SALUS SECURITY

JAN 2026

CODE SECURITY ASSESSMENT

PAY THE FLY

Overview

Project Summary

- Name: PayTheFly - Contract
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
  - https://github.com/paythefly/PayTheFlyContract
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	PayTheFly - Contract
Version	v3
Type	Solidity
Dates	Jan 31 2026
Logs	Jan 28 2026; Jan 30 2026; Jan 31 2026

Vulnerability Summary

Total High-Severity issues	1
Total Medium-Severity issues	1
Total Low-Severity issues	0
Total informational issues	3
Total	4

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users’ sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients’ reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users’ sensitive information at risk, would be detrimental to the client’s reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client’s business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Signed payments can be permanently invalidated via pay() serialNo preemption	High	Front-running	Resolved
2	Removing an admin does not revoke their existing confirmations, potentially inflating confirmCount	Medium	Business Logic	Resolved
3	Redundant Code	Informational	Redundancy	Resolved
4	Events are not indexed	Informational	Logging	Acknowledged
5	Gas optimization suggestions	Informational	Gas Optimization	Resolved

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Signed payments can be permanently invalidated via pay() serialNo preemption	
Severity: High	Category: Front-running
Target: - contracts/v1/evm/PayTheFly.sol	

Description

The functions `` `pay()` `` and `` `payWithSign()` `` share the same replay-protection namespace: `` `usedPaymentSerialNos[projectId][serialNo]` ``. The uniqueness key includes only (`` `projectId` ``, `` `serialNo` ``) and does not include `` `token` ``, `` `amount` ``. As a result, an attacker can call the unsigned `` `pay()` `` function with the same `` `projectId` `` and `` `serialNo` `` that a project signer authorized for `` `payWithSign()` ``, causing all future `` `payWithSign()` `` calls for that signed request to revert with payment serial no already used.

A malicious actor can grief and deny service for signed payments at very low cost. In the strongest form, the attacker can front-run a user's `` `payWithSign()` `` transaction by extracting `` `projectId` `` and `` `serialNo` `` from calldata in the public mempool, then submitting a minimal-value `` `pay()` `` that marks the serial number as used.

The victim's `` `payWithSign()` `` subsequently reverts, forcing the project to re-issue a new signature (new `` `serialNo` ``) and causing payment failures, operational overhead, and potentially lost revenue.

contracts/v1/evm/PayTheFly.sol:L286-L450

```
function pay(
    string calldata projectId, address token, uint256 amount, string calldata serialNo
) external payable override nonReentrant whenDepositsNotPaused
projectActive(request.projectId) {
    require(!usedPaymentSerialNos[projectId][serialNo], "PayTheFly: payment serial no
already used");
    …
    usedPaymentSerialNos[projectId][serialNo] = true;
    …
}
function payWithSign(
    PaymentRequest calldata request,
    bytes calldata signature
) external payable override nonReentrant whenDepositsNotPaused
projectActive(request.projectId) {
    …
    require(!usedPaymentSerialNos[projectId][serialNo], "PayTheFly: payment serial no
already used");
    …
}
```

Recommendation

Use a replay key based on `` `(projectId, token, amount, serialNo)` `` instead of `` `(projectId, serialNo)` ``.

Status

This issue has been resolved by the team with commit 4fe944b.

2. Removing an admin does not revoke their existing confirmations, potentially inflating confirmCount	
Severity: Medium	Category: Business Logic
Target: - contracts/PayTheFlyPro.sol	

Description

The `_executeRemoveAdmin()` removes an admin from `_admins` and clears `_isAdmin/_adminIndex`, but it does not clear that admin's existing `_confirmations[proposalId][admin]` nor adjust `Proposal.confirmCount`. As a result, `confirmCount` can become inconsistent with the current admin set and may include confirmations from removed admins.

```
function _executeRemoveAdmin(uint256 proposalId, bytes memory params) internal {
   address admin = abi.decode(params, (address));
   if (!_isAdmin[admin]) revert AdminNotFound();
   if (_admins.length <= _threshold) revert ThresholdTooHigh();
   …
   _admins.pop();
   delete _isAdmin[admin];
   delete _adminIndex[admin];

   emit AdminRemoved(admin, proposalId);
}
```

`executeProposal()` checks `p.confirmCount >= _threshold`. If `confirmCount` remains inflated by confirmations from removed admins, proposals may be executable with fewer confirmations from the current admins than intended.

Recommendation

When removing an admin, revoke their confirmations for pending proposals and decrement `confirmCount`.

Status

This issue has been resolved by the team with commit e974121.

2.3 Informational Findings

3. Redundant Code	
Severity: Informational	Category: Redundancy
Target: - contracts/v1/tvm/PayTheFly.sol	

Description

The `pay()` function (lines 291-384) and `payWithSign()` function (lines 392-485) contain identical code, representing clear code redundancy.

contracts/v1/tvm/PayTheFly.sol:L291-L485

```
function pay(
    PaymentRequest calldata request,
    bytes calldata signature
) external payable override nonReentrant whenDepositsNotPaused
projectActive(request.projectId) {
    ...
}
function payWithSign(
    PaymentRequest calldata request,
    bytes calldata signature
) external payable override nonReentrant whenDepositsNotPaused
projectActive(request.projectId) {
    ...
}
```

Recommendation

Consider removing the redundant code.

Status

This issue has been resolved by the team with commit 4fe944b.

4. Events are not indexed	
Severity: Informational	Category: Logging
Target: - contracts/v1/tvm/interfaces/IPayTheFly.sol	

Description

Events allow capturing the changed parameters so that off-chain tools/interfaces can register such changes that allow users to evaluate them. The emitted events are not indexed, making off-chain scripts such as front-ends of dApps difficult to filter the events efficiently.

In the `IPayTheFly` contract, the `ProjectCreated`, `ProjectUpdated` `Transaction` event is not indexed.

Recommendation

Consider adding the indexed keyword in these events.

Status

This issue has been acknowledged by the team.

5. Gas optimization suggestions	
Severity: Informational	Category: Gas Optimization
Target: - contracts/PayTheFlyPro.sol	

Description

Memory reading saves more gas than storage reading multiple times when the state is not changed. So caching the storage variables in memory and using the memory instead of storage reading is effective. Cache array length outside of the loop can save gas.

contracts/PayTheFlyPro.sol:L159, L526

```
for (uint256 i = 0; i < tokens.length; i++)
for (uint256 i = 0; i < _admins.length && idx < p.confirmCount; i++)
```

Recommendation

Consider using the above suggestions to save gas.

Status

This issue has been resolved by the team with commit e974121.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit d508680:

File	SHA-1 hash
contracts/v1/evm/PayTheFly.sol	b19ebc4067d097c70b283300e5f730352b1d7223
contracts/v1/tvm/ChainIdHelper.so	5ea1166fc4817119744826c1bcd5a9d1810cec74
contracts/v1/tvm/Migrations.sol	66f0f6732002556acf4ceb082035d7255109cd23
contracts/v1/tvm/libraries/TronSafeERC20.sol	a42274700a56538a10fdb688ba6b3ecba23bb6c1
contracts/v1/tvm/PayTheFly.sol	5e001eb6381ecab7423d483872d967396fb1517b
contracts/PayTheFlyPro.sol	22a0d3c8c21c224c4a3160e0c82caba84db67537
contracts/PayTheFlyProFactory.sol	5e3ccd9a64a5057c0d2961b77776525c9fdd6627