Audit report of Arenaton

Prepared By: - Kishan Patel
Prepared On: - 26/08/2023.

Prepared for:

Table of contents

THIS AUDIT REPORT WILL CONTAIN CONFIDENTIAL INFORMATION ABOUT THE SMART CONTRACT AND INTELLECTUAL PROPERTY OF THE CUSTOMER AS WELL AS INFORMATION ABOUT POTENTIAL VULNERABILITIES OF THEIR EXPLOITATION.

THE INFORMATION FROM THIS AUDIT REPORT CAN BE USED INTERNALLY BY THE CUSTOMER OR IT CAN BE DISCLOSED PUBLICLY AFTER ALL VULNERABILITIES ARE FIXED - UPON THE DECISION OF THE CUSTOMER.

1. Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions). Because the total numbers of test cases are unlimited, the audit makes no statements or warranties on the security of the code.

It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

2. Introduction

Kishan Patel (Consultant) was contacted by Arenaton. (Customer) to conduct a Smart Contracts Code Review and Security Analysis. This report presents the findings of the security assessment of Customer`s smart contracts and its code review conducted between 26/08/2023 – 29/08/2023.

The project has 13 files. It contains approx 1200 lines of Solidity code. All the functions and state variables are well commented on using the natspec documentation, but that does not create any vulnerability.

3. Project information

Token Name	ATON & VUND Token
Token Symbol	ATON & VUND
Platform	Ethereum
Order Started Date	26/08/2023
Order Completed Date	29/08/2023

4. List of attacks checked

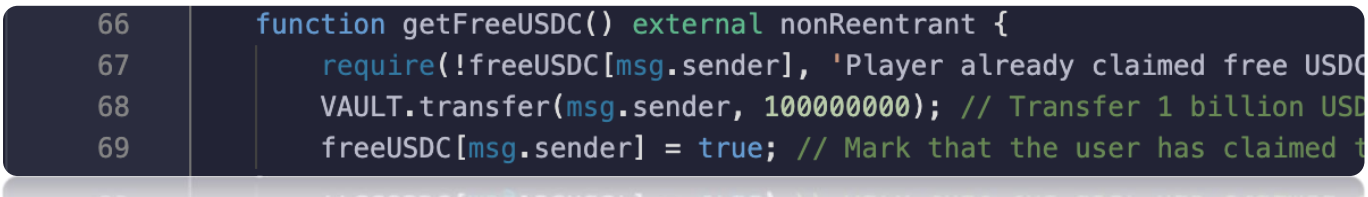
- Over and under flows
- Short address attack
- Visibility & Delegate call
- Reentrancy / TheDAO hack
- Forcing ETH to a contract
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Byte array vulnerabilities
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference

5. Severity Definitions

Risk	Level Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
Medium	Medium-level vulnerabilities are important to fix; however, they can’t lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can’t have significant impact on execution

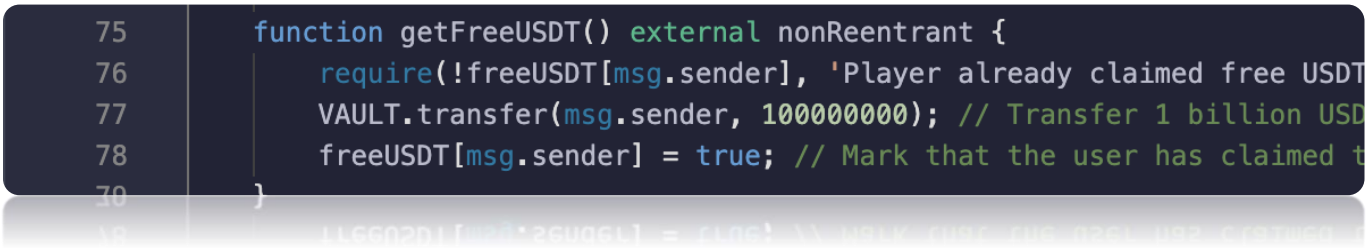
6. Good things in code

- **Good required condition in functions:-**
  - **Filename: Arenaton.sol**
    - Smart contract is checking that msg.sender already claimed 1 billion free USDC.



