BlockSwap

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **November 10th, 2021 - December 7th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	11/11/2021	Roberto Reigada
0.2	Document Updates	12/07/2021	Roberto Reigada
0.3	Draft Review	12/07/2021	Gabi Urrutia
1.0	Remediation Plan	12/16/2021	Roberto Reigada
1.1	Remediation Plan Review	12/17/2021	Gabi Urrutia

CONTACTS

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

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

BlockSwap engaged Halborn to conduct a security audit on their smart contracts beginning on November 11th, 2021 and ending on December 7th, 2021. The security assessment was scoped to the smart contracts provided in the Github repository bsn-eng/Stakehouse-Halborn

1.2 AUDIT SUMMARY

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

The purpose of this audit is to:

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

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

1.3 TEST APPROACH & METHODOLOGY

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

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

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

RISK SCALE - LIKELIHOOD

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

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

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

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

1.4 SCOPE

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

- StakeHouseAccessControls.sol
- StakeHouseRegistry.sol
- StakeHouseUniverse.sol
- banking/Banking.sol
- banking/CollateralisedSlotManager.sol
- banking/SlotSettlementPool.sol
- banking/SlotToken.sol
- banking/dETH.sol
- banking/sETH.sol
- banking/savETH.sol
- banking/savETHReservePool.sol
- banking/savETHManager.sol
- brand/BrandCentral.sol
- brand/BrandNFT.sol
- brand/skLOOT.sol
- brand/skLOOTFactory.sol
- guards/ModuleGuards.sol
- helpers/FlagHelper.sol
- accounts/AccountManager.sol
- accounts/TransactionManager.sol
- accounts/Streamer.sol
- accounts/BalanceReporter.sol
- accounts/ETH2ReportValidator.sol
- accounts/ETH2ValidationLib.sol
- proxies/StakeHouseUpgradeableProxy.sol
- proxies/UniverseUpgradeableProxy.sol
- proxies/UpgradeableBeacon.sol

Commit ID: 18a4dac9a15f908f23746969caca7190aadc137f
Fixed Commit ID: be2b2fc9bfa3e8f42f936030c7a9d3eba42d5317

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	0	4	11

LIKELIHOOD

IMPACT

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

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - MAPPING IS NOT DECREASED AFTER A DEPOSIT	High	SOLVED - 12/09/2021
HAL02 - WEAK PRNG IN SKLOOTFACTORY CONTRACT	Low	SOLVED - 12/09/2021
HAL03 - LOST ADJUSTED DEPOSIT AMOUNT DIFFERENCE	Low	SOLVED - 12/09/2021
HAL04 - FUNCTION NOT EXPOSED IN TRANSACTIONMANAGER CONTRACT	Low	SOLVED - 12/09/2021
HAL05 - LACK OF ZERO ADDRESS CHECK	Low	SOLVED - 12/09/2021
HAL06 - USE OF DEPRECATED SETUPROLE FUNCTION	Informational	ACKNOWLEDGED
HAL07 - ACCOUNTMANAGER.GETACCOUNT VIEW FUNCTION CAN BE REMOVED	Informational	ACKNOWLEDGED
HAL08 - TRANSACTIONMANAGER.GETWITHDRAWALADDRE VIEW FUNCTION CAN BE REMOVED	Informational	ACKNOWLEDGED
HAL09 - CONSTANT KECCAK VARIABLES ARE TREATED AS EXPRESSIONS, NOT CONSTANTS	Informational	SOLVED - 12/09/2021
HAL10 - USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS	Informational	SOLVED - 12/09/2021
HAL11 - NO NEED TO INITIALIZE UINT256 I VARIABLE TO 0	Informational	SOLVED - 12/09/2021
HAL12 - LONG REVERT STRINGS	Informational	ACKNOWLEDGED
HAL13 - INCORRECT MESSAGE IN REQUIRE STATEMENT	Informational	SOLVED - 12/09/2021
HAL14 - STATE VARIABLES MISSING CONSTANT MODIFIER	Informational	ACKNOWLEDGED
HAL15 - STATE VARIABLES MISSING IMMUTABLE MODIFIER	Informational	SOLVED - 12/09/2021
HAL16 - POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	SOLVED - 12/09/2021

FINDINGS & TECH DETAILS

3.1 (HAL-01) MAPPING IS NOT DECREASED AFTER A DEPOSIT - HIGH

Description:

In the contract BalanceReporter, the following function is defined:

Listing 1: BalanceReporter.sol (Lines 272)

```
252 function _addTopUpToQueue(
253     address _stakeHouse,
254     bytes calldata _blsPublicKey,
255     uint256 _amount
256 ) internal {
257     stakeHouseMemberQueue[_stakeHouse][_blsPublicKey] += _amount;
258     stakeHouseTotalDepositedForMembers[_stakeHouse][_blsPublicKey]
            += _amount;
259
260     if (stakeHouseMemberQueue[_stakeHouse][_blsPublicKey] >= 1
        ether) {
261         uint256 depositAmount = stakeHouseMemberQueue[_stakeHouse
            ][_blsPublicKey];
262
263         // Deposit amount sent to the deposit contract must be a
            multiple of 1 gwei so adjust deposit amount accordingly
264         depositAmount -= depositAmount % 1 gwei;
265
266         bytes memory _blsSignature = universe.accountManager().
            getSignatureByBLSKey(_blsPublicKey);
267
268         /// Deposit amount is divided by 1 gwei, because Deposit
            contract only tracks the balance up to 1 gwei precision
269         bytes32 leaf = ETH2Validation.getDepositDataRoot(
            _blsPublicKey, _blsSignature, WITHDRAWAL_CREDENTIALS,
            depositAmount / 1 gwei);
270
271         /// Send deposit topup to the contract
272         DepositContract.deposit{value: depositAmount}(
            _blsPublicKey, WITHDRAWAL_CREDENTIALS, _blsSignature,
            leaf);
273
```

```
274         emit FundsSentToDepositContract(_stakeHouse, _blsPublicKey
                , depositAmount);
275     }
276
277     emit ETHQueueDeposit(_stakeHouse, _blsPublicKey, _amount);
278 }
```

The function _addTopUpToQueue makes use of the following mapping: stakeHouseMemberQueue[_stakeHouse][_blsPublicKey] to calculate the amount of Ether that should be sent to the Deposit Contract:

Listing 2: BalanceReporter.sol

```
24 /// @notice StakeHouse -> Member ID (Validator pub key) -> ETH
      queued to be sent to the deposit contract
25 mapping(address => mapping(bytes => uint256)) public
      stakeHouseMemberQueue;
```

As we can see, after the call to DepositContract.deposit{value: depositAmount}(_blsPublicKey, WITHDRAWAL_CREDENTIALS, _blsSignature, leaf); the mapping amount is not decreased. This means that:

1. In the second call to _addTopUpToQueue the function will try to deposit a wrong amount of Ether into the Deposit Contract.
2. The contract will not have enough funds to be sent to the Deposit Contract causing any call to _addTopUpToQueue to revert. Hence, noone will be able to use the functions slashAndBuySlot and buySlashedSlot.

Risk Level:

Likelihood - 5
Impact - 4

Recommendation:

stakeHouseMemberQueue should be decreased after the DepositContract.deposit call by exactly the amount sent to the Deposit Contract.

Remediation Plan:

SOLVED: The BalanceReporter.sol contract now correctly decreases the stakeHouseMemberQueue mapping as suggested:

Listing 3: BalanceReporter.sol (Lines 274)

```
254 function _addTopUpToQueue(
255     address _stakeHouse,
256     bytes calldata _blsPublicKey,
257     uint256 _amount
258 ) internal {
259     stakeHouseMemberQueue[_stakeHouse][_blsPublicKey] += _amount;
260     stakeHouseTotalDepositedForMembers[_stakeHouse][_blsPublicKey]
            += _amount;
261
262     if (stakeHouseMemberQueue[_stakeHouse][_blsPublicKey] >= 1
        ether) {
263         uint256 depositAmount = stakeHouseMemberQueue[_stakeHouse
            ][_blsPublicKey];
264
265         // Deposit amount sent to the deposit contract must be a
            multiple of 1 gwei so adjust deposit amount accordingly
266         depositAmount -= depositAmount % 1 gwei;
267
268         bytes memory _blsSignature = universe.accountManager().
            getSignatureByBLSKey(_blsPublicKey);
269
270         /// Deposit amount is divided by 1 gwei, because Deposit
            contract only tracks the balance up to 1 gwei precision
271         bytes32 leaf = ETH2Validation.getDepositDataRoot(
            _blsPublicKey, _blsSignature, WITHDRAWAL_CREDENTIALS,
            depositAmount / 1 gwei);
272
273         // Adjust the member queue by the amount of value being
            sent to the deposit contract
274         stakeHouseMemberQueue[_stakeHouse][_blsPublicKey] -=
            depositAmount;
```

```
275
276         /// Send deposit topup to the contract
277         DepositContract.deposit{value: depositAmount}(
                _blsPublicKey, WITHDRAWAL_CREDENTIALS, _blsSignature,
                leaf);
278
279         emit FundsSentToDepositContract(_stakeHouse, _blsPublicKey
                , depositAmount);
280     }
281
282     emit ETHQueueDeposit(_stakeHouse, _blsPublicKey, _amount);
283 }
```

3.2 (HAL-02) WEAK PRNG IN SKLOOTFACTORY CONTRACT - LOW

Description:

In the contract skLootFactory, the following function is defined:

Listing 4: skLootFactory.sol (Lines 265)

```
253 function skLootItemClaim(
254     address _stakeHouse,
255     address _recipient,
256     uint256 _brandTokenId
257 ) external {
258     require(msg.sender == address(brandCentral), "Only brand
            central");
259
260     // Source of entropy
261     uint256 numberOfKnotsInHouse = StakeHouseRegistry(_stakeHouse)
            .numberOfMemberKNOTs();
262     uint256 numberOfHouseKnotInUniverse = brandCentral.universe().
            numberOfStakeHouses();
263     uint256 totalKnotsInUniverse = numberOfKnotsInHouse +
            numberOfHouseKnotInUniverse;
264
265     bool isSpecial = _blockNumber() % 50 == 0;
266
267     // Generate a pseudo random number using above and blockchain
            entropy
268     // in theory, miners dont manipulate basefee as that is burnt
            - EIP1559
269     uint256 pseudoRandomNumber = uint256(keccak256(abi.
            encodePacked(
270         block.difficulty,
271         block.timestamp,
272         block.basefee,
273         numberOfKnotsInHouse,
274         numberOfHouseKnotInUniverse,
275         isSpecial,
276         totalKnotsInUniverse
277     )));
278
```

```
279     // pluck an item depending on whether its special or knot :)
280     string memory pickedItem;
281     if (isSpecial) {
282         string[6] memory _specialLuckyDipGems =
                specialLuckyDipGems();
283         pickedItem = _specialLuckyDipGems[pseudoRandomNumber %
                _specialLuckyDipGems.length];
284     } else {
285         string[8] memory _luckyDipItems = luckyDipItems();
286         pickedItem = _luckyDipItems[pseudoRandomNumber %
                _luckyDipItems.length];
287     }
288
289     // mint the token
290     uint256 tokenId = skLoot.mint(pickedItem, skLOOT.ItemType.
            sItem, _brandTokenId, _recipient);
291
292     emit skLootItemClaimedForKnot(tokenId);
293 }
```

This function allows anyone to claim a skLoot item from the open pool when a new member is added to any StakeHouse. The item given is from a lucky dip list where special draws can be made from a rare gem list.

