

// HALBORN

BlockSwap.xyz

Smart Contract Security Audit

Prepared by: Halborn

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

Visit: [Halborn.com](https://Halborn.com)

DOCUMENT REVISION HISTORY	6
CONTACTS	6
1 EXECUTIVE OVERVIEW	7
1.1 INTRODUCTION	8
1.2 AUDIT SUMMARY	8
1.3 TEST APPROACH & METHODOLOGY	8
RISK METHODOLOGY	9
1.4 SCOPE	11
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	12
3 FINDINGS & TECH DETAILS	14
3.1 (HAL-01) MAPPING IS NOT DECREASED AFTER A DEPOSIT - HIGH	15
Description	15
Risk Level	16
Recommendation	17
Remediation Plan	17
3.2 (HAL-02) WEAK PRNG IN SKLOOTFACTORY CONTRACT - LOW	19
Description	19
Risk Level	20
Recommendation	20
Remediation Plan	21
3.3 (HAL-03) LOST ADJUSTED DEPOSIT AMOUNT DIFFERENCE - LOW	22
Description	22
Risk Level	23
Recommendation	23

Remediation Plan	23
3.4 (HAL-04) FUNCTION NOT EXPOSED IN TRANSACTIONMANAGER CONTRACT - LOW	24
Description	24
Risk Level	25
Recommendation	25
Remediation Plan	25
3.5 (HAL-05) LACK OF ZERO ADDRESS CHECK - LOW	27
Description	27
Code Location	27
Risk Level	36
Recommendation	36
Remediation Plan	36
3.6 (HAL-06) USE OF DEPRECATED SETUPROLE FUNCTION - INFORMATIONAL	37
Description	37
Code Location	37
Risk Level	37
Recommendation	37
Remediation Plan	37
3.7 (HAL-07) ACCOUNTMANAGER.GETACCOUNT VIEW FUNCTION CAN BE REMOVED - INFORMATIONAL	38
Description	38
Risk Level	38
Recommendation	39
Remediation Plan	39
3.8 (HAL-08) TRANSACTIONMANAGER.GETWITHDRAWALADDRESS VIEW FUNCTION CAN BE REMOVED - INFORMATIONAL	40

Description	40
Risk Level	40
Recommendation	40
Remediation Plan	41
<b>3.9 (HAL-09) CONSTANT KECCAK VARIABLES ARE TREATED AS EXPRESSIONS, NOT CONSTANTS - INFORMATIONAL</b>	<b>42</b>
Description	42
Risk Level	42
Recommendation	43
Remediation Plan	43
<b>3.10 (HAL-10) USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS - INFORMATIONAL</b>	<b>44</b>
Description	44
Code Location	44
Proof of Concept	44
Risk Level	45
Recommendation	45
Remediation Plan	46
<b>3.11 (HAL-11) NO NEED TO INITIALIZE UINT256 I VARIABLE TO 0 - INFORMATIONAL</b>	<b>47</b>
Description	47
Code Location	47
Risk Level	47
Recommendation	47
Remediation Plan	48
<b>3.12 (HAL-12) LONG REVERT STRINGS - INFORMATIONAL</b>	<b>49</b>
Description	49

Code Location	49
Recommendation	52
Remediation Plan	53
3.13 (HAL-13) INCORRECT MESSAGE IN REQUIRE STATEMENT - INFORMATIONAL	54
Description	54
Code Location	54
Risk Level	55
Recommendation	55
Remediation Plan	55
3.14 (HAL-14) STATE VARIABLES MISSING CONSTANT MODIFIER - INFORMATIONAL	56
Description	56
Risk Level	56
Recommendation	56
Remediation Plan	56
3.15 (HAL-15) STATE VARIABLES MISSING IMMUTABLE MODIFIER - INFORMATIONAL	57
Description	57
Risk Level	57
Recommendation	57
Remediation Plan	58
3.16 (HAL-16) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL	59
Description	59
Risk Level	59
Recommendation	59

Remediation Plan		60
4	MANUAL TESTING	61
4.1	CONTRACT INITIALIZATION	62
5	AUTOMATED TESTING	64
5.1	STATIC ANALYSIS REPORT	65
Description		65
Slither results		65
5.2	AUTOMATED SECURITY SCAN	102
Description		102
MythX results		102

## 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

DRAFT

## 1.1 INTRODUCTION

BlockSwap.xyz 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.xyz 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:

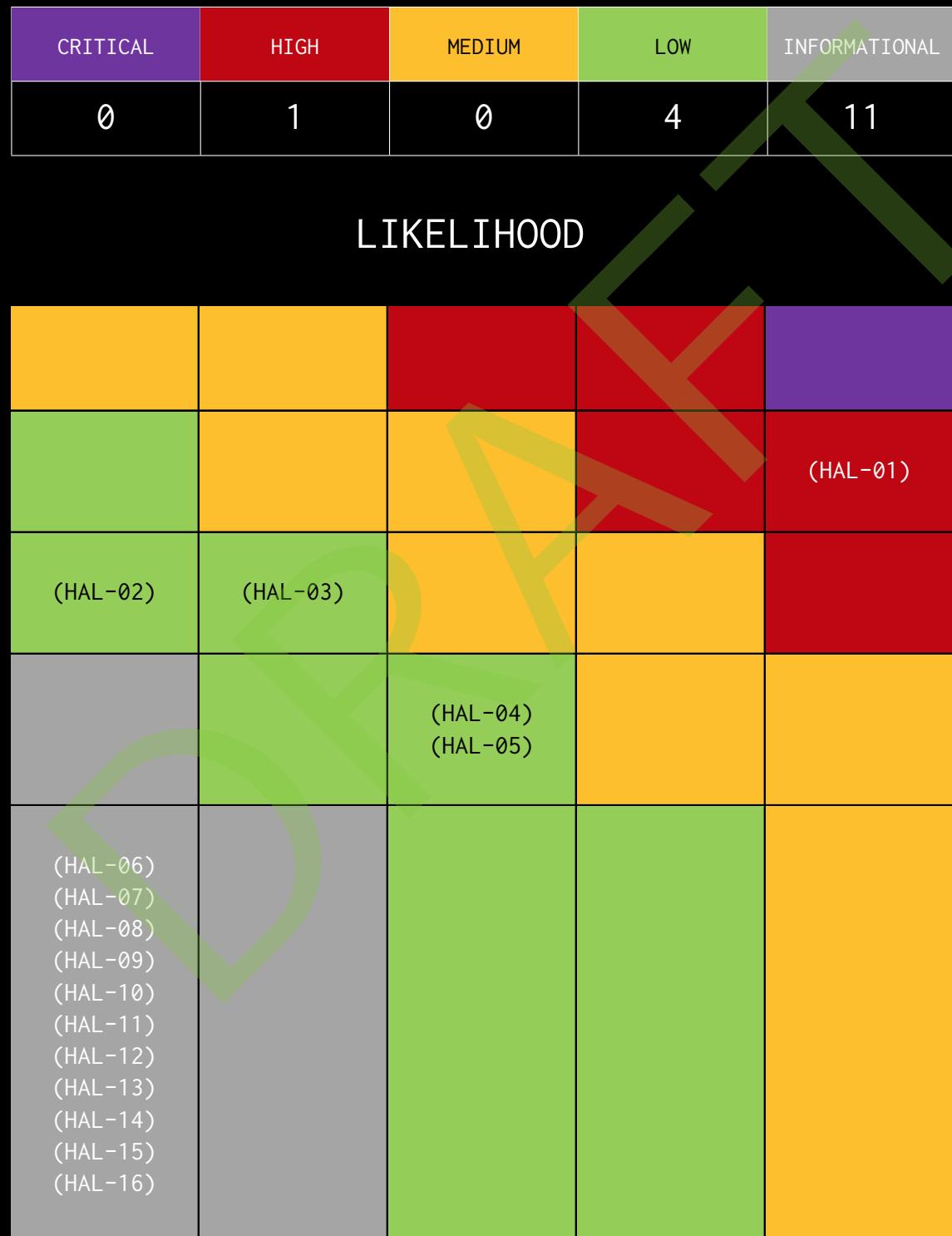
The security assessment was scoped to the following smart contracts:

- 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



# EXECUTIVE OVERVIEW

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.GETWITHDRAWALADDRESS 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

DRAFT

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

```
274         emit FundsSentToDepositContract(_stakeHouse, _blsPublicKey  
275             , depositAmount);  
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  
25     queued to be sent to the deposit contract  
26 mapping(address => mapping(bytes => uint256)) public  
27     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]
261         += _amount;
262     if (stakeHouseMemberQueue[_stakeHouse][_blsPublicKey] >= 1
263         ether) {
264         uint256 depositAmount = stakeHouseMemberQueue[_stakeHouse]
265             [_blsPublicKey];
266         // Deposit amount sent to the deposit contract must be a
267         // multiple of 1 gwei so adjust deposit amount accordingly
268         depositAmount -= depositAmount % 1 gwei;
269         bytes memory _blsSignature = universe.accountManager().
270             getSignatureByBLSKey(_blsPublicKey);
271         /// Deposit amount is divided by 1 gwei, because Deposit
272         // contract only tracks the balance up to 1 gwei precision
273         bytes32 leaf = ETH2Validation.getDepositDataRoot(
274             _blsPublicKey, _blsSignature, WITHDRAWAL_CREDENTIALS,
275             depositAmount / 1 gwei);
276         // Adjust the member queue by the amount of value being
277         // sent to the deposit contract
278         stakeHouseMemberQueue[_stakeHouse][_blsPublicKey] -=
279             depositAmount;
```

```
275      /// Send deposit topup to the contract
276      DepositContract.deposit{value: depositAmount}(
277          _blsPublicKey, WITHDRAWAL_CREDENTIALS, _blsSignature,
278          leaf);
279      emit FundsSentToDepositContract(_stakeHouse, _blsPublicKey
280          , depositAmount);
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
259         central");
260
261     // Source of entropy
262     uint256 numberKnotsInHouse = StakeHouseRegistry(_stakeHouse)
263         .numberOfMemberKNOTS();
264     uint256 numberHouseKnotInUniverse = brandCentral.universe().
265         numberOfStakeHouses();
266     uint256 totalKnotsInUniverse = numberKnotsInHouse +
267         numberHouseKnotInUniverse;
268
269     bool isSpecial = _blockNumber() % 50 == 0;
270
271     // Generate a pseudo random number using above and blockchain
272     // entropy
273     // in theory, miners dont manipulate basefee as that is burnt
274     // - EIP1559
275     uint256 pseudoRandomNumber = uint256(keccak256(abi.
276         encodePacked(
277             block.difficulty,
278             block.timestamp,
```

```
279     // pluck an item depending on whether its special or knot :)
280     string memory pickedItem;
281     if (isSpecial) {
282         string[6] memory _specialLuckyDipGems =
283             specialLuckyDipGems();
284         pickedItem = _specialLuckyDipGems[pseudoRandomNumber %
285             _specialLuckyDipGems.length];
286     } else {
287         string[8] memory _luckyDipItems = luckyDipItems();
288         pickedItem = _luckyDipItems[pseudoRandomNumber %
289             _luckyDipItems.length];
290     }
291     // mint the token
292     uint256 tokenId = skLoot.mint(pickedItem, skLOOT.ItemType.
293         sItem, _brandTokenId, _recipient);
294     emit skLootItemClaimedForKnot(tokenId);
295 }
```

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.xyz](#) 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]
259         += _amount;
260     if (stakeHouseMemberQueue[_stakeHouse][_blsPublicKey] >= 1
261         ether) {
262         uint256 depositAmount = stakeHouseMemberQueue[_stakeHouse]
263             ][_blsPublicKey];
264         // Deposit amount sent to the deposit contract must be a
265         // multiple of 1 gwei so adjust deposit amount accordingly
266         depositAmount -= depositAmount % 1 gwei;
267
268         bytes memory _blsSignature = universe.accountManager().
269             getSignatureByBLSKey(_blsPublicKey);
270
271         /// Deposit amount is divided by 1 gwei, because Deposit
272         // contract only tracks the balance up to 1 gwei precision
273         bytes32 leaf = ETH2Validation.getDepositDataRoot(
274             _blsPublicKey, _blsSignature, WITHDRAWAL_CREDENTIALS,
275             depositAmount / 1 gwei);
276
277         /// Send deposit topup to the contract
278         DepositContract.deposit{value: depositAmount}(
279             _blsPublicKey, WITHDRAWAL_CREDENTIALS, _blsSignature,
280             leaf);
281
282     }
283 }
```

```
274         emit FundsSentToDepositContract(_stakeHouse, _blsPublicKey  
275             , depositAmount);  
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.xyz` 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.
209                         EXITED));
210     blsPublicKeyToLastState[_blsPublicKey] = _report;
211     /// Initialize the rage quit of the Knot
212     universe.rageQuitKnot(
213         _stakeHouse,
214         _blsPublicKey,
215         _rageQuitter,
216         _amountOfETHInDepositQueue
217     );
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:96:           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.xyz` 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,
```

## FINDINGS & TECH DETAILS

```
128     ETH2Validation.ETH2DataReportSignature calldata
129     _reportSignature
130 ) external override onlyValidStakeHouse(_stakehouse) {
131     require(_isReportSignatureValid(_blsPublicKey, _eth2Report,
132         _reportSignature), 'Report signature invalid');
133
134     _performETH2DataCheckRageQuit(_blsPublicKey, _eth2Report);
135
136     accountManager.rageQuitKnot(
137         msg.sender,
138         _blsPublicKey,
139         _stakehouse,
140         0, // As the KNOT was never active, there will be no funds
141         // in queue for top up if it is exiting
142         _eth2Report
143     );
144 }
```

### 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  
22     _stakeHouse) external initializer {  
23     slotSettlementPool = _slotSettlementPool;  
24     stakeHouse = _stakeHouse;  
25     __ERC20_init(  
26         "sETH",  
27         "sETH"  
28     );  
29     __ERC20Permit_init("sETH");  
30 }  
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)
129     external initializer {
130     brandCentral = _brandCentral;
131
132     __ERC721_init("skLootBag", "skLootBag");
```

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

StakeHouseUpgradeableProxy.sol

**Listing 18: StakeHouseUpgradeableProxy.sol**

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

UniverseUpgradeableProxy.sol

**Listing 19: UniverseUpgradeableProxy.sol**

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

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         );
41     _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             ;
91         require(_sETHBeacon != address(0), "sETH beacon cannot be
92             zero address");
93         require(_stakeHouseRegistryBeacon != address(0), "Registry
94             beacon cannot be zero address");
95     }
96     accessControls = _accessControls;
97     StakeHouseUpgradeableProxy accountManagerProxy = new
98         StakeHouseUpgradeableProxy(
99             _accountManagerLogic,
100            address(this),
101            abi.encodeWithSelector(
102                AccountManager(_accountManagerLogic).init.selector,
103                address(this)
104            );
105 }
```

```
105     accountManager = AccountManager(address(accountManagerProxy));  
106  
107     StakeHouseUpgradeableProxy transactionManagerProxy = new  
108         StakeHouseUpgradeableProxy(  
109             _transactionManagerLogic,  
110             address(this),  
111             abi.encodeWithSelector(  
112                 TransactionManager(_transactionManagerLogic).init.  
113                     selector,  
114                     address(this),  
115                     address(accountManagerProxy),  
116                     _depositRouter,  
117                     _minDataEpochHeight  
118             )  
119         );  
120  
121     transactionManager = TransactionManager(address(  
122         transactionManagerProxy));  
123  
124     StakeHouseUpgradeableProxy settlementProxy = new  
125         StakeHouseUpgradeableProxy(  
126             _settlementPoolLogic,  
127             address(this),  
128             abi.encodeWithSelector(  
129                 SlotSettlementPool(_settlementPoolLogic).init.selector  
130             ,  
131             address(this),  
132             _sETHBeacon  
133         );  
134     slotSettlementPool = SlotSettlementPool(address(  
135         settlementProxy));  
136  
137     StakeHouseUpgradeableProxy saveETHReservePoolProxy = new  
138         StakeHouseUpgradeableProxy(  
139             _saveETHReservePoolLogic,  
140             address(this),  
141             abi.encodeWithSelector(  
142                 saveETHReservePool(_saveETHReservePoolLogic).init.  
143                     selector,  
144                     address(this),  
145                     _saveETHLogic  
146             )  
147         );
```

```
141     );
142
143     saveETHPool = saveETHReservePool(address(
144         saveETHReservePoolProxy));
145
146     address _savETHManagerLogic = address(new savETHManager());
147
148     StakeHouseUpgradeableProxy saveETHManagerProxy = new
149         StakeHouseUpgradeableProxy(
150             _savETHManagerLogic,
151             address(this),
152             abi.encodeWithSelector(
153                 savETHManager(_savETHManagerLogic).init.selector,
154                 address(this)
155             )
156         );
157
158     savETHMan = savETHManager(address(saveETHManagerProxy));
159
160     emit CoreModulesInit();
161 }
162
163 /// @dev Due to Solidity stack limitations on how many vars can be
164 ///      passed into a fn, this inits brand central separately
165 /// @param _brandCentralLogic Logic contract for Brand Central
166 /// @param _brandNftLogic Logic contract for the brand NFT
167 /// @param _lootFactoryLogic Logic contract for skLootFactory
168 /// @param _skLootLogic Logic contract or skLoot NFT
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
```

## FINDINGS & TECH DETAILS

```
        StakeHouseUpgradeableProxy(
180            address(_lootFactoryLogic),
181            address(this),
182            abi.encodePacked(""))
183    );
184
185    address _skLootFactory = address(lootFactoryProxy);
186
187    StakeHouseUpgradeableProxy bNFTProxy = new
188        StakeHouseUpgradeableProxy(
189            address(_brandNftLogic),
190            address(this),
191            abi.encodePacked(""))
192
193    address _brandNft = address(bNFTProxy);
194
195    StakeHouseUpgradeableProxy brandCentralProxy = new
196        StakeHouseUpgradeableProxy(
197            _brandCentralLogic,
198            address(this),
199            abi.encodeWithSelector(
200                BrandCentral(_brandCentralLogic).init.selector,
201                address(this),
202                _brandNft,
203                _skLootFactory,
204                _claimAuction
205            )
206        );
207
208    address brandCentralAddress = address(brandCentralProxy);
209    brandCentral = BrandCentral(brandCentralAddress);
210
211    // init proxies
212    BrandNFT(_brandNft).init(brandCentralAddress);
213    skLOOTFactory(_skLootFactory).init(brandCentral, _skLootLogic)
214    ;
215
216    emit BrandCentralInit();
217 }
```

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.xyz 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.xyz` 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.xyz 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.xyz team acknowledged this issue.

DRAFT

### 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.xyz](#) 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 postincrement(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: 0xlecedede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b
Gas price: 0.0 gwei Gas limit: 6721975 Nonce: 44
test.postiincrement confirmed Block: 13622335 Gas used: 21620 (0.32%)

<Transaction '0xlecedede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b'>
>>> test_contract.preiincrement(1)
Transaction sent: 0x205f09a4d2268de4cla40f35bb2ec2847bf2ab8d584909b42c7la022b047614a
Gas price: 0.0 gwei Gas limit: 6721975 Nonce: 45
test.preiincrement confirmed Block: 13622336 Gas used: 21593 (0.32%)

<Transaction '0x205f09a4d2268de4cla40f35bb2ec2847bf2ab8d584909b42c7la022b047614a'>
>>> 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: 0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20e1de05
Gas price: 0.0 gwei Gas limit: 6721975 Nonce: 47
test.preiincrement confirmed Block: 13622338 Gas used: 22601 (0.34%)

<Transaction '0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20e1de05'>
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

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

### Remediation Plan:

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

DRAFT

### 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` reassigned 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.xyz` team solved the issue by not initializing any `uint256` variable to `0` anymore.

DRAFT

## 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.xyz team acknowledged this issue.

DRAFT

### 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
61     calldata _blsSignature
62 ) external override onlyModule {
63     require(_getLifeCycleStatus(_blsPublicKey) == uint256(
64         LifecycleStatus.UNBEGUN), 'Lifecycle status not 0');
65     require(_blsPublicKey.length == 48, 'Invalid public key length
66         ');
67     require(_blsSignature.length == 96, 'Signature length is not
68         incorrect');
69     require(_depositor != address(0), "Depositor cannot be zero");
70
71     // Update validator lifecycle status to INITIALS_REGISTERED
72     (1)
73     _setLifeCycleStatus(_blsPublicKey, uint256(LifecycleStatus.
74         INITIALS_REGISTERED));
75
76     // Create account object
77     Account memory account;
78
79     // Update the account initials
80     account.initials = ValidatorInitials(_depositor, _blsSignature
81         );
82
83     // Map the account BLS public key to the index in the account
84     // array
85     blsPubKeyToAccountArrayIndex[_blsPublicKey] = accounts.length;
```

```
78  
79     /// Add account to the account array  
80     accounts.push(account);  
81  
82     emit InitialsRegistered(_depositor, _blsPublicKey,  
83     _blsSignature);  
84 }
```

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.xyz](#) 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.xyz` 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.xyz` team solved the issue by adding the `immutable` modifier to all the state variables mentioned.

DRAFT

### 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.xyz](#) team solved the issue by marking most of the functions mentioned as external.

DRAFT

# MANUAL TESTING

DRAFT

## 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	-	<pre> 31   function init(StakeHouseUniverse _universe) external initializer { 32     _initModuleGuards(_universe); 33   } </pre>
BalanceReporter ETH2ReportValidator	ETH2ReportValidator EIP712Upgradeable, ModuleGuards	No Yes. This function is internal. The function is initialized by TransactionManager contract to TransactionManager inherits from BalanceReporter which at the same time inherits from ETH2ReportValidator	Yes	<pre> 25   function _init_ETH2ReportValidator( 26     address _depositRouter, 27     uint256 _minEpochHeight, 28     string memory _eip712ContractName 29   ) internal initializer { 30     EIP712_init_unchained(_eip712ContractName, "1"); 31     _REPORT_TYPEHASH = keccak256( 32       "Report(bytes blsPubKey,bytes32 reportHash,uint256 deadline,uint256 blsPubKeyInternalNonce)" 33     ); 34 35     require(_depositRouter != address(0), "Deposit Router can not be address 0"); 36 37     depositRouter = _depositRouter; 38 39     require(_minEpochHeight &gt; 0, "Epoch height non-positive"); 40     lastKnownEpochHeight = _minEpochHeight; 41   } </pre>
ETH2Validation Streamer	-	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	<pre> 20   function init( 21     StakeHouseUniverse _universe, 22     address _accountManager, 23     address _depositRouter, 24     uint256 _minDataEpoch 25   ) external initializer { 26     WITHDRAWAL_CREDENTIALS = ETH2Validation._computeWithdrawalCredentials(_accountManager); 27     require(WITHDRAWAL_CREDENTIALS.length == 32, "Withdrawal credentials computed incorrectly"); 28 29     require(_accountManager != address(0), "Account manager can not be address 0"); 30     accountManager = AccountManager(_accountManager); 31 32     _initModuleGuards(_universe); 33     _init_ETH2ReportValidator(_depositRouter, _minDataEpoch, "TransactionManager"); 34   } </pre>
Banking CollateralisedSlotManager dETH savETH	- - ERC20, ERC20Permit ERC20Upgradeable, ERC20PermitUpgradeable, Exponential	No No Yes, correctly initializes ERC20Upgradeable and ERC20PermitUpgradeable	-	<pre> 18   function init(savETHReservePool _reservePool) external initializer { 19     require(address(_reservePool) != address(0), "Reserve pool cannot be zero address"); 20 21     ERC20_init("savETH", "savETH"); 22     ERC20Permit_init("savETH"); 23     reservePool = _reservePool; 24   } </pre>
savETHManager	Initializable, ISavETHManager	Yes. His parent contracts do not need to be initialized	Yes	<pre> 14   function init(StakeHouseUniverse _universe) external initializer { 15     _universe = _universe; 16   } </pre>
savETHReservePool	Initializable, ISavETHReservePool, Exponential, ModuleGuards	Yes, correctly initializes ModuleGuards. At the same time, it creates a new StakeHouseUpgradeableProxy which implementation is a savETH contract, correctly initializing savETH contract as well	Yes	<pre> 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         saveETH._saveETHLogic.selector, 62         address(this) 63       ) 64     ); 65 66     saveETHToken = saveETH(address(saveETHProxy)); 67 68     _initModuleGuards(_universe); 69   } </pre>
sETH	ERC20Upgradeable, ERC20PermitUpgradeable, Exponential	Yes, correctly initializes ERC20Upgradeable and ERC20PermitUpgradeable	Yes	<pre> 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   } </pre>
SlotSettlementPool	Initializable, ISlotSettlementPool, CollateralisedSlotManager, Exponential, ModuleGuards	Yes, correctly initializes ModuleGuards. The rest of his parent contracts do not need to be initialized	Yes	<pre> 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   } </pre>

SlotToken	ERC20			
BrandCentral	Initializable, IBrandCentral, ModuleGuards	Yes, correctly initializes ModuleGuards. The rest of his parent contracts do not need to be initialized	Yes	<pre> 54     function init( 55         StakehouseUniverse _universe, 56         BrandNFT _brandNFT, 57         skLOOTFactory _skLOOTFactory!, 58         BrandCentralClaimAuction _claimAuction! 59     ) external initializer { 60         __initModuleGuards(_universe); 61         brandNFT = _brandNFT; 62         skLOOTFactory = _skLOOTFactory!; 63         claimAuction = _claimAuction; 64     } 65 }</pre>
BrandCentralClaimAuction	ERC721	No		
BrandNFT	ERC721Upgradeable	Yes, correctly initializes ERC721Upgradeable	Yes	<pre> 47     function init(address _brandCentral) external initializer { 48         brandCentral = IBrandCentral(_brandCentral); 49         ... 50     } 51 }</pre>
skLOOT	ERC721Upgradeable	Yes, correctly initializes ERC721Upgradeable	Yes	<pre> 39     function init(skLOOTFactory _lootFactory) external initializer { 40         lootFactory = _lootFactory; 41         ... 42     } 43 }</pre>
skLOOTFactory	ERC721Upgradeable	Yes, correctly initializes ERC721Upgradeable. At the same time, it creates a new StakeHouseUpgradeableProxy which implements logic in a skLOOT contract, correctly initializing skLOOT contract as well		<pre> 128     function init(BrandCentral _brandCentral, address _skLootLogic!) external initializer { 129         brandCentral = _brandCentral; 130         ... 131         _ERC721_init("skLootBag", "skLootBag"); 132         StakeHouseUpgradeableProxy skLootProxy = new StakeHouseUpgradeableProxy( 133             _skLootLogic!, 134             address(brandCentral.universe()), 135             abi.encodeWithSelector( 136                 skLOOT(_skLootLogic!).init.selector, 137                 address(this) 138             ) 139         ); 140         skLoot = skLOOT(address(skLootProxy)); 141     } 142 }</pre>
StakeHouseAccessControls	AccessControl			
StakeHouseRegistry	Initializable, IMembershipRegistry, ModuleGuards	No		
StakeHouseUniverse	Initializable, IStakHouseUniverse, Banking, Pausable	Yes, correctly initializes ModuleGuards. The rest of his parent contracts do not need to be initialized	Yes	<pre> 38     function init() external initializer { 39         __initModuleGuards(_universe); 40     } 41 }</pre>

init() Deployes multiple StakeHouseUpgradeableProxy. Each of those StakeHouseUpgradeableProxy deploy and initialize the following contracts:  
 - AccountManager  
 - StakingManager  
 - StakSweepPool  
 - saveETHReservePool  
 - saveETHManager

superchargeAndInitBrandCentral() Deployes 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.