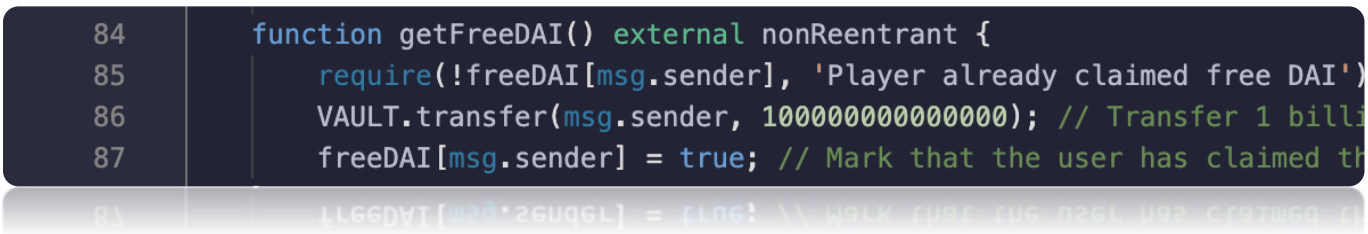
```
66      function getFreeUSDC() external nonReentrant {
67          require(!freeUSDC[msg.sender], 'Player already claimed free USDC
68          VAULT.transfer(msg.sender, 100000000); // Transfer 1 billion USD
69          freeUSDC[msg.sender] = true; // Mark that the user has claimed t
```

    - Smart contract is checking that msg.sender already claimed 1 billion free USDT.



```
75      function getFreeUSDT() external nonReentrant {
76          require(!freeUSDT[msg.sender], 'Player already claimed free USDT
77          VAULT.transfer(msg.sender, 100000000); // Transfer 1 billion USD
78          freeUSDT[msg.sender] = true; // Mark that the user has claimed t
79      }
```

    - Smart contract is checking that msg.sender already claimed 1 billion free DAI.



```
84      function getFreeDAI() external nonReentrant {
85          require(!freeDAI[msg.sender], 'Player already claimed free DAI')
86          VAULT.transfer(msg.sender, 100000000000000); // Transfer 1 billi
87          freeDAI[msg.sender] = true; // Mark that the user has claimed th
```

- Smart contract is checking that amountVUND is bigger or equal to 1000000, msg.sender has given allowance bigger than _amountCoinIn to VAULT address, msg.sender has given allowance bigger than _amountATON to VAULT address, msg.sender has more balance than _amountCoinIn in VAULT address, msg.sender has more balance than _amountATON in ATON address.

```
210     function addOracleOpenRequest(
211         string memory _eventId,
212         uint256 _amountCoinIn,
213         address _coinAddress,
214         uint256 _amountATON,
215         uint8 _team
216     ) external {
217         VAULT.distributeCommission(msg.sender);
218 
219         // Ensure the Event has not started yet and is still active
220         bytes8 eventIdBytes = Tools._stringToBytes8(_eventId);
221         address player = msg.sender;
222 
223         (uint256 amountVUND, ) = VAULT.convertCoinToVund(_coinAddress, _
224 
225         // Check minimum stake requirement
226         require(amountVUND >= 1000000, 'Minimum Stake amount not reache
```

- Smart contract is checking that startDate is smaller than currentTime, event is active, close event for eventIdBytes is already added.

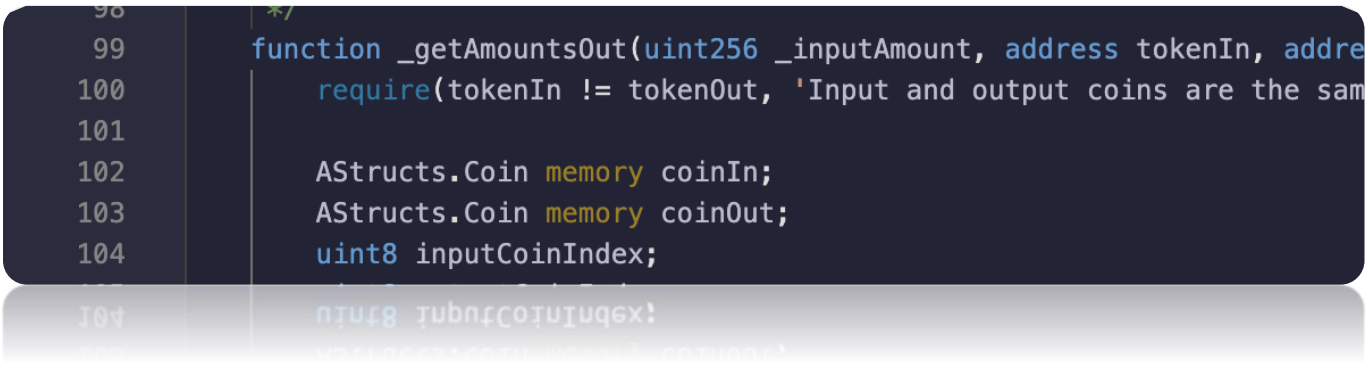
```
269     function addOracleCloseRequest(string memory _eventId) external {
270         VAULT.distributeCommission(msg.sender);
271 
272         // Ensure the Event has not started yet and is still active
273         bytes8 eventIdBytes = Tools._stringToBytes8(_eventId);
274         AStructs.EventDTO memory eventInfo = VAULT.getEventDTO(eventIdBy
275         require(eventInfo.startDate < block.timestamp, 'Event has not st
```