As we can see in case that `block.number % 50 == 0` the item will be a very rare gem. As it is true that it is not possible to force a transaction in a specific `block.number` users that are aware of this implementation will definitely try to do the call in a `block.number` multiple of 50 in order to acquire a gem.

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to increase the complexity of how `isSpecial` value is calculated and not make it dependant on just the `block.number`. The best

approach would be using Chainlink VRF to generate a random number and based on that number decide if the item given will be special or not.

Remediation Plan:

SOLVED: The BlockSwap team increased the complexity of how isSpecial value is calculated and it is not dependant only on just the block.number anymore. It is worth mentioning that still this is not totally random as the smart contract does not make use of ChainLink VRF.

3.3 (HAL-03) LOST ADJUSTED DEPOSIT AMOUNT DIFFERENCE - LOW

Description:

In the contract BalanceReporter the following function _addTopUpToQueue is defined:

Listing 5: BalanceReporter.sol (Lines 264)

```
252 function _addTopUpToQueue(
253     address _stakeHouse,
254     bytes calldata _blsPublicKey,
255     uint256 _amount
256 ) internal {
257     stakeHouseMemberQueue[_stakeHouse][_blsPublicKey] += _amount;
258     stakeHouseTotalDepositedForMembers[_stakeHouse][_blsPublicKey]
            += _amount;
259
260     if (stakeHouseMemberQueue[_stakeHouse][_blsPublicKey] >= 1
        ether) {
261         uint256 depositAmount = stakeHouseMemberQueue[_stakeHouse
            ][_blsPublicKey];
262
263         // Deposit amount sent to the deposit contract must be a
            multiple of 1 gwei so adjust deposit amount accordingly
264         depositAmount -= depositAmount % 1 gwei;
265
266         bytes memory _blsSignature = universe.accountManager().
            getSignatureByBLSKey(_blsPublicKey);
267
268         /// Deposit amount is divided by 1 gwei, because Deposit
            contract only tracks the balance up to 1 gwei precision
269         bytes32 leaf = ETH2Validation.getDepositDataRoot(
            _blsPublicKey, _blsSignature, WITHDRAWAL_CREDENTIALS,
            depositAmount / 1 gwei);
270
271         /// Send deposit topup to the contract
272         DepositContract.deposit{value: depositAmount}(
            _blsPublicKey, WITHDRAWAL_CREDENTIALS, _blsSignature,
            leaf);
273
```

```
274         emit FundsSentToDepositContract(_stakeHouse, _blsPublicKey
                , depositAmount);
275     }
276
277     emit ETHQueueDeposit(_stakeHouse, _blsPublicKey, _amount);
278 }
```

As we can see in the comments, the deposit amount sent to the Deposit Contract must be a multiple of 1 GWEI so the deposit amount is adjusted accordingly. Although, in this case, the difference from the msg.value sent by the user and the amount sent to the Deposit Contract will remain in the BalanceReporter/TransactionManager contract and will be lost by the user.

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended to send the remaining amount back to the user or directly not allowing the user to use a msg.value that is not multiple of 1 GWEI.

Remediation Plan:

SOLVED: The BlockSwap team added a require statement that checks now that the amount sent by the user is multiple of 1 GWEI.

3.4 (HAL-04) FUNCTION NOT EXPOSED IN TRANSACTIONMANAGER CONTRACT - LOW

Description:

In the contract AccountManager the following function is defined:

Listing 6: AccountManager.sol (Lines 203)

```
197 function rageQuitKnot(
198     address _rageQuitter,
199     bytes calldata _blsPublicKey,
200     address _stakeHouse,
201     uint256 _amountOfETHInDepositQueue,
202     ETH2Validation.ETH2DataReport calldata _report
203 ) external override onlyModule {
204     /// Perform the critical checks before exiting the stakehouse
205     _performStakeHousePreFlightChecks(_rageQuitter, _blsPublicKey)
        ;
206
207     /// Set user lifecycle status to exited
208     _setLifecycleStatus(_blsPublicKey, uint256(LifecycleStatus.
        EXITED));
209
210     blsPublicKeyToLastState[_blsPublicKey] = _report;
211
212     /// Initialize the rage quit of the Knot
213     universe.rageQuitKnot(
214         _stakeHouse,
215         _blsPublicKey,
216         _rageQuitter,
217         _amountOfETHInDepositQueue
218     );
219 }
```

The function contains the onlyModule modifier, which means that it can only be called by some other module although the call is not implemented anywhere:

```
root@halborn:~/halborn/projects/blockswap/contracts# grep -Rin "\.rageQuitKnot"
StakeHouseUniverse.sol:260:        slotSettlementPool.rageQuitKnotOnBehalfOf(
banking/SlotSettlementPool.sol:362:        universe.saveETHPool().rageQuitKnot(_stakeHouse, _memberId, _savETHKnotKeeper);
accounts/BalanceReporter.sol:238:        universe.slotSettlementPool().rageQuitKnotOnBehalfOf(
accounts/AccountManager.sol:213:        universe.rageQuitKnot(
root@halborn:~/halborn/projects/blockswap/contracts#
```

This does not occur with other functions in the AccountManager manager contract, as can be seen below:

```
root@halborn:~/halborn/projects/blockswap/contracts# grep -Rin "\.joinStakeHouseAndCreateBrand\|\.createStakehouse\|\.joinStakehouse"
accounts/TransactionManager.sol:86:        accountManager.createStakehouse(
accounts/TransactionManager.sol:101:        accountManager.joinStakehouse(
accounts/TransactionManager.sol:117:        accountManager.joinStakeHouseAndCreateBrand(
root@halborn:~/halborn/projects/blockswap/contracts#
```

Hence, the function rageQuitKnot should be exposed in the TransactionManager contract as it is done with the other functions.

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

It is recommended to expose the function AccountManager.rageQuitKnot in the TransactionManager contract as it is done with the rest of the functions of AccountManager.

Remediation Plan:

SOLVED: The BlockSwap team implemented the function in the TransactionManager contract:

Listing 7: TransactionManager.sol (Lines 134)

```
124 function rageQuit(
125     bytes calldata _blsPublicKey,
126     address _stakehouse,
127     ETH2Validation.ETH2DataReport calldata _eth2Report,
128     ETH2Validation.ETH2DataReportSignature calldata
            _reportSignature
129 ) external override onlyValidStakeHouse(_stakehouse) {
```

```
130     require(_isReportSignatureValid(_blsPublicKey, _eth2Report,
            _reportSignature), 'Report signature invalid');
131
132     _performETH2DataCheckRageQuit(_blsPublicKey, _eth2Report);
133
134     accountManager.rageQuitKnot(
135         msg.sender,
136         _blsPublicKey,
137         _stakehouse,
138         0, // As the KNOT was never active, there will be no funds
                in queue for top up if it is exiting
139         _eth2Report
140     );
141 }
```

3.5 (HAL-05) LACK OF ZERO ADDRESS CHECK - LOW

Description:

Checking addresses against zero-address during initialization is a security best-practice. However, such checks are missing in multiple constructors. Allowing zero-addresses can lead to contract reverts and force redeployment if there are no setters for such address variables.

Code Location:

savETHManager.sol

Listing 8: savETHManager.sol

```
14 function init(StakeHouseUniverse _universe) external initializer {
15     universe = _universe;
16 }
```

savETHReservePool.sol

Listing 9: savETHReservePool.sol

```
54 function init(StakeHouseUniverse _universe, address _saveETHLogic)
       external initializer {
55  dETHToken = new dETH(address(this));
56
57  StakeHouseUpgradeableProxy saveETHProxy = new
       StakeHouseUpgradeableProxy(
58      _saveETHLogic,
59      address(_universe),
60      abi.encodeWithSelector(
61          savETH(_saveETHLogic).init.selector,
62          address(this)
63      )
64  );
65
66  saveETHToken = savETH(address(saveETHProxy));
```

```
67
68  __initModuleGuards(_universe);
69 }
```

sETH.sol

Listing 10: sETH.sol

```
21 function init(SlotSettlementPool _slotSettlementPool, address
       _stakeHouse) external initializer {
22     slotSettlementPool = _slotSettlementPool;
23     stakeHouse = _stakeHouse;
24
25     __ERC20_init(
26         "sETH",
27         "sETH"
28     );
29
30     __ERC20Permit_init("sETH");
31 }
```

SlotSettlementPool.sol

Listing 11: SlotSettlementPool.sol

```
50 function init(
51     StakeHouseUniverse _universe,
52     address _sETHBeacon
53 ) external initializer {
54     __initModuleGuards(_universe);
55
56     slot = new SlotToken(address(this));
57
58     sETHBeacon = _sETHBeacon;
59 }
```

SlotToken.sol

Listing 12: SlotToken.sol

```
13 constructor(address _slotSettlementPool) {
14     slotSettlementPool = _slotSettlementPool;
15 }
```

BrandCentral.sol

Listing 13: BrandCentral.sol

```
54 function init(
55     StakeHouseUniverse _universe,
56     BrandNFT _brandNFT,
57     skLOOTFactory _skLootFactory,
58     BrandCentralClaimAuction _claimAuction
59 ) external initializer {
60     __initModuleGuards(_universe);
61
62     brandNFT = _brandNFT;
63     skLootFactory = _skLootFactory;
64     claimAuction = _claimAuction;
65 }
```

BrandCentralClaimAuction.sol

Listing 14: BrandCentralClaimAuction.sol

```
13 constructor(uint256 _startBlock, IERC20 _shbToken) {
14     isRestrictedBrandTicker["bsn"] = true;
15     isRestrictedBrandTicker["cbsn"] = true;
16     isRestrictedBrandTicker["dart"] = true;
17     isRestrictedBrandTicker["saver"] = true;
18     isRestrictedBrandTicker["stake"] = true;
19     isRestrictedBrandTicker["house"] = true;
20     isRestrictedBrandTicker["poly"] = true;
21     isRestrictedBrandTicker["wolf"] = true;
22     isRestrictedBrandTicker["elevt"] = true;
23     isRestrictedBrandTicker["mynt"] = true;
24     isRestrictedBrandTicker["club"] = true;
25     isRestrictedBrandTicker["impfi"] = true;
26     isRestrictedBrandTicker["colab"] = true;
```

```
27      isRestrictedBrandTicker["cland"] = true;
28
29      startBlock = _startBlock;
30
31      // auto calculate end block
32      endBlock = startBlock + TOTAL_AUCTION_LENGTH_IN_BLOCKS;
33
34      shbToken = _shbToken;
35
36      emit Deployed();
37  }
```

