verichains

SECURITY AUDIT OF

ORAKL SMART CONTRACTS

PAYMENT UPDATE

Public Report

Sep 22, 2023

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.
ERC20	ERC20 (BEP20 in Binance Smart Chain or *x*RP20 in other chains) tokens are blockchain-based assets that have value and can be sent and received. The primary difference with the primary coin is that instead of running on their own blockchain, ERC20 tokens are issued on a network that supports smart contracts such as Ethereum or Binance Smart Chain.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Sep 22, 2023. We would like to thank the Orakl for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the Orakl Smart Contracts Payment Update. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified no vulnerable issue in the contract code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Orakl Smart Contracts Payment Update

Orakl Network is a decentralized oracle network that allows smart contracts to securely access off-chain data and other resources.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the Orakl Smart Contracts Payment Update.

It was conducted on commit `c7149c9ab7a5fea0ce1429eb0ea5120013968f1c` from git repository *https://github.com/Bisonai/orakl/*.

1.3. Audit methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that were considered during the audit of the smart contract:

- Integer Overflow and Underflow
- Timestamp Dependence
- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy
- Explicit visibility of functions state variables (external, internal, private and public)
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure smart contract. However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

2. AUDIT RESULT

2.1. Overview

The Orakl Smart Contracts Payment Update was written in `Solidity` language, with the required version to be `^0.8.16`. The source code was written based on OpenZeppelin's library.

2.1.1. VRF contracts

A Verifiable Random Function (VRF) is a cryptographic function that generates a random value, or output, based on some input data (called the "seed"). Importantly, the VRF output is verifiable, meaning that anyone who has access to the VRF output and the seed can verify that the output was generated correctly.

In the context of the blockchain, VRFs can be used to provide a source of randomness that is unpredictable and unbiased. This can be useful in various decentralized applications (dApps) that require randomness as a key component, such as in randomized auctions or as part of a decentralized games.

Orakl Network VRF allows smart contracts to use VRF to generate verifiably random values, which can be used in various dApps that require randomness. Orakl Network VRF can be used with two different payment approaches:

- Prepayment
- Direct Payment

Prepayment requires user to create an account, deposit tokens and assign consumer before being able to request for VRF. It is more suitable for users that know that they will use VRF often and possibly from multiple smart contracts.

Direct Payment allows user to pay directly for VRF without any extra prerequisites. This approach is great for infrequent use, or for users that do not want to hassle with Prepayment settings and want to use VRF as soon as possible.

2.1.2. Request-Response contracts

The Orakl Network Request-Response serves as a solution to cover a wide range of use cases. While it may not be possible to bring every data feed directly to the blockchain, the Request-Response allows users to specify within their smart contracts the specific data they require and how they should be processed before they are received on-chain. This feature returns data in Single Word Response format, providing users with greater flexibility and control over their data, and allowing them to access a wide range of external data sources.

Orakl Network Request-Response can be used with two different payment approaches like the VRF features:

- Prepayment
- Direct Payment

2.1.3. Oracle Data Feed contracts

The Orakl Network Data Feed is a secure, reliable, and decentralized source of off-chain data accessible to smart contracts on-chain. The data feed is updated at predefined time intervals, as well as if the data value deviates more than a predefined threshold, to ensure that the data remains accurate and up-to-date. Data feeds can be used in many different on-chain protocols:

- Lending and borrowing
- Mirrored assets
- Stablecoins
- Asset management
- Options and futures
- and many more!

2.1.4. Prepayment contract

The Prepayment Contract is a versatile and dynamic smart contract that offers users a range of account types, each tailored to meet specific needs and preferences. In this contract, the **permanent account** is updated to include four distinct account types: Fiat_subscription, Klay_subscription, Klay_discount, and Klay_regular. Each of these account types serves a unique purpose and provides various benefits to users.

Klay_regular Account:

- Users can freely create Klay_regular accounts.
- For each request made from a Klay_regular account, a specified amount is charged.
- There is no limit on the number of requests that can be made per period, providing users with flexibility in their usage.

Klay_discount Account:

- Klay_discount accounts are created by the owner of the contract.
- For each request made from a Klay_discount account, a discount is applied following the fee ratio set by the contract owner. This allows users to benefit from reduced fees for their transactions.
- There is no limit on the number of requests that can be made per period, providing users with flexibility in their usage.

Klay_subscription Account:

- Klay_subscription accounts are created by the contract owner with specific parameters, including a subscription price and a request period count (reqPeriodCount).
- Users with Klay_subscription accounts are granted a limited number of requests equal to the reqPeriodCount for each subscription period.
- The fee is charged only for the first request made during the subscription period, and the fee is based on the subscription price set by the contract owner.

Fiat_subscription Account:

- Similar to Klay_subscription accounts, Fiat_subscription accounts are also created by the contract owner.
- These accounts have a request period count (reqPeriodCount) that limits the number of requests users can make during a subscription period.
- Unlike Klay_subscription accounts, fees are not charged at the time of the request. Instead, users are required to pay the subscription fee at a later time, providing a deferred payment option.

2.2. Findings

During the audit process, the audit team found no vulnerability issues in the given version of Orakl Smart Contracts Payment Update.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Sep 22, 2023*	Public Report	Verichains Lab

Table 2. Report versions history