- Smart contract is checking that eventInfo is active,event is not already started, and vundAmount is not bigger or equal to USD value.

```
513     function _newStake(
514         bytes8 _eventIdBytes,
515         uint256 _amountCoinIn,
516         address _coinAddress,
517         uint256 _amountATON,
518         uint8 _team,
519         address _player
520     ) internal {
521         VAULT.distributeCommission(msg.sender);
522 
523         (, uint256 pct_denom) = VAULT.getPremium();
524 
525         // Ensure event is active and hasn't started yet
526         AStructs.EventDTO memory eventInfo = VAULT.getEventDTO(_eventIdB
527         require(eventInfo.active, 'Event is not active');
528         require(eventInfo.startDate > block.timestamp, 'Event already s
```

Filename: Swap.sol

- Smart contract is checking that tokenIn and tokenOut arresses are not same.

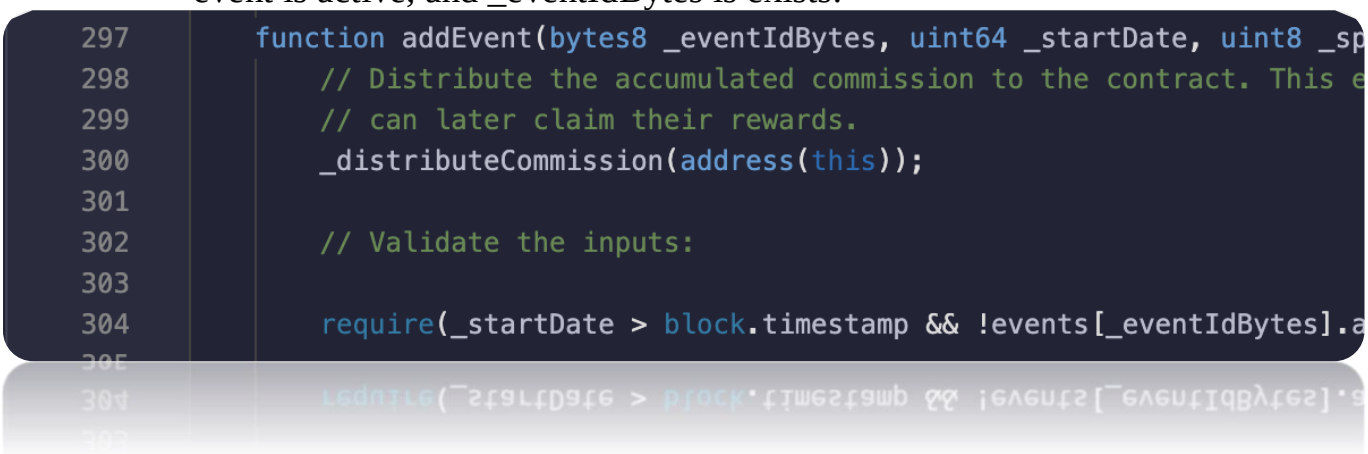


```
 99     function _getAmountsOut(uint256 _inputAmount, address tokenIn, addre
100         require(tokenIn != tokenOut, 'Input and output coins are the sam
101 
102         AStructs.Coin memory coinIn;
103         AStructs.Coin memory coinOut;
104         uint8 inputCoinIndex;
```

