verichains

SECURITY AUDIT OF

STAY IN DESTINY WORLD SMART CONTRACTS

Public Report

Dec 10, 2021

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 prepared by Verichains Lab on Dec 10, 2021. We would like to thank the Stay In Destiny World 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 Stay In Destiny World Smart Contracts. 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 some vulnerable issues in the contract code. Stay In Destiny World team has resolved and updated all the recommendations.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Stay In Destiny World Smart Contracts

- Stay In Destiny World is built on BSC platform, featuring Survival & Rare NFT-Collectibles concept with the ambition of an evolving gaming community world & sustainable game economic.
- A community - driven game that appreciates & rewards all type of players for their contributions from free users to serious crypto investors.
- Strong native blockchain & cryptocurrency development team with a subscription list of over 6,000 ready players from our loyal community and our well-know community partner.
- Stay In Destiny World is a mix of easy Click to Play & evolving Action role-playing (Action RPG) gameplay that does not require player's time or extensive gaming experience to participate.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the token and time locked wallet of the Stay In Destiny World Smart Contracts. It was conducted on the source code provided by the Stay In Destiny World team.

It was conducted on commit b31010628ff61314bc706ef952dd0693a0cedc92 from git repository *https://github.com/siwmainproject/siw-contracts*.

1.2.1. SIW Token contracts

The following files were made available in the course of the review:

SHA256 SUM	FILE
b41df713d2b73ba7d8bda1651cff5a69491a4003ec346e979cc77b5df596e258	SIWToken.sol
d889de0fb6386a889387ebbf4d3f41f75f6c96d1d520c38f2cb8193e54c1de1b	TokenExtention.sol

1.2.2. Time Lock Reward contracts

The following files were made available in the course of the review:

SHA256 SUM	FILE
5cad3aca3343d6c6e29fd424a346cea0fc9beb94abc843da51556f674b5f14f2	TimeLockedWallet.sol
389362b7b011b6f4ffda178117f3dcaa2aaaf9fb8ef925c29075c9ee7ad4efa8	TimeLockedWalletFactory.sol

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 was 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.

SEVERITY LEVEL	DESCRIPTION
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

2.1.1. SIW Token contracts

This table lists some properties of the audited Stay In Destiny World Smart Contracts (as of the report writing time).

PROPERTY	VALUE
Name	SIW Token
Symbol	SIW
Decimals	18
Max Supply	250,000,000 ($x10^{18}$) Note: the number of decimals is 18, so the total representation token will be 250,000,000 or 250 million.

Table 2. The Stay In Destiny World Smart Contracts properties

SIWToken contract extends ERC20, Pausable and TokenExtension contracts. With Ownable, by default, Token Owner is contract deployer but he can transfer ownership to another address at any time. He can pause/unpause contract using Pausable contract, user can only transfer tokens when contract is not paused.

TokenExtension contract provides bot protection integration, blacklist and tax on selling for the SIW token. The owner can add/remove addresses to/from blacklist, addresses in blacklist can not transfer SIW tokens. He can then burn tokens in blacklisted addresses and mint it back to the treasury pool. This feature will be used in case hacker attack and steal WIS tokens.

2.1.2. Time Lock Reward contracts

TimeLockedWallet is the contract used for locking tokens of owner and holding tokens for 3rd party vesting. When it is using to lock tokens of owner, isEnableTransfer must be set to false and owner can use withdraw for vesting token as schedule. When it is used to hold tokens of the 3rd party, isEnableTransfer is set to false and the owner will call transfer to send token to 3rd manually, there is no locking in this case.

TimeLockedWalletFactory is the contract used for manage a list of TimeLockedWallet address.

2.2. Findings

This section contains a detailed analysis of all the vulnerabilities that were discovered by the audit team during the audit process.

Stay In Destiny World fixed the code according to Verichain's draft report in commit 4be337883b2bf5213fb57c7fc065b8a73abe3a58.

2.2.1. TimeLockedWallet.sol - Possible fund lost CRITICAL

This contract enables native token (ETH, BNB...) receiving:

```
receive() payable external  {
    emit Received(msg.sender, msg.value);
}
```

But there's no method to recover/transfer these funds.

RECOMMENDATION

Disable receive function or add logic for fund transferring.

UPDATES

- *Dec 10, 2021*: This recommendation has been acknowledged and fixed by the Stay In Destiny World team.

2.2.2. TimeLockedWallet.sol - meaningless locking HIGH

This locking contract is used to store locking for owner, limited by time:

```
function withdraw() onlyOwner external {
    require(block.timestamp >= nextUnlockDate, "TokenTimelock: current ti…
  me is before unlock time");
    uint256 balance = token.balanceOf(address(this));
    require(balance > 0, "TokenTimelock: balance is 0");
    if(isTGETime && balance > tgeUnlockAmount) {
        balance = tgeUnlockAmount;
        isTGETime = false;
        //first time unlock
        nextUnlockDate += dayAfters * 1 days;
    } else if(balance > monthlyUnlockAmount) {
        balance = monthlyUnlockAmount;
        //monthly unlock after first time
        nextUnlockDate += 30 days;
    }
```

```
    token.transfer(owner, balance);
    emit Withdrew(msg.sender, balance);
}
```

But the amounts, and also token can be modified by owner anytime:

```
function setTGEUnlockAmount(uint256 _tgeUnlockAmount) onlyOwner external {
    tgeUnlockAmount = _tgeUnlockAmount;
}

function setPeriodicUnlockAmount(uint256 _periodicUnlockAmount) onlyOwner…
   external {
    monthlyUnlockAmount = _periodicUnlockAmount;
}

function setToken(IERC20 _token) onlyOwner external {
    token = _token;
}
```

With these, owner can change to, full amount for example, to claim once. That just makes the locking useless.