# 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#8)
- 0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#8)
- 0.8.0 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#8)
- 0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#8)
- 0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#8)
- 0.8.0 (node_modules/@openzeppelin/contracts/introspection/IERC165.sol#8)
- 0.8.5 (contracts/StakeHouseAccessControls.sol#8)

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

Context, msgData() (node_modules/@openzeppelin/contracts/context/Context.sol#22) is never used and should be removed
Strings, collectString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#39-50) is never used and should be removed
Strings, toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#44-54) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-without-headers

Pragma version"0.8.0" (node_modules/@openzeppelin/contracts/introspection/IERC165.sol#8) 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#8) 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#8) 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/math/Math.sol#8) 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/introspection/IERC165.sol#8) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version"0.8.9" (contracts/StakeHouseAccessControls.sol#8) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version"0.8.9" (contracts/StakeHouseAccessControls.sol#8) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
pragma solidity ^0.8.0; (contracts/StakeHouseAccessControls.sol#8) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter StakeHouseAccessControls._admin(address), address (contracts/StakeHouseAccessControls.sol#8) is not in mixedCase
Parameter StakeHouseAccessControls._iProxyAdmin(address), address (contracts/StakeHouseAccessControls.sol#55) is not in mixedCase
Parameter StakeHouseAccessControls._coreModule(address), address (contracts/StakeHouseAccessControls.sol#55) is not in mixedCase
Parameter StakeHouseAccessControls._lockCoreModule(address), address (contracts/StakeHouseAccessControls.sol#125) is not in mixedCase
Parameter StakeHouseAccessControls._removeCoreModule(address), address (contracts/StakeHouseAccessControls.sol#126) is not in mixedCase
Parameter StakeHouseAccessControls._addCoreModuleAdmin(address), address (contracts/StakeHouseAccessControls.sol#147) is not in mixedCase
Parameter StakeHouseAccessControls._removeCoreModuleAdmin(address), address (contracts/StakeHouseAccessControls.sol#148) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-name-conventions

renounceRole(bytes32,address) should be declared external
Address (contracts/StakeHouseAccessControls.addCoreModuleAdmin(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

# AUTOMATED TESTING

# AUTOMATED TESTING

## StakeHouseUniverse.sol

```
skLOOTFactory.skLocItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length]" (contracts/brand/skLOOTFactory.sol#286)*  
skLOOTFactory.skLocItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length]" (contracts/brand/skLOOTFactory.sol#286)*  
skLOOTFactory.skLocItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length]" (contracts/brand/skLOOTFactory.sol#286)*  
skLOOTFactory.skLocItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.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: ignores uint256 resultPtx.encode.am_0 - 2.0x8d3d <> 240) (contracts/helpers/Base64.sol#52)  
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: ignores uint256 resultPtx.encode.am_0 - 1.0x8d3d <> 288) (contracts/helpers/Base64.sol#55)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup  
ERC20Upgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#86) shadows:  
- ContextUpgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)  
ERC20Permit._gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20Permit/ERC20PermitUpgradeable.sol#93) shadows:  
- IERC165Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/introspection/IERC165Upgradeable.sol#311)  
- ERC20Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#341)  
ContextUpgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30) shadows:  
- ERC165Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/introspection/ERC165Upgradeable.sol#35)  
- ContextUpgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#35)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-shadowing  
savETHReservePool.addNonceToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/savETHReservePool.sol#208)  
savETHReservePool.addNonceToOpenPoolAndWithdrawETH(address,bytes,address) (contracts/banking/savETHReservePool.sol#216-239) ignores return value by dETHToken.transfer(currentKeeper,dETHFromChangegate) (contracts/banking/savETHReservePool.sol#235)  
savETHReservePool.lendFromRootToOpenPool(address,bytes,address) (contracts/banking/savETHReservePool.sol#242-245) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),savETHRequiredForIsolation) (contracts/banking/savETHReservePool.sol#242)  
savETHReservePool.withdrawETH(address,uint256) (contracts/banking/savETHReservePool.sol#255-304) 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.claimAuctionBidFromOwner(string,address) (contracts/brand/BrandCentralClaimAuction.sol#109-115) ignores return value by shbToken.transferFrom(msg.sender,address(this),shbbidAmount) (contracts/brand/BrandCentralClaimAuction.sol#115)  
BrandCentralClaimAuction.bidFromTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#118-199) ignores return value by shbToken.transfer(msg.sender,auction.shbbid) (contracts/brand/BrandCentralClaimAuction.sol#118)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfers
```

## banking/Banking.sol

```

skLOOTFactory,skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PENG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"  
skLOOTFactory,skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PENG: "pickedItem = _specialLuckyDipItems[pseudoRandomNumber % _specialLuckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"  
233)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PENG  
  

Base64.encode(Bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256, uint256)(resultPrf_encode_am_0 - 2,0x8d3d << 240) (contracts/helpers/Base64.sol#52)  
Base64.encode(Bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(uint256, uint256)(resultPrf_encode_am_0 - 1,0x8d3d << 248) (contracts/helpers/Base64.sol#55)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup  
  

ERC16Upgradable_gaps (node_modules/@openzeppelin/contracts-upgradeable/ERC16Upgradable.sol#86) shadows:  
- ContextUpgradable_gaps (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#80)  
ERC20PermitModule_gaps (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/Permit/ERC20PermitModule.sol#93) shadows:  
- EIP1155Upgradeable_gaps (node_modules/@openzeppelin/contracts-upgradeable/introspection/EIP1155Upgradeable.sol#111)  
- ERC20Upgradable_gaps (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradable.sol#94)  
ContextUpgradable_gaps (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#141) shadows:  
- ERC16Upgradable_gaps (node_modules/@openzeppelin/contracts-upgradeable/introspection/ERC16Upgradable.sol#35)  
ContextUpgradable_gaps (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#141) shadows:  
- ERC16Upgradable_gaps (node_modules/@openzeppelin/contracts-upgradeable/introspection/ERC16Upgradable.sol#35)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#context-variable-shadowing  
  

saveETHReservePool.addEthToPool(address,bytes,address) (contract/banking/saveETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,addrEthRecipient) (contracts/banking/saveETHReservePool.sol#208)  
saveETHReservePool.addEthToPoolWithReward(address,bytes,address) (contract/banking/saveETHReservePool.sol#214-235) ignores return value by saveETHToken.transfer(currentKeeper,addrEthRecipient) (contracts/banking/saveETHReservePool.sol#235)  
saveETHReservePool.deposit(address,bytes,uint256) (contract/banking/saveETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),addrEthRecipient) (contracts/banking/saveETHReservePool.sol#257)  
saveETHReservePool.withdraw(address,bytes,uint256) (contract/banking/saveETHReservePool.sol#285-304) ignores return value by saveETHToken.transfer(_owner,addrEthRecipient) (contracts/banking/saveETHReservePool.sol#301)  
saveETHReservePool.deposit(address,uint256) (contract/banking/saveETHReservePool.sol#307-324) ignores return value by saveETHToken.transferFrom(_owner,address(this),addrEthRecipient) (contracts/banking/saveETHReservePool.sol#327)  
BrandCentralClaimAuction.claimShb(uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(may.sender,address(this),shbBidDecount) (contracts/brand/BrandCentralClaimAuction.sol#105)  
BrandCentralClaimAuction.claimShb(uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-199) ignores return value by shbToken.transfer(may.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#105)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer  
  

BrandCentralClaimAuction._shbForConstructorCost(address,bytes) (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:  
- BLOCKS PER DAY * 2448 * FEE * 1000 / BLOCK (contracts/brand/BrandCentralClaimAuction.sol#127)  
- 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:  
- encodePacked((string,bytes)) (contracts/helpers/Base64.sol#47)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply  
  

skLOOTFactory,skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a dangerous strict equality:  
- isSpecial1 = blockhash(blockNumber) % 50 == 5 (contracts/brand/skLOOTFactory.sol#45)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities  
  

Reentrancy in BrandCentralClaimAuction.bidForTicket(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):  
External calls:  
- ShbToken.transferFrom(bidder,auction,shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)  
Start variables written after the call(s):  
- auction.ebbid = shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)  
- auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)  
- auction.biddingEnd = auction.biddingEnd + 1 (contracts/brand/BrandCentralClaimAuction.sol#134)  
- auction.biddingEnd += auction.biddingEnd + 1 BID EXTENSION IN BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#152)  
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-180):  
External calls:  
- shareToken.mint(address(this),shareToMint) (contracts/banking/SlotSettlementPool.sol#181-184)  
State variables written after the call(s):  
- _increaseCollateralizedBalance(shareToken,recipient,_memberId,sharesPaid) (contracts/banking/SlotSettlementPool.sol#187)  
- stakeToken.transfer(stakeTokenTotalCollateralBalance(ATTRIBOKEN) += amount (contracts/banking/CollateralizedLotManager.sol#40)  
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(addresses) (Contracts/banking/SlotSettlementPool.sol#64-84):  
External calls:  
- ATTRIBOKY = new BeaconProxy(mETHBeacon,shb.encodeWithSelector(mETH(mETHBeacon).init.selector,address(this),_stakehouse)) (contracts/banking/SlotSettlementPool.sol#68-75)  
  

Reentrancy in StakeHouseUniverse._stakeHouseRegistry(stakeHouseRegistry,shb,shb.encodeWithSelector(mETH(mETHBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235):  
External calls:  
- stakeHouseRegistry.setShareHolder(stakeHouseRegistry,shb,shb.encodeWithSelector(mETH(mETHBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#230)  
Reentrancy in StakeHouseUniverse._stakeHouse(stakeHouse,stakeHouseAddress) (Contracts/StakeHouseUniverse.sol#245):  
External calls:  
- stakeHouseRegistry.setShareHolder(stakeHouse,stakeHouseAddress) (Contracts/StakeHouseUniverse.sol#246)  
State variables written after the call(s):  
- _registerSendToManager(stakeHouse,_memberId) (Contracts/StakeHouseUniverse.sol#248)  
Reentrancy in StakeHouseUniverse._stakeHouse(stakeHouse,_memberId) (Contracts/StakeHouseUniverse.sol#242):  
External calls:  
- stakeHouseRegistry.setShareHolder(stakeHouseRegistry,shb,shb.encodeWithSelector(mETH(mETHBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#243)  
- stakeHouseRegistry.setShareHolder(stakeHouse,stakeHouseAddress) (Contracts/StakeHouseUniverse.sol#244)  
State variables written after the call(s):  
- _registerSendToManager(stakeHouse,_memberId) (Contracts/StakeHouseUniverse.sol#248)  
Reentrancy in skLOOTFactory.openLoootCapAndRemoveToken(uint256,skLOOTFactory,skAggComponent,address) (Contracts/brand/skLOOTFactory.sol#203-246):  
External calls:  
- skLOOTItemClaim(string(shb.encodePacked((_,coords.x.toString(),_,coords.y.toString()))),skLOOTItemTypes.land,brandTokenId,recipient) (contracts/brand/skLOOTFactory.sol#221-226)  
- skLoot.mint(skLootBalTokenInfo_.tokenId).building,skLOOTItemTypes.building,brandTokenId,recipient) (contracts/brand/skLOOTFactory.sol#228-233)  
- skLoot.mint(skLootBalTokenInfo_.tokenId).character,skLOOTItemTypes.character,brandTokenId,recipient) (contracts/brand/skLOOTFactory.sol#235-240)  
State variables written after the call(s):  
- shb.balanceOf(stakeHouse) (Contracts/brand/skLOOTFactory.sol#241)  
- shb.balanceOf(stakeHouseRegistry,_componentToRemove) = true (contracts/brand/skLOOTFactory.sol#243)

```

## banking/CollateralisedSlotManager.sol

## banking/SlotSettlementPool.sol

```

skLOOTFactory.ejectItemClaim(addresses, address, uint256)(contracts/brand/skLOOTFactory.sol#23-25) uses a weak PWN: "pickedItem = _specieLuckyBipGems[pseudoRandomNumber % _specialLuckyBipGems.length]" (contracts/brand/skLOOTFactory.sol#23)
skLOOTFactory.ejectItemClaim(addresses, address, uint256)(contracts/brand/skLOOTFactory.sol#23-25) uses a weak PWN: "pickedItem = _luckyBipItems[pseudoRandomNumber % _luckyBipItems.length]" (contracts/brand/skLOOTFactory.sol#26)
Erc20Transfer(https://github.com/crytic/slither/contracts/standard/Erc20Token.sol#14-16) contains a known shift operation: "mature(uint256,uint256)" (resultPtr encode imm_0 = 2,0x0d3d < 240) (contracts/helpers/Based4e.sol#2)
Base64.encode(bytes)(contracts/helpers/Base64.sol#21-22) contains a known shift operation: "bytes encode imm_24, uint256(resultPtr encode imm_0 = 1,0x00 < 248)" (contracts/helpers/Base64.sol#20)
Erc20Transfer(https://github.com/crytic/slither/wit/Detector-Documentation#shift-parameter-mix)
ERC20Upgradable(_gap module /OpenZeppelin/contracts-upgradeable/ERC20/ERC20Upgradeable.sol#14-21) shadows:
- ContextUpgradable, _gap module /OpenZeppelin/contracts-upgradeable/math/ContextUpgradable.sol#14
ERC20Upgradable(_gap module /OpenZeppelin/contracts-upgradeable/math/ContextUpgradable.sol#14) shadows:
- ContextUpgradable, _gap module /OpenZeppelin/contracts-upgradeable/math/ContextUpgradable.sol#14
- ERC20Upgradable, _gap module /OpenZeppelin/contracts-upgradeable/math/ContextUpgradable.sol#14
ERC20Upgradable(_gap module /OpenZeppelin/contracts-upgradeable/math/ContextUpgradable.sol#14) shadows:
- ContextUpgradable, _gap module /OpenZeppelin/contracts-upgradeable/math/ContextUpgradable.sol#14
- ERC20Upgradable, _gap module /OpenZeppelin/contracts-upgradeable/math/ContextUpgradable.sol#14
- ERC165Upgradable, _gap module /OpenZeppelin/contracts-upgradeable/math/ContextUpgradable.sol#15
- ContextUpgradable, _gap module /OpenZeppelin/contracts-upgradeable/math/ContextUpgradable.sol#15
Reference: https://github.com/crytic/slither/wit/Detector-Documentation#state-variable-shadowing

sevETHReservePool.addLiquidityPool(addresses, bytes, address)(contracts/banking/sevETHReservePool.sol#191-211) ignores return value by saveETHReserve.transfer(currentKeeper, saveETHReservePool.sol#208)
sevETHReservePool.addLiquidityPoolWithRate(addresses, bytes, address)(contracts/banking/sevETHReservePool.sol#218-239) ignores return value by dETHToken.transferFrom(currentKeeper, dETHReserveXchageRate)(contracts/banking/sevETHReservePool.sol#218)
sevETHReservePool.isLastLiquidityPoolAdded(addresses, bytes, address)(contracts/banking/sevETHReservePool.sol#242-269) ignores return value by saveETHReserve.transferFrom(_newkeeper, address(this), sevETHReserveRequiredForOperation)(contracts/banking/sevETHReservePool.sol#242)
sevETHReservePool.getLiquidityPool(addresses, uint256)(contracts/banking/sevETHReservePool.sol#275-304) ignores return value by dETHToken.transferFrom(_owner, sevETHReserveXchageRate)(contracts/banking/sevETHReservePool.sol#275)
sevETHReservePool.deposit(uint256)(contracts/banking/sevETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner, address(this), amount)(contracts/banking/sevETHReservePool.sol#307)
BankCentralClaimAuction.bidForTicket(uint256)(contracts/brand/BrandCentralClaimAuction.sol#109-121) ignores return value by sbToken.transferFrom(auction_bidder, auction_shbid)(contracts/brand/BrandCentralClaimAuction.sol#121)
BankCentralClaimAuction.bidForTicket(uint256)(contracts/brand/BrandCentralClaimAuction.sol#110-124) ignores return value by sbToken.transferFrom(mg_sender, addresses(this), sbhdbid)(contracts/brand/BrandCentralClaimAuction.sol#124)
BankCentralClaimAuction.claimAuction(uint256)(contracts/brand/BrandCentralClaimAuction.sol#125-139) ignores return value by sbToken.transferFrom(mg_sender, auction_sbhd)(contracts/brand/BrandCentralClaimAuction.sol#125)

```

```

BrandCentralClaimAuction.silhtherContractingWithSameVariables) (contracts/brand/BrandCentralClaimAuction.sol#14-26) performs a multiplication on the result of a division
    -BLOCKS PER DAY = 84600 / SECONDS PER BLOCK (contracts/brand/BrandCentralClaimAuction.sol#127)
    -TOTAL_AUCTION_LENGTH IN BLOCKS BYES PER DAY = AUCTION_LENGTH IN DAYS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64Encoder.silhtherContractingWithSameVariables) (contracts/base/Base64Encoder.sol#1-10) performs a multiplication on the result of a division:
    -encodedData = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#14)
References: https://github.com/crytic/silhther/whitepaper/Detector-Documentation#divide-before-multiplying

silLottoFactory.selectItemClaimAddreses,address,bytes) (contracts/silLottoFactory.sol#253-258) uses a dangerous strict equality:
    - isSpecial = blockhash % 50 == 0 (contracts/brand/silLottoFactory.sol#265)
References: https://github.com/crytic/silhther/whitepaper/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicket(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
    External calls:
        - transfer(auction.bidder,auction.bid) (contracts/brand/BrandCentralClaimAuction.sol#125)
    State Variables written after the call(s):
        - auction.bidId = _aibidAndBalance (contracts/brand/BrandCentralClaimAuction.sol#130)
        - auction.bidder = bidder (contracts/brand/BrandCentralClaimAuction.sol#131)
        - auction.biddingHash = blockhash % + BLOCKS_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#134)
        - auction.biddingHash = auction.biddingHash + BID_EXTENSION_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#182)
    Reentrancy in Banker.buySlotSettlementPool.buySlotSettlement(address,bytes,int256,address) (contracts/banking/SlotSettlementPool.sol#161-190):
        External calls:
            - shareToken.mint(address(this),shareStakeMint) (contracts/banking/SlotSettlementPool.sol#1-18)
        State Variables:
            - _shareTokenTotalBalance(address(this),recipient,memberId,shareStakeMint) (contracts/banking/SlotSettlementPool.sol#187)
            - stakeSharesSTETHTotalBalance(_xTokenId) + amount (contracts/banking/CollateralisedDistroManager.sol#140)
    Reentrancy in StakeholderPool.deployStakeholderPool(address) (contracts/banking/StakeholderPool.sol#1-10):
        External calls:
            - sETHProxy = new BeaconProxy(txEthStakePool,abi.encodeWithSelector(sETH.txEthStakePool.init.selector,address(this),stateHouse)) (contracts/banking/SlotSettlementPool.sol#10)

