VANTAGEPOINT
security at the speed of development

Smart Contract Audit of Venue.One

v1.0
Final Report

Prepared for:

Veno Technology Inc

7 Feb 2023

FOR GENERAL DISTRIBUTION

Table Of Content

1. Executive Summary

1.1. Overview

Vantage Point Security Pte. Ltd. was engaged by Venue.One to conduct an Algorand smart contract audit of their online prediction protocol. The online prediction protocol includes an automated market maker application where users can enter or exit binary predictions and add or remove liquidity to the markets, a Parimutuel application which supports prediction on one or more outcomes and a constant rate exchange application which allows USDC(ASA ID 31566704) to be swapped for Algo at a constant rate set by the protocol.
Supporting documents

- AMM documentation
- Parimutuel documentation
- CRE documentation

PyTeal code repository

- https://github.com/VenueMarkets/contracts-internal

Vantage Point performed this audit by first understanding the Venue.One applications' business logic based on the documents provided. We sought clarifications on potential issues, discrepancies, flaws, and plans on how manual operations involved would be carried out through discussions with the Venue.One team.
The smart contract audit was conducted on the provided PyTeal code to identify weaknesses, vulnerabilities, and non-compliance to Algorand best practices.
Two low-risk and one observational findings were identified from the smart contract audit.
Insufficient Txn.Sender Validation for Privileged Operations

- Two instances were found where the transaction senders were not validated for administrative operations. These two instances can be found on both the Automated Market Maker and Parimutuel smart contracts allowing anyone to trigger a call to withdraw the platform fees however, the risk was minimal considering the sink address can only be controlled by the administrator.

Discrepancy Between Smart Contract Design Specification and Code

- Discrepancies between the web application and the smart contracts were found wherein a minimum value of 1 USDC was enforced when entering the market with a prediction and adding liquidity through the web application however, was found not to be validated in the smart contract.

Unnecessary Local or Global States and Checks

- A few global state variables were found unused which may increase the minimum balance requirement during application creation or opt-ins without any value added to the smart contract.

The outcome of this Algorand smart contract audit is provided as a detailed technical report that provides the project owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and detailed recommendation that will resolve the identified technical issue.
In addition, following commit was made after the completion of audit and has been reviewed for its impact on the logical and functional security aspect of the smart contracts in scope. No new findings were identified during the review.

- b16d47cd19d9a3cb368f8a8b898f9186e63a084d

2. Project Details

2.1. Scope

App Name	Smart Contract Audit of Venue.One
Testing Window	4 Nov 2022 to 17 Nov 2022
Target URL	https://github.com/VenueMarkets/contracts-internal
Svn / Git Revision Number	f0ee6e07d9d84935c98a3cd68018231d50011941
Project Type	Smart Contract Audit
App Type	Algorand Smart Contract
Items Completed	https://github.com/VenueMarkets/contracts-internal/blob/main/contracts/assets/amm.py https://github.com/VenueMarkets/contracts-internal/blob/main/contracts/assets/cre.py https://github.com/VenueMarkets/contracts-internal/blob/main/contracts/assets/parimutuel.py
Issue Opening Date	17 Nov 2022 ● Discrepancy Between Smart Contract Design Specification and Code [Low] ● Unnecessary Local or Global States and Checks [Observational] 11 Nov 2022 ● Insufficient Txn.Sender() Validation for Privileged Operations [Low]
Issue Closing Date	18 Nov 2022 ● Discrepancy Between Smart Contract Design Specification and Code [Low] 17 Nov 2022 ● Insufficient Txn.Sender() Validation for Privileged Operations [Low] ● Unnecessary Local or Global States and Checks [Observational]

2.2. Status

Component	Review Type	Status
Algorand Smart Contract	Smart Contract Audit	Completed

3. Risk Assessment

This chapter contains an overview of the vulnerabilities discovered during the project. The vulnerabilities are sorted based on the scoring categories CRITICAL, HIGH, MEDIUM and LOW. The category OBSERVATIONAL refers to vulnerabilities that have no risk score and therefore have no immediate impact on the system.

3.1. Summary of Vulnerabilities

Vulnerability Title	Risk Score	Closed
Discrepancy Between Smart Contract Design Specification and Code	Low	☑
Insufficient Txn.Sender() Validation for Privileged Operations	Low	☑
Unnecessary Local or Global States and Checks	Observational	☑

