Axion Network

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **September 9th, 2021 - October 7th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	10/04/2021	Juned Ansari
0.2	Document Updates	10/05/2021	Juned Ansari
0.3	Document Updates	10/06/2021	Juned Ansari
0.4	Draft Review	10/06/2021	Gabi Urrutia
1.0	Remediation Plan	10/07/2021	Juned Ansari
1.1	Remediation Plan Review	10/07/2021	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
Juned Ansari	Halborn	Juned.Ansari@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Axion Network engaged Halborn to conduct a security assessment on their smart contracts v3 beginning on September 9th, 2021 and ending October 7th, 2021. Axion is an ethical, community-driven cryptocurrency that rewards long-term investing with high-yield interest rates and weekly dividends.

Though this security audit's outcome is satisfactory, only the most essential aspects were tested and verified to achieve objectives and deliverables set in the scope due to time and resource constraints. It is essential to note the use of the best practices for secure development.

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 to achieve the following:

- Ensure that all Nameless Contract functions are intended.
- Identify potential security issues with the assets in scope.

In summary, Halborn identified several security risk that were mostly addressed by Axion Network 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 Axion Network contract solidity 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 (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 : axion-contracts-v3 github repository
The security assessment was scoped to the following smart contract:

Listing 1: axion-contracts-v3-main

```
1 contracts/abstracts/
2 contracts/libs/AxionSafeCast.sol
3 contracts/stake/
4 contracts/enums/
5 contracts/v2.1/
6 contracts/DataReader.sol
7 contracts/Token.sol
8 contracts/accelerator/
9 contracts/interfaces/
```

OUT-OF-SCOPE : External libraries and economics attacks
FIXED-COMMIT-ID : 1c837d204115ef0511e148b24c724695f0c04b74

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	3	8	5

LIKELIHOOD

IMPACT

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

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - FRONT-RUNNING ATTACK ON INITIALIZATION FUNCTIONS	High	SOLVED - 10/06/2021
HAL02 - LACK OF INTEGER OVERFLOW PROTECTION	Medium	NOT APPLICABLE
HAL03 - UNCHECKED TRANSFER	Medium	SOLVED - 10/06/2021
HAL04 - MISSING RE-ENTRANCY PROTECTION	Low	SOLVED - 10/06/2021
HAL05 - MULTIPLE CALLS MAY LEADS TO DENIAL OF SERVICE(DOS)	Low	RISK ACCEPTED
HAL06 - EXTERNAL FUNCTION CALLS WITHIN LOOP	Low	RISK ACCEPTED
HAL07 - UNUSED RETURN	Low	PARTIALLY SOLVED - 10/06/2021
HAL08 - DIVIDE BEFORE MULTIPLY	Low	NOT APPLICABLE
HAL09 - MISSING ZERO-ADDRESS CHECK	Low	RISK ACCEPTED
HAL10 - USAGE OF BLOCK-TIMESTAMP	Low	NOT APPLICABLE
HAL11 - UNINITIALIZED VARIABLE	Low	SOLVED - 10/06/2021
HAL12 - USAGE OF STRICT-EQUALITIES	Informational	NOT APPLICABLE
HAL13 - PRAGMA TOO RECENT	Informational	ACKNOWLEDGED
HAL14 - MISSING EVENTS EMITTING	Informational	SOLVED - 10/06/2021
HAL15 - REDUNDANT BOOLEAN COMPARISON	Informational	SOLVED - 10/06/2021
HAL16 - POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	SOLVED - 10/06/2021

FINDINGS & TECH DETAILS

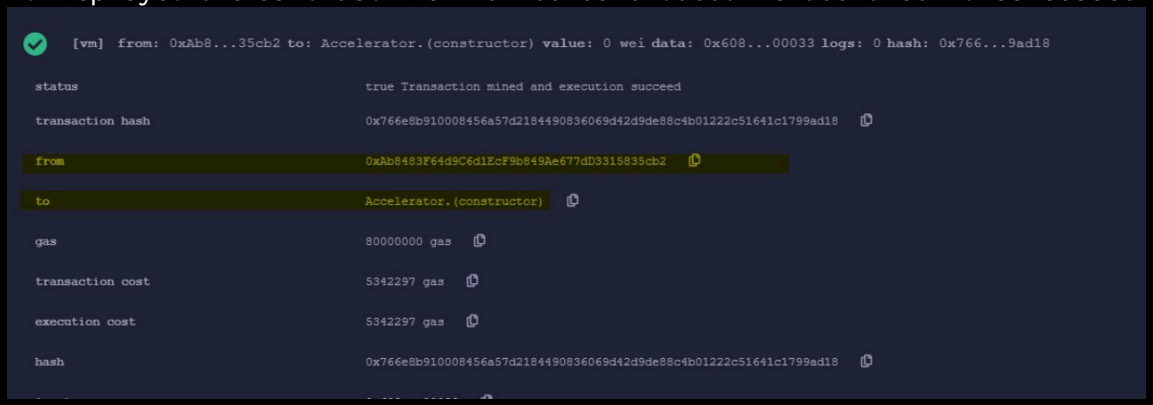
3.1 (HAL-01) FRONT-RUNNING ATTACK ON INITIALIZATION FUNCTIONS - MEDIUM

Description:

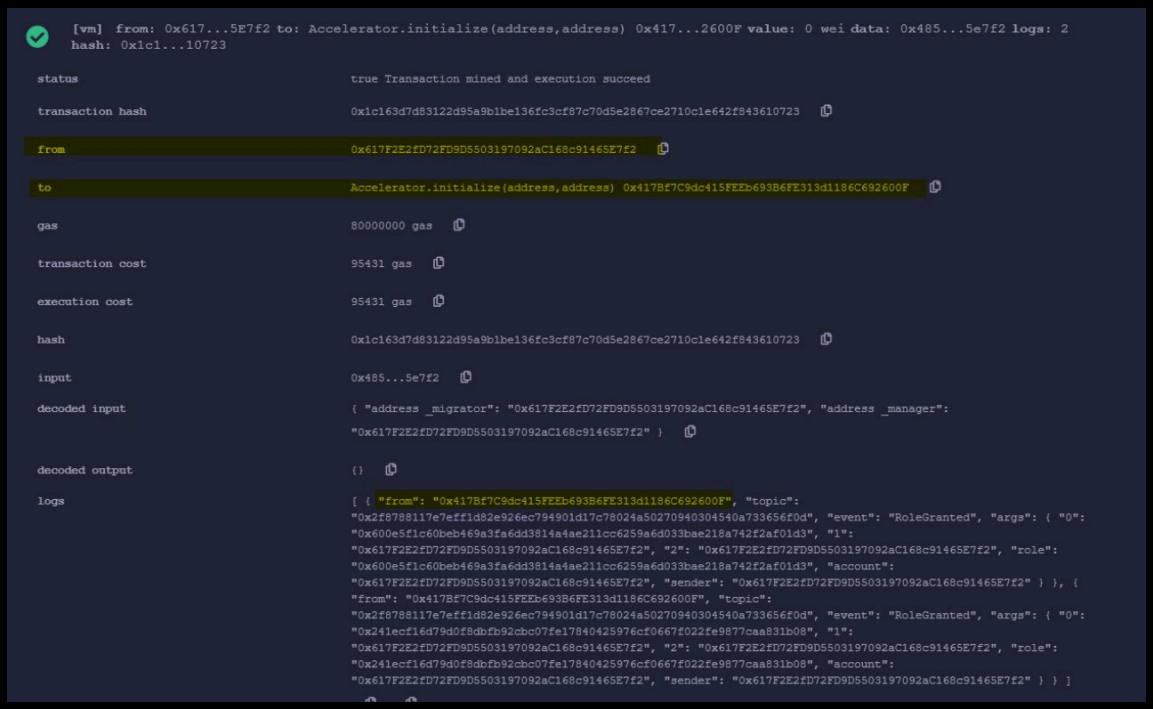
The declaration of `function initialize(address _manager, address _migrator)..` is used in almost all scope contracts. It is required a call to the `initialize` function after deploying it to initialize the `manager`, `migrator`, and `external_caller_role` roles. There is no `require` checking within the `initialize` function. There are functions that can be front-run, allowing an attacker to incorrectly initialize the contracts.

Attack scenario:

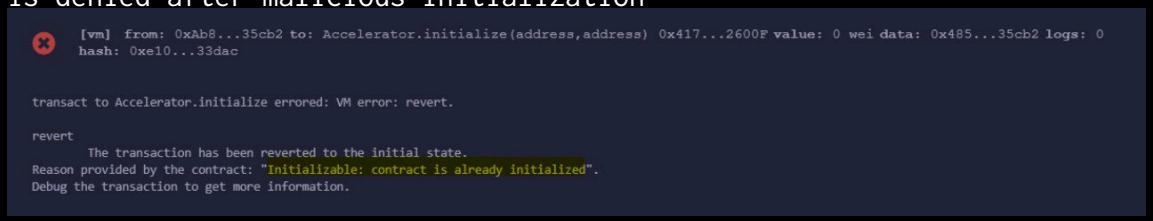
1. Deployed the contract from "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2"



2. Calling initialize function from "0x617F2E2fD72FD9D5503197092aC168c91465E7f2"



3. Call from owner address (“0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2”) is denied after malicious initialization



Code Location:

Listing 2: VentureCapital.sol (Lines 147,148,149,150,152)

```
257     function initialize(address _manager, address _migrator)
            public initializer {
258         _setupRole(MANAGER_ROLE, _manager);
259         _setupRole(MIGRATOR_ROLE, _migrator);
260
261         _setupRole(EXTERNAL_CALLER_ROLE, _manager);
262         _setupRole(EXTERNAL_CALLER_ROLE, _migrator);
263     }
```

Listing 3: Accelerator.sol (Lines 503,504)

```
500     function initialize(address _migrator, address _manager)
            external initializer {
501         /** Setup roles and addresses */
502         _setupRole(MIGRATOR_ROLE, _migrator);
503         _setupRole(MANAGER_ROLE, _manager);
504     }
```

Listing 4: BPD.sol

```
116     function initialize(address _migrator, address _stakeManager)
            external initializer {
117         _setupRole(MIGRATOR_ROLE, _migrator);
118         _setupRole(EXTERNAL_CALLER_ROLE, _stakeManager);
119     }
```

Listing 5: StakeBurner.sol

```
287     function initialize(address _manager, address _migrator)
            external initializer {
288         _setupRole(MANAGER_ROLE, _manager);
289         _setupRole(MIGRATOR_ROLE, _migrator);
290     }
```

Listing 6: StakeMinter.sol

```
213     function initialize(address _manager, address _migrator)
            external initializer {
214         _setupRole(MANAGER_ROLE, _manager);
215         _setupRole(MIGRATOR_ROLE, _migrator);
216     }
```

Listing 7: StakeReminter.sol

```
 85     function initialize(address _manager, address _migrator)
            external initializer {
 86         _setupRole(MANAGER_ROLE, _manager);
 87         _setupRole(MIGRATOR_ROLE, _migrator);
 88     }
```

Listing 8: StakeToken.sol

```
153     function initialize(
154         address _manager,
155         address _migrator,
156         string memory name,
157         string memory symbol
158     ) external initializer {
159         _setupRole(MANAGER_ROLE, _manager);
160         _setupRole(MIGRATOR_ROLE, _migrator);
161 
162         enabled = true; // Initially Enabled
163         transferEnabled = false; // Initially disabled
164         __ERC721_init(name, symbol);
165         __ERC721Enumerable_init();
166     }
```

Listing 9: StakeUpgrader.sol

```
151     function initialize(address _manager, address _migrator)
            external initializer {
152         _setupRole(MANAGER_ROLE, _manager);
153         _setupRole(MIGRATOR_ROLE, _migrator);
154     }
```

Listing 10: StakeCustodian.sol

```
45     function initialize(
46         address _migrator,
47         address _stakeMinter,
48         address _stakeBurner,
49         address _stakeUpgrader
50     ) external initializer {
51         _setupRole(MIGRATOR_ROLE, _migrator);
52         _setupRole(EXTERNAL_CALLER_ROLE, _stakeMinter);
53         _setupRole(EXTERNAL_CALLER_ROLE, _stakeBurner);
54         _setupRole(EXTERNAL_CALLER_ROLE, _stakeUpgrader);
55     }
```

Listing 11: StakeManager.sol

```
631     function initialize(address _manager, address _migrator)
            external initializer {
632         _setupRole(MANAGER_ROLE, _manager);
633         _setupRole(MIGRATOR_ROLE, _migrator);
634     }
```

Listing 12: Token.sol

```
46      function initialize(
47          address _manager,
48          address _migrator,
49          string memory _name,
50          string memory _symbol
51      ) public initializer {
52          _setupRole(MANAGER_ROLE, _manager);
53          _setupRole(MIGRATOR_ROLE, _migrator);
54          __ERC20_init(_name, _symbol);
55
56          /** I do not understand this */
57          swapIsOver = false;
58      }
```

Listing 13: DataReader.sol

```
46      function initialize(
47          address _manager,
48          address _staking,
49          address _stakingV1,
50          address _auction,
51          address _auctionV1
52      ) public initializer {
53          _setupRole(MANAGER_ROLE, _manager);
54
55          staking = IStakingDataV2(_staking);
56          stakingV1 = IStakingV1(_stakingV1);
57          auction = IAuctionDataV2(_auction);
58          auctionV1 = IAuctionV1(_auctionV1);
59      }
```

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

It is recommended to declare a constructor instead of an initialize function to set up roles at the time of deployment to mitigate the issue. Otherwise, add a require statement to each initialize function to verify that the function is called by the contract owner only, and post verification roles should be setup. Otherwise, setting the owner in the contract's constructor to the msg.sender and adding the onlyOwner modifier to all **initializers** would be enough for access control. Another solution is using a factory pattern that will deploy and initialize the contracts atomically to prevent front-running of the initialization.

