

Tycho Router V3

Security Assessment

Prepared for **Propeller Heads** by **Maximilian Krüger** (snd.github.io)

2026-05-14

Disclaimer

The assessment comprised nine days of threat modeling, manual code auditing, and vulnerability research, plus thirteen days of modeling the system and sweeping the input space.

Further time investment might result in additional findings.

The findings should not be considered an exhaustive list of security issues in the codebase.

Table of Contents

- [Summary](#)
- [Methods](#)
- [Scope](#)
- [User Checklist](#)
- [Executor Checklist](#)
- [Preventive Measures Checklist](#)
- [Findings](#)
 - [1. Compromised Admin Account Can Disable Asset Theft Countermeasures](#)
 - [2. Timelock Duration Based on Block Number Instead of Timestamp](#)
 - [3. Repeated Executor Addition Resets Timelock](#)
 - [4. Redundant Payable Modifier and Explicit Check](#)
 - [5. Deposit Function Is Not Pausable, Exposing Deposits During Attack Mitigation](#)
 - [6. Missing Expiration Block Parameter in ExecutorSet Event](#)
 - [7. Missing Contract Address Validation for Fee Calculator](#)
 - [8. Permit2 Can Be Set to Non-Contract Address](#)
 - [9. Inconsistent Naming for State Variables and Colliding Parameters](#)
 - [10. Low Delay Accepted for Executor Timelock During Deployment](#)
 - [11. Incorrect Cyclic Swap Accounting Enables Asset Theft](#)
 - [12. Zero Input Amount Bypasses Vault Delta Validation Allowing Asset Theft](#)
 - [13. List of Sequential Swaps Allowed to Be Empty](#)
 - [14. Duplicated Single Swap Logic](#)
 - [15. Zero Address Represents Both Missing Address and Native Token](#)
 - [16. Incomplete Signature Input Enables Replay and Solver Balance Theft](#)
 - [17. Input Parsing Ambiguity Allows Divergent Transfer Type in CurveExecutor's getTransferData Function](#)
 - [18. Delegatecall Where Staticcall Would Suffice](#)
 - [19. Fee Calculator Update Lacks Timelock](#)
 - [20. Unnecessary Low-Level Calls](#)

Summary

Propeller Heads engaged Maximilian Krüger to assess the security of the TychoRouter Ethereum smart contract between 2026-02-06 and 2026-05-05.

The assessment comprised nine days of threat modeling, manual code auditing, and vulnerability research, plus thirteen days of modeling the system and sweeping the input space. See [Methods](#) for more details.

The assessment resulted in 20 findings with the following severities:

Severity	Count
High	3
Medium	5
Low	5
Informational	7

The most severe findings are [11](#) and [12](#), which allowed an attacker to steal all of TychoRouter's assets - the worst-case scenario.

Propeller Heads accepted the risk of the informational finding [14](#).

Finding [16](#) was partially fixed. The team communicated that a full fix would result in unacceptably high gas usage.

All other findings were fixed.

Before version 3 (V3), TychoRouter was not supposed to own any assets. Its sole purpose was the execution of complex swap sequences. Ether and ERC-20 tokens owned by TychoRouter were considered lost, and keeping them safe was a low priority.

This has changed in V3. Now TychoRouter also acts as a vault. Users can deposit and withdraw tokens to efficiently use them during swaps, reducing the number of gas-intensive transfers in and out.

Acting as both a router and a vault, V3 is a significantly riskier product than previous versions. An important assumption - that assets owned by the router are not safe - was reversed. Highly flexible transfer functionality, which gives users control over a large input space, coexists with substantial asset balances.

The worst-case scenario involves an attacker stealing assets or allowances held by `TychoRouter`. The assessment focused primarily on preventing this worst-case scenario.

A security assessment cannot guarantee that the assessed system is secure. It is possible that further vulnerabilities remain undetected by this assessment.

`TychoRouter`'s functionality can be extended to support additional protocols by registering executors. Even if it were possible to guarantee that no further vulnerabilities exist, the on-chain addition of a new executor can introduce vulnerabilities. We recommend that developers of new executors follow the [Executor Checklist](#) and that users set up alerts for the `ExecutorSet` event and review new executors within the three-day timelock period.