- **Filename: Vault.sol**
  - Smart contract is checking that _player address has given allowance to this contract in token contract bigger than _amountCoin, transferFrom method called to token contract is successfully done for _player address, burnFrom method called to ATON contract is successfully.

```
255     function retrieveCoin(
256         address _player,
257         uint256 _amountCoin,
258         address _token,
259         uint256 _amountBurn
260     ) external onlyRole(ARENATON_ROLE) nonReentrant {
261         ERC20 token = ERC20(_token);
262 
263         // Ensure the contract is authorized to retrieve the specified a
264         require(token.allowance(_player, address(this)) >= _amountCoin);
265 
266         // Transfer the COIN tokens from the player to this contract.
267         require(token.transferFrom(_player, address(this), _amountCoin))
268 
269         if (_token == address(ATON)) {
270             // Burn the specified amount of ATON tokens from the contrac
271             require(ATON.burnFrom(_amountBurn));
```

  - Smart contract is checking that _startDate is bigger than currentTime, event is active, and _eventIdBytes is exists.

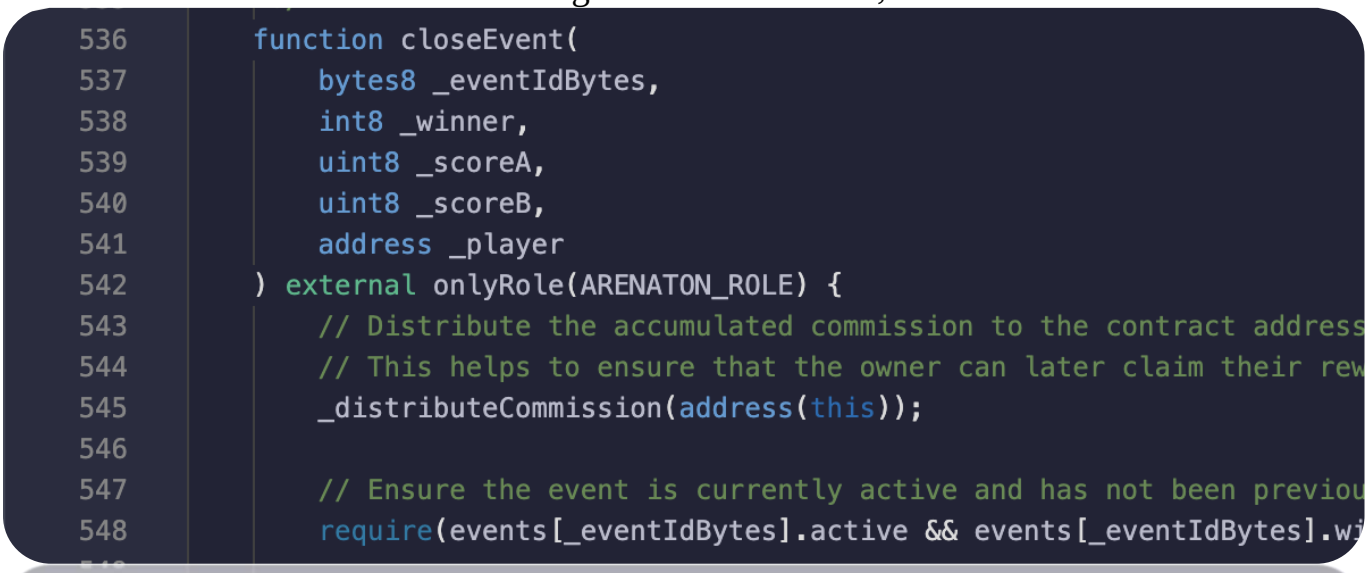


```
297     function addEvent(bytes8 _eventIdBytes, uint64 _startDate, uint8 _sp
298         // Distribute the accumulated commission to the contract. This e
299         // can later claim their rewards.
300         _distributeCommission(address(this));
301 
302         // Validate the inputs:
303 
304         require(_startDate > block.timestamp && !events[_eventIdBytes].a
```

- Smart contract is checking that event is active, startDate is bigger than current time, and coin is active.