Remediation Plan:

SOLVED: Values will be hardcoded by the Axion Network team.

3.2 (HAL-02) LACK OF INTEGER OVERFLOW PROTECTION - MEDIUM

Description:

The overflow happens when an arithmetic operation reaches the maximum size of a type. For instance in the VentureCapital.sol contract on getTokenInterestEarned method, multiplication of contracts.stakingV2.getTotalSharesOf(accountAddress)*tokenPricePerShare[tokenAddress] in the return calculation on the interest earned by an address for a specific dividend token may end up overflowing the integer. In computer programming, an integer overflow occurs when an arithmetic operation attempts to create a numeric value that is outside of the range that can be represented with a given number of bits -- either larger than the maximum or lower than the minimum re-presentable value.

Code Location:

Listing 14: VentureCapital.sol (Lines 310,311)

```
303     function getTokenInterestEarned(address accountAddress,
            address tokenAddress)
304         external
305         view
306         returns (uint256)
307     {
308         if (isVcaRegistered[accountAddress] == false) {
309             return
310                 ((contracts.stakingV2.getTotalSharesOf(
                        accountAddress) *
311                     tokenPricePerShare[tokenAddress]) -
312                     contracts.stakingV2.getDeductBalances(
                            accountAddress, tokenAddress)) / 1e36;
313         }
314
315         return getTokenInterestEarnedInternal(accountAddress,
                tokenAddress);
316     }
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Currently not all the smart contracts and the operations within them are using the SafeMath library which makes some operations vulnerable to overflows/underflows. It is recommended to use the SafeMath library for arithmetic operations consistently throughout **ALL** the mathematical operations in the smart contract system.

Reference:

Ethereum Smart Contract Best Practices - Integer Overflow and Underflow

Remediation Plan:

NOT APPLICABLE: The Axion Network team claims that due to their use of Pragma > 0.8.0 safe math is not necessary, the run time will fail if there is an overflow.

3.3 (HAL-03) UNCHECKED TRANSFER - MEDIUM

Description:

In contract Token.sol, StakeManager.sol, VentureCapital.sol, Accelerator.sol, and StakingV21.sol the return value of some external transfer/transferFrom calls are not checked. Several tokens do not revert in case of failure and return false. If one of these tokens is used, a deposit would not revert if the transfer fails, and an attacker could deposit tokens for free.

Code Location:

Listing 15: Token.sol (Lines 121)

```
116     function recovery(
117         address recoverFor,
118         address tokenToRecover,
119         uint256 amount
120     ) external onlyMigrator {
121         IERC20(tokenToRecover).transfer(recoverFor, amount);
122     }
123 }
```

Listing 16: StakeManager.sol (Lines 504,505,506,507)

```
502     function getTodaysInterest() internal returns (uint256) {
503         uint256 amountTokenInDay = IERC20Upgradeable(contracts.
                token).balanceOf(address(this));
504         IERC20Upgradeable(contracts.token).transfer(
505             0x000000000000000000000000000000000000dEaD,
506             amountTokenInDay
507         );
```

Listing 17: VentureCapital.sol (Lines 136)

```
135         if (tokenAddress != address(0
                xFFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfF)) {
136             IERC20Upgradeable(tokenAddress).transfer(to,
                    tokenInterestEarned);
137         } else {
138             to.transfer(tokenInterestEarned);
139         }
```

Listing 18: VentureCapital.sol (Lines 147,148,149,150)

```
144     function withdrawOriginDivTokens(address tokenAddress)
            external onlyExternalCaller {
145         /** 0xFF... is our ethereum placeholder address */
146         if (tokenAddress != address(0
                xFFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfF)) {
147             IERC20Upgradeable(tokenAddress).transfer(
148                 msg.sender,
149                 originWithdrawableTokenAmounts[tokenAddress]
150             );
151         } else {
```

Listing 19: Accelerator.sol (Lines 193)

```
192         /** Transfer tokens to contract */
193         IERC20(_token).transferFrom(msg.sender, address(this),
                _amount);
```

Listing 20: Accelerator.sol (Lines 222)

```
221         //** Transfer tokens to Manager */
222         IERC20(_token).transfer(recipient, _recipientAmount);
```

Listing 21: StakingV21.sol (Lines 116,123)

```
111     function transferTokens(address vcAuction, address
            stakeManager) external onlyMigrator {
112         for (uint8 i = 0; i < divTokens.length(); i++) {
113             if (divTokens.at(i) != address(0
                    xFFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfF)) {
```

```
114                 IERC20Upgradeable token = IERC20Upgradeable(
                        divTokens.at(i));
115
116                 token.transfer(vcAuction, token.balanceOf(address(
                        this)));
117             } else {
118                 payable(vcAuction).transfer(address(this).balance)
                        ;
119             }
120         }
121
122         IERC20Upgradeable axn = IERC20Upgradeable(addresses.
                mainToken);
123         axn.transfer(stakeManager, axn.balanceOf(address(this)));
124     }
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

It is recommended to use SafeERC20, or ensure that the transfer/transferFrom return value is checked.

Remediation Plan:

SOLVED: The Axion Network team solved the issue by using SafeERC20 implementation and added the safetransfer function to the code.

3.4 (HAL-04) MISSING RE-ENTRANCY PROTECTION - LOW

Description:

It was identified that axion-contracts-v3 are missing nonReentrant guard. In VentureCapital.sol, function withdrawOriginDivTokens, contract StakeReminter.sol function remintStakeInternal, and contract StakeMinter.so function convertToNft are missing nonReentrant guard. Also, in these functions, external calls are called before all state changes are resolved, and read/write to persistent state following external call, making it vulnerable to a Reentrancy attack.

Although administrative restrictions are imposed but to protect against cross-function reentrancy attacks, it may be necessary to use a mutex. By using this lock, an attacker can no longer exploit the function with a recursive call. OpenZeppelin has it's own mutex implementation called ReentrancyGuard which provides a modifier to any function called "nonReentrant" that guards the function with a mutex against the Reentrancy attacks.

Code Location:

Listing 22: VentureCapital.sol (Lines 147,148,149,150,152)

```
144     function withdrawOriginDivTokens(address tokenAddress)
            external onlyExternalCaller {
145         /** 0xFF... is our ethereum placeholder address */
146         if (tokenAddress != address(0
                xFFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfF)) {
147             IERC20Upgradeable(tokenAddress).transfer(
148                 msg.sender,
149                 originWithdrawableTokenAmounts[tokenAddress]
150             );
151         } else {
152             payable(msg.sender).transfer(
                    originWithdrawableTokenAmounts[tokenAddress]);
153         }
```

```
154
155         originWithdrawableTokenAmounts[tokenAddress] = 0;
156     }
```

Listing 23: StakeReminter.sol (Lines 80,81)

```
71      function remintStakeInternal(
72          uint256 payout,
73          uint256 topup,
74          uint256 stakingDays
75      ) internal {
76          if (topup != 0) {
77              payout = payout + topup;
78          }
79
80          contracts.token.burn(msg.sender, payout); // Burn the
                payout amount before restaking
81          contracts.stakeMinter.externalStake(payout, stakingDays,
                msg.sender);
82      }
```

Listing 24: StakeMinter.sol (Lines 100)

```
94      function convertToNft(uint256 stakeId) external {
95          require(
96              contracts.stakeCustodian.removeStake(msg.sender,
                    stakeId),
97              'STAKE MINTER: Not owner of stake or already converted
                    .'
98          );
99
100         contracts.stakeToken.mint(msg.sender, stakeId); // 120k
101     }
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

Change the code to follow the checks-effects-interactions pattern and use ReentrancyGuard through the nonReentrant modifier.

Remediation Plan:

SOLVED: The Axion Network team claims that
- Listing 22: This code no longer exists in their not-backwards branch.
- Listing 23: Before calling external stake they burn the users token, thus re-entrancy would not benefit a hacker.
- Listing 24: This would result in reminting the same stake, but removeStake is called first, the stake would not exist thus re-entrancy should not be a problem.

3.5 (HAL-05) MULTIPLE CALLS MAY LEADS TO DENIAL OF SERVICE(DOS) - LOW

Description:

In contract StakeMinter.sol, StakeReminter.sol, and VentureCapital.sol multiple calls are executed in the same transaction. This call is executed following another call within the same transaction. It is possible that the call never gets executed if a prior call fails permanently and it may leads to DOS. This might be caused intentionally by a malicious user.

Code Location:

Listing 25: VentureCapital.sol (Lines 310)

```
303     function getTokenInterestEarned(address accountAddress,
            address tokenAddress)
304         external
305         view
306         returns (uint256)
307     {
308         if (isVcaRegistered[accountAddress] == false) {
309             return
310                 ((contracts.stakingV2.getTotalSharesOf(
                        accountAddress) *
311                     tokenPricePerShare[tokenAddress]) -
312                     contracts.stakingV2.getDeductBalances(
                            accountAddress, tokenAddress)) / 1e36;
313         }
314
315         return getTokenInterestEarnedInternal(accountAddress,
                tokenAddress);
316     }
```

Listing 26: StakeMinter.sol (Lines 100)

```
94      function convertToNft(uint256 stakeId) external {
95          require(
96              contracts.stakeCustodian.removeStake(msg.sender,
                    stakeId),
97              'STAKE MINTER: Not owner of stake or already converted
                    .'
98          );
99
100         contracts.stakeToken.mint(msg.sender, stakeId); // 120k
101     }
```

Listing 27: StakeReminter.sol (Lines 46)

```
42          uint256 end = contracts.stakeManager.getStakeEnd(stakeId);
43
44          require(end != 0 && end <= block.timestamp, 'RESTAKER:
                Stake not mature or not set.');
45
46          uint256 payout = contracts.stakeBurner.externalBurnStake(
                stakeId, msg.sender);
47
48          remintStakeInternal(payout, topup, stakingDays);
```

Listing 28: StakeReminter.sol (Lines 80,81)

```
76          if (topup != 0) {
77              payout = payout + topup;
78          }
79
80          contracts.token.burn(msg.sender, payout); // Burn the
                payout amount before restaking
81          contracts.stakeMinter.externalStake(payout, stakingDays,
                msg.sender);
82      }
```

Risk Level:

Likelihood - 1

Impact - 4

Recommendation:

If possible, Refactor the code such that each transaction only executes one external call or make sure that all users can be trusted (i.e. they're part of your own codebase).

Remediation Plan:

RISK ACCEPTED: The Axion Network team accepts the risk and claims that this issue only affects their backwards compatibility, issues listed without backwards compatibility do not apply. Backwards compatibility has been removed in the not-backwards branch.

3.6 (HAL-06) EXTERNAL FUNCTION CALLS WITHIN LOOP - LOW

Description:

Calls inside a loop increase Gas usage or might lead to a denial-of-service attack. In one of the functions discovered there is a for loop on variable `i` that iterates up to the `divTokens` and `v2DivTokens` array length. If this integer is evaluated at extremely large numbers this can cause a DoS.

Code Location:

Listing 29: VentureCapital.sol (Lines 73,74,75,76)