```



## banking/SlotToken.sol

## banking/dETH.sol

# AUTOMATED TESTING

## banking/sETH.sol

```

skLOOTFactory, skLoictemClaim(address, address, uint256) (contracts/brand/skLOOTFactory.sol#203-209) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % luckyDipItems.length]" (contracts/brand/skLOOTFactory.sol#208)
skLOOTFactory, skLoictemClaim(address, address, uint256) (contracts/brand/skLOOTFactory.sol#203-209) uses a weak PRNG: "pickedItem = _specialLuckyDips[pseudoRandomNumber % specialLuckyDipGems.length]" (contracts/brand/skLOOTFactory.sol#208)
Base64.encodeBytes (contracts/helpers/Base64.sol#12-42) contains an incorrect shift operation: masters(uint256, uint256)(resultPrf, encode_msm_0 - 1, 0x8d3 <> 240) (contracts/helpers/Base64.sol#42)
Base64.encodeBytes (contracts/helpers/Base64.sol#12-42) contains an incorrect shift operation: masters(uint256, uint256)(resultPrf, encode_msm_0 - 1, 0xd <> 240) (contracts/helpers/Base64.sol#42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

ERC20Upgradeable, gap (mode modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#161) shadows:
- ContextUpgradeable, _gap (mode modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#160)
ERC20Upgradeable, _gap (mode modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#161) shadows:
- EIP1155Upgradeable, _gap (mode modules/@openzeppelin/contracts-upgradeable/token/ERC20/EIP1155Upgradeable.sol#161)
- EIP20Upgradeable, _gap (mode modules/@openzeppelin/contracts-upgradeable/token/ERC20/EIP20Upgradeable.sol#161)
- ContextUpgradeable, _gap (mode modules/@openzeppelin/contracts-upgradeable/token/ERC20/ContextUpgradeable.sol#161)
- EIP721Upgradeable, _gap (mode modules/@openzeppelin/contracts-upgradeable/token/ERC20/EIP721Upgradeable.sol#161)
- ERC165Upgradeable, _gap (mode modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#161)
- ContextUpgradeable, _gap (mode modules/@openzeppelin/contracts-upgradeable/utils/context/ContextUpgradeable.sol#160)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-shadowing

savETHReservePool.addEthToOpenPool(address, bytes, address) (contract/banking/savETHReservePool.sol#19-21) ignores return value by savETHReservePool.transferFrom(currentFeeRecipient, address) (contract/banking/savETHReservePool.sol#19)
savETHReservePool.addEthToOpenPool(address, bytes, address) (contract/banking/savETHReservePool.sol#19-21) ignores return value by dETHToken.transferFromExchageRate() (contract/banking/savETHReservePool.sol#19)
ePool#235
savETHReservePool.lendFromOpenPool(address, bytes, address) (contract/banking/savETHReservePool.sol#142-149) ignores return value by saveETHToken.transferFrom(newLender, address(this)) (contract/banking/savETHReservePool.sol#142)
savETHReservePool.withdraw(address, bytes, address) (contract/banking/savETHReservePool.sol#107-129) ignores return value by dETHToken.transferFrom(_owner, address(this), amount) (contract/banking/savETHReservePool.sol#127)
BaseCentralClaimAuction.claimSBH(uint256) (contracts/brand/BaseCentralClaimAuction.sol#109-161) ignores return value by sbHToken.transferFrom(mag, render, auction.bidId) (contracts/brand/BaseCentralClaimAuction.sol#109)
BaseCentralClaimAuction.claimSBH(uint256) (contracts/brand/BaseCentralClaimAuction.sol#109-161) ignores return value by sbHToken.transfer(mag, render, auction.bidId) (contracts/brand/BaseCentralClaimAuction.sol#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction.slashConstructorSetVariables() (contracts/brand/BaseCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
-BLOCKS * THE DAY = 84640 * 84640 / BLOCK * (currentAUCTIONLength / BrandCentralClaimAuction.sol#27)
-TOTAL AUCTION LENGTH IN BLOCKS = BLOCKS PER DAY * AUCTION LENGTH IN DAYS (contracts/brand/BaseCentralClaimAuction.sol#28)
Base64.encodeBytes (contracts/helpers/Base64.sol#12-42) performs a multiplication on the result of a division:
- encodeBytes (contracts/helpers/Base64.sol#12-42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#division-before-multiply

skLOOTFactory, skLoictemClaim(address, address, uint256) (contracts/brand/skLOOTFactory.sol#203-209) uses a dangerous strict equality:
- isSpecial = blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#205)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicket(string, uint256) (contracts/brand/BaseCentralClaimAuction.sol#109-161):
External call:
- StakeHouseToken.transfer(Recipient, auction.bidId) (contracts/brand/BaseCentralClaimAuction.sol#120)
    State variables written after the call():
        - auction.bidId = sbHToken.balanceOf(auction.bidder) (contracts/brand/BaseCentralClaimAuction.sol#120)
        - auction.bidder = msg.sender (contracts/brand/BaseCentralClaimAuction.sol#120)
        - auction.bidPrice = auction.bidPrice + 1 (contracts/brand/BaseCentralClaimAuction.sol#120)
        - auction.biddingEnd = auction.biddingEnd + 1 (BID EXTENSION IN BLOCKS) (contracts/brand/BaseCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address, bytes, uint256, address) (contracts/banking/SlotSettlementPool.sol#161-160):
External call:
- shareToken.mint(address(this), shareForMint) (contracts/banking/SlotSettlementPool.sol#161-164)
    State variables written after the call():
        - increaseCollateralisedBalance(shareToken, recipient, memberId, shareForMint) (contracts/banking/SlotSettlementPool.sol#167)
        - stakeHouseTotalCollateralisedBalance(ATTHouseToken) += amount (contracts/banking/CollectiveStakePoolManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(addresses) (contracts/banking/SlotSettlementPool.sol#64-64):
External call:
- sETHProxy = new BeaconProxy(sETHBeacon, sbHToken.encodeWithSelector(sETHBeacon).init.selector, address(this), _stakeHouse) (contracts/banking/SlotSettlementPool.sol#68-75)

State variables written after the call():
- stakedHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in StakeHouseUniverse.registerStakeHouse(address, bytes, uint256) (contracts/banking/savETHReservePool.sol#113-145):
External call:
- saveETHReservePool.mint(address(this), saveETHCountScaled, manisize) (contracts/banking/savETHReservePool.sol#116)
    State variables written after the call():
        - saveETHReservePool.mint(address(this), saveETHCountScaled, manisize) (contracts/banking/savETHReservePool.sol#114)
Reentrancy in StakeHouseUniverse.registerStakeHouse(address, bytes, uint256) (contracts/banking/SlotSettlementPool.sol#85-117):
External call:
- stakeHouseTotalCollateralisedBalance(ATTHouseToken) += amount (contracts/banking/SlotSettlementPool.sol#97)
    State variables written after the call():
        - increaseCollateralisedBalance(shareToken, recipient, memberId, shareForSendManager) (contracts/banking/SlotSettlementPool.sol#104)
        - stake.mint(SLOT PER STAKEHOUSE MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
    State variables written after the call():
        - increaseCollateralisedBalance(shareToken, recipient, memberId, shareForSendManager) (contracts/banking/SlotSettlementPool.sol#116)
Reentrancy in StakeHouseUniverse.registerStakeHouse(address, bytes, uint256) (contracts/banking/SlotSettlementPool.sol#88-117):
External call:
- stakeHouseTotalCollateralisedBalance(ATTHouseToken) (contracts/banking/SlotSettlementPool.sol#97)
    - slot.mint(SLOT PER STAKEHOUSE MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
    - shareToken.mint(recipient, shareToSendManager) (contracts/banking/SlotSettlementPool.sol#110)
    State variables written after the call():
        - increaseCollateralisedBalance(shareToken, recipient, memberId, shareForSendManager) (contracts/banking/SlotSettlementPool.sol#115)
Reentrancy in StakeHouseUniverse.registerStakeHouse(address, bytes, uint256) (contracts/banking/SlotSettlementPool.sol#104-104):
External call:
- registerMemberOfStakeHouse(stakeHouseAddress, stakeHouseName) (contracts/banking/StakeHouseRegistry.sol#229-235)
    - stakeHouse = StakeHouse(stakeHouseAddress) (contracts/banking/StakeHouseRegistry.sol#229)
    State variables written after the call():
        - registerMemberOfStakeHouse(stakeHouseAddress, stakeHouseName) (contracts/banking/StakeHouseRegistry.sol#229)
Reentrancy in skLOOTFactory.openLockBagAndRemoveItem(uint256, skLOOTFactory.skBagComponents, address) (contracts/brand/skLOOTFactory.sol#203-246):

```

# AUTOMATED TESTING

```
External call:
- skLoot.mint(string(_abi.encodePacked{_, coords.x.toString(), _, coords.y.toString()}),_,skLOOT.itemType,_tokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
- skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.itemType,_tokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.itemType,_tokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
skLoot.mint(skLootBagTokenInfo[_tokenId].item,skLOOT.itemType,_tokenId,_recipient) (contracts/brand/skLOOTFactory.sol#238-243)
- skLootBagTokenIdComponentsRemoved(_tokenId) (componentToBeRemoved) (contract/brand/skLOOTFactory.sol#243)
Reentrancy in StakeElementPool.slashAndBuyPool(addresses,bytes,address,uint256,bool) (contracts/banking/StakeElementPool.sol#120-132):
    slash(_stakeHouse,_memberId,_amount,_sLocketRequired) (contracts/banking/StakeElementPool.sol#120)
        - StakedHouseRegistry(_stakeHouse).kick(_memberId) (contracts/banking/StakeElementPool.sol#154)
        - buySlashedPool(_stakeHouse,_amount,_sLocketRequired) (contracts/banking/StakeElementPool.sol#155)
            - shareToken.mint(address(this),sharedToken) (contracts/banking/StakeElementPool.sol#151-154)
State variable written after the call():
buySlashedPool(_stakeHouse,_amount,_sLocketRequired) (contracts/banking/StakeElementPool.sol#151)
buySlashedPoolCurrentLot(_stakeHouse,_amount,_sLocketRequired) (contracts/banking/StakeElementPool.sol#152)
buySlashedPool(_stakeHouse,_memberId,_amount,_sLocketRequired) (contracts/banking/StakeElementPool.sol#153)
buySlashedPool(_stakeHouse,_memberId,_amount,_sLocketRequired) (contracts/banking/StakeElementPool.sol#154)
useUniversal.selectorWrittenAfterTheCall() (contracts/StakeHouseUniverse.sol#16-215):
    External call:
        - new StakeHouseUpgradeableProxy(address(_lotRecipientLogic),address(this),abi.encodePacked{}) (contracts/StakeHouseUniverse.sol#179-183)
        - buySlashedPool(new StakeHouseUpgradeableProxy(address(_brandNFT),address(this),abi.encodePacked{})) (contracts/StakeHouseUniverse.sol#179-183)
        - brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodeWithSelector(brandCentral.brandCentralLogic)).init.selector,address(this),_brandNFT,_skLootFactory,_claimAuction) (contracts/StakeHouseUniverse.sol#195-200)
useUniversal.selectorWrittenAfterTheCall() (contracts/StakeHouseUniverse.sol#200)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
AccountManager.registerValidatorInitialised(addresses,bytes,account) (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Reentrancy in StakeElementPool.slashAndBuyPool(addresses,bytes,address,uint256,bool) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradable.sol#376-397) ignores return value by ERC721Receiver.onERC721Received(_to,_from,_tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradable.sol#383-383)
ERC1967Upgrade.upgradeAndCall(addresses,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#163-172) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#179-182)
ERC1967Upgrade.upgradeAndCallSecure(addresses,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#179-187) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#179-187)
ERC1967Upgrade.upgradeAndCallSecure(addresses,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#179-187) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(addresses),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#179-187)
ERC1967Upgrade.upgradeAndCall(addresses,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(addresses),oldImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192)
ERC1967Upgrade.upgradeAndCall(addresses,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#182-192)
ERC721._checkOnERC721Received(addresses,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver.onERC721Received(_msgSender(),_from,_tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#374-386)
Reentrancy in StakeElementPool.slashAndBuyPool(addresses,bytes,address,uint256,bool) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradable.sol#376-397) ignores return value by accountManager.createStakehouse(_msgSender(),_blshPublickey,_ticker,_buildingTypeId,_ethReport) (contracts/accounts/TransactionManager.sol#76-88)
skLOOTFactory.openSkLootBagAndRemoveToken(uint256,_skLootFactory,_skLootBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-214) ignores return value by skLoot.mint(string(_abi.encodePacked{_, coords.x.toString()}),_,coords.y.toString()) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openSkLootBagAndRemoveToken(skLootTokenIdComponents,address) (contracts/brand/skLOOTFactory.sol#203-214) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.itemType,_brandNFT,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openSkLootBagAndRemoveToken(uint256,_skLootFactory,_skLootBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-214) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].character,skLOOT.itemType,_brandNFT,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
skLOOTFactory.openSkLootBagAndRemoveToken(uint256,_skLootFactory,_skLootBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-214) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].item,skLOOT.itemType,_brandNFT,_recipient) (contracts/brand/skLOOTFactory.sol#238-243)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#unused-return
```

## banking/savETH.sol

External call:

- `stakehouseRegistryMemberShareHeld`(`nameId`) (contracts/banking/SlotSettlementPool.sol#97)
- `slot.main(SLOT PER STAKEHOUSE MEMBER)` (contracts/banking/SlotSettlementPool.sol#103)

State variables written after the call(s):

- `stakehouseRegistryMemberShareHeld(stakehouse, nameId)` (contracts/banking/SlotSettlementPool.sol#104)

Reentrancy in `SlotSettlementPool.setSharesHeld`(`address, bytes, address`):

- External call:
- `new BeaconProxy(stakeHouseRegistryBeacon, abi.encodeWithSelector(StakeHouseRegistryBeacon.init.selector, address(this)))` (contracts/StakeHouseUniverse.sol#229-235)
- State variables written after the call(s):
- `registeredMemberToStakehouse[stakeHouseAddress, firstMember]` (contracts/StakeHouseUniverse.sol#424)

Reentrancy in `StakeHouseUniverse.setSharesHeld`(`address, uint256, string, bytes`):

- External call:
- `stakehouseProxy = new BeaconProxy(stakeHouseRegistryBeacon, abi.encodeWithSelector(StakeHouseRegistryBeacon.init.selector, address(this)))` (contracts/StakeHouseUniverse.sol#229-235)
- State variables written after the call(s):
- `registeredMemberToStakehouse[stakeHouseAddress, firstMember]` (contracts/StakeHouseUniverse.sol#424)

Reentrancy in `StakeHouseUniverse.setSharesHeld`(`address, bytes, address, uint256, string, bytes`):

- External call:
- `sklootFactory = new skLOOTFactory(stakeHouseRegistryBeacon, abi.encodeWithSelector(skLOOTFactory.init.selector, (bytes4)keccak256('skLOOTFactory'), x, y, z))` (contracts/staking/SkLOOTFactory.sol#120-123)
- State variables written after the call(s):
- `stakeHouse_skl00t[stakeHouse, memberId, amount, _skl00tRequired]` (contracts/staking/SkLOOTFactory.sol#129)
- `buySharesDslot(stakeHouse, memberId, amount, _slasher)` (contracts/staking/SlotSettlementPool.sol#154)
- `stakeShares[stakeHouse][memberId][address(this)].sharesFalling` (contracts/staking/SlotSettlementPool.sol#181-184)
- State variables written after the call(s):
- `stakeSharesDslot[stakeHouse][memberId][amount, _slasher]` (contracts/staking/SlotSettlementPool.sol#153)
- `stakeSharesCurrent[stakeHouse][amount]` (contracts/staking/SlotSettlementPool.sol#175)
- `stakeSharesHeld[stakeHouse][memberId][amount]` (contracts/staking/SlotSettlementPool.sol#176)

Reentrancy in `SlotSettlementPool.setSharesHeld`(`address, bytes, address, uint256)`:

- External call:
- `skl00t_main(string calldata encodedFees, (bytes4)keccak256('skl00tMain'), x, y, z)` (contracts/staking/SkLOOTFactory.sol#120-123)
- State variables written after the call(s):
- `stakeHouse_skl00t[stakeHouse, memberId, amount, _skl00tRequired]` (contracts/staking/SkLOOTFactory.sol#129)
- `buySharesDslot(stakeHouse, memberId, amount, _slasher)` (contracts/staking/SlotSettlementPool.sol#154)
- `stakeShares[stakeHouse][memberId][address(this)].sharesFalling` (contracts/staking/SlotSettlementPool.sol#181-184)
- State variables written after the call(s):
- `stakeSharesDslot[stakeHouse][memberId][amount, _slasher]` (contracts/staking/SlotSettlementPool.sol#153)
- `stakeSharesCurrent[stakeHouse][amount]` (contracts/staking/SlotSettlementPool.sol#175)
- `stakeSharesHeld[stakeHouse][memberId][amount]` (contracts/staking/SlotSettlementPool.sol#176)



## banking/savETHReservePool.sol

```
State variables written after the call(s):
    _msgSender() (contract/contracts/erc721/ERC721.sol#20)
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation/syntax-reentrancy-vulnerabilities-1

AccountManager.registerListInitialised(address, bytes, bytes, address) (contract/contracts/accountmanager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation/syntax-initialised-local-variables

ERC721Upgradable.transferFrom(address, address, uint256, bytes) (node_modules/openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradable.sol#376-397) ignores return value by IERC721Receiver.onERC721Received(_msgSender(), from,to, tokenId, data) (node_modules/openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradable.sol#350-363)
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/cryptography/keccak/SHA3.sol#10-11)
ERC1967Upgrade.upgradeToAndCall(addresses, bytes, bool) (node_modules/openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#179-187) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/openzeppelin/cryptography/keccak/SHA3.sol#10-11)
ERC1967Upgrade.upgradeToAndCall(address, bytes, bool) (node_modules/openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#187-197) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature("upgradeTo(address,bytes)",to, data)) (node_modules/openzeppelin/cryptography/keccak/SHA3.sol#10-11)
ERC1967Upgrade.upgradeToAndCall(address, bytes, bool) (node_modules/openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#197-207) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature("upgradeToAndCall(address,bytes)",to, data)) (node_modules/openzeppelin/cryptography/keccak/SHA3.sol#10-11)
ERC1967Upgrade.upgradeToAndCall(address, bytes, bool) (node_modules/openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#207-217) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature("upgradeToAndCall(address,bytes,bytes)",to, data, msgValue)) (node_modules/openzeppelin/cryptography/keccak/SHA3.sol#10-11)
ERC721._checkMinted(address, uint256) (contract/contracts/token/ERC721/ERC721.sol#186-196)
ERC721Upgradable._checkMinted(address, uint256) (node_modules/openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradable.sol#356-366)
ERC721Upgradable._checkMinted(address, uint256, bytes) (node_modules/openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradable.sol#376-397)
skLootFactory.openToTokenTransfer(uint256, skLootFactory.skbAgmComponent, address) (contracts/brand/skLootFactory.sol#201-216) ignores return value by skLoot.maint(string,abi.encodePacked((,cooks.x,toString()), ,cooks.y,toString()))
skLootFactory.openToBrandTransfer(uint256, skLootFactory.skbAgmComponent, address) (contracts/brand/skLootFactory.sol#201-216) ignores return value by skLoot.maint(string,abi.encodePacked((,cooks.x,toString()), ,cooks.y,toString()))
skLootFactory.openToBrandTransfer(uint256, skLootFactory.skbAgmComponent, address) (contracts/brand/skLootFactory.sol#221-236)
skLootFactory.openToBrandTransfer(uint256, skLootFactory.skbAgmComponent, address) (contracts/brand/skLootFactory.sol#203-246) ignores return value by skLoot.maint(skLoot.BagTokenInfo[_tokenId].building,skLoot.ItemType,_brandTokenId,_recipient) (contracts/brand/skLootFactory.sol#221-233)
skLootFactory.openToBrandTransfer(uint256, skLootFactory.skbAgmComponent, address) (contracts/brand/skLootFactory.sol#203-246) ignores return value by skLoot.maint(skLoot.BagTokenInfo[_tokenId].characters,_skLoot.ItemType,_character, _breed)
skLootFactory.openToBrandTransfer(uint256, skLootFactory.skbAgmComponent, address) (contracts/brand/skLootFactory.sol#203-246) ignores return value by skLoot.maint(skLoot.BagTokenInfo[_tokenId].breeds,_breed)
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation/syntax-return
```

## banking/savETHManager.sol

```

skLOOTFactory, skLootItemClaim(address, address,uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PENG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length] (contracts/brand/skLOOTFactory.sol#286)"  

skLOOTFactory, skLootItemClaim(address, address,uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PENG: "pickedItem = _specialLuckyDipItems[pseudoRandomNumber % _specialLuckyDipItems.length] (contracts/brand/skLOOTFactory.sol#283)"  

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PENG  

Base64 encode(Bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: nature(uint256, uint256)(resultPrv_encode_am_0 - 2,0xd3d << 240) (contracts/helpers/Base64.sol#52)  

Base64.encode(Bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: nature(uint256, uint256)(resultPrv_encode_am_0 - 1,0xd3d << 248) (contracts/helpers/Base64.sol#55)  

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup  

ERC165Upgradeable_gpe (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/ERC165Upgradeable.sol#86) shadows:  

- ContextUpgradeable_gpe (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#80)  

ERC20Permit_gpe (node_modules/@openzeppelin/contracts-upgradeable/erc20-permit/ERC20PermitUpgradeable.sol#93) shadows:  

- EIP712Upgradeable_gpe (node_modules/@openzeppelin/contracts-upgradeable/erc721/EIP712Upgradeable.sol#111)  

- ERC20Upgradeable_gpe (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#94)  

ContextUpgradeable_gpe (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#141) shadows:  

- ERC165Upgradeable_gpe (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/ERC165Upgradeable.sol#35)  

ContextUpgradeable_gpe (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#35)  

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing  

savETHReservePool.addNftForOpenPool(address,bytes,address) (contract/banking/savETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHRecipient) (contracts/banking/savETHReservePool.sol#208)  

savETHReservePool.addNftForOpenPool(address,bytes,address) (contract/banking/savETHReservePool.sol#214-238) ignores return value by saveETHToken.transfer(currentKeeper,saveETHRecipient) (contracts/banking/savETHReservePool.sol#238)  

savETHReservePool.addNftForOpenPool(address,bytes,address) (contract/banking/savETHReservePool.sol#242-249) ignores return value by saveETHToken.transferFrom(needKeeper,address(this),savETHRequiredForIsolation) (contracts/banking/savETHReservePool.sol#249)  

savETHReservePool.deposit(address,uint256) (contracts/banking/savETHReservePool.sol#307-324) ignores return value by ETHToken.transferFrom(_owner,ETHFromExchangeRate) (contracts/banking/savETHReservePool.sol#301)  

savETHReservePool.withdraw(address,uint256) (contracts/banking/savETHReservePool.sol#325-327) ignores return value by ETHToken.transferFrom(_owner,address(this)) (contracts/banking/savETHReservePool.sol#327)  

BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(may.sender,address(this),shbBidAmount) (contracts/brand/BrandCentralClaimAuction.sol#105)  

BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(may.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#105)  

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer


```

**BrandCentralClaimAuction**.shbBidConstructor(ConstructorVisible) (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:

```

BLOCKS_Per_DAY * 2448 * FEE_IN_BLOCKS (contracts/brand/BrandCentralClaimAuction.sol#127)  

-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:

```

- (address(this).balance * 1000000000000000000) / 1000000000000000000 (contracts/helpers/Base64.sol#52)

```

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#division-before-multiplication

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a dangerous strict equality:

```

- isSpecial = blockhash(block.slot) % 50 == 0 (contracts/brand/skLOOTFactory.sol#245)

```

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicket(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):

```

External calls:  

- StakeToken.mint(address,recipient,amount,shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)  

  Stake variables written after the call():  

- auction.shbbid = shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)  

  auction.bider = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)  

- auction.biddingEnd = auction.biddingEnd + Bid_EXTENSION_IN_BLOCKS (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#161-180):

```

External calls:  

- StakeToken.mint(address(this),shareToMint) (contracts/banking/SlotSettlementPool.sol#161-164)  

  Stake variables written after the call():  

- increaseCollateralBalance(shbToken,recipient,_memberId) += amount (contracts/banking/SlotSettlementPool.sol#167)  

  - stakeHouseSTTokenTotalCollateralBalance(_shbToken) += amount (contracts/banking/SlotSettlementPool.sol#40)  

  - stakeHouseSTToken.deployStakeHouseShareToken(address) (Contracts/banking/SlotSettlementPool.sol#46-48)  

- ETHTokenProxy = new BeaconProxy(ETHBeacon,abi.encodeWithSelector(ETH_STAKEHOUSE,init.selector,address(this),_stakeHouse)) (contracts/banking/SlotSettlementPool.sol#48-75)

```

State variables written after the call():

```

- increaseCollateralBalance(shbToken,recipient,_memberId) += amount (Contracts/banking/SlotSettlementPool.sol#48-75)

```

Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(address) (Contracts/banking/SlotSettlementPool.sol#46-48):

```

External calls:  

- StakeHouseUniverse.createStakeHouse(stakeHouseName,amount,amountCasted,manisize) (contracts/banking/savETHReservePool.sol#156)  

  Stake variables written after the call():  

- dTHUnderManagement += amount (contracts/banking/savETHReservePool.sol#142)

```

Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (Contracts/banking/SlotSettlementPool.sol#88-117):

```

External calls:  

- StakeHouseUniverse.createStakeHouse(stakeHouseName,amount,amountCasted,manisize) (Contracts/banking/savETHReservePool.sol#156)  

  Stake variables written after the call():  

- increaseCollateralBalance(shbToken,recipient,_memberId) += amount (Contracts/banking/SlotSettlementPool.sol#116)

```

Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (Contracts/banking/SlotSettlementPool.sol#88-117):

```

External calls:  

- StakeHouseUniverse.createStakeHouse(stakeHouseName,amount,amountCasted,manisize) (Contracts/banking/savETHReservePool.sol#156)  

  Stake variables written after the call():  

- increaseCollateralBalance(shbToken,recipient,_memberId) += amount (Contracts/banking/SlotSettlementPool.sol#116)

```

Reentrancy in StakeHouseUniverse.createStakeHouse(stakeHouseName,amount,amountCasted,manisize) (Contracts/banking/savETHReservePool.sol#156):

```

External calls:  

- StakeHouseUniverse.deployStakeHouse(stakeHouseName,amount,amountCasted,manisize) (Contracts/banking/savETHReservePool.sol#156)
  Stake variables written after the call():  

- registerMember(stakeHouse,_memberId) (Contracts/banking/StakeHouseUniverse.sol#249)  

  StakeHouse(_stakeHouse,_memberId) = stakeHouse (Contracts/banking/StakeHouseUniverse.sol#462)

```

Reentrancy in skLOOTFactory.openNftLoopholeAndRewriter(uint256,skLOOTFactory,skAggComponent,address) (Contracts/brand/skLOOTFactory.sol#203-246):

```

External calls:  

- StakeHouseRegistry.yawBeaconProxy(stakeHouseRegistry,abi.encodeWithSelector(stakeHouseRegistry.beacon,abi.encodeWithSelector(stakeHouseRegistry.beacon,init.selector,address(this)))) (Contracts/StakeHouseUniverse.sol#229-235)  

- StakeHouseUniverse.deployStakeHouse(stakeHouseName,amount,amountCasted,manisize) (Contracts/StakeHouseUniverse.sol#249)

```

State variables written after the call():

```

- registerMember(stakeHouse,_memberId) (Contracts/banking/StakeHouseUniverse.sol#249)

```

Reentrancy in skLOOTFactory.openNftLoopholeAndRewriter(uint256,skLOOTFactory,skAggComponent,address) (Contracts/brand/skLOOTFactory.sol#203-246):

```

External calls:  

- string abi.encodePacked((coordon.x.toString(),",coordon.y.toString(),)),skLOOT_JTokenType.land,brandTokenId,recipient) (Contracts/brand/skLOOTFactory.sol#221-226)  

- skLoot.mint(skLootTokenInfo[tokenId].building,skLoot.itemType,skLoot.brandTokenId,recipient) (Contracts/brand/skLOOTFactory.sol#228-233)  

  skLoot.mint(skLootTokenInfo[tokenId].character,skLoot.itemType,skLoot.brandTokenId,recipient) (Contracts/brand/skLOOTFactory.sol#235-240)

