SECURITY AUDIT OF

ESCROWDAPP CONTRACT

Public Report

Nov 15, 2022

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 Nov 15, 2022. We would like to thank the EscrowDapp 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 EscrowDapp Contract. 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 issue in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About EscrowDapp Contract

EscrowDapp (*https://www.escrowdapp.com/*) is a blockchain escrow service acts as a neutral third party between buyer and seller to protect both parties from potential fraudulent actions of the other.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of EscrowDapp Contract. It was conducted on the source code provided by the EscrowDapp team.

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

SHA256 Sum	File
`c066d6142efd75c8194b9746362c4f8524b85fe806d2b491d95f009c26e97594`	`EscrowFactory.sol`
`d77c05cd055121b1392f24a3ce7e5f824626a6c2df8ea1300e134f06610cf930`	`Escrow.sol`
`1883a12c6c4ba8eec464580599ad632202b94a1e47906a127adff365bbed8d9f`	`SafeMath.sol`

Table 1. Files audit scope version 1.0

SHA256 Sum	File
`93e3412f8ce8434dc6925d75a52c56fcb147e9288d47a00b04d71c89191d811f`	`EscrowFactory.sol`
`160a61f1191fb22d79c510f0e8ba5adb4d57f2a975ddcab71421ed2fc9a605dd`	`Escrow.sol`

Table 2. Files audit scope version 1.1

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 3. 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 EscrowDapp Contract was written in `Solidity` language, with the required version to be `0.8.15`, and includes 3 contracts: `EscrowFactory.sol`, `Escrow.sol` and `SafeMath.sol`.

The EscrowDapp team uses this product to serve as a third party which allows the buyer escrowing tokens or native currency and transfer to the seller that amount after the condition is met (seller delivered and buyer confirmed).

2.1.1. EscrowFactory contract

EscrowFactory contract is initialized with a list of trusted handlers & trusted token addresses.

Trusted handlers can modify the fee percent, trusted handlers, trusted token addresses, and withdraw the amount of the contract.

Due to the appearance of `receive()` and `fallback()` functions, the contract is able to receive ether from anyone.

2.1.2. Escrow contract

Escrow contract only accepts buyer, seller, and trusted address for interacting.

The buyer deposits tokens for this contract. The buyer can approve the contract release token after the seller has delivered. But on the other hand, if the buyer rejects and the seller rejects, the status of the escrow is changed to dispute.

When the escrow status is ongoing, revised either the buyer or seller may cancel.

Trusted handlers may add more trusted addresses and call the `fund()` function while the status is not completed or canceled.

Due to the appearance of `receive()` and `fallback()` functions, the contract is able to receive ether from anyone.

Note: The extra ETH that was deposited after the contract was initialized cannot be withdrawn by anyone.

2.2. Findings

During the audit process, the audit team found some vulnerability issues in the given files of EscrowDapp Contract.

2.2.1. Escrow.sol - The `fallback()` function causes the permanent freezing of the contract's amount CRITICAL

Position

- L74 Escrow.sol

Description

The attacker can control the amount by sending arbitrary ETHs to the contract. This led to a permanent freeze on the contract's amount. Both the seller and the buyer are unable to withdraw any funds.

Reproduce

Step 1: The attacker sends 0 (or any) ETH to the contract.

Result: Because the amount is zero (or any), neither the buyer nor the seller can withdraw correct funds from the contract.

RECOMMENDATION

Replace `=` with `+=`, fixed code:

```
fallback() external payable {
    escrowDetail.amount += msg.value;
}

receive() external payable {
    escrowDetail.amount += msg.value; //Code Consistency
}
```

UPDATES

- *Nov 22, 2022*: This issue has been acknowledged and fixed by EscrowDapp team.

2.2.2. Escrow.sol - Buyer abuse `cancel()` function perform theft of seller's amount after delivered CRITICAL

Position

- L148-160 Escrow.sol

Description

After the seller has delivered the products or services, the buyer waits for the deadline to end and cancels the payment. Furthermore, the `cancel()` function does not check for revised request when buyer rejected delivered status.