```
57     function ensureIsVcaRegisteredInternal(address staker)
           internal {
58         if (isVcaRegistered[staker] == false) {
59             if (contracts.stakingV2.getIsVCARegistered(staker) ==
                   false) {
60                 uint256 totalShares = contracts.stakingV2.
                       resolveTotalSharesOf(staker);
61
62                 totalSharesOf[staker] = totalShares;
63                 contracts.stakeManager.addTotalVcaRegisteredShares
                       (totalShares);
64
65                 for (uint256 i = 0; i < divTokens.length(); i++) {
66                     deductBalances[staker][divTokens.at(i)] = (
                           totalShares *
67                         tokenPricePerShare[divTokens.at(i)])
68                         .toInt256();
69                 }
70             } else {
71                 totalSharesOf[staker] = contracts.stakingV2.
                       getTotalSharesOf(staker);
72                 for (uint256 i = 0; i < divTokens.length(); i++) {
73                     deductBalances[staker][divTokens.at(i)] =
                           contracts
74                         .stakingV2
```

```
75                          .getDeductBalances(staker, divTokens.at(i)
                                )
76                          .toInt256();
77                  }
78              }
79
80              isVcaRegistered[staker] = true;
81          }
82      }
```

Listing 30: VentureCapital.sol (Lines 286,287,288)

```
282         address[] memory v2DivTokens = contracts.stakingV2.
                getDivTokens();
283
284         for (uint256 i = 0; i < v2DivTokens.length; i++) {
285             divTokens.add(v2DivTokens[i]);
286             tokenPricePerShare[v2DivTokens[i]] = contracts.
                    stakingV2.getTokenPricePerShare(
287                 v2DivTokens[i]
288             );
289         }
290     }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

If possible, use pull over push strategy for external calls.

Reference:

External Calls Recommendation

Remediation Plan:

RISK ACCEPTED: The Axion Network team accepts the risk and claims that this issue only affects their backwards compatibility, issues listed without backwards compatibility do not apply. Backwards compatibility has been removed in the not-backwards branch.

3.7 (HAL-07) UNUSED RETURN - LOW

Description:

The return value of an external call is not stored in a local or state variable. In contract StakeBurner.sol, StakeMinter.sol, StakeUpgrader.sol, VentureCapital.sol, Accelerator.sol, and StakingV21.sol, there are instances where external methods are being called and return value are being ignored.

Code Location:

Listing 31: StakeBurner.sol (Lines 196)

```
195         // Add to stake custodian as the v1 or v2 stake is now a
                v3 stake that has been withdrawn
196         contracts.stakeCustodian.addStake(staker, sessionId);
197
198         return payout;
199     }
```

Listing 32: StakeMinter.sol (Lines 84,85,86,87)

```
79      function stakeInternal(
80          uint256 amount,
81          uint256 stakingDays,
82          address staker
83      ) internal {
84          contracts.stakeCustodian.addStake(
85              staker,
86              contracts.stakeManager.createStake(staker, amount,
                    stakingDays)
87          );
88      }
```

Listing 33: StakeUpgrader.sol (Lines 112)

```
110             })
111         );
```

```
112         contracts.stakeCustodian.addStake(msg.sender, sessionId);
113     }
```

Listing 34: VentureCapital.sol (Lines 253)

```
252     function addDivToken(address tokenAddress) external override
            onlyExternalCaller {
253         divTokens.add(tokenAddress);
254     }
```

Listing 35: VentureCapital.sol (Lines 285)

```
284         for (uint256 i = 0; i < v2DivTokens.length; i++) {
285             divTokens.add(v2DivTokens[i]);
286             tokenPricePerShare[v2DivTokens[i]] = contracts.
                    stakingV2.getTokenPricePerShare(
287                 v2DivTokens[i]
288             );
289         }
290     }
```

Listing 36: Accelerator.sol (Lines 268)

```
266         /** Check allowance */
267         if (IERC20(_tokenInAddress).allowance(address(this),
                uniswap) < 2**255) {
268             IERC20(_tokenInAddress).approve(uniswap, 2**255);
269         }
```

Listing 37: StakingV21.sol (Lines 118)

```
113             if (divTokens.at(i) != address(0
                    xFFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfF)) {
114                 IERC20Upgradeable token = IERC20Upgradeable(
                        divTokens.at(i));
115
116                 token.transfer(vcAuction, token.balanceOf(address(
                        this)));
117             } else {
118                 payable(vcAuction).transfer(address(this).balance)
                        ;
```

```
119             }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

Ensure that all the return values of the function calls are used. Add return value check to avoid unexpected crash of the contract. Return value check will help in handling the exceptions better way.

Remediation Plan:

PARTIALLY SOLVED: The Axion Network team solved the issue of Listing 32, and accepts the risk of Listing 34 and Listing 36. Further, Axion Network team claims that Listing 31, Listing 35 and Listing 31 only affects their backwards compatibility, issues listed without backwards compatibility do not apply. Backwards compatibility has been removed in the not-backwards branch.

3.8 (HAL-08) DIVIDE BEFORE MULTIPLY - LOW

Description:

Solidity integer division might truncate. As a result, the loss of precision can sometimes be avoided by multiplying before division, although the manual implementation of the precision/decimal calculation is being taken care of by the developer. In this audit, there are multiple instances found where division is being performed before multiplication operation in contract file.

Code Location:

Listing 38: BPD.sol (Lines 89)

```
88          for (uint256 i = bpdInterval[0]; i < bpdInterval[1]; i++)
                {
89              bpdAmount += (shares / bpdShares[i]) * (uint256(
                    bpdPools[i]) * 1e8); // x 1e8 since we have one
                    decimal
90          }
```

Listing 39: StakeManager.sol (Lines 199,200,201,202)

```
119         addToGlobalTotals(
120             newAmount - (stakeUpgrade.amount / 1e12) * 1e12,
121             newShares - (stakeUpgrade.shares / 1e12) * 1e12
122         );
```

Listing 40: StakeManager.sol (Lines 240)

```
539         uint256 shares = (numerator * 1e18) / denominator;
540         return (shares / 1e12) * 1e12;
541     }
```

Listing 41: StakeManager.sol (Lines 528,529,530,537,538,539,540,541)

```
525     function updateShareRate(uint256 _payout) internal {
526         uint256 currentTokenTotalSupply = contracts.token.
                totalSupply(); // 71848521428571428571428571428571
527
528         uint256 growthFactor =
529             (_payout * 1e18) /
530                 (currentTokenTotalSupply + (uint256(statFields.
                        totalStakedAmount) * 1e12) + 1); //we calculate
                         the total AXN supply as circulating + staked
531
532         if (settings.shareRateScalingFactor == 0) {
533             //use a shareRateScalingFactor which can be set in
                    order to tune the speed of shareRate increase
534             settings.shareRateScalingFactor = 1e18;
535         }
536
537         interestFields.shareRate = (
538             ((uint256(interestFields.shareRate) *
539                 (1e36 + (uint256(settings.shareRateScalingFactor)
                        * growthFactor))) / 1e36)
540         )
541             .toUint128(); //1e18 used for precision.
542     }
```

Listing 42: Accelerator.sol (Lines 304,305,320)

```
303         //** Add additional axion if stake length is greater then
                1year */
304         uint256 payout = (100 * _axionBought) / splitAmounts[0];
305         payout = payout + (payout * baseBonus) / 100;
306         if (_days >= bonusStartDays && bought[_currentDay] <
                maxBoughtPerDay) {
307             // Get amount for sale left
308             uint256 payoutWithBonus = maxBoughtPerDay - bought[
                    _currentDay];
309             // Add to payout
310             bought[_currentDay] += payout;
311             if (payout > payoutWithBonus) {
312                 uint256 payoutWithoutBonus = payout -
                        payoutWithBonus;
313
314                 payout =
```

```
315                     (payoutWithBonus +
316                         (payoutWithBonus * ((_days /
                               bonusStartDays) + bonusStartPercent)) /
317                         100) +
318                     payoutWithoutBonus;
319             } else {
320                 payout = payout + (payout * ((_days /
                        bonusStartDays) + bonusStartPercent)) / 100; //
                         multiply by percent divide by 100
321             }
322         } else {
323             //** If not returned above add to bought and return
                    payout. */
324             bought[_currentDay] += payout;
325         }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

Consider doing multiplication operation before division to prevail precision in the values in non floating data type. It is recommended to use SafeMath.sol.

Remediation Plan:

NOT APPLICABLE: The Axion Network team accepts the risk of Listing 42 and claims that they remove precision to allow for their stakes to be a single word struct. Further, Axion Network team claims
- Listing 38: BPD Shares have 0 decimal precision
- Listing 39: Amount and shares have 6 decimal precision
- Listing 40: Shares have 6 decimal precision
- Listing 41: only affects their backwards compatibility, issues listed without backwards compatibility do not apply. Backwards compatibility has been removed in the not-backwards branch.

FINDINGS & TECH DETAILS

3.9 (HAL-09) MISSING ZERO-ADDRESS CHECK - LOW

Description:

There are multiple instances found where Address validation is missing. Lack of zero address validation has been found when assigning user supplied address values to state variables directly. In Accelerator.sol contract function setRecipient lacks a zero-check on _recipient, function setToken lacks a zero-check on _token, function setVentureCapital lacks a zero-check on _ventureCapital, function setStaking lacks a zero-check on _staking, function setStakeManager lacks a zero-check on _stakeManager , and function startAddresses lacks a zero-check on _staking, _axion, _token, _uniswap and _recipient. In StakingV21.sol contract function transferTokens lacks zero address check on payable(vcAuction).transfer(address(this).balance).

Code Location:

Listing 43: Accelerator.sol (Lines 448)

```
447     function setRecipient(address payable _recipient) external
            onlyManager {
448         recipient = _recipient;
449     }
```

Listing 44: Accelerator.sol (Lines 462)

```
461     function setToken(address _token) external onlyManager {
462         token = _token;
463         IVentureCapital(ventureCapital).addDivToken(_token);
464     }
```

Listing 45: Accelerator.sol (Lines 470)

```
469     function setVentureCapital(address _ventureCapital) external
            onlyManager {
```

```
470         ventureCapital = _ventureCapital;
471     }
```

Listing 46: Accelerator.sol (Lines 477)

```
476     function setStaking(address _staking) external onlyManager {
477         staking = _staking;
478     }
```

Listing 47: Accelerator.sol (Lines 484)

```
484     function setStakeManager(address _stakeManager) external
            onlyManager {
485         stakeManager = _stakeManager;
486     }
```

Listing 48: Accelerator.sol (Lines 513,514,515,516,517)

```
506 function startAddresses(
507         address _staking,
508         address _axion,
509         address _token,
510         address payable _uniswap,
511         address payable _recipient
512     ) external onlyMigrator {
513         staking = _staking;
514         axion = _axion;
515         token = _token;
516         uniswap = _uniswap;
517         recipient = _recipient;
518     }
```

Listing 49: StakingV21.sol (Lines 118)

```
111     function transferTokens(address vcAuction, address
            stakeManager) external onlyMigrator {
112         for (uint8 i = 0; i < divTokens.length(); i++) {
113             if (divTokens.at(i) != address(0
                    xFFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfFfF)) {
114                 IERC20Upgradeable token = IERC20Upgradeable(
                        divTokens.at(i));
```

```
115
116                 token.transfer(vcAuction, token.balanceOf(address(
                        this)));
117             } else {
118                 payable(vcAuction).transfer(address(this).balance)
                        ;
119             }
120         }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

Although administrative restrictions are imposed to this function due to the OpenZeppelin RBAC it is better to add proper address validation when assigning a value to a variable from user supplied inputs.

Remediation Plan:

RISK ACCEPTED: The Axion Network team accepts the risk.

3.10 (HAL-10) USAGE OF BLOCK-TIMESTAMP - LOW

Description:

During a manual review, usage of block.timestamp in StakeBurner.sol, StakeManager.sol, StakeReminter.sol, and StakingV21.sol were observed. The contract developers should be aware that this does not mean current time. now is an alias for block.timestamp. The value of block.timestamp can be influenced by miners to a certain degree, so the testers should be warned that this may have some risk if miners collude on time manipulation to influence the price oracles. Miners can influence the timestamp by a tolerance of 900 seconds.

Code Location:

Listing 50: StakeBurner.sol (Lines 151,152,153,154)