BrandNFT.sol

Listing 15: BrandNFT.sol

```
47  function init(address _brandCentral) external initializer {
48      brandCentral = BrandCentral(_brandCentral);
49
50      __ERC721_init("StakeHouseBrand", "SHNFT");
51  }
```

skLOOT.sol

Listing 16: skLOOT.sol

```
13  function init(skLOOTFactory _lootFactory) external initializer {
14      lootFactory = _lootFactory;
15
16      __ERC721_init("skLoot", "skLoot");
17  }
```

skLOOTFactory.sol

Listing 17: skLOOTFactory.sol

```
128 function init(BrandCentral _brandCentral, address _skLootLogic)
        external initializer {
129     brandCentral = _brandCentral;
130
131     __ERC721_init("skLootBag", "skLootBag");
```

```
132
133     StakeHouseUpgradeableProxy skLootProxy = new
            StakeHouseUpgradeableProxy(
134         _skLootLogic,
135         address(brandCentral.universe()),
136         abi.encodeWithSelector(
137             skLOOT(_skLootLogic).init.selector,
138             address(this)
139         )
140     );
141
142     skLoot = skLOOT(address(skLootProxy));
143 }
```

StakeHouseUpgradeableProxy.sol

Listing 18: StakeHouseUpgradeableProxy.sol

```
34 constructor(address _logic, address universe_, bytes memory _data)
       payable ERC1967Proxy(_logic, _data) {
35     _setUniverse(universe_);
36 }
```

UniverseUpgradeableProxy.sol

Listing 19: UniverseUpgradeableProxy.sol

```
34 constructor(address _logic, address accessControls_, bytes memory
      _data) payable ERC1967Proxy(_logic, _data) {
35     _setAccessControls(accessControls_);
36 }
```

StakeHouseAccessControls.sol

Listing 20: StakeHouseAccessControls.sol

```
37 constructor(address _superAdmin) {
38     _setRoleAdmin(CORE_MODULE_ADMIN_ROLE, CORE_MODULE_ADMIN_ROLE);
39     _setRoleAdmin(CORE_MODULE_MANAGER_ROLE, CORE_MODULE_ADMIN_ROLE
           );
40     _setRoleAdmin(CORE_MODULE_ROLE, CORE_MODULE_MANAGER_ROLE);
```

```
41
42      _setupRole(DEFAULT_ADMIN_ROLE, _superAdmin);
43      _setupRole(CORE_MODULE_ADMIN_ROLE, _superAdmin);
44      _setupRole(CORE_MODULE_MANAGER_ROLE, _superAdmin);
45  }
```

StakeHouseUniverse.sol

Listing 21: StakeHouseUniverse.sol

```
76  function init(
77      StakeHouseAccessControls _accessControls,
78      address _settlementPoolLogic,
79      address _sETHBeacon,
80      address _saveETHReservePoolLogic,
81      address _saveETHLogic,
82      address _stakeHouseRegistryBeacon,
83      address _accountManagerLogic,
84      address _transactionManagerLogic,
85      address _depositRouter,
86      uint256 _minDataEpochHeight
87  ) external initializer {
88      {
89          require(_accessControls.isAdmin(msg.sender), "Only admin")
                ;
90          require(_sETHBeacon != address(0), "sETH beacon cannot be
                zero address");
91          require(_stakeHouseRegistryBeacon != address(0), "Registry
                beacon cannot be zero address");
92      }
93
94      accessControls = _accessControls;
95
96      StakeHouseUpgradeableProxy accountManagerProxy = new
            StakeHouseUpgradeableProxy(
97          _accountManagerLogic,
98          address(this),
99          abi.encodeWithSelector(
100             AccountManager(_accountManagerLogic).init.selector,
101             address(this)
102         )
103     );
104
```

```
105     accountManager = AccountManager(address(accountManagerProxy));
106
107     StakeHouseUpgradeableProxy transactionManagerProxy = new
            StakeHouseUpgradeableProxy(
108         _transactionManagerLogic,
109         address(this),
110         abi.encodeWithSelector(
111             TransactionManager(_transactionManagerLogic).init.
                    selector,
112             address(this),
113             address(accountManagerProxy),
114             _depositRouter,
115             _minDataEpochHeight
116         )
117     );
118
119     transactionManager = TransactionManager(address(
            transactionManagerProxy));
120
121     StakeHouseUpgradeableProxy settlementProxy = new
            StakeHouseUpgradeableProxy(
122         _settlementPoolLogic,
123         address(this),
124         abi.encodeWithSelector(
125             SlotSettlementPool(_settlementPoolLogic).init.selector
                    ,
126             address(this),
127             _sETHBeacon
128         )
129     );
130
131     slotSettlementPool = SlotSettlementPool(address(
            settlementProxy));
132
133     StakeHouseUpgradeableProxy saveETHReservePoolProxy = new
            StakeHouseUpgradeableProxy(
134         _saveETHReservePoolLogic,
135         address(this),
136         abi.encodeWithSelector(
137             savETHReservePool(_saveETHReservePoolLogic).init.
                    selector,
138             address(this),
139             _saveETHLogic
140         )
```

```
141     );
142
143     saveETHPool = savETHReservePool(address(
            saveETHReservePoolProxy));
144
145     address _savETHManagerLogic = address(new savETHManager());
146
147     StakeHouseUpgradeableProxy saveETHManagerProxy = new
            StakeHouseUpgradeableProxy(
148         _savETHManagerLogic,
149         address(this),
150         abi.encodeWithSelector(
151             savETHManager(_savETHManagerLogic).init.selector,
152             address(this)
153         )
154     );
155
156     savETHMan = savETHManager(address(saveETHManagerProxy));
157
158     stakeHouseRegistryBeacon = _stakeHouseRegistryBeacon;
159
160     emit CoreModulesInit();
161 }
162
163 /// @dev Due to Solidity stack limitations on how many vars can be
        passed into a fn, this inits brand central separately
164 /// @param _brandCentralLogic Logic contract for Brand Central
165 /// @param _brandNftLogic Logic contract for the brand NFT
166 /// @param _lootFactoryLogic Logic contract for skLootFactory
167 /// @param _skLootLogic Logic contract or skLoot NFT
168 /// @param _claimAuction Address of the Brand Central SHB auction
       contract
169 function superchargeAndInitBrandCentral(
170     address _brandCentralLogic,
171     address _brandNftLogic,
172     address _lootFactoryLogic,
173     address _skLootLogic,
174     address _claimAuction
175 ) external {
176     require(accessControls.isAdmin(msg.sender), "Only admin");
177     require(address(brandCentral) == address(0), "Only init once")
            ;
178
179     StakeHouseUpgradeableProxy lootFactoryProxy = new
```

```
            StakeHouseUpgradeableProxy(
180             address(_lootFactoryLogic),
181             address(this),
182             abi.encodePacked("")
183         );
184
185         address _skLootFactory = address(lootFactoryProxy);
186
187         StakeHouseUpgradeableProxy bNFTProxy = new
                StakeHouseUpgradeableProxy(
188             address(_brandNftLogic),
189             address(this),
190             abi.encodePacked("")
191         );
192
193         address _brandNft = address(bNFTProxy);
194
195         StakeHouseUpgradeableProxy brandCentralProxy = new
                StakeHouseUpgradeableProxy(
196             _brandCentralLogic,
197             address(this),
198             abi.encodeWithSelector(
199                 BrandCentral(_brandCentralLogic).init.selector,
200                 address(this),
201                 _brandNft,
202                 _skLootFactory,
203                 _claimAuction
204             )
205         );
206
207         address brandCentralAddress = address(brandCentralProxy);
208         brandCentral = BrandCentral(brandCentralAddress);
209
210         // init proxies
211         BrandNFT(_brandNft).init(brandCentralAddress);
212         skLOOTFactory(_skLootFactory).init(brandCentral, _skLootLogic)
                ;
213
214         emit BrandCentralInit();
215     }
```

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

Add proper address validation when every state variable assignment is done from user supplied input.

Remediation Plan:

SOLVED: The BlockSwap team solved the issue by validating that every address input is different from zero.

3.6 (HAL-06) USE OF DEPRECATED SETUPROLE FUNCTION - INFORMATIONAL

Description:

Multiple contracts make use of the deprecated function `_setupRole` from the `AccessControl` contract. As per the `AccessControl.sol` contract documentation, this function is deprecated: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/AccessControl.sol#L183

Code Location:

- StakeHouseAccessControls.sol:42: `_setupRole(DEFAULT_ADMIN_ROLE, _superAdmin);`
- StakeHouseAccessControls.sol:43: `_setupRole(CORE_MODULE_ADMIN_ROLE, _superAdmin);`
- StakeHouseAccessControls.sol:44: `_setupRole(CORE_MODULE_MANAGER_ROLE, _superAdmin);`
- StakeHouseAccessControls.sol:115: `_setupRole(CORE_MODULE_ROLE, _address);`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use the `_grantRole` function instead.

Remediation Plan:

ACKNOWLEDGED: The BlockSwap team acknowledged this issue.

3.7 (HAL-07) ACCOUNTMANAGER.GETACCOUNT VIEW FUNCTION CAN BE REMOVED - INFORMATIONAL

Description:

In the contract AccountManager there is the following public state variable declared:

Listing 22: AccountManager.sol

```
27 Account[] public accounts;
```

At the same time, the contract contains the following view function:

Listing 23: AccountManager.sol

```
46 function getAccount(uint256 _index) external view returns(Account
       memory userAccount) {
47     require(_index < accounts.length, 'The index requested does
           not exist');
48
49     userAccount = accounts[_index];
50 }
```

As accounts is already declared as a public state variable the compiler already creates a view function to access and read each of the elements of the array, hence is not needed to declare an extra view function.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended either to declare accounts state variable as private keeping the new view function or to keep the accounts state variable as public and remove the new view function.

Remediation Plan:

ACKNOWLEDGED: The BlockSwap team acknowledged this issue.

3.8 (HAL-08) TRANSACTIONMANAGER.GETWITHDRAWALADDRESS VIEW FUNCTION CAN BE REMOVED - INFORMATIONAL

Description:

In the contract TransactionManager the following public state variable is declared:

Listing 24: TransactionManager.sol

```
15 AccountManager public accountManager;
```

At the same time, the contract contains the following view function:

Listing 25: TransactionManager.sol

```
123 function getWithdrawalAddress() external view returns (address) {
124     return address(accountManager);
125 }
```

As accountManager is already declared as a public state variable the compiler already creates a view function to read it, hence is not needed to declare an extra view function.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended either to declare accountManager state variable as private keeping the new view function or to keep the accountManager state

variable as public and remove the new view function.

Remediation Plan:

ACKNOWLEDGED: BlockSwap team acknowledged this issue.

