Audit report of Arenaton

Prepared By: - Kishan Patel
Prepared On: - 03/07/2024.

Prepared for: Arenaton

Created by Kishan Patel. All rights reserved.

Table of contents

Created by Kishan Patel. All rights reserved.

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.

Created by Kishan Patel. All rights reserved.

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.

Created by Kishan Patel. All rights reserved.

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 03/07/2024 – 05/07/2024.

The project has 10 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	Arenaton
Token Symbol	ATON
Platform	Ethereum
Order Started Date	03/07/2024
Order Completed Date	05/07/2024

Created by Kishan Patel. All rights reserved.

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

Created by Kishan Patel. All rights reserved.

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

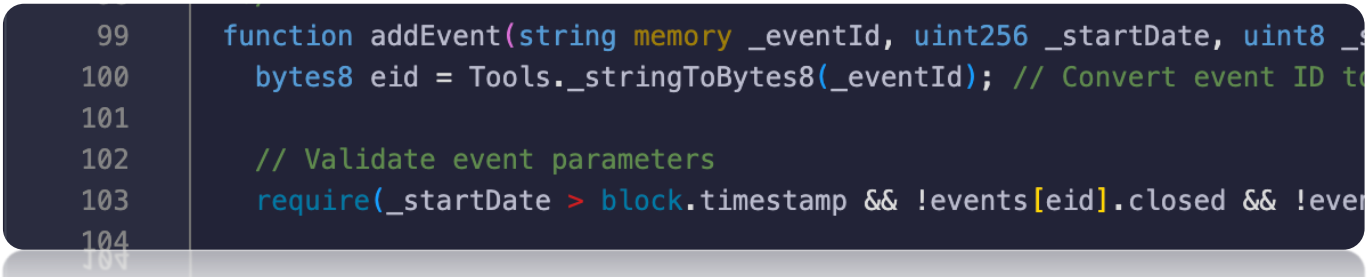
Created by Kishan Patel. All rights reserved.

6. Good things in code

- **Good required condition in functions:-**
  - **Filename: Arenaton.sol**
    - Here smart contract is checking that msg.sender is owner. (**Note: It is not necessary because onlyOwner modifier is enough**)

```
81  function setAuthorizedAddress(address authorizedAddress) external only
82    require(msg.sender == owner, "Caller is not the owner");
83    authorizedAddresses[authorizedAddress] = !authorizedAddresses[author
84  }
```

    - Here smart contract is checking that _startDate is bigger than current time, events with eid is not closed and not active.