```
151                 require(
152                     end != 0 && end <= block.timestamp,
153                     'STAKE BURNER: stake not mature or not set.'
154                 );
155             }
```

Listing 51: StakeBurner.sol (Lines 168,169,170,171)

```
168                 require(
169                     end != 0 && end <= block.timestamp,
170                     'STAKE BURNER: stake not mature or not set.'
171                 );
172             }
```

Listing 52: StakeBurner.sol (Lines 220)

```
220         if (stakingDays > daysStaked) {
221             uint256 payOutAmount = (amountAndInterest *
                    secondsStaked) / stakingSeconds;
```

Listing 53: StakeBurner.sol (Lines 227)

```
227         } else if (daysStaked < stakingDays + 14) {
228             return (amountAndInterest, 0);
```

Listing 54: StakeBurner.sol (Lines 230)

```
230         } else if (daysStaked < stakingDays + 714) {
231             return (amountAndInterest, 0);
```

Listing 55: StakeBurner.sol (Lines 279)

```
279         if (payout != 0) {
280             contracts.token.mint(staker, payout);
```

Listing 56: StakeBurner.sol (Lines 269,270,271,272,273)

```
269             interest += contracts.bpd.getBpdAmount(
270                 shares,
271                 start,
272                 block.timestamp < intendedEnd ? block.timestamp :
                        intendedEnd
273             );
274         }
```

Listing 57: StakeManager.sol (Lines 179,180,181,182)

```
179         require(
180             newShares > stakeUpgrade.shares,
181             'STAKING: New shares are not greater then previous
                    shares'
182         );
183
```

Listing 58: StakeManager.sol (Lines 169,170,171,172,173)

```
169             newAmount += contracts.bpd.getBpdAmount(
170                 stakeUpgrade.shares,
171                 stakeUpgrade.start,
```

```
172                 block.timestamp < intendedEnd ? block.timestamp :
                        intendedEnd
173             );
174         }
```

Listing 59: StakeManager.sol (Lines 269)

```
269         if (block.timestamp >= interestFields.
                nextAddInterestTimestamp) addDailyInterest();
270
```

Listing 60: StakeManager.sol (Lines 364)

```
364         if (block.timestamp >= interestFields.
                nextAddInterestTimestamp) addDailyInterest();
365
```

Listing 61: StakeManager.sol (Lines 414)

```
414         if (interestPerShare.length != 0) {
415             lastInterest = interestPerShare[
```

Listing 62: StakeManager.sol (Lines 466,467,468,469)

```
466         require(
467             block.timestamp >= interestFields.
                    nextAddInterestTimestamp,
468             'Staking: Too early to add interest.'
469         );
470         uint256 todaysSharePayout; // free
```

Listing 63: StakeManager.sol (Lines 472)

```
472         if (statFields.sharesTotalSupply == 0) {
473             statFields.sharesTotalSupply = 1e6;
```

Listing 64: StakeReminter.sol (Lines 44)

```
44         require(end != 0 && end <= block.timestamp, 'RESTAKER:
               Stake not mature or not set.');
```

Listing 65: StakingV21.sol (Lines 256,263,266)

```
243     function getAmountOutAndPenalty(
244         uint256 amount,
245         uint256 start,
246         uint256 end,
247         uint256 stakingInterest
248     ) public view returns (uint256, uint256) {
249         uint256 stakingSeconds = end.sub(start);
250         uint256 stakingDays = stakingSeconds.div(stepTimestamp);
251         uint256 secondsStaked = block.timestamp.sub(start);
252         uint256 daysStaked = secondsStaked.div(stepTimestamp);
253         uint256 amountAndInterest = amount.add(stakingInterest);
254
255         // Early
256         if (stakingDays > daysStaked) {
257             uint256 payOutAmount = amountAndInterest.mul(
                    secondsStaked).div(stakingSeconds);
258
259             uint256 earlyUnstakePenalty = amountAndInterest.sub(
                    payOutAmount);
260
261             return (payOutAmount, earlyUnstakePenalty);
262             // In time
263         } else if (daysStaked < stakingDays.add(14)) {
264             return (amountAndInterest, 0);
265             // Late
266         } else if (daysStaked < stakingDays.add(714)) {
267             return (amountAndInterest, 0);
```

Risk Level:

Likelihood - 2

Impact - 3

Recommendation:

Use block.number instead of block.timestamp or now to reduce the risk of MEV attacks. Check if the timescale of the project occurs across years, days and months rather than seconds. If possible, it is recommended to use Oracles.

Remediation Plan:

NOT APPLICABLE: The Axion Network team claims that the time required is over 900 seconds.

3.11 (HAL-11) UNINITIALIZED VARIABLE - LOW

Description:

On the VentureCapital.sol contract, state variable axion is not initialized, by default it holds 0x0 address, and variable is considered on the other calculation progresses in function updateTokenPricePerShare. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability.

Code Location:

Listing 66: VentureCapital.sol (Lines 39)

```
39     address public axion;
40     Contracts internal contracts;
41 }
```

Listing 67: VentureCapital.sol (Lines 242)

```
236     function updateTokenPricePerShare(address tokenAddress,
            uint256 amountBought)
237         external
238         payable
239         override
240         onlyExternalCaller
241     {
242         if (tokenAddress != axion) {
243             tokenPricePerShare[tokenAddress] =
244                 tokenPricePerShare[tokenAddress] + //increase the
                        token price per share with the amount bought
                        divided by the total Vca registered shares
245                 (amountBought * (1e36)) /
246                 (contracts.stakeManager.
                        getTotalVcaRegisteredShares() + 1e12);
247         }
248     }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendations:

If is recommended to initialize all internal variables on the same function, either on the constructor or a custom `init` method. However, using uninitialized variables and expecting them to have a value could cause unexpected behaviours on the execution flow.

Remediation Plan:

SOLVED: The Axion Network team solved the issue by adding and initializing `axion` in the manager controllable `init()` function, also declared `axion` as internal.

3.12 (HAL-12) USAGE OF STRICT-EQUALITIES - INFORMATIONAL

Description:

Use of strict equalities that can be easily manipulated by an attacker.

Code Location:

Listing 68: StakeManager.sol (Lines 472)

```
465     function addDailyInterest() public {
466         require(
467             block.timestamp >= interestFields.
                    nextAddInterestTimestamp,
468             'Staking: Too early to add interest.'
469         );
470         uint256 todaysSharePayout; // free
471         uint256 interest = getTodaysInterest();
472         if (statFields.sharesTotalSupply == 0) {
473             statFields.sharesTotalSupply = 1e6;
474         } // Is this necessary? cost 1000 gas for the if statement
                , 212832.. Only needed for testing?
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendations:

Don’t use strict equality to determine if an account has enough Ether or tokens.

Remediation Plan:

NOT APPLICABLE: The Axion Network team claims that they are checking a contract owned variable, not ether or tokens of a user.

3.13 (HAL-13) PRAGMA TOO RECENT - INFORMATIONAL

Description:

Axion Network in-scope main branch contract uses one of the latest pragma version (0.8.0) which was released on December 16, 2020. The latest pragma version (0.8.7) was released in August 2021. Many pragma versions have been lately released, going from version 0.7.x to the recently released version 0.8.x. in just 6 months.

Reference: https://github.com/ethereum/solidity/releases

In the Solitidy Github repository, there is a json file where are all bugs finding in the different compiler versions. It should be noted that pragma 0.6.12 and 0.7.6 are widely used by Solidity developers and have been extensively tested in many security audits.

Reference: https://github.com/ethereum/solidity/blob/develop/docs/bugs_by_version.json

Code Location:

Listing 69: (Lines 3)

```
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity >=0.8.0;
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendations:

If possible, consider using the latest stable pragma version that has been thoroughly tested to prevent potential undiscovered vulnerabilities such as pragma between 0.6.12 - 0.7.6.

Remediation Plan:

ACKNOWLEDGED: The Axion Network team accepts the risk and continues using pragma version 0.8.0.

3.14 (HAL-14) MISSING EVENTS EMITTING - INFORMATIONAL

Description:

It has been observed that important functionality is missing emitting event for some functions on the Accelerator.sol contract. These functions should emit events. Events are a method of informing the transaction initiator about the actions taken by the called function. It logs its emitted parameters in a specific log history, which can be accessed outside of the contract using some filter parameters. These functions should emit events.

Code Location:

Listing 70: Accelerator.sol (Lines 403)

```
402     function setMinStakeDays(uint256 _days) external onlyManager {
403         minStakeDays = _days;
404     }
```

Listing 71: Accelerator.sol (Lines 420)

```
419     function setMaxBoughtPerDay(uint256 _amount) external
            onlyManager {
420         maxBoughtPerDay = _amount;
421     }
```

Listing 72: Accelerator.sol (Lines 427)

```
426     function setBaseBonus(uint8 _amount) external onlyManager {
427         baseBonus = _amount;
428     }
```

Listing 73: Accelerator.sol (Lines 434)

```
433     function setBonusStartPercent(uint8 _amount) external
            onlyManager {
434         bonusStartPercent = _amount;
435     }
```

Listing 74: Accelerator.sol (Lines 441)

```
440     function setBonusStartDays(uint16 _amount) external
            onlyManager {
441         bonusStartDays = _amount;
442     }
```

Listing 75: Accelerator.sol (Lines 455)

```
454     function setStart(uint256 _start) external onlyManager {
455         start = _start;
456     }
```

Listing 76: Accelerator.sol (Lines 533,534,535,536,537,538,539)

```
533     function startVariables(
534         uint256 _minStakeDays,
535         uint256 _start,
536         uint256 _secondsInDay,
537         uint256 _maxBoughtPerDay,
538         uint8 _bonusStartPercent,
539         uint16 _bonusStartDays,
540         uint8 _baseBonus,
541         uint8[3] calldata _splitAmounts
542     ) external onlyMigrator {
543         uint8 total = _splitAmounts[0] + _splitAmounts[1] +
                _splitAmounts[2];
544         require(total == 100, 'ACCELERATOR: Split Amounts must ==
                100');
545
546         minStakeDays = _minStakeDays;
547         start = _start;
548         secondsInDay = _secondsInDay;
549         maxBoughtPerDay = _maxBoughtPerDay;
550         bonusStartPercent = _bonusStartPercent;
```

```
551         bonusStartDays = _bonusStartDays;
552         baseBonus = _baseBonus;
553         splitAmounts = _splitAmounts;
554     }
555 }
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendations:

For best security practices, consider as much as possible declaring events at the end of the function. Events can be used to detect the end of the operation.

Remediation Plan:

SOLVED: The Axion Network team solved the issue by adding events to the above functions.

3.15 (HAL-15) REDUNDANT BOOLEAN COMPARISON - INFORMATIONAL

Description:

In the solidity language, Boolean constants can be used directly and do not need to be compare to true or false. In the Axion Network contracts, boolean constants are compared with true or false.

Code Location:

Listing 77: Accelerator.sol (Lines 182,183,184,185)

```
182         require(
183             allowedTokens[_token] == true,
184             'AUTOSTAKER: This token is not allowed to be used on
                    this contract'
185         );
```

Listing 78: VentureCapital.sol (Lines 58,59)

```
57      function ensureIsVcaRegisteredInternal(address staker)
            internal {
58          if (isVcaRegistered[staker] == false) {
59              if (contracts.stakingV2.getIsVCARegistered(staker) ==
                    false) {
60                  uint256 totalShares = contracts.stakingV2.
                        resolveTotalSharesOf(staker);
```

Listing 79: VentureCapital.sol (Lines 293)

```
292     function getDeductBalances(address staker, address token)
            external view returns (int256) {
293         if (isVcaRegistered[staker] == false) {
294             return contracts.stakingV2.getDeductBalances(staker,
                    token).toInt256();
295         }
```

Listing 80: VentureCapital.sol (Lines 308)