3.9 (HAL-09) CONSTANT KECCAK VARIABLES ARE TREATED AS EXPRESSIONS, NOT CONSTANTS - INFORMATIONAL

Description:

In the contract StakeHouseAccessControls, the roles are declared the following way:

Listing 26: StakeHouseAccessControls.sol

```
11 bytes32 public constant PROXY_ADMIN_ROLE = keccak256("
      PROXY_ADMIN_ROLE");
12 bytes32 public constant CORE_MODULE_ADMIN_ROLE = keccak256("
      CORE_MODULE_ADMIN_ROLE");
13 bytes32 public constant CORE_MODULE_MANAGER_ROLE = keccak256("
      CORE_MODULE_MANAGER_ROLE");
14 bytes32 public constant CORE_MODULE_ROLE = keccak256("
      CORE_MODULE_ROLE");
```

This results in the keccak256 operation being performed whenever the variable is used, increasing gas costs relative to just storing the output hash.

- Each usage of a “constant” costs ~100gas more per access (still a little better than storing the result in storage, but not by much).
- Since these are not real constants, they can’t be referenced from a real constant environment (e.g., from assembly, or from another library).

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to either:
1. Keep the variables as constant and hard-code the bytes32 string into the smart contracts.
2. Declare all the roles as immutable and perform the hashing assignment in the constructors.

Remediation Plan:

SOLVED: The BlockSwap team solved the issue by adding the immutable modifier to the state variables mentioned, and they are now initialized in the constructor.

3.10 (HAL-10) USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS - INFORMATIONAL

Description:

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

Code Location:

BrandCentralClaimAuction.sol
- Line 226: `for(uint256 i = 0; i < AUCTION_LENGTH_IN_DAYS; i++){`
- Line 253: `for (uint256 i = 0; i < bStr.length; i++){`

BrandCentral.sol
- Line 255: `for(uint256 i = 0; i < _recipients.length; i++){`

BrandNFT.sol
- Line 142: `for (uint256 i = 0; i < bStr.length; i++){`

SlotSettlementPool.sol
- Line 329: `for(uint256 i = 0; i < _collateralisedSlotOwners.length; i ++){`

Proof of Concept:

For example, based in the following test contract:

Listing 27: 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  }
```

```
>>> test_contract.postiincrement(1)
Transaction sent: 0xlecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 44
  test.postiincrement confirmed   Block: 13622335   Gas used: 21620 (0.32%)

<Transaction '0xlecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b'>
>>> 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: 0x98c04430526a59balcf947c114b62666a4417165947d31bf300cd6ae68328033
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 46
  test.postiincrement confirmed   Block: 13622337   Gas used: 22673 (0.34%)

<Transaction '0x98c04430526a59balcf947c114b62666a4417165947d31bf300cd6ae68328033'>
>>> test_contract.preiincrement(10)
Transaction sent: 0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20elde05
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 47
  test.preiincrement confirmed   Block: 13622338   Gas used: 22601 (0.34%)

<Transaction '0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20elde05'>
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

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

Remediation Plan:

SOLVED: The BlockSwap team solved the issue by using ++i to increment the value of the iterator in all the loops.

3.11 (HAL-11) NO NEED TO INITIALIZE UINT256 I VARIABLE TO 0 - INFORMATIONAL

Description:

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

Code Location:

BrandCentralClaimAuction.sol
Line 226: `for(uint256 i = 0; i < AUCTION_LENGTH_IN_DAYS; i++)`
Line 253: `for (uint256 i = 0; i < bStr.length; i++)`

BrandCentral.sol
Line 255: `for(uint256 i = 0; i < _recipients.length; i++)`

BrandNFT.sol
Line 142: `for (uint256 i = 0; i < bStr.length; i++)`

SlotSettlementPool.sol
Line 329: `for(uint256 i = 0; i < _collateralisedSlotOwners.length; i++)`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to not initialize `i` variable to 0 to save some gas. For example: `for(uint256 i; i < AUCTION_LENGTH_IN_DAYS; i++)`

Remediation Plan:

SOLVED: The BlockSwap team solved the issue by not initializing any uint256 variable to 0 anymore.

3.12 (HAL-12) LONG REVERT STRINGS - INFORMATIONAL

Description:

Shortening revert strings to fit in 32 bytes will decrease deployment time gas and will decrease runtime gas when the revert condition has been met.

Revert strings that are longer than 32 bytes require at least one additional `mstore`, along with additional overhead for computing memory offset, etc.

Code Location:

AccountManager.sol
Line 47: `require(_index < accounts.length, 'The index requested does not exist');`
Line 107: `require(_depositor == accounts[index].initials.depositor, 'Pre-registered depositor mismatch');`
Line 311: `require(_isFollowDistanceCovered(_blsPublicKey), 'Sufficient blocks must pass before streaming');`

BalanceReporter.sol
Line 50: `require(_eth2Report.effectiveBalance == (32 ether / 1 gwei), "No new rewards if effective balance is not 32 ether");`

ETH2ReportValidator.sol
Line 35: `require(_depositRouter != address(0), 'Deposit router can not be address 0');`

TransactionManager.sol
Line 142: `Effective balance not equal to 32 ETH`

CollateralisedSlotManager.sol
Line 48: `require(!isUserEnabledForKnotWithdrawal[_user][_memberId], "User already enabled for withdrawal");`

dETH.sol

Line 17: require(address(_reservePool)!= address(0), "Reserve pool cannot be zero address");

savETH.sol

Line 19: require(address(_reservePool)!= address(0), "Reserve pool cannot be zero address");

Line 56: require(err == MathError.NO_ERROR, "Failed to calc SaveETH amount to transfer");

SlotSettlementPool.sol

Line 141: require(stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] + _amount <= SLASHING_COLLATERAL,"Slashing cannot exceed amount of collateral in pool");

Line 167: require(stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] >= _amount,"Cannot buy more SLOT than has been slashed");

Line 318: require(_collateralisedSlotOwners.length > 0, "No collateralised owners specified");

Line 341: require(collateralisedKnotBalance >= (SLASHING_COLLATERAL - roundingErrorBuffer),"Not enough collateralised SLOT to rage quit");

Line 347: require(slotBalInWallet >= 4 ether, "Need the other 4 SLOT to rage quit the Knot");

BrandCentral.sol

Line 147: require(!isLootBagAvailableForKnot(_memberId), "Cannot move until skLootBag minted");

Line 148: require(skOpenLootClaimed[_memberId], "Cannot move until skLoot item minted");

Line 163: require(!hasKnotClaimedBrandNFT(_memberId), "Cannot claim a brand and a skLootBag");

Line 184: require(!skOpenLootClaimed[_memberId], "skLOOT already claimed from the open pool");

Line 201: require(!skOpenLootClaimed[_memberId], "skLOOT already claimed from the open pool");

Line 202: require(!communityNetNftClaimed[msg.sender], "skLoot claimed by community member");

BrandCentralClaimAuction.sol

Line 257: require(bStr[i] >= 0x61 && bStr[i] <= 0x7A, "Name can only contain the 26 letters of the roman alphabet");

BrandNFT.sol

Line 63: require(bytes(_ticker).length >= 3 && bytes(_ticker).length <= 5,"Name must be between 3 and 5 characters");

Line 146: require(bStr[i] >= 0x61 && bStr[i] <= 0x7A, "Name can only contain the 26 letters of the roman alphabet");

ModuleGuards.sol

Line 37: require(StakeHouseAccessControls(universe.accessControls()).isCoreModule(msg.sender),"Only core modules are allowed to call this function");

StakeHouseAccessControls.sol

Line 93: require(!isAdmin(_address), "Admin cannot also have proxy admin");

Line 94: require(!isCoreModule(_address), "Proxy admin cannot also be a core module");

Line 95: require(!isCoreModuleManager(_address), "Proxy admin cannot also be a core module manager");

Line 96: require(!isCoreModuleAdmin(_address), "Admin cannot also be a core module admin");

Line 110: require(isCoreModuleAdmin(msg.sender)|| isCoreModuleManager(msg.sender), "Only admin or core module manager");

Line 111: require(!isProxyAdmin(_address), "Proxy admin cannot also be core module");

Line 112: require(!isAdmin(_address), "Admin cannot also be a core module");

Line 113: require(!isCoreModuleManager(_address), "Core module manager cannot also be a core module");

Line 114: require(!isCoreModuleAdmin(_address), "Admin cannot also be a core module");

Line 138: require(!isProxyAdmin(_address), "Proxy admin cannot also be core module");

Line 139: require(!isAdmin(_address), "Admin cannot also be a core module");

Line 140: require(!isCoreModule(_address), "Core module manager cannot also be a core module");
Line 141: require(!isCoreModuleAdmin(_address), "Manager cannot also be a core module admin");
Line 156: require(!isProxyAdmin(_address), "Proxy admin cannot also be core module admin");
Line 157: require(!isAdmin(_address), "Admin cannot also be a core module admin");
Line 158: require(!isCoreModule(_address), "Core module admin cannot also be a core module");
Line 159: require(!isCoreModuleManager(_address), "Manager cannot also be a core module admin");

StakeHouseRegistry.sol
Line 41: require(universe.memberKnotToStakeHouse(_memberId)== address(this), "Member added to another StakeHouse");

StakeHouseUniverse.sol
Line 90: require(_sETHBeacon != address(0), "sETH beacon cannot be zero address");
Line 91: require(_stakeHouseRegistryBeacon != address(0), "Registry beacon cannot be zero address");
Line 329: require(_rageQuitter != address(0), "Rage quitter cannot be zero address");
Line 356: require(memberKnotToStakeHouse[_memberId] != address(0), "Member is not assigned to any StakeHouse");
Line 366: require(memberKnotToStakeHouse[_memberId] != address(0), "Member is not assigned to any StakeHouse");
Line 413: require(memberKnotToStakeHouse[_memberId] != address(0), "Member is not assigned to any StakeHouse");
Line 469: require(accessControls.isCoreModule(msg.sender),"Only core modules are allowed to call this function");

Recommendation:

It is recommended to shorten the revert strings to fit in 32 bytes.

Remediation Plan:

ACKNOWLEDGED: The BlockSwap team acknowledged this issue.

3.13 (HAL-13) INCORRECT MESSAGE IN REQUIRE STATEMENT - INFORMATIONAL

Description:

In the contract AccountManager a require string is displaying a wrong message which may lead to confusion.