```

State variables written after the call():

```

- skLootTokenInfo[tokenId].componentRemoved = true (Contracts/brand/skLOOTFactory.sol#243)

```

Reentrancy in StakeHouseUniverse.createStakeHouse(stakeHouseName,amount,amountCasted,manisize) (Contracts/banking/savETHReservePool.sol#156):

```

External calls:  

- StakeHouseUniverse.createStakeHouse(stakeHouseName,amount,amountCasted,manisize) (Contracts/banking/savETHReservePool.sol#156)  

  Stake variables written after the call():  

- buySlashedStakeHouse(stakeHouse,amount,shbToken) (Contracts/banking/SlotSettlementPool.sol#131)  

  - stakeHouseCurrentSlashed(stakeHouse) -= amount (Contracts/banking/SlotSettlementPool.sol#175)  

- buySlashedStakeHouse(stakeHouse,_memberId,amount,shbToken) (Contracts/banking/SlotSettlementPool.sol#131)  

  - stakeHouseMemberCurrentSlashed(stakeHouse,_memberId) -= amount (Contracts/banking/SlotSettlementPool.sol#176)

```

# AUTOMATED TESTING

DRAFT

```
Reentrancy_in_StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address,address) (contracts/StakeHouseUniverse.sol#109-219):
    External calls:
        - loctFactoryProxy = new StakeHouseUpgradeableProxy(address(.locfFactoryLogic), address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
        - BNFTProxy = new StakeHouseUpgradeableProxy(address(.brandNftLogic), address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#187-191)
        - BNFTProxy = new StakeHouseUpgradeableProxy(address(.brandNftLogic), address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#197-201)
    State variable written after the call:
        - _brandNft = BNFTProxy(address(this)) (contracts/StakeHouseUniverse.sol#205)
    State variable written after the call:
        - _brandNft = BNFTProxy(address(this)) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/wlit/Detector-Documentation#reentrancy-vulnerabilities-1
AccountManager.registerAllDataInitials(address,bytes,bytes,account) (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wlit/Detector-Documentation#uninitialised-local-variable
ERC721Upgradeable._checkERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/erc721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721Receiver.onERC721Received()
    (msg.sender(),from,toKenId,data) (node_modules/@openzeppelin/contexts-upgradeable/token/ERC721/ERC721Upgradeable.sol#353-359)
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#97-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature("upgr
    adeToAndCall(address,bytes,bool)"),(node_modules/@openzeppelin/contexts-upgradeable/token/ERC721/ERC721Upgradeable.sol#109))
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bytes,bytes) (node_modules/@openzeppelin/contexts-upgradeable/token/ERC721/ERC721Upgradeable.sol#109-116) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature("upgr
    adeToAndCallSecure(address,bytes,bytes,bytes)"),(node_modules/@openzeppelin/contexts-upgradeable/token/ERC721/ERC721Upgradeable.sol#117-120))
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bytes,bytes) (node_modules/@openzeppelin/contexts-upgradeable/token/ERC721/ERC721Upgradeable.sol#117-120) ignores return value by Address.functionDelegateCall(IBeacon(newSecon
    dImplementation()),data) (node_modules/@openzeppelin/contexts-upgradeable/token/ERC721/ERC721Upgradeable.sol#121-124) ignores return value by Address.functionDelegateCall(IBeacon(newSecondImplementation()),data) (node_modul
    es/@openzeppelin/contract/token/ERC721/ERC721.sol#374-386)
TransactOnBehalfOf._mintStakehouse(bytes,string,uint16,ETHEREUM,...) (node_modules/@openzeppelin/contexts-upgradeable/token/ERC721/ERC721.sol#436-450) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,toKenId,_data) (node_modul
    es/@openzeppelin/contract/token/ERC721/ERC721.sol#451-459)
skLOOTFactory.openXLootBagAndRemoveItem(uint256,skLOOTFactory,skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked(),_coords.x,toScrip
    )),skLoot.ItemType._brand,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLOOTFactory.openXLootBagAndRemoveItem(uint256,skLOOTFactory,skBagComponent,address) (contracts/brand/skLOOTFactory.sol#228-233)
skLOOTFactory.openXLootBagAndRemoveItem(uint256,skLOOTFactory,skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootBagTokenInfo[_tokenId].building,skLOOT.ItemType._brand
    ,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#230-240)
Reference: https://github.com/crytic/slither/wlit/Detector-Documentation#unused-return
```

## brand/BrandCentral.sol

```

skLOOTFactory.skLocItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PRNG: "pickedItem = _luckyDipItems(pseudoRandomNumber % luckyDipItems.length) (contracts/brand/skLOOTFactory.sol#286)"  

skLOOTFactory.skLocItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length] (contracts/brand/skLOOTFactory.sol#286)"  

skLocItemClaim(address) (contracts/helpers/Base64.sol#12-42) contains an incorrect shift operation: nature(uint256, uint256) resultFor_encode_am_0 - 2,0xd3 <> 240) (contracts/helpers/Base64.sol#52)  

skLocItemClaim(address) (contracts/helpers/Base64.sol#12-42) contains an incorrect shift operation: nature(uint256, uint256) resultFor_encode_am_0 - 1,0xd3 <> 240) (contracts/helpers/Base64.sol#55)  

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-prngs

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-42) contains an incorrect shift operation: nature(uint256, uint256) resultFor_encode_am_0 - 2,0xd3 <> 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-42) contains an incorrect shift operation: nature(uint256, uint256) resultFor_encode_am_0 - 1,0xd3 <> 240) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC165Upgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/tokens/ERC20/ERC165Upgradable.sol#86) shadows:  

- ContextUpgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/tokens/ContextUpgradeable.sol#30)
ERC165Upgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/introspection/ERC165Upgradable.sol#93) shadows:  

- ERC165Upgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/tokens/ERC20/ERC165Upgradable.sol#41)
- ERC20Upgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/tokens/ERC20/ERC20Upgradable.sol#94)
ContextUpgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/introspection/ContextUpgradeable.sol#93) shadows:  

- ERC165Upgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/introspection/ERC165Upgradable.sol#35)
- ContextUpgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/introspection/ContextUpgradeable.sol#80)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

saveETHReservePool.addNonceToOpenPool(address,bytes,address) (contracts/banking/saveETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/saveETHReservePool.sol#208)
saveETHReservePool.addNonceToOpenPool(address,bytes,address) (contracts/banking/saveETHReservePool.sol#214-235) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangeRate) (contracts/banking/saveETHReservePool.sol#235)
saveETHReservePool.deposit(address,uint256) (contracts/banking/saveETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(newKeeper,Address(this),saveETHRequiresForIsolation) (contracts/banking/saveETHReservePool.sol#270)
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.transfer(_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(mag.sender,auction.ehbBid) (contracts/brand/BrandCentralClaimAuction.sol#109)
BrandCentralClaimAuction.claimHb(uint256) (contracts/brand/BrandCentralClaimAuction.sol#108-199) ignores return value by shbToken.transfer(mag.sender,auction.ehbBid) (contracts/brand/BrandCentralClaimAuction.sol#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-branch-for

```

```

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#47)  

-TOTAL_AUCTION_LENGTH_IN_BLOCKS * BLOCKS_PER_DAY * SECONDS_PER_DAY / SECONDS_PER_HOUR / SECONDS_PER_DAY / DAVIS (contracts/brand/BrandCentralClaimAuction.sol#29)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-42) contains an incorrect shift operation on the result of a division:  

-encodes = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#7)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLocItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a dangerous strict equality:  

- isSpecial == blockNumber() % 50 == 0 (contracts/brand/skLOOTFactory.sol#245)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#strict-equality

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
External calls:  

- shbToken.transfer(auction.bidder,auction.ehbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
State variables written after the call(s):  

- auction.bidder = msg.sender (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)
Reentrancy in StakeHouseUniverse.buyStakeHouse(address,bytes,address,uint256,address) (contracts/banking/StakeHouseUniverse.sol#161-180):
External calls:  

- shareToken.mint(address(stakeHouse),recipient,_memberId,sharesMint) (contracts/banking/SlotSettlementPool.sol#181-184)
State variables written after the call(s):  

- _increaseCollateralisedBalance(address(stakeToken),recipient,_memberId,sharesMint) (contracts/banking/SlotSettlementPool.sol#187)
- stakeHouseTotalStake(stakeHouse,_stakeToken) += amount (contracts/banking/CollateralisedStakeManager.sol#40)
Reentrancy in StakeHouseUniverse.deployStakeHouseToken(addresses) (contracts/banking/StakeHouseUniverse.sol#46-49):
External calls:  

- aETHTProxy = new BeaconProxy(aETHTBeacon,abi.encodeWithSelector(aETHT.aInit.selector,address(this),stakeHouse)) (contracts/banking/StakeHouseUniverse.sol#48-51)
- stakeHouseShareTokens[_stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#80)
Reentrancy in saveETHReservePool.mintETHReserves(address,bytes,uint256) (contracts/banking/saveETHReservePool.sol#111-145):
External calls:  

- saveETHToken.mint(address(this),saveETHAmountCalculated,mantissa) (contracts/banking/saveETHReservePool.sol#116)
State variables written after the call(s):  

- ORWNodeManagement += amount (contracts/banking/saveETHReservePool.sol#43)
Reentrancy in StakeHouseUniverse.mintStakeHouseBatch(address,bytes,address) (contracts/banking/StakeHouseUniverse.sol#88-117):

```

```

External calls:  

- universe.recordMemberSlotShareMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
- universe.recordMemberSlotShareBurned(_memberId) (contracts/banking/SlotSettlementPool.sol#103)
State variables written after the call(s):  

- stakeHouseTotalStake[_stakeHouse] += SLOT PER STAKEHOUSE MEMBER (contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in StakeHouseUniverse.mintLOTAndSharesBatch(addresses,bytes,address) (contracts/banking/StakeHouseUniverse.sol#88-117):
External calls:  

- universe.recordMemberSlotShareMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#97)
- universe.recordMemberSlotShareBurned(_memberId) (contracts/banking/SlotSettlementPool.sol#103)
- stakeToken.mint(address(this),shareToSendRecipient) (contracts/banking/SlotSettlementPool.sol#110)
State variables written after the call(s):  

- _increaseCollateralisedBalance(address(stakeToken),recipient,_memberId,shareToSendRecipient) (contracts/banking/SlotSettlementPool.sol#114)
- stakeHouseTotalStake(stakeHouse,_stakeToken) += amount (contracts/banking/CollateralisedStakeManager.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/banking/StakeHouseUniverse.sol#218-265):
External calls:  

- stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,abi.encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
- slotSettlementPool.deploy(stakeHouseProxy,slotSettlementPool,slotSettlementPool.address) (contracts/StakeHouseUniverse.sol#249)
- registeredMembers[stakeHouse][stakeHouseAddress][firstMember] (contracts/StakeHouseUniverse.sol#428)
- memberKnots[stakeHouse][stakeHouseAddress][firstMember] = _stakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLocItemClaim().openLockComponent(skLocItemClaim): (contracts/brand/skLOOTFactory.sol#203-246):
External calls:  

- skLocItemClaim._stakeHouse,_memberId,_amount,_skLocItemClaimRequired((coord.x,toString()),coord.y,toString(),),skLocItemClaim._recipient) (contracts/brand/skLOOTFactory.sol#221-226)
- skLocItemClaim._stakeHouse,_memberId,_amount,_skLocItemClaimRequired((coord.x,toString()),coord.y,toString(),),skLocItemClaim._recipient) (contracts/brand/skLOOTFactory.sol#242-253)
- skLocItemClaim._stakeHouse,_memberId,_amount,_skLocItemClaimRequired((coord.x,toString()),coord.y,toString(),),skLocItemClaim._recipient) (contracts/brand/skLOOTFactory.sol#250-259)
State variables written after the call(s):  

- buySlashedSlot(_stakeHouse,_memberId,_amount,_skLocItemClaimRequired) (contracts/banking/SlotSettlementPool.sol#171)
- buySlashedSlot(_stakeHouse,_memberId,_amount,_skLocItemClaimRequired) (contracts/banking/SlotSettlementPool.sol#175)
- buySlashedSlot(_stakeHouse,_memberId,_amount,_skLocItemClaimRequired) (contracts/banking/SlotSettlementPool.sol#177)
- buySlashedSlot(_stakeHouse,_memberId,_amount,_skLocItemClaimRequired) (contracts/banking/SlotSettlementPool.sol#179)
Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):

```



## brand/BrandNFT.sol

```

skLOOTFactory.skLocItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _specialLuckyDipGems(pseudoRandomNumber % _specialLuckyDipGems.length)" (contracts/brand/skLOOTFactory.sol#28)
skLOOTFactory.skLocItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a weak PRNG: "pickedItem = _luckyDipItems(pseudoRandomNumber % _luckyDipItems.length)" (contracts/brand/skLOOTFactory.sol#28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Base64 encode(Bytes) (contract/Helpers/Base64.sol#12-62) contains an incorrect shift operation: returnPrx_ecnode_am_0 - 2^(8*8*8 << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(Bytes) (contract/Helpers/Base64.sol#12-62) contains an incorrect shift operation: returnPrx_ecnode_am_0 - 1^(8*8*8 << 248) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ERC20/ERC20Upgradeable.sol#86) shadows:
- ERC20Upgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMath/ERC20SafeMath.sol#10)
- ERC20Upgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMath/ERC20SafeMath.sol#11)
- ContextUpgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#80)
ERC721Pausable._gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Pausable.sol#86):
- ERC165Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#25)
- ContextUpgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#80)

saveETHReservePool.addNonceToOpenPool(address,bytes,address) (contract/banking/saveETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/saveETHReservePool.sol#108)
saveETHReservePool.addNonceToOpenPool(address,bytes,address) (contract/banking/saveETHReservePool.sol#191-211) ignores return value by dETHToken.transfer(currentKeeper,dETHToSend) (contracts/banking/saveETHReservePool.sol#235)
saveETHReservePool.withdrawFromOpenPool(address,bytes,address) (contract/banking/saveETHReservePool.sol#191-211) ignores return value by saveETHToken.transferFrom(newKeeper,address(this),saveETHRequiredForIsolation) (contracts/banking/saveETHReservePool.sol#108)
saveETHReservePool.withdraw(address,uint256) (contract/banking/saveETHReservePool.sol#285-304) ignores return value by saveETHToken.transfer(owner,dETHFromExchangeRate) (contracts/banking/saveETHReservePool.sol#301)
saveETHReservePool.deposit(address,uint256) (contract/banking/saveETHReservePool.sol#307-329) ignores return value by dETHToken.transferFrom(_owner,address(_this),amount) (contracts/banking/saveETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicket(string,uint16) (contract/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transfer(auction.bidder,auction.address(shbBid)) (contracts/brand/BrandCentralClaimAuction.sol#125)
BrandCentralClaimAuction.claimHbb(uint256) (contract/brand/BrandCentralClaimAuction.sol#109-169) ignores return value by shbToken.transfer(mag.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfers

BrandCentralClaimAuction._bidForTicket(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
    REVERSE = 4 ** 16 - 1 // REVERSE FOR BLOCKS = 4 ** 16 - 1 // REVERSE FOR DAVIS (contracts/brand/BrandCentralClaimAuction.sol#12)
    -TOTAL AUCTION LENGTH IN BLOCKS = BLOCKS PER DAY * AUCTION LENGTH IN DAYS (contracts/brand/BrandCentralClaimAuction.sol#28)
Base64.encode(Bytes) (contract/Helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
    -encodedData = 4 ** (16 + 2) // (contracts/Helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory._skLocItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-293) uses a dangerous strict equality:
    ticketId = blockhash(blockId) % 50 == 0 (contracts/brand/skLOOTFactory.sol#245)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicket(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
    External calls:
        - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
        - auction.shbBid = shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
        - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)
        - auction.biddingEnd = auction.biddingEnd + Bid EXTENSION IN BLOCKS (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#161-191):
        - shareToken.name(address(this),shareToMint) (contracts/banking/SlotSettlementPool.sol#161-169)
        - State variables written after the call():
            - increaseCollateralBalance(shbToken,recipient,_memberId,shareToMint) (contracts/banking/CollateralizedLotManager.sol#107)
            - stakehouseTotalCollateralBalance(shbToken) += amount (Contracts/banking/CollateralizedLotManager.sol#40)
    Reentrancy in SlotSettlementPool.deployStakehouseShareToken(address) (contracts/banking/SlotSettlementPool.sol#164-164):
        - External call:
            - shbToken.mint(address(this),amountScaled.mantissa) (contracts/banking/saveETHReservePool.sol#136)
        - State variables written after the call():
            - stakehouseShareTokens[stakehouse] = token (Contracts/banking/SlotSettlementPool.sol#80)
    Reentrancy in StakeHouseUniverse.mintSLOTAndSharesBatch(address,bytes,uint256) (contracts/banking/SlotSettlementPool.sol#113-145):
        - External calls:
            - saveETHToken.mint(address(this),amountScaled.mantissa) (contracts/banking/saveETHReservePool.sol#136)
        - State variables written after the call():
            - stakehouseTotalCollateralBalance(shbToken) += amount (Contracts/banking/SlotSettlementPool.sol#114)
    Reentrancy in StakeHouseUniverse.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#116-117):
        - External calls:
            - shbToken.receives(slashedAndShyBolt.shareToMint,_memberId) (contracts/banking/SlotSettlementPool.sol#197)
            - slot.name(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
        - State variables written after the call():
            - stakehouseTotalCollateralBalance(shbToken,PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#104)
    Reentrancy in StakeHouseUniverse.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
        - External calls:
            - shbToken.receives(slashedAndShyBolt.shareToMint,_memberId) (contracts/banking/SlotSettlementPool.sol#197)
            - slot.name(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
            - shareToken.mint(_recipient,shareToSendOrManage) (contracts/banking/SlotSettlementPool.sol#110)
            - stakehouseTotalCollateralBalance(shbToken,PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#115)
        - State variables written after the call():
            - increaseCollateralBalance(shbToken,_recipient,_memberId,shareToSendOrManage) (contracts/banking/SlotSettlementPool.sol#116)
    Reentrancy in StakeHouseUniverse.newDebtToken(address,uint256,string,bytes) (contracts/banking/SlotSettlementPool.sol#124-126):
        - External calls:
            - stakehouseRegistry.mintDebtToken(shbToken,recipient,_memberId,amount) (Contracts/banking/StakeHouseRegistry.sol#235)
        - State variables written after the call():
            - _registeredMembers[stakehouse](stakehouseAddress) = registeredMember (Contracts/banking/StakeHouseUniverse.sol#49)
    Reentrancy in skLOOTFactory.openToLocLeapAndDemovetle(uint256,skLOOTFactory.skhapComponent,adresses) (contracts/brand/skLOOTFactory.sol#203-246):
        - External call:
            - stringabi.encodeWithFixedBytes((coorder.x,toString()),,coorder.y,toString(),)).skLOOT_itemType,land,brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
        - skLoc.mint(skLocTokenInfo_.tokenId).building,skLoc.itemType,Building,brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#228-233)
        - skLoc.mint(skLocTokenInfo_.tokenId).character,skLoc.itemType,Character,brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
        - skLocBuyTokenInfoComponentRemoveTokenId_.componentId != true (contracts/brand/skLOOTFactory.sol#243)
    Reentrancy in SlotSettlementPool.slashAndShyBolt(address,bytes,address,uint256,Bool) (contracts/banking/SlotSettlementPool.sol#120-132):
        - iAbn_.Stakehouse,_memberId,_amount,_lockRequired) (contracts/banking/SlotSettlementPool.sol#128)
        - StakehouseRegistry._Stakehouse._id(_memberId) (contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot._slotId(_slotId).shyBoltId (Contracts/banking/SlotSettlementPool.sol#151)
        - stakehouseTotalCollateralBalance(shbToken) -= _amount (Contracts/banking/SlotSettlementPool.sol#154)
        - buySlashedSlot._stakHouse._memberId,_amount,_shaker) (Contracts/banking/SlotSettlementPool.sol#175)
        - buySlashedSlot._stakHouse._memberId,_amount,_shaker) (Contracts/banking/SlotSettlementPool.sol#176)
        - stakehouseMemberCurrentSlotSlashed._stakeHouse._memberId -= _amount (Contracts/banking/SlotSettlementPool.sol#176)