```
303     function getTokenInterestEarned(address accountAddress,
            address tokenAddress)
304         external
305         view
306         returns (uint256)
307     {
308         if (isVcaRegistered[accountAddress] == false) {
309             return
310                 ((contracts.stakingV2.getTotalSharesOf(
                        accountAddress) *
311                     tokenPricePerShare[tokenAddress]) -
312                     contracts.stakingV2.getDeductBalances(
                            accountAddress, tokenAddress)) / 1e36;
313         }
```

Listing 81: VentureCapital.sol (Lines 329,337)

```
328     function getTotalSharesOf(address account) external view
            returns (uint256) {
329         if (isVcaRegistered[account] == false) {
330             return contracts.stakingV2.getTotalSharesOf(account);
331         }
332
333         return totalSharesOf[account];
334     }
335
336     function getIsVCARegistered(address staker) external view
            returns (bool) {
337         if (isVcaRegistered[staker] == false) {
338             return contracts.stakingV2.getIsVCARegistered(staker);
339         }
340
341         return true;
342     }
```

Listing 82: StakeToken.sol (Lines 41)

```
40     function mint(address staker, uint256 id) external override
           onlyExternalCaller {
41         require(enabled == true, 'STAKE TOKEN: Contract is
               disabled');
```

```
42          _safeMint(staker, id);
43      }
```

Listing 83: StakeToken.sol (Lines 79)

```
74      function transferFrom(
75          address from,
76          address to,
77          uint256 tokenId
78      ) public virtual override(ERC721Upgradeable,
            IERC721Upgradeable) onlyMigrator pausable {
79          require(transferEnabled == true, 'STAKE TOKEN: transfer is
                disabled.');
```

Listing 84: StakeBurner.sol (Lines 132,133,134,135)

```
132         require(
133             contracts.stakeManager.getStakeWithdrawnOrExists(
                    sessionId) == false,
134             'STAKE BURNER: stake is withdrawn or already v3.'
135         );
```

Listing 85: StakeBurner.sol (Lines 158)

```
149         if (shares != 0) {
150             if (requireMature) {
151                 require(
152                     end != 0 && end <= block.timestamp,
153                     'STAKE BURNER: stake not mature or not set.'
154                 );
155             }
156             // if shares are not 0 it means it is v2 or has been
                    upgraded and saved to v2
157
158             require(withdrawn == false, 'STAKE BURNER: stake
                    withdrawn on v2.');
159         } else {
```

Listing 86: StakeMinter.sol (Lines 187)

```
185         if (shares != 0) {
186             // if shares are not 0 it means it is v2 or has been
                    upgraded and saved to v2
187             require(withdrawn == false, 'STAKE BURNER: stake
                    withdrawn on v2.');
188         } else {
```

Listing 87: StakeMinter.sol (Lines 168,169,170,171)

```
168         require(
169             contracts.stakeManager.getStakeWithdrawnOrExists(id)
                    == false,
170             'STAKE MINTER: stake is withdrawn or already v3.'
171         );
```

Listing 88: StakeUpgrader.sol (Lines 64,65,66,67)

```
62     function maxShareLegacyUpgrade(uint256 sessionId) external
           pausable {
63         require(sessionId <= settings.lastSessionIdV2, 'UNSTAKER:
               invalid stakeId.');
64         require(
65             contracts.stakeManager.getStakeWithdrawnOrExists(
                    sessionId) == false,
66             'UNSTAKER: stake is withdrawn or already v3.'
67         );
```

Listing 89: StakeUpgrader.sol (Lines 84)

```
81         if (shares != 0) {
82             // if shares are not 0 it means it is v2 or has been
                    upgraded and saved to v2
83
84             require(withdrawn == false, 'UNSTAKER: stake withdrawn
                     on v2.');
85         } else {
```

Listing 90: StakeUpgrader.sol (Lines 120)

```
119     function maxShareUpgradeInternal(uint256 stakingDays) internal
            view {
120         require(settings.maxShareEventActive == true, 'STAKING:
                Max Share event is not active');
121         require(
122             stakingDays <= settings.maxShareMaxDays,
123             'STAKING: Max Share Upgrade - Stake must be less then
                    max share max days'
124         );
125     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

It is recommended to compare boolean constants directly in the require modifier.

Remediation Plan:

SOLVED: The Axion Network team solved the issue by removing boolean constants comparison with true or false, and implemented comparison of boolean constants directly in the require modifier.

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

Description:

In public functions, array arguments are immediately copied to memory, while external functions can read directly from calldata. Reading calldata is cheaper than memory allocation. Public functions need to write the arguments to memory because public functions may be called internally. Internal calls are passed internally by pointers to memory. Thus, the function expects its arguments being located in memory when the compiler generates the code for an internal function.
Also, methods do not necessarily have to be public if they are only called within the contract-in such case they should be marked internal.

Code Location:

Below are smart contracts and their corresponding functions affected:

Accelerator.sol:
getSplitAmounts()

DataReader.sol:
initialize(address,address,address,address,address)

StakeBurner.sol:
init(address,address,address,address,address,address,address,address)

StakeManager.sol:
init(address,address,address,address,address,address)

StakeMinter.sol:
init(address,address,address,address,address,address,address,address,address)
restore(uint32,uint32)

StakeReminter.sol:
init(address,address,address,address)

StakeToken.sol:
init(address,address,address,address)

StakeUpgrader.sol:
init(address,address,address,address,address)

Token.sol:
init(address,address,address,address) initialize(address,address,string,string)

VentureCapital.sol:
initialize(address,address)

AuctionV21.sol:
calculateStepsFromStart()

StakingV21.sol:
calculateStakingInterest(uint256,uint256,uint256) calculateStepsFromStart()
getAmountOutAndPenalty(uint256,uint256,uint256,uint256)

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider as much as possible declaring external variables instead of public variables. As for best practice, you should use external if you expect that the function will only be called externally and use public if you need to call the function internally. To sum up, all can access to public functions, external functions only can be accessed externally and internal functions can only be called within the contract.

Remediation Plan:

SOLVED: The Axion Network team solved the issue by declaring external functions instead of public.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped contract. 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. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Results:

```
INFO:Detectors:
Token.recovery(address,address,uint256) (contracts/Token.sol#116-122) ignores return value by IERC20(tokenToRecover).transfer(recoverFor,amount) (contracts/Token.sol#121)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Token.swapTokenBalance (contracts/Token.sol#26) is never initialized. It is used in:
        - Token.getSwapTokenBalance(uint256) (contracts/Token.sol#94-96)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
StakingV21.transferTokens(address,address) (contracts/v2.1/StakingV21.sol#111-124) sends eth to arbitrary user
        Dangerous calls:
        - address(vcAuction).transfer(address(this).balance) (contracts/v2.1/StakingV21.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
StakingV21.transferTokens(address,address) (contracts/v2.1/StakingV21.sol#111-124) ignores return value by token.transfer(vcAuction,token.balanceOf(address(this))) (contracts/v2.1/StakingV21
.sol#116)
StakingV21.transferTokens(address,address) (contracts/v2.1/StakingV21.sol#111-124) ignores return value by axn.transfer(stakeManager,axn.balanceOf(address(this))) (contracts/v2.1/StakingV21.
sol#123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
StakingV21.addresses (contracts/v2.1/StakingV21.sol#50) is never initialized. It is used in:
        - StakingV21.transferTokens(address,address) (contracts/v2.1/StakingV21.sol#111-124)
StakingV21.stakingV1 (contracts/v2.1/StakingV21.sol#51) is never initialized. It is used in:
        - StakingV21.resolveTotalSharesOf(address) (contracts/v2.1/StakingV21.sol#132-163)
StakingV21.stepTimestamp (contracts/v2.1/StakingV21.sol#62) is never initialized. It is used in:
        - StakingV21.getAmountOutAndPenalty(uint256,uint256,uint256,uint256) (contracts/v2.1/StakingV21.sol#243-284)
        - StakingV21.calculateStepsFromStart() (contracts/v2.1/StakingV21.sol#286-288)
StakingV21.startContract (contracts/v2.1/StakingV21.sol#63) is never initialized. It is used in:
        - StakingV21.calculateStepsFromStart() (contracts/v2.1/StakingV21.sol#286-288)
StakingV21.lastSessionIdV1 (contracts/v2.1/StakingV21.sol#67) is never initialized. It is used in:
        - StakingV21.resolveTotalSharesOf(address) (contracts/v2.1/StakingV21.sol#132-163)
StakingV21.sessionDataOf (contracts/v2.1/StakingV21.sol#71) is never initialized. It is used in:
        - StakingV21.resolveTotalSharesOf(address) (contracts/v2.1/StakingV21.sol#132-163)
StakingV21.sessionsOf (contracts/v2.1/StakingV21.sol#73) is never initialized. It is used in:
        - StakingV21.sessionsOf_(address) (contracts/v2.1/StakingV21.sol#127-129)
        - StakingV21.resolveTotalSharesOf(address) (contracts/v2.1/StakingV21.sol#132-163)
StakingV21.payouts (contracts/v2.1/StakingV21.sol#75) is never initialized. It is used in:
        - StakingV21.calculateStakingInterest(uint256,uint256,uint256) (contracts/v2.1/StakingV21.sol#177-193)
StakingV21.maxShareEventActive (contracts/v2.1/StakingV21.sol#83) is never initialized. It is used in:
        - StakingV21.getMaxShareEventActive() (contracts/v2.1/StakingV21.sol#195-197)
StakingV21.maxShareMaxDays (contracts/v2.1/StakingV21.sol#85) is never initialized. It is used in:
        - StakingV21.getMaxShareMaxDays() (contracts/v2.1/StakingV21.sol#199-201)
```

```
INFO:Detectors:
AuctionV21.auctionsOf (contracts/v2.1/AuctionV21.sol#56) is never initialized. It is used in:
        - AuctionV21.auctionsOf_(address) (contracts/v2.1/AuctionV21.sol#96-98)
AuctionV21.start (contracts/v2.1/AuctionV21.sol#63) is never initialized. It is used in:
        - AuctionV21.calculateStepsFromStart() (contracts/v2.1/AuctionV21.sol#106-108)
AuctionV21.stepTimestamp (contracts/v2.1/AuctionV21.sol#64) is never initialized. It is used in:
        - AuctionV21.calculateStepsFromStart() (contracts/v2.1/AuctionV21.sol#106-108)
AuctionV21.addresses (contracts/v2.1/AuctionV21.sol#67) is never initialized. It is used in:
        - AuctionV21.burnTokenBalance() (contracts/v2.1/AuctionV21.sol#100-103)
AuctionV21.auctions (contracts/v2.1/AuctionV21.sol#86) is never initialized. It is used in:
        - AuctionV21.getAuctionModes() (contracts/v2.1/AuctionV21.sol#114-122)
AuctionV21.ventureAutoStakeDays (contracts/v2.1/AuctionV21.sol#87) is never initialized. It is used in:
        - AuctionV21.getVentureAutoStakeDays() (contracts/v2.1/AuctionV21.sol#110-112)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
```