Code Location:

AccountManager.sol

Listing 28: AccountManager.sol (Lines 64)

```
59 function registerValidatorInitials(
60     address _depositor, bytes calldata _blsPublicKey, bytes
          calldata _blsSignature
61 ) external override onlyModule {
62     require(_getLifeCycleStatus(_blsPublicKey) == uint256(
          LifecycleStatus.UNBEGUN), 'Lifecycle status not 0');
63     require(_blsPublicKey.length == 48, 'Invalid public key length
          ');
64     require(_blsSignature.length == 96, 'Signature length is not
          incorrect');
65     require(_depositor != address(0), "Depositor cannot be zero");
66
67     /// Update validator lifecycle status to INITIALS_REGISTERED
          (1)
68     _setLifecycleStatus(_blsPublicKey, uint256(LifecycleStatus.
          INITIALS_REGISTERED));
69
70     // Create account object
71     Account memory account;
72
73     /// Update the account initials
74     account.initials = ValidatorInitials(_depositor, _blsSignature
          );
75
76     /// Map the account BLS public key to the index in the account
           array
77     blsPubKeyToAccountArrayIndex[_blsPublicKey] = accounts.length;
```

```
78
79      /// Add account to the account array
80      accounts.push(account);
81
82      emit InitialsRegistered(_depositor, _blsPublicKey,
            _blsSignature);
83  }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to correct the require message to:
`require(_blsSignature.length == 96, 'Signature length is incorrect');`

Remediation Plan:

SOLVED: The BlockSwap team fixed the require statement message.

3.14 (HAL-14) STATE VARIABLES MISSING CONSTANT MODIFIER - INFORMATIONAL

Description:

State variables can be declared as constant or immutable. In both cases, the variables cannot be modified after the contract has been constructed. For constant variables, the value has to be fixed at compile-time, while for immutable, it can still be assigned at construction time. The following state variable is missing the constant modifier:

AccountManager.sol
- Line 24: `uint256 public BLOCK_DELTA = 2048;`

BalanceReporter.sol
- Line 19: `IDepositContract public DepositContract = IDepositContract(0x00000000219ab540356cBB839Cbe05303d7705Fa);`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to add the constant modifier to the state variable mentioned.

Remediation Plan:

ACKNOWLEDGED: The BlockSwap team acknowledged this issue.

3.15 (HAL-15) STATE VARIABLES MISSING IMMUTABLE MODIFIER - INFORMATIONAL

Description:

The immutable keyword was added to Solidity in 0.6.5. State variables can be marked immutable which causes them to be read-only, but only assignable in the constructor. The following state variables are missing the immutable modifier:

dETH.sol
- Line 13: address public reservePool;

SlotToken.sol
- Line 10: address public slotSettlementPool;

BrandCentralClaimAuction.sol
- Line 55: uint256 public startBlock;
- Line 58: uint256 public endBlock;
- Line 64: IERC20 public shbToken;

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to add the immutable modifier to the state variables mentioned.

Remediation Plan:

SOLVED: The BlockSwap team solved the issue by adding the immutable modifier to all the state variables mentioned.

3.16 (HAL-16) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In the following contracts there are functions marked as public but they are never directly called within the same contract or in any of their descendants:

StakeHouseRegistry.sol
- getMemberInfoAtIndex() (StakeHouseRegistry.sol#99-107)

ETH2ValidationLib.sol
- getDepositDataRoot() (ETH2ValidationLib.sol#32-51)

BrandCentralClaimAuction.sol
- tokenURI() (BrandCentralClaimAuction.sol#214-216)

BrandNFT.sol
- tokenURI() (BrandNFT.sol#98-127)

skLOOT.sol
- tokenURI() (skLOOT.sol#77-105)

skLOOTFactory.sol
- tokenURI() (skLOOTFactory.sol#297-331)

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

If the functions are not intended to be called internally or by their descendants, it is better to mark all of these functions as external to

reduce gas costs.

Remediation Plan:

SOLVED: The BlockSwap team solved the issue by marking most of the functions mentioned as external.

MANUAL TESTING

4.1 CONTRACT INITIALIZATION

No issues found on the initialization:

Contract	Inherits from	Has initialize function?	Has initializer modifier?	Code
AccountManager	Initializable, Streamer, IAccountManager	Yes. The initialize function correctly initializes ModuleGuards from the Streamer contract. There are no parent contracts pending to be initialized	Yes	`function init(StakeHouseUniverse _universe!) external initializer { __initModuleGuards(_universe!); }`
BalanceReporter	ETH2ReportValidator	No	-	-
ETH2ReportValidator	EIP712Upgradeable, ModuleGuards	Yes. Although this function is internal. The function is initialized by TransactionManager contract as TransactionManager inherits from BalanceReporter which at the same time inherits from ETH2ReportValidator	Yes	`function __init_ETH2ReportValidator(address _depositRouter!, uint256 _minEpochHeight!, string memory _eip712ContractName!) internal initializer { __EIP712_init_unchained(_eip712ContractName!, "1"); _REPORT_TYPEHASH = keccak256("Report(bytes blsPubKey,bytes32 reportHash,uint256 deadline,uint256 blsPubKeyInternalNonce)"); require(_depositRouter! != address(0), 'Deposit router can not be address 0'); depositRouter = _depositRouter!; require(_minEpochHeight! > 0, 'Epoch height non-positive'); lastKnownEpochHeight = _minEpochHeight!; }`
ETH2Validation	-	No, (library)	-	-
Streamer	ModuleGuards	No, AccountManager inherits from Streamer and correctly initializes the ModuleGuards contract	-	-
TransactionManager	BalanceReporter, ITransactionManager	Yes, the initialize functions correctly initializes ModuleGuards and ETH2ReportValidator	Yes	`function init(StakeHouseUniverse _universe!, address _accountManager!, address _depositRouter!, uint256 _minDataEpoch!) external initializer { WITHDRAWAL_CREDENTIALS = ETH2Validation._computeWithdrawalCredentials(_accountManager!); require(WITHDRAWAL_CREDENTIALS.length == 32, 'Withdrawal credentials computed incorrectly'); require(_accountManager! != address(0), 'Account manager can not be address 0'); accountManager = AccountManager(_accountManager!); __initModuleGuards(_universe!); __init_ETH2ReportValidator(_depositRouter!, _minDataEpoch!, 'TransactionManager'); }`
Banking	-	No	-	-
CollateralisedSlotManager	-	No	-	-
dETH	ERC20, ERC20Permit	No	-	-
savETH	ERC20Upgradeable, ERC20PermitUpgradeable, Exponential	Yes, correctly initializes ERC20Upgradeable and ERC20PermitUpgradeable	Yes	`function init(savETHReservePool _reservePool!) external initializer { require(address(_reservePool!) != address(0), "Reserve pool cannot be zero address"); __ERC20_init("savETH", "savETH"); __ERC20Permit_init("savETH"); reservePool = _reservePool!; }`
savETHManager	Initializable, ISavETHManager	Yes. His parent contracts do not need to be initialized	Yes	`function init(StakeHouseUniverse _universe!) external initializer { universe = _universe!; }`
savETHReservePool	Initializable, ISavETHReservePool, Exponential, ModuleGuards	Yes, correctly initializes ModuleGuards. At the same time, it creates a new StakeHouseUpgradeable Proxy which implementation is a savETH contract, correctly initializing savETH contract as well	Yes	`function init(StakeHouseUniverse _universe!, address _saveETHLogic!) external initializer { dETHToken = new dETH(address(this)); StakeHouseUpgradeableProxy saveETHProxy = new StakeHouseUpgradeableProxy(_saveETHLogic!, address(_universe!), abi.encodeWithSelector(saveETH(_saveETHLogic!).init.selector, address(this))); saveETHToken = saveETH(address(saveETHProxy)); __initModuleGuards(_universe!); }`
sETH	ERC20Upgradeable, ERC20PermitUpgradeable, Exponential	Yes, correctly initializes ERC20Upgradeable and ERC20PermitUpgradeable	Yes	`function init(SlotSettlementPool _slotSettlementPool!, address _stakeHouse!) external initializer { slotSettlementPool = _slotSettlementPool!; stakeHouse = _stakeHouse!; __ERC20_init( "sETH", "sETH" ); __ERC20Permit_init("sETH"); }`
SlotSettlementPool	Initializable, ISlotSettlementPool, CollateralisedSlotManager, Exponential, ModuleGuards	Yes, correctly initializes ModuleGuards. The rest of his parent contracts do not need to be initialized	Yes	`function init( StakeHouseUniverse _universe!, address _sETHBeacon! ) external initializer { __initModuleGuards(_universe!); slot = new SlotToken(address(this)); sETHBeacon = _sETHBeacon!; }`

MANUAL TESTING

SlotToken	ERC20	-	-	-
BrandCentral	Initializable, IBrandCentral, ModuleGuards	Yes, correctly initializes ModuleGuards.The rest of his parent contracts do not need to be initialized	Yes	
BrandCentralClaimAuction	ERC721	No	-	-
BrandNFT	ERC721Upgradeable	Yes, correctly initializes ERC721Upgradeable	Yes	
skLOOT	ERC721Upgradeable	Yes, correctly initializes ERC721Upgradeable	Yes	
skLOOTFactory	ERC721Upgradeable	Yes, correctly initializes ERC721Upgradeable. At the same time, it creates a new StakeHouseUpgradeable Proxy which implementation is a skLOOT contract, correctly initializing skLOOT contract as well		
StakeHouseAccessControls	AccessControl	No	-	-
StakeHouseRegistry	Initializable, IMembershipRegistry, ModuleGuards	Yes, correctly initializes ModuleGuards.The rest of his parent contracts do not need to be initialized	Yes	
StakeHouseUniverse	Initializable, IStakeHouseUniverse, Banking, Pausable	Yes. No parent contract needs to be initialized. The contract contains two init functions: init() and superchargeAndInitBrand Central()	Yes, init() got the initializer modifier. superchargeAndInitBrand Central() does not have the initializer modifier but it contains a require check that forces it to only be called once	init(): Deploys multiple StakeHouseUpgradeableProxy. Each of those StakeHouseUpgradeableProxy deploy and initialize the following contracts: - AccountManager - TransactionManager - SlotSettlementPool - savETHReservePool - savETHManager superchargeAndInitBrandCentral(): Deploys multiple StakeHouseUpgradeableProxy. Each of those StakeHouseUpgradeableProxy: - Points a StakeHouseUpgradeableProxy to a skLootFactory and initializes the skLootFactory contract. - Points a StakeHouseUpgradeableProxy to a BrandNFT and initializes the BrandNFT contract. - Points a StakeHouseUpgradeableProxy to a BrandCentral and initializes the BrandCentral contract.