Users can lose assets by using `TychoRouter` incorrectly. We recommend that users of `TychoRouter` follow the [User Checklist](#).

Methods

The security assessment was primarily performed using manual code auditing and vulnerability research.

An internal document comprehensively describing all secrets involved in the project was created in collaboration with the team. Analysis of the document resulted in several recommendations that were implemented.

Attack trees were constructed for several threat scenarios and used in vulnerability research.

After manual vulnerability research identified the critical and complex finding [11](#), which elevated the severity of the assessment, it became clear that a manual approach alone would be impractical for identifying similar vulnerabilities due to the enormous size of the input space and the number of possible execution paths.

A partial model of the system was built in Rust, and the input space was swept using a parallelized brute-force approach.

The model discovered the critical finding [12](#), confirmed finding [11](#), and identified four additional variants of it.

Modeling in Rust was chosen over fuzzing the EVM bytecode directly due to TychoRouter's relatively small codebase, the vast input space, and the overhead associated with EVM execution. The model can execute approximately 3 million times per second on commodity hardware. Performance scales nearly linearly with the number of CPU cores.

The model was delivered via a [pull request](#) and can be found [here](#).

For more information on the model, refer to its comprehensive documentation.

Only executors that give callers control over the target contract are modeled. These are easier to model and present a broader attack surface.

We recommend that the team keep the model up to date, sweep the input space regularly, and model the remaining executors.

Scope

The assessment focused on various commits of the [tycho-execution repository](#).

Towards the end of the assessment, the tycho-execution repository was [copied](#) into the `crates/tycho-execution` directory of the [tycho-indexer repository](#).

A linked commit is provided with each finding.

Only the following files, which are all in the `foundry/src` directory, were in scope:

- Dispatcher
- FeeCalculator
- RestrictTransferFrom (later renamed to TransferManager)
- TychoRouter
- Vault
- executors
 - BalancerV2Executor
 - BalancerV3Executor
 - BebopExecutor
 - CurveExecutor
 - ERC4626Executor
 - EkuboExecutor
 - FluidV1Executor
 - HashflowExecutor
 - MaverickV2Executor
 - RocketpoolExecutor
 - SlipstreamsExecutor
 - UniswapV2Executor
 - UniswapV3Executor
 - UniswapV4Executor
 - WethExecutor

User Checklist

Follow this checklist when using `TychoRouter` . It covers essential security requirements but is not exhaustive on its own.

- **Always** set the `minAmountOut` parameter of `TychoRouter` 's `swap...` functions to the minimum acceptable output amount.
 - For example, if you expect to receive **1000** USDC and accept 5% slippage, set `minAmountOut` to **$950 * 10^{**6}$** .
 - If you set `minAmountOut` to **1** , it's possible that you'll receive **1** due to faulty swap sequences or slippage.
- Set up alerts for the `ExecutorSet` event and [review](#) every new executor before its three-day timelock expires. Executor code runs within `TychoRouter` 's context. A malicious executor can transfer `TychoRouter` 's assets and manipulate user balances.
- Verify the price data used for `minAmountOut` against at least one other independent price source.
 - Incorrect price data may set `minAmountOut` too low, resulting in significant losses.
- Never approve infinite allowances, including those for Permit2.
- Set Permit2 allowance and deadline as low as is practical.

Executor Checklist

Follow this checklist when developing a new executor or when reviewing a new executor before its three-day timelock expires. It covers essential security requirements but is not exhaustive on its own.

`TychoRouter` interacts with protocols like Uniswap or Curve through executor contracts, and can support new protocols without redeployment.

`TychoRouter` calls executors via `delegatecall`. Without `delegatecall`, assets would need to be transferred between `TychoRouter` and the executor contracts. That would be costly.

Delegatecall is dangerous. Executor code runs within `TychoRouter`'s context, where it can freely transfer the router's assets and write to the router's storage, including users' vault balances.