AUTOMATED TESTING

```
Reentrancy in VentureCapital.withdrawOriginDivTokens(address) (contracts/accelerator/VentureCapital.sol#144-156):
        External calls:
        - IERC20Upgradeable(tokenAddress).transfer(msg.sender,originWithdrawableTokenAmounts[tokenAddress]) (contracts/accelerator/VentureCapital.sol#147-150)
        External calls sending eth:
        - address(msg.sender).transfer(originWithdrawableTokenAmounts[tokenAddress]) (contracts/accelerator/VentureCapital.sol#152)
        State variables written after the call(s):
        - originWithdrawableTokenAmounts[tokenAddress] = 0 (contracts/accelerator/VentureCapital.sol#155)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
VentureCapital.withdrawDivTokenInternal(address,address,address) (contracts/accelerator/VentureCapital.sol#116-142) ignores return value by IERC20Upgradeable(tokenAddress).
transfer(to,tokenInterestEarned) (contracts/accelerator/VentureCapital.sol#136)
VentureCapital.withdrawOriginDivTokens(address) (contracts/accelerator/VentureCapital.sol#144-156) ignores return value by IERC20Upgradeable(tokenAddress).transfer(msg.send
er,originWithdrawableTokenAmounts[tokenAddress]) (contracts/accelerator/VentureCapital.sol#147-150)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
VentureCapital.axion (contracts/accelerator/VentureCapital.sol#39) is never initialized. It is used in:
        - VentureCapital.updateTokenPricePerShare(address,uint256) (contracts/accelerator/VentureCapital.sol#236-248)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Accelerator.axionBuyAndStakeEth(uint256,uint256,uint256,uint256) (contracts/accelerator/Accelerator.sol#89-131) sends eth to arbitrary user
        Dangerous calls:
        - recipient.transfer(_recipientAmount) (contracts/accelerator/Accelerator.sol#115)
Accelerator.swapEthForTokens(address,address,uint256,uint256,uint256) (contracts/accelerator/Accelerator.sol#143-163) sends eth to arbitrary user
        Dangerous calls:
        - IUniswapV2Router02(uniswap).swapExactETHForTokens{value: _amountIn}(_amountOutMin,path,_to,_deadline)[1] (contracts/accelerator/Accelerator.sol#156-162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Accelerator.axionBuyAndStake(address,uint256,uint256,uint256,uint256,uint256) (contracts/accelerator/Accelerator.sol#173-239) ignores return value by IERC20(_token).transfe
rFrom(msg.sender,address(this),_amount) (contracts/accelerator/Accelerator.sol#193)
Accelerator.axionBuyAndStake(address,uint256,uint256,uint256,uint256,uint256) (contracts/accelerator/Accelerator.sol#173-239) ignores return value by IERC20(_token).transfe
r(recipient,_recipientAmount) (contracts/accelerator/Accelerator.sol#222)
INFO:Detectors:
BPD.getBpdAmount(uint256,uint256,uint256) (contracts/stake/BPD.sol#79-93) performs a multiplication on the result of a division:
        -bpdAmount += (shares / bpdShares[i]) * (uint256(bpdPools[i]) * 1e8) (contracts/stake/BPD.sol#89)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
StakeBurner.unstakeLegacyStakeInternal(uint256,address,bool) (contracts/stake/StakeBurner.sol#126-199) ignores return value by contracts.stakeCustodian.addStake(staker,sess
ionId) (contracts/stake/StakeBurner.sol#196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
StakeMinter.stakeInternal(uint256,uint256,address) (contracts/stake/StakeMinter.sol#79-88) ignores return value by contracts.stakeCustodian.addStake(staker,contracts.stakeM
anager.createStake(staker,amount,stakingDays)) (contracts/stake/StakeMinter.sol#84-87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
StakeManager.upgradeStakeInternal(StakeUpgrade) (contracts/stake/StakeManager.sol#156-225) performs a multiplication on the result of a division:
        -addToGlobalTotals(newAmount - (stakeUpgrade.amount / 1e12) * 1e12,newShares - (stakeUpgrade.shares / 1e12) * 1e12) (contracts/stake/StakeManager.sol#199-202)
StakeManager.getStakersSharesAmountInternal(uint256,uint256) (contracts/stake/StakeManager.sol#231-241) performs a multiplication on the result of a division:
        -(shares / 1e12) * 1e12 (contracts/stake/StakeManager.sol#240)
StakeManager.updateShareRate(uint256) (contracts/stake/StakeManager.sol#525-542) performs a multiplication on the result of a division:
        -growthFactor = (_payout * 1e18) / (currentTokenTotalSupply + (uint256(statFields.totalStakedAmount) * 1e12) + 1) (contracts/stake/StakeManager.sol#528-530)
        -interestFields.shareRate = (((uint256(interestFields.shareRate) * (1e36 + (uint256(settings.shareRateScalingFactor) * growthFactor))) / 1e36)).toUint128() (contrac
ts/stake/StakeManager.sol#537-541)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
StakeManager.addDailyInterest() (contracts/stake/StakeManager.sol#465-497) uses a dangerous strict equality:
        - statFields.sharesTotalSupply == 0 (contracts/stake/StakeManager.sol#472)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
AuctionV21.getAuctionModes().i (contracts/v2.1/AuctionV21.sol#117) is a local variable never initialized
AuctionV21.getAuctionModes().auctionModes (contracts/v2.1/AuctionV21.sol#115) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Reentrancy in StakeManager.addDailyInterest() (contracts/stake/StakeManager.sol#465-497):
        External calls:
        - interest = getTodaysInterest() (contracts/stake/StakeManager.sol#471)
                - IERC20Upgradeable(contracts.token).transfer(0x000000000000000000000000000000000000dEaD,amountTokenInDay) (contracts/stake/StakeManager.sol#504-507)
        State variables written after the call(s):
        - interestFields.nextAddInterestTimestamp += settings.secondsInDay (contracts/stake/StakeManager.sol#488)
        - updateShareRate(interest) (contracts/stake/StakeManager.sol#490)
                - interestFields.shareRate = (((uint256(interestFields.shareRate) * (1e36 + (uint256(settings.shareRateScalingFactor) * growthFactor))) / 1e36)).toUint128()
 (contracts/stake/StakeManager.sol#537-541)
        - statFields.sharesTotalSupply = 1e6 (contracts/stake/StakeManager.sol#473)
Reentrancy in StakeManager.unsetStakeInternal(address,uint256,uint256,uint256) (contracts/stake/StakeManager.sol#351-385):
        External calls:
        - addDailyInterest() (contracts/stake/StakeManager.sol#364)
                - IERC20Upgradeable(contracts.token).transfer(0x000000000000000000000000000000000000dEaD,amountTokenInDay) (contracts/stake/StakeManager.sol#504-507)
        - contracts.ventureCapital.subTotalSharesOfAndRebalance(staker,shares) (contracts/stake/StakeManager.sol#367)
        State variables written after the call(s):
        - stakeData[id].payout = (payout / 1e18).toUint40() (contracts/stake/StakeManager.sol#373)
        - stakeData[id].status = StakeStatus.Withdrawn (contracts/stake/StakeManager.sol#374)
        - removeFromGlobalTotals(amount,shares) (contracts/stake/StakeManager.sol#370)
                - statFields.sharesTotalSupply -= (shares / 1e12).toUint72() (contracts/stake/StakeManager.sol#394)
                - statFields.totalStakedAmount -= (amount / 1e12).toUint72() (contracts/stake/StakeManager.sol#395)
                - statFields.totalVcaRegisteredShares -= (shares / 1e12).toUint72() (contracts/stake/StakeManager.sol#396)
Reentrancy in StakeManager.upgradeStakeInternal(StakeUpgrade) (contracts/stake/StakeManager.sol#156-225):
        External calls:
        - contracts.ventureCapital.addTotalSharesOfAndRebalance(stakeUpgrade.staker,newShares - stakeUpgrade.shares) (contracts/stake/StakeManager.sol#186-189)
        - contracts.bpd.addBpdMaxShares(stakeUpgrade.shares,stakeUpgrade.start,stakeUpgrade.start + (uint256(settings.secondsInDay) * stakeUpgrade.stakingDays),newShares,bl
ock.timestamp,newEnd) (contracts/stake/StakeManager.sol#190-197)
        - createStakeInternal(NewStake(stakeUpgrade.id,newAmount,newShares,block.timestamp,5555,interestPerShare.length)) (contracts/stake/StakeManager.sol#204-213)
                - IERC20Upgradeable(contracts.token).transfer(0x000000000000000000000000000000000000dEaD,amountTokenInDay) (contracts/stake/StakeManager.sol#504-507)
        State variables written after the call(s):
        - createStakeInternal(NewStake(stakeUpgrade.id,newAmount,newShares,block.timestamp,5555,interestPerShare.length)) (contracts/stake/StakeManager.sol#204-213)
                - interestFields.shareRate = (((uint256(interestFields.shareRate) * (1e36 + (uint256(settings.shareRateScalingFactor) * growthFactor))) / 1e36)).toUint128()
 (contracts/stake/StakeManager.sol#537-541)
INFO:Detectors:
StakeManager.getStakeInterestInternal(uint256,uint256,uint256).firstInterest (contracts/stake/StakeManager.sol#411) is a local variable never initialized
StakeManager.getStakeInterestInternal(uint256,uint256,uint256).lastInterest (contracts/stake/StakeManager.sol#410) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
StakeUpgrader.maxShareLegacyUpgrade(uint256) (contracts/stake/StakeUpgrader.sol#62-113) ignores return value by contracts.stakeCustodian.addStake(msg.sender,sessionId) (con
tracts/stake/StakeUpgrader.sol#112)
Reentrancy in VentureCapital.ensureIsVcaRegisteredInternal(address) (contracts/accelerator/VentureCapital.sol#57-82):
        External calls:
        - contracts.stakeManager.addTotalVcaRegisteredShares(totalShares) (contracts/accelerator/VentureCapital.sol#63)
        State variables written after the call(s):
        - isVcaRegistered[staker] = true (contracts/accelerator/VentureCapital.sol#80)
Reentrancy in VentureCapital.transferSharesAndRebalance(address,address,uint256) (contracts/accelerator/VentureCapital.sol#187-194):
        External calls:
        - subTotalSharesOfAndRebalanceInternal(from,shares) (contracts/accelerator/VentureCapital.sol#192)
                - contracts.stakeManager.addTotalVcaRegisteredShares(totalShares) (contracts/accelerator/VentureCapital.sol#63)
        - addTotalSharesOfAndRebalanceInternal(to,shares) (contracts/accelerator/VentureCapital.sol#193)
                - contracts.stakeManager.addTotalVcaRegisteredShares(totalShares) (contracts/accelerator/VentureCapital.sol#63)
        State variables written after the call(s):
        - addTotalSharesOfAndRebalanceInternal(to,shares) (contracts/accelerator/VentureCapital.sol#193)
                - deductBalances[staker][divTokens.at(i)] = (totalSharesOf[staker] * tokenPricePerShare[divTokens.at(i)]).toInt256() - tokenInterestEarned (contracts/accele
rator/VentureCapital.sol#226-228)
                - deductBalances[staker][divTokens.at(i)] = (totalShares * tokenPricePerShare[divTokens.at(i)]).toInt256() (contracts/accelerator/VentureCapital.sol#66-68)
                - deductBalances[staker][divTokens.at(i_scope_0)] = contracts.stakingV2.getDeductBalances(staker,divTokens.at(i_scope_0)).toInt256() (contracts/accelerator/
VentureCapital.sol#73-76)
        - addTotalSharesOfAndRebalanceInternal(to,shares) (contracts/accelerator/VentureCapital.sol#193)
                - isVcaRegistered[staker] = true (contracts/accelerator/VentureCapital.sol#80)
        - addTotalSharesOfAndRebalanceInternal(to,shares) (contracts/accelerator/VentureCapital.sol#193)
                - totalSharesOf[staker] += shares (contracts/accelerator/VentureCapital.sol#202)
                - totalSharesOf[staker] = totalShares (contracts/accelerator/VentureCapital.sol#62)
                - totalSharesOf[staker] = contracts.stakingV2.getTotalSharesOf(staker) (contracts/accelerator/VentureCapital.sol#71)
```

```
INFO:Detectors:
VentureCapital.addDivToken(address) (contracts/accelerator/VentureCapital.sol#252-254) ignores return value by divTokens.add(tokenAddress) (contracts/accelerator/VentureCapital.sol#253)
VentureCapital.init(address,address,address,address,address,address) (contracts/accelerator/VentureCapital.sol#265-290) ignores return value by divTokens.add(v2DivTokens[i]) (contracts/accelerator/VentureCapital.sol#285)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Accelerator.sendAndBurn(uint256,uint256,uint256,uint256) (contracts/accelerator/Accelerator.sol#291-331) performs a multiplication on the result of a division:
        -payout = (100 * _axionBought) / splitAmounts[0] (contracts/accelerator/Accelerator.sol#304)
        -payout = payout + (payout * baseBonus) / 100 (contracts/accelerator/Accelerator.sol#305)
Accelerator.sendAndBurn(uint256,uint256,uint256,uint256) (contracts/accelerator/Accelerator.sol#291-331) performs a multiplication on the result of a division:
        -payout = (100 * _axionBought) / splitAmounts[0] (contracts/accelerator/Accelerator.sol#304)