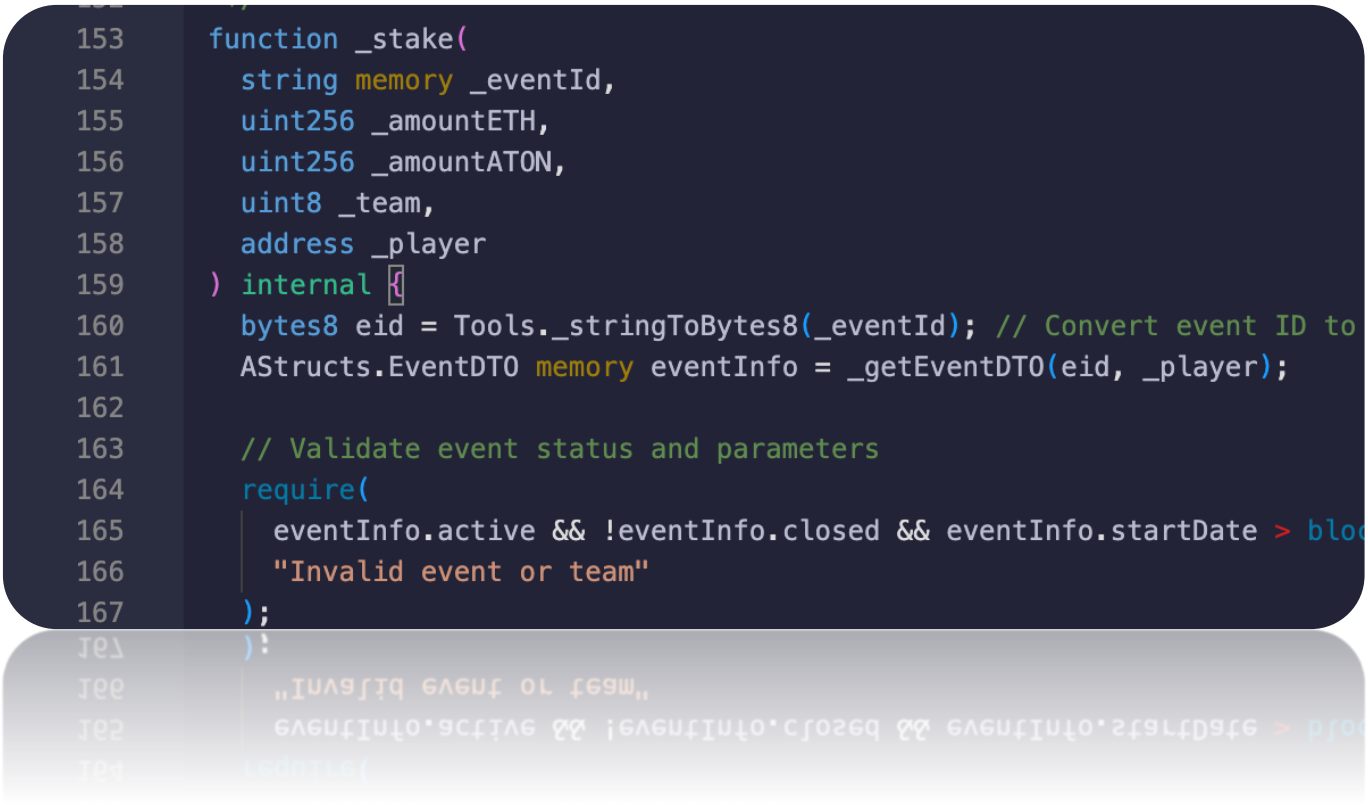
```
99  function addEvent(string memory _eventId, uint256 _startDate, uint8 _
100   bytes8 eid = Tools._stringToBytes8(_eventId); // Convert event ID t
101
102   // Validate event parameters
103   require(_startDate > block.timestamp && !events[eid].closed && !ever
104
```

    - Here smart contract is checking that isETH or _amountATON is bigger than 0, if isGasless is true then msg.sender is authorized user to stake it.

```
122 function stake(
123   string memory _eventId,
124   uint256 _amountATON,
125   uint8 _team,
126   bool isGasless,
127   address _player
128 ) external payable nonReentrant {
129   bool isETH = msg.value > 0;
130   require(isETH || _amountATON > 0, "Cannot stake 0 value");
131
```

Created by Kishan Patel. All rights reserved.