- The executor must never call `ERC20.transfer`, `ERC20.transferFrom`, or `Permit2.transferFrom`.
 - Instead, it must communicate transfer information through the `getTransferData` and `getCallbackTransferData` methods.
 - This allows `TychoRouter` to perform transfers in a manner that provides some safeguards.
 - The executor must never assign to a state variable or perform other operations that write to `TychoRouter`'s storage.
-
- The executor should not execute `delegatecall`.
 - If `delegatecall` is unavoidable, ensure that the caller cannot control the `delegatecall`'s target, as that would enable arbitrary code execution.
 - The executor should verify that callbacks, which are handled via the `handleCallback` function, originate from the correct protocol by checking `msg.sender`.
 - The `data` argument of an executor's `handleCallback` function contains the raw ABI-encoded call data, which must be decoded manually.
 - The return data of an executor's `handleCallback` function contains the raw ABI-encoded return data, which must be encoded manually.

Preventive Measures Checklist

1. Avoid omnipotent, admin, or root accounts that can grant or revoke all permissions.
2. Delegatecall is one of Solidity's most dangerous operations. Never use delegatecall without a strong justification. Document the justification.
3. Never use delegatecall to call functions that have no side effects (`view` or `pure`). Staticcall suffices and is far safer.
4. Timelock all authorized operations that change the system's behavior. This gives the team and users time to detect and react to malicious changes.
5. For each public function, decide carefully whether it should be pausable.
6. To reduce attack surface, compute parameters from context if possible instead of relying on user input.
7. Hardcode or validate constructor arguments that are essential to security, such as fixed addresses (Permit2, WETH) and timelock durations.
8. Avoid giving a type's zero value any meaning other than zero. For example, avoid having `address(0)` mean the native token and avoid having `0` mean that an error occurred.
9. Ensure functions are idempotent or explicitly define behavior for repeated calls with identical arguments.
10. Validate that all arguments intended as contracts contain deployed code.
11. For every user input array, check whether an empty array is acceptable.
12. Include all relevant parameters in security-critical events to prevent blind spots during monitoring.
13. Always use `block.timestamp` for time-based logic. Avoid using `block.number` as a proxy for elapsed time.
14. Avoid nontrivial duplicate logic. Duplicates create the risk that fixes are not applied to all copies. Duplicates multiply testing and review effort of the logic in question.
15. Prefer high-level calls (`Interface(address).functionName(param1, param2)`) over low-level calls (`address.call(raw_calldata)`). High-level calls require less code and provide automatic safeguards.
16. To enforce a zero message value, omit the `payable` modifier instead of using an explicit check like `msg.value > 0`.
17. Use leading underscore (`_foo`) for non-public entities.
18. Use trailing underscore (`foo_`) for function parameters that clash with existing names.

Findings

1. Compromised Admin Account Can Disable Asset Theft Countermeasures

Severity: Medium (Compromised `DEFAULT_ADMIN_ROLE` can steal all assets remaining on the router after executor timelock expires)

Difficulty: High (Requires compromised `DEFAULT_ADMIN_ROLE` multi-sig)

Fix Status: Fixed

CWE: [269 - Improper Privilege Management](#)

Git Commit: [ebf1d0e605b795775472bb793f43f958344f219e](#)

Exploit Scenario

1. Attacker Bob gains access to an account with `DEFAULT_ADMIN_ROLE`.
2. Bob grants `DEFAULT_ADMIN_ROLE` to an account ACC controlled solely by Bob.
3. Bob gives ACC the `EXECUTOR_SETTER_ROLE` and sets a new executor EXE whose code Bob controls.
4. Bob revokes the `PAUSER_ROLE` from all accounts. This eliminates the team's ability to pause executor code.
5. Bob revokes the `EXECUTOR_SETTER_ROLE` from all accounts. This eliminates the team's ability to remove EXE.
6. Bob has disabled all mechanisms that could prevent EXE's execution upon timelock expiration.
7. The team's only remaining countermeasure is warning users to withdraw assets and hoping they will withdraw before the timelock expires.
8. Users typically do not withdraw assets quickly enough to mitigate this risk.
9. EXE's timelock expires.
10. Bob makes swaps that execute EXE. Because EXE executes via `delegatecall` in the router's context, attacker-controlled code manipulates the vault's accounting logic to inflate ACC's balances of all tokens owned by the router.
11. Finally, Bob withdraws all remaining assets from the router.