        -payout = payout + (payout * ((_days / bonusStartDays) + bonusStartPercent)) / 100 (contracts/accelerator/Accelerator.sol#320)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Accelerator.getSplitAmounts()._splitAmounts (contracts/accelerator/Accelerator.sol#343) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Accelerator.swapTokensForTokens(address,address,address,uint256,uint256,uint256) (contracts/accelerator/Accelerator.sol#252-280) ignores return value by IERC20(_tokenInAddress).approve(uniswap,2 ** 255) (contracts/accelerator/Accelerator.sol#268)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
StakeBurner.unstakeLegacyStakeInternal(uint256,address,bool) (contracts/stake/StakeBurner.sol#126-199) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(end != 0 && end <= block.timestamp,STAKE BURNER: stake not mature or not set.) (contracts/stake/StakeBurner.sol#151-154)
        - require(bool,string)(end != 0 && end <= block.timestamp,STAKE BURNER: stake not mature or not set.) (contracts/stake/StakeBurner.sol#168-171)
StakeBurner.getPayoutAndPenaltyInternal(uint256,uint256,uint256,uint256) (contracts/stake/StakeBurner.sol#208-247) uses timestamp for comparisons
        Dangerous comparisons:
        - stakingDays > daysStaked (contracts/stake/StakeBurner.sol#220)
        - daysStaked < stakingDays + 14 (contracts/stake/StakeBurner.sol#227)
        - daysStaked < stakingDays + 714 (contracts/stake/StakeBurner.sol#230)
StakeBurner.handlePayoutAndPenalty(address,uint256,uint256,uint256,uint256) (contracts/stake/StakeBurner.sol#258-284) uses timestamp for comparisons
        Dangerous comparisons:
        - payout != 0 (contracts/stake/StakeBurner.sol#279)
        - block.timestamp < intendedEnd (contracts/stake/StakeBurner.sol#269-273)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
StakeReminter.remintStake(uint256,uint256,uint256) (contracts/stake/StakeReminter.sol#34-49) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(end != 0 && end <= block.timestamp,RESTAKER: Stake not mature or not set.) (contracts/stake/StakeReminter.sol#44)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Reentrancy in StakeManager.addDailyInterest() (contracts/stake/StakeManager.sol#465-497):
        External calls:
        - interest = getTodaysInterest() (contracts/stake/StakeManager.sol#471)
                - IERC20Upgradeable(contracts.token).transfer(0x0000000000000000000000000000000000dEaD,amountTokenInDay) (contracts/stake/StakeManager.sol#504-507)
        State variables written after the call(s):
        - interestPerShare.push(todaysSharePayout) (contracts/stake/StakeManager.sol#487)
        - updateShareRate(interest) (contracts/stake/StakeManager.sol#490)
                - settings.shareRateScalingFactor = 1e18 (contracts/stake/StakeManager.sol#534)
Reentrancy in StakeManager.addDailyInterest() (contracts/stake/StakeManager.sol#465-497):
        External calls:
        - interest = getTodaysInterest() (contracts/stake/StakeManager.sol#471)
                - IERC20Upgradeable(contracts.token).transfer(0x0000000000000000000000000000000000dEaD,amountTokenInDay) (contracts/stake/StakeManager.sol#504-507)
        Event emitted after the call(s):
        - DailyInterestAdded(interest,statFields.sharesTotalSupply,todaysSharePayout,block.timestamp) (contracts/stake/StakeManager.sol#491-496)
Reentrancy in StakeManager.createExistingStake(uint256,uint256,uint256,uint256,uint256,uint256) (contracts/stake/StakeManager.sol#87-116):
        External calls:
        - createStakeInternal(NewStake(id,amount,shares,start,stakingDays,firstInterestDay)) (contracts/stake/StakeManager.sol#98-107)
                - IERC20Upgradeable(contracts.token).transfer(0x0000000000000000000000000000000000dEaD,amountTokenInDay) (contracts/stake/StakeManager.sol#504-507)
        Event emitted after the call(s):
        - ExistingStakeCreated(id,stakeData[id].amount,stakeData[id].shares,stakeData[id].start,stakeData[id].stakingDays) (contracts/stake/StakeManager.sol#109-115)
Reentrancy in StakeManager.createStake(address,uint256,uint256) (contracts/stake/StakeManager.sol#38-77):
        External calls:
        - createStakeInternal(NewStake(statFields.lastStakeId,amount,shares,block.timestamp,stakingDays,interestPerShare.length)) (contracts/stake/StakeManager.sol#49-58)
                - IERC20Upgradeable(contracts.token).transfer(0x0000000000000000000000000000000000dEaD,amountTokenInDay) (contracts/stake/StakeManager.sol#504-507)
        - contracts.ventureCapital.addTotalSharesOfAndRebalance(staker,shares) (contracts/stake/StakeManager.sol#60)
        - contracts.bpd.addBpdShares(shares,block.timestamp,stakingDays) (contracts/stake/StakeManager.sol#64)
        Event emitted after the call(s):
        - StakeCreated(staker,statFields.lastStakeId,stakeData[statFields.lastStakeId].amount,stakeData[statFields.lastStakeId].shares,stakeData[statFields.lastStakeId].start,stakeData[statFields.lastStakeId].stakingDays) (contracts/stake/StakeManager.sol#67-74)
Reentrancy in StakeManager.unsetStakeInternal(address,uint256,uint256,uint256,uint256) (contracts/stake/StakeManager.sol#351-385):
        External calls:
        - addDailyInterest() (contracts/stake/StakeManager.sol#364)
                - IERC20Upgradeable(contracts.token).transfer(0x0000000000000000000000000000000000dEaD,amountTokenInDay) (contracts/stake/StakeManager.sol#504-507)
        - contracts.ventureCapital.subTotalSharesOfAndRebalance(staker,shares) (contracts/stake/StakeManager.sol#367)
        Event emitted after the call(s):
        - StakeDeleted(staker,id.toUint128(),stakeData[id].amount,stakeData[id].shares,stakeData[id].start,stakeData[id].stakingDays) (contracts/stake/StakeManager.sol#377-384)
Reentrancy in StakeManager.upgradeStakeInternal(StakeUpgrade) (contracts/stake/StakeManager.sol#156-225):
        External calls:
        - contracts.ventureCapital.addTotalSharesOfAndRebalance(stakeUpgrade.staker,newShares - stakeUpgrade.shares) (contracts/stake/StakeManager.sol#186-189)
        - contracts.bpd.addBpdMaxShares(stakeUpgrade.shares,stakeUpgrade.start,stakeUpgrade.start + (uint256(settings.secondsInDay) * stakeUpgrade.stakingDays),newShares,bl
StakeManager.upgradeStakeInternal(StakeUpgrade) (contracts/stake/StakeManager.sol#156-225) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(newShares > stakeUpgrade.shares,STAKING: New shares are not greater then previous shares) (contracts/stake/StakeManager.sol#179-182)
        - block.timestamp < intendedEnd (contracts/stake/StakeManager.sol#169-173)
StakeManager.createStakeInternal(StakeBase.NewStake) (contracts/stake/StakeManager.sol#267-278) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= interestFields.nextAddInterestTimestamp (contracts/stake/StakeManager.sol#269)
StakeManager.unsetStakeInternal(address,uint256,uint256,uint256,uint256) (contracts/stake/StakeManager.sol#351-385) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= interestFields.nextAddInterestTimestamp (contracts/stake/StakeManager.sol#364)
StakeManager.getStakeInterestInternal(uint256,uint256,uint256) (contracts/stake/StakeManager.sol#405-425) uses timestamp for comparisons
        Dangerous comparisons:
        - interestPerShare.length != 0 (contracts/stake/StakeManager.sol#414)
StakeManager.addDailyInterest() (contracts/stake/StakeManager.sol#465-497) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= interestFields.nextAddInterestTimestamp,Staking: Too early to add interest.) (contracts/stake/StakeManager.sol#466-469)
        - statFields.sharesTotalSupply == 0 (contracts/stake/StakeManager.sol#472)
INFO:Detectors:
StakingV21.transferTokens(address,address).vcAuction (contracts/v2.1/StakingV21.sol#111) lacks a zero-check on :
                - address(vcAuction).transfer(address(this).balance) (contracts/v2.1/StakingV21.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
StakingV21.transferTokens(address,address) (contracts/v2.1/StakingV21.sol#111-124) has external calls inside a loop: token.transfer(vcAuction,token.balanceOf(address(this))) (contracts/v2.1/StakingV21.sol#116)
StakingV21.transferTokens(address,address) (contracts/v2.1/StakingV21.sol#111-124) has external calls inside a loop: address(vcAuction).transfer(address(this).balance) (contracts/v2.1/StakingV21.sol#118)
StakingV21.resolveTotalSharesOf(address) (contracts/v2.1/StakingV21.sol#132-163) has external calls inside a loop: (shares) = stakingV1.sessionDataOf(account,v1SessionsOfAccount[i_scope_0]) (contracts/v2.1/StakingV21.sol#155)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
StakingV21.getAmountOutAndPenalty(uint256,uint256,uint256,uint256) (contracts/v2.1/StakingV21.sol#243-284) uses timestamp for comparisons
        Dangerous comparisons:
        - stakingDays > daysStaked (contracts/v2.1/StakingV21.sol#256)
        - daysStaked < stakingDays.add(14) (contracts/v2.1/StakingV21.sol#263)
        - daysStaked < stakingDays.add(714) (contracts/v2.1/StakingV21.sol#266)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
INFO:Detectors:
VentureCapital.ensureIsVcaRegisteredInternal(address) (contracts/accelerator/VentureCapital.sol#57-82) has external calls inside a loop: deductBalances[staker][divTokens.at
(i_scope_0)] = contracts.stakingV2.getDeductBalances(staker,divTokens.at(i_scope_0)).toInt256() (contracts/accelerator/VentureCapital.sol#73-76)
VentureCapital.init(address,address,address,address,address,address) (contracts/accelerator/VentureCapital.sol#265-290) has external calls inside a loop: tokenPricePerShare
[v2DivTokens[i]] = contracts.stakingV2.getTokenPricePerShare(v2DivTokens[i]) (contracts/accelerator/VentureCapital.sol#286-288)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in VentureCapital.addTotalSharesOfAndRebalanceInternal(address,uint256) (contracts/accelerator/VentureCapital.sol#196-205):
        External calls:
        - ensureIsVcaRegistered(staker) (contracts/accelerator/VentureCapital.sol#198)
                - contracts.stakeManager.addTotalVcaRegisteredShares(totalShares) (contracts/accelerator/VentureCapital.sol#63)
        State variables written after the call(s):
        - rebalance(staker,oldTotalSharesOf) (contracts/accelerator/VentureCapital.sol#204)
                - deductBalances[staker][divTokens.at(i)] = (totalSharesOf[staker] * tokenPricePerShare[divTokens.at(i)]).toInt256() - tokenInterestEarned (contracts/accele
rator/VentureCapital.sol#226-228)
        - totalSharesOf[staker] += shares (contracts/accelerator/VentureCapital.sol#202)
Reentrancy in VentureCapital.ensureIsVcaRegisteredInternal(address) (contracts/accelerator/VentureCapital.sol#57-82):
        External calls:
        - contracts.stakeManager.addTotalVcaRegisteredShares(totalShares) (contracts/accelerator/VentureCapital.sol#63)
        State variables written after the call(s):
        - deductBalances[staker][divTokens.at(i)] = (totalShares * tokenPricePerShare[divTokens.at(i)]).toInt256() (contracts/accelerator/VentureCapital.sol#66-68)
Reentrancy in VentureCapital.subTotalSharesOfAndRebalanceInternal(address,uint256) (contracts/accelerator/VentureCapital.sol#207-215):
        External calls:
        - ensureIsVcaRegistered(staker) (contracts/accelerator/VentureCapital.sol#209)
                - contracts.stakeManager.addTotalVcaRegisteredShares(totalShares) (contracts/accelerator/VentureCapital.sol#63)
        State variables written after the call(s):
        - rebalance(staker,oldTotalSharesOf) (contracts/accelerator/VentureCapital.sol#214)
                - deductBalances[staker][divTokens.at(i)] = (totalSharesOf[staker] * tokenPricePerShare[divTokens.at(i)]).toInt256() - tokenInterestEarned (contracts/accele
rator/VentureCapital.sol#226-228)
        - totalSharesOf[staker] -= shares (contracts/accelerator/VentureCapital.sol#212)
Accelerator.setMinStakeDays(uint256) (contracts/accelerator/Accelerator.sol#402-404) should emit an event for:
        - minStakeDays = _days (contracts/accelerator/Accelerator.sol#403)
Accelerator.setMaxBoughtPerDay(uint256) (contracts/accelerator/Accelerator.sol#419-421) should emit an event for:
        - maxBoughtPerDay = _amount (contracts/accelerator/Accelerator.sol#420)
Accelerator.setBaseBonus(uint8) (contracts/accelerator/Accelerator.sol#426-428) should emit an event for:
        - baseBonus = _amount (contracts/accelerator/Accelerator.sol#427)
Accelerator.setBonusStartPercent(uint8) (contracts/accelerator/Accelerator.sol#433-435) should emit an event for:
        - bonusStartPercent = _amount (contracts/accelerator/Accelerator.sol#434)
Accelerator.setBonusStartDays(uint16) (contracts/accelerator/Accelerator.sol#440-442) should emit an event for:
        - bonusStartDays = _amount (contracts/accelerator/Accelerator.sol#441)
Accelerator.setStart(uint256) (contracts/accelerator/Accelerator.sol#454-456) should emit an event for:
        - start = _start (contracts/accelerator/Accelerator.sol#455)
Accelerator.startVariables(uint256,uint256,uint256,uint256,uint8,uint16,uint8,uint8[3]) (contracts/accelerator/Accelerator.sol#520-541) should emit an event for:
        - minStakeDays = _minStakeDays (contracts/accelerator/Accelerator.sol#533)
        - start = _start (contracts/accelerator/Accelerator.sol#534)
        - secondsInDay = _secondsInDay (contracts/accelerator/Accelerator.sol#535)
        - maxBoughtPerDay = _maxBoughtPerDay (contracts/accelerator/Accelerator.sol#536)
        - bonusStartPercent = _bonusStartPercent (contracts/accelerator/Accelerator.sol#537)
        - bonusStartDays = _bonusStartDays (contracts/accelerator/Accelerator.sol#538)
        - baseBonus = _baseBonus (contracts/accelerator/Accelerator.sol#539)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Accelerator.setRecipient(address)._recipient (contracts/accelerator/Accelerator.sol#447) lacks a zero-check on :
                - recipient = _recipient (contracts/accelerator/Accelerator.sol#448)
Accelerator.setToken(address)._token (contracts/accelerator/Accelerator.sol#461) lacks a zero-check on :
                - token = _token (contracts/accelerator/Accelerator.sol#462)
Accelerator.setVentureCapital(address)._ventureCapital (contracts/accelerator/Accelerator.sol#469) lacks a zero-check on :
                - ventureCapital = _ventureCapital (contracts/accelerator/Accelerator.sol#470)
Accelerator.setStaking(address)._staking (contracts/accelerator/Accelerator.sol#476) lacks a zero-check on :
                - staking = _staking (contracts/accelerator/Accelerator.sol#477)
Accelerator.setStakeManager(address)._stakeManager (contracts/accelerator/Accelerator.sol#483) lacks a zero-check on :
                - stakeManager = _stakeManager (contracts/accelerator/Accelerator.sol#484)
Accelerator.startAddresses(address,address,address,address,address)._staking (contracts/accelerator/Accelerator.sol#507) lacks a zero-check on :
                - staking = _staking (contracts/accelerator/Accelerator.sol#513)
Accelerator.startAddresses(address,address,address,address,address)._axion (contracts/accelerator/Accelerator.sol#508) lacks a zero-check on :
                - axion = _axion (contracts/accelerator/Accelerator.sol#514)
Accelerator.startAddresses(address,address,address,address,address)._token (contracts/accelerator/Accelerator.sol#509) lacks a zero-check on :
                - token = _token (contracts/accelerator/Accelerator.sol#515)
```

```
INFO:Detectors:
Pragma version>=0.8.0 (contracts/interfaces/IStakingV1.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/interfaces/IStakingV21.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/v2.1/StakingV21.sol#3) 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
INFO:Detectors:
StakingV21 (contracts/v2.1/StakingV21.sol#14-289) should inherit from IStakingV1 (contracts/interfaces/IStakingV1.sol#5-11)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
INFO:Detectors:
StakingV21.shareRateScalingFactor (contracts/v2.1/StakingV21.sol#86) is never used in StakingV21 (contracts/v2.1/StakingV21.sol#14-289)
StakingV21.paused (contracts/v2.1/StakingV21.sol#100) is never used in StakingV21 (contracts/v2.1/StakingV21.sol#14-289)
StakingV21.bpd (contracts/v2.1/StakingV21.sol#103) is never used in StakingV21 (contracts/v2.1/StakingV21.sol#14-289)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
calculateStakingInterest(uint256,uint256,uint256) should be declared external:
        - StakingV21.calculateStakingInterest(uint256,uint256,uint256) (contracts/v2.1/StakingV21.sol#177-193)
getAmountOutAndPenalty(uint256,uint256,uint256,uint256) should be declared external:
        - StakingV21.getAmountOutAndPenalty(uint256,uint256,uint256,uint256) (contracts/v2.1/StakingV21.sol#243-284)
calculateStepsFromStart() should be declared external:
        - StakingV21.calculateStepsFromStart() (contracts/v2.1/StakingV21.sol#286-288)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Detectors:
Pragma version>=0.8.0 (contracts/interfaces/IAuctionV1.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/interfaces/IAuctionV21.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/interfaces/IToken.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/v2.1/AuctionV21.sol#3) 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
INFO:Detectors:
calculateStepsFromStart() should be declared external:
        - AuctionV21.calculateStepsFromStart() (contracts/v2.1/AuctionV21.sol#106-108)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
VentureCapital.getTokenInterestEarned(address,address) (contracts/accelerator/VentureCapital.sol#303-316) compares to a boolean constant:
        -isVcaRegistered[accountAddress] == false (contracts/accelerator/VentureCapital.sol#308)
VentureCapital.getTotalSharesOf(address) (contracts/accelerator/VentureCapital.sol#328-334) compares to a boolean constant:
        -isVcaRegistered[account] == false (contracts/accelerator/VentureCapital.sol#329)
VentureCapital.getIsVCARegistered(address) (contracts/accelerator/VentureCapital.sol#336-342) compares to a boolean constant:
        -isVcaRegistered[staker] == false (contracts/accelerator/VentureCapital.sol#337)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Pragma version>=0.8.0 (contracts/abstracts/ExternallyCallable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/abstracts/Manageable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/abstracts/Migrateable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/accelerator/VentureCapital.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/enums/StakeStatus.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/interfaces/IStakeManager.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/interfaces/IStakingV1.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/interfaces/IStakingV21.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/interfaces/IToken.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.8.0 (contracts/interfaces/IVentureCapital.sol#3) 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
INFO:Detectors:
Reentrancy in VentureCapital.withdrawDivTokenInternal(address,address,address) (contracts/accelerator/VentureCapital.sol#116-142):
        External calls:
        - to.transfer(tokenInterestEarned) (contracts/accelerator/VentureCapital.sol#138)
        Event emitted after the call(s):
        - WithdrawLiquidDiv(from,tokenAddress,tokenInterestEarned) (contracts/accelerator/VentureCapital.sol#141)
Reentrancy in VentureCapital.withdrawOriginDivTokens(address) (contracts/accelerator/VentureCapital.sol#144-156):
        External calls:
        - address(msg.sender).transfer(originWithdrawableTokenAmounts[tokenAddress]) (contracts/accelerator/VentureCapital.sol#152)
        State variables written after the call(s):
        - originWithdrawableTokenAmounts[tokenAddress] = 0 (contracts/accelerator/VentureCapital.sol#155)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
initialize(address,address) should be declared external:
        - VentureCapital.initialize(address,address) (contracts/accelerator/VentureCapital.sol#257-263)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

According to the test results, some of the findings found by these tools were considered as false positives while some of these findings were real security concerns. All relevant findings were reviewed by the auditors and relevant findings addressed on the report as security concerns.

4.2 AUTOMATED SECURITY SCAN

Description:

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

Results:

ExternallyCallable.sol, Manageable.sol, Migrateable.sol, Pausable.sol

Report for contracts/abstracts/ExternallyCallable.sol
https://dashboard.mythx.io/#/console/analyses/9a4d7a18-1af2-477a-973b-b45290b4016f

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

Report for contracts/abstracts/Manageable.sol
https://dashboard.mythx.io/#/console/analyses/e86ef117-72c1-4d3a-b7dd-8f7b316f0baa

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

Report for contracts/abstracts/Migrateable.sol
https://dashboard.mythx.io/#/console/analyses/d2eb1127-8ff6-47fd-9d8d-ae470989cee5

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

Report for contracts/abstracts/Pausable.sol
https://dashboard.mythx.io/#/console/analyses/723c4927-1903-4ba0-80e5-ba462c0ec5b8

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

BPD.sol, StakeBase.sol, StakeToken.sol

Report for contracts/stake/BPD.sol
https://dashboard.mythx.io/#/console/analyses/4e7cb086-199f-4237-8d59-16fbf20537b5

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

Report for contracts/stake/StakeToken.sol
https://dashboard.mythx.io/#/console/analyses/f6bfc850-a8cf-4a5a-b8fc-9079b1deaa1c

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

Report for contracts/stake/StakeBase.sol
https://dashboard.mythx.io/#/console/analyses/e8aea1b6-d3ef-4985-86bc-6a49dd858ac8

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

StakeBurner.sol, StakeCustodian.sol, StakeMinter.sol

Report for contracts/stake/StakeBurner.sol
https://dashboard.mythx.io/#/console/analyses/d8cedd9f-49e8-4df0-aa7a-cd31d2e402aa

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
272	(SWC-116) Timestamp Dependence	Low	A control flow decision is made based on The block.timestamp environment variable.

Report for contracts/stake/StakeCustodian.sol
https://dashboard.mythx.io/#/console/analyses/ff2a4a2c-9dc7-4049-8755-9991fe9573f7

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

Report for contracts/stake/StakeMinter.sol
https://dashboard.mythx.io/#/console/analyses/7fa22e78-1c33-4dee-b20e-6bc49df3ec54

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
100	(SWC-113) DoS with Failed Call	Low	Multiple calls are executed in the same transaction.
100	(SWC-107) Reentrancy	Low	Read of persistent state following external call

StakeReminter.sol, StakeUpgrader.sol, StakeManager.sol

Report for contracts/stake/StakeReminter.sol
https://dashboard.mythx.io/#/console/analyses/2586f269-79ed-46df-a120-7eb27aed368b

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
44	(SWC-116) Timestamp Dependence	Low	A control flow decision is made based on The block.timestamp environment variable.
46	(SWC-113) DoS with Failed Call	Low	Multiple calls are executed in the same transaction.
80	(SWC-113) DoS with Failed Call	Low	Multiple calls are executed in the same transaction.
80	(SWC-107) Reentrancy	Low	Write to persistent state following external call
80	(SWC-107) Reentrancy	Low	Read of persistent state following external call
81	(SWC-107) Reentrancy	Low	Write to persistent state following external call
81	(SWC-107) Reentrancy	Low	Read of persistent state following external call

Report for contracts/stake/StakeManager.sol
https://dashboard.mythx.io/#/console/analyses/30fe45f0-858d-45ce-95b1-6ce1a9fcda38

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
466	(SWC-116) Timestamp Dependence	Low	A control flow decision is made based on The block.timestamp environment variable.

Report for contracts/stake/StakeUpgrader.sol
https://dashboard.mythx.io/#/console/analyses/989abb04-9dbe-4e08-8822-989cc1f9c859

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

AuctionV21.sol, StakingV21.sol

Report for contracts/v2.1/StakingV21.sol
https://dashboard.mythx.io/#/console/analyses/87f633e4-e1cc-4598-a86c-7c469915f98a

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
103	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
104	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

Report for contracts/v2.1/AuctionV21.sol
https://dashboard.mythx.io/#/console/analyses/3ea2a9e3-b269-4890-b0ba-24058a98ca50

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

Accelerator.sol

Report for contracts/accelerator/Accelerator.sol
https://dashboard.mythx.io/#/console/analyses/7c727f3b-804a-49a0-9233-4ffa4765091a

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
56	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
57	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
58	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

VentureCapital.sol

Report for contracts/accelerator/VentureCapital.sol
https://dashboard.mythx.io/#/console/analyses/32e99b16-89fc-441a-af15-13c7fa6540ea

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

AxionSafeCast.sol

Report for contracts/libs/AxionSafeCast.sol
https://dashboard.mythx.io/#/console/analyses/53cc3d01-1ec9-4bc3-af95-ec810156626c

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

All relevant valid findings were founded in the manual code review.

THANK YOU FOR CHOOSING

// HALBORN