```



## brand/skLOOT.sol

```

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length]" (contracts/brand/skLOOTFactory.sol#253)
skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length]" (contracts/brand/skLOOTFactory.sol#256)

Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(resultPtr, encode_asm_0 - 2, 0x8d3d << 240) (contracts/helpers/Base64.sol#52)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) contains an incorrect shift operation: mstore(resultPtr, encode_asm_0 - 1, 0x8d << 240) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/solidity/wiki/Detector-Documentation#shift-parameter-mixup

ERC165Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/contracts/upgradeable/tokens/ERC20/ERC165Upgradeable.sol#86) shadows:
- ContextUpgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#80)
ERC20Permit._gap (node_modules/@openzeppelin/contracts-upgradeable/contracts/upgradeable/tokens/erc20/ERC20Permit.sol#93) shadows:
- EIP712Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/EIP712Upgradeable.sol#111)
- ERC20Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/tokens/ERC20/ERC20Upgradeable.sol#96)
ContextUpgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/contracts/upgradeable/tokens/ERC721/ERC721Upgradeable.sol#141) shadows:
- ERC165Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
ContextUpgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/contracts/upgradeable/utils/context/Upgradeable.sol#80)
Reference: https://github.com/crytic/solidity/wiki/Detector-Documentation#state-variable-shadowing

saveETHReservePool.addEthToOpenPool(address,bytes,address) (contracts/banking/saveETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHPending) (contracts/banking/saveETHReservePool.sol#80)
saveETHReservePool.addEthToOpenPool(address,bytes,address) (contracts/banking/saveETHReservePool.sol#214-235) ignores return value by saveETHToken.transfer(currentKeeper,saveETHPending) (contracts/banking/saveETHReservePool.sol#235)
saveETHReservePool.deposit(address,bytes,address) (contract/banking/saveETHReservePool.sol#142-149) ignores return value by saveETHToken.transferFrom(_newKeeper,address(this),saveETHRequiredForIsolation) (contracts/banking/saveETHReservePool.sol#142)
saveETHReservePool.withdraw(address,uint256) (contracts/banking/saveETHReservePool.sol#285-304) ignores return value by saveETHToken.transfer(_owner,ETHReserveExchangeRate) (contracts/banking/saveETHReservePool.sol#301)
saveETHReservePool.deposit(address,uint256) (contracts/banking/saveETHReservePool.sol#307-329) ignores return value by ETHReserve.transferFrom(_owner,address(this)) (contracts/banking/saveETHReservePool.sol#327)
BaseCentralClaimAuction.claimShb(uint256) (contracts/brand/BaseCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(mag.sender,address(this),shbBalance) (contracts/brand/BaseCentralClaimAuction.sol#105)
BaseCentralClaimAuction.claimShb(uint256) (contracts/brand/BaseCentralClaimAuction.sol#109-199) ignores return value by shbToken.transfer(mag.sender,auction.shbBid) (contracts/brand/BaseCentralClaimAuction.sol#105)
Reference: https://github.com/crytic/solidity/wiki/Detector-Documentation#unchecked-transfer

BrandCentralClaimAuction._slashedContract(keccak256(variable)) (contracts/brand/BaseCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
-BLOCKS PER DAY = 5460 / SECONDS PER BLOCK (contracts/brand/BaseCentralClaimAuction.sol#27)
-TOTAL AUCTION LENGTH IN BLOCKS = BLOCKS PER DAY * AUCTION_LENGTH_IN_DAYS (contracts/brand/BaseCentralClaimAuction.sol#28)
Base64.encode(bytes) (contracts/helpers/Base64.sol#12-62) performs a multiplication on the result of a division:
- (bytes.length - 1) * 256 / 8 = bytes.length * 32 (contracts/helpers/Base64.sol#27)
Reference: https://github.com/crytic/solidity/wiki/Detector-Documentation#divide-before-multiply

skLOOTFactory.skLootItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a dangerous strict equality:
- isSpecial = blockhash(blockNumber) % 50 == 0 (contracts/brand/skLOOTFactory.sol#253)
Reference: https://github.com/crytic/solidity/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BaseCentralClaimAuction.sol#109-161):
External calls:
- auction.bidForTicker(bidder,option,shbBid) (contracts/brand/BaseCentralClaimAuction.sol#125)
State variables written after the call():
- auction.shbBid = shbBidAmount (contracts/brand/BaseCentralClaimAuction.sol#128)
- auction.auctionId = auctionId + 1 (contracts/brand/BaseCentralClaimAuction.sol#131)
- auction.biddingEnd = auction.biddingEnd + 1 (contracts/brand/BaseCentralClaimAuction.sol#134)
- auction.biddingEnd = auction.biddingEnd + 1 (BLOCKS PER DAY * EXTENSION_IN_BLOCKS) (contracts/brand/BaseCentralClaimAuction.sol#152)
Reentrancy in StakePool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#146-150):
External calls:
- shareToken.mint(address(this),shareToMint) (contracts/banking/SlotSettlementPool.sol#146-148)
State variables written after the call():
- _stakehouseTotalCollateralBalance[_stakehouse] += amount (contracts/banking/SlotSettlementPool.sol#40)
- stakehouseTotalCollateralBalance[_stakehouse] += amount (contracts/banking/SlotSettlementPool.sol#44)
Reentrancy in StakePool.deployStakeHouseShareToken(addresses) (contracts/banking/SlotSettlementPool.sol#66-75):
External calls:
- stETHProxy = new BeaconProxy(stETHBeacon,shb).encodeWithSelector(stETHBeacon.init.selector,address(this),stakeHouse) (contracts/banking/SlotSettlementPool.sol#68)
State variables written after the call():
- _stakehouseTotalCollateralBalance[_stakehouse] += amount (contracts/banking/SlotSettlementPool.sol#75)
Reentrancy in saveETHReservePool.mintETHReserves(address,bytes,uint256) (contracts/banking/saveETHReservePool.sol#113-145):
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():
- stakehouseTotalSlot1stStakeHouse += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in StakePool.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)
- stakehouseTotalSlot1stStakeHouse += SLOT_PER_STAKEHOUSE_MEMBER (contracts/banking/SlotSettlementPool.sol#104)
shareToken.mint(_stakehouse,amount) (contracts/banking/SlotSettlementPool.sol#110)
shareToken.mint(_stakehouse,amount) (contracts/banking/SlotSettlementPool.sol#115)
State variables written after the call():
- _stakehouseTotalCollateralBalance[_stakehouse] += amount (contracts/banking/SlotSettlementPool.sol#40)
Reentrancy in StakeHouseUniverse.newStakeHouse(address,uint256,string,bytes) (contracts/stakeHouseUniverse.sol#210-265):
External calls:
- stETHProxy = new BeaconProxy(stakeHouseRegistryBeacon,shb).encodeWithSelector(stakeHouseRegistry(stakeHouseRegistryAddress).init.selector,address(this)) (contracts/StakeHouseUniverse.sol#229-235)
- slotSettlementPool.deployStakeHouseShareToken(stakeHouseAddress) (contracts/StakeHouseUniverse.sol#245)
State variables written after the call():
- _stakehouseTotalSlot1stStakeHouse += stakeHouseTotalSlot1stStakeHouse (contracts/StakeHouseUniverse.sol#428)
- memberCountOfStakeHouse(_memberId) = stakeHouseTotalSlot1stStakeHouse (contracts/StakeHouseUniverse.sol#462)
Reentrancy in skLOOTFactory.openLockBagAndRemoveIt(uint256,skLOOTFactory.skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246):
External calls:
- skLOOT.mint(string(shb.encodePacked((coode.x.toString(),)),skLOOT.ItemType.sLAND,brandTokenId,recipient) (contracts/brand/skLOOTFactory.sol#221-226)
- skLOOT.mint(skLootItemTokenInfo_.tokenId).building,skLOOT.itemType_,Building.brandTokenId_,recipient) (contracts/brand/skLOOTFactory.sol#228-230)
- skLOOT.mint(skLootItemTokenInfo_.tokenId).character,skLOOT.itemType_,Character.brandTokenId_,recipient) (contracts/brand/skLOOTFactory.sol#235-240)
State variables written after the call():
- skLOOTBuyTokenIdComponentRemoved[_tokenId] = true (contracts/brand/skLOOTFactory.sol#243)
Reentrancy in StakePool.buySlashedSlot(address,bytes,uint256,Bool) (contracts/banking/SlotSettlementPool.sol#120-132):
External calls:
- slash_stakeHouse,_memberId,_amount,_isACKRequired) (contracts/banking/SlotSettlementPool.sol#128)
- stakeHouseTotalSlot1stStakeHouse -= amount (contracts/banking/SlotSettlementPool.sol#134)
- buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
- shareToken.mint(address(this),shareToMint) (contracts/banking/SlotSettlementPool.sol#135-138)
- buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#138)
- stakeHouseCurrentSlotSlashed[_stakeHouse] -= amount (contracts/banking/SlotSettlementPool.sol#175)
- buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#175)
- stakeHouseCurrentSlotSlashed[_stakeHouse] -= amount (contracts/banking/SlotSettlementPool.sol#176)

```

# AUTOMATED TESTING

DRAFT

```
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)
        - EMTFFactory = new StakeHouseUpgradeableProxy(_brandEMTFLogic,abi.encodePacked(this),abi.encodePacked("0x0000000000000000000000000000000000000000")) (contracts/StakeHouseUniverse.sol#187-191)
        - useUniversal = new StakeHouseUpgradeableProxy(_universalLogic,abi.encodePacked(this),abi.encodeWithSelector(_universal.selector,address(this)),_brandNFTs,_skLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#193-205)
        State variable mutation after the call(s):
            - _brandCentral = _brandCentral(_brandCentralAddress) (contracts/StakeHouseUniverse.sol#200)
    Reference: https://github.com/crytic/solidity-Detector-Documentation#reentrancy-vulnerabilities-1
AccountManager.registerValidatorInitial(addresses,bytes,bytes) (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/solidity-Detector-Documentation#uninitialized-local-variables
ERC721Upgradable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/tokens/ERC721/ERC721Upgradable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,to tokenId,data) (node_modules/@openzeppelin/contracts-upgradeable/tokens/ERC721/ERC721Upgradable.sol#393-395)
ERC1967Upgrade._upgradeAndCall(address,bytes,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/Upgrade.sol#163-172) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#179-187) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature('upgradeTo(address)',newImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#187-190)
ERC1967Upgrade._upgradeAndCall(address,bytes,bytes,abi.encodeWithSignature('upgradeTo(address)',newImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#187-190)
ERC1967Upgrade._upgradeAndCall(address,bytes,bytes,abi.encodeWithSignature('upgradeTo(address)',newImplementation)) (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#192-194)
ERC1967Upgrade._upgradeAndCall(address,bytes,bytes,abi.encodeWithSignature('upgradeTo(address)',newImplementation)) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#197-200)
Treasurer._mintLootBagAndRemoveItem(uint256,skLOOTFactory,skBagComponent,address) (contracts/accounts/TransactionManager.sol#77-99) ignores return value by accountManager.createSkLootToken(msg.sender,tokenId,skLootBag,skBagComponent,skBagComponent.address) (contracts/brand/skLOOTFactory.sol#220-233)
skLOOTFactory._openSkLootBagAndRemoveItem(uint256,skLOOTFactory,skBagComponent,address) (contracts/brand/skLOOTFactory.sol#220-246) ignores return value by skLoot.mint(string(abi.encodePacked({_coords.x.toSafmin()},{_coords.y.toIntString()}))) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#221-226)
skLOOTFactory._mintLootBagInfo(uint256,skLootBagTokenInfo,skLootBagComponent,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#220-233)
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#220-246)
Reference: https://github.com/crytic/solidity-Detector-Documentation#unused-return
```

brand/skLOOTFactory.sol

skLOOTFactory.skLOOTClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#235-259) uses a **west FENS** \*pickedItem = luckyNumber % luckyRippled.length (contracts/brand/skLOOTFactory.sol#286)\*

skLOOTFactory.skLOOTClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#235-259) uses a **west FENS** \*pickedItem = \_luckyNumber % luckyRippled.length (contracts/brand/skLOOTFactory.sol#286)\*

Basic64.encodeByres((contract/Helpers/Base64.sol#12-62)) contains an **incorrect shift operation**: metore.uint256 << 240 (contract/Helpers/Base64.sol#52)

Basic64.encodeByres((contract/Helpers/Base64.sol#12-62)) contains an **incorrect shift operation**: metore.uint256 << 240 (contract/Helpers/Base64.sol#52)

Reference: https://github.com/crytic/silcher/wiki/Detector-Documentation#weak-FENS

ERC20Upgradeable.\_gap (node\_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#161) shadows:

- Context.Upgradeable.\_gap (node\_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#150)

ERC20Upgradeable.\_gap (node\_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#160) shadows:

- Context.Upgradeable.\_gap (node\_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#151)

ERC20Upgradeable.\_gap (node\_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#161) shadows:

- Context.Upgradeable.\_gap (node\_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#152)

ERC20Upgradeable.\_gap (node\_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#162) shadows:

- Context.Upgradeable.\_gap (node\_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#153)

ERC20Upgradeable.\_gap (node\_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#163) shadows:

- Context.Upgradeable.\_gap (node\_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#154)

Reference: https://github.com/crytic/silcher/wiki/Detector-Documentation#weak-variable-shadowing

avETHTReservePool.addLootToPool(address,bytes,address) (contracts/banking/avETHTReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/avETHTReservePool.sol#208)

avETHTReservePool.editorLootToPoolWithAddress(address,bytes,address) (contracts/banking/avETHTReservePool.sol#212-239) ignores return value by dETHToken.transfer(currentKeeper,dETHTokenChangeOwner) (contracts/banking/avETHTReservePool.sol#208)

avETHTReservePool.isolateLootFromPoolWithAddress(address,bytes,address) (contracts/banking/avETHTReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(newkeeper,address(this),avETHTRequiredForIsolation) (contracts/banking/avETHTReservePool.sol#208)

avETHTReservePool.setDeposit(address,uint256) (contracts/banking/avETHTReservePool.sol#285-304) ignores return value by dETHToken.transferFrom(\_owner,address(this),amount) (contracts/banking/avETHTReservePool.sol#201)

BrandCentralClaimAction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAction.sol#109-161) ignores return value by sbhToken.transferFrom(auction.bidder,auction.sbbid) (contracts/brand/BrandCentralClaimAction.sol#125)

BrandCentralClaimAction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAction.sol#110-163) ignores return value by sbhToken.transferFrom(may.sender,address(this),sbhBalance) (contracts/brand/BrandCentralClaimAction.sol#125)

BrandCentralClaimAction.claimSBH(uint256) (contracts/brand/BrandCentralClaimAction.sol#168-199) ignores return value by sbhToken.transfer(may.sender,auction.sbbid) (contracts/brand/BrandCentralClaimAction.sol#168)

Reference: https://github.com/crytic/silcher/wiki/Detector-Documentation#unchecked-transfers

BrandCentralClaimAction.claimSBH(uint256) (contracts/brand/BrandCentralClaimAction.sol#168-199) performs a multiplication on the result of a division:  
BLOCKS PER DAY \* (BLOCKS PER DAY / SECONDS PER BLOCK) \* SECONDS PER DAY / (BLOCKS PER DAY \* SECONDS PER BLOCK)

- TOTAL AUCTION LENGTH IN BLOCKS - BLOCKS PER DAY \* AUCTION LENGTH IN DAYS (contracts/brand/BrandCentralClaimAction.sol#28)

Base64.encodeByres((contract/Helpers/Base64.sol#12-62)) performs a multiplication on the result of a division:  
-encodes = 4 \* ((len + 2) / 3) (contracts/Helpers/Base64.sol#17)

Reference: https://github.com/crytic/silcher/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BrandCentralClaimAction.bidForTicker(string,uint256) (contract/brand/BrandCentralClaimAction.sol#109-161):

- External calls:
  - sbhToken.transfer(auction.bidder,auction.sbbid) (contract/brand/BrandCentralClaimAction.sol#125)
  - auction.sbbid = sbhBalanceOf (contract/brand/BrandCentralClaimAction.sol#128)
  - auction.bidder = msg.sender (contract/brand/BrandCentralClaimAction.sol#129)
  - auction.bidder = auction.bidder.giveD\_BID\_EXTENSION\_IN\_BLOCKS (contract/brand/BrandCentralClaimAction.sol#152)
  - auction.bidder = auction.bidder.giveD\_BID\_EXTENSION\_IN\_BLOCKS (contract/brand/BrandCentralClaimAction.sol#153)

Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):

- External calls:
  - sbhToken.mint(address(this),shareTotalMin) (contract/banking/SlotSettlementPool.sol#111-184)

State variables written after the call(s):

- increaseCollateralizedBalance(uintToCheck) (contract/banking/CollateralizedSlotManager.sol#40)

Reentrancy in SlotSettlementPool.deploySharesOnToken(addresses) (contracts/banking/SlotSettlementPool.sol#6-8):

- External calls:
  - ethToken.mint(address(this),shareTotalMin) (contract/banking/SlotSettlementPool.sol#111-145)

State variables written after the call(s):

- stakeHouseShares[stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#180)

Reentrancy in StakeHouseRegistry.selectStakeHouse(stakeHouse) (contracts/banking/SlotSettlementPool.sol#113-145):

- External calls:
  - saveETHToken.mint(address(this),shareTotalHouseSelected) (contract/banking/avETHTReservePool.sol#113-145)

State variables written after the call(s):

- stakeHouseShares[stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#180)

Reentrancy in StakeHouseRegistry.selectStakeHouse(stakeHouse) (contracts/banking/SlotSettlementPool.sol#113-145):

- External calls:
  - saveETHToken.mint(address(this),shareTotalHouseSelected).manisize (contract/banking/avETHTReservePool.sol#113)

State variables written after the call(s):

- stakeHouseShares[stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#180)

Reentrancy in StakeHouseRegistry.selectStakeHouse(stakeHouse) (contracts/banking/SlotSettlementPool.sol#113-145):

- External calls:
  - stakeHouseShares[stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#180)

State variables written after the call(s):

- stakeHouseShares[stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#180)

External calls:

- sbhToken.mint(stakeHouse,shareSharesSelected) (contract/banking/SlotSettlementPool.sol#117)

Reentrancy in StakeHouseRegistry.selectStakeHouse(stakeHouse) (contracts/banking/SlotSettlementPool.sol#113-145):

- External calls:
  - universe.recordDebt(stakehouseSelected,memberId) (contract/banking/SlotSettlementPool.sol#117)
  - sbhToken.mint(stakeHouse,shareSharesSelected) (contract/banking/SlotSettlementPool.sol#117)
  - shareToken.mint(address(this),shareRecipient.sendToRecipient) (contract/banking/SlotSettlementPool.sol#110)
  - shareToken.mint(address(this),shareRecipient.sendToRecipient) (contract/banking/SlotSettlementPool.sol#111)

State variables written after the call(s):

- increaseCollateralizedBalance(address/shareToken\_,recipient,memberId,shareRecipient.sendToRecipient) (contract/banking/CollateralizedSlotManager.sol#40)

Reentrancy in StakeHouseUniverse.selectStakeHouse(stakeHouse,bytes,address,uint256) (contracts/StakeHouseUniverse.sol#26-265):

- External calls:
  - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,byte).encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this)) (contracts/StakeHouseUniverse.sol#229-235)

State variables written after the call(s):

- stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,byte).encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this)) (contracts/StakeHouseUniverse.sol#229-235)

Reentrancy in StakeHouseRegistry.selectStakeHouse(stakeHouse,bytes,address,uint256) (contracts/StakeHouseUniverse.sol#26-265):

- External calls:
  - stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,byte).encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this)) (contracts/StakeHouseUniverse.sol#229-235)

State variables written after the call(s):

- stakeHouseProxy = new BeaconProxy(stakeHouseRegistryBeacon,byte).encodeWithSelector(StakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this)) (contracts/StakeHouseUniverse.sol#229-235)

Reentrancy in StakeHouseRegistry.selectStakeHouse(stakeHouse,bytes,address,uint256) (contracts/StakeHouseUniverse.sol#26-265):

- External calls:
  - sbhToken.mint(string,address encodeWithSelector((contract/banking/SlotSettlementPool.sol#117),#skLOOT\_type).send,memberId,recipient) (contracts/brand/skLOOTFactory.sol#221-224)
  - sbhToken.mint(string,address encodeWithSelector((contract/banking/SlotSettlementPool.sol#117),#skLOOT\_type).send,memberId,recipient) (contracts/brand/skLOOTFactory.sol#228-239)
  - sbhToken.mint(string,address encodeWithSelector((contract/banking/SlotSettlementPool.sol#117),#skLOOT\_type).send,memberId,recipient) (contracts/brand/skLOOTFactory.sol#235-240)

State variables written after the call(s):

- stakeHouseShares[stakeHouse].memberId = memberId (contract/banking/SlotSettlementPool.sol#117)

Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,address,uint256,byte) (contracts/banking/SlotSettlementPool.sol#120-132):

- External calls:
  - slashToken.mint(stakeHouse,memberId,\_sbhTicketRequired) (contract/banking/SlotSettlementPool.sol#120)

State variables written after the call(s):

- stakeHouseShares[stakeHouse].memberId = memberId (contract/banking/SlotSettlementPool.sol#124)

Reentrancy in slotSettlementPool.selectStakeHouse(stakeHouse,memberId) (contracts/banking/SlotSettlementPool.sol#113):

- External calls:
  - sbhToken.mint(stakeHouse,memberId,amount,\_sbhTicketRequired) (contract/banking/SlotSettlementPool.sol#113)

State variables written after the call(s):

- stakeHouseShares[stakeHouse].memberId = memberId (contract/banking/SlotSettlementPool.sol#113)

Reentrancy in slotSettlementPool.selectStakeHouse(stakeHouse,memberId) (contracts/banking/SlotSettlementPool.sol#113):

- External calls:
  - sbhToken.mint(stakeHouse,memberId,amount,\_sbhTicketRequired) (contract/banking/SlotSettlementPool.sol#113)

State variables written after the call(s):

- stakeHouseShares[stakeHouse].memberId = memberId (contract/banking/SlotSettlementPool.sol#113)

Reentrancy in slotSettlementPool.selectStakeHouse(stakeHouse,memberId) (contracts/banking/SlotSettlementPool.sol#113):

- External calls:
  - sbhToken.mint(stakeHouse,memberId,amount,\_sbhTicketRequired) (contract/banking/SlotSettlementPool.sol#113)

State variables written after the call(s):

- stakeHouseShares[stakeHouse].memberId = memberId (contract/banking/SlotSettlementPool.sol#113)



## guards/ModuleGuards.sol

Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address,address) (contracts/StakeHouseUniverse.sol#169-215):  
 External calls:  
 - locFactoryProxy = new StakeHouseUpgradeableProxy(address(this),abi.encodePacked()), (contracts/StakeHouseUniverse.sol#179-183)  
 - BNFTProxy = new StakeHouseUpgradeableProxy(address(\_brandFactory),abi.encodePacked()), (contracts/StakeHouseUniverse.sol#187-191)  
 State variable written after the call():  
 - \_brandFactory = \_brandFactory.upgradeToAndCall(\_brandFactory,abi.encodeWithSelector(\_brandFactory.selector,abi.encodeWithSelector(\_brandFactory.selector,address(this)),\_brandBNFT,\_skLocFactory,\_claimAuction)) (contracts/StakeHouseUniverse.sol#191-205)  
 State variable written after the call():  
 - \_brandBNFT = \_brandBNFT.upgradeToAndCall(\_brandBNFT.addresses) (contracts/StakeHouseUniverse.sol#208)  
 Reference: <https://github.com/cryptic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1>  
 Reference: <https://github.com/cryptic/slither/wiki/Detector-Documentation#uninitialized-local-variables>  
 ERC721Upgradable.\_checkNonERC721Received(address,address,uint256,bytes) (node\_modules/Openzeppelin/contracts-upgradeable/solidity/ERC721/ERC721Upgradable.sol#36-59) ignores return value by ERC721ReceiverUpgradeable(to).onERC721Received(\_msgSender(),from,toTokenId,data) (node\_modules/Openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradable.sol#35-53)  
 ERC1967Upgrade.\_upgradeAndCall(address,bytes,bool) (node\_modules/Openzeppelin/contracts/proxy/ERC1967/Upgrade.sol#1-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node\_modules/Openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#107-109)  
 ERC1967Upgrade.\_upgradeAndCall(address,bytes,bytes,bytes) (node\_modules/Openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#109-120) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature("upgradeTo(address)",newImplementation)) (node\_modules/Openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#110-119)  
 ERC1967Upgrade.\_upgradeAndCall(address,bytes,bytes) (node\_modules/Openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#112-122) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node\_modules/Openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#113-121)  
 ERC721Upgradable.\_upgradeAndCall(address,bytes,bytes) (node\_modules/Openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#123-125) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (node\_modules/Openzeppelin/contracts/token/ERC721/ERC721.sol#126-134)  
 Transient variable written after the call():  
 - \_msgSender() (contracts/account/Teeணांत्रिमनManager.sol#77-99) ignores return value by ERC721Receiver(to).onERC721Received(\_msgSender(),from,toTokenId,data) (node\_modules/Openzeppelin/contracts/token/ERC721/ERC721.sol#149-159)  
 accountManager, tokenKey, rToken, \_msgSender() (contracts/account/Teeணांत्रिमनManager.sol#84-88)  
 skLOOTfactory, openSKootBagAndRemoveToken(uint256,skLOOTfactory,skBAGComponent,address) (contracts/brand/skLOOTfactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked(),coo...),toString()),  
 \_brandTokenId,\_recipient (contracts/brand/skLOOTfactory.sol#221-226)  
 skLOOTfactory, openSKootBagAndRemoveToken(uint256,skLOOTfactory,skBAGComponent,address) (contracts/brand/skLOOTfactory.sol#228-233)  
 skLOOTfactory, openSKootBagAndRemoveToken(uint256,skLOOTfactory,skBAGComponent,address) (contracts/brand/skLOOTfactory.sol#230-246) ignores return value by skLoot.mint(skLootTokenInfo[\_tokenId].building,skLOOT.ItemType.sBuilding,\_brandTokenId,\_recipient) (contracts/brand/skLOOTfactory.sol#232-240)  
 Reference: <https://github.com/cryptic/slither/wiki/Detector-Documentation#unused-return>

## helpers/FlagHelper.sol

FlagHelper.exists(uint256) (contracts/helpers/FlagHelper.sol#1-20) 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.hasFlagOut(uint256) (contracts/helpers/FlagHelper.sol#26-28) is never used and should be removed  
 FlagHelper.isFlagSet(uint256) (contracts/helpers/FlagHelper.sol#11-13) is never used and should be removed  
 FlagHelper.kickMember(uint256) (contracts/helpers/FlagHelper.sol#9-10) is never used and should be removed  
 FlagHelper.rageOut(uint256) (contracts/helpers/FlagHelper.sol#12-14) is never used and should be removed  
 FlagHelper.setFlag(uint256,uint256) (contracts/helpers/FlagHelper.sol#32-44) is never used and should be removed  
 Reference: <https://github.com/cryptic/slither/wiki/Detector-Documentation#dead-code>

Pragma version=0.8.0 (contracts/helpers/FlagHelper.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.13/0.7.6  
 solc=0.8.5 is not recommended for deployment  
 Reference: <https://github.com/cryptic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

## accounts/AccountManager.sol

skLOOTfactory.skLocItemClaim(address,address,uint256) (contracts/brand/skLOOTfactory.sol#253-258) uses a weak PENG: "pickedItem = \_luckyDipItems(pseudoRandomNumber % luckyDipItems.length)" (contracts/brand/skLOOTfactory.sol#256)  
 skLOOTfactory.skLocItemClaim(address,address,uint256) (contracts/brand/skLOOTfactory.sol#253-258) uses a weak PENG: "pickedItem = \_specialLuckyDipGems(pseudoRandomNumber % \_specialLuckyDipGems.length)" (contracts/brand/skLOOTfactory.sol#258)  
 Reference: <https://github.com/cryptic/slither/wiki/Detector-Documentation#weak-PENG>

Base64 encode bytes (contracts/helpers/Base64.sol#1-7) contains an incorrect Shift operation: native(uint256,uint256) encode am = 0 <= am <= 240 (contracts/helpers/Base64.sol#2)  
 Base64 encode bytes (contracts/helpers/Base64.sol#1-7) contains an incorrect Shift operation: native(uint256,uint256) encode am = 240 < am < 245 (contracts/helpers/Base64.sol#15)

Reference: <https://github.com/cryptic/slither/wiki/Detector-Documentation#incorrect-parameter-mixup>

