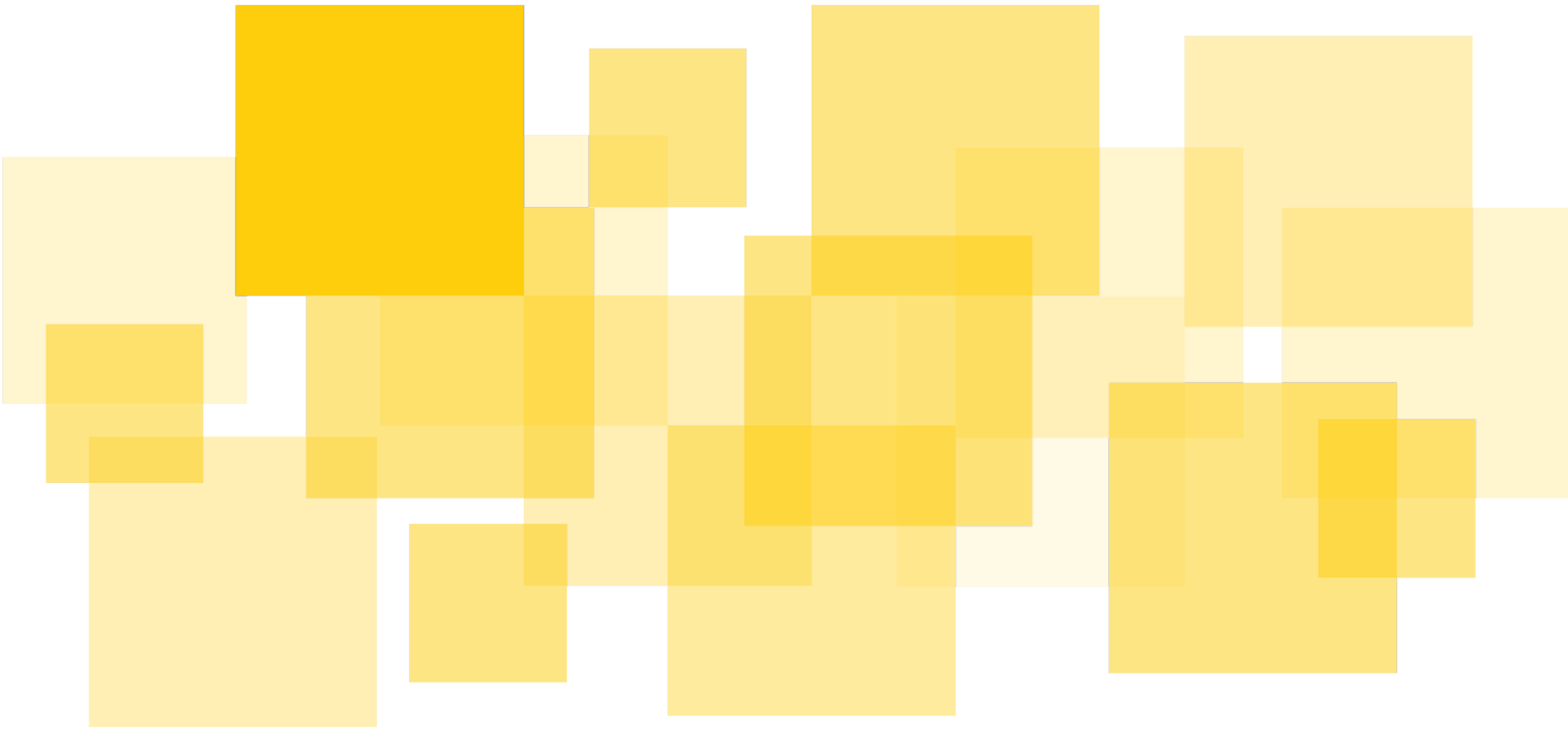
Security Audit Report

World Mobile Ownership Contract

December 29th, 2022



Prepared for WorldMobile, Inc by

runtime verification

Table of Contents

Summary

Runtime Verification, Inc. acted as security code auditor on the Ownership smart contract of World Mobile Group. The review was conducted from September 26 to October 10, 2022. The follow-up code audit is from 19 October to 26 October where we reviewed the fixes for onchain code.