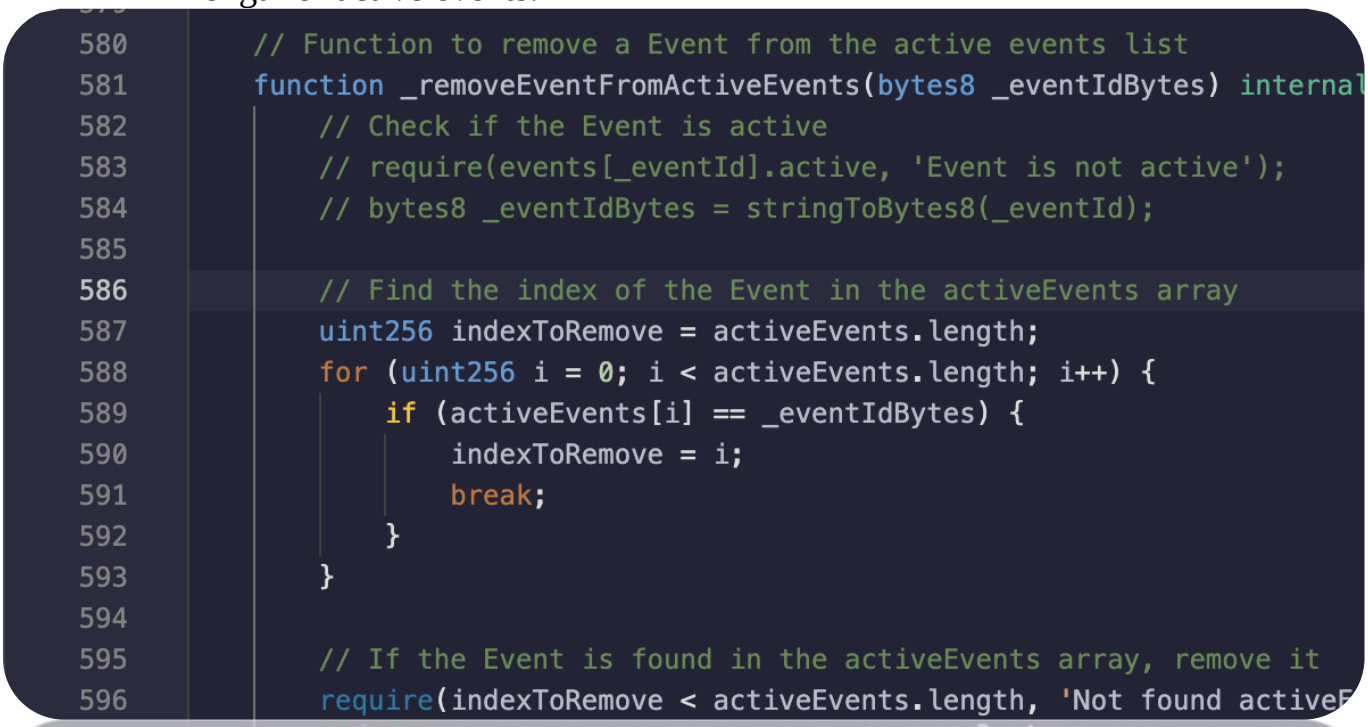
```
454     function addStake(
455         bytes8 _eventIdBytes,
456         address _coinAddress,
457         uint256 _amountCoinIn,
458         uint256 _amountATON,
459         uint8 _team,
460         address _player
461     ) external onlyRole(ARENATON_ROLE) nonReentrant {
462         _distributeCommission(address(this));
463 
464         (uint256 amountVUND, ) = _convertCoinToVUND(_coinAddress, _amoun
465         if (_coinAddress != address(this)) {
466             _mint(address(this), amountVUND);
467         }
468         // Increment the stake count for the Event
469         events[_eventIdBytes].stakeCount++;
470 
471         // Ensure the Event's start date is in the future and the Event
472         require(
473             coinList[coinIndex[_coinAddress]].active && events[_eventIdB
474             'addStake failed'
475         );
```

- Smart contract is checking that event is active, and winner is not -1.



