts/SavvyPositionManager.sol

```
1029  function _totalValue(address owner) internal view returns (
 ↳ uint256) {
1030          uint256 totalValue = 0;
1031
1032          Sets.AddressSet storage depositedTokens = _accounts[owner
```

```
 ↳ ].depositedTokens;
033         for (uint256 i = 0; i < depositedTokens.values.length; i
 ↳ ++) {
034             address yieldToken = depositedTokens.values[i];
035             uint256 shares = _accounts[owner].balances[yieldToken
 ↳ ];
036             (
037                 address baseToken_,
038                 uint256 amountBaseTokens
039             ) = _yieldStrategyManager.convertSharesToBaseTokens(
 ↳ yieldToken, shares);
040
041             totalValue += _normalizeBaseTokensToDebt(baseToken_,
 ↳ amountBaseTokens);
042         }
043
044         return totalValue;
045     }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to have on-chain monitoring to be able to react fast if this situation happens.

Remediation Plan:

RISK ACCEPTED: The Savvy team accepted the risk of this issue.

3.26 (HAL-26) REDUNDANT HARDHAT/CONSOLE.SOL IMPORT - INFORMATIONAL

Description:

In the contracts, hardhat/console is imported to several places. In the production environment, It is recommended to delete hardhat/console.

Code Location:

All the contracts.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider deleting hardhat console import.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They removed the duplicated import.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.27 (HAL-27) OPTIMAL COMPARISON - INFORMATIONAL

Description:

When comparing integers, it is cheaper to use strict > & < operators over >= & <= operators, even if you must increment or decrement one of the operands. Note: before using this technique, it's important to consider whether incrementing/decrementing one of the operators could result in an over/underflow.
This optimization is applicable when the optimizer is turned off.

Code Location:

It is located in multiple locations inside the following contracts:

Listing 34

```
1 - VeERC20Upgradeable.sol
2 - VeSvy.sol
3 - YieldStrategyManager.sol
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to refactor the integers comparisons with strict operators.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They changed the integer comparisons with strict operators to have better gas costs.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.28 (HAL-28) OPTIMAL MULTIPLICATIONS AND DIVISIONS BY POWERS OF 2 - INFORMATIONAL

Description:

Right shift or Left shift instead of dividing or multiplying by powers of two.

Code Location:

It is located on multiple locations inside the following contracts:
- VeERC20Upgradeable.sol
- VeSvy.sol
- YieldStrategyManager.sol
- Allowlist.sol

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use shifting instead of multiplying by powers of 2 to reduce the gas consumption.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They moved from multiplications by powers of 2 to shifting.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.29 (HAL-29) OPTIMISATION OF IFS AND ZERO ADDRESS - INFORMATIONAL

Description:

It is redundant to check `if(x == true)` or any form of boolean comparison. You can slightly reduce gas consumption by using if (x) instead. When applicable, you can also use assembly to save more gas by using `iszeroiszero(x)` instead of if (x) and `iszero(x)` for if `(!x)`.

Code Location:

You can find this issue in lines 32 and 40 of Allowlist.

Listing 35: contracts/utils/Allowlist.sol

```
29
30   /// @inheritdoc IAllowlist
31   function add(address caller) external override {
32     _onlyAdmin();
33     Checker.checkState(disabled == false, "add allowlist is
↳ disabled");
34     addresses.add(caller);
35     emit AccountAdded(caller);
36   }
37
38   /// @inheritdoc IAllowlist
39   function remove(address caller) external override {
40     _onlyAdmin();
41     Checker.checkState(disabled == false, "add allowlist is
↳ disabled");
42     addresses.remove(caller);
43     emit AccountRemoved(caller);
44   }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to refactor the code of the boolean comparisons to reduce the gas consumption.

Remediation Plan:

PARTIALLY SOLVED: The Savvy team solved the issue partially.

3.30 (HAL-30) PACK STRUCTS - INFORMATIONAL

Description:

When creating structs, make sure that the variables are listed in ascending order by data type. The compiler will pack the variables that can fit into one 32 byte slot. If the variables are not listed in ascending order, the compiler may not pack the data into one slot, causing additional sload and sstore instructions when reading/storing the struct into the contract's storage.

Code Location:

You can find this issue in lines 8 and 15 of ISavvyTokenParams.

Listing 36: contracts/interfaces/savvy/ISavvyTokenParams.sol

```
 8     /// @notice Defines base token parameters.
 9     struct BaseTokenParams {
10         uint8 decimals;
11         uint256 conversionFactor;
12         bool enabled;
13     }
14
15     /// @notice Defines yield token parameters.
16     struct YieldTokenParams {
17         uint8 decimals;
18         address baseToken;
19         address adapter;
20         uint256 maximumLoss;
21         uint256 maximumExpectedValue;
22         uint256 creditUnlockRate;
23         uint256 activeBalance;
24         uint256 harvestableBalance;
25         uint256 totalShares;
26         uint256 expectedValue;
27         uint256 pendingCredit;
28         uint256 distributedCredit;
29         uint256 lastDistributionBlock;
```

```
30          uint256 accruedWeight;
31          bool enabled;
32      }
33      }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to pack structs in ascending order.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They packed the structs of the specified file, saving gas on deployment.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.31 (HAL-31) USE CALLDATA INSTEAD OF MEMORY FOR FUNCTION ARGUMENTS THAT DO NOT GET MUTATED - INFORMATIONAL

Description:

Mark data types as calldata instead of memory where possible. This makes it so that the data is not automatically loaded into memory. If the data passed into the function does not need to be changed (like updating values in an array), it can be passed in as calldata. The one exception to this is if the argument must later be passed into another function that takes an argument that specifies memory storage.

Code Location:

You can find this issue in both VeERC20Upgradeable and YieldStrategyManager:

Listing 37: contracts/VeERC20Upgradeable.sol

```
36     function __ERC20_init(string memory name_, string memory
↳ symbol_) internal initializer {
37         __Context_init_unchained();
38         __ERC20_init_unchained(name_, symbol_);
39     }
40
41     function __ERC20_init_unchained(string memory name_, string
↳ memory symbol_) internal initializer {
42         _name = name_;
43         _symbol = symbol_;
44     }
```

Listing 38: contracts/YieldStrategyManager.sol

```
581
582     function setBorrowingLimiter(
```

```
583         Limiters.LinearGrowthLimiter memory borrowingLimiter_
584     ) external {
585         // This is first time so savvyPositionManager should be
 ↳ zero.
586         if (savvyPositionManager != address(0)) {
587             revert Unauthorized();
588         }
589         _borrowingLimiter = borrowingLimiter_;
590     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to mark the data type as calldata instead of memory.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They changed from memory to calldata on places that didn't mutate the state, saving gas.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.32 (HAL-32) USE ASSEMBLY TO CHECK FOR ADDRESS(0) - INFORMATIONAL

Description:

Use assembly to check for address(0) to make the gas fees lower.

Code Location:

You can apply this optimisation in multiple smart contracts:
- YieldYakAdapter
- InfoAggregator
- SavvyActionBatcher
- SavvyLGE
- SavvyBooster
- SavvyPositionManager
- SavvyPriceFeed
- SavvyRedlist
- SavvySage
- SavvySyntheticToken
- VeERC20Upgradeable
- VeSvy
- YieldStrategyManager

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to create a helper function that checks if the address is `address(0)` and use it in all the functions that are doing the `if( address(0))` check to reduce gas costs.

Listing 39

```
function assemblyOwnerNotZero(address _addr) public pure {
        assembly {
            if iszero(_addr) {
                mstore(0x00, "zero address")
                revert(0x00, 0x20)
            }
        }
    }
```

Remediation Plan:

ACKNOWLEDGED: The Savvy team acknowledged this issue.

3.33 (HAL-33) USE 1E18 INSTEAD OF 10**18 - INFORMATIONAL

Description:

It is recommended to use scientific notation (1e18) instead of exponential (10**18).

Code Location:

Listing 40: contracts/libraries/Math.sol

```
7       uint256 public constant WAD = 10**18;
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to change the code when initializing the variable:

Listing 41: contracts/libraries/Math.sol

```
7       uint256 public constant WAD = 10e18;
```

Remediation Plan:

SOLVED: The Savvy team solved this issue. They simply changed from 10**18 to 1E18.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.34 (HAL-34) ADAPTER'S MAX SLIPPAGE VARIABLE IS NEVER USED - INFORMATIONAL

Description:

The variable MAX_SLIPPAGE on all the adapters it is never used, it should be removed to make the gas of the deployment cheaper.

Code Location:

All the adapters under ./contracts/adapters.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to remove the declaration of the MAX_SLIPPAGE variable.

Remediation Plan:

SOLVED: The Savvy team solved this issue. They removed the unused variable.

Commit ID: 10cd1d78ab55b20e143323bdf7f0f7080fc48a8f

3.35 (HAL-35) EMIT POOL INFORMATION ON THE ADD POOL FUNCTION - INFORMATIONAL

Description:

The current implementation of the add pool function does not emit any information about the added pool to the rest of the system. (Pool ID) This makes it difficult for users to track and monitor the status of the newly added pool.

Code Location:

SavvyBooster.sol#L154

Listing 42: SavvyBooster.sol

```
1    function addPool(
2        uint256 amount_,
3        uint256 duration_
4    ) external override onlyOwner lock {
5        Checker.checkArgument(amount_ > 0, "amount must be greater
↳ than zero");
6        Checker.checkArgument(duration_ > 0, "duration must be
↳ greater than zero");
7
8        amount_ = TokenUtils.safeTransferFrom(address(svyToken), msg
↳ .sender, address(this), amount_);
9
10       uint256 totalDebtBalance = 0;
11       uint256 totalVeSvyBalance = 0;
12       uint256 poolStart = block.timestamp;
13       uint256 poolLength = pools.length;
14       if (poolLength > 0) {
15           PoolInfo memory lastPool = pools[poolLength - 1];
16           poolStart = lastPool.startTime + lastPool.duration;
17           totalDebtBalance = lastPool.totalDebtBalance;
18           totalVeSvyBalance = lastPool.totalVeSvyBalance;
19       }
```

```
20        PoolInfo memory pool = PoolInfo({
21            remainingEmissions: amount_,
22            emissionRatio: amount_ / duration_,
23            duration: duration_,
24            startTime: poolStart,
25            totalDebtBalance: totalDebtBalance,
26            totalVeSvyBalance: totalVeSvyBalance
27        });
28        pools.push(pool);
29
30        emit Deposit(amount_);
31    }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to implement a mechanism that emits information about the newly added pool to the rest of the system. This could be done using events or logging, and should include relevant information such as the pool's name, its id.

Remediation Plan:

SOLVED: The Savvy team solved this issue by adding pool length.

Commit ID: a6c125af3654a3c6b3c762fe28e71d54b859383c

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contracts in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contracts in the repository and was able to compile them correctly into their abis and binary format, Slither was run against the contracts. 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:

```
YieldYakAdapter._deposit(uint256,address) (contracts/adapters/yieldyak/YieldYakAdapter.sol#78-93) sends eth to arbitrary user
        Dangerous calls:
        - vault.depositFor{value: amount}(recipient) (contracts/adapters/yieldyak/YieldYakAdapter.sol#84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

Multicall.multicall(bytes[]) (contracts/base/Multicall.sol#12-23) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/base/Multicall.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

IERC20Metadata is re-used:
        - IERC20Metadata (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#13-28)
        - IERC20Metadata (contracts/interfaces/IERC20Metadata.sol#6-22)
Math is re-used:
        - Math (node_modules/@openzeppelin/contracts/utils/math/Math.sol#9-43)
        - Math (contracts/libraries/Math.sol#6-93)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#name-reused

Reentrancy in SavvyLGE.setProtocolToken(address) (contracts/SavvyLGE.sol#147-152):
        External calls:
        - withdrawProtocolToken() (contracts/SavvyLGE.sol#150)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#93)
                - protocolToken.safeTransfer(depositTokenWallet,currentBalance) (contracts/SavvyLGE.sol#355)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        External calls sending eth:
        - withdrawProtocolToken() (contracts/SavvyLGE.sol#150)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        State variables written after the call(s):
        - _setProtocolToken(_protocolToken) (contracts/SavvyLGE.sol#151)
                - protocolToken = newProtocolToken (contracts/SavvyLGE.sol#365)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

TestVeSvy.stake(uint256) (contracts/test/TestVeSvy.sol#18-22) ignores return value by IERC20(svyToken).transferFrom(msg.sender,address(this),amount) (contracts/test/TestVeSvy.sol#19)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

SavvySage.weightings (contracts/SavvySage.sol#70) is never initialized. It is used in:
        - SavvySage.getWeight(address,address) (contracts/SavvySage.sol#132-139)
        - SavvySage.setWeights(address,address[],uint256[]) (contracts/SavvySage.sol#180-215)
        - SavvySage._savvyAction(uint256,address,function(address,uint256)) (contracts/SavvySage.sol#456-469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

SavvyLoanMigrator (contracts/SavvyLoanMigrator.sol#21-121) is an upgradeable contract that does not protect its initiliaze functions: SavvyLoanMigrator.initialize(ISavvyLoanMigrator.InitializationParams) (contracts/SavvyLoanMigrator.sol#32-46). Anyone can delete the contract with: Multicall.multicall(bytes[]) (contracts/base/Multicall.sol#12-23)SavvyPositionManager (contracts/SavvyPositionManager.sol#29-1260) is an upgradeable contract that does not protect its initiliaze functions: ISavvyAdminActions.initialize(ISavvyAdminActions.InitializationParams) (contracts/interfaces/savvy/ISavvyAdminActions.sol#91)SavvyPositionManager.initialize(ISavvyAdminActions.InitializationParams) (contracts/SavvyPositionManager.sol#158-190). Anyone can delete the contract with: Multicall.multicall(bytes[]) (contracts/base/Multicall.sol#12-23)Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unprotected-upgradeable-contract


SavvyLGE.getClaimable(address) (contracts/SavvyLGE.sol#244-259) performs a multiplication on the result of a division:
        -accruedPerSecond = totalOwed / userBuyInfos[userAddress].duration (contracts/SavvyLGE.sol#251)
        -owed = (block.timestamp - lastClaim) * accruedPerSecond (contracts/SavvyLGE.sol#254)
SavvyLGE._getAllotmentsPerDepositToken(address,uint256) (contracts/SavvyLGE.sol#302-314) performs a multiplication on the result of a division:
        -allotmentPerDepositToken = pricePerAllotment * vestModeBooster / BASIS_POINTS (contracts/SavvyLGE.sol#307)
        -allotmentPerDepositToken = allotmentPerDepositToken * nftBoost / BASIS_POINTS (contracts/SavvyLGE.sol#311)
SavvyLoanMigrator._convertToDebt(uint256,address,address) (contracts/SavvyLoanMigrator.sol#109-119) performs a multiplication on the result of a division:
        -underlyingValue = shares * yieldStrategyManager.getBaseTokensPerShare(yieldToken) / 10 ** TokenUtils.expectDecimals(yieldToken) (contracts/SavvyLoanMigrator.sol#117)
        -underlyingValue * 10 ** (18 - decimals[baseToken]) (contracts/SavvyLoanMigrator.sol#118)
BeefyVaultMock.withdraw(uint256,address,uint256) (contracts/test/beefy/BeefyVaultMock.sol#85-104) performs a multiplication on the result of a division:
        -_r = (balance() * _shares) / totalSupply() (contracts/test/beefy/BeefyVaultMock.sol#94)
        -withdrawnAmt = _r - (_r * forcedSlippage) / PERCENT_RESOLUTION (contracts/test/beefy/BeefyVaultMock.sol#98)
BeefyVaultMock.withdraw(uint256,address,uint256) (contracts/test/beefy/BeefyVaultMock.sol#85-104) performs a multiplication on the result of a division:
        -_r = (balance() * _shares) / totalSupply() (contracts/test/beefy/BeefyVaultMock.sol#94)
        -Checker.checkState(withdrawnAmt >= _r - (_r * maxSlippage) / PERCENT_RESOLUTION,too much slippage) (contracts/test/beefy/BeefyVaultMock.sol#99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