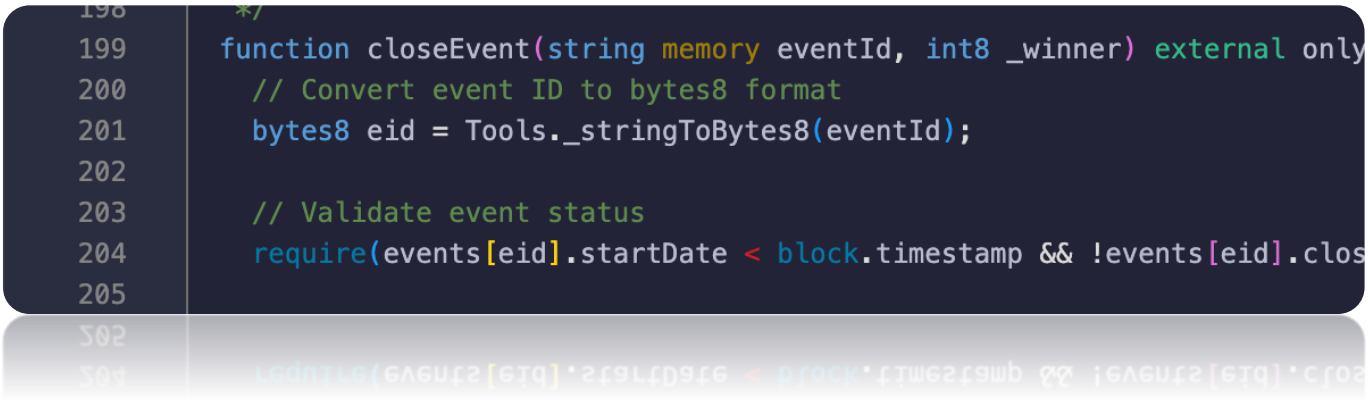
- Here smart contract is checking that event is active and start date of event is bigger than current time and _team value is 1 or 2.



```
function _stake(
  string memory _eventId,
  uint256 _amountETH,
  uint256 _amountATON,
  uint8 _team,
  address _player
) internal {
  bytes8 eid = Tools._stringToBytes8(_eventId); // Convert event ID to
  AStructs.EventDTO memory eventInfo = _getEventDTO(eid, _player);

  // Validate event status and parameters
  require(
    eventInfo.active && !eventInfo.closed && eventInfo.startDate > bloc
    "Invalid event or team"
  );
```

- Here smart contract is checking that event is not closed and is active, startDate of event is smaller than current time.

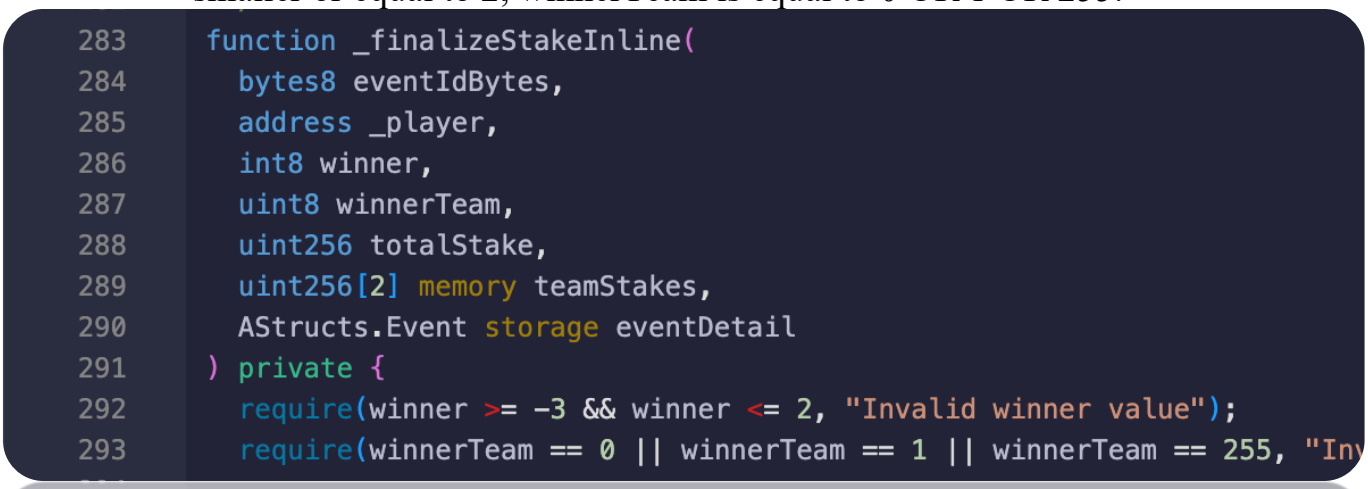


```
function closeEvent(string memory eventId, int8 _winner) external only
  // Convert event ID to bytes8 format
  bytes8 eid = Tools._stringToBytes8(eventId);

  // Validate event status
  require(events[eid].startDate < block.timestamp && !events[eid].clos
```