```
54      function init(
55          StakeHouseUniverse _universe!,
56          BrandNFT _brandNFT!,
57          skLOOTFactory _skLootFactory!,
58          BrandCentralClaimAuction _claimAuction!
59      ) external initializer {
60          __initModuleGuards(_universe!);
61
62          brandNFT = _brandNFT!;
63          skLootFactory = _skLootFactory!;
64          claimAuction = _claimAuction!;
65      }
```

```
47      function init(address _brandCentral!) external initializer {
48          brandCentral = BrandCentral(_brandCentral!);
49
50          __ERC721_init("StakeHouseBrand", "SHNFT");
51      }
```

```
39      function init(skLOOTFactory _lootFactory!) external initializer {
40          lootFactory = _lootFactory!;
41
42          __ERC721_init("skLoot", "skLoot");
43      }
```

```
128     function init(BrandCentral _brandCentral!, address _skLootLogic!) external initializer {
129         brandCentral = _brandCentral!;
130
131         __ERC721_init("skLootBag", "skLootBag");
132
133         StakeHouseUpgradeableProxy skLootProxy = new StakeHouseUpgradeableProxy(
134             _skLootLogic!,
135             address(brandCentral.universe()),
136             abi.encodeWithSelector(
137                 skLOOT(_skLootLogic!).init.selector,
138                 address(this)
139             )
140         );
141
142         skLoot = skLOOT(address(skLootProxy));
143     }
```

```
28      function init(StakeHouseUniverse _universe!) external initializer {
29          __initModuleGuards(_universe!);
30      }
```

AUTOMATED TESTING

5.1 STATIC ANALYSIS REPORT

Description:

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

Slither results:

StakeHouseAccessControls.sol

```
Different versions of Solidity is used:
        - Version used: ['^0.8.9', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3)
        - 0.8.9 (contracts/StakeHouseAccessControls.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#39-50) is never used and should be removed
Strings.toString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#14-34) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (contracts/StakeHouseAccessControls.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter StakeHouseAccessControls.isAdmin(address)._address (contracts/StakeHouseAccessControls.sol#48) is not in mixedCase
Parameter StakeHouseAccessControls.isProxyAdmin(address)._address (contracts/StakeHouseAccessControls.sol#53) is not in mixedCase
Parameter StakeHouseAccessControls.isCoreModule(address)._address (contracts/StakeHouseAccessControls.sol#58) is not in mixedCase
Parameter StakeHouseAccessControls.isCoreModuleManager(address)._address (contracts/StakeHouseAccessControls.sol#63) is not in mixedCase
Parameter StakeHouseAccessControls.isCoreModuleAdmin(address)._address (contracts/StakeHouseAccessControls.sol#68) is not in mixedCase
Parameter StakeHouseAccessControls.addAdmin(address)._address (contracts/StakeHouseAccessControls.sol#73) is not in mixedCase
Parameter StakeHouseAccessControls.removeAdmin(address)._address (contracts/StakeHouseAccessControls.sol#84) is not in mixedCase
Parameter StakeHouseAccessControls.addProxyAdmin(address)._address (contracts/StakeHouseAccessControls.sol#91) is not in mixedCase
Parameter StakeHouseAccessControls.removeProxyAdmin(address)._address (contracts/StakeHouseAccessControls.sol#102) is not in mixedCase
Parameter StakeHouseAccessControls.addCoreModule(address)._address (contracts/StakeHouseAccessControls.sol#109) is not in mixedCase
Parameter StakeHouseAccessControls.removeCoreModule(address)._address (contracts/StakeHouseAccessControls.sol#120) is not in mixedCase
Parameter StakeHouseAccessControls.lockCoreModule(address)._coreModule (contracts/StakeHouseAccessControls.sol#128) is not in mixedCase
Parameter StakeHouseAccessControls.addCoreModuleManager(address)._address (contracts/StakeHouseAccessControls.sol#136) is not in mixedCase
Parameter StakeHouseAccessControls.removeCoreModuleManager(address)._address (contracts/StakeHouseAccessControls.sol#147) is not in mixedCase
Parameter StakeHouseAccessControls.addCoreModuleAdmin(address)._address (contracts/StakeHouseAccessControls.sol#154) is not in mixedCase
Parameter StakeHouseAccessControls.removeCoreModuleAdmin(address)._address (contracts/StakeHouseAccessControls.sol#165) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#160-164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

StakeHouseRegistry.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#
283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
savETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReserv
ePool.sol#236)
savETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),saveETHRequiredForIsolation) (contracts/banking/
savETHReservePool.sol#266)
savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.
sol#158)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#190)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
```

AUTOMATED TESTING

```
BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isspecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,,,)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)
Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bNFTProxy = new StakeHouseUpgradeableProxy(address(_brandNftLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandNft,_skLootFactory,_claimAuction)) (contracts/StakeHo
useUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received
(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/c
ontracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzep
pelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgr
adeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (no
de_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modul
es/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.
sender,_blsPublicKey,_ticker,_buildingTypeId,_ethIReport) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),)
),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brand
TokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_bra
ndTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

StakeHouseUniverse.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#
283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

saveETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
saveETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReserv
ePool.sol#235)
saveETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),savETHRequiredForIsolation) (contracts/banking/
savETHReservePool.sol#246)
saveETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
saveETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.
sol#158)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
```

AUTOMATED TESTING

banking/Banking.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipOems[pseudoRandomNumber % _specialLuckyDipOems.length] (contracts/brand/skLOOTFactory.sol#283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

saveETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/saveETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/saveETHReservePool.sol#208)
saveETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/saveETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/saveETHReservePool.sol#235)
saveETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/saveETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),saveETHRequiredForIsolation) (contracts/banking/saveETHReservePool.sol#266)
saveETHReservePool.withdraw(address,uint256) (contracts/banking/saveETHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/saveETHReservePool.sol#301)
saveETHReservePool.deposit(address,uint256) (contracts/banking/saveETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/saveETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#159)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = (_blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in saveETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/saveETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/saveETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/saveETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(IStakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,)),,skLOOT.ItemType.sLand, _brandTokenId, _recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding, _brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)
Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - hBTTProxy = new StakeHouseUpgradeableProxy(address(_brandHTxLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandHTt,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender, blsPublicKey, _ticker, _buildingTypeId, _eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,)),,skLOOT.ItemType.sLand, _brandTokenId, _recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding, _brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter, _brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

banking/CollateralisedSlotManager.sol

```
CollateralisedSlotManager._enableUserForKnotWithdrawal(address,address,bytes) (contracts/banking/CollateralisedSlotManager.sol#47-54) is never used and should be removed
CollateralisedSlotManager._increaseCollateralisedBalance(address,address,bytes,uint256) (contracts/banking/CollateralisedSlotManager.sol#32-43) is never used and should be removed
CollateralisedSlotManager._markUserAsWithdrawn(address,bytes) (contracts/banking/CollateralisedSlotManager.sol#57-62) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.9 (contracts/banking/CollateralisedSlotManager.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

banking/SlotSettlementPool.sol

AUTOMATED TESTING

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#283)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
savETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#235)
savETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),savETHRequiredForIsolation) (contracts/banking/savETHReservePool.sol#266)
savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#158)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isspecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance[address(shareToken),_recipient,_memberId,sharesToMint] (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deploySakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(IStakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,))),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)
Reentrancy in StakeHouseUniverse.supercHargeAndInitBrandCentral(address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bNFTProxy = new StakeHouseUpgradeableProxy(address(_brandNftLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandNft,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender,_blsPublicKey,_ticker,_buildingTypeId,_eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,))),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

banking/SlotToken.sol

```
SlotToken.constructor(address)._slotSettlementPool (contracts/banking/SlotToken.sol#13) lacks a zero-check on :
        - slotSettlementPool = _slotSettlementPool (contracts/banking/SlotToken.sol#14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Different versions of Solidity is used:
        - Version used: ['0.8.9', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - 0.8.9 (contracts/banking/SlotToken.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (contracts/banking/SlotToken.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter SlotToken.mint(uint256)._amount (contracts/banking/SlotToken.sol#19) is not in mixedCase
Parameter SlotToken.burn(uint256)._amount (contracts/banking/SlotToken.sol#25) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

name() should be declared external:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#61-63)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#69-71)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#86-88)
totalSupply() should be declared external:
        - ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#93-95)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#100-102)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#112-115)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#120-122)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#131-134)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#149-163)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#177-180)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#196-204)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

banking/dETH.sol

```
ERC20Permit.constructor(string).name (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol#35) shadows:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#61-63) (function)
        - IERC20Metadata.name() (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#16) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Variable 'ECDSA.tryRecover(bytes32,bytes).r (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#59)' in ECDSA.tryRecover(bytes32,bytes) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#54-83) potentially
used before declaration: r = mload(uint256)(signature + 0x20) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#76)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

ERC20Permit.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol#40-59) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= deadline,ERC20Permit: expired deadline) (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

ECDSA.tryRecover(bytes32,bytes) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#54-83) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#64-68)
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#75-78)
ECDSA.tryRecover(bytes32,bytes32,bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#112-124) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#119-122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.9', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Counters.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol#3)
        - 0.8.9 (contracts/banking/dETH.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
Counters.decrement(Counters.Counter) (node_modules/@openzeppelin/contracts/utils/Counters.sol#31-37) is never used and should be removed
Counters.reset(Counters.Counter) (node_modules/@openzeppelin/contracts/utils/Counters.sol#39-41) is never used and should be removed
ECDSA.recover(bytes32,bytes) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#88-103) is never used and should be removed
ECDSA.recover(bytes32,bytes32,bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#131-139) is never used and should be removed
ECDSA.toEthSignedMessageHash(bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#201-205) is never used and should be removed
ECDSA.tryRecover(bytes32,bytes) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#54-83) is never used and should be removed
ECDSA.tryRecover(bytes32,bytes32,bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#112-124) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Counters.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (contracts/banking/dETH.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function ERC20Permit.DOMAIN_SEPARATOR() (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol#72-74) is not in mixedCase
Variable ERC20Permit._PERMIT_TYPEHASH (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol#27-28) is not in mixedCase
Function IERC20Permit.DOMAIN_SEPARATOR() (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#58) is not in mixedCase
Variable EIP712._CACHED_DOMAIN_SEPARATOR (node_modules/@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol#30) is not in mixedCase
Variable EIP712._CACHED_CHAIN_ID (node_modules/@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol#31) is not in mixedCase
Variable EIP712._HASHED_NAME (node_modules/@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol#33) is not in mixedCase
Variable EIP712._HASHED_VERSION (node_modules/@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol#34) is not in mixedCase
Variable EIP712._TYPE_HASH (node_modules/@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol#35) is not in mixedCase
Contract dETH (contracts/banking/dETH.sol#8-33) is not in CapWords
Parameter dETH.mint(uint256)._amount (contracts/banking/dETH.sol#23) is not in mixedCase
Parameter dETH.burn(uint256)._amount (contracts/banking/dETH.sol#29) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