InfoAggregator._findIndexFromFullSavvyPositionArray(ISavvyInfoAggregatorStructs.FullSavvyPosition[],address) (contracts/InfoAggregator.sol#879-889) uses a dangerous strict equality:
        - arr[i].token == token || arr[i].token == address(0) (contracts/InfoAggregator.sol#884)
InfoAggregator.getSVYAPY(address) (contracts/InfoAggregator.sol#308-320) uses a dangerous strict equality:
        - totalUserDepositedAmount == 0 (contracts/InfoAggregator.sol#313)
SavvyBooster._getCollateralWeightRatio(address,uint256) (contracts/SavvyBooster.sol#339-346) uses a dangerous strict equality:
        - totalCollateralWeight == 0 (contracts/SavvyBooster.sol#345)
YieldStrategyManager._loss(address) (contracts/YieldStrategyManager.sol#892-906) uses a dangerous strict equality:
        - amountBaseTokens == 0 (contracts/YieldStrategyManager.sol#898)
YieldStrategyManager.calculateUnrealizedActiveBalance(address) (contracts/YieldStrategyManager.sol#666-686) uses a dangerous strict equality:
        - harvestable == 0 (contracts/YieldStrategyManager.sol#681)
YieldStrategyManager.convertSharesToYieldTokens(address,uint256) (contracts/YieldStrategyManager.sol#650-656) uses a dangerous strict equality:
        - totalShares == 0 (contracts/YieldStrategyManager.sol#652)
YieldStrategyManager.convertYieldTokensToShares(address,uint256) (contracts/YieldStrategyManager.sol#337-343) uses a dangerous strict equality:
        - yieldTokenParams.totalShares == 0 (contracts/YieldStrategyManager.sol#339)
YieldStrategyManager.depositPrepare(address) (contracts/YieldStrategyManager.sol#737-771) uses a dangerous strict equality:
        - activeBalance == 0 (contracts/YieldStrategyManager.sol#756)
YieldStrategyManager.depositPrepare(address) (contracts/YieldStrategyManager.sol#737-771) uses a dangerous strict equality:
        - harvestable == 0 (contracts/YieldStrategyManager.sol#767)
YieldStrategyManager.preemptivelyHarvest(address) (contracts/YieldStrategyManager.sol#364-382) uses a dangerous strict equality:
        - harvestable == 0 (contracts/YieldStrategyManager.sol#378)
Limiters.get(Limiters.LinearGrowthLimiter) (contracts/libraries/Limiters.sol#95-103) uses a dangerous strict equality:
        - elapsed == 0 (contracts/libraries/Limiters.sol#97)
BeefyVaultMock.deposit(uint256) (contracts/test/beefy/BeefyVaultMock.sol#47-71) uses a dangerous strict equality:
        - totalSupply() == 0 (contracts/test/beefy/BeefyVaultMock.sol#64)
BeefyVaultMock.getPricePerFullShare() (contracts/test/beefy/BeefyVaultMock.sol#106-113) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/test/beefy/BeefyVaultMock.sol#108)
BeefyVaultMock.withdraw(uint256,address,uint256) (contracts/test/beefy/BeefyVaultMock.sol#85-104) uses a dangerous strict equality:
        - _shares == type()(uint256).max (contracts/test/beefy/BeefyVaultMock.sol#91)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
```

```
        External calls:
        - _preemptivelyHarvestDeposited(owner) (contracts/SavvyPositionManager.sol#1199)
                - _yieldStrategyManager.preemptivelyHarvest(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#779)
        - _distributeUnlockedCreditDeposited(owner) (contracts/SavvyPositionManager.sol#1203)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        - _syncAccount(owner) (contracts/SavvyPositionManager.sol#1207)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        State variables written after the call(s):
        - _syncAccount(owner) (contracts/SavvyPositionManager.sol#1207)
                - account.debt += amount (contracts/SavvyPositionManager.sol#952)
                - account.lastAccruedWeights[yieldToken] = currentAccruedWeight (contracts/SavvyPositionManager.sol#943)
Reentrancy in SavvyPositionManager._borrowCredit(address,uint256,address) (contracts/SavvyPositionManager.sol#1184-1218):
        External calls:
        - _preemptivelyHarvestDeposited(owner) (contracts/SavvyPositionManager.sol#1199)
                - _yieldStrategyManager.preemptivelyHarvest(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#779)
        - _distributeUnlockedCreditDeposited(owner) (contracts/SavvyPositionManager.sol#1203)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        - _syncAccount(owner) (contracts/SavvyPositionManager.sol#1207)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - _updateDebt(owner,SafeCast.toInt256(amount)) (contracts/SavvyPositionManager.sol#1208)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        State variables written after the call(s):
        - _updateDebt(owner,SafeCast.toInt256(amount)) (contracts/SavvyPositionManager.sol#1208)
                - account.debt += amount (contracts/SavvyPositionManager.sol#952)
        - _updateDebt(owner,SafeCast.toInt256(amount)) (contracts/SavvyPositionManager.sol#1208)
                - _userDebtBalances[user] = debt (contracts/SavvyPositionManager.sol#716)
        - _updateDebt(owner,SafeCast.toInt256(amount)) (contracts/SavvyPositionManager.sol#1208)
                - totalDebt += amount (contracts/SavvyPositionManager.sol#953)
        - _updateDebt(owner,SafeCast.toInt256(amount)) (contracts/SavvyPositionManager.sol#1208)
                - totalDebtBalance = totalDebtBalance - _userDebtBalances[user] + debt (contracts/SavvyPositionManager.sol#715)
Reentrancy in SavvyPositionManager._depositYieldToken(address,uint256,address) (contracts/SavvyPositionManager.sol#805-834):
        External calls:
        - yieldTokenParams = _yieldStrategyManager.depositPrepare(yieldToken) (contracts/SavvyPositionManager.sol#812)
        - _distributeUnlockedCreditDeposited(recipient) (contracts/SavvyPositionManager.sol#815)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        - _syncAccount(recipient,yieldToken) (contracts/SavvyPositionManager.sol#819)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        State variables written after the call(s):
        - _syncAccount(recipient,yieldToken) (contracts/SavvyPositionManager.sol#819)
                - account.debt += amount (contracts/SavvyPositionManager.sol#952)
                - account.lastAccruedWeights[yieldToken] = currentAccruedWeight (contracts/SavvyPositionManager.sol#943)
Reentrancy in SavvyPositionManager._depositYieldToken(address,uint256,address) (contracts/SavvyPositionManager.sol#805-834):
        External calls:
        - yieldTokenParams = _yieldStrategyManager.depositPrepare(yieldToken) (contracts/SavvyPositionManager.sol#812)
        - _distributeUnlockedCreditDeposited(recipient) (contracts/SavvyPositionManager.sol#815)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        - _syncAccount(recipient,yieldToken) (contracts/SavvyPositionManager.sol#819)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - shares = _issueSharesForAmount(recipient,yieldToken,amount) (contracts/SavvyPositionManager.sol#820)
                - shares = _yieldStrategyManager.issueSharesForAmount(yieldToken,amount) (contracts/SavvyPositionManager.sol#1060)
        State variables written after the call(s):
        - shares = _issueSharesForAmount(recipient,yieldToken,amount) (contracts/SavvyPositionManager.sol#820)
                - _accounts[recipient].balances[yieldToken] += shares (contracts/SavvyPositionManager.sol#1061)
Reentrancy in SavvyPositionManager._issueSharesForAmount(address,address,uint256) (contracts/SavvyPositionManager.sol#1051-1064):
        External calls:
        - shares = _yieldStrategyManager.issueSharesForAmount(yieldToken,amount) (contracts/SavvyPositionManager.sol#1060)
        State variables written after the call(s):
        - _accounts[recipient].balances[yieldToken] += shares (contracts/SavvyPositionManager.sol#1061)
Reentrancy in SavvyPositionManager._syncAccount(address,address) (contracts/SavvyPositionManager.sol#928-944):
        External calls:
        - _updateDebt(owner,- SafeCast.toInt256(unrealizedCredit)) (contracts/SavvyPositionManager.sol#942)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        State variables written after the call(s):
        - account.lastAccruedWeights[yieldToken] = currentAccruedWeight (contracts/SavvyPositionManager.sol#943)
Reentrancy in SavvyPositionManager._withdraw(address,address,uint256,address) (contracts/SavvyPositionManager.sol#847-876):
        External calls:
        - _preemptivelyHarvestDeposited(owner) (contracts/SavvyPositionManager.sol#856)
                - _yieldStrategyManager.preemptivelyHarvest(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#779)
        - _distributeUnlockedCreditDeposited(owner) (contracts/SavvyPositionManager.sol#860)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        - _syncAccount(owner) (contracts/SavvyPositionManager.sol#866)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        State variables written after the call(s):
        - _syncAccount(owner) (contracts/SavvyPositionManager.sol#866)
                - account.debt += amount (contracts/SavvyPositionManager.sol#952)
                - account.lastAccruedWeights[yieldToken] = currentAccruedWeight (contracts/SavvyPositionManager.sol#943)
Reentrancy in SavvyPositionManager._withdraw(address,address,uint256,address) (contracts/SavvyPositionManager.sol#847-876):
        External calls:
        - _preemptivelyHarvestDeposited(owner) (contracts/SavvyPositionManager.sol#856)
                - _yieldStrategyManager.preemptivelyHarvest(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#779)
        - _distributeUnlockedCreditDeposited(owner) (contracts/SavvyPositionManager.sol#860)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        - _syncAccount(owner) (contracts/SavvyPositionManager.sol#866)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - _burnShares(owner,yieldToken,shares) (contracts/SavvyPositionManager.sol#867)
                - _yieldStrategyManager.burnShares(yieldToken,shares) (contracts/SavvyPositionManager.sol#1075)
        State variables written after the call(s):
        - _burnShares(owner,yieldToken,shares) (contracts/SavvyPositionManager.sol#867)
                - account.balances[yieldToken] -= shares (contracts/SavvyPositionManager.sol#1074)
Reentrancy in SavvyPositionManager.donate(address,uint256) (contracts/SavvyPositionManager.sol#654-678):
        External calls:
        - _yieldStrategyManager.distributeUnlockedCredit(yieldToken) (contracts/SavvyPositionManager.sol#659)
        - _syncAccount(msg.sender) (contracts/SavvyPositionManager.sol#662)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - _accounts[msg.sender].lastAccruedWeights[yieldToken] = _yieldStrategyManager.donate(yieldToken,amount,shares) (contracts/SavvyPositionManager.sol#668)
        State variables written after the call(s):
        - _accounts[msg.sender].lastAccruedWeights[yieldToken] = _yieldStrategyManager.donate(yieldToken,amount,shares) (contracts/SavvyPositionManager.sol#668)
Reentrancy in SavvySyntheticToken.flashLoan(IERC3156FlashBorrower,address,uint256,bytes) (contracts/SavvySyntheticToken.sol#214-236):
        External calls:
        - (checker.checkState(receiver.onFlashLoan(msg.sender,token,amount,fee,data) == CALLBACK_SUCCESS,flash loan failed) (contracts/SavvySyntheticToken.sol#228-231)
        State variables written after the call(s):
        - _burn(address(receiver),amount + fee) (contracts/SavvySyntheticToken.sol#233)
                - _balances[account] = accountBalance - amount (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#288)
        - _burn(address(receiver),amount + fee) (contracts/SavvySyntheticToken.sol#233)
                - _totalSupply -= amount (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#290)
```