Recommendations

The exploit scenario could have been stopped if the team had maintained access to a `PAUSER_ROLE` without the attacker gaining access to an `UNPAUSER_ROLE`, which allows unpausing of the contract. Prevent accounts with other roles from revoking `PAUSER_ROLE`. Prevent accounts with other roles from gaining access to `PAUSER_ROLE`.

The only way this can be achieved in OpenZeppelin's `AccessControl` is by revoking `DEFAULT_ADMIN_ROLE` from all accounts after deployment and initial permission setup. This removes the dangerous `DEFAULT_ADMIN_ROLE` entirely. To keep the ability to grant the same role, make an account with `EXECUTOR_SETTER_ROLE` admin for `EXECUTOR_SETTER_ROLE`. Apply the same to other roles.

Preventive Measures

Avoid omnipotent, admin, or root accounts that can grant or revoke all permissions.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-execution/pull/52>.

2. Timelock Duration Based on Block Number Instead of Timestamp

Severity: Low (Executor could get unlocked earlier than expected)

Difficulty: High (Requires block time anomaly)

Fix Status: Fixed

CWE: [807 - Reliance on Untrusted Inputs in a Security Decision](#)

Git Commit: [ebf1d0e605b795775472bb793f43f958344f219e](#)

Description

An executor's timelock is released after a certain `block.number`:

```
function _callSwapOnExecutor(
    address executor,
    uint256 amount,
    bytes calldata data,
    bool isFirstSwap,
    bool isSplitSwap,
    address receiver
) internal returns (uint256 calculatedAmount) {
    uint64 activationBlock =
    executorsActivationBlock[executor];
    // ...
    if (block.number < activationBlock) {
        revert
    }
    Dispatcher__ExecutorIsTimelocked(executor);
    // ....
}
```

Source: [Dispatcher.sol](#)

Block production times vary and historical anomalies are documented on the [Ethereum block time chart](#).

`block.timestamp` is a more reliable metric for measuring elapsed real-world time than `block.number`.

Exploit Scenario

1. An executor is added and locked until the block that is expected in three days.
2. A block time anomaly occurs.
3. The executor is unlocked after eight hours.
4. Users have too little time to perform due diligence on the new executor.

Recommendations

Lock the executor until `block.timestamp` instead of `block.number`.

Preventive Measures

Always use `block.timestamp` for time-based logic. Avoid using `block.number` as a proxy for elapsed time.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-execution/pull/49>.

3. Repeated Executor Addition Resets Timelock

Severity: Low (DoS due to sudden failure of swaps using specific executor)

Difficulty: Medium (Executor is added a second time)

Fix Status: Fixed

CWE: [837 - Improper Enforcement of a Single, Unique Action](#)

Git Commit: [ebf1d0e605b795775472bb793f43f958344f219e](#)

Description

If an executor is added a second time, its timelock is overwritten:

```
function _setExecutor(address target) internal {
    if (target.code.length == 0) {
        revert Dispatcher__NonContractExecutor();
    }

    executorsActivationBlock[target] =
        uint64(block.number +
blocksToDelayExecutorActivation);
    emit ExecutorSet(target);
}
```

Source: [Dispatcher.sol](#)

Exploit Scenario

1. Executor EXE is active and being used in swaps.
2. EXE is added a second time, which resets its timelock.
3. EXE is unavailable until the timelock expires.
4. Swaps that use EXE fail.

Recommendations

Prevent duplicate executor additions by rejecting or ignoring subsequent requests.

Preventive Measures

Ensure functions are idempotent or explicitly define behavior for repeated calls with identical arguments.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-execution/pull/49>.

4. Redundant Payable Modifier and Explicit Check

Severity: Informational

Difficulty: Not applicable

Fix Status: Fixed

CWE: [1076 - Insufficient Adherence to Expected Conventions](#)

Git Commit: [ebf1d0e605b795775472bb793f43f958344f219e](#)

Description

Several functions with **payable** modifier explicitly revert if `msg.value > 0`:

- [splitSwapUsingVault](#)
- [splitSwapPermit2](#)
- [sequentialSwapUsingVault](#)