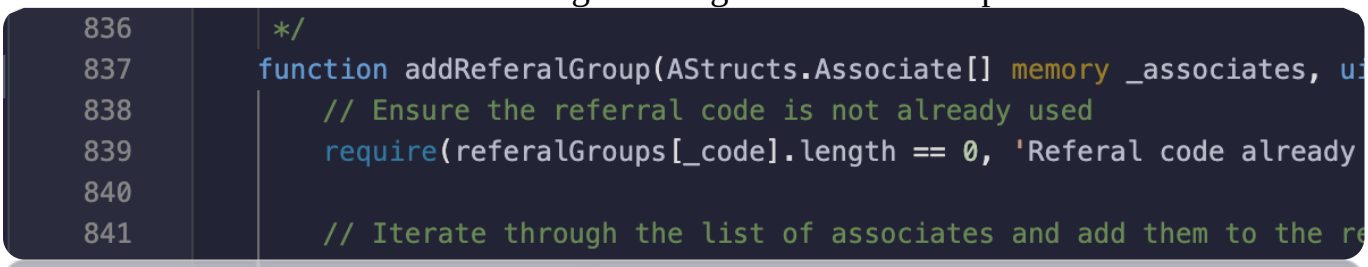
```
536     function closeEvent(
537         bytes8 _eventIdBytes,
538         int8 _winner,
539         uint8 _scoreA,
540         uint8 _scoreB,
541         address _player
542     ) external onlyRole(ARENATON_ROLE) {
543         // Distribute the accumulated commission to the contract address
544         // This helps to ensure that the owner can later claim their rew
545         _distributeCommission(address(this));
546 
547         // Ensure the event is currently active and has not been previou
548         require(events[_eventIdBytes].active && events[_eventIdBytes].wi
```

- Smart contract is checking that indexToRemove is smaller than length of active events.



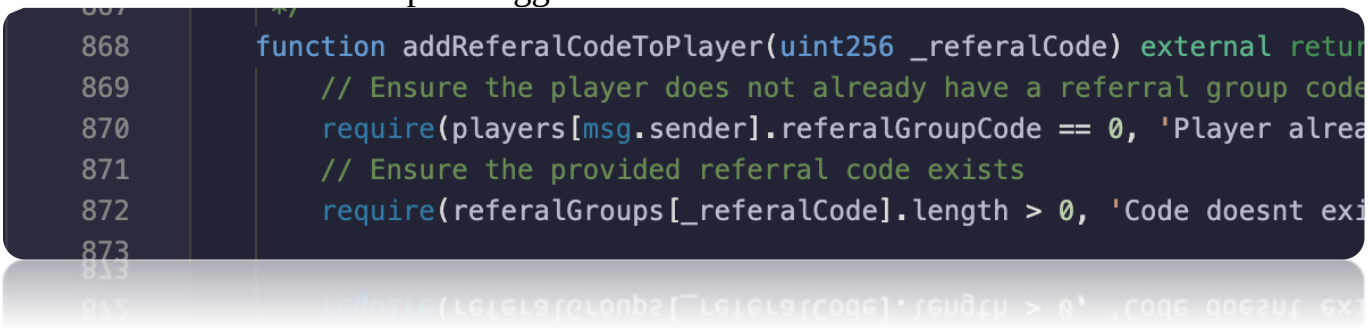
```
580     // Function to remove a Event from the active events list
581     function _removeEventFromActiveEvents(bytes8 _eventIdBytes) internal
582         // Check if the Event is active
583         // require(events[_eventId].active, 'Event is not active');
584         // bytes8 _eventIdBytes = stringToBytes8(_eventId);
585 
586         // Find the index of the Event in the activeEvents array
587         uint256 indexToRemove = activeEvents.length;
588         for (uint256 i = 0; i < activeEvents.length; i++) {
589             if (activeEvents[i] == _eventIdBytes) {
590                 indexToRemove = i;
591                 break;
592             }
593         }
594 
595         // If the Event is found in the activeEvents array, remove it
596         require(indexToRemove < activeEvents.length, 'Not found activeE
```

- Smart contract is checking that length of referalGroups is not 0.