```
Reentrancy in YieldStrategyManager.harvest(address,uint256,uint256) (contracts/YieldStrategyManager.sol#522-563):
	External calls:
	- amountBaseTokens = _unwrap(yieldToken,harvestableAmount,savvyPositionManager,minimumAmountOut) (contracts/YieldStrategyManager.sol#552)
		- TokenUtils.safeTransferFrom(yieldToken,msg.sender,address(this),amount) (contracts/YieldStrategyManager.sol#809)
		- TokenUtils.safeApprove(yieldToken,address(adapter),amount) (contracts/YieldStrategyManager.sol#811)
		- (success,data) = token.call(abi.encodeWithSelector(IERC20Minimal.approve.selector,spender,value)) (contracts/libraries/TokenUtils.sol#84-86)
		- (success,data) = token.call(abi.encodeWithSelector(IERC20Minimal.transferFrom.selector,owner,recipient,amount)) (contracts/libraries/TokenUtils.sol#102-104)
		- amountUnwrapped = adapter.unwrap(amount,recipient) (contracts/YieldStrategyManager.sol#812)
	State variables written after the call(s):
	- _distributeCredit(yieldToken,credit) (contracts/YieldStrategyManager.sol#561)
		- yieldTokenParams.accruedWeight += unlockedCredit * FIXED_POINT_SCALAR / yieldTokenParams.totalShares (contracts/YieldStrategyManager.sol#843)
		- yieldTokenParams.pendingCredit = amount + lockedCredit (contracts/YieldStrategyManager.sol#846)
		- yieldTokenParams.distributedCredit = 0 (contracts/YieldStrategyManager.sol#847)
		- yieldTokenParams.lastDistributionBlock = block.number (contracts/YieldStrategyManager.sol#848)
Reentrancy in SavvyPositionManager.repayWithBaseToken(address,uint256,address) (contracts/SavvyPositionManager.sol#554-597):
	External calls:
	- _distributeUnlockedCreditDeposited(recipient) (contracts/SavvyPositionManager.sol#567)
		- _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
	- _syncAccount(recipient) (contracts/SavvyPositionManager.sol#570)
		- ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
	State variables written after the call(s):
	- _syncAccount(recipient) (contracts/SavvyPositionManager.sol#570)
		- account.debt += amount (contracts/SavvyPositionManager.sol#952)
		- account.lastAccruedWeights[yieldToken] = currentAccruedWeight (contracts/SavvyPositionManager.sol#943)
Reentrancy in SavvyPositionManager.repayWithBaseToken(address,uint256,address) (contracts/SavvyPositionManager.sol#554-597):
	External calls:
	- _distributeUnlockedCreditDeposited(recipient) (contracts/SavvyPositionManager.sol#567)
		- _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
	- _syncAccount(recipient) (contracts/SavvyPositionManager.sol#570)
		- ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
	- (credit,actualAmount) = _yieldStrategyManager.repayWithBaseToken(baseToken,_amount,debt) (contracts/SavvyPositionManager.sol#580-583)
	- _updateDebt(recipient,-SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#586)
		- ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
	State variables written after the call(s):
	- _updateDebt(recipient,-SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#586)
		- account.debt += amount (contracts/SavvyPositionManager.sol#952)
	- _updateDebt(recipient,-SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#586)
		- _userDebtBalances[user] = debt (contracts/SavvyPositionManager.sol#716)
	- _updateDebt(recipient,-SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#586)
		- totalDebt += amount (contracts/SavvyPositionManager.sol#953)
	- _updateDebt(recipient,-SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#586)
		- totalDebtBalance = totalDebtBalance - _userDebtBalances[user] + debt (contracts/SavvyPositionManager.sol#715)
Reentrancy in SavvyPositionManager.repayWithCollateral(address,uint256,uint256) (contracts/SavvyPositionManager.sol#600-651):
	External calls:
	- TokenUtils.safeApprove(yieldToken,address(_yieldStrategyManager),type(uint256).max) (contracts/SavvyPositionManager.sol#619)
	- (amountBaseTokens,amountYieldTokens,actualShares) = _yieldStrategyManager.repayWithCollateral(yieldToken,address(this),shares,minimumAmountOut,unrealizedDebt) (contracts/SavvyPositionManager.sol#620-624)
	- _distributeUnlockedCreditDeposited(msg.sender) (contracts/SavvyPositionManager.sol#627)
		- _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
	- _syncAccount(msg.sender,yieldToken) (contracts/SavvyPositionManager.sol#633)
		- ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
	State variables written after the call(s):
	- _syncAccount(msg.sender,yieldToken) (contracts/SavvyPositionManager.sol#633)
		- account.debt += amount (contracts/SavvyPositionManager.sol#952)
		- account.lastAccruedWeights[yieldToken] = currentAccruedWeight (contracts/SavvyPositionManager.sol#943)
Reentrancy in SavvyPositionManager.repayWithCollateral(address,uint256,uint256) (contracts/SavvyPositionManager.sol#600-651):
	External calls:
	- TokenUtils.safeApprove(yieldToken,address(_yieldStrategyManager),type(uint256).max) (contracts/SavvyPositionManager.sol#619)
	- (amountBaseTokens,amountYieldTokens,actualShares) = _yieldStrategyManager.repayWithCollateral(yieldToken,address(this),shares,minimumAmountOut,unrealizedDebt) (contracts/SavvyPositionManager.sol#620-624)
	- _distributeUnlockedCreditDeposited(msg.sender) (contracts/SavvyPositionManager.sol#627)
		- _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
	- _syncAccount(msg.sender,yieldToken) (contracts/SavvyPositionManager.sol#633)
		- ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
	- _burnShares(msg.sender,yieldToken,actualShares) (contracts/SavvyPositionManager.sol#634)
		- _yieldStrategyManager.burnShares(yieldToken,shares) (contracts/SavvyPositionManager.sol#1075)
	State variables written after the call(s):
	- _burnShares(msg.sender,yieldToken,actualShares) (contracts/SavvyPositionManager.sol#634)
		- account.balances[yieldToken] -= shares (contracts/SavvyPositionManager.sol#1074)
Reentrancy in SavvyPositionManager.repayWithCollateral(address,uint256,uint256) (contracts/SavvyPositionManager.sol#600-651):
	External calls:
	- TokenUtils.safeApprove(yieldToken,address(_yieldStrategyManager),type(uint256).max) (contracts/SavvyPositionManager.sol#619)
	- (amountBaseTokens,amountYieldTokens,actualShares) = _yieldStrategyManager.repayWithCollateral(yieldToken,address(this),shares,minimumAmountOut,unrealizedDebt) (contracts/SavvyPositionManager.sol#620-624)
	- _distributeUnlockedCreditDeposited(msg.sender) (contracts/SavvyPositionManager.sol#627)
		- _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
	- _syncAccount(msg.sender,yieldToken) (contracts/SavvyPositionManager.sol#633)
		- ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
	- _burnShares(msg.sender,yieldToken,actualShares) (contracts/SavvyPositionManager.sol#634)
		- _yieldStrategyManager.burnShares(yieldToken,shares) (contracts/SavvyPositionManager.sol#1075)
	- _updateDebt(msg.sender,-SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#635)
		- ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
	State variables written after the call(s):
	- _updateDebt(msg.sender,-SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#635)
		- account.debt += amount (contracts/SavvyPositionManager.sol#952)
	- _updateDebt(msg.sender,-SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#635)
		- _userDebtBalances[user] = debt (contracts/SavvyPositionManager.sol#716)
	- _updateDebt(msg.sender,-SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#635)
		- totalDebt += amount (contracts/SavvyPositionManager.sol#953)
	- _updateDebt(msg.sender,-SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#635)
		- totalDebtBalance = totalDebtBalance - _userDebtBalances[user] + debt (contracts/SavvyPositionManager.sol#715)
Reentrancy in YieldStrategyManager.repayWithCollateral(address,address,uint256,uint256,int256) (contracts/YieldStrategyManager.sol#479-519):
	External calls:
	- TokenUtils.safeTransferFrom(yieldToken,msg.sender,address(this),amountYieldTokens) (contracts/YieldStrategyManager.sol#501)
	- amountBaseTokens = unwrap(yieldToken,amountYieldTokens,recipient,minimumAmountOut) (contracts/YieldStrategyManager.sol#502)
		- (success,data) = token.call(abi.encodeWithSelector(IERC20Minimal.approve.selector,spender,value)) (contracts/libraries/TokenUtils.sol#84-86)
		- TokenUtils.safeApprove(yieldToken,address(adapter),amount) (contracts/YieldStrategyManager.sol#438)
		- amountUnwrapped = adapter.unwrap(amount,recipient) (contracts/YieldStrategyManager.sol#439)
	State variables written after the call(s):
	- preemptivelyHarvest(yieldToken) (contracts/YieldStrategyManager.sol#516)
		- _yieldTokens[yieldToken].activeBalance -= harvestable (contracts/YieldStrategyManager.sol#909)
		- _yieldTokens[yieldToken].harvestableBalance += harvestable (contracts/YieldStrategyManager.sol#910)
Reentrancy in SavvyPositionManager.repayWithDebtToken(uint256,address) (contracts/SavvyPositionManager.sol#510-551):
	External calls:
	- _distributeUnlockedCreditDeposited(recipient) (contracts/SavvyPositionManager.sol#517)
		- _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
	- _syncAccount(recipient) (contracts/SavvyPositionManager.sol#520)
		- ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
	State variables written after the call(s):
	- _syncAccount(recipient) (contracts/SavvyPositionManager.sol#520)
		- account.debt += amount (contracts/SavvyPositionManager.sol#952)
		- account.lastAccruedWeights[yieldToken] = currentAccruedWeight (contracts/SavvyPositionManager.sol#943)
```

```
Reentrancy in SavvyPositionManager.repayWithDebtToken(uint256,address) (contracts/SavvyPositionManager.sol#510-551):
        External calls:
        - _distributeUnlockedCreditDeposited(recipient) (contracts/SavvyPositionManager.sol#517)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        - _syncAccount(recipient) (contracts/SavvyPositionManager.sol#520)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - _updateDebt(recipient,- SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#538)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        State variables written after the call(s):
        - _updateDebt(recipient,- SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#538)
                - account.debt += amount (contracts/SavvyPositionManager.sol#952)
        - _updateDebt(recipient,- SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#538)
                - _userDebtBalances[user] = debt (contracts/SavvyPositionManager.sol#716)
        - _updateDebt(recipient,- SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#538)
                - totalDebt += amount (contracts/SavvyPositionManager.sol#953)
        - _updateDebt(recipient,- SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#538)
                - totalDebtBalance = totalDebtBalance - _userDebtBalances[user] + debt (contracts/SavvyPositionManager.sol#715)
Reentrancy in SavvySage.setFlowRate(address,uint256) (contracts/SavvySage.sol#286-295):
        External calls:
        - _swap(baseToken) (contracts/SavvySage.sol#291)
                - ISavvyPositionManager(savvy).syncAccount(address(this)) (contracts/SavvySage.sol#461)
                - ISavvySwap(savvySwap[baseToken]).swap(swapDelta) (contracts/SavvySage.sol#543)
        State variables written after the call(s):
        - flowRate[baseToken] = _flowRate (contracts/SavvySage.sol#293)
Reentrancy in SavvySage.setSavvy(address) (contracts/SavvySage.sol#232-251):
        External calls:
        - TokenUtils.safeApprove(debtToken,savvy,0) (contracts/SavvySage.sol#240)
        State variables written after the call(s):
        - savvy = _savvy (contracts/SavvySage.sol#243)
Reentrancy in VeSvy.stake(uint256) (contracts/VeSvy.sol#197-221):
        External calls:
        - _claim(sender) (contracts/VeSvy.sol#208)
                - savvyBooster.updatePendingRewardsWithVeSvy(_addr,userVeSvyBal,totalVeSvyBal) (contracts/VeSvy.sol#363)
        State variables written after the call(s):
        - users[sender].amount += _amount (contracts/VeSvy.sol#210)
Reentrancy in SavvyPositionManager.syncAccount(address) (contracts/SavvyPositionManager.sol#387-392):
        External calls:
        - _preemptivelyHarvestDeposited(owner) (contracts/SavvyPositionManager.sol#389)
                - _yieldStrategyManager.preemptivelyHarvest(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#779)
        - _distributeUnlockedCreditDeposited(owner) (contracts/SavvyPositionManager.sol#390)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        - _syncAccount(owner) (contracts/SavvyPositionManager.sol#391)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        State variables written after the call(s):
        - _syncAccount(owner) (contracts/SavvyPositionManager.sol#391)
                - account.debt += amount (contracts/SavvyPositionManager.sol#952)
                - account.lastAccruedWeights[yieldToken] = currentAccruedWeight (contracts/SavvyPositionManager.sol#943)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

VeSvy._assertNotContract(address) (contracts/VeSvy.sol#226-233) uses tx.origin for authorization: _addr != tx.origin (contracts/VeSvy.sol#227)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-usage-of-txorigin

SavvyBooster._getPoolIndex(uint256) (contracts/SavvyBooster.sol#320-332) contains a tautology or contradiction:
        - i >= 0 (contracts/SavvyBooster.sol#326)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction


SavvyPositionManager.repayWithBaseToken(address,uint256,address).debt (contracts/SavvyPositionManager.sol#577) is a local variable never initialized
SavvyLGE._setVestModes(ISavvyLGE.VestMode[]).i (contracts/SavvyLGE.sol#374) is a local variable never initialized
SavvyPositionManager.repayWithDebtToken(uint256,address).debt (contracts/SavvyPositionManager.sol#527) is a local variable never initialized
SavvyPositionManager.repayWithCollateral(address,uint256,uint256).unrealizedDebt (contracts/SavvyPositionManager.sol#616) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#388-409) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#395-405)
InfoAggregator.initialize(address[],ISavvyInfoAggregatorStructs.SupportTokenInfo[],ISavvyPriceFeed,ISavvyToken,ISavvyBooster,IVeSvy) (contracts/InfoAggregator.sol#51-86) ignores return value by savvyPositionManagers.add(savvyPositionManager) (contracts/InfoAggregator.sol#74)
InfoAggregator.addSavvyPositionManager(address[]) (contracts/InfoAggregator.sol#89-99) ignores return value by savvyPositionManagers.add(savvyPositionManager) (contracts/InfoAggregator.sol#97)
SavvyBooster.addSavvyPositionManagers(ISavvyPositionManager[]) (contracts/SavvyBooster.sol#94-117) ignores return value by savvyPositionManagers.add(address(savvyPositionManager)) (contracts/SavvyBooster.sol#115)
SavvyLoanMigrator.migrateVaults(address,address,uint256,uint256,uint256) (contracts/SavvyLoanMigrator.sol#49-107) ignores return value by savvyPositionManager.repayWithDebtToken(cmintable,msg.sender) (contracts/SavvyLoanMigrator.sol#83)
SavvyPositionManager.repayWithCollateral(address,uint256,uint256) (contracts/SavvyPositionManager.sol#600-651) ignores return value by _yieldStrategyManager.syncYieldToken(yieldToken,amountYieldTokens,false) (contracts/SavvyPositionManager.sol#637)
SavvyPositionManager._withdraw(address,address,uint256,address) (contracts/SavvyPositionManager.sol#847-876) ignores return value by _yieldStrategyManager.syncYieldToken(yieldToken,amountYieldTokens,false) (contracts/SavvyPositionManager.sol#868)
SavvyPositionManager._issueSharesForAmount(address,address,uint256) (contracts/SavvyPositionManager.sol#1051-1064) ignores return value by _accounts[recipient].depositedTokens.add(yieldToken) (contracts/SavvyPositionManager.sol#1057)
SavvyPositionManager._burnShares(address,address,uint256) (contracts/SavvyPositionManager.sol#1071-1080) ignores return value by account.depositedTokens.remove(yieldToken) (contracts/SavvyPositionManager.sol#1078)
SavvySage.withdrawFromSavvy(address,uint256,uint256) (contracts/SavvySage.sol#355-361) ignores return value by ISavvyPositionManager(savvy).withdrawBaseToken(yieldToken,shares,address(this),minimumAmountOut) (contracts/SavvySage.sol#360)
SavvySage._savvyDeposit(address,uint256) (contracts/SavvySage.sol#483-490) ignores return value by ISavvyPositionManager(savvy).depositBaseToken(token,amount,address(this),0) (contracts/SavvySage.sol#484-489)
SavvySage._savvyWithdraw(address,uint256) (contracts/SavvySage.sol#496-505) ignores return value by ISavvyPositionManager(savvy).withdrawBaseToken(token,wantShares,address(this),minimumAmountOut) (contracts/SavvySage.sol#507)
SavvySwap.initialize(address,address,address,address) (contracts/SavvySwap.sol#142-163) ignores return value by ticks.next() (contracts/SavvySwap.sol#159)
WrapTokenGateway.refreshAllowance(address) (contracts/WrapTokenGateway.sol#49-51) ignores return value by WAVAX.approve(savvy,type()(uint256).max) (contracts/WrapTokenGateway.sol#50)
WrapTokenGateway.depositBaseToken(address,address,uint256,address,uint256) (contracts/WrapTokenGateway.sol#54-67) ignores return value by ISavvyPositionManager(_savvyPositionManager).depositBaseToken(_yieldToken,_amount,_recipient,_minimumAmountOut) (contracts/WrapTokenGateway.sol#66)
YieldStrategyManager.addBaseToken(address,address,ISavvyAdminActions.BaseTokenConfig) (contracts/YieldStrategyManager.sol#190-221) ignores return value by _supportedBaseTokens.add(baseToken) (contracts/YieldStrategyManager.sol#220)
YieldStrategyManager.addYieldToken(address,ISavvyAdminActions.YieldTokenConfig) (contracts/YieldStrategyManager.sol#224-257) ignores return value by _supportedYieldTokens.add(yieldToken) (contracts/YieldStrategyManager.sol#256)
TestFlashBorrower.takeLoan(address,address,uint256) (contracts/test/TestFlashBorrower.sol#13-21) ignores return value by IERC20Minimal(token).approve(flashLender,amount + fee) (contracts/test/TestFlashBorrower.sol#19)
TestFlashBorrower.takeLoan(address,address,uint256) (contracts/test/TestFlashBorrower.sol#13-21) ignores return value by IERC3156FlashLender(flashLender).flashLoan(IERC3156FlashBorrower(address(this)),token,amount,bytes()) (contracts/test/TestFlashBorrower.sol#20)
TestVeSvy.stake(uint256) (contracts/test/TestVeSvy.sol#18-22) ignores return value by IERC20(svyToken).approve(address(veSvy),amount) (contracts/test/TestVeSvy.sol#20)
Allowlist.add(address) (contracts/utils/Allowlist.sol#30-35) ignores return value by addresses.add(caller) (contracts/utils/Allowlist.sol#33)
Allowlist.remove(address) (contracts/utils/Allowlist.sol#38-43) ignores return value by addresses.remove(caller) (contracts/utils/Allowlist.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

buy(uint256,address,uint256,uint256) should be declared external:
        - SavvyLGE.buy(uint256,address,uint256,uint256) (contracts/SavvyLGE.sol#202-215)
initialize(ISavvyLoanMigrator.InitializationParams) should be declared external:
        - SavvyLoanMigrator.initialize(ISavvyLoanMigrator.InitializationParams) (contracts/SavvyLoanMigrator.sol#32-46)
initialize() should be declared external:
        - SavvyPriceFeed.initialize() (contracts/SavvyPriceFeed.sol#19-21)
name() should be declared external:
        - VeERC20Upgradeable.name() (contracts/VeERC20Upgradeable.sol#49-51)
symbol() should be declared external:
        - VeERC20Upgradeable.symbol() (contracts/VeERC20Upgradeable.sol#57-59)
decimals() should be declared external:
        - VeERC20Upgradeable.decimals() (contracts/VeERC20Upgradeable.sol#74-76)
initialize(IERC20) should be declared external:
        - VeSvy.initialize(IERC20) (contracts/VeSvy.sol#93-116)
initialize(address,address,address) should be declared external:
        - WrapTokenGateway.initialize(address,address,address) (contracts/WrapTokenGateway.sol#30-39)
initialize() should be declared external:
        - YieldStrategyManager.initialize() (contracts/YieldStrategyManager.sol#52-54)
initialize(address,address) should be declared external:
        - BeefyTokenAdapter.initialize(address,address) (contracts/adapters/beefy/BeefyTokenAdapter.sol#24-31)
initialize(address,address) should be declared external:
        - VesperTokenAdapter.initialize(address,address) (contracts/adapters/vesper/VesperTokenAdapter.sol#25-33)
initialize(address,address,address) should be declared external:
        - YieldYakAdapter.initialize(address,address,address) (contracts/adapters/yieldyak/YieldYakAdapter.sol#28-44)
transfer(address,uint256) should be declared external:
        - TestERC20.transfer(address,uint256) (contracts/test/TestERC20.sol#27-38)
approve(address,uint256) should be declared external:
        - TestERC20.approve(address,uint256) (contracts/test/TestERC20.sol#40-44)
transferFrom(address,address,uint256) should be declared external:
        - TestERC20.transferFrom(address,address,uint256) (contracts/test/TestERC20.sol#46-67)
available() should be declared external:
        - BeefyVaultMock.available() (contracts/test/beefy/BeefyVaultMock.sol#39-41)
withdraw(uint256,address) should be declared external:
        - BeefyVaultMock.withdraw(uint256,address) (contracts/test/beefy/BeefyVaultMock.sol#81-83)
initialize() should be declared external:
        - Allowlist.initialize() (contracts/utils/Allowlist.sol#20-22)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
. analyzed (124 contracts with 78 detectors), 653 result(s) found
```