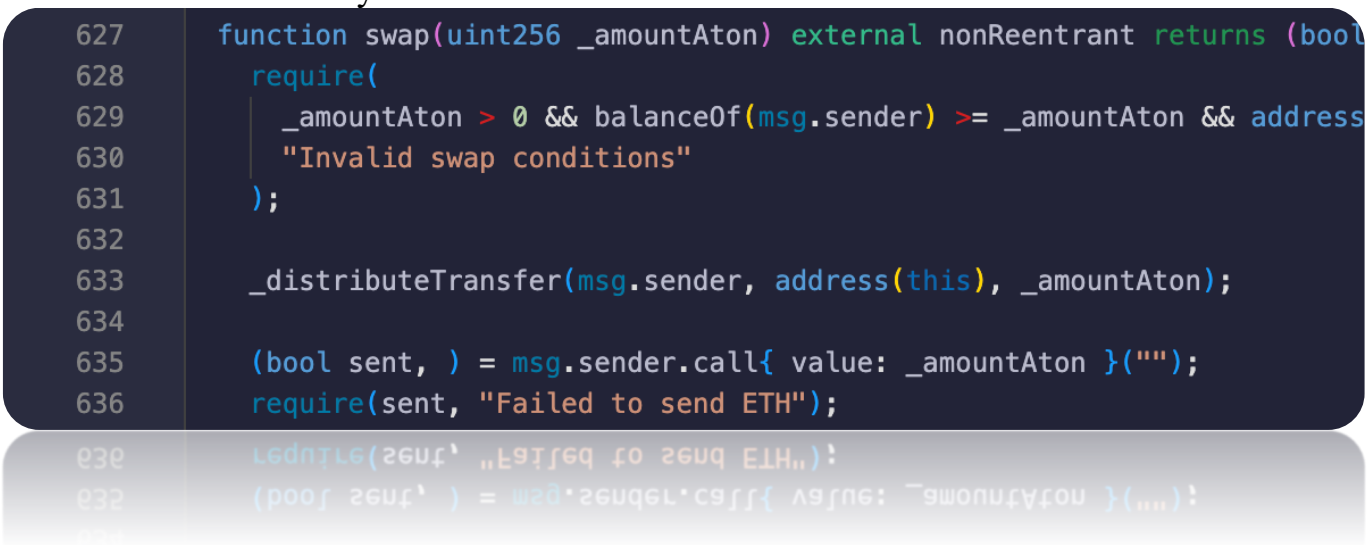
Created by Kishan Patel. All rights reserved.

- Here smart contract is checking that winner is bigger than or equal to -3 AND smaller or equal to 2, winnerTeam is equal to 0 OR 1 OR 255.



```
283  function _finalizeStakeInline(
284    bytes8 eventIdBytes,
285    address _player,
286    int8 winner,
287    uint8 winnerTeam,
288    uint256 totalStake,
289    uint256[2] memory teamStakes,
290    AStructs.Event storage eventDetail
291  ) private {
292    require(winner >= -3 && winner <= 2, "Invalid winner value");
293    require(winnerTeam == 0 || winnerTeam == 1 || winnerTeam == 255, "Inv
```

- Here smart contract is checking that _amountAton is bigger than 0, balance of msg.sender is bigger or equal to _amountAton value, balance of contract address is bigger or equal _amountAton value, transfer to msg.sender is successfully.



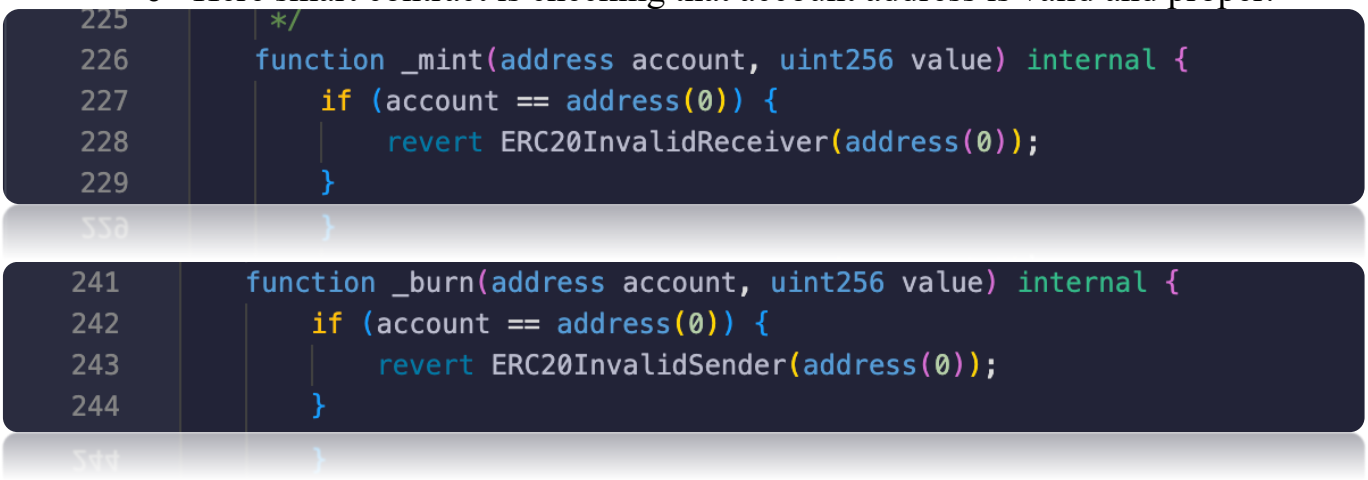
```
627  function swap(uint256 _amountAton) external nonReentrant returns (bool
628    require(
629      _amountAton > 0 && balanceOf(msg.sender) >= _amountAton && address
630      "Invalid swap conditions"
631    );
632
633    _distributeTransfer(msg.sender, address(this), _amountAton);
634
635    (bool sent, ) = msg.sender.call{ value: _amountAton }("");
636    require(sent, "Failed to send ETH");
```

Created by Kishan Patel. All rights reserved.

- **Filename: ERC20.sol**
  - Here smart contract is checking that from and to addresses are valid and proper.

```
170     */
171     function _transfer(address from, address to, uint256 value) internal
172         if (from == address(0)) {
173             revert ERC20InvalidSender(address(0));
174         }
175         if (to == address(0)) {
176             revert ERC20InvalidReceiver(address(0));
177         }
```

  - Here smart contract is checking that account address is valid and proper.



```
225     */
226     function _mint(address account, uint256 value) internal {
227         if (account == address(0)) {
228             revert ERC20InvalidReceiver(address(0));
229         }
```



```
241     function _burn(address account, uint256 value) internal {
242         if (account == address(0)) {
243             revert ERC20InvalidSender(address(0));
244         }
```

  - Here smart contract is checking that owner and spender addresses are valid and proper.

```
283     */
284     function _approve(address owner, address spender, uint256 value, boo
285         if (owner == address(0)) {
286             revert ERC20InvalidApprover(address(0));
287         }
288         if (spender == address(0)) {
289             revert ERC20InvalidSpender(address(0));
290         }
```

Created by Kishan Patel. All rights reserved.

7. Critical vulnerabilities in code

- **No Critical vulnerabilities found**

8. Medium vulnerabilities in code

- **No Medium vulnerabilities found**

9. Low vulnerabilities in code

9.1. Suggestions to add code validations:-

Filename: ERC20.sol

=> You have implemented required validation in contract.
=> There are some place where you can improve validation and security of your code.
=> These are all just suggestion it is not bug.

- **Function: - _approve**

```
284     function _approve(address owner, address spender, uint256 value, boo
285         if (owner == address(0)) {
286             revert ERC20InvalidApprover(address(0));
287         }
288         if (spender == address(0)) {
289             revert ERC20InvalidSpender(address(0));
290         }
291         _allowances[owner][spender] = value;
292         if (emitEvent) {
293             emit Approval(owner, spender, value);
294         }
```

- Here in _approve function smart contract can check that owner has sufficient balance to give allowance to spender.

Created by Kishan Patel. All rights reserved.

10. Summary

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

Critical	Medium	Low
0	0	1

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.

- **Suggestions:** Please try to implement suggested code validations.

Created by Kishan Patel. All rights reserved.