World Mobile Group engaged Runtime Verification in checking the security of their ownership contract. The ownership contract manages the Non-Fungible Tokens and Fungible Tokens (WorldMobileToken) which are used by the Earth Node operators to operate the nodes. There are two roles in this contract, the Admin and NFT Owners (representing the node operators). The NFTs and WorldMobileTokens are minted and distributed to the Admin only. The owner acquired WorldMobileTokens through an independent process which is not part of this contract. Admin uses a whitelist to distribute a certain amount of NFTs to the owners by locking the NFTs in the contract with designated owners. The owners can use their WorldMobileTokens to retrieve the NFTs designated for them and then use the NFT to run the Earth Node. After use, owners can release the NFTs back to the contract in exchange of the locked WorldMobileTokens. Admins is very powerful in the sense that he can do anything except spending the locked WorldMobileTokens which do not belong to him. The retrieved NFTs are tradable and can be used to claim the locked WorldMobileTokens.

The nature of EUTXO models allows anyone to create an EUTXO to be locked by an onchain contract. The Admin has the power to claim any non-conforming EUTXOs. Thus, we **strongly** recommend owners or any other users try not to create such EUTXOs and only transact through the official website. Otherwise you may encounter unexpected losses of funds which cannot be reclaimed by any party.

Introduction

Runtime Verification engaged in the code audit for the onchain Ownership contract.

Scope

The code audit phase focuses on reviewing the onchain contract of one code repository. Any other repos or components such as frontend and backend repositories are ***not*** part of this engagement. The code review is conducted on *source code* of the commit hash

c26a0890ee0cf8107fa4ddac5404535d8e2e4569 of the contracts repo. The following source code files in this commit hash are included in the review:

- src/Contract/Ownership/OnChain/Validator.hs

The review is limited to only the above source files which are the onchain code of the contracts. Any other artifacts such as testing or deployment scripts in the repository are not part of this review.

During the main audit, we found one issue that can cause permanent loss of WorldMobileTokens of owners who do not transact through the official UI. There is no easy fix for this issue and only irrational owners can do this and result in the permanent loss of his own WorldMobileTokens. Thus there is no code change of the on-chain contract required. The other minor issues found in the onchain code have been fixed in the commit hash 786d204bf6aeb5fe9a8215556b1444d9e052c11d, and the patch has been verified by us.

Protocol Configurations

The onchain protocol takes a configuration for the smart contracts and different configurations would result in different onchain contract addresses and hashes. By the end of this audit, the team has not decided their concrete configurations. Thus Runtime Verification cannot independently verify the hash and address of the onchain plutus code compiled from the onchain source code given in the above commit. The configuration contains the following parameters:

1. NFT policy id: b97859c71e4e73af3ae83c30a3172c434c43041f6ff19c297fb76094.
2. Fungible token (WorldMobileToken) asset class with the policy id and the token name: 1d7f33bd23d85e1a25d87d86fac4f199c3197a2f7afeb662a0f34e1e.776f726c646d6f62696c65746f6b656e.
3. The number of the parameterized fungible tokens used to release a single NFT locked by the contract: 100,000.000000 (with 6 decimals).
4. The pubkey hash of the Admin: addr1v882upquzrcuuf6zupd8m58t9a40lag6qnqkdrr2kq5xsjsdysv8t.

Thus Runtime Verification independently verifies the hash and address of the onchain plutus code compiled from the onchain source code given in the above commit. The onchain mainnet address for the ownership contract is addr1w82u502esm0zmv77t0csd6jrgr6wupy5zr7pdwdczpyerpgf6r666.

Assumptions

All the code reviews are based on the following assumptions and trust model:

1. We assume the offchain code which interacts with the onchain contract is unsafe.

2. This code repository is based on the V1 of Plutus API. We also assume the correctness of the Plutus platform. In particular, we assume that the Plutus Core compiler works correctly by compiling the Haskell code to the Plutus Core code.
3. The NFT minting policy shall ensure that all minted NFTs are unique
4. The NFT minting policy shall not allow minting any more NFTs (with the same minting policy but with different token names).
5. The number of parameterized fungible tokens to release a single NFT shall be greater than 1.
6. The policy ids of the parameterized NFT and fungible tokens shall not be the same.