```
SavvyProtocolToken.constructor(string,string)._name (contracts/SavvyProtocolToken.sol#21) shadows:
        - ERC20._name (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#42) (state variable)
SavvyProtocolToken.constructor(string,string)._symbol (contracts/SavvyProtocolToken.sol#22) shadows:
        - ERC20._symbol (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#43) (state variable)
SavvySyntheticToken.constructor(string,string,uint256)._name (contracts/SavvySyntheticToken.sol#57) shadows:
        - ERC20._name (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#42) (state variable)
SavvySyntheticToken.constructor(string,string,uint256)._symbol (contracts/SavvySyntheticToken.sol#57) shadows:
        - ERC20._symbol (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#43) (state variable)
ISavvyState.admin().admin (contracts/interfaces/savvy/ISavvyState.sol#31) shadows:
        - ISavvyState.admin() (contracts/interfaces/savvy/ISavvyState.sol#31) (function)
ISavvyState.totalDebt().totalDebt (contracts/interfaces/savvy/ISavvyState.sol#35) shadows:
        - ISavvyState.totalDebt() (contracts/interfaces/savvy/ISavvyState.sol#35) (function)
ISavvyState.pendingAdmin().pendingAdmin (contracts/interfaces/savvy/ISavvyState.sol#40) shadows:
        - ISavvyState.pendingAdmin() (contracts/interfaces/savvy/ISavvyState.sol#40) (function)
ISavvyState.savvySwap().savvySwap (contracts/interfaces/savvy/ISavvyState.sol#50) shadows:
        - ISavvyState.savvySwap() (contracts/interfaces/savvy/ISavvyState.sol#50) (function)
ISavvyState.svyBooster().svyBooster (contracts/interfaces/savvy/ISavvyState.sol#64) shadows:
        - ISavvyState.svyBooster() (contracts/interfaces/savvy/ISavvyState.sol#64) (function)
ISavvyState.minimumCollateralization().minimumCollateralization (contracts/interfaces/savvy/ISavvyState.sol#73) shadows:
        - ISavvyState.minimumCollateralization() (contracts/interfaces/savvy/ISavvyState.sol#73) (function)
ISavvyState.protocolFee().protocolFee (contracts/interfaces/savvy/ISavvyState.sol#78) shadows:
        - ISavvyState.protocolFee() (contracts/interfaces/savvy/ISavvyState.sol#78) (function)
ISavvyState.protocolFeeReceiver().protocolFeeReceiver (contracts/interfaces/savvy/ISavvyState.sol#83) shadows:
        - ISavvyState.protocolFeeReceiver() (contracts/interfaces/savvy/ISavvyState.sol#83) (function)
ISavvyState.allowlist().allowlist (contracts/interfaces/savvy/ISavvyState.sol#88) shadows:
        - ISavvyState.allowlist() (contracts/interfaces/savvy/ISavvyState.sol#88) (function)
ISavvyState.redlistActive().redlistActive (contracts/interfaces/savvy/ISavvyState.sol#93) shadows:
        - ISavvyState.redlistActive() (contracts/interfaces/savvy/ISavvyState.sol#93) (function)
ISavvyState.wrapTokenGateway().wrapTokenGateway (contracts/interfaces/savvy/ISavvyState.sol#98) shadows:
        - ISavvyState.wrapTokenGateway() (contracts/interfaces/savvy/ISavvyState.sol#98) (function)
IYieldStrategyManagerStates.setBorrowingLimiter(limiters.LinearGrowthLimiter).borrowingLimiter (contracts/interfaces/savvy/IYieldStrategyManagerStates.sol#59) shadows:
        - IYieldStrategyManagerStates.borrowingLimiter() (contracts/interfaces/savvy/IYieldStrategyManagerStates.sol#331) (function)
ISavvySwap.allowlist().allowlist (contracts/interfaces/savvySwap/ISavvySwap.sol#77) shadows:
        - ISavvySwap.allowlist() (contracts/interfaces/savvySwap/ISavvySwap.sol#77) (function)
ERC20Mock.constructor(string,string,uint256,uint8)._name (contracts/test/ERC20Mock.sol#13) shadows:
        - ERC20._name (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#42) (state variable)
ERC20Mock.constructor(string,string,uint256,uint8)._symbol (contracts/test/ERC20Mock.sol#14) shadows:
        - ERC20._symbol (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#43) (state variable)
ERC721Mock.constructor(string,string)._name (contracts/test/ERC721Mock.sol#8) shadows:
        - ERC721._name (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#24) (state variable)
ERC721Mock.constructor(string,string)._symbol (contracts/test/ERC721Mock.sol#8) shadows:
        - ERC721._symbol (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#27) (state variable)
BeefyVaultMock.getPricePerFullShare()._totalSupply (contracts/test/beefy/BeefyVaultMock.sol#107) shadows:
        - ERC20._totalSupply (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#40) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SavvySwap.initialize(address,address,address,address) (contracts/SavvySwap.sol#142-163) should emit an event for:
        - buffer = _buffer (contracts/SavvySwap.sol#157)
SavvySwap.setCollateralSource(address) (contracts/SavvySwap.sol#193-196) should emit an event for:
        - buffer = _newCollateralSource (contracts/SavvySwap.sol#195)
YieldStrategyManager.setSavvyPositionManager(address) (contracts/YieldStrategyManager.sol#64-66) should emit an event for:
        - savvyPositionManager = savvyPositionManager_ (contracts/YieldStrategyManager.sol#65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

SavvyLGE.initialize(address,address,address,uint256,uint256,uint256,uint256,ISavvyLGE.VestMode[]) (contracts/SavvyLGE.sol#73-104) should emit an event for:
        - vestStartOffset = _vestStartOffset (contracts/SavvyLGE.sol#87)
        - pricePerAllotment = 10 ** ERC20(_depositToken).decimals() (contracts/SavvyLGE.sol#91)
SavvySwap.initialize(address,address,address,address) (contracts/SavvySwap.sol#142-163) should emit an event for:
        - conversionFactor = 10 ** (debtTokenDecimals - baseTokenDecimals) (contracts/SavvySwap.sol#156)
SavvySyntheticToken.setMaxFlashLoan(uint256) (contracts/SavvySyntheticToken.sol#177-179) should emit an event for:
        - maxFlashLoanAmount = _maxFlashLoanAmount (contracts/SavvySyntheticToken.sol#178)
VeSvy.setMaxCap(uint256) (contracts/VeSvy.sol#150-153) should emit an event for:
        - maxCap = _maxCap (contracts/VeSvy.sol#152)
VeSvy.setGenerationRate(uint256) (contracts/VeSvy.sol#157-160) should emit an event for:
        - generationRate = _generationRate (contracts/VeSvy.sol#159)
VeSvy.setInvVoteThreshold(uint256) (contracts/VeSvy.sol#165-169) should emit an event for:
        - invVoteThreshold = _invVoteThreshold (contracts/VeSvy.sol#168)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

SavvyLGE.initialize(address,address,address,uint256,uint256,uint256,uint256,ISavvyLGE.VestMode[])._depositTokenWallet (contracts/SavvyLGE.sol#76) lacks a zero-check on :
                - depositTokenWallet = _depositTokenWallet (contracts/SavvyLGE.sol#94)
SavvyPositionManager.setPendingAdmin(address).value (contracts/SavvyPositionManager.sol#200) lacks a zero-check on :
                - pendingAdmin = value (contracts/SavvyPositionManager.sol#202)
SavvyPositionManager.setSavvySwap(address)._savvySwap (contracts/SavvyPositionManager.sol#290) lacks a zero-check on :
                - savvySwap = _savvySwap (contracts/SavvyPositionManager.sol#293)
SavvyPositionManager.setProtocolFeeReceiver(address).value (contracts/SavvyPositionManager.sol#313) lacks a zero-check on :
                - protocolFeeReceiver = value (contracts/SavvyPositionManager.sol#316)
SavvyPriceFeed.updateSvyPriceFeed(address).newFeed (contracts/SavvyPriceFeed.sol#35) lacks a zero-check on :
                - svyPriceFeed = newFeed (contracts/SavvyPriceFeed.sol#38)
SavvyRedlist.initialize(bool,address,address,address,address[]).allowlist_ (contracts/SavvyRedlist.sol#55) lacks a zero-check on :
                - allowlist = allowlist_ (contracts/SavvyRedlist.sol#71)
SavvyRedlist.initialize(bool,address,address,address,address[]).protocolToken_ (contracts/SavvyRedlist.sol#53) lacks a zero-check on :
                - protocolToken = protocolToken_ (contracts/SavvyRedlist.sol#73)
SavvyRedlist.initialize(bool,address,address,address,address[]).vaProtocolToken_ (contracts/SavvyRedlist.sol#54) lacks a zero-check on :
                - vaProtocolToken = vaProtocolToken_ (contracts/SavvyRedlist.sol#74)
SavvyRedlist.setAllowlist(address).allowlist_ (contracts/SavvyRedlist.sol#100) lacks a zero-check on :
                - allowlist = allowlist_ (contracts/SavvyRedlist.sol#102)
SavvySage.initialize(address,address)._debtToken (contracts/SavvySage.sol#85) lacks a zero-check on :
                - debtToken = _debtToken (contracts/SavvySage.sol#90)
SavvySage.setSavvy(address)._savvy (contracts/SavvySage.sol#232) lacks a zero-check on :
                - savvy = _savvy (contracts/SavvySage.sol#243)
SavvySwap.initialize(address,address,address,address)._syntheticToken (contracts/SavvySwap.sol#143) lacks a zero-check on :
                - syntheticToken = _syntheticToken (contracts/SavvySwap.sol#152)
SavvySwap.initialize(address,address,address,address)._baseToken (contracts/SavvySwap.sol#144) lacks a zero-check on :
                - baseToken = _baseToken (contracts/SavvySwap.sol#153)
SavvySwap.initialize(address,address,address,address)._buffer (contracts/SavvySwap.sol#145) lacks a zero-check on :
                - buffer = _buffer (contracts/SavvySwap.sol#157)
SavvySwap.initialize(address,address,address,address)._allowlist (contracts/SavvySwap.sol#146) lacks a zero-check on :
                - allowlist = _allowlist (contracts/SavvySwap.sol#162)
SavvySwap.setCollateralSource(address)._newCollateralSource (contracts/SavvySwap.sol#193) lacks a zero-check on :
                - buffer = _newCollateralSource (contracts/SavvySwap.sol#195)
WrapTokenGateway.initialize(address,address,address)._allowlist (contracts/WrapTokenGateway.sol#32) lacks a zero-check on :
                - allowlist = _allowlist (contracts/WrapTokenGateway.sol#36)
WrapTokenGateway.initialize(address,address,address)._savvyRedlist (contracts/WrapTokenGateway.sol#33) lacks a zero-check on :
                - savvyRedlist = _savvyRedlist (contracts/WrapTokenGateway.sol#37)
WrapTokenGateway.withdrawBaseToken(address,address,uint256,address,uint256)._recipient (contracts/WrapTokenGateway.sol#74) lacks a zero-check on :
                - (success) = _recipient.call{value: amount}(new bytes(0)) (contracts/WrapTokenGateway.sol#86)
YieldStrategyManager.setSavvyPositionManager(address).savvyPositionManager_ (contracts/YieldStrategyManager.sol#64) lacks a zero-check on :
                - savvyPositionManager = savvyPositionManager_ (contracts/YieldStrategyManager.sol#65)
BeefyTokenAdapter.initialize(address,address)._token (contracts/adapters/beefy/BeefyTokenAdapter.sol#25) lacks a zero-check on :
                - token = _token (contracts/adapters/beefy/BeefyTokenAdapter.sol#29)
BeefyTokenAdapter.initialize(address,address)._baseToken (contracts/adapters/beefy/BeefyTokenAdapter.sol#26) lacks a zero-check on :
```

