SharkTeam

No.20220721002C

Smart Contract Audit Report

MatrixMarket

July. 21st, 2022

Copyright

1. Overview

SharkTeam recently received the requirements for MatrixMarket smart contracts audit. In this audit, the SharkTeam security experts communicate with MatrixMarket team to conduct smart contract security audit under controllable operation, so as to avoid any risk in the audit process as far as possible.

Project Overview :

Project Name	MatrixMarket
Description	NFT, GameFi
Language	Cadence
Codebase	Private Repo
MD5	2916AF0CB2777EFE3373C1A467808DD6

Audit method :

SharkTeam security experts conducted a detailed manual audit of the smart contracts line-by-line. From the four dimensions of high-level language, virtual machine, blockchain, and business logic, much more audit items of smart contracts have been comprehensively audited. In particular, the permissions of resources created by Cadence transactions and saved in the account are strictly audited.

Audit Scope :

Contract Files	MD5
MatrixMarket.cdc	5901492672E3131836C51081FD8116B3

MatrixMarketOpenOffer.cdc	00ACEB10237BF02DCB5369A456D1E518

Audit results :

The MatrixMarket smart contract audit results: **Pass**.

2. Findings

2.1 Summary

Vulnerability list :

ID	Item	Severity	Category	Status
1	Invalid-Assertion	Info	Language	Unresolved
2	Event-Emit-Condition-error	Info	Business	Unresolved

2.2 Detailed Results

2.2.1 Invalid-Assertion [Info]

Description:

Location: MatrixMarketOpenOffer.cdc#113, 116

```
111    for cut in cuts {
112        assert(cut.receiver.check(), message: "invalid cut receiver")
113        price = price - cut.amount
114        cutsInfo[cut.receiver.address] = cut.amount
115    }
116    assert(price > 0.0, message: "price must be > 0")
```

Description: in the for loop, when price – cut.amount overflows, the run-time will report an error, and the execution of the contract will be terminated, which makes the assert of price invalid.

Recommendation:

Put the assert check of price in the loop, as follows:

```
111    for cut in cuts {
112        assert(cut.receiver.check(), message: "invalid cut receiver")
113        assert(price > cut.amount, message: "price must be > 0")
114        price = price - cut.amount
115        cutsInfo[cut.receiver.address] = cut.amount
116    }
117    // assert(price > 0.0, message: "price must be > 0")
```

2.2.2 Event-Emit-Condition-error [Info]

Description:

Location: MatrixMarketOpenOffer.cdc#176

```
175    destroy() {
176        if !self.details.purchased {
177            emit OfferCompleted(
178                bidId: self.details.bidId,
179                purchased: self.details.purchased,
180            )
181        }
182    }
```

Description: There exists a logic error that the OfferCompleted event is emited when self.details.purchased==false instead of true.

Recommendation:

Change the if condition to self.details.purchased==true, as follows:

```
176    destroy() {
177        if self.details.purchased {
178            emit OfferCompleted(
179                bidId: self.details.bidId,
180                purchased: self.details.purchased,
181            )
182        }
183    }
```

Appendix A: Vulnerability Severity Classification

The nature of vulnerabilities is unintentional and unexpected security flaws or risks, which can be divided into four threat levels: High, Medium, Low and Info. The classification is mainly based on the impact, likelihood of utilization and other factors.

The impact is defined mainly according to the three dimensions of confidentiality, integrity and availability;

The likelihood of utilization is defined mainly according to three dimensions: attack vector, attack complexity and authentication.

Impact / Likelihood	critical	high	medium	low
low	High	High	Medium	Low
medium	High	Medium	Low	Low
high	Medium	Low	Low	Info
Ex-high	Low	Low	Info	Info

Disclaimer

SharkTeam has tried the best to ensure the accuracy and reliability of the content when writing this report, but SharkTeam will not be responsible for the loss and damage caused by the omission, inaccuracy or error in this report. The safety audit analysis and other contents of this report are based on the materials provided by the project team. This audit only focuses on the audit items provided in this report, and other unknown security vulnerabilities are not within the scope of this audit. SharkTeam cannot determine the security status of facts that appear or exist after the report. SharkTeam is not responsible for the background or other circumstances of the project.

The content, services and any resources involved in this report cannot be used as the basis for any form of investment, taxation, law, and supervision, and there is no relevant responsibility.

About SharkTeam

SharkTeam focus on smart contract security, composed of members with many years of practical experience in front-line cyber security and blockchain. We are proficient in the underlying principles of blockchain and smart contract, and has perfect capabilities in blockchain vulnerability mining and smart contract audit. We can provide comprehensive threat modeling, smart contract audit and emergency response services, SharkTeam has helped several well-known blockchain projects find and fix security vulnerabilities, and is committed to protecting the security of users' digital assets and privacy.

SharkTeam

In Math，We Trust !


https://t.me/sharkteamorg


https://twitter.com/sharkteamorg