Reproduce

Step 1: The buyer waits for the deadline to end when the seller delivered.

Step 2: The buyer calls `cancel()` function.

Result: The buyer receives the products or services, and the seller does not receive any amount.

RECOMMENDATION

Do not allow cancel when escrow status is delivered. Furthermore, the contract must compare the revised request deadline to the timestamp. Fixed code:

```
function cancel() external {
    require(uint8(escrowDetail.status) < 4 && escrowDetail.status !=
EscrowStatus.Delivered, '___NOT_ELIGIBLE___');
    require(msg.sender == escrowDetail.buyer || msg.sender == escrowDetail.seller,
'___INVALID_BUYER_SELLER___');

    if (
        msg.sender == escrowDetail.buyer &&
        (escrowDetail.status == EscrowStatus.Ongoing || escrowDetail.status ==
EscrowStatus.RequestRevised)
    ) {
        require(block.timestamp >= escrowDetail.deadline && block.timestamp >=
escrowDetail.requestRevisedDeadline, '___NOT_EXPIRED___');
    }

    sendAndStatusUpdate(escrowDetail.buyer, EscrowStatus.Cancelled);
}
```

UPDATES

- *Nov 22, 2022*: This issue has been acknowledged and fixed by EscrowDapp team.

2.2.3. Escrow.sol - Anyone can deposit more ETH into the contract after completed or canceled HIGH

Position

- L73-75 Escrow.sol
- L77 Escrow.sol

Description

Users who wrongly transferred ether to the contract cannot withdraw their funds once the escrow status is completed or canceled.

RECOMMENDATION

When receiving, the status must be not completed or canceled, fixed code:

```
fallback() external payable {
    require(uint8(escrowDetail.status) < 5, '___INVALID_STATUS___');
    escrowDetail.amount += msg.value;
}

receive() external payable {
    require(uint8(escrowDetail.status) < 5, '___INVALID_STATUS___');
    escrowDetail.amount += msg.value;
}
```

UPDATES

- *Nov 22, 2022*: This issue has been acknowledged and fixed by EscrowDapp team.

2.3. Additional notes and recommendations

2.3.1. Escrow.sol - Do not specify a time limit for the duration in constructor INFORMATIVE

The buyer has control over the `duration` time, so he can pass a small number such as 1. This causes too fast a performance of the deal between buyer and seller.

RECOMMENDATION

The `duration` time buyer passed should be at least one day, according to `_deliverRejectDuration` variable at `buyerDeliverReject()` function.

UPDATES

- *Nov 22, 2022*: This issue has been acknowledged and fixed by EscrowDapp team.

2.3.2. Escrow.sol - Trusted addresses have more privilege than necessary at `fund()` function INFORMATIVE

Apart from status completed and canceled, the `fund()` function allows trusted addresses to call. So, trusted addresses can intervene in between buyer and seller transactions in unnecessary cases.

RECOMMENDATION

In the disputing case, the `fund()` function simply allows trusted addresses to be called. . Fixed code:

```
function fund(address payable toFund) external trusted {
    require(toFund == escrowDetail.buyer || toFund == escrowDetail.seller,
'___INVALID_BUYER_SELLER___');
    require(escrowDetail.status == EscrowStatus.Dispute, '___DO_NOT_ALLOW___');
    sendAndStatusUpdate(toFund, EscrowStatus.Complete);
}
```

UPDATES

- *Nov 22, 2022*: This issue has been acknowledged and will not be fixed.

2.3.3. EscrowFactory.sol - Typo `_standartDuration` variable INFORMATIVE

Have a typing mistake at `_standartDuration` variable.

RECOMMENDATION

Fix to `_standardDuration`.

UPDATES

- *Nov 22, 2022*: This issue has been acknowledged and fixed by EscrowDapp team.