```
InfoAggregator._getUserAvailableCreditUSD(ISavvyPositionManager,address) (contracts/InfoAggregator.sol#566-577) has external calls inside a loop: minCollateralization = savvyPositionManager_.minimumCollateralization() (contracts/InfoAggregator.sol#570)
InfoAggregator._getUserDebtValueUSD(ISavvyPositionManager,address) (contracts/InfoAggregator.sol#546-559) has external calls inside a loop: (debt) = savvyPositionManager_.accounts(user_) (contracts/InfoAggregator.sol#550)
InfoAggregator._getUserDepositedTokenPrice(address,address,address) (contracts/InfoAggregator.sol#616-637) has external calls inside a loop: yieldStrategyManager = ISavvyPositionManager(savvyPositionManager_).yieldStrategyManager() (contracts/InfoAggregator.sol#621)
InfoAggregator._getUserDepositedTokenPrice(address,address,address) (contracts/InfoAggregator.sol#616-637) has external calls inside a loop: yieldStrategyManager.isSupportedYieldToken(yieldToken_) == false (contracts/InfoAggregator.sol#622)
InfoAggregator._getUserDepositedTokenPrice(address,address,address) (contracts/InfoAggregator.sol#616-637) has external calls inside a loop: (shares) = ISavvyPositionManager(savvyPositionManager_).positions(user_,yieldToken_) (contracts/InfoAggregator.sol#626)
InfoAggregator._getUserDepositedTokenPrice(address,address,address) (contracts/InfoAggregator.sol#616-637) has external calls inside a loop: yieldTokenParams = yieldStrategyManager.getYieldTokenParameters(yieldToken_) (contracts/InfoAggregator.sol#628)
InfoAggregator._getUserDepositedTokenPrice(address,address,address) (contracts/InfoAggregator.sol#616-637) has external calls inside a loop: pricePerShare = ITokenAdapter(yieldTokenParams.adapter).price() (contracts/InfoAggregator.sol#629)
InfoAggregator.getAvailableDepositTokenAmount(address) (contracts/InfoAggregator.sol#368-386) has external calls inside a loop: amount = IERC20(baseToken).balanceOf(user_) (contracts/InfoAggregator.sol#377)
InfoAggregator.getAvailableCreditToken(address) (contracts/InfoAggregator.sol#389-408) has external calls inside a loop: minCollateralization = savvyPositionManager.minimumCollateralization() (contracts/InfoAggregator.sol#398)
InfoAggregator.getPoolDeposited(address,address) (contracts/InfoAggregator.sol#411-433) has external calls inside a loop: yieldStrategyManager = ISavvyPositionManager(savvyPositionManager).yieldStrategyManager() (contracts/InfoAggregator.sol#426)
InfoAggregator.getPoolDeposited(address,address) (contracts/InfoAggregator.sol#411-433) has external calls inside a loop: yieldStrategyManager.isSupportedYieldToken(yieldToken) == true (contracts/InfoAggregator.sol#427)
SavvyActionBatcher.borrowAllCredit(address) (contracts/SavvyActionBatcher.sol#27-42) has external calls inside a loop: ISavvyPositionManager(savvyPositionManager).borrowCreditFrom(sender,borrowableAmount,recipient_) (contracts/SavvyActionBatcher.sol#39)
Multicall.multicall(bytes[]) (contracts/base/Multicall.sol#12-23) has external calls inside a loop: (success,result) = address(this).delegatecall(data[i]) (contracts/base/Multicall.sol#15)
SavvyRedlist._isNFTOwner(address,address) (contracts/SavvyRedlist.sol#190-196) has external calls inside a loop: IERC721(nftCollection_).balanceOf(account_) > 0 (contracts/SavvyRedlist.sol#195)
SavvySage._getTotalBuffered(address) (contracts/SavvySage.sol#422-433) has external calls inside a loop: (balance) = ISavvyPositionManager(savvy).positions(address(this),yieldToken) (contracts/SavvySage.sol#427)
SavvySage._getTotalBuffered(address) (contracts/SavvySage.sol#422-433) has external calls inside a loop: params = yieldStrategyManager.getYieldTokenParameters(yieldToken) (contracts/SavvySage.sol#428-429)
SavvySage._getTotalBuffered(address) (contracts/SavvySage.sol#422-433) has external calls inside a loop: tokensPerShare = yieldStrategyManager.getBaseTokensPerShare(yieldToken) (contracts/SavvySage.sol#430-431)
SavvySage.setWeights(address,address[],uint256[]) (contracts/SavvySage.sol#180-215) has external calls inside a loop: params = yieldStrategyManager.getYieldTokenParameters(yieldToken) (contracts/SavvySage.sol#203-204)
SavvySage.setWeights(address,address[],uint256[]) (contracts/SavvySage.sol#180-215) has external calls inside a loop: baseToken = ITokenAdapter(params.adapter).baseToken() (contracts/SavvySage.sol#205-206)
SavvySage.refreshStrategies() (contracts/SavvySage.sol#364-388) has external calls inside a loop: params = yieldStrategyManager.getYieldTokenParameters(yieldToken) (contracts/SavvySage.sol#381-382)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#395)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#388-409) potentially used before declaration: retval == IERC721Receiver.onERC721Received.selector (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#396)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#397)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#388-409) potentially used before declaration: reason.length == 0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#398)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#397)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#388-409) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#402)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in SavvyLGE._buy(uint256,address,uint256,uint256) (contracts/SavvyLGE.sol#267-296):
        External calls:
        - depositToken.safeTransferFrom(msg.sender,depositTokenWallet,deposited) (contracts/SavvyLGE.sol#289)
        State variables written after the call(s):
        - allotmentSupply += allotments (contracts/SavvyLGE.sol#294)
        - totalDeposited += deposited (contracts/SavvyLGE.sol#295)
        - _updateUserBuyInfo(deposited,allotments,vestModeIndex) (contracts/SavvyLGE.sol#293)
                - userBuyInfo.duration = Math.findWeightedAverage(allotments,userBuyInfo.allotments,vestModes[vestModeIndex].duration,userBuyInfo.duration) (contracts/SavvyLGE.sol#342-347)
                - userBuyInfo.deposited += deposited (contracts/SavvyLGE.sol#348)
                - userBuyInfo.allotments += allotments (contracts/SavvyLGE.sol#349)
Reentrancy in SavvySage._swap(address) (contracts/SavvySage.sol#514-545):
        External calls:
        - _savvyAction(want,baseToken,_savvyWithdraw) (contracts/SavvySage.sol#530)
                - ISavvyPositionManager(savvy).syncAccount(address(this)) (contracts/SavvySage.sol#461)
        State variables written after the call(s):
        - currentSwapped[baseToken] += swapDelta (contracts/SavvySage.sol#542)
Reentrancy in BeefyVaultMock.deposit(uint256) (contracts/test/beefy/BeefyVaultMock.sol#47-71):
        External calls:
        - TokenUtils.safeTransferFrom(token,msg.sender,address(this),_amount) (contracts/test/beefy/BeefyVaultMock.sol#60)
        State variables written after the call(s):
        - _mint(msg.sender,_shares) (contracts/test/beefy/BeefyVaultMock.sol#69)
                - _balances[account] += amount (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#263)
        - _mint(msg.sender,_shares) (contracts/test/beefy/BeefyVaultMock.sol#69)
                - _totalSupply += amount (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#262)
Reentrancy in SavvyLGE.setProtocolToken(address) (contracts/SavvyLGE.sol#147-152):
        External calls:
        - withdrawProtocolToken() (contracts/SavvyLGE.sol#150)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#93)
                - protocolToken.safeTransfer(depositTokenWallet,currentBalance) (contracts/SavvyLGE.sol#355)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        External calls sending eth:
        - withdrawProtocolToken() (contracts/SavvyLGE.sol#150)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        State variables written after the call(s):
        - _setProtocolToken(_protocolToken) (contracts/SavvyLGE.sol#151)
                - totalProtocolToken = currentBalance (contracts/SavvyLGE.sol#366)
Reentrancy in SavvySage.setSavvy(address) (contracts/SavvySage.sol#232-251):
        External calls:
        - TokenUtils.safeApprove(debtToken,savvy,0) (contracts/SavvySage.sol#240)
        State variables written after the call(s):
        - yieldStrategyManager = ISavvyPositionManager(savvy).yieldStrategyManager() (contracts/SavvySage.sol#244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in SavvyPositionManager._borrowCredit(address,uint256,address) (contracts/SavvyPositionManager.sol#1184-1218):
        External calls:
        - _preemptivelyHarvestDeposited(owner) (contracts/SavvyPositionManager.sol#1199)
                - _yieldStrategyManager.preemptivelyHarvest(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#779)
        - _distributeUnlockedCreditDeposited(owner) (contracts/SavvyPositionManager.sol#1203)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        - _syncAccount(owner) (contracts/SavvyPositionManager.sol#1207)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - _updateDebt(owner,SafeCast.toInt256(amount)) (contracts/SavvyPositionManager.sol#1208)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - _yieldStrategyManager.decreaseBorrowingLimiter(amount) (contracts/SavvyPositionManager.sol#1212)
        - TokenUtils.safeMint(debtToken,recipient,amount) (contracts/SavvyPositionManager.sol#1215)
        Event emitted after the call(s):
        - Borrow(owner,amount,recipient) (contracts/SavvyPositionManager.sol#1217)
Reentrancy in SavvyLGE._buy(uint256,address,uint256,uint256) (contracts/SavvyLGE.sol#267-296):
        External calls:
        - depositToken.safeTransferFrom(msg.sender,depositTokenWallet,deposited) (contracts/SavvyLGE.sol#289)
        Event emitted after the call(s):
        - AllotmentsBought(msg.sender,deposited,allotments) (contracts/SavvyLGE.sol#291)
Reentrancy in SavvyPositionManager._depositYieldToken(address,uint256,address) (contracts/SavvyPositionManager.sol#805-834):
        External calls:
        - yieldTokenParams = _yieldStrategyManager.depositPrepare(yieldToken) (contracts/SavvyPositionManager.sol#812)
        - _distributeUnlockedCreditDeposited(recipient) (contracts/SavvyPositionManager.sol#815)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        - _syncAccount(recipient,yieldToken) (contracts/SavvyPositionManager.sol#819)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - shares = _issueSharesForAmount(recipient,yieldToken,amount) (contracts/SavvyPositionManager.sol#820)
                - shares = _yieldStrategyManager.issueSharesForAmount(yieldToken,amount) (contracts/SavvyPositionManager.sol#1060)
        - yieldTokenParams = _yieldStrategyManager.syncYieldToken(yieldToken,amount,true) (contracts/SavvyPositionManager.sol#823)
        Event emitted after the call(s):
        - DepositYieldToken(msg.sender,yieldToken,amount,recipient) (contracts/SavvyPositionManager.sol#831)
Reentrancy in SavvyPositionManager._withdraw(address,address,uint256,address) (contracts/SavvyPositionManager.sol#847-876):
        External calls:
        - _preemptivelyHarvestDeposited(owner) (contracts/SavvyPositionManager.sol#856)
                - _yieldStrategyManager.preemptivelyHarvest(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#779)
        - _distributeUnlockedCreditDeposited(owner) (contracts/SavvyPositionManager.sol#860)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        - _syncAccount(owner) (contracts/SavvyPositionManager.sol#866)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - _burnShares(owner,yieldToken,shares) (contracts/SavvyPositionManager.sol#867)
                - _yieldStrategyManager.burnShares(yieldToken,shares) (contracts/SavvyPositionManager.sol#1075)
        - _yieldStrategyManager.syncYieldToken(yieldToken,amountYieldTokens,false) (contracts/SavvyPositionManager.sol#868)
        Event emitted after the call(s):
        - WithdrawYieldToken(owner,yieldToken,shares,recipient) (contracts/SavvyPositionManager.sol#873)
Reentrancy in SavvyLGE._withdrawProtocolToken() (contracts/SavvyLGE.sol#353-357):
        External calls:
        - protocolToken.safeTransfer(depositTokenWallet,currentBalance) (contracts/SavvyLGE.sol#355)
        Event emitted after the call(s):
        - ProtocolTokenWithdrawn(address(protocolToken),currentBalance) (contracts/SavvyLGE.sol#356)
Reentrancy in SavvyPositionManager.addBaseToken(address,ISavvyAdminActions.BaseTokenConfig) (contracts/SavvyPositionManager.sol#236-242):
        External calls:
        - _yieldStrategyManager.addBaseToken(debtToken,baseToken_,config) (contracts/SavvyPositionManager.sol#239)
        Event emitted after the call(s):
        - AddBaseToken(baseToken_) (contracts/SavvyPositionManager.sol#241)
```

