tangem × PESSIMISTIC

SECURITY ANALYSIS

by Pessimistic

This report is public

September 26, 2025

ABSTRACT

In this report, we consider the security of smart contracts of Tangem project. Our task is to find and describe security issues in the smart contracts of the platform.

DISCLAIMER

The audit does not give any warranties on the security of the code. A single audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, a security audit is not investment advice.

SUMMARY

In this report, we considered the security of Tangem smart contracts. We described the audit process in the section below.

The audit showed several issues of medium severity: Fee debt is not accumulating (M01), Fee may be unpaid, Unhandled underflow (M03). Also, several low-severity issues were found. Note was commented on by the developers. All the tests passed.

Overall code quality is good.

After the initial audit, the codebase was updated. M01 and L02 issues were removed as false positives. M03 was commented on and moved to notes (Unhandled underflow). Fee may be unpaid issue was partially fixed and commented on. The rest of the low-severity issues and notes were fixed or commented on. One low-severity issue L06 was found and fixed by the developers.

GENERAL RECOMMENDATIONS

We do not have any further recommendations.

PROJECT OVERVIEW

Project description

For the audit, we were provided with Tangem project on a public GitHub repository, commit 2a5c3819189e1f5281984f18c7af03461cacf8a6.

The scope of the audit included:

- **TangemAaveV3YieldModule.sol**;
- **TangemYieldModuleFactory.sol**;
- **TangemYieldProcessor.sol**;
- **YieldModuleLiquidUpgradeable.sol**;
- dependencies.

The documentation for the project included **Tangem_Yield_Module_Contracts_Functional_Requirements.pdf** (sha1sum `a09de4bbd0f6031d027fad8e60d8cd2deee8d91d`).

The list of networks for deployment:

- Ethereum;
- Avalanche;
- Arbitrum;
- Optimism;
- Base;
- Gnosis;
- Scroll;
- BSC;
- ZKsync Era;
- Polygon POS;
- Sonic.

All 73 tests pass successfully. The code coverage is 94.08%.

The total LOC of audited sources is 436.

Codebase update #1

For the recheck, the developers added the code to the new public repository on commit f685229ee1f83e5f95122005d8dd66ce17b3b9f8.

All the 73 tests passed. The code coverage was 93.18%.

One issue of medium severity was removed as false positive. Another medium severity issue was partially fixed and commented on, and the third one was moved to notes. The developers also fixed or provided comments for low-severity issues and notes. Also, they found and fixed the new low-severity issue.

AUDIT PROCESS

We started the audit on September 10 and finished on September 16, 2025.

We inspected the materials provided for the audit. During the work, we stayed in touch with the developers and discussed confusing or suspicious parts of the code.

We manually analyzed all the contracts within the scope of the audit and checked their logic. Among other, we verified the following properties of the contracts:

- Standard Solidity issues;
- Integration with Aave V3;
- Whether fees are correctly accounted for and whether they could be bypassed.

We scanned the project with the following tools:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules;
- Semgrep rules for smart contracts;
- AI-tool Savant.Chat;
- AI-tool AUDITAGENT.

We ran tests and calculated the code coverage.

We combined all the verified issues we found during the manual audit or discovered by automated tools in the private report.

We performed the recheck on September 23-26, 2025. One issue of medium severity was removed as false positive. Another medium severity issue was partially fixed and commented on, and the third one was moved to notes. The developers fixed or commented on low-severity issues. They also discovered and fixed L06.

We also re-ran the tests and recalculated code coverage. Finally, we updated the report.

MANUAL ANALYSIS

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

The audit showed no critical issues.

Medium severity issues

Medium severity issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

M02. Fee may be unpaid (partially fixed, commented)

According to the **YieldModuleLiquidUpgradeable** contract, if the owner withdraws all funds from the yield module and some fees have not been paid due to an error or if the owner revoked approval for yield tokens (during `exitProtocol`), they are recorded in `feeDebts`. If the owner no longer interacts with Aave V3, these `feeDebts` in **YieldModuleLiquidUpgradeable** will remain unpaid indefinitely.

> The issue has been partially fixed in the `YieldModuleLiquidUpgradeable.send` function and commented on by the developers.