2.3.4. Escrow.sol - Unnecessary usage of SafeMath library in Solidity 0.8.0+ INFORMATIVE

All SafeMath usages in the contract are for overflow checking, solidity `0.8.0+` already do that by default, the only usage of SafeMath now is to have a custom revert message which isn't the case in the auditing contracts.

RECOMMENDATION

We suggest changing all methods from `SafeMath` library to normal arithmetic operator in the files that we regarded above.

UPDATES

- *Nov 22, 2022*: This issue has been acknowledged and fixed by EscrowDapp team.

2.3.5. Unnecessary view function INFORMATIVE

Contracts can directly access to public states, so we do not need to attend public functions to view them.

RECOMMENDATION

Remove all of the public functions to view public states.

UPDATES

- *Nov 22, 2022*: This issue has been acknowledged by EscrowDapp team.

2.3.6. Unnecessary payable modifier INFORMATIVE

The functions do not receive any ETH, but they continue to function in the absence of the "payable" modifier.

RECOMMENDATION

Remove all occurrences of the `payable` modifier from functions that do not receive ETH.

UPDATES

- *Nov 22, 2022*: This issue has been acknowledged and fixed by EscrowDapp team.

2.3.7. Escrow.sol - Unnecessary states in contract INFORMATIVE

In `escrow.sol`, the mutable `duration`, `feePercent` and `deliverRejectDuration` states are superfluous. All of them waste gas from the contract.

RECOMMENDATION

Remove `duration` and `deliverRejectDuration` states.

Change `feePercent` to an immutable state.

UPDATES

- *Nov 22, 2022*: This issue has been acknowledged and fixed by EscrowDapp team.