Methodology

We conduct the code review manually. Although manual code review cannot guarantee to find all possible security vulnerabilities as mentioned in Disclaimer, we have followed the following approaches to make our review as thorough as possible. First, we rigorously reasoned about the business logic of the contracts, validating security-critical properties to ensure the absence of loopholes in the business logic. Second, we carefully modeled the design in pseudo code so that we can find details which may be missing from the specification. Finally, we modeled the dependency (token dependency for transactions and token minting) among the contracts; tried out all possible combinations of eUTXOs to construct desired and undesired transactions; checked the following categories of vulnerabilities for each possible transaction:

1. Token related issues. Tokens are assets and are the means to share data (together with the datum) and to achieve access control in Plutus contracts. So the correctness of minting and usage of the tokens and their datum is paramount for the correct functioning of Plutus contracts.
2. Plutus specific implementation issues. There are semantic gaps between the source code language (Haskell) and the onchain assembly code (Plutus Core). It is easy for developers to ignore issues that come inherently in the Plutus platform.

We considered the following attack surfaces for token related issues:

1. Fake token attacks
2. Unsound minting policies
3. Side minting attacks
4. Token uniqueness attacks
5. Violation of single transaction execution
6. Contract clone attacks
7. Deadly transaction repetition

All the issues have been reported to and are either acknowledged or addressed by World Mobile Group.

Severity Classification

The following bug severity classification system is used in the report:

- P1: Highest priority; loss of funds, deadlock of funds, hijack of protocol, arbitrary minting of tokens, etc.
- P2: Severe disruption, DDOS, etc. No funds are technically lost, but makes the protocol unusable for long periods of time
- P3: Inconvenient, or non-intuitive behavior of the protocol; loss of funds through users fault, but where the mistake is very easy to make; etc.
- P4: Code organization, performance, clarity; no major / logical impact on the protocol, but technically changes the behavior
- P5: Purely documentation / cosmetic, like wording in comments or error messages, etc.

Disclaimer

List of Findings

The following list of findings consists of issues identified by the auditor from Runtime Verification (A1-A3).

A1. WorldMobileTokens of malicious owners could be locked permanently by the contract

The nature of EUTXOs allows anyone to create a valid EUTXO locked by the contract. In this case, an owner of the parameterized fungible tokens can create an EUTXO whose datum is `TokensLockedWith *non_existing_nftid*` and with value containing the exact amount of fungible tokens and extra Ada which is required by any UTXOs contains non-ada Values.

Because no one owns the NFT whose token name is `*non_existing_nftid*` and thus no one (including the Admin) can consume this EUTXO and reclaim the fungible tokens.

Severity: P3

Recommendation:

We strongly recommend owners of the parameterized fungible tokens not to do such operations.

Status: Not to be fixed.

A2. Multiple of the required amount of fungible tokens can be locked with the same NFT.

Description

The design of the contract requires the Admin to consume any unconforming UTXOs locked by the contract. However, an NFT owner can send the required amount of WorldMobileTokens to the contract to create a valid EUTXO which locks the sent WorldMobileTokens with a datum *TokensLockedWith his_nftid* and he can do so multiple times as long as he owns enough FTs. These EUTXOs can be spent by him with a single transaction in which he deposits the NFT back to the contract and reclaims all the WorldMobileTokens locked by these EUTXOs according to the code logic.

In this case there is no loss of funds, but if the owner of the NFT loses the NFT and then he may lose all the WorldMobileTokens locked by the contract.

Severity: P4

Recommendation:

There is no good fix in the onchain contract for this issue. So we strongly recommend WorldMobileToken owners shall never create such EUTXOs manually without going through the official UI.

Status: Not to be fixed.

A3. Unused function definition found.

The onchain code defines an unused function `filterValueByDataum.`

Severity: P5

Recommendation:

Remove the unused function definition and remove the imported functions `snd` and `filter` which are used in its function definition.

Status: Fixed in commit hash 786d204bf6aeb5fe9a8215556b1444d9e052c11d.

A4. Typo in function name.

The onchain code defines a function with the name `findScriptOutputValueContainingTockenLockWith`. In which the `Tocken` shall be `Token` .

Severity: P5

Recommendation:

Rename the function to `findScriptOutputValueContainingTokenLockWith`.

Status: Fixed in commit hash 786d204bf6aeb5fe9a8215556b1444d9e052c11d.