Paytr Protocol Security Review

May 27, 2024 - May 31, 2024

Date: April 26, 2025

Conducted by: **KeySecurity**
gkrastenov, Lead Security Researcher

Table of Contents

1 About KeySecurity

KeySecurity is a innovative Web3 security company that hires top-talented security researchers for your project. We have conducted over 40+ security reviews for various projects, collectively holding over $300,000,000 in TVL. For security audit inquiries, you can reach out to us on Twitter/X or Telegram `@gkrastenov` or check our previous work `here`.

2 About Paytr

The `Paytr Protocol` is a decentralised, permissionless, protocol that facilitates earning money by making early payments. Companies integrating Paytr will benefit from more invoices being paid on time.

3 Disclaimer

Audits are a time, resource, and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

4 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost or a functionality of the protocol is affected.
- **Low** - any kind of unexpected behaviour that's not so critical.

4.2 Likelihood

- **High** - direct attack vector; the cost is relatively low to the amount of funds that can be lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - too many or too unlikely assumptions; provides little or no incentive.

4.3 Actions required by severity level

- **Critical** - client **must** fix the issue.
- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

5 Executive summary

Overview

Project Name	Paytr
Repository	https://github.com/paytr-protocol/contracts
Commit hash	152c0fd315471e3c825d0703b8e9387684aeef9d
Review Commit hash	08e17bfb571df6de5ae24cc7d860107200411b0d
Documentation	https://paytr.gitbook.io/product-docs
Methods	Manual review

Scope

Paytr.sol

Timeline

May 27, 2024	Audit kick-off
May 30, 2024	Preliminary report
May 31, 2024	Mitigation review

Issues Found

Severity	**Count**
High	0
Medium	2
Low	0
Information	2
Total	**4**

6 Findings

6.1 Medium

6.1.1 Escrow pay invoices can be paid out anytime

Severity: *Medium*

Context: Paytr.sol#L176

Description: Escrow pay invoices can be paid out anytime without needing to release escrow by calling the `releaseEscrowPayment` function. Every user can exploit this, rendering escrow useless.

Recommendation: In the `payOutERC20Invoice` function, check if the payout invoice is escrow and if its due date has expired.

Resolution and Client comment: Resolved. The check in the `payOurERC20Invoicefunction` has been modified to:

```
if(paymentERC20.dueDate > block.timestamp || paymentERC20.dueDate == 0 ) revert
    ReferenceNotDue();
```

6.1.2 Payment with _amount = 0 can be created

Severity: *Medium*

Context: Global

Description: It is possible to create a payment with `amount = 0` and `feeAmount > 0`. This payment cannot be paid out because, during the payment process, `paymentERC20.amount` is checked to ensure it is different from 0. The fee amount will be stuck in the contract, and the payment position cannot be deleted.

Recommendation: Add a check for `amount != 0` during the creation of the payment.

Resolution and Client comment: Resolved. The payment creation received an additional check:

```
if(_amount == 0) revert ZeroAmount();
```

6.2 Information

6.2.1 payInvoiceERC20Escrow and payInvoiceERC20 functions can be combined

Severity: *Information*

Context: Global

Description: The `payInvoiceERC20Escrow` and `payInvoiceERC20` functions can be combined because the only difference is the setting of the `dueDate` parameter when the payment is created. It

will be more efficient to have only one function with a type of payment, for example `Standard` and `Escrow`.

Recommendation: Combine both functions into one and create a new `Struct{Standard, Escrow}` which will be used to set the due date for the payment.

Resolution and Client comment: Acknowledged.

6.2.2 Redundant erros and function

Severity: *Information*

Context: Global

Description: In the `Paytr` contract, there are several places with redundant code (errors and functions) that are never used.

1. Remove the following errors: `InvalidFeeAmount()`, `NotAuthorized()`, `DueDateNotZero()`, `InvalidNewDueDate()`, `InvalidFeeAmountMultiplier()`
2. Remove the following function: `getContractBaseAssetBalanc()`

Recommendation: Remove redundant code.

Resolution and Client comment: Resolved.