APPENDIX

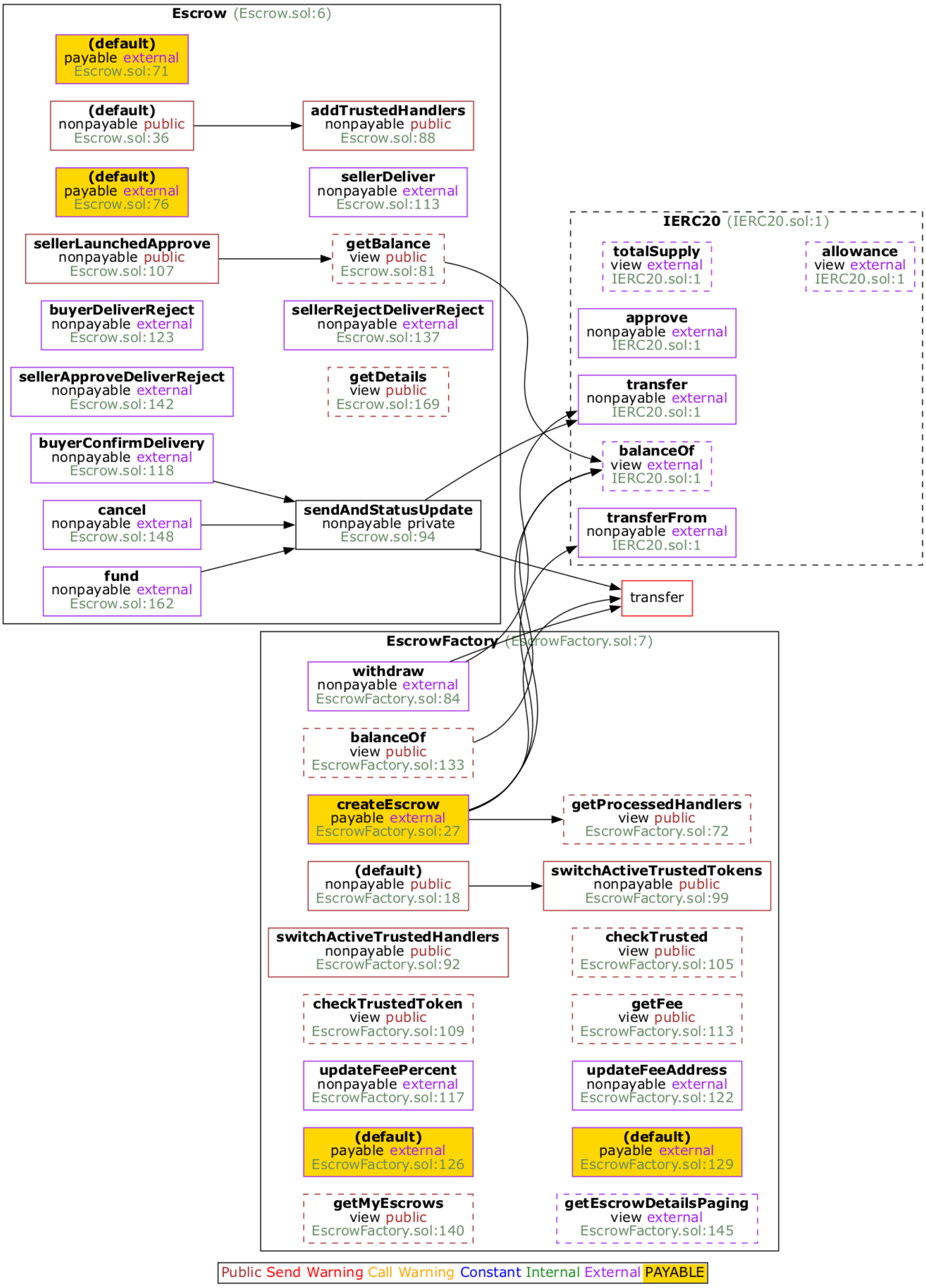
Escrow (Escrow.sol:6)
(default) payable external Escrow.sol:71
(default) nonpayable public Escrow.sol:36
addTrustedHandlers nonpayable public Escrow.sol:88
(default) payable external Escrow.sol:76
sellerDeliver nonpayable external Escrow.sol:113
sellerLaunchedApprove nonpayable public Escrow.sol:107
getBalance view public Escrow.sol:81
buyerDeliverReject nonpayable external Escrow.sol:123
sellerRejectDeliverReject nonpayable external Escrow.sol:137
sellerApproveDeliverReject nonpayable external Escrow.sol:142
getDetails view public Escrow.sol:169
buyerConfirmDelivery nonpayable external Escrow.sol:118
sendAndStatusUpdate nonpayable private Escrow.sol:94
cancel nonpayable external Escrow.sol:148
fund nonpayable external Escrow.sol:162
IERC20 (IERC20.sol:1)
totalSupply view external IERC20.sol:1
allowance view external IERC20.sol:1
approve nonpayable external IERC20.sol:1
transfer nonpayable external IERC20.sol:1
balanceOf view external IERC20.sol:1
transferFrom nonpayable external IERC20.sol:1
transfer
EscrowFactory (EscrowFactory.sol:7)
withdraw nonpayable external EscrowFactory.sol:84
balanceOf view public EscrowFactory.sol:133
createEscrow payable external EscrowFactory.sol:27
getProcessedHandlers view public EscrowFactory.sol:72
(default) nonpayable public EscrowFactory.sol:18
switchActiveTrustedTokens nonpayable public EscrowFactory.sol:99
switchActiveTrustedHandlers nonpayable public EscrowFactory.sol:92
checkTrusted view public EscrowFactory.sol:105
checkTrustedToken view public EscrowFactory.sol:109
getFee view public EscrowFactory.sol:113
updateFeePercent nonpayable external EscrowFactory.sol:117
updateFeeAddress nonpayable external EscrowFactory.sol:122
(default) payable external EscrowFactory.sol:126
(default) payable external EscrowFactory.sol:129
getMyEscrows view public EscrowFactory.sol:140
getEscrowDetailsPaging view external EscrowFactory.sol:145
Public Send Warning Call Warning Constant Internal External PAYABLE

Image 1. Escrow Factory Smart contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Nov 15, 2022*	Private Report	Verichains Lab
1.1	*Nov 22, 2022*	Public Report	Verichains Lab

Table 4. Report versions history