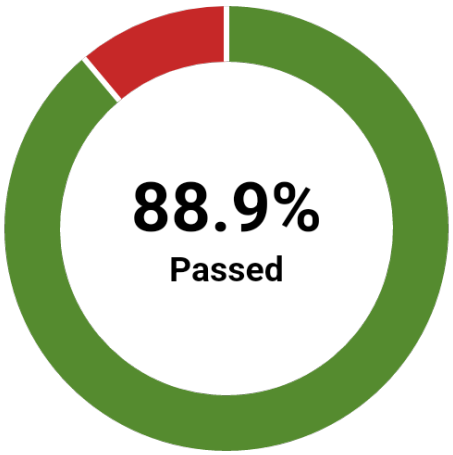
3.2. Vulnerabilities Statistics

Findings Overview

Total	Open	Closed
3	0	3

0	Critical	No findings
0	High	No findings
0	Medium	No findings
2	Low	0 Open, 2 Closed.
1	Observational	0 Open, 1 Closed.

Test Case Status



Open	0.0%
Passed	88.9%
Failed	11.1%

4. Detailed Description of Vulnerabilities

4.1. Discrepancy Between Smart Contract Design Specification and Code

Low

Closed

BACKGROUND

Discrepancy between design versus code could create confusion to participants and the community as the expected outcomes may not exactly match with the smart contract's actual behavior.

DESCRIPTION

Affected File/Code
Instance 1

- https://github.com/VenueMarkets/contracts-internal/blob/main/contracts/assets/amm.py [enter_market_with_prediction*]
- https://github.com/VenueMarkets/contracts-internal/blob/main/contracts/assets/parimutuel.py [enter_market_with_prediction*]

It was noted that the web application front-end for the AMM and parimutuel smart contract has a minimum amount requirement of USDC 1e6 (US$1) for entering the market with prediction and adding liquidity. However, the same was neither validated nor enforced in the smart contract's logic.
Based on the value of the swap_fee_bps and the amount, not enforcing minimum of 1e6 USDC (US$1 with 6 decimal points) as a valid "amount" value may mean that the _calculate_swap_fee function returns 0, making it a swap without fees.
Example when entering the market with a prediction value of 1 USDC:

- swap_fee_bps: 100
- amount entered by user: 1 USDC = 1,000,000

Calculation for the swap fee that is supposedly going to the contract:

- (1,000,000 x 100)/10,000 = 10,000 or 0.01 USDC

Example when entering the market with a prediction value of 0.000099 USDC: [enter_market_with_prediction*]

- swap_fee_bps: 100
- amount entered by user: 0.000099 USDC = 99T
- (99 x 100)/10,000 = 9,900/10,000 = 0.99 will become 0 because this is in decimal = 0 USDC

This means that whenever the user enters the market with prediction value of 0.000099 USDC at 100 swap_fee_bps, no swap fee will be subtracted from the prediction amount. An attacker in this case can enter the market with prediction multiple times to reach a certain prediction amount without paying any swap fees. For example, for an attacker to enter the market with a prediction value of 1 USDC, the attacker has to place approximately 10000 predictions at 0.000099 USDC however, each transaction will cost 10000 multiplied by the Algo fees. Suppose the Algo fees will be at 2,000 microAlgos, the total amount that will cost all the placement of predictions will be 10,000 x 2,000 microAlgos = 20,000,000 microAlgos (~20 Algos) which in turn, will be about 5.42 USD (As of November 17, 2022).

Given the current condition, a "free" swap with very small USDC amount does not actually benefit the user and such behavior is discouraged by transaction fees.
It is also noted that only USDC and VENO are expected to be the assets entered in the smart contracts for the currencies supported which both have 6 decimals in value.

Instance 2

Affected File/Code

- https://github.com/VenueMarkets/contracts-internal/blob/main/contracts/assets/amm.py [on_add_liquidity]

Similar to instance 1, when adding liquidity, the minimum in the web application is also 1 USDC. This however does not conform with the smart contract code because no validation for the minimum was performed.

IMPACT

Discrepancy on the design vs the smart contract code may result into unexpected behavior that could be leveraged by an attacker to potentially gain financially.

RECOMMENDATIONS

Ensure the smart contract's logic, front-end and documentations are consistent with sufficient validations so that users who interact with the smart contract are not faced with unexpected behavior arising from unknown or undocumented logic within the smart contract.

COMMENT

Reviewed on 18 Nov 2022

Based on the commit bc428165adbab93f4620863434077c43ce8bf668, this issue is closed.
It was noted that the new commit adds a clear information on the difference between the web front-end and the smart contract logic in allowing transactions with USDC value less than 1e6 or US$1. As the application's logic does not benefit anyone who tries to send transactions below 1e6 USDC as enforced in the web application given current conditions, this issue is closed.

REFERENCES

Algorand Best Practices
https://developer.algorand.org/docs/get-details/dapps/avm/teal/guidelines/

4.2. Insufficient Txn.Sender() Validation for Privileged Operations

Low

Closed

BACKGROUND

Privileged operations which only can only be used by specific group of users such as administrators should validate if the Txn.Sender() has the right privileges. Without sufficient validations, the smart contract may allow non-privileged accounts to access privileged methods or operations within the smart contract.