```
Reentrancy in SavvyBooster.addPool(uint256,uint256) (contracts/SavvyBooster.sol#120-150):
        External calls:
        - svyToken.safeTransferFrom(msg.sender,address(this),amount_) (contracts/SavvyBooster.sol#147)
        Event emitted after the call(s):
        - Deposit(amount_) (contracts/SavvyBooster.sol#149)
Reentrancy in SavvyPositionManager.addYieldToken(address,ISavvyAdminActions.YieldTokenConfig) (contracts/SavvyPositionManager.sol#245-259):
        External calls:
        - _yieldStrategyManager.addYieldToken(yieldToken,config) (contracts/SavvyPositionManager.sol#248)
        - TokenUtils.safeApprove(yieldToken,config.adapter,type()(uint256).max) (contracts/SavvyPositionManager.sol#250)
        - TokenUtils.safeApprove(ITokenAdapter(config.adapter).baseToken(),config.adapter,type()(uint256).max) (contracts/SavvyPositionManager.sol#251)
        - TokenUtils.safeApprove(yieldToken,address(_yieldStrategyManager),type()(uint256).max) (contracts/SavvyPositionManager.sol#253)
        - TokenUtils.safeApprove(ITokenAdapter(config.adapter).baseToken(),address(_yieldStrategyManager),type()(uint256).max) (contracts/SavvyPositionManager.sol#254)
        Event emitted after the call(s):
        - AddYieldToken(yieldToken) (contracts/SavvyPositionManager.sol#256)
        - MaximumLossUpdated(yieldToken,config.maximumLoss) (contracts/SavvyPositionManager.sol#258)
        - TokenAdapterUpdated(yieldToken,config.adapter) (contracts/SavvyPositionManager.sol#257)
Reentrancy in SavvyLGE.claim() (contracts/SavvyLGE.sol#225-233):
        External calls:
        - protocolToken.safeTransfer(msg.sender,owed) (contracts/SavvyLGE.sol#230)
        Event emitted after the call(s):
        - ProtocolTokensClaimed(msg.sender,owed) (contracts/SavvyLGE.sol#232)
Reentrancy in SavvySwap.claim(uint256,address) (contracts/SavvySwap.sol#232-246):
        External calls:
        - TokenUtils.safeBurn(syntheticToken,debtAmount) (contracts/SavvySwap.sol#243)
        - ISavvySage(buffer).withdraw(baseToken,amount,recipient) (contracts/SavvySwap.sol#244)
        Event emitted after the call(s):
        - Claim(msg.sender,recipient,amount) (contracts/SavvySwap.sol#245)
Reentrancy in SavvyBooster.claimSvyRewards() (contracts/SavvyBooster.sol#223-253):
        External calls:
        - svyToken.safeTransfer(account,rewardAmount) (contracts/SavvyBooster.sol#248)
        Event emitted after the call(s):
        - ClaimAccount_rewardAmount.pendingRewards) (contracts/SavvyBooster.sol#250)
Reentrancy in SavvyPositionManager.configureBorrowingLimit(uint256,uint256) (contracts/SavvyPositionManager.sol#321-325):
        External calls:
        - _yieldStrategyManager.configureBorrowingLimit(maximum,blocks) (contracts/SavvyPositionManager.sol#323)
        Event emitted after the call(s):
        - BorrowingLimitUpdated(maximum,blocks) (contracts/SavvyPositionManager.sol#324)
Reentrancy in SavvyPositionManager.configureCreditUnlockRate(address,uint256) (contracts/SavvyPositionManager.sol#328-332):
        External calls:
        - _yieldStrategyManager.configureCreditUnlockRate(yieldToken,blocks) (contracts/SavvyPositionManager.sol#330)
        Event emitted after the call(s):
        - CreditUnlockRateUpdated(yieldToken,blocks) (contracts/SavvyPositionManager.sol#331)
Reentrancy in SavvyPositionManager.configureRepayLimit(address,uint256,uint256) (contracts/SavvyPositionManager.sol#276-280):
        External calls:
        - _yieldStrategyManager.configureRepayLimit(baseToken_,maximum,blocks) (contracts/SavvyPositionManager.sol#278)
        Event emitted after the call(s):
        - RepayLimitUpdated(baseToken_,maximum,blocks) (contracts/SavvyPositionManager.sol#279)
Reentrancy in SavvyPositionManager.configureRepayWithCollateralLimit(address,uint256,uint256) (contracts/SavvyPositionManager.sol#283-287):
        External calls:
        - _yieldStrategyManager.configureRepayWithCollateralLimit(baseToken_,maximum,blocks) (contracts/SavvyPositionManager.sol#285)
        Event emitted after the call(s):
        - RepayWithCollateralLimitUpdated(baseToken_,maximum,blocks) (contracts/SavvyPositionManager.sol#286)
Reentrancy in SavvySwap.deposit(uint256,address) (contracts/SavvySwap.sol#204-215):
        External calls:
        - TokenUtils.safeTransferFrom(syntheticToken,msg.sender,address(this),amount) (contracts/SavvySwap.sol#213)
        Event emitted after the call(s):
        - Deposit(msg.sender,owner,amount) (contracts/SavvySwap.sol#214)
Reentrancy in BeefyVaultMock.deposit(uint256) (contracts/test/beefy/BeefyVaultMock.sol#47-71):
        External calls:
        - TokenUtils.safeTransferFrom(token,msg.sender,address(this),_amount) (contracts/test/beefy/BeefyVaultMock.sol#60)
        Event emitted after the call(s):
        - Transfer(address(0),account,amount) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#264)
                - _mint(msg.sender,_shares) (contracts/test/beefy/BeefyVaultMock.sol#69)
Reentrancy in SavvyPositionManager.depositBaseToken(address,uint256,address,uint256) (contracts/SavvyPositionManager.sol#415-431):
        External calls:
        - _onlyRedlisted() (contracts/SavvyPositionManager.sol#422)
                - msg.sender != wrapTokenGateway && (redlistActive && ! ISavvyRedlist(savvyRedlist).isRedlisted(msg.sender)) (contracts/SavvyPositionManager.sol#740-741)
        - amountYieldTokens = _wrap(yieldToken,amount,minimumAmountOut) (contracts/SavvyPositionManager.sol#427)
                - (success,data) = token.call(abi.encodeWithSelector(IERC20Minimal.transferFrom.selector,owner,recipient,amount)) (contracts/libraries/TokenUtils.sol#102-104)
                - TokenUtils.safeTransferFrom(baseToken_,msg.sender,address(this),amount) (contracts/SavvyPositionManager.sol#905)
                - wrappedShares = adapter.wrap(amount,address(this)) (contracts/SavvyPositionManager.sol#906)
        - _depositYieldToken(yieldToken,amountYieldTokens,recipient) (contracts/SavvyPositionManager.sol#430)
                - yieldTokenParams = _yieldStrategyManager.depositPrepare(yieldToken) (contracts/SavvyPositionManager.sol#812)
                - shares = _yieldStrategyManager.issueSharesForAmount(yieldToken,amount) (contracts/SavvyPositionManager.sol#1060)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
                - yieldTokenParams = _yieldStrategyManager.syncYieldToken(yieldToken,amount,true) (contracts/SavvyPositionManager.sol#823)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        Event emitted after the call(s):
        - DepositYieldToken(msg.sender,yieldToken,amount,recipient) (contracts/SavvyPositionManager.sol#831)
                - _depositYieldToken(yieldToken,amountYieldTokens,recipient) (contracts/SavvyPositionManager.sol#430)
Reentrancy in SavvyPositionManager.depositYieldToken(address,uint256,address) (contracts/SavvyPositionManager.sol#395-412):
        External calls:
        - _onlyRedlisted() (contracts/SavvyPositionManager.sol#401)
                - msg.sender != wrapTokenGateway && (redlistActive && ! ISavvyRedlist(savvyRedlist).isRedlisted(msg.sender)) (contracts/SavvyPositionManager.sol#740-741)
        - shares = _depositYieldToken(yieldToken,amount,recipient) (contracts/SavvyPositionManager.sol#406)
                - yieldTokenParams = _yieldStrategyManager.depositPrepare(yieldToken) (contracts/SavvyPositionManager.sol#812)
                - shares = _yieldStrategyManager.issueSharesForAmount(yieldToken,amount) (contracts/SavvyPositionManager.sol#1060)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
                - yieldTokenParams = _yieldStrategyManager.syncYieldToken(yieldToken,amount,true) (contracts/SavvyPositionManager.sol#823)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        Event emitted after the call(s):
        - DepositYieldToken(msg.sender,yieldToken,amount,recipient) (contracts/SavvyPositionManager.sol#831)
                - shares = _depositYieldToken(yieldToken,amount,recipient) (contracts/SavvyPositionManager.sol#406)
Reentrancy in SavvyPositionManager.donate(address,uint256) (contracts/SavvyPositionManager.sol#654-678):
        External calls:
        - _yieldStrategyManager.distributeUnlockedCredit(yieldToken) (contracts/SavvyPositionManager.sol#659)
        - _syncAccount(msg.sender) (contracts/SavvyPositionManager.sol#662)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - _accounts[msg.sender].lastAccruedWeights[yieldToken] = _yieldStrategyManager.donate(yieldToken,amount,shares) (contracts/SavvyPositionManager.sol#668)
        - TokenUtils.safeBurnFrom(debtToken,msg.sender,amount) (contracts/SavvyPositionManager.sol#671)
        - _yieldStrategyManager.increaseBorrowingLimiter(amount) (contracts/SavvyPositionManager.sol#675)
        Event emitted after the call(s):
        - Donate(msg.sender,yieldToken,amount) (contracts/SavvyPositionManager.sol#677)
Reentrancy in SavvySyntheticToken.flashLoan(IERC3156FlashBorrower,address,uint256,bytes) (contracts/SavvySyntheticToken.sol#214-236):
        External calls:
        - Checker.checkState(receiver.onFlashLoan(msg.sender,token,amount,fee,data) == CALLBACK_SUCCESS,flash loan failed) (contracts/SavvySyntheticToken.sol#228-231)
        Event emitted after the call(s):
        - Transfer(account,address(0),amount) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#292)
                - _burn(address(receiver),amount + fee) (contracts/SavvySyntheticToken.sol#233)
Reentrancy in SavvyPositionManager.harvest(address,uint256) (contracts/SavvyPositionManager.sol#681-704):
        External calls:
        - (baseToken_,amountBaseTokens,feeAmount,distributeAmount,credit) = _yieldStrategyManager.harvest(yieldToken,minimumAmountOut,protocolFee) (contracts/SavvyPositionManager.sol#684-694)
        - TokenUtils.safeTransfer(baseToken_,protocolFeeReceiver,feeAmount) (contracts/SavvyPositionManager.sol#697)
        - TokenUtils.safeTransfer(baseToken_,savvySwap,distributeAmount) (contracts/SavvyPositionManager.sol#698)
        - IERC20TokenReceiver(savvySwap).onERC20Received(baseToken_,distributeAmount) (contracts/SavvyPositionManager.sol#701)
        Event emitted after the call(s):
        - Harvest(yieldToken,minimumAmountOut,amountBaseTokens,credit) (contracts/SavvyPositionManager.sol#703)
Reentrancy in SavvyPositionManager.initialize(ISavvyAdminActions.InitializationParams) (contracts/SavvyPositionManager.sol#158-190):
        External calls:
        - _yieldStrategyManager.setBorrowingLimiter(Limiters.createLinearGrowthLimiter(params.borrowingLimitMaximum,params.borrowingLimitBlocks,params.borrowingLimitMinimum)) (contracts/SavvyPositionManager.sol#176-182)
        Event emitted after the call(s):
        - AdminUpdated(admin) (contracts/SavvyPositionManager.sol#184)
        - BorrowingLimitUpdated(params.borrowingLimitMaximum,params.borrowingLimitBlocks) (contracts/SavvyPositionManager.sol#189)
        - MinimumCollateralizationUpdated(minimumCollateralization) (contracts/SavvyPositionManager.sol#186)
        - ProtocolFeeReceiverUpdated(protocolFeeReceiver) (contracts/SavvyPositionManager.sol#188)
        - ProtocolFeeUpdated(protocolFee) (contracts/SavvyPositionManager.sol#187)
        - SavvySwapUpdated(savvySwap) (contracts/SavvyPositionManager.sol#185)
Reentrancy in SavvySage.registerToken(address,address) (contracts/SavvySage.sol#266-283):
        External calls:
        - TokenUtils.safeApprove(baseToken,savvy,type()(uint256).max) (contracts/SavvySage.sol#281)
        Event emitted after the call(s):
        - RegisterToken(baseToken,_savvySwap) (contracts/SavvySage.sol#282)
Reentrancy in SavvyBooster.removePool(uint256) (contracts/SavvyBooster.sol#153-175):
        External calls:
        - svyToken.safeTransfer(msg.sender,poolToRemove.remainingEmissions) (contracts/SavvyBooster.sol#172)
        Event emitted after the call(s):
        - Withdraw(poolToRemove.remainingEmissions) (contracts/SavvyBooster.sol#174)
Reentrancy in SavvyPositionManager.repayWithBaseToken(address,uint256,address) (contracts/SavvyPositionManager.sol#554-597):
        External calls:
        - _distributeUnlockedCreditDeposited(recipient) (contracts/SavvyPositionManager.sol#567)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        - _syncAccount(recipient) (contracts/SavvyPositionManager.sol#570)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - (credit,actualAmount) = _yieldStrategyManager.repayWithBaseToken(baseToken_,amount,debt) (contracts/SavvyPositionManager.sol#580-583)
        - _updateDebt(recipient,- SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#586)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - TokenUtils.safeTransferFrom(baseToken_,msg.sender,savvySwap,actualAmount) (contracts/SavvyPositionManager.sol#589)
        - IERC20TokenReceiver(savvySwap).onERC20Received(baseToken_,actualAmount) (contracts/SavvyPositionManager.sol#592)
        Event emitted after the call(s):
        - RepayWithBaseToken(msg.sender,baseToken_,actualAmount,recipient,credit) (contracts/SavvyPositionManager.sol#594)
```

```
Reentrancy in SavvyPositionManager.repayWithCollateral(address,uint256,uint256) (contracts/SavvyPositionManager.sol#600-651):
        External calls:
        - TokenUtils.safeApprove(yieldToken,address(_yieldStrategyManager),type()(uint256).max) (contracts/SavvyPositionManager.sol#619)
        - (amountBaseTokens,amountYieldTokens,actualShares) = _yieldStrategyManager.repayWithCollateral(yieldToken,address(this),shares,minimumAmountOut,unrealizedDebt) (contracts/SavvyPositionManager.sol#620-624)
        - _distributeUnlockedCreditDeposited(msg.sender) (contracts/SavvyPositionManager.sol#627)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        - _syncAccount(msg.sender,yieldToken) (contracts/SavvyPositionManager.sol#633)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - _burnShares(msg.sender,yieldToken,actualShares) (contracts/SavvyPositionManager.sol#634)
                - _yieldStrategyManager.burnShares(yieldToken,shares) (contracts/SavvyPositionManager.sol#1075)
        - _updateDebt(msg.sender,- SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#635)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - _yieldStrategyManager.syncYieldToken(yieldToken,amountYieldTokens,false) (contracts/SavvyPositionManager.sol#637)
        - _yieldStrategyManager.decreaseRepayWithCollateralLimiter(baseToken_,amountBaseTokens) (contracts/SavvyPositionManager.sol#640)
        - TokenUtils.safeTransfer(baseToken_,savvySwap,amountBaseTokens) (contracts/SavvyPositionManager.sol#643)
        - IERC20TokenReceiver(savvySwap).onERC20Received(baseToken_,amountBaseTokens) (contracts/SavvyPositionManager.sol#646)
        Event emitted after the call(s):
        - RepayWithCollateral(msg.sender,yieldToken,baseToken_,actualShares,credit) (contracts/SavvyPositionManager.sol#648)
Reentrancy in SavvyPositionManager.repayWithDebtToken(uint256,address) (contracts/SavvyPositionManager.sol#510-551):
        External calls:
        - _distributeUnlockedCreditDeposited(recipient) (contracts/SavvyPositionManager.sol#517)
                - _yieldStrategyManager.distributeUnlockedCredit(depositedTokens.values[i]) (contracts/SavvyPositionManager.sol#884)
        - _syncAccount(recipient) (contracts/SavvyPositionManager.sol#520)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - _updateDebt(recipient,- SafeCast.toInt256(credit)) (contracts/SavvyPositionManager.sol#538)
                - ISavvyBooster(svyBooster).updatePendingRewardsWithDebt(user,debtAmount,totalDebtAmount) (contracts/SavvyPositionManager.sol#721)
        - TokenUtils.safeBurnFrom(debtToken,msg.sender,credit) (contracts/SavvyPositionManager.sol#541)
        - _yieldStrategyManager.increaseBorrowingLimiter(credit) (contracts/SavvyPositionManager.sol#545)
        Event emitted after the call(s):
        - RepayWithDebtToken(msg.sender,credit,recipient) (contracts/SavvyPositionManager.sol#548)
Reentrancy in SavvyPositionManager.setBaseTokenEnabled(address,bool) (contracts/SavvyPositionManager.sol#262-266):
        External calls:
        - _yieldStrategyManager.setBaseTokenEnabled(baseToken_,enabled) (contracts/SavvyPositionManager.sol#264)
        Event emitted after the call(s):
        - BaseTokenEnabled(baseToken_,enabled) (contracts/SavvyPositionManager.sol#265)
Reentrancy in SavvySage.setFlowRate(address,uint256) (contracts/SavvySage.sol#286-295):
        External calls:
        - _swap(baseToken) (contracts/SavvySage.sol#291)
                - ISavvyPositionManager(savvy).syncAccount(address(this)) (contracts/SavvySage.sol#461)
                - ISavvySwap(savvySwap[baseToken]).swap(swapDelta) (contracts/SavvySage.sol#543)
        Event emitted after the call(s):
        - SetFlowRate(baseToken_,flowRate) (contracts/SavvySage.sol#294)
Reentrancy in SavvyPositionManager.setMaximumExpectedValue(address,uint256) (contracts/SavvyPositionManager.sol#342-346):
        External calls:
        - _yieldStrategyManager.setMaximumExpectedValue(yieldToken,value) (contracts/SavvyPositionManager.sol#344)
        Event emitted after the call(s):
        - MaximumExpectedValueUpdated(yieldToken,value) (contracts/SavvyPositionManager.sol#345)
Reentrancy in SavvyPositionManager.setMaximumLoss(address,uint256) (contracts/SavvyPositionManager.sol#349-353):
        External calls:
        - _yieldStrategyManager.setMaximumLoss(yieldToken,value) (contracts/SavvyPositionManager.sol#351)
        Event emitted after the call(s):
        - MaximumLossUpdated(yieldToken,value) (contracts/SavvyPositionManager.sol#352)
Reentrancy in SavvyLGE.setProtocolToken(address) (contracts/SavvyLGE.sol#147-152):
        External calls:
        - withdrawProtocolToken() (contracts/SavvyLGE.sol#150)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#93)
                - protocolToken.safeTransfer(depositTokenWallet,currentBalance) (contracts/SavvyLGE.sol#355)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        External calls sending eth:
        - withdrawProtocolToken() (contracts/SavvyLGE.sol#150)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        Event emitted after the call(s):
        - ProtocolTokenUpdated(_protocolToken,currentBalance) (contracts/SavvyLGE.sol#367)
                - _setProtocolToken(_protocolToken) (contracts/SavvyLGE.sol#151)
Reentrancy in SavvySage.setSavvy(address) (contracts/SavvySage.sol#232-251):
        External calls:
        - TokenUtils.safeApprove(debtToken,savvy,0) (contracts/SavvySage.sol#240)
        - TokenUtils.safeApprove(debtToken,savvy,type()(uint256).max) (contracts/SavvySage.sol#248)
        Event emitted after the call(s):
        - SetSavvy(savvy) (contracts/SavvySage.sol#250)
Reentrancy in SavvyPositionManager.setTokenAdapter(address,address) (contracts/SavvyPositionManager.sol#335-339):
        External calls:
        - _yieldStrategyManager.setTokenAdapter(yieldToken,adapter) (contracts/SavvyPositionManager.sol#337)
        Event emitted after the call(s):
        - TokenAdapterUpdated(yieldToken,adapter) (contracts/SavvyPositionManager.sol#338)
Reentrancy in SavvyPositionManager.setYieldTokenEnabled(address,bool) (contracts/SavvyPositionManager.sol#269-273):
        External calls:
        - _yieldStrategyManager.setYieldTokenEnabled(yieldToken,enabled) (contracts/SavvyPositionManager.sol#271)
        Event emitted after the call(s):
        - YieldTokenEnabled(yieldToken,enabled) (contracts/SavvyPositionManager.sol#272)
Reentrancy in SavvyPositionManager.snap(address) (contracts/SavvyPositionManager.sol#356-360):
        External calls:
        - expectedValue = _yieldStrategyManager.snap(yieldToken) (contracts/SavvyPositionManager.sol#358)
        Event emitted after the call(s):
        - Snap(yieldToken,expectedValue) (contracts/SavvyPositionManager.sol#359)
Reentrancy in VeSvy.stake(uint256) (contracts/VeSvy.sol#197-221):
        External calls:
        - _claim(sender) (contracts/VeSvy.sol#208)
                - savvyBooster.updatePendingRewardsWithVeSvy(_addr,userVeSvyBal,totalVeSvyBal) (contracts/VeSvy.sol#363)
        - svy.safeTransferFrom(sender,address(this),_amount) (contracts/VeSvy.sol#218)
        Event emitted after the call(s):
        - Staked(sender,_amount) (contracts/VeSvy.sol#220)
Reentrancy in SavvyPositionManager.sweepTokens(address,uint256) (contracts/SavvyPositionManager.sol#363-371):
        External calls:
        - TokenUtils.safeTransfer(rewardToken,admin,amount) (contracts/SavvyPositionManager.sol#368)
        Event emitted after the call(s):
        - SweepTokens(rewardToken,amount) (contracts/SavvyPositionManager.sol#370)
Reentrancy in VeSvy.unstake(uint256) (contracts/VeSvy.sol#315-335):
        External calls:
        - _updateFactor(sender) (contracts/VeSvy.sol#330)
                - savvyBooster.updatePendingRewardsWithVeSvy(_addr,userVeSvyBal,totalVeSvyBal) (contracts/VeSvy.sol#363)
        - svy.safeTransfer(sender,_amount) (contracts/VeSvy.sol#333)
        Event emitted after the call(s):
        - Unstaked(sender,_amount) (contracts/VeSvy.sol#334)
```