ERC20Upgradeable.\_gap (node\_modules/Openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#146) shadow:  
 - ContextUpgradeable, \_gap (node\_modules/Openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol#83) shadow:  
 - ERC20Upgradeable, \_gap (node\_modules/Openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#111)  
 - ContextUpgradeable, \_gap (node\_modules/Openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#56)  
 - ContextUpgradeable, \_gap (node\_modules/Openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#80)  
 - ERC1967Upgradeable, \_gap (node\_modules/Openzeppelin/contracts-upgradeable/token/ERC20/ERC1967Upgradeable.sol#15)  
 - ContextUpgradeable, \_gap (node\_modules/Openzeppelin/contracts-upgradeable/token/ERC20/ERC1967Upgradeable.sol#15)  
 Reference: <https://github.com/cryptic/slither/wiki/Detector-Documentation#incorrect-parameter-mixup>

saveETHReservePool.addNonceToOpenPool(address,bytes,address) (contracts/banking/saveETHReservePool.sol#151-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/saveETHReservePool.sol#208)  
 saveETHReservePool.addNonceToOpenPool(address,bytes,address) (contracts/banking/saveETHReservePool.sol#151-211) ignores return value by dETHToken.transfer(currentKeeper,dETHFromExchangePool) (contracts/banking/saveETHReservePool.sol#212)  
 saveETHReservePool.isolateNonceToOpenPool(address,bytes,address) (contracts/banking/saveETHReservePool.sol#142-269) ignores return value by saveETHToken.transferFrom(neoKeepers,address(this),saveETHRequiredForIsolation) (contracts/banking/saveETHReservePool.sol#304)  
 saveETHReservePool.deposit(address,uint256) (contracts/banking/saveETHReservePool.sol#107-129) ignores return value by dETHToken.transferFrom(coner,ETHReserveRate) (contracts/banking/saveETHReservePool.sol#301)  
 saveETHReservePool.deposit(address,uint256) (contracts/banking/saveETHReservePool.sol#107-129) ignores return value by dETHToken.transferFrom(\_owner,address(this),amount) (contracts/banking/saveETHReservePool.sol#327)  
 BrandCentralClaimAuction.bidForFlicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#10-16) ignores return value by shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#10)  
 BrandCentralClaimAuction.bidForFlicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#10-16) ignores return value by shbToken.transferFrom(may.sender,address(this),\_shbDebunk) (contracts/brand/BrandCentralClaimAuction.sol#10)  
 BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-190) ignores return value by shbToken.transfer(may.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#190)  
 Reference: <https://github.com/cryptic/slither/wiki/Detector-Documentation#unchecked-transfer>

BrandCentralClaimAuction.slitherConstructor(ConstantVariables) (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)  
 - BASE64\_ENCODED\_BALANCE = native(BASE64\_ENCODED\_BALANCE / 86400) (contracts/brand/BrandCentralClaimAuction.sol#29)  
 Base64 encode bytes (contracts/helpers/Base64.sol#1-42) performs a multiplication on the result of a division:  
 - encodedLen = 4 \* ((len + 2) / 3) (contracts/helpers/Base64.sol#1)  
 Reference: <https://github.com/cryptic/slither/wiki/Detector-Documentation#divide-before-multiply>

skLOOTfactory.skLocItemClaim(address,address,uint256) (contracts/brand/skLOOTfactory.sol#253-258) uses a dangerous strict equality:  
 - isSpecial (node\_modules/Openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#45)  
 Reference: <https://github.com/cryptic/slither/wiki/Detector-Documentation#dangerous-strict-equalities>

Meant to be: BrandCentralClaimAuction.bidForFlicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):  
 External calls:  
 - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)  
 - State variable written after the call():  
 - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#129)  
 - auction.biddingEnd = block.timestamp (contracts/brand/BrandCentralClaimAuction.sol#134)  
 - auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#134)  
 Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-190):  
 External calls:  
 - shbToken.transfer(auction.bidder,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)  
 - State variable written after the call():  
 - \_increaseCollateralizedBalance(address(shareToken),\_recipient,\_memberId,shareToken) (contracts/banking/SlotSettlementPool.sol#127)  
 State variable written after the call():  
 - stakeHouseShareToken[stakeHouse] = token (contracts/banking/SlotSettlementPool.sol#128)  
 Reentrancy in SlotSettlementPool.mintSharesBatch(address,bytes,uint256) (contracts/banking/SlotSettlementPool.sol#113-145):  
 External calls:  
 - stakeHouseShareToken[mintSharesBatch] (contracts/banking/SlotSettlementPool.sol#113)  
 - saveETHReservePool.mintSharesBatch(address,bytes,uint256) (contracts/banking/saveETHReservePool.sol#136)  
 State variable written after the call():  
 - saveETHReservePool.mintSharesBatch(address,bytes,uint256) (contracts/banking/saveETHReservePool.sol#142)  
 Reentrancy in SlotSettlementPool.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):

# AUTOMATED TESTING

```
External calls:
- unvested.rewardForMemberSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#89)
- slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
State variables written after the call(s):
- slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in SlotSettlementPool.mintSLOTAndShareBatch(address,bytes,address)
External calls:
- slot.mint(_memberIdSharesMinted,_memberId) (contracts/banking/SlotSettlementPool.sol#89)
- slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
shareToken.mint(_recipient,shareTokenSendFromManager) (contracts/banking/SlotSettlementPool.sol#110)
shareToken.mint(_recipient,shareTokenSendFromRecipient) (contracts/banking/SlotSettlementPool.sol#115)
State variables written after the call(s):
- _increaseCollateralizedBalance(address(_shareTokenId),_recipient,_memberId,sharesToSendFromManager) (contracts/banking/SlotSettlementPool.sol#116)
Reentrancy in StakeHouseUniverse.settleStakehouseBatch(address,uint256,string,bytes)
External calls:
- stakeHouseRegistry.name().call() (contracts/banking/StakeHouseRegistry(stakeHouseRegistry).init.selector,address(this)) (contracts/StakeHouseUniverse.sol#220-225)
- slotSettlementPool.deployToStakehouseShareToken(stakeHouseUniverseAddresses) (contracts/StakeHouseUniverse.sol#198)
State variables written after the call(s):
- registeredStakehouses[_stakeHouseId] = stakeHouse (contracts/StakeHouseUniverse.sol#142)
- memberKnotsToStakehouse[_memberId] = stakeHouse (contracts/StakeHouseUniverse.sol#142)
Reentrancy in skLOOTFactory.openSkLootBagAndDemoteVite(uint256,skLOOTFactory,skBagComponent,address)
External calls:
- abi.encodePacked((,coords.y.toString(),),skLOOT.itemType,brand,brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
- skLoot.mint(skLootTokenInfo._tokenId,building,skLOOT.itemType,building,brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#229-233)
- skLoot.mint(skLootTokenInfo._tokenId,character,skLOOT.itemType,character,brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
skLoot.mint(_tokenId,building,brandTokenId,_recipient) (after the call)
- skLootBagTokenIdToComponentRemoved[_componentTokenId] = true (contracts/brand/skLOOTFactory.sol#245)
Reentrancy in SlotSettlementPool.settleAndBuyLoot(address,bytes,address,uint256,bool)
External calls:
- slash._stakeHouse,_memberId,_amount,_skLootRequired (contracts/banking/SlotSettlementPool.sol#128)
- StakeHouseRegistry(_stakeHouse).lock(_memberId) (contracts/banking/SlotSettlementPool.sol#129)
- buySlashedLoot(_stakeHouse,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
- shareToken.mint(address(this),shareToLoot) (contracts/banking/SlotSettlementPool.sol#134)
State variables written after the call(s):
- buySlashedLoot(_stakeHouse,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
- stakedHouseCurrentLotStake[_stakeHouse] -= _amount (contracts/banking/SlotSettlementPool.sol#135)
- buySlashedLoot(_stakeHouse,_memberId,_amount,_slasher) (contracts/banking/SlotSettlementPool.sol#131)
- stakedHouseCurrentLotStake[_stakeHouse][_memberId] -= _amount (contracts/banking/SlotSettlementPool.sol#176)
Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address,address)
External calls:
- lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
- brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodePacked(_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)
References: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
AccountManager.registerValidatorInitial(address,bases,bytes), account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
References: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender), from,_tokenId, data (node_modules/@openzeppelin/contracts-upgradeable/erc721/ERC721Upgradable.sol#383-393)
ERC721Upgrades.upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967Upgrade.sol#72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC721Upgrades.upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967Upgrade.sol#107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#107-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo,address),oldImplementation) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#107-109)
ERC721Upgrades.upgradeToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newFeeEpoch).implementation(),data) (no data)
ERC721Upgrades.upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newFeeEpoch).implementation(),data) (no data)
ERC721Upgrades._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/contract/erc721/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender), from,_tokenId,_data) (node_modules/@openzeppelin/contracts/contract/erc721/ERC721.sol#374-386)
TransactOnManager.createStakehouse(msg.sender,blkPubKey,vkter,buildingsId,whbReport) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender,blkPubKey,vkter,buildingsId,whbReport) (contracts/accounts/TransactionManager.sol#77-89)
skLOOTFactory.openSkLootBagAndRemoveItem(uint256,skLOOTFactory,skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked(,coords.y.toString(),,coords.y.toString(),)),)
skLoot.itemType,brand,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(skLootTokenInfo._tokenId),building,skLOOT.itemType,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(skLootTokenInfo._tokenId),character,skLOOT.itemType,brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
References: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

## accounts/TransactionManager.sol

```

skLOOTFactory, skLocItemClaim(address, address, uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length]" (contracts/brand/skLOOTFactory.sol#286)*
skLOOTFactory, skLocItemClaim(address, address, uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length]" (contracts/brand/skLOOTFactory.sol#286)*
233)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-prngs

Base64 encode(bytes) (contracts/helpers/Base64.sol#12-42) contains an incorrect shift operation: mstore(0x256, uint256)(resultPtr, encode, amount 0 - 2, 0x8d3d << 240) (contracts/helpers/Base64.sol#52)
Base64 encode(bytes) (contracts/helpers/Base64.sol#12-42) contains an incorrect shift operation: mstore(0x256, uint256)(resultPtr, encode, amount_0 - 1, 0x8d3d << 240) (contracts/helpers/Base66.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC16Upgradable.._gap (mode modules/#OpenZeppelin/contracts-upgradeable/token/ERC16Upgradable.sol#86) shadows:
- ContextUpgradeable.._gap (mode modules/#OpenZeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC16Upgradable.._gap (mode modules/#OpenZeppelin/contracts-upgradeable/token/ERC16Upgradable.sol#93) shadows:
- EIP712Upgradeable.._gap (mode modules/#OpenZeppelin/contracts-upgradeable/token/ERC16Upgradable.sol#411)
- ERC20Upgradable.._gap (mode modules/#OpenZeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradable.sol#94)
ContextUpgradeable.._gap (mode modules/#OpenZeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#114) shadows:
- ERC16Upgradable.._gap (mode modules/#OpenZeppelin/contracts-upgradeable/utils/introspection/ERC16Upgradable.sol#35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-shadowing

saveETHReservePool.addNonceToOpenPool(address, bytes, address) (contracts/banking/saveETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper, saveETHToSend) (contracts/banking/saveETHReservePool.sol#108)
saveETHReservePool.addNonceToOpenPool(address, bytes, address, bytes, address) (contracts/banking/saveETHReservePool.sol#214-235) ignores return value by dETHToken.transfer(currentKeeper, dETHFromChangestate) (contracts/banking/saveETHReservePool.sol#214-235)
saveETHReservePool.withdrawFromOpenPool(address, bytes, address) (contracts/banking/saveETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(newKeeper, address(this), saveTHRequiresForIsolation) (contracts/banking/saveETHReservePool.sol#242-269)
saveETHReservePool.deposit(address, uint256) (contracts/banking/saveETHReservePool.sol#307-326) ignores return value by dETHToken.transfer(owner, dETHFromExchangeRate) (contracts/banking/saveETHReservePool.sol#301)
saveETHReservePool.withdraw(address, uint256) (contracts/banking/saveETHReservePool.sol#328-347) ignores return value by dETHToken.transfer(_owner, address(this), amount) (contracts/banking/saveETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string, uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender, address(this), shbBidGobalCount) (contracts/brand/BrandCentralClaimAuction.sol#109)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender, auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#199)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-technique

BrandCentralClaimAuction.slitherConstructorContractVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
REVERT_REASON = string(abi.encodePacked("AUCTION LENGTH IN BLOCKS = BLOCKS PER DAY * AUCTION LENGTH IN DAYS (", contracts/brand/BrandCentralClaimAuction.sol#28)
- Total Auction Length in Blocks = Blocks Per Day * Auction Length in Days (contracts/brand/BrandCentralClaimAuction.sol#28)
Base64 encode(bytes) (contracts/helpers/Base64.sol#12-42) performs a multiplication on the result of a division:
- encodes = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

skLOOTFactory.skLocItemClaim(address, address, uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a dangerous strict equality:
- _luckyDipItems[_memberId] == _luckyDipItems[_memberId] (contracts/brand/skLOOTFactory.sol#253-259)
- _specialLuckyDipGems[_memberId] == _specialLuckyDipGems[_memberId] (contracts/brand/skLOOTFactory.sol#253-259)
Reentrancy in BrandCentralClaimAuction.bidForTicker(string, uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
External calls:
- shbToken.transfer(auction.bidder, auction.shbid) (contracts/brand/BrandCentralClaimAuction.sol#125)
- ShbBidManager.._shbBidAmount = shbBidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
- auction.bidder..msg.sender (contracts/brand/BrandCentralClaimAuction.sol#159)
- auction..bidder (contracts/brand/BrandCentralClaimAuction.sol#161)
- auction..biddingEnd = auction.biddingEnd + 1 (contracts/brand/BrandCentralClaimAuction.sol#162)
- auction..biddingEnd = auction.biddingEnd + 1 (contracts/brand/BrandCentralClaimAuction.sol#162)
Reentrancy in SlotSettlementPool.buySlashedSlot(address, bytes, uint256, address) (contracts/banking/SlotSettlementPool.sol#161-191):
External calls:
- shbToken.mint(address(this), shareToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
State variables written after the call(s):
- _increasedCollateralisedTotalCollateral(address(shareToken), recipient, memberId, shareToMint) (contracts/banking/SlotSettlementPool.sol#107)
- stakeHouse..stakeHouseTotalCollateral(address(shareToken)) + amount (contracts/banking/CollateralisedStakeManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakehouseShareToken(addresses) (Contracts/banking/SlotSettlementPool.sol#64-65):
External call:
- shbToken.mint(address(this), stakeHouse..shbEncodeWithSelector(stETH(stETHBeacon).init.selector, address(this), _stakehouse)) (contracts/banking/SlotSettlementPool.sol#68-75)
State variables written after the call(s):
- stakeHouse.shareTokens[_stakehouse] = token (contracts/banking/SlotSettlementPool.sol#64)
Reentrancy in StakeHouseUniverse.mintSLOTAndSharesBatch(address, bytes, address) (contracts/banking/SlotSettlementPool.sol#133-145):
External calls:
- saveETHToken.mint(address(this), saveETHAmountScaled.mantissa) (contracts/banking/saveETHReservePool.sol#136)
State variables written after the call(s):
- stakeHouse..stakeHouseTotalCollateral(address(_stakehouse)) + amount (contracts/banking/saveETHReservePool.sol#142)
Reentrancy in StakeHouseUniverse.mintSLOTAndSharesBatch(address, bytes, address) (contracts/banking/SlotSettlementPool.sol#88-117):
External calls:
- stakeHouse..stakeHouseTotalCollateral(_stakehouse) + amount (contracts/banking/StakeHouseRegistryBeacon.sol#88-117)
- stakeHouse..stakeHouseTotalCollateral(_stakehouse) + amount (contracts/banking/StakeHouseRegistryBeacon.sol#103)
State variables written after the call(s):
- _stakeHouse..stakeHouseTotalCollateral(_stakehouse) + amount (contracts/banking/StakeHouseRegistryBeacon.sol#104)
Reentrancy in StakeHouseUniverse.mintSLOTAndSharesBatch(address, bytes, address) (contracts/banking/SlotSettlementPool.sol#88-117):
External calls:
- stakeHouse..stakeHouseTotalCollateral(_stakehouse) + amount (contracts/banking/StakeHouseRegistryBeacon.sol#88-117)
- stakeHouse..stakeHouseTotalCollateral(_stakehouse) + amount (contracts/banking/StakeHouseRegistryBeacon.sol#103)
- shareToken..mint(_recipient, shareToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
State variables written after the call(s):
- _increasedCollateralisedBalance(address(shareToken), _recipient, memberId, shareToSendToManager) (contracts/banking/SlotSettlementPool.sol#116)
Reentrancy in StakeHouseUniverse.registerMember(stakeHouseAddress, firstMember) (Contracts/banking/StakeHouseUniverse.sol#229-235):
External calls:
- stakeHouseRegistry..registerMember(stakeHouseAddress, firstMember) (Contracts/banking/StakeHouseUniverse.sol#229-235)
State variables written after the call(s):
- _registeredMembers[stakeHouseAddress][firstMember] = true (Contracts/banking/StakeHouseUniverse.sol#229-235)
Reentrancy in skLOOTFactory.openStakeLoootAndRemoveToken(skLOOTFactory, address) (contracts/brand/skLOOTFactory.sol#203-246):
External calls:
- skLOOT..encodePacked((,coorder.x.toString(), ,coorder.y.toString(), )) (Contracts/brand/skLOOTFactory.sol#221-226)
- skLOOT..mint(skLocTokenInfo..tokenId..building, skLOOT..itemType..building, brandTokenId..recipient) (Contracts/brand/skLOOTFactory.sol#228-239)
- skLOOT..mint(skLocTokenInfo..tokenId..character, skLOOT..itemType..character, brandTokenId..recipient) (Contracts/brand/skLOOTFactory.sol#235-240)
- skLocTokenInfo..tokenId..character..removeTokenId (Contracts/brand/skLOOTFactory.sol#235-240)
- skLocTokenInfo..tokenId..character..removeTokenId (Contracts/brand/skLOOTFactory.sol#235-240)
Reentrancy in StakeHouseUniverse.buySlashedSlotAndBuyMyLot(address, bytes, address, uint256, bool) (contracts/banking/SlotSettlementPool.sol#120-132):
External calls:
- stakeHouse..memberId..amount..isWkRequired (Contracts/banking/SlotSettlementPool.sol#128)
- StakeHouseRegistry.._stakeHouse..wktRequired (Contracts/banking/SlotSettlementPool.sol#154)
- buySlashedSlot.._stakeHouse..stakeHouse..stakeHouse..amount (Contracts/banking/SlotSettlementPool.sol#133)
- buySlashedSlot.._stakeHouse..stakeHouse..stakeHouse..amount (Contracts/banking/SlotSettlementPool.sol#175)
- buySlashedSlot.._stakeHouse..memberId..amount..stakeHouse..amount (Contracts/banking/SlotSettlementPool.sol#131)
- stakeHouseMemberCurrentSlotSlashed.._stakeHouse..memberId..amount..stakeHouse..amount (Contracts/banking/SlotSettlementPool.sol#176)

```



## accounts/Streamer.sol

```

skLOOTFactory, skLocItemClaim(address, address, uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length]" (contracts/brand/skLOOTFactory.sol#286)*
skLOOTFactory, skLocItemClaim(address, address, uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length]" (contracts/brand/skLOOTFactory.sol#286)*
[...]
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-prng

Base64 encode(bytes) (contract/Helpers/Base64.sol#12-42) contains an incorrect shift operation: mstore(0x20, uint256(resultPtr.encode().asm_0 - 2).Orb3d <> 240) (contracts/helpers/Base64.sol#52)
Base64 encode(bytes) (contract/Helpers/Base64.sol#12-42) contains an incorrect shift operation: mstore(0x20, uint256(resultPtr.encode().asm_0 - 1).Orb3d <> 240) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC16Upgradable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/ERC16Upgradable.sol#86) shadows:
- ContextUpgradeable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/utility/ContextUpgradeable.sol#30)
ERC16Upgradable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/introspection/ERC16Upgradable.sol#93) shadows:
- EIP712Upgradeable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/introspection/EIP712Upgradeable.sol#411)
- ERC20Upgradeable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#94)
ContextUpgradeable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/utility/ContextUpgradeable.sol#114) shadows:
- ERC16Upgradable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/utility/introspection/ERC16Upgradable.sol#35)
[...]
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-shadowing

saveETHReservePool.addNonceToOpenPool(address, bytes, address) (contracts/banking/saveETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper, saveETHToSend) (contracts/banking/saveETHReservePool.sol#108)
saveETHReservePool.addNonceToOpenPool(address, bytes, address) (contracts/banking/saveETHReservePool.sol#214-235) ignores return value by dETHToken.transfer(currentKeeper, dETHFromChangestate) (contracts/banking/saveETHReservePool.sol#108)
saveETHReservePool.depositFromNewKeepers(address, bytes, address) (contracts/banking/saveETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(newKeeper, address(this), saveTHRequiresForIsolation) (contracts/banking/saveETHReservePool.sol#108)
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)
[...]
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#return-value-ignores
BrandCentralClaimAuction.bidForTicker(string, uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161) ignores return value by shbToken.transferFrom(msg.sender, address(this), shbBidGobalCount) (contracts/brand/BrandCentralClaimAuction.sol#109)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender, auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-technique

BrandCentralClaimAuction.slitherConstructorContractVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
REVERT_IF_BID_IS_ZERO = 100 * (BLOCKS_PER_DAY * AUCTION_LENGTH_IN_BLOCKS / AUCTION_LENGTH_IN_DAYS) (contracts/brand/BrandCentralClaimAuction.sol#28)
- REVERT_IF_BID_IS_ZERO = 100 * (BLOCKS_PER_DAY * AUCTION_LENGTH_IN_BLOCKS / AUCTION_LENGTH_IN_DAYS) (contracts/brand/BrandCentralClaimAuction.sol#28)
Base64 encode(bytes) (contract/Helpers/Base64.sol#12-42) performs a multiplication on the result of a division:
- encodes = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

skLOOTFactory.setLockType(skLOOTFactory.LockType.BID_EXTENSION) (contracts/brand/skLOOTFactory.sol#253-259) uses a dangerous strict equality:
- skLOOTFactory.LockType.BID_EXTENSION == skLOOTFactory.LockType.BID_EXTENSION (contracts/brand/skLOOTFactory.sol#253-259)
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.shbid) (contracts/brand/BrandCentralClaimAuction.sol#125)
- ShbHoldingProxy.auctionBidder = shbHoldingProxy.auctionBidder (contracts/brand/BrandCentralClaimAuction.sol#128)
- auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#159)
- auction.shbid = auction.bidder (contracts/brand/BrandCentralClaimAuction.sol#161)
- auction.biddingEnd = auction.biddingEnd + 1 (contracts/brand/BrandCentralClaimAuction.sol#162)
Reentrancy in SlotSettlementPool.buySlashedSlot(address, bytes, uint256, address) (contracts/banking/SlotSettlementPool.sol#161-191):
shbToken.mint(address(this), shareToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
State variables written after the call(s):
- _increaseCollateralisedBalance(shbToken, recipient, memberId, shareToMint) (contracts/banking/SlotSettlementPool.sol#107)
- stakeHouseTotalCollateralizedTokenAmount(ATRIBToken) += amount (contracts/banking/CollateralisedLotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakehouseShareToken(addresses) (Contracts/banking/SlotSettlementPool.sol#64-65):
External call:
- shbToken.mint(address(this), stakeHouse, shb.encodeWithSelector(stETH.stETHBeacon).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#68)
Reentrancy in StakeHouseUniverse.mintSLOTAndSharesBatch(address, bytes, address) (Contracts/banking/SlotSettlementPool.sol#88-117):
External calls:
- saveETHToken.mint(address(this), saveETHAmountScaled.mantissa) (contracts/banking/saveETHReservePool.sol#136)
- StakeHouseRegistry.setStakeHouseAddress(stakeHouseAddress) (Contracts/banking/StakeHouseRegistry.sol#142)
Reentrancy in StakeHouseUniverse.mintSLOTAndSharesBatch(address, bytes, address) (Contracts/banking/SlotSettlementPool.sol#88-117):
External calls:
- shbToken.mint(stakeHouseTotalCollateralizedBalanceBatched, address, bytes, address) (Contracts/banking/SlotSettlementPool.sol#88-117):
External calls:
- stakeHouseRegistry.setStakeHouseAddress(stakeHouseAddress) (Contracts/banking/StakeHouseRegistry.sol#142)
- shbToken.mint(STAKEHOUSE_MEMBER, stakeHouse, abi.encodeWithSelector(stETH.stETHBeacon).init.selector, address(this)) (Contracts/banking/StakeHouseUniverse.sol#229-235)
State variables written after the call(s):
- _registerMember(stakeHouse, firstMember) (Contracts/banking/StakeHouseRegistry.sol#140)
- _registerMember(stakeHouse, secondMember) (Contracts/banking/StakeHouseRegistry.sol#141)
Reentrancy in StakeHouseUniverse.registerMember(stakeHouse, firstMember, string, bytes) (Contracts/banking/StakeHouseUniverse.sol#229-235):
External calls:
- stakeHouseRegistry = new Beacon(stakeHouseRegistryBeacon, abi.encodeWithSelector(stakeHouseRegistry.stakeHouseRegistryBeacon).init.selector, address(this)) (Contracts/banking/StakeHouseUniverse.sol#229-235)
- stakeHouseRegistry.setStakeHouseAddress(stakeHouseAddress) (Contracts/banking/StakeHouseRegistry.sol#140)
State variables written after the call(s):
- _registerMember(stakeHouse, firstMember) (Contracts/banking/StakeHouseRegistry.sol#140)
- _registerMember(stakeHouse, secondMember) (Contracts/banking/StakeHouseRegistry.sol#141)
Reentrancy in skLOOTFactory.openStakeCapAndRemoveItemFunction(skLOOTFactory, skCapComponent, address) (Contracts/brand/skLOOTFactory.sol#203-246):
External calls:
- skLOOTFactory(stakingabi, encodePacked((,coorder.x.toString(),,coorder.y.toString(),)),skLOOTItemTypes.x.land, brandTokenId_x_recipient) (Contracts/brand/skLOOTFactory.sol#221-226)
- skLOOT.mint(skLocateTokenInfo[tokenId].building, skLocItemTypes.x.land, brandTokenId_x_recipient) (Contracts/brand/skLOOTFactory.sol#228-239)
- skLOOT.mint(skLocateTokenInfo[tokenId].character, skLocItemTypes.x.character, brandTokenId_x_recipient) (Contracts/brand/skLOOTFactory.sol#235-240)
- skLocateTokenInfo[tokenId].removed = true (Contracts/brand/skLOOTFactory.sol#243)
Reentrancy in StakeHouseUniverse.buySlashedSlot(address, bytes, address, uint256, bool) (Contracts/banking/SlotSettlementPool.sol#120-132):
External calls:
- stakeHouse.memberId, amount, isWkRequired (Contracts/banking/SlotSettlementPool.sol#128)
- StakeHouseRegistry._stakeHouse_idискRequired (Contracts/banking/SlotSettlementPool.sol#134)
- buySlashedSlot(_stakeHouse_idиск, _stakeHouse, _slasher) (Contracts/banking/SlotSettlementPool.sol#133)
- stakeHouseCurrentBalances[_stakeHouse_idиск] -= amount (Contracts/banking/SlotSettlementPool.sol#175)
- buySlashedSlot(_stakeHouse_memberId, amount, _slasher) (Contracts/banking/SlotSettlementPool.sol#131)
- stakeHouseMemberCurrentBalances[_stakeHouse_idиск] -= amount (Contracts/banking/SlotSettlementPool.sol#176)

```



## accounts/BalanceReporter.sol