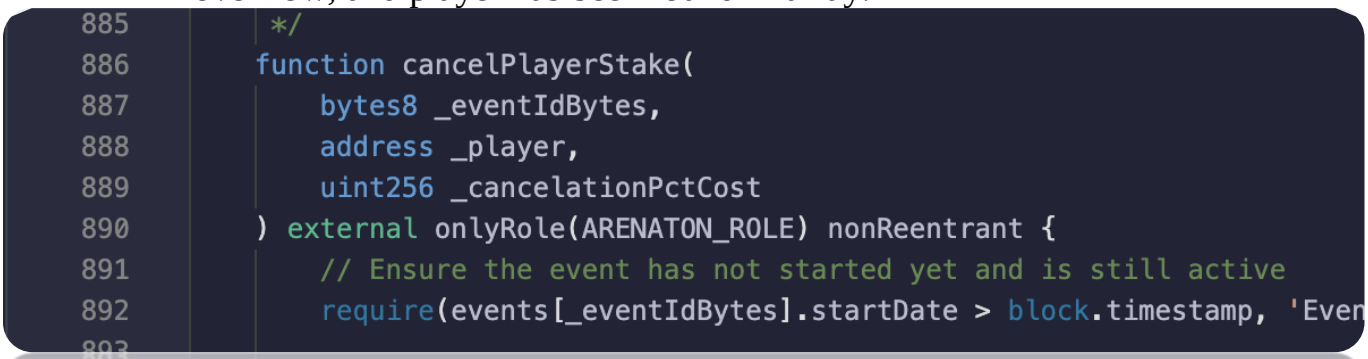
```
836      */
837     function addReferalGroup(AStructs.Associate[] memory _associates, ui
838         // Ensure the referral code is not already used
839         require(referalGroups[_code].length == 0, 'Referal code already
840
841         // Iterate through the list of associates and add them to the re
```

- Smart contract is checking that referalGroupCode is not 0, and length of referalGroups is bigger than 0.



```
868     function addReferalCodeToPlayer(uint256 _referalCode) external retur
869         // Ensure the player does not already have a referral group code
870         require(players[msg.sender].referalGroupCode == 0, 'Player alrea
871         // Ensure the provided referral code exists
872         require(referalGroups[_referalCode].length > 0, 'Code doesnt exi
873
```

- Smart contract is checking that startDate is bigger than currentTime, event is active, amountVUNDtoReturn and amountATON are not overflow, and player has been found in array.



```
885      */
886     function cancelPlayerStake(
887         bytes8 _eventIdBytes,
888         address _player,
889         uint256 _cancelationPctCost
890     ) external onlyRole(ARENATON_ROLE) nonReentrant {
891         // Ensure the event has not started yet and is still active
892         require(events[_eventIdBytes].startDate > block.timestamp, 'Even
893
```

- Smart contract is checking that coin is active, balance of player in tokenIn contract is bigger than _amountIn, balance of current contract in tokenOut contract is bigger than _amountOut, transferFrom method of tokenIn contract is successfully called and transfer method of tokenOut contract is successfully called.

```
1008    function swap(
1009        address _player,
1010        address _tokenIn,
1011        address _tokenOut,
1012        uint256 _amountIn,
1013        uint256 _amountOut,
1014        uint256 _comissionPct
1015    ) external nonReentrant onlyRole(SWAP_ROLE) returns (bool) {
1016        // Retrieve the properties of the input and output tokens from t
1017        AStructs.Coin memory coinIn = coinList[coinIndex[_tokenIn]];
1018        AStructs.Coin memory coinOut = coinList[coinIndex[_tokenOut]];
1019
1020        // Check if the input token (CoinIn) is allowed for swapping.
1021        require(coinIn.active, 'CoinIn not active');
```

Filename: ATONToken.sol

- Smart contract is checking that balance of msg.sender is bigger than or equal to _burnAmount.

```
45      function burnFrom(uint256 _burnAmount) external returns (bool) {
46          require(balanceOf(msg.sender) >= _burnAmount, 'Insufficient ATON
47          _burn(msg.sender, _burnAmount);
48
49          return true;
```

7. Critical vulnerabilities in code

- **No Critical vulnerabilities found**

8. Medium vulnerabilities in code

- **No Medium vulnerabilities found**

9. Low vulnerabilities in code

- **No Low vulnerabilities found**

10. Summary

- **Number of problems in the smart contract as per severity level**

Critical	Medium	Low
0	**0**	**0**

According to the assessment, the smart contract code is well secured. The code is written with all validation and all security is implemented. Code is performing well and there is no way to steal funds from this contract.

- **Good Point:** Code performance and quality are good. All kind of necessary validation added into smart contract and all validations are working as excepted.