```
Reentrancy in SavvyBooster.withdraw() (contracts/SavvyBooster.sol#274-280):
        External calls:
        - savyToken.safeTransfer(msg.sender,amount) (contracts/SavvyBooster.sol#277)
        Event emitted after the call(s):
        - Withdraw(amount) (contracts/SavvyBooster.sol#278)
Reentrancy in SavvySwap.withdraw(uint256,address) (contracts/SavvySwap.sol#218-229):
        External calls:
        - TokenUtils.safeTransfer(syntheticToken,recipient,amount) (contracts/SavvySwap.sol#227)
        Event emitted after the call(s):
        - Withdraw(msg.sender,recipient,amount) (contracts/SavvySwap.sol#228)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

SavvyBooster.removePool(uint256) (contracts/SavvyBooster.sol#153-175) uses timestamp for comparisons
        Dangerous comparisons:
        - Checker.checkArgument(period_ > currentPoolIndex && period_ <= pools.length,period must be for a queued future pool only) (contracts/SavvyBooster.sol#158-161)
        - pools.length > 1 (contracts/SavvyBooster.sol#164)
        - i < pools.length - 1 (contracts/SavvyBooster.sol#165)
SavvyBooster.updatePendingRewardsWithVeSvy(address,uint256,uint256) (contracts/SavvyBooster.sol#178-198) uses timestamp for comparisons
        Dangerous comparisons:
        - i < pools.length (contracts/SavvyBooster.sol#186)
SavvyBooster.claimSvyRewards() (contracts/SavvyBooster.sol#223-253) uses timestamp for comparisons
        Dangerous comparisons:
        - Checker.checkState(claimableAmount > 0,no claimable rewards) (contracts/SavvyBooster.sol#237)
SavvyBooster._getClaimableRewards(address,uint256) (contracts/SavvyBooster.sol#296-313) uses timestamp for comparisons
        Dangerous comparisons:
        - i <= currentPoolIndex (contracts/SavvyBooster.sol#303)
SavvyBooster._getPoolIndex(uint256) (contracts/SavvyBooster.sol#320-332) uses timestamp for comparisons
        Dangerous comparisons:
        - Checker.checkState(poolLength > 0,no pool) (contracts/SavvyBooster.sol#324)
        - i >= 0 (contracts/SavvyBooster.sol#326)
        - timeValue_ > pools[i].startTime (contracts/SavvyBooster.sol#327)
SavvyBooster._getCollateralWeightRatio(address,uint256) (contracts/SavvyBooster.sol#339-346) uses timestamp for comparisons
        Dangerous comparisons:
        - totalCollateralWeight == 0 (contracts/SavvyBooster.sol#345)
SavvyBooster._updateTotalDebt(address,uint256,uint256) (contracts/SavvyBooster.sol#419-431) uses timestamp for comparisons
        Dangerous comparisons:
        - i < pools.length (contracts/SavvyBooster.sol#427)
SavvyLGE.buy(uint256,address,uint256,uint256) (contracts/SavvyLGE.sol#202-215) uses timestamp for comparisons
        Dangerous comparisons:
        - Checker.checkState(block.timestamp >= lgeStartTimestamp,lge has not begun) (contracts/SavvyLGE.sol#208)
SavvyLGE.claim() (contracts/SavvyLGE.sol#225-233) uses timestamp for comparisons
        Dangerous comparisons:
        - Checker.checkState(owed > 0,no claimable amount) (contracts/SavvyLGE.sol#227)
SavvyLGE.getClaimable(address) (contracts/SavvyLGE.sol#244-259) uses timestamp for comparisons
        Dangerous comparisons:
        - Checker.checkState(block.timestamp > vestStartTimestamp,vesting has not started) (contracts/SavvyLGE.sol#248)
        - claimed[userAddress] + owed > totalOwed (contracts/SavvyLGE.sol#255)
SavvySage.onERC20Received(address,uint256) (contracts/SavvySage.sol#298-321) uses timestamp for comparisons
        Dangerous comparisons:
        - localBalance < flowAvailable[baseToken] (contracts/SavvySage.sol#312)
VeSvy._claim(address) (contracts/VeSvy.sol#243-254) uses timestamp for comparisons
        Dangerous comparisons:
        - amount > 0 (contracts/VeSvy.sol#249)
VeSvy._claimable(address) (contracts/VeSvy.sol#284-310) uses timestamp for comparisons
        Dangerous comparisons:
        - (userVeSvyBalance + pending) > maxVeSvyCap (contracts/VeSvy.sol#303)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp


AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#186-189)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#388-409) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#401-403)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#201-221) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#213-216)
EnumerableSet.values(EnumerableSet.AddressSet) (node_modules/@openzeppelin/contracts/utils/structs/EnumerableSet.sol#274-283) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/structs/EnumerableSet.sol#278-280)
EnumerableSet.values(EnumerableSet.UintSet) (node_modules/@openzeppelin/contracts/utils/structs/EnumerableSet.sol#347-356) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/structs/EnumerableSet.sol#351-353)
console._sendLogPayload(bytes) (node_modules/hardhat/console.sol#7-14) uses assembly
        - INLINE ASM (node_modules/hardhat/console.sol#10-13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

InfoAggregator.initialize(address[],ISavvyInfoAggregatorStructs.SupportTokenInfo[],ISavvyPriceFeed,ISavvyToken,ISavvyBooster,IVeSvy) (contracts/InfoAggregator.sol#51-86) compares to a boolean constant:
        -Checker.checkArgument(savvyPositionManagers.contains(savvyPositionManager) == false,same SavvyPositionManager already exists) (contracts/InfoAggregator.sol#70-73)
InfoAggregator.addSavvyPositionManager(address[]) (contracts/InfoAggregator.sol#89-99) compares to a boolean constant:
        -Checker.checkArgument(savvyPositionManagers.contains(savvyPositionManager) == false,same SavvyPositionManager exists) (contracts/InfoAggregator.sol#96)
InfoAggregator.addSupportTokens(ISavvyInfoAggregatorStructs.SupportTokenInfo[]) (contracts/InfoAggregator.sol#102-110) compares to a boolean constant:
        -Checker.checkState(_checkSupportTokenExist(_supportTokens[i].yieldToken) != true,same token already exists) (contracts/InfoAggregator.sol#107)
InfoAggregator.getPoolDeposited(address,address) (contracts/InfoAggregator.sol#411-433) compares to a boolean constant:
        -yieldStrategyManager.isSupportedYieldToken(yieldToken) == true (contracts/InfoAggregator.sol#427)
InfoAggregator._getUserDepositedTokenPrice(address,address,address) (contracts/InfoAggregator.sol#616-637) compares to a boolean constant:
        -yieldStrategyManager.isSupportedYieldToken(yieldToken_) == false (contracts/InfoAggregator.sol#622)
SavvyBooster.addSavvyPositionManagers(ISavvyPositionManager[]) (contracts/SavvyBooster.sol#94-117) compares to a boolean constant:
        -Checker.checkArgument(savvyPositionManagers.contains(savvyPositionManager) == false,same SavvyPositionManager already exist) (contracts/SavvyBooster.sol#102-105)
SavvyBooster.onlySavvyPositionManager() (contracts/SavvyBooster.sol#77-83) compares to a boolean constant:
        -Checker.checkState(savvyPositionManagers.contains(msg.sender) == true,only savvyPositionManager can call this function) (contracts/SavvyBooster.sol#78-81)
SavvySwap.notPaused() (contracts/SavvySwap.sol#182-185) compares to a boolean constant:
        -Checker.checkState(isPaused == false,paused) (contracts/SavvySwap.sol#183)
SavvySyntheticToken.setAllowedMinter(address,bool) (contracts/SavvySyntheticToken.sol#126-133) compares to a boolean constant:
        -state == true (contracts/SavvySyntheticToken.sol#128)
SavvySyntheticToken.ensureUnpaused() (contracts/SavvySyntheticToken.sol#67-70) compares to a boolean constant:
        -Checker.checkState(paused[msg.sender] == false,sender is paused to mint) (contracts/SavvySyntheticToken.sol#68)
YieldYakAdapter._deposit(uint256,address) (contracts/adapters/yieldyak/YieldYakAdapter.sol#78-93) compares to a boolean constant:
        -isWAVAX == true (contracts/adapters/yieldyak/YieldYakAdapter.sol#82)
YieldYakAdapter._withdraw(uint256,address) (contracts/adapters/yieldyak/YieldYakAdapter.sol#95-107) compares to a boolean constant:
        -isWAVAX == true (contracts/adapters/yieldyak/YieldYakAdapter.sol#101)
YieldYakAdapter._getBaseTokenBalance(address) (contracts/adapters/yieldyak/YieldYakAdapter.sol#109-115) compares to a boolean constant:
        -isWAVAX == true (contracts/adapters/yieldyak/YieldYakAdapter.sol#110)
Math.uoperation(uint256,uint256,bool) (contracts/libraries/Math.sol#28-34) compares to a boolean constant:
        -addOperation == true (contracts/libraries/Math.sol#29)
Allowlist.add(address) (contracts/utils/Allowlist.sol#30-35) compares to a boolean constant:
        -Checker.checkState(disabled == false,add allowlist is disabled) (contracts/utils/Allowlist.sol#32)
Allowlist.remove(address) (contracts/utils/Allowlist.sol#38-43) compares to a boolean constant:
        -Checker.checkState(disabled == false,add allowlist is disabled) (contracts/utils/Allowlist.sol#40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
Different versions of Solidity are used:
        - Version used: ['>=0.4.22<0.9.0', '^0.8.0', '^0.8.1', '^0.8.17', '^0.8.2']
        - ^0.8.0 (node_modules/@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol#2)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/IAccessControlUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/Pausable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/structs/EnumerableSet.sol#4)
        - ^0.8.17 (contracts/InfoAggregator.sol#2)
        - ^0.8.17 (contracts/SavvyActionBatcher.sol#2)
        - ^0.8.17 (contracts/SavvyBooster.sol#2)
        - ^0.8.17 (contracts/SavvyLGE.sol#3)
        - ^0.8.17 (contracts/SavvyLoanMigrator.sol#2)
        - ^0.8.17 (contracts/SavvyPositionManager.sol#2)
        - ^0.8.17 (contracts/SavvyPriceFeed.sol#2)
        - ^0.8.17 (contracts/SavvyProtocolToken.sol#2)
        - ^0.8.17 (contracts/SavvyRedlist.sol#2)
        - ^0.8.17 (contracts/SavvySage.sol#2)
        - ^0.8.17 (contracts/SavvySwap.sol#2)
        - ^0.8.17 (contracts/SavvySyntheticToken.sol#2)
        - ^0.8.17 (contracts/VeERC20Upgradeable.sol#3)
        - ^0.8.17 (contracts/VeSvy.sol#2)
        - ^0.8.17 (contracts/WrapTokenGateway.sol#2)
        - ^0.8.17 (contracts/YieldStrategyManager.sol#2)
        - ^0.8.17 (contracts/adapters/beefy/BeefyTokenAdapter.sol#2)
        - ^0.8.17 (contracts/adapters/vesper/VesperTokenAdapter.sol#2)
        - ^0.8.17 (contracts/adapters/yieldyak/YieldYakAdapter.sol#2)
        - ^0.8.17 (contracts/base/ErrorMessages.sol#2)
        - ^0.8.17 (contracts/base/Errors.sol#2)
```