> `Comment from the developers:`
> There is no way for a user to avoid fee, the previous possibility is fixed. If the fee won't be paid for some other reason, the fee funds will stay at the module, and can be retrieved by Tangem later. The priority is to make sure our users can use their funds, even if fee payment fails, not our fee.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

L01. AToken zero address check (fixed)

The `YieldModuleLiquidUpgradeable.initYieldToken` function does not verify the **aToken** address returned by `_initProtocolToken`. If a yield token is not supported by Aave V3, the getReserveData returns `address(0)`.

This does not impact security because `pool.supply` in Aave V3 already checks the token validity. But an early check of the **aToken** address can be added to prevent unnecessary gas consumption when the token is unsupported.

> The issue has been fixed and is not present in the latest version of the code.

L03. The effective balance is overestimated (fixed)

The `YieldModuleLiquidUpgradeable.effectiveBalance` funciton can overestimate the actual available balance, as it does not account for `feeDebts`.

It is also important to handle the underflow case when the sum of `_calculateServiceFee` and `feeDebts` exceeds the owner's and protocol's balances.

> The issue has been fixed and is not present in the latest version of the code.

L04. Unindexed parameters (fixed)

Consider declaring parameters of the `YieldModuleDeployed` event as `indexed` to enhance off-chain filtering in the **TangemYieldModuleFactory** contract.

> The issue has been fixed and is not present in the latest version of the code.

L05. Unnecessary modifier (commented)

The `whenNotPaused` modifier does not affect the security of the `TangemYieldModuleFactory.calculateYieldModuleAddress` function because the function is declared as `view` and only pre-calculate the **YieldModule** address.

> `Comment from the developers:`
>
> This modifier is added to avoid unnecessary approvals from the client. If the contract is paused, the implementation can change, leading to the future module address changing. If the client software will do approval as a first step without checking, it will give the approval to a potentially incorrect address, wasting funds for network fee. With the pause, the deployment cannot be completed anyway, so there is no real reason to allow this function usage during pause.

L06. Withdrawing zero-amount from Aave V3 (fixed)

The `YieldModuleLiquidUpgradeable.withdrawAndDeactivate` function reverts if `amountToExit` is equal to 0, since this value is passed to Aave V3 `pool.withdraw`, which reverts on zero amounts. As a result, `yieldTokensData` will not be updated.

> The issue has been fixed and is not present in the latest version of the code.

Notes

N01. Project admins' roles (commented)

In the current implementation, the system depends heavily on the admin role. Thus, there are scenarios that can lead to undesirable consequences for the project and its users, e.g., if admin's private keys become compromised. The description of admins' roles:

- The `IMPLEMENTATION_SETTER_ROLE` can change the implementation in the **TangemYieldModuleFactory** contract;
- **TangemYieldProcessor** contract:
  - The `PAUSER_ROLE` can pause/upause contract and influnce to the operations, including setting service fee and fee receiver addresses, collecting service fee, entering/exiting protocol;
  - The `PROTOCOL_ENTERER_ROLE` can supply to Aave V3 and create or increase owner's position;
  - The `PROTOCOL_EXITER_ROLE` can close owner's position by calling `exitProtocol` and withdrawing funds from Aave V3;
  - The `PROPERTY_SETTER_ROLE` can change the service fee rate fee receiver and the `SERVICE_FEE_COLLECTOR_ROLE` can withdraw all fees to the fee receiver.

We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig.

> According to the developers' comment, they will use multisig to manage the project.

N02. Unhandled underflow (commented)

The `YieldModuleLiquidUpgradeable.withdrawAndDeactivate` may revert due to underflow at line 155 if the `feeDebts` exceeds the protocol balance. This can lock the owner's funds until the `exitProtocol` function is called by the processor or `_enterProtocol` is used to increase the protocol balance.

> `Comment from the developers:`
> If for some reason `feeDebts` are greater than the `protocolBalance` - then all the funds there are a part of our fee. There are no owner funds left, so there is no issue here.

This analysis was performed by Pessimistic:

Daria Korepanova, Senior Security Engineer
Yhtyyar Sahatov, Security Engineer
Irina Vikhareva, Project Manager
Alexander Seleznev, CEO

September 26, 2025