ERC20Permit._PERMIT_TYPEHASH (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol#27-28) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

name() should be declared external:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#61-63)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#69-71)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#86-88)
totalSupply() should be declared external:
        - ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#93-95)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#100-102)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#112-115)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#120-122)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#131-134)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#149-163)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#177-180)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#196-204)
permit(address,address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - ERC20Permit.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol#40-59)
nonces(address) should be declared external:
        - ERC20Permit.nonces(address) (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol#64-66)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

banking/sETH.sol

AUTOMATED TESTING

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
savETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#235)
savETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),savETHRequiredForIsolation) (contracts/banking/savETHReservePool.sol#266)
savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#288-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#158)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
            - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
            - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
            - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,))),skLOOT.ItemType.sSland,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokensIdToComponentsRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse, memberId, _amount, isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
            - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse, memberId, _amount, _slasher) (contracts/banking/SlotSettlementPool.sol#131)
            - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse, memberId, _amount, _slasher) (contracts/banking/SlotSettlementPool.sol#131)
            - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse, memberId, _amount, _slasher) (contracts/banking/SlotSettlementPool.sol#131)
            - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)
Reentrancy in StakeHouseUniverse.supercharge_AndInitBrandCentral(address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bNFTProxy = new StakeHouseUpgradeableProxy(address(_brandNftLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandNft,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender,_blsPublicKey,_ticker,_buildingTypeId,_eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,))),skLOOT.ItemType.sSland,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

banking/savETH.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#283)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReserveFool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReserveFool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReserveFool.sol#208)
savETHReserveFool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReserveFool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReserveFool.sol#235)
savETHReserveFool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReserveFool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),savETHRequiredForIsolation) (contracts/banking/savETHReserveFool.sol#266)
savETHReserveFool.withdraw(address,uint256) (contracts/banking/savETHReserveFool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReserveFool.sol#301)
savETHReserveFool.deposit(address,uint256) (contracts/banking/savETHReserveFool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReserveFool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#158)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#181-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#106)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked(|,coords.x.toString(),, ,coords.y.toString(),,)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)
Reentrancy in StakeHouseUniverse.supercharge_AndInitBrandCentral(address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - hHTProxy = new StakeHouseUpgradeableProxy(address(_brandHtLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandHt,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender, blsPublicKey, ticker, buildingTypeId, eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked(|,coords.x.toString(),, ,coords.y.toString(),,)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

banking/savETHReservePool.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#256)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
savETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#235)
savETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),saveETHRequiredForIsolation) (contracts/banking/savETHReservePool.sol#266)
savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-325) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#153)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,))),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - stakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)
Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bNFTProxy = new StakeHouseUpgradeableProxy(address(_brandNftLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandNft,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#195-206)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender,_blsPublicKey,_ticker,_buildingTypeId,_eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,))),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

banking/savETHManager.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
savETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#235)
savETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),saveETHRequiredForIsolation) (contracts/banking/savETHReservePool.sol#266)
savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#159)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
            - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
            - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(IStakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
            - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,))),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
            - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
            - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
            - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
            - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)
Reentrancy in StakeHouseUniverse.supercharqeAndInitBrandCentral(address,address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bNFTProxy = new StakeHouseUpgradeableProxy(address(_brandNftLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandNft,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender,_blsPublicKey,_ticker,_buildingTypeId,_eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,))),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

brand/BrandCentral.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
savETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#235)
savETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),savETHRequiredForIsolation) (contracts/banking/savETHReservePool.sol#266)
savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#158)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,bytes,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#229-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,)))),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)
Reentrancy in StakeHouseUniverse.superchargedAndInitBrandCentral(address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bNFTProxy = new StakeHouseUpgradeableProxy(address(_brandMfiLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandNft,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender,_blsPublicKey,_ticker,_buildingTypeId,_eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,))),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

AUTOMATED TESTING

brand/BrandNFT.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#283)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
savETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#235)
savETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),savETHRequiredForIsolation) (contracts/banking/savETHReservePool.sol#266)
savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#288-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#158)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,))),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)
Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bNFTProxy = new StakeHouseUpgradeableProxy(address(_brandNftLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandNft,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender,_blsPublicKey,_ticker,_buildingTypeId,_eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,))),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

brand/skLOOT.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#283)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#410) shadows:
        - ERC165Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
savETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#238)
savETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),saveETHRequiredForIsolation) (contracts/banking/savETHReservePool.sol#266)
savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#158)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#190)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,))),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)
Reentrancy in StakeHouseUniverse.supercharge_AndInitBrandCentral(address,address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bNFTProxy = new StakeHouseUpgradeableProxy(address(_brandNftLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandNft,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#78-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#78-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender,_blsPublicKey,_ticker,_buildingTypeId,_eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,))),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

brand/skLOOTFactory.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#
283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

saveTHReservePool.addKnotTooOpenPool(address,bytes,address) (contracts/banking/saveTHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/saveTHReservePool.sol#208)
saveTHReservePool.addKnotTooOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/saveTHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/saveTHReserv
ePool.sol#235)
saveTHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/saveTHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),saveETHRequiredForIsolation) (contracts/banking/
saveTHReservePool.sol#266)
saveTHReservePool.withdraw(address,uint256) (contracts/banking/saveTHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/saveTHReservePool.sol#301)
saveTHReservePool.deposit(address,uint256) (contracts/banking/saveTHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/saveTHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.
sol#150)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = (_blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#136)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,,,)),skLOOT.ItemType.sLand, _brandTokenId, _recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding, _brandTokenId, _recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter, _brandTokenId, _recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)
Reentrancy in StakeHouseUniverse.supercargeAndInitBrandCentral(address,address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bNFTProxy = new StakeHouseUpgradeableProxy(address(_brandNftLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandNft,_skLootFactory,_claimAuction)) (contracts/StakeHo
useUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received
(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/c
ontracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzep
pelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgr
adeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (no
de_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modul
es/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.
sender,_blsPublicKey,_ticker,_buildingTypeId,_eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),)
)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brand
TokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_bra
ndTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

AUTOMATED TESTING

guards/ModuleGuards.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
savETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#235)
savETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),saveETHRequiredForIsolation) (contracts/banking/savETHReservePool.sol#266)
savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-326) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#159)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
            - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
            - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#210-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(IStakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
            - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,)),),skLOOT.ItemType.sLand, _brandTokenId, _recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding, _brandTokenId, _recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter, _brandTokenId, _recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse, _memberId, _amount, _isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
            - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
            - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
            - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
            - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)
Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - hHTProxy = new StakeHouseUpgradeableProxy(address(_brandHtsLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandHft,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETHIValidation.ETH2DataReport,ETHIValidation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender, blsPublicKey, ticker, buildingTypeId, ethIReport) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,)),),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

helpers/FlagHelper.sol

```
FlagHelper.exists(uint256) (contracts/helpers/FlagHelper.sol#18-20) is never used and should be removed
FlagHelper.getFlag(uint256,uint256) (contracts/helpers/FlagHelper.sol#46-49) is never used and should be removed
FlagHelper.hasRageQuit(uint256) (contracts/helpers/FlagHelper.sol#26-28) is never used and should be removed
FlagHelper.isKicked(uint256) (contracts/helpers/FlagHelper.sol#22-24) is never used and should be removed
FlagHelper.kickMember(uint256) (contracts/helpers/FlagHelper.sol#5-10) is never used and should be removed
FlagHelper.rageQuit(uint256) (contracts/helpers/FlagHelper.sol#12-14) is never used and should be removed
FlagHelper.setFlag(uint256,uint256,bool) (contracts/helpers/FlagHelper.sol#32-44) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.9 (contracts/helpers/FlagHelper.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

accounts/AccountManager.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
savETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#235)
savETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),saveETHRequiredForIsolation) (contracts/banking/savETHReservePool.sol#266)
savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#158)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer


BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReserveReserveMintETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#106)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#248)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked(I,coords.x.toString(),, ,coords.y.toString(),,)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)
Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bNFTProxy = new StakeHouseUpgradeableProxy(address(_brandMetaLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandNft,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender,_blsPublicKey,_ticker,_buildingTypeId,_eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

AUTOMATED TESTING

accounts/TransactionManager.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
savETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#235)
savETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),saveETHRequiredForIsolation) (contracts/banking/savETHReservePool.sol#266)
savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#150)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer


BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):

        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#106)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked(I,coords.x.toString(),, ,coords.y.toString(),,,,)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)

Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bNFTProxy = new StakeHouseUpgradeableProxy(address(_brandNftLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandNft,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender,_blsPublicKey,_ticker,_buildingTypeId,_eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

accounts/Streamer.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#
283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
savETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReserv
ePool.sol#235)
savETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),savETHRequiredForIsolation) (contracts/banking/
savETHReservePool.sol#266)
savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.
sol#150)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#106)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked(|,coords.x.toString(),, ,coords.y.toString(),,,,)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)
Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bHFTProxy = new StakeHouseUpgradeableProxy(address(_brandHftLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandHft,_skLootFactory,_claimAuction)) (contracts/StakeHo
useUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received
(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/c
ontracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzep
pelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgr
adeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (no
de_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modul
es/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-356)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.
sender,_blsPublicKey,_ticker,_buildingTypeId,_eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked(|,coords.x.toString(),, ,coords.y.toString(),)
)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brand
TokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_bra
ndTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

AUTOMATED TESTING

accounts/BalanceReporter.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
savETHReservePool.addKnotToOpenPoolAndWithdrawAllDETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#235)
savETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),saveETHRequiredForIsolation) (contracts/banking/savETHReservePool.sol#266)
savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#150)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#106)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked(I,coords.x.toString(),, ,coords.y.toString(),,,,)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)

Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bNFTProxy = new StakeHouseUpgradeableProxy(address(_brandNftLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandNft,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-387) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender,_blsPublicKey,_ticker,_buildingTypeId,_eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

accounts/ETH2ReportValidator.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#
283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

saveTHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/saveTHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/saveTHReservePool.sol#208)
saveTHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/saveTHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/saveTHReserv
ePool.sol#235)
saveTHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/saveTHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),saveTHRequiredForIsolation) (contracts/banking/
saveTHReservePool.sol#266)
saveTHReservePool.withdraw(address,uint256) (contracts/banking/saveTHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/saveTHReservePool.sol#301)
saveTHReservePool.deposit(address,uint256) (contracts/banking/saveTHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/saveTHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.
sol#150)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):

        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#106)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked(I,coords.x.toString(),, ,coords.y.toString(),,,,)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)

Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bNFTProxy = new StakeHouseUpgradeableProxy(address(_brandNftLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandNft,_skLootFactory,_claimAuction)) (contracts/StakeHo
useUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-387) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received
(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/c
ontracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzep
pelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgr
adeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (no
de_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modul
es/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.
sender,_blsPublicKey,_ticker,_buildingTypeId,_eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),)
)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brand
TokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_bra
ndTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

accounts/ETH2ValidationLib.sol