RECOMMENDATION

Removing setTGEUnlockAmount, setPeriodicUnlockAmount and setToken functions.

UPDATES

- *Dec 10, 2021*: This recommendation has been acknowledged and fixed by the Stay In Destiny World team.

2.3. Additional notes and recommendations

2.3.1. BPContract function INFORMATIVE

Since we do not control the logic of the bpContract, there is no guarantee that bpContract will not contain any security related issues. With the current context, in case the bpContract is compromised, there is not yet a way to exploit the Stay In Destiny World Smart Contracts, but we still note that here as a warning for avoiding any related issue in the future.

By the way, if having any issue, the bpContract function can be easily disabled anytime by the contract owner using the setEnableBP function. In addition, bpContract is only used in a short time in token public sale IDO then the contract owner will disable it forever by the setBPDisabledForever function.

2.3.2. Treasury pool contract INFORMATIVE

Since we do not control the logic of the treasuryPool, there is no guarantee that owner can not withdraw tokens from the pool after destroyBlackMoney from users(hackers) wallet and refundBlackMoney to the pool.

APPENDIX

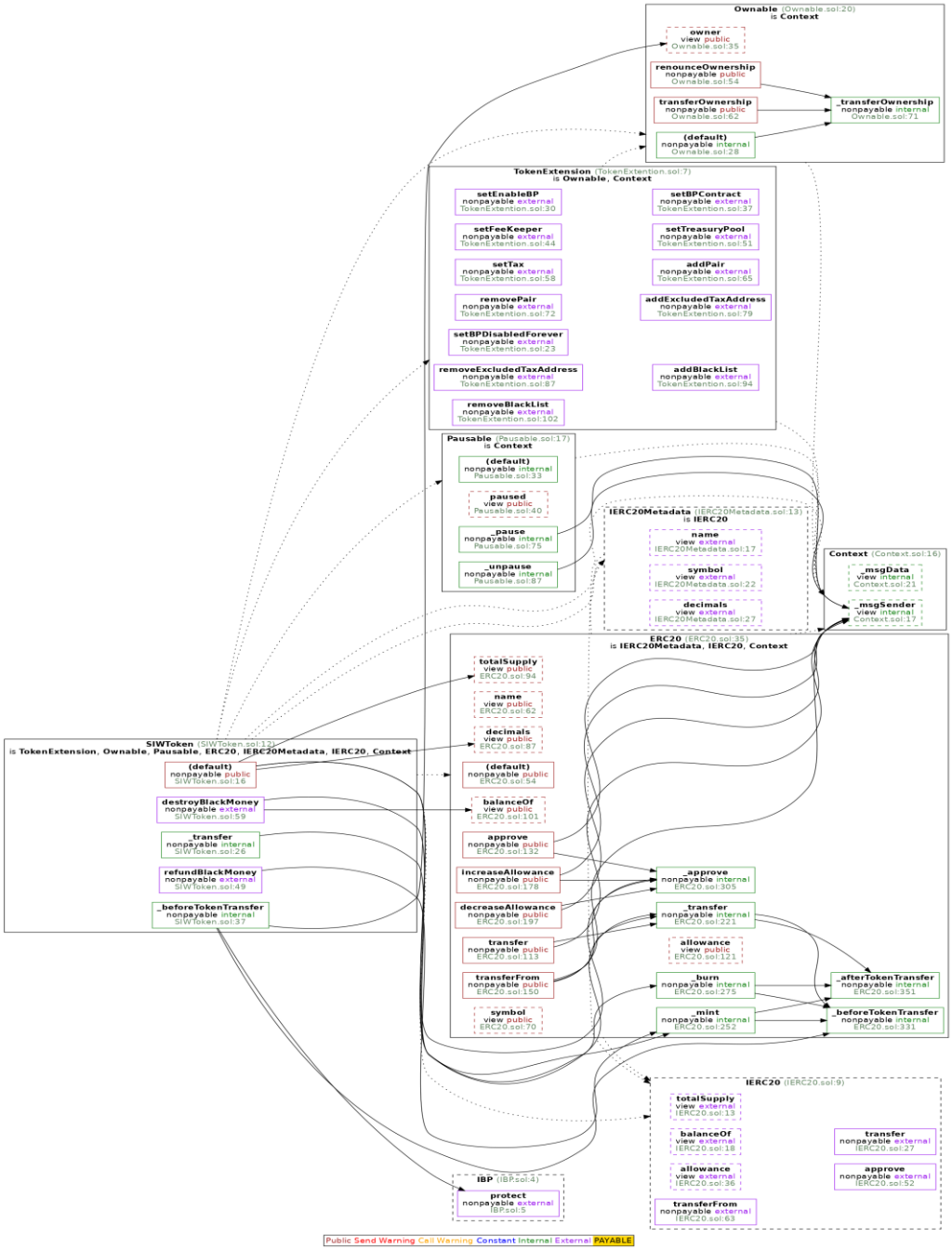

Image 1. Token call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Dec 10, 2021*	Public Report	Verichains Lab

Table 3. Report versions history