```
        - ^0.8.17 (contracts/base/Multicall.sol#2)
        - ^0.8.17 (contracts/base/Mutex.sol#2)
        - ^0.8.17 (contracts/interfaces/IAllowlist.sol#2)
        - ^0.8.17 (contracts/interfaces/IERC20Burnable.sol#2)
        - ^0.8.17 (contracts/interfaces/IERC20Metadata.sol#2)
        - ^0.8.17 (contracts/interfaces/IERC20Minimal.sol#2)
        - ^0.8.17 (contracts/interfaces/IERC20Mintable.sol#2)
        - ^0.8.17 (contracts/interfaces/IERC20TokenReceiver.sol#2)
        - ^0.8.17 (contracts/interfaces/IERC3156FlashBorrower.sol#2)
        - ^0.8.17 (contracts/interfaces/IERC3156FlashLender.sol#2)
        - ^0.8.17 (contracts/interfaces/IInfoAggregator.sol#2)
        - ^0.8.17 (contracts/interfaces/IMulticall.sol#2)
        - ^0.8.17 (contracts/interfaces/ISavvyActionBatcher.sol#2)
        - ^0.8.17 (contracts/interfaces/ISavvyBooster.sol#2)
        - ^0.8.17 (contracts/interfaces/ISavvyLoanMigrator.sol#2)
        - ^0.8.17 (contracts/interfaces/ISavvyPositionManager.sol#2)
        - ^0.8.17 (contracts/interfaces/ISavvyPriceFeed.sol#2)
        - ^0.8.17 (contracts/interfaces/ISavvyRedlist.sol#2)
        - ^0.8.17 (contracts/interfaces/ISavvyToken.sol#2)
        - ^0.8.17 (contracts/interfaces/ITokenAdapter.sol#2)
        - ^0.8.17 (contracts/interfaces/IVeERC20.sol#2)
        - ^0.8.17 (contracts/interfaces/IVeSvy.sol#2)
        - ^0.8.17 (contracts/interfaces/IWrapTokenGateway.sol#2)
        - ^0.8.17 (contracts/interfaces/IYieldStrategyManager.sol#2)
        - ^0.8.17 (contracts/interfaces/external/IMAVAX9.sol#2)
        - ^0.8.17 (contracts/interfaces/external/beefy/IBeefyStrategy.sol#2)
        - ^0.8.17 (contracts/interfaces/external/beefy/IBeefyVault.sol#2)
        - ^0.8.17 (contracts/interfaces/external/vesper/IVesperVault.sol#2)
        - ^0.8.17 (contracts/interfaces/external/yieldyak/IYieldYakVault.sol#2)
        - ^0.8.17 (contracts/interfaces/infoaggregator/ISavvyFrontend.sol#2)
        - ^0.8.17 (contracts/interfaces/infoaggregator/ISavvyInfoAggregatorStructs.sol#2)
        - ^0.8.17 (contracts/interfaces/infoaggregator/ISavvyOverview.sol#2)
        - ^0.8.17 (contracts/interfaces/infoaggregator/ISavvyPool.sol#2)
        - ^0.8.17 (contracts/interfaces/infoaggregator/ISavvyPositions.sol#2)
        - ^0.8.17 (contracts/interfaces/infoaggregator/ISavvyUserBalance.sol#2)
        - ^0.8.17 (contracts/interfaces/infoaggregator/ISavvyUserPortfolio.sol#2)
        - ^0.8.17 (contracts/interfaces/savvy/ISavvyActions.sol#2)
        - ^0.8.17 (contracts/interfaces/savvy/ISavvyAdminActions.sol#2)
        - ^0.8.17 (contracts/interfaces/savvy/ISavvyErrors.sol#2)
        - ^0.8.17 (contracts/interfaces/savvy/ISavvyEvents.sol#2)
        - ^0.8.17 (contracts/interfaces/savvy/ISavvyImmutables.sol#2)
        - ^0.8.17 (contracts/interfaces/savvy/ISavvyLGE.sol#2)
        - ^0.8.17 (contracts/interfaces/savvy/ISavvyState.sol#2)
        - ^0.8.17 (contracts/interfaces/savvy/ISavvyTokenParams.sol#2)
        - ^0.8.17 (contracts/interfaces/savvy/IYieldStrategyManagerActions.sol#2)
        - ^0.8.17 (contracts/interfaces/savvy/IYieldStrategyManagerStates.sol#2)
        - ^0.8.17 (contracts/interfaces/savvySwap/ISavvySage.sol#2)
        - ^0.8.17 (contracts/interfaces/savvySwap/ISavvySwap.sol#2)
        - ^0.8.17 (contracts/libraries/Checker.sol#2)
        - ^0.8.17 (contracts/libraries/FixedPointMath.sol#2)
        - ^0.8.17 (contracts/libraries/Limiters.sol#2)
        - ^0.8.17 (contracts/libraries/LiquidityMath.sol#2)
        - ^0.8.17 (contracts/libraries/Math.sol#2)
        - ^0.8.17 (contracts/libraries/SafeCast.sol#2)
        - ^0.8.17 (contracts/libraries/Sets.sol#2)
        - ^0.8.17 (contracts/libraries/Tick.sol#2)
        - ^0.8.17 (contracts/libraries/TokenUtils.sol#2)
        - ^0.8.17 (contracts/test/ERC20Mock.sol#2)
```

```
        - ^0.8.17 (contracts/test/ERC721Mock.sol#2)
        - ^0.8.17 (contracts/test/SavvySageMock.sol#2)
        - ^0.8.17 (contracts/test/TestERC20.sol#2)
        - ^0.8.17 (contracts/test/TestFlashBorrower.sol#2)
        - ^0.8.17 (contracts/test/TestSavvyToken.sol#2)
        - ^0.8.17 (contracts/test/TestVeSvy.sol#2)
        - ^0.8.17 (contracts/test/beefy/BeefyVaultMock.sol#2)
        - ^0.8.17 (contracts/utils/Allowlist.sol#2)
        - >=0.4.22<0.9.0 (node_modules/hardhat/console.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

SavvySage.refreshStrategies() (contracts/SavvySage.sol#364-388) has costly operations inside a loop:
        - delete _yieldTokens[registeredBaseTokens[i]] (contracts/SavvySage.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Mutex._isLocked() (contracts/base/Mutex.sol#27-29) is never used and should be removed
SavvySage._savvyDeposit(address,uint256) (contracts/SavvySage.sol#483-490) is never used and should be removed
SavvySage._savvyDonate(address,uint256) (contracts/SavvySage.sol#475-477) is never used and should be removed
SavvySage._savvyWithdraw(address,uint256) (contracts/SavvySage.sol#496-505) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (contracts/interfaces/IERC20PermitAllowed.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version^0.8.17 (contracts/test/TestErc20Receiver.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version^0.8.17 (contracts/utils/UniswapIUniswapV2Router02.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Pragma version^0.8.17 (contracts/test/TestFlashBorrower.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/test/TestSavvyToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/test/TestVeSvy.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/test/beefy/BeefyVaultMock.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/utils/Allowlist.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version>=0.4.22<0.9.0 (node_modules/hardhat/console.sol#2) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#164)
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#157-166):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#184-193):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
Low level call in WrapTokenGateway.withdrawBaseToken(address,address,uint256,address,uint256) (contracts/WrapTokenGateway.sol#70-88):
        (success) = _recipient.call{value: amount}(new bytes(0)) (contracts/WrapTokenGateway.sol#86)
Low level call in Multicall.multicall(bytes[]) (contracts/base/Multicall.sol#12-23):
        - (success,result) = address(this).delegatecall(data[i]) (contracts/base/Multicall.sol#15)
Low level call in TokenUtils.expectDecimals(address) (contracts/libraries/TokenUtils.sol#27-37):
        - (success,data) = token.staticcall(abi.encodeWithSelector(IERC20Metadata.decimals.selector)) (contracts/libraries/TokenUtils.sol#28-30)
Low level call in TokenUtils.safeBalanceOf(address,address) (contracts/libraries/TokenUtils.sol#47-57):
        - (success,data) = token.staticcall(abi.encodeWithSelector(IERC20Minimal.balanceOf.selector,account)) (contracts/libraries/TokenUtils.sol#48-50)
Low level call in TokenUtils.safeTransfer(address,address,uint256) (contracts/libraries/TokenUtils.sol#66-74):
        - (success,data) = token.call(abi.encodeWithSelector(IERC20Minimal.transfer.selector,recipient,amount)) (contracts/libraries/TokenUtils.sol#67-69)
Low level call in TokenUtils.safeApprove(address,address,uint256) (contracts/libraries/TokenUtils.sol#83-91):
        - (success,data) = token.call(abi.encodeWithSelector(IERC20Minimal.approve.selector,spender,value)) (contracts/libraries/TokenUtils.sol#84-86)
Low level call in TokenUtils.safeTransferFrom(address,address,address,uint256) (contracts/libraries/TokenUtils.sol#101-109):
        - (success,data) = token.call(abi.encodeWithSelector(IERC20Minimal.transferFrom.selector,owner,recipient,amount)) (contracts/libraries/TokenUtils.sol#102-104)
Low level call in TokenUtils.safeMint(address,address,uint256) (contracts/libraries/TokenUtils.sol#118-126):
        - (success,data) = token.call(abi.encodeWithSelector(IERC20Mintable.mint.selector,recipient,amount)) (contracts/libraries/TokenUtils.sol#119-121)
Low level call in TokenUtils.safeBurn(address,uint256) (contracts/libraries/TokenUtils.sol#134-142):
        - (success,data) = token.call(abi.encodeWithSelector(IERC20Burnable.burn.selector,amount)) (contracts/libraries/TokenUtils.sol#135-137)
Low level call in TokenUtils.safeBurnFrom(address,address,uint256) (contracts/libraries/TokenUtils.sol#151-159):
        - (success,data) = token.call(abi.encodeWithSelector(IERC20Burnable.burnFrom.selector,owner,amount)) (contracts/libraries/TokenUtils.sol#152-154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

SavvyProtocolToken (contracts/SavvyProtocolToken.sol#12-46) should inherit from IERC20Mintable (contracts/interfaces/IERC20Mintable.sol#8-17)
SavvySyntheticToken (contracts/SavvySyntheticToken.sol#20-237) should inherit from IERC20Burnable (contracts/interfaces/IERC20Burnable.sol#8-24)
SavvySyntheticToken (contracts/SavvySyntheticToken.sol#20-237) should inherit from IERC20Mintable (contracts/interfaces/IERC20Mintable.sol#8-17)
ERC20Mock (contracts/test/ERC20Mock.sol#9-29) should inherit from IERC20Mintable (contracts/interfaces/IERC20Mintable.sol#8-17)
TestFlashBorrower (contracts/test/TestFlashBorrower.sol#8-32) should inherit from IERC3156FlashBorrower (contracts/interfaces/IERC3156FlashBorrower.sol#7-25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance

Function IUniswapV2Router01.WETH() (contracts/utils/Uniswap/IUniswapV2Router02.sol#7) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
Variable ISavvyRedlist.isRedlistNFT(address).nftCollection_ (contracts/interfaces/ISavvyRedlist.sol#44) is too similar to SavvyRedlist.nftCollections (contracts/SavvyRedlist.sol#31)
Variable ISavvyRedlist.addNFTCollection(address).nftCollection_ (contracts/interfaces/ISavvyRedlist.sol#50) is too similar to SavvyRedlist.nftCollections (contracts/SavvyRedlist.sol#31)
Variable SavvyRedlist.addNFTCollection(address).nftCollection_ (contracts/SavvyRedlist.sol#120) is too similar to SavvyRedlist.nftCollections (contracts/SavvyRedlist.sol#31)
Variable SavvyRedlist.removeNFTCollection(address).nftCollection_ (contracts/SavvyRedlist.sol#139) is too similar to SavvyRedlist.nftCollections (contracts/SavvyRedlist.sol#31)
Variable SavvyRedlist._isNFTOwner(address,address).nftCollection_ (contracts/SavvyRedlist.sol#192) is too similar to SavvyRedlist.nftCollections (contracts/SavvyRedlist.sol#31)
Variable ISavvyRedlist.removeNFTCollection(address).nftCollection_ (contracts/interfaces/ISavvyRedlist.sol#56) is too similar to SavvyRedlist.nftCollections (contracts/SavvyRedlist.sol#31)
Variable SavvyRedlist.isRedlistNFT(address).nftCollection_ (contracts/SavvyRedlist.sol#113) is too similar to SavvyRedlist.nftCollections (contracts/SavvyRedlist.sol#31)
Variable SavvySwap._updateAccount(SavvySwap.UpdateAccountParams).tick_scope_0 (contracts/SavvySwap.sol#477) is too similar to SavvySwap._updateAccount(SavvySwap.UpdateAccountParams).tick_scope_1 (contracts/SavvySwap.sol#487)
Variable YieldStrategyManager._borrowingLimiter (contracts/YieldStrategyManager.sol#35) is too similar to YieldStrategyManager.setBorrowingLimiter(Limiters.LinearGrowthLimiter).borrowingLimiter_ (contracts/YieldStrategyManager.sol#582)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

console.slitherConstructorConstantVariables() (node_modules/hardhat/console.sol#4-1532) uses literals with too many digits:
        - CONSOLE_ADDRESS = address(0x000000000000000000636F6e736F6c652e6c6f67) (node_modules/hardhat/console.sol#5)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#87) is never used in InfoAggregator (contracts/InfoAggregator.sol#25-1059)
OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#87) is never used in SavvyActionBatcher (contracts/SavvyActionBatcher.sol#13-44)
PausableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#102) is never used in SavvyBooster (contracts/SavvyBooster.sol#19-448)
OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#87) is never used in SavvyPriceFeed (contracts/SavvyPriceFeed.sol#14-94)
OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#87) is never used in SavvyRedlist (contracts/SavvyRedlist.sol#15-217)
AccessControlUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#247) is never used in SavvySage (contracts/SavvySage.sol#27-555)
AccessControlUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#247) is never used in SavvySwap (contracts/SavvySwap.sol#27-573)
VeERC20Upgradeable.__gap (contracts/VeERC20Upgradeable.sol#168) is never used in VeSvy (contracts/VeSvy.sol#34-366)
VeERC20Upgradeable._allowances (contracts/VeERC20Upgradeable.sol#15) is never used in VeSvy (contracts/VeSvy.sol#34-366)
VeSvy.ERC721_RECEIVED (contracts/VeSvy.sol#60) is never used in VeSvy (contracts/VeSvy.sol#34-366)
OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#87) is never used in WrapTokenGateway (contracts/WrapTokenGateway.sol#17-139)
OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#87) is never used in YieldStrategyManager (contracts/YieldStrategyManager.sol#16-913)
BeefyTokenAdapter.MAXIMUM_SLIPPAGE (contracts/adapters/beefy/BeefyTokenAdapter.sol#16) is never used in BeefyTokenAdapter (contracts/adapters/beefy/BeefyTokenAdapter.sol#15-85)
BeefyTokenAdapter.underlying_index (contracts/adapters/beefy/BeefyTokenAdapter.sol#22) is never used in BeefyTokenAdapter (contracts/adapters/beefy/BeefyTokenAdapter.sol#15-85)
VesperTokenAdapter.MAXIMUM_SLIPPAGE (contracts/adapters/vesper/VesperTokenAdapter.sol#17) is never used in VesperTokenAdapter (contracts/adapters/vesper/VesperTokenAdapter.sol#16-87)
YieldYakAdapter.MAXIMUM_SLIPPAGE (contracts/adapters/yieldyak/YieldYakAdapter.sol#18) is never used in YieldYakAdapter (contracts/adapters/yieldyak/YieldYakAdapter.sol#17-121)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

BeefyTokenAdapter.underlying_index (contracts/adapters/beefy/BeefyTokenAdapter.sol#22) should be constant
BeefyVaultMock.min (contracts/test/beefy/BeefyVaultMock.sol#12) should be constant
SavvyLGE.maxDuration (contracts/SavvyLGE.sol#46) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

```
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#59-61)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#67-70)
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#142-144)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#155-157)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#173-177)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
name() should be declared external:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-117)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#136-140)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#201-210)
balanceOf(address) should be declared external:
        - ERC721.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#62-65)
name() should be declared external:
        - ERC721.name() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#79-81)
symbol() should be declared external:
        - ERC721.symbol() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#86-88)
tokenURI(uint256) should be declared external:
        - ERC721.tokenURI(uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#93-98)
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#112-122)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#136-138)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#150-159)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#164-170)
initialize(address[],ISavvyInfoAggregatorStructs.SupportTokenInfo[],ISavvyPriceFeed,ISavvyToken,ISavvyBooster,IVeSvy) should be declared external:
        - InfoAggregator.initialize(address[],ISavvyInfoAggregatorStructs.SupportTokenInfo[],ISavvyPriceFeed,ISavvyToken,ISavvyBooster,IVeSvy) (contracts/InfoAggregator.sol#51-86)
getTotalDebtAmount() should be declared external:
        - InfoAggregator.getTotalDebtAmount() (contracts/InfoAggregator.sol#177-186)
getTotalDepositedAmount() should be declared external:
        - InfoAggregator.getTotalDepositedAmount() (contracts/InfoAggregator.sol#189-198)
initialize(IInfoAggregator) should be declared external:
        - SavvyActionBatcher.initialize(IInfoAggregator) (contracts/SavvyActionBatcher.sol#18-24)
initialize(IERC20,IERC20) should be declared external:
        - SavvyBooster.initialize(IERC20,IERC20) (contracts/SavvyBooster.sol#51-67)
getAllotmentsPerDepositToken(address,uint256,uint256) should be declared external:
        - SavvyLGE.getAllotmentsPerDepositToken(address,uint256,uint256) (contracts/SavvyLGE.sol#182-199)
```

THANK YOU FOR CHOOSING

// HALBORN