DESCRIPTION

Following instances of insufficient Txn.sender() validations for privileged operations were noted. However, as the transfer is done to a pre-defined fee_sink_address, the impact of this finding is limited to allowing unplanned transfer of platform_fees.

Instance 1

Affected File/Code

- https://github.com/VenueMarkets/contracts-internal/blob/main/contracts/assets/amm.py [on_withdraw_platform_fees]

It was noted that the on_withdraw_platform_fees operation allows anyone to withdraw platform fees from the AMM smart contract.

Based on the code, the sender of the transaction was not validated. Even though the withdrawal address was set to the fee_sink_address, anyone can still call the function and hence, might affect withdrawal schedules.

Instance 2

Affected Code/File

- https://github.com/VenueMarkets/contracts-internal/blob/main/contracts/assets/parimutuel.py [on_withdraw_platform_fees]

It was noted that the on_withdraw_platform_fees operation allows anyone to withdraw platform fees from the parimutuel smart contract.

Based on the code, the transaction sender was not validated. Even though the withdrawal address was set to the fee_sink_address, anyone can still call the function and hence, might affect withdrawal schedules.

IMPACT

Insufficient validation of the Txn.Sender() for privileged operations may allow under-privileged account to access administrative or sensitive operations that is designed to be only accessible to small group of stakeholders or smart contract creators.

RECOMMENDATIONS

Implement sufficient validation to ensure only the correct and specific accounts can access privileged operations.

COMMENT

Reviewed on 17 Nov 2022

Based on the commit f4115bf57c150f30505b35889e1dd968dd7fc78c, this issue is closed. Txn.sender() is now validated against the Global.creator_address().

REFERENCES

PyTeal Documentation - Voting Example
https://pyteal.readthedocs.io/en/stable/examples.html?highlight=Txn.Sender#application-mode

Algorand Developer Portal - Transaction Reference
https://developer.algorand.org/docs/get-details/transactions/transactions/

4.3. Unnecessary Local or Global States and Checks

Observational

Closed

BACKGROUND

Unused local or global states and unnecessary checks may consume opcode costs or increase the amount of Algo that needs to be available in the user's account during application creation and opt-ins.

DESCRIPTION

Following unused local states and global states were identified within the smart contract.
Instance 1
Affected Code/File

- https://github.com/VenueMarkets/contracts-internal/blob/main/contracts/assets/amm.py [self.name, self.primary_outcome, self.secondary_outcome]

Instance 2
Affected Code/File

- https://github.com/VenueMarkets/contracts-internal/blob/main/contracts/assets/parimutuel.py [self.title, self.subtitle]

IMPACT

Unused local and global states increase minimum balance during application creation or opt-ins with no value added to the smart contract's operations or user flows.

RECOMMENDATIONS

Re-evaluate the need for local and global states to see if such is necessary in carrying out operations and transaction group validations through the smart contract's life cycle and user flows. Remove any unnecessary local or global states.

COMMENT

Reviewed on 17 Nov 2022

Based on the commit 5232f021e8157d777391ca41f75c04597f62c32c at https://github.com/VenueMarkets/contracts-internal, this issue is closed.
The documentation was updated mentioning the highlighted states are used for transparency purposes.

REFERENCES

Algorand Developer Portal - Minimum Balance for Smart Contract
https://developer.algorand.org/docs/get-details/parameter_tables/?from_query=minimum%20balance#minimum-balance-for-smart-contract

5. Appendix

5.1. Disclaimer

The material contained in this document is confidential and only for use by the company receiving this information from Vantage Point Security Pte. Ltd. (Vantage Point). The material will be held in the strictest confidence by the recipients and will not be used, in whole or in part, for any purpose other than the purpose for which it is provided without prior written consent by Vantage Point. The recipient assumes responsibility for further distribution of this document. In no event shall Vantage Point be liable to anyone for direct, special, incidental, collateral or consequential damages arising out of the use of this material, to the maximum extent permitted under law.

The security testing team made every effort to cover the systems in the test scope as effectively and completely as possible given the time budget available. There is however no guarantee that all existing vulnerabilities have been discovered. Furthermore, the security assessment applies to a snapshot of the current state at the examination time.

5.2. Risk Rating

All vulnerabilities found by Vantage Point will receive an individual risk rating based on the following four categories.

Critical

A CRITICAL finding requires immediate attention and should be given the highest priority by the business as it will impact business interest critically.

High

A HIGH finding requires immediate attention and should be given higher priority by the business.

Medium

A MEDIUM finding has the potential to present a serious risk to the business.

Low

A LOW finding contradicts security best practices and have minimal impact on the business.

Observational

An OBSERVATIONAL finding relates primarily to non-compliance issues,security best practices or are considered an additional security feature that would increase the security stance of the environment which could be considered in the future version of smart contract.