```
ETH2Validation._computeWithdrawalCredentials(address) (contracts/accounts/ETH2ValidationLib.sol#71-76) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.9 (contracts/accounts/ETH2ValidationLib.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter ETH2Validation.getDepositDataRoot(bytes,bytes,bytes,uint256)._blsPublicKey (contracts/accounts/ETH2ValidationLib.sol#33) is not in mixedCase
Parameter ETH2Validation.getDepositDataRoot(bytes,bytes,bytes,uint256)._signature (contracts/accounts/ETH2ValidationLib.sol#34) is not in mixedCase
Parameter ETH2Validation.getDepositDataRoot(bytes,bytes,bytes,uint256)._withdrawalCredentials (contracts/accounts/ETH2ValidationLib.sol#35) is not in mixedCase
Parameter ETH2Validation.getDepositDataRoot(bytes,bytes,bytes,uint256)._amount (contracts/accounts/ETH2ValidationLib.sol#36) is not in mixedCase
Function ETH2Validation._to_little_endian_64(uint64) (contracts/accounts/ETH2ValidationLib.sol#55-67) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

getDepositDataRoot(bytes,bytes,bytes,uint256) should be declared external:
        - ETH2Validation.getDepositDataRoot(bytes,bytes,bytes,uint256) (contracts/accounts/ETH2ValidationLib.sol#32-51)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

proxies/StakeHouseUpgradeableProxy.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
savETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#235)
savETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),saveETHRequiredForIsolation) (contracts/banking/savETHReservePool.sol#266)
savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#150)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer


BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):

        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#106)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked(I,coords.x.toString(),, ,coords.y.toString(),,,,)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)

Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bNFTProxy = new StakeHouseUpgradeableProxy(address(_brandNftLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandNft,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-387) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender,_blsPublicKey,_ticker,_buildingTypeId,_eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

AUTOMATED TESTING

proxies/UniverseUpgradeableProxy.sol

```
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#283)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 2,0x3d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256,uint256)(resultPtr_encode_asm_0 - 1,0x3d << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC20PermitUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#93) shadows:
        - EIP712Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#111)
        - ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#361)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#418) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

savETHReservePool.addKnotToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)
savETHReservePool.addKnotToOpenPoolAndWithdrawAlldETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#214-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#235)
savETHReservePool.isolateKnotFromOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),saveETHRequiredForIsolation) (contracts/banking/savETHReservePool.sol#266)
savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#285-304) ignores return value by dETHToken.transfer(_owner,dETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)
savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(this),_amount) (contracts/banking/savETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender,address(this),_shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#150)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer


BrandCentralClaimAuction.slitherConstructorConstantVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
        -BLOCKS_PER_DAY = 86400 / SECONDS_PER_BLOCK (contracts/brand/BrandCentralClaimAuction.sol#27)
        -TOTAL_AUCTION_LENGTH_IN_BLOCKS = BLOCKS_PER_DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
        -encodedLen = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
        - isSpecial = _blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
        External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        State variables written after the call(s):
        - auction.shbBid = _shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = _blockNumber() + BLOCKS_PER_DAY (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingEnd = auction.biddingEnd + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
        - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToMint) (contracts/banking/SlotSettlementPool.sol#187)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#64-84):
        External calls:
        - sETHProxy = new BeaconProxy(sETHBeacon,abi.encodeWithSelector(sETH(sETHBeacon).init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
        State variables written after the call(s):
        - stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in savETHReservePool.mintDETHReserves(address,bytes,uint256) (contracts/banking/savETHReservePool.sol#113-145):
        External calls:
        - saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (contracts/banking/savETHReservePool.sol#136)
        State variables written after the call(s):
        - dETHUnderManagement += _amount (contracts/banking/savETHReservePool.sol#142)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):

        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        State variables written after the call(s):
        - stakeHouseTotalSlot[_stakeHouse] += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#106)
Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        External calls:
        - universe.recordMemberSlotSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
        - slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - shareToken.mint(_recipient,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
        - shareToken.mint(address(this),sharesToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
        State variables written after the call(s):
        - _increaseCollateralisedBalance(address(shareToken),_recipient,_memberId,sharesToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
                - stakeHouseSETHTotalCollateralBalance[_sETHToken] += _amount (contracts/banking/CollateralisedSlotManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/StakeHouseUniverse.sol#218-265):
        External calls:
        - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
        - slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
        State variables written after the call(s):
        - _registerMemberToStakeHouse(stakeHouseAddress,_firstMember) (contracts/StakeHouseUniverse.sol#248)
                - memberKnotToStakeHouse[_memberId] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
        External calls:
        - skLoot.mint(string(abi.encodePacked(I,coords.x.toString(),, ,coords.y.toString(),,,,)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        State variables written after the call(s):
        - skLootBagTokenIdToComponentRemoved[_tokenId][_componentToRemove] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in SlotSettlementPool.slashAndBuySlot(address,bytes,address,uint256,bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        External calls:
        - slash(_stakeHouse,_memberId,_amount,_isKickRequired) (contracts/banking/SlotSettlementPool.sol#128)
                - StakeHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - shareToken.mint(address(this),sharesToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
        State variables written after the call(s):
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseCurrentSlotSlashed[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
                - stakeHouseMemberCurrentSlotSlashed[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)

Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):
        External calls:
        - lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - bNFTProxy = new StakeHouseUpgradeableProxy(address(_brandNftLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(BrandCentral(_brandCentralLogic).init.selector,address(this),_brandNft,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#195-205)
        State variables written after the call(s):
        - brandCentral = BrandCentral(brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,bytes).account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#63-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#79-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#97-100)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#190)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
TransactionManager.createStakehouse(bytes,string,uint256,ETH2Validation.ETH2DataReport,ETH2Validation.ETH2DataReportSignature) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender,_blsPublicKey,_ticker,_buildingTypeId,_eth2Report) (contracts/accounts/TransactionManager.sol#86-88)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked((,coords.x.toString(),, ,coords.y.toString(),,)),skLOOT.ItemType.sLand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType.sBuilding,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openskLootBagAndRemoveItem(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.ItemType.sCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

proxies/UpgradeableBeacon.sol

```
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-36) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#32-34)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.9', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/proxy/beacon/IBeacon.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3)
        - 0.8.9 (contracts/StakeHouseAccessControls.sol#3)
        - 0.8.9 (contracts/proxies/UpgradeableBeacon.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#89-95) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#108-114) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#168-170) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#141-143) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#39-50) is never used and should be removed
Strings.toString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#14-34) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/proxy/beacon/IBeacon.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (contracts/StakeHouseAccessControls.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.9 (contracts/proxies/UpgradeableBeacon.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter StakeHouseAccessControls.isAdmin(address)._address (contracts/StakeHouseAccessControls.sol#48) is not in mixedCase
Parameter StakeHouseAccessControls.isProxyAdmin(address)._address (contracts/StakeHouseAccessControls.sol#53) is not in mixedCase
Parameter StakeHouseAccessControls.isCoreModule(address)._address (contracts/StakeHouseAccessControls.sol#58) is not in mixedCase
Parameter StakeHouseAccessControls.isCoreModuleManager(address)._address (contracts/StakeHouseAccessControls.sol#63) is not in mixedCase
Parameter StakeHouseAccessControls.isCoreModuleAdmin(address)._address (contracts/StakeHouseAccessControls.sol#68) is not in mixedCase
Parameter StakeHouseAccessControls.addAdmin(address)._address (contracts/StakeHouseAccessControls.sol#73) is not in mixedCase
Parameter StakeHouseAccessControls.removeAdmin(address)._address (contracts/StakeHouseAccessControls.sol#86) is not in mixedCase
Parameter StakeHouseAccessControls.addProxyAdmin(address)._address (contracts/StakeHouseAccessControls.sol#91) is not in mixedCase
Parameter StakeHouseAccessControls.removeProxyAdmin(address)._address (contracts/StakeHouseAccessControls.sol#102) is not in mixedCase
Parameter StakeHouseAccessControls.addCoreModule(address)._address (contracts/StakeHouseAccessControls.sol#109) is not in mixedCase
Parameter StakeHouseAccessControls.removeCoreModule(address)._address (contracts/StakeHouseAccessControls.sol#120) is not in mixedCase
Parameter StakeHouseAccessControls.lockCoreModule(address)._coreModule (contracts/StakeHouseAccessControls.sol#128) is not in mixedCase
Parameter StakeHouseAccessControls.addCoreModuleManager(address)._address (contracts/StakeHouseAccessControls.sol#136) is not in mixedCase
Parameter StakeHouseAccessControls.removeCoreModuleManager(address)._address (contracts/StakeHouseAccessControls.sol#147) is not in mixedCase
Parameter StakeHouseAccessControls.addCoreModuleAdmin(address)._address (contracts/StakeHouseAccessControls.sol#154) is not in mixedCase
Parameter StakeHouseAccessControls.removeCoreModuleAdmin(address)._address (contracts/StakeHouseAccessControls.sol#165) is not in mixedCase
Parameter UpgradeableBeacon.updateImplementation(address)._implementation (contracts/proxies/UpgradeableBeacon.sol#48) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable UpgradeableBeacon.constructor(StakeHouseAccessControls,address)._implementation (contracts/proxies/UpgradeableBeacon.sol#34) is too similar to UpgradeableBeacon.implementation_ (contracts/proxies/UpgradeableBeacon.sol#18)
Variable UpgradeableBeacon.updateImplementation(address)._implementation (contracts/proxies/UpgradeableBeacon.sol#48) is too similar to UpgradeableBeacon.implementation_ (contracts/proxies/UpgradeableBeacon.sol#18)
Variable UpgradeableBeacon._setImplementation(address)._implementation (contracts/proxies/UpgradeableBeacon.sol#60) is too similar to UpgradeableBeacon.implementation_ (contracts/proxies/UpgradeableBeacon.sol#18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#160-164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- The unchecked transfer flagged by Slither can be ignored as the tokens used in the BlockSwap smart contracts are known and follow the ERC20 standard.
- Slither successfully flagged the weak PRNG in the skLootFactory contract.

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 all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

MythX only found some issues in the following smart contracts:

banking/savETHReservePool.sol

Report for contracts/banking/savETHReservePool.sol
https://dashboard.mythx.io/#/console/analyses/ac3c0fd7-2272-44c4-a78f-a3874b88d54e

Line	SWC Title	Severity	Short Description
278	(SWC-113) DoS with Failed Call	Low	Multiple calls are executed in the same transaction.

brand/BrandCentral.sol

Report for contracts/brand/BrandCentral.sol
https://dashboard.mythx.io/#/console/analyses/9aed8d12-8a58-499e-8483-d1a4de703dbe

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

brand/skLOOTFactory.sol

Report for contracts/brand/skLOOTFactory.sol
https://dashboard.mythx.io/#/console/analyses/decbb62f-ab22-4500-9c7d-e477c4a88cb6

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

accounts/AccountManager.sol

Report for contracts/accounts/AccountManager.sol
https://dashboard.mythx.io/#/console/analyses/b54falef-e4ed-4041-8dcd-b501990a2f0b

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

accounts/ETH2ReportValidator.sol

Report for contracts/accounts/ETH2ReportValidator.sol
https://dashboard.mythx.io/#/console/analyses/06ee6ec8-3cd5-4245-bded-f7b6841e24a0

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

- Mythx correctly flagged `block.number` being used as a source of randomness.

THANK YOU FOR CHOOSING

// HALBORN