```

skLOOTFactory, skLocItemClaim(address, address, uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length]" (contracts/brand/skLOOTFactory.sol#286)*
skLOOTFactory, skLocItemClaim(address, address, uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length]" (contracts/brand/skLOOTFactory.sol#286)*
253)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-prngs

Base64 encode(bytes) (contracts/helpers/Base64.sol#12-42) contains an incorrect shift operation: nature(uint256, uint256) resultFor_encode_am_0 = 2_0x8d << 240) (contracts/helpers/Base64.sol#52)
Base64 encode(bytes) (contracts/helpers/Base64.sol#12-42) contains an incorrect shift operation: nature(uint256, uint256) resultFor_encode_am_0 - 1_0x8d << 240) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC16Upgradable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/tokens/ERC16Upgradable.sol#86) shadows:
- ContextUpgradeable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC16Upgradable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/introspection/ERC16Upgradable.sol#93) shadows:
- EIP161Upgradeable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/introspection/EIP161Upgradeable.sol#411)
- ERC20Upgradeable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#36)
ContextUpgradeable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/introspection/ContextUpgradeable.sol#114) shadows:
- ERC16Upgradable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/introspection/ERC16Upgradable.sol#35)
- ContextUpgradeable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/introspection/ContextUpgradeable.sol#80)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-shadowing

saveETHReservePool.addNonceToOpenPool(address, bytes, address) (contracts/banking/saveETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper, saveETHToSend) (contracts/banking/saveETHReservePool.sol#108)
saveETHReservePool.addNonceToOpenPool(address, bytes, address, bytes, address) (contracts/banking/saveETHReservePool.sol#214-235) ignores return value by dETHToken.transfer(currentKeeper, dETHFromChangestate) (contracts/banking/saveETHReservePool.sol#108)
saveETHReservePool.depositFromNewKeepers(address, bytes, address) (contracts/banking/saveETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(newKeeper, address(this), saveTHRequiresForIsolation) (contracts/banking/saveETHReservePool.sol#108)
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)
saveETHReservePool.deposit(address, uint256, bytes) (contracts/banking/saveETHReservePool.sol#330-352) ignores return value by dETHToken.transferFrom(_owner, address(this), amount) (contracts/banking/saveETHReservePool.sol#351)
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#109)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#108-109) ignores return value by shbToken.transfer(msg.sender, auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-technique

BrandCentralClaimAuction.slitherConstructorContractVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
REVERT_REASON = string(abi.encodePacked("AUCTION LENGTH IN BLOCKS = BLOCKS PER DAY * AUCTION LENGTH IN DAYS")) (contracts/brand/BrandCentralClaimAuction.sol#28)
- Base64 encode(bytes) (contracts/helpers/Base64.sol#12-42) performs a multiplication on the result of a division:
- encodes = 4 * ((16n + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

skLOOTFactory.skLocItemClaim(address, address, uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a dangerous strict equality:
- _luckyDipItems[_memberId].balance = _luckyDipItems[_memberId].balance - _bidExtension (contracts/brand/skLOOTFactory.sol#253-259)
- _specialLuckyDipGems[_memberId].balance = _specialLuckyDipGems[_memberId].balance - _bidExtension (contracts/brand/skLOOTFactory.sol#253-259)
External calls:
- shbToken.transfer(auction.bidder, auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#125)
- ShbHoldingManager._shbHolding = shbHolding (contracts/BrandCentralClaimAuction.sol#128)
- auction.bidder = msg.sender (contracts/BrandCentralClaimAuction.sol#135)
- auction.shbBid = auction.biddingPrice + _bidExtension (contracts/BrandCentralClaimAuction.sol#134)
- auction.biddingPrice = auction.biddingPrice + _bidExtension (BLOCKS) (contracts/BrandCentralClaimAuction.sol#132)
Reentrancy in SlotSettlementPool.buySlashedSlot(address, bytes, uint256, address) (contracts/banking/SlotSettlementPool.sol#161-191):
- shbToken.mint(address(this), shareForMint) (contracts/banking/SlotSettlementPool.sol#181-184)
State variables written after the call(s):
- _increaseCollateralisedBalance(shareForMint, recipient, memberId, shareForMint) (contracts/banking/SlotSettlementPool.sol#107)
- stakeHouseTotalCollateralisedToken(shareForMint) += amount (contracts/banking/CollectsAlliedSlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakehouseShareToken(addresses) (Contracts/banking/SlotSettlementPool.sol#64-65):
External call:
- shbToken.mint(address(this), shareForMint) (Contracts/banking/SlotSettlementPool.sol#181-184)
State variables written after the call(s):
- stakeHouseShareTokens[stakehouse] = token (Contracts/banking/SlotSettlementPool.sol#65)
Reentrancy in StakeHouseUniverse.mintSLOTAndSharesBatch(address, bytes, uint256) (Contracts/banking/saveETHReservePool.sol#133-145):
External call:
- saveETHToken.mint(address(this), saveEthAmountScaled.mantissa) (Contracts/banking/saveETHReservePool.sol#136)
State variables written after the call(s):
- stakeHouseTotalCollateralisedToken(mantissa) += amount (Contracts/banking/saveETHReservePool.sol#136)
Reentrancy in StakeHouseUniverse.mintSLOTAndSharesBatch(address, bytes, address) (Contracts/banking/SlotSettlementPool.sol#88-117):
External calls:
- stakeHouseRegistry.registerMember(stakeHouseMember, memberId) (Contracts/banking/SlotSettlementPool.sol#89)
- slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (Contracts/banking/SlotSettlementPool.sol#103)
State variables written after the call(s):
- stakeHouseTotalCollateralisedToken(STAKEHOUSE_MEMBER) (Contracts/banking/SlotSettlementPool.sol#104)
- _registerMember(stakeHouseMember, memberId) (Contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in StakeHouseUniverse.mintSLOTAndSharesBatch(address, bytes, address) (Contracts/banking/SlotSettlementPool.sol#88-117):
External calls:
- stakeHouseRegistry.registerMember(stakeHouseMember, memberId) (Contracts/banking/SlotSettlementPool.sol#89)
- slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (Contracts/banking/SlotSettlementPool.sol#103)
- shareToken.mint(_recipient, shareForSendToManager) (Contracts/banking/SlotSettlementPool.sol#110)
- shareForSendToManager = shareForSendToManager + shareToken.mint(_recipient, shareForSendToManager) (Contracts/banking/SlotSettlementPool.sol#115)
State variables written after the call(s):
- _increaseCollateralisedBalance(address(shareForSendToManager), recipient, memberId, shareForSendToManager) (Contracts/banking/SlotSettlementPool.sol#116)
Reentrancy in StakeHouseUniverse.mintSLOTAndSharesBatch(address, bytes, string, bytes) (Contracts/banking/StakeHouseUniverse.sol#229-235):
External calls:
- stakeHouseRegistry.registerMember(stakeHouseMember, memberId) (Contracts/banking/StakeHouseUniverse.sol#230)
- stakeHouseRegistry.registerMember(stakeHouseMember, memberId) (Contracts/banking/StakeHouseUniverse.sol#234)
State variables written after the call(s):
- _registerMember(stakeHouseMember, memberId) (Contracts/banking/StakeHouseUniverse.sol#240)
- stakeHouseTotalCollateralisedToken(mantissa) += amount (Contracts/banking/StakeHouseUniverse.sol#240)
Reentrancy in skLOOTFactory.openStakeCapAndRemoveItemFunction(skLOOTFactory, skCapComponent, address) (Contracts/brand/skLOOTFactory.sol#203-246):
External calls:
- shbToken.transfer((string(shbToken.encodePacked({_coordinator.x.toString()})), _coordinator.y.toHexString()), skLOOTItemType.x.land, brandTokenId, _recipient) (Contracts/brand/skLOOTFactory.sol#221-226)
- skLOOT.mint(skLocItemTokenInfo[tokenId].building, skLOOT.itemType, building, brandTokenId, recipient) (Contracts/brand/skLOOTFactory.sol#228-239)
- skLOOT.mint(skLocItemTokenInfo[tokenId].character, skLOOT.itemType, character, skLOOT.brandTokenId, recipient) (Contracts/brand/skLOOTFactory.sol#235-240)
- skLocItemTokenInfo[tokenId].removed = true (Contracts/brand/skLOOTFactory.sol#243)
Reentrancy in StakeHouseUniverse.buySlashedAndBuyLot(address, bytes, address, uint256, bool) (Contracts/banking/SlotSettlementPool.sol#120-132):
External calls:
- stakeHouse_memberId, amount, _isTicketRequired (Contracts/banking/SlotSettlementPool.sol#128)
- StakeHouseRegistry._stakeHouse_id(_memberId) (Contracts/banking/SlotSettlementPool.sol#134)
- buySlashedLot(_stakeHouse_id, _stakeHouse, _slasher) (Contracts/banking/SlotSettlementPool.sol#133)
- buySlashedLot(_stakeHouse_id, _stakeHouse, _amount, _slasher) (Contracts/banking/SlotSettlementPool.sol#175)
- buySlashedLot(_stakeHouse_memberId, _stakeHouse, _amount) (Contracts/banking/SlotSettlementPool.sol#131)
- stakeHouseMemberCurrentSlotSlashed[_stakeHouse_id] -= _amount (Contracts/banking/SlotSettlementPool.sol#176)

```



## accounts/ETH2ReportValidator.sol

```

sLOOTFactory.sLocItemClaim(address,address,uint256) (contracts/brand/sLOOTFactory.sol#253-259) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length]" (contracts/brand/sLOOTFactory.sol#286)*
sLOOTFactory.sLocItemClaim(address,address,uint256) (contracts/brand/sLOOTFactory.sol#253-259) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length]" (contracts/brand/sLOOTFactory.sol#286)*
253)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-prngs

Base64 encode(bytes) (contracts/helpers/Base64.sol#12-42) contains an incorrect shift operation: nature(uint256, uint256) resultFor.encode(amount < 2, 0x8d3d <> 240) (contracts/helpers/Base64.sol#52)
Base64 encode(bytes) (contracts/helpers/Base64.sol#12-42) contains an incorrect shift operation: nature(uint256, uint256) resultFor.encode(amount - 1, 0x8d3d <> 240) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC16Upgradable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/token/ERC16Upgradable.sol#86) shadows:
- ContextUpgradeable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC16Upgradable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/token/ERC16Upgradable.sol#103) shadows:
- EIP712Upgradeable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/introspection/EIP712Upgradeable.sol#411)
- ERC20Upgradeable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#36)
ContextUpgradeable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/utils/introspection/ContextUpgradeable.sol#114) shadows:
- ERC16Upgradable._gap (mode modules/#OpenZeppelin/contracts-upgradeable/token/ERC16Upgradable.sol#35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-shadowing

saveETHReservePool.addNonceToOpenPool(address,bytes,address) (contracts/banking/saveETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/saveETHReservePool.sol#108)
saveETHReservePool.addNonceToOpenPool(address,bytes,address) (contracts/banking/saveETHReservePool.sol#214-225) ignores return value by dETHToken.transfer(currentKeeper,dETHFromChangestate) (contracts/banking/saveETHReservePool.sol#235)
saveETHReservePool.withdrawFromOpenPool(address,bytes,address) (contracts/banking/saveETHReservePool.sol#242-249) ignores return value by saveETHToken.transferFrom(newKeeper,address(this),saveETHRequiredForRelocation) (contracts/banking/saveETHReservePool.sol#240)
saveETHReservePool.deposit(address,uint256) (contracts/banking/saveETHReservePool.sol#307-324) ignores return value by dETHToken.transfer(owner,dETHFromExchangeRate) (contracts/banking/saveETHReservePool.sol#301)
saveETHReservePool.withdraw(address,uint256) (contracts/banking/saveETHReservePool.sol#325-329) ignores return value by dETHToken.transfer(_owner,address(this),amount) (contracts/banking/saveETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-151) 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-technique

BrandCentralClaimAuction.silberConstructor(ConstructorVariables) (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
REVERT_IF_BID_IS_ZERO = (BLOCKS_PER_DAY * AUCTION_LENGTH_IN_BLOCKS / AUCTION_LENGTH_IN_DAYS) (contracts/brand/BrandCentralClaimAuction.sol#28)
- AUCTION LENGTH IN BLOCKS = BLOCKS PER DAY * AUCTION LENGTH IN DAYS (contracts/brand/BrandCentralClaimAuction.sol#28)
Base64 encode(bytes) (contracts/helpers/Base64.sol#12-42) performs a multiplication on the result of a division:
- encodes = 4 * ((amount + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

sLOOTFactory.sLocItemClaim(address,uint256) (contracts/brand/sLOOTFactory.sol#253-259) uses a dangerous strict equality:
- _luckyDipItems[_memberId] == _luckyDipItems[_memberId] (contracts/brand/sLOOTFactory.sol#253-259)
Reentrancy in BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-161):
External calls:
- shbToken.transfer(auction.bidder,auction.shbid) (contracts/brand/BrandCentralClaimAuction.sol#125)
State variables written after the call():
- auction.ebbid = _shbidAmount (contracts/brand/BrandCentralClaimAuction.sol#128)
- auction.bidder = msg.sender (contracts/brand/BrandCentralClaimAuction.sol#159)
- auction.bidTime = auction.bidTime + 1 (contracts/brand/BrandCentralClaimAuction.sol#134)
- auction.biddingEnd = auction.biddingEnd + 1 (BLOCK EXTENSION IN BLOCKS) (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-191):
External calls:
- shbToken.mint(address(this),shareToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
State variables written after the call():
- _increasedCollateralisedBalance(shareToMint,recipient,_memberId,shareToMint) (contracts/banking/SlotSettlementPool.sol#107)
- stakeHouseTotalCollateralisedToken(ATTRIToken) += amount (contracts/banking/CollectsAlliedLotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakehouseShareToken(addresses) (Contracts/banking/SlotSettlementPool.sol#64-69):
External call:
- ethBeaconProxy(stakeHouse,shb).encodeWithSelector(stETH(stETHBeacon).init.selector,address(this),_stakehouse) (Contracts/banking/SlotSettlementPool.sol#68-75)
State variables written after the call():
- stakeHouseShareTokens(_stakehouse) = token (Contracts/banking/SlotSettlementPool.sol#64)
Reentrancy in StakeHouseUniverse.mintSLOTAndSharesBatch(address,bytes,address) (Contracts/banking/SlotSettlementPool.sol#88-145):
External calls:
- saveETHToken.mint(address(this),saveEthAmountScaled.mantissa) (Contracts/banking/saveETHReservePool.sol#136)
State variables written after the call():
- stakeHouseTotalCollateralisedToken(mantissa) += amount (Contracts/banking/saveETHReservePool.sol#114)
Reentrancy in StakeHouseUniverse.mintSLOTAndSharesBatch(address,bytes,address) (Contracts/banking/SlotSettlementPool.sol#88-117):
External calls:
- stakeHouseRegistry(stakeHouseRegistryBeacon,abi.encodeWithSelector(stakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (Contracts/StakeHouseUniverse.sol#229-235)
State variables written after the call():
- _registerMember(stakeHouse(stakeHouseAddress),firstMember) (Contracts/StakeHouseUniverse.sol#40)
Reentrancy in sLOOTFactory.openStakeCapAndRemove(slashed,skLootFactory,address) (Contracts/brand/sLOOTFactory.sol#203-246):
External calls:
- stakeHouseRegistry(stakeHouseRegistryBeacon,abi.encodeWithSelector(stakeHouseRegistry(stakeHouseRegistryBeacon).register.selector,address(this))) (Contracts/StakeHouseUniverse.sol#229-235)
- sLoot.mint(sLootTokenInfo[tokenId].building,skLoot.itemType,skLoot.building,brandTokenId,recipient) (Contracts/brand/sLOOTFactory.sol#228-239)
- skLoot.mint(sLootTokenInfo[tokenId].character,skLoot.itemType,skLoot.character,brandTokenId,recipient) (Contracts/brand/sLOOTFactory.sol#235-240)
- skLootTokenInfo[tokenId].isRemoved = true (Contracts/brand/sLOOTFactory.sol#243)
Reentrancy in StakeHouseUniverse.buySlashedAndBuyMyLot(address,bytes,address,uint256,tool) (Contracts/banking/SlotSettlementPool.sol#120-132):
External calls:
- stakeHouse.memberId,amount,_isTicketRequired (Contracts/banking/SlotSettlementPool.sol#128)
- StakeHouseRegistry._stakeHouse._ticketRequired (Contracts/banking/SlotSettlementPool.sol#154)
- buySlashedSlot(_stakeHouse,_stakeHouse,_staker) (Contracts/banking/SlotSettlementPool.sol#131)
- buySlashedSlot(_stakeHouse,_stakeHouse,_staker) -= amount (Contracts/banking/SlotSettlementPool.sol#175)
- buySlashedSlot(_stakeHouse,_memberId,_amount,_staker) (Contracts/banking/SlotSettlementPool.sol#131)
- stakeHouseMemberCurrentSlotSlashed[_stakeHouse] -= amount (Contracts/banking/SlotSettlementPool.sol#176)

```

# AUTOMATED TESTING

accounts/ETH2ValidationLib.sol

```
ETHEValidation_computerWithdrawalData(credentials) (contract://accounts/ETHEValidationLib.sol:71-76) is never used and should be removed
Reference: https://github.com/crytic/solcher/wiki/Detector-Documentation#dead-code

Pragma version 5.9 (contract://accounts/ETHEValidationLib.sol:10) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Parameter ETCValidation.getDepositDataRoot(bytes,bytes,uint256)_1Signature (contract://accounts/ETHEValidationLib.sol:89) is not in mixedCase
Parameter ETCValidation.getDepositDataRoot(bytes,bytes,uint256)_1Signature (contract://accounts/ETHEValidationLib.sol:89) is not in mixedCase
Parameter ETCValidation.withdrawalCredentials (contract://accounts/ETHEValidationLib.sol:105) is not in mixedCase
Parameter ETCValidation.withdrawalCredentials (contract://accounts/ETHEValidationLib.sol:105) is not in mixedCase
Reference: https://github.com/crytic/solcher/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

getDepositDataRoot(bytes,bytes,uint256)_1Signature should be declared external!
- ETCValidation.getDepositDataRoot(bytes,bytes,uint256) (contract://accounts/ETHEValidationLib.sol:92-91)
Reference: https://github.com/crytic/solcher/wiki/Detector-Documentation#public-functions-that-can-be-declared-external
```

proxies/StakeHouseUpgradeableProxy.sol

```
soloLootFactory.soloLootClaim(address, address, uint256)(contractors.brand.soloLootFactory.sol@253-259) uses a weak PRNG: "pickedItem = luckyLipGems[pseudoRandomNumber % luckyLipGems.length]" (contractors.brand.soloLootFactory.sol@286)*
soloLootFactory.soloLootClaim(address, address, uint256)(contractors.brand.soloLootFactory.sol@253-259) uses a weak PRNG: "pickedItem = soloLootItems[pseudoRandomNumber % soloLootItems.length]" (contractors.brand.soloLootFactory.sol@286)*
References: https://github.com/crytic/solcher/wiki/Detector-Documentation#weak-prng

Base64_encode(bytes)(contractors.helpers/Base64.sol@12-43) contains an incorrect shift operation: matoro(uint256,uint256)(resultTx, encode_emm = 2,0x132d <> 240) (contractors/helpers/Base64.sol@52)
Base64_encode(bytes)(contractors.helpers/Base64.sol@12-43) contains an incorrect shift operation: matoro(uint256,uint256)(resultTx, encode_emm = 0,1,0x3d <> 240) (contractors/helpers/Base64.sol@55)
References: https://github.com/crytic/solcher/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradable.sol@81) shadows:
- ContextUpgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradable.sol@81)
- ContextUpgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradable.sol@93) shadows:
  - EIP116Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP116Upgradable.sol@111)
  - ERC20Upgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradable.sol@96)
- EIP116Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradable.sol@111)
- ERC20Upgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradable.sol@114) shadows:
  - EIP116Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/introspection/ERC116Upgradable.sol@35)
- ERC116Upgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/introspection/ERC116Upgradable.sol@35)
  - ContextUpgradable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradable.sol@80)
References: https://github.com/crytic/solcher/wiki/Detector-Documentation#variable-shadowing

soloETHReservePool.addDepositorOpenPool(addresses,bytes,adresses) (contractors/banking/soloETHReservePool.sol@151-211) ignores return value by saveETHReserve.transfer(currentDeeper,saveETHReserve) (contractors/banking/soloETHReservePool.sol@210)
soloETHReservePool.addDepositorOpenPool(addresses,bytes,adresses) (contractors/banking/soloETHReservePool.sol@214-239) ignores return value by dETHReserve.transfer((currentDeeper,dETHReserve)exchangeRate) (contractors/banking/soloETHReservePool.sol@235)
saveETHReservePool.addDepositorOpenPool(addresses,bytes,adresses) (contractors/banking/soloETHReservePool.sol@422-462) ignores return value by saveETHReserve.transfer((newkeeper,address(this)),saveETHRequiredForIsolation) (contractors/banking/soloETHReservePool.sol@461)
saveETHReservePool.withdraw(addresses,uint256) (contractors/banking/soloETHReservePool.sol@283-304) ignores return value by dETHReserve.transfer(_owner,dETHReserveChangeRate) (contractors/banking/soloETHReservePool.sol@303)
saveETHReservePool.withdraw(addresses,uint256) (contractors/banking/soloETHReservePool.sol@307-338) ignores return value by dETHReserve.transfer(_owner,amount) (contractors/banking/soloETHReservePool.sol@337)
soloETHReservePool.withdraw(addresses,uint256) (contractors/banking/soloETHReservePool.sol@339-360) ignores return value by dETHReserve.transfer(_owner,amount) (contractors/banking/soloETHReservePool.sol@360)
soloCentralClaimAuction.bidForTicket(string, uint256) (contractors/brand/BrandCentralClaimAuction.sol@19-161) ignores return value by shbToken.transferFrom(may_senders,addres(this),shbBid) (contractors/brand/BrandCentralClaimAuction.sol@198)
soloCentralClaimAuction.bidForTicket(string, uint256) (contractors/brand/BrandCentralClaimAuction.sol@188-199) ignores return value by shbToken.transfer(may_senders,auction.shbBid) (contractors/brand/BrandCentralClaimAuction.sol@199)
References: https://github.com/crytic/solcher/wiki/Detector-Documentation#unchecked-transfers
```

```

External calls:
- unvested.rewardForMemberSharesMinted(_memberId) (contracts/banking/SlotSettlementPool.sol#89)
- slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
State variables written after the call(s):
- slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in SlotSettlementPool.mintSLOTAndShareBatch(address,bytes,address)
External calls:
- slot.mint(_memberIdSharesMinted,_memberId) (contracts/banking/SlotSettlementPool.sol#89)
- slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
shareToken.mint(_recipient,shareTokenSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
shareTokenSendToManager.mint(_recipient,shareTokenSendToManager) (contracts/banking/SlotSettlementPool.sol#115)
State variables written after the call(s):
- _increaseCollateralizedBalance(address(shareToken),_recipient,_memberId,sharesMintedManager) (contracts/banking/SlotSettlementPool.sol#116)
Reentrancy in StakeHouseUniverse.setStakehouseBatch(address,uint256,string,bytes,address)
External calls:
- stakeHouseRegistry.name().call() (contracts/banking/StakeHouseRegistry(stakeHouseRegistry).init.selector,address(this)) (contracts/StakeHouseUniverse.sol#220-225)
- slotSettlementPool.deployToStakehouseShareToken(stakeHouseUniverseAddresses) (contracts/StakeHouseUniverse.sol#198)
State variables written after the call(s):
- registeredStakehouses[_stakehouseId] = stakeHouse (contracts/StakeHouseUniverse.sol#142)
- memberKnotsToStakehouse[_memberId] = stakeHouse (contracts/StakeHouseUniverse.sol#142)
Reentrancy in skLOOTFactory.openSkLootBagAndDemoteVite(uint256,skLOOTFactory,skBagComponent,address)
External calls:
- abi.encodePacked((,coords.y.toString(),),skLOOT.itemType.bLand,brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
- skLoot.mint(skLootTokenInfo._tokenId,building,skLOOT.itemType.bBuilding,brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#229-233)
- skLoot.mint(skLootTokenInfo._tokenId,character,skLOOT.itemType.eCharacter,brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
skLoot.mint(_tokenId,skLootTokenInfo._tokenId) (after the call)
- skLootBagTokenIdComponentsRemoved[_componentTokenId] = true (contracts/brand/skLOOTFactory.sol#245)
Reentrancy in SlotSettlementPool.setStakehouseByLot(address,bytes,address,uint256,bool)
External calls:
- slash._stakeHouse,_memberId,_amount,_skLockRequired) (contracts/banking/SlotSettlementPool.sol#128)
- StakeHouseRegistry(_stakeHouse).lock(_memberId) (contracts/banking/SlotSettlementPool.sol#129)
- buyStakehouse(_stakeHouse,_amount,_spender,_allowance) (contracts/banking/SlotSettlementPool.sol#133)
- shareToken.mint(address(this),shareToken) (contracts/banking/SlotSettlementPool.sol#134)
State variables written after the call(s):
- buyStakehouse(_stakeHouse,_amount,_spender,_allowance) (contracts/banking/SlotSettlementPool.sol#133)
- stakedHouseCurrentLotStakeholder[_stakeHouse] = _amount (contracts/banking/SlotSettlementPool.sol#135)
- buySlashedStakeHouse(_stakeHouse,_memberId,_amount,_spender) (contracts/banking/SlotSettlementPool.sol#131)
- stakedHouseNoncurrentLotStakeholder[_stakeHouse](_memberId) = _amount (contracts/banking/SlotSettlementPool.sol#176)

Reentrancy in StakeHouseUniverse.superchargeAndInitBrandCentral(address,address,address,address,address)
External calls:
- lootFactoryProxy = new StakeHouseUpgradeableProxy(address(_lootFactoryLogic),address(this),abi.encodePacked()) (contracts/StakeHouseUniverse.sol#179-183)
- brandCentralProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address(this),abi.encodePacked(_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)
References: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
AccountManager.registerValidatorInitial(address,bases,bytes), account (contracts/accounts/AccountManager.sol#71) is a local variable never initialized
References: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradable._checkOnERC721Received(addresses,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender), from,_tokenId, data (node_modules/@openzeppelin/contracts-upgradeable/erc721/ERC721Upgradable.sol#383-393)
ERC721Upgrades._upgradeTokenIdCall(addresses,bytes,bool) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC721Upgrades.sol#72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#89)
ERC1967Upgrade._upgradeTokenIdCallSecure(addresses,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#107) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#107-107) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo,address),oldImplementation) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#107-109)
ERC1967Upgrade._upgradeWithAContractCall(addresses,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#182-192) ignores return value by Address.functionDelegateCall(IBeacon(newBeacon).implementation(),data) (no data)
ERC721L1._checkOnERC721Received(addresses,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/erc721/ERC721.sol#368-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender), from,_tokenId,_data) (node_modules/@openzeppelin/contracts/token/erc721/ERC721.sol#368-390)
TransactOnManager.createStakehouse(msg.sender,blkPubKey,vkter,buildingsId,whbReport) (contracts/accounts/TransactionManager.sol#77-89) ignores return value by accountManager.createStakehouse(msg.sender,blkPubKey,vkter,buildingsId,whbReport) (contracts/accounts/TransactionManager.sol#77-89)
skLOOTFactory.openSkLootBagAndRemoveItem(uint256,skLoOTFactory,skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(string(abi.encodePacked(,coords.y.toString(),,coords.y.toString(),)),)
skLoot.itemType.bLand,brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#221-226)
skLoot.itemType.eBuilding,brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#229-233)
skLoot.itemType.eCharacter,brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#235-240)
skLOOTFactory.openSkLootBagAndRemoveItem(uint256,skLOOTFactory,skBagComponent,address) (contracts/brand/skLOOTFactory.sol#203-246) ignores return value by skLoot.mint(skLootTokenInfo._tokenId),building,skLOOT.itemType.bBuilding,_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(skLootTokenInfo._tokenId),character,skLOOT.itemType.eCharacter,_brandTokenId,_recipient) (contracts/brand/skLOOTFactory.sol#229-233)
References: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

```

## proxies/UniverseUpgradeableProxy.sol

```

skLOOTFactory.skLocItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PRNG: "pickedItem = _luckyDipItems[pseudoRandomNumber % _luckyDipItems.length]" (contracts/brand/skLOOTFactory.sol#286)*
skLOOTFactory.skLocItemClaim(address,address,uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a weak PRNG: "pickedItem = _specialLuckyDipGems[pseudoRandomNumber % _specialLuckyDipGems.length]" (contracts/brand/skLOOTFactory.sol#286)*
[...]
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-prngs

Base64 encode(bytes) (contracts/helpers/Base64.sol#12-42) contains an incorrect shift operation: mstore(uint256, uint256) resultPtr.encode(asm 0 - 2, 0x8d << 240) (contracts/helpers/Base64.sol#52)
Base64 encode(bytes) (contracts/helpers/Base64.sol#12-42) contains an incorrect shift operation: mstore(uint256, uint256) resultPtr.encode(asm_0 - 1, 0x8d << 240) (contracts/helpers/Base64.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#shift-parameter-mixup

ERC20Upgradable._gap (mode modules/#openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradable.sol#86) shadows:
- ContextUpgradable._gap (mode modules/#openzeppelin/contracts-upgradeable/tokens/ContextUpgradable.sol#30)
ERC20Upgradable._gap (mode modules/#openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradable.sol#93) shadows:
- ERC165Upgradable._gap (mode modules/#openzeppelin/contracts-upgradeable/introspection/ERC165Upgradable.sol#41)
- ERC20Upgradable._gap (mode modules/#openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradable.sol#94)
ContextUpgradable._gap (mode modules/#openzeppelin/contracts-upgradeable/tokens/ContextUpgradable.sol#114) shadows:
- ERC165Upgradable._gap (mode modules/#openzeppelin/contracts-upgradeable/introspection/ERC165Upgradable.sol#40)
- ContextUpgradable._gap (mode modules/#openzeppelin/contracts-upgradeable/tokens/ContextUpgradable.sol#80)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-shadowing

saveETHReservePool.addNonceForOpenPool(address,bytes,address) (contracts/banking/saveETHReservePool.sol#191-211) ignores return value by saveETHToken.transfer(currentKeeper,saveETHToSend) (contracts/banking/saveETHReservePool.sol#108)
saveETHReservePool.addNonceForOpenPool(address,bytes,address) (contracts/banking/saveETHReservePool.sol#214-229) ignores return value by dETHToken.transfer(currentKeeper,dETHFromChangestate) (contracts/banking/saveETHReservePool.sol#235)
saveETHReservePool.withdrawFromOpenPool(address,bytes,address) (contracts/banking/saveETHReservePool.sol#242-269) ignores return value by saveETHToken.transferFrom(newKeeper,address(this),saveETHRequiredForIsolation) (contracts/banking/saveETHReservePool.sol#241)
saveETHReservePool.deposit(address,uint256) (contracts/banking/saveETHReservePool.sol#307-324) ignores return value by dETHToken.transfer(owner,dETHFromExchangeRate) (contracts/banking/saveETHReservePool.sol#301)
saveETHReservePool.withdraw(address,uint256) (contracts/banking/saveETHReservePool.sol#325-329) ignores return value by dETHToken.transfer(_owner,address(this),amount) (contracts/banking/saveETHReservePool.sol#327)
BrandCentralClaimAuction.bidForTicker(string,uint256) (contracts/brand/BrandCentralClaimAuction.sol#109-151) ignores return value by shbToken.transferFrom(msg.sender,address(this),shbBidGobalCount) (contracts/brand/BrandCentralClaimAuction.sol#105)
BrandCentralClaimAuction.claimSHB(uint256) (contracts/brand/BrandCentralClaimAuction.sol#188-199) ignores return value by shbToken.transfer(msg.sender,auction.shbBid) (contracts/brand/BrandCentralClaimAuction.sol#195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-technique

BrandCentralClaimAuction.slitherConstructorContractVariables() (contracts/brand/BrandCentralClaimAuction.sol#14-263) performs a multiplication on the result of a division:
REVERT_REASON = string(abi.encodePacked("REVERT_REASON = ", uint256("TOTAL AUCTION LENGTH IN BLOCKS - BLOCKS PER DAY * AUCTION LENGTH IN DAYS"), abi.encodePacked(" contracts/brand/BrandCentralClaimAuction.sol#28"))
Base64 encode(bytes) (contracts/helpers/Base64.sol#12-42) performs a multiplication on the result of a division:
- encodes = 4 * ((len + 2) / 3) (contracts/helpers/Base64.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

skLOOTFactory.skLocItemClaim(address,uint256) (contracts/brand/skLOOTFactory.sol#253-259) uses a dangerous strict equality:
- _luckyDipItems[_memberId] == _luckyDipItems[_memberId] (contracts/brand/skLOOTFactory.sol#253-259)
- _specialLuckyDipGems[_memberId] == _specialLuckyDipGems[_memberId] (contracts/brand/skLOOTFactory.sol#253-259)
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.shbid) (contracts/brand/BrandCentralClaimAuction.sol#125)
- ShbHolding = shbHolding - auction.shbid (contracts/brand/BrandCentralClaimAuction.sol#128)
- auction.bidder == msg.sender (contracts/brand/BrandCentralClaimAuction.sol#159)
- auction.bidder == auction.bidder (contracts/brand/BrandCentralClaimAuction.sol#134)
- auction.biddingEnd = auction.biddingEnd + 1 (contracts/brand/BrandCentralClaimAuction.sol#152)
Reentrancy in SlotSettlementPool.buySlashedSlot(address,bytes,uint256,address) (contracts/banking/SlotSettlementPool.sol#161-191):
shbToken.mint(address(this),shareToMint) (contracts/banking/SlotSettlementPool.sol#181-184)
State variables written after the call(s):
- _increasedCollateralisedTotalCollaterals(shbToken,recipient,_memberId,shareToMint) (contracts/banking/SlotSettlementPool.sol#107)
- stakeHouseSharesTotal(shbToken) += amount (contracts/banking/CollectsAllignedlotManager.sol#40)
Reentrancy in SlotSettlementPool.deployStakeHouseShareToken(addresses) (Contracts/banking/SlotSettlementPool.sol#64-69):
External call:
- saveETHToken.mint(addresses),saveEthAmountScaled.mantissa) (contracts/banking/saveETHReservePool.sol#136)
State variables written after the call(s):
- stakeHouseShares(stakeHouse) = token (contracts/banking/SlotSettlementPool.sol#64)
Reentrancy in StakeHouseRegistry.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/saveETHReservePool.sol#113-145):
External call:
- saveETHToken.mint(addresses),saveEthAmountScaled.mantissa) (contracts/banking/saveETHReservePool.sol#136)
State variables written after the call(s):
- stakeHouseShares(stakeHouse) = token (contracts/banking/SlotSettlementPool.sol#113)
Reentrancy in StakeHouseRegistry.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
External calls:
- stakeHouseRegistry.mintSLOTAndSharesBatch(slashed,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117)
- slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
State variables written after the call(s):
- _increasedCollateralisedBalance(slashed,_memberId,slot.mint(SLOT_PER_STAKEHOUSE_MEMBER)) (contracts/banking/SlotSettlementPool.sol#104)
Reentrancy in StakeHouseRegistry.mintSLOTAndSharesBatch(address,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117):
External calls:
- stakeHouseRegistry.mintSLOTAndSharesBatch(slashed,bytes,address) (contracts/banking/SlotSettlementPool.sol#88-117)
- slot.mint(SLOT_PER_STAKEHOUSE_MEMBER) (contracts/banking/SlotSettlementPool.sol#103)
- shareToken.mint(_recipient,shareToSendToManager) (contracts/banking/SlotSettlementPool.sol#110)
- shareToken.mint(_recipient,shareToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#115)
State variables written after the call(s):
- _increasedCollateralisedBalance(slashed,shareToSendToManager,_memberId,shareToSendToRecipient) (contracts/banking/SlotSettlementPool.sol#116)
Reentrancy in StakeHouseRegistry.registerMember(stakeHouseAddress,firstMember) (Contracts/banking/StakeHouseUniverse.sol#229-235):
External calls:
- stakeHouseRegistry = new Beacon(stakeHouseRegistryBeacon,abi.encodeWithSelector(IStakeHouseRegistry(stakeHouseRegistryBeacon).init.selector,address(this))) (contracts/StakeHouseUniverse.sol#229-235)
- stakeHouseRegistry.registerMember(stakeHouseAddress,firstMember) (Contracts/banking/StakeHouseUniverse.sol#240)
State variables written after the call(s):
- _registeredMembers(stakeHouse(stakeHouseAddress),firstMember) (Contracts/banking/StakeHouseUniverse.sol#240)
Reentrancy in skLOOTFactory.openStakeLootAndRemoveItem(slashed,skLOOTFactory,address) (contracts/brand/skLOOTFactory.sol#203-246):
External calls:
- skLOOTFactory(string(abi.encodePacked(")),coorder.x.toString()),,coorder.y.toIntString(),)) (contracts/brand/skLOOTFactory.sol#221-226)
- skLocToken.mint(skLocTokenInfo[tokenId].building,skLocTokenInfo[tokenId].recipient) (contracts/brand/skLOOTFactory.sol#228-239)
- skLocToken.mint(skLocTokenInfo[tokenId].character,skLocTokenInfo[tokenId].recipient) (contracts/brand/skLOOTFactory.sol#235-240)
State variables written after the call(s):
- buySlashedItem(_itemType,_character,_skLocTokenInfo[tokenId].recipient) (Contracts/banking/SlotSettlementPool.sol#134)
- stakeHouseMemberCurrentSlotSlashed[_stakeHouse,_memberId] -= 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 StakeHouseRegistry.mintSLOTAndSharesBatch(stakeHouse,amount,_memberId) (Contracts/banking/StakeHouseRegistry.sol#120-132):
External calls:
- stakeHouse.mint(amount,_memberId) (Contracts/banking/StakeHouseRegistry.sol#128)
- - StakeHouseRegistry._stakeHouse._mint(_memberId) (Contracts/banking/StakeHouseRegistry.sol#124)
- buySlashedSlot(_stakeHouse,_memberId,_amount,_slasher) (Contracts/banking/SlotSettlementPool.sol#131)
- - stakeHouseMemberCurrentSlotSlashed[_stakeHouse,_memberId] -= 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)

```

```

rentrancyIn StakeHouseUniverse.superchargeAndInitCentral(address,address,address,address,address) (contracts/StakeHouseUniverse.sol#187-215):
    External calls:
        - msg.sender = new StakeHouseUpgradeableProxy(address, _lootFactoryLogic(this), abi.encodeWithSelector(_brandNftLogic(this), abi.encodeWithSelector(_brandNftLogic(this), abi.encodeWithSelector(_brandNftLogic(this), abi.encodeWithSelector(_brandNftLogic(this), _ekLootFactory(this), _claimAuction))))) (contracts/StakeHouseUniverse.sol#187-193)
        - StakeHouseUpgradeableProxy = new StakeHouseUpgradeableProxy(_brandCentralLogic,address,(this),abi.encodeWithSelector(BrandCentral._brandCentralLogic).init.selector,address(this),_brandNft,_ekLootFactory,_claimAuction)) (contracts/StakeHouseUniverse.sol#187-193)
useUniversalVariables:
    - state variables written after the call(s):
        - _brandCentral = BrandCentral(_brandCentralAddress) (contracts/StakeHouseUniverse.sol#208)
Reference: https://github.com/crytic/slither/vikit-Detector-Documentation#rentrancy-vulnerabilities-1

AccountManager.registerValidatorInitials(address,bytes,byte[],account) (contracts/accounts/AccountManager.sol#1) is a local variable never initialized
Reference: https://github.com/crytic/slither/vikit-Detector-Documentation#uninitialised-local-variables

ERC721Upgradable.checkERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradable.sol#376-397) ignores return value by ERC721ReceiverUpgradable(to),onERC721Received(_msgSender()),from,to,etc, data (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradable.sol#393-395)
ERC1967Upgrade.upgradeToAndCallSuccess(address,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#65-72) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#70)
ERC1967Upgrade.upgradeToAndCallSuccess(address,bytes,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#107-110) ignores return value by Address.functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#107)
ERC1967Upgrade.upgradeToAndCallSuccess(address,bytes,bytes,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#107-109) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature("upgradeTo(address,oldImplementation)"),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#107)
ERC1967Upgrade.upgradeToAndCallSuccess(address,bytes,bytes,bytes,bytes,bool) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#107-112) ignores return value by Address.functionDelegateCall(newImplementation,abi.encodeWithSignature("upgradeTo(address,oldImplementation)"),data) (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#107)
ERC721.checkERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by ERC721Receiver(to),onERC721Received(_msgSender()),from,to,tokenId,data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369)
ERC721.onERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by AccountManager.createStakehouse(_msgSender(),_stLootFactory,_ticker,_buildingType,_etc,etc) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390)
ETHTxDataReport.ETH2Validation(ETHDatasReport,ETHDatasReport,TransactionManager.sol#177-189) ignores return value by AccountManager.createStakehouse(_msgSender(),_stLootFactory,_ticker,_buildingType,_etc,etc) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390)
stLootFactory.openStLootAndDeployment(uint16,ekLootFactory,ekLootComponent,addees) (contracts/brand/stLootFactory.sol#203-246) ignores return value by stLoot.mint(string(abi.encodePacked({_coords.x,toString()},{_coords.y,toString()})),_brandNft,_ekLoot,_ekLootFactory,_ekLootComponent,_addees) (contracts/brand/stLootFactory.sol#203-246)
stLootFactory.openStLootAndDeployment(uint16,ekLootFactory,ekLootComponent,addees) (contracts/brand/stLootFactory.sol#203-246) ignores return value by stLoot.mint(string(abi.encodePacked({_tokenId}),_building,_ekLoot,_itemType,_buildingTokenId,_recipient) (contracts/brand/stLootFactory.sol#203-246)
stLootFactory.openStLootAndDeployment(uint16,ekLootFactory,ekLootComponent,addees) (contracts/brand/stLootFactory.sol#203-246) ignores return value by stLoot.mint(string(abi.encodePacked({_tokenId}),_character,_stLoot,_itemType,_character,_brandNft,_recipient) (contracts/brand/stLootFactory.sol#203-246)
Reference: https://github.com/crytic/slither/vikit-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.verifyResult(result, address) (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#15)
    - 0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#3)
    - 0.8.0 (node_modules/@openzeppelin/contracts/proxy/beacon/IBeacon.sol#8)
    - 0.8.0 (node_modules/@openzeppelin/contracts/proxy/beacon/Proxy.sol#12)
    - 0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#12)
    - 0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#8)
    - 0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#8)
    - 0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#8)
    - 0.8.9 (contracts/StakehouseAccessControls.sol#9)
    - 0.8.9 (contracts/proxies/UpgradeableBeacon.sol#9)
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.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
Message.sendValue(address, uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#144-146) is never used and should be removed
Address.functionDelegatecall(address, bytes, string) (node_modules/@openzeppelin/contracts/utils/Address.sol#170-187) is never used and should be removed
Address.functionStaticCall(address, bytes, string) (node_modules/@openzeppelin/contracts/utils/Address.sol#184-193) is never used and should be removed
Message.sendValue(address, uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#194-196) is never used and should be removed
Address.sendValue(address, uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-199) is never used and should be removed
Address.verifyCallResult(bool, bytes, string) (node_modules/@openzeppelin/contracts/utils/Address.sol#215-216) is never used and should be removed
String.toBytes(string) (node_modules/@openzeppelin/contracts/utils/String.sol#14-16) is never used and should be removed
String.toInt(string) (node_modules/@openzeppelin/contracts/utils/String.sol#17-19) is never used and should be removed
String.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/String.sol#24-26) is never used and should be removed
String.toUint(string) (node_modules/@openzeppelin/contracts/utils/String.sol#27-29) is never used and should be removed
String.toUInt(string) (node_modules/@openzeppelin/contracts/utils/String.sol#30-32) is never used and should be removed
String.toUInt256(string) (node_modules/@openzeppelin/contracts/utils/String.sol#33-35) is never used and should be removed
String.toUInt256(string) (node_modules/@openzeppelin/contracts/utils/String.sol#36-38) 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/access/IAccessControl.sol#11), 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/context/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/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/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/introspection/IERC165.sol#9) 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/introspection/ERC165.sol#9) 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/introspection/IERC165.sol#9) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc=0.8.0 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#154-159):
  - (address, recipient, callValue, amount) (node_modules/@openzeppelin/contracts/utils/Address.sol#157)
Low level call in Address.functionCall(address, bytes, uint256) (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{value: value}(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._admin(address) (contracts/StakeHouseAccessControls.sol#4), _admin is not in mixedCase
Parameter StakeHouseAccessControls._addCoreModule(address) (contracts/StakeHouseAccessControls.sol#10), _addCoreModule is not in mixedCase
Parameter StakeHouseAccessControls._coreModule(address) (contracts/StakeHouseAccessControls.sol#10), _coreModule is not in mixedCase
Parameter StakeHouseAccessControls._coreModuleManager(address) (contracts/StakeHouseAccessControls.sol#4), _coreModuleManager is not in mixedCase
Parameter StakeHouseAccessControls._lockCoreModule(address) (contracts/StakeHouseAccessControls.sol#12), _lockCoreModule is not in mixedCase
Parameter StakeHouseAccessControls._removeCoreModule(address) (contracts/StakeHouseAccessControls.sol#12), _removeCoreModule is not in mixedCase
Parameter StakeHouseAccessControls._removeCoreModuleManager(address) (contracts/StakeHouseAccessControls.sol#7), _removeCoreModuleManager is not in mixedCase
Parameter StakeHouseAccessControls._removeCoreModuleManager(address) (contracts/StakeHouseAccessControls.sol#12), _removeCoreModuleManager is not in mixedCase
Parameter StakeHouseAccessControls._removeCoreModuleAdmin(address) (contracts/StakeHouseAccessControls.sol#16), _removeCoreModuleAdmin is not in mixedCase
Parameter StakeHouseAccessControls._removeCoreModuleAdmin(address) (contracts/StakeHouseAccessControls.sol#16), _removeCoreModuleAdmin is not in mixedCase
Parameter UpgradeableBeacon.updateImplementation(address, implementation) (contracts/proxies/UpgradeableBeacon.sol#48), updateImplementation is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conventions-v0-solidity

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

Variable UpgradableBeacon.constructor(StakeHouseAccessControls,address, implementation) (contracts/proxies/UpgradeableBeacon.sol#18), constructor(StakeHouseAccessControls,address, implementation) is too similar to UpgradeableBeacon.implementation, (contracts/proxies/UpgradeableBeacon.sol#18)
Variable UpgradableBeacon.setImplementation(address, implementation) (contracts/proxies/UpgradeableBeacon.sol#40), setImplementation(address, implementation) is too similar to UpgradableBeacon.implementation, (contracts/proxies/UpgradeableBeacon.sol#40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

```

- 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-9483-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 randomness.

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 randomness.

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 randomness.
300	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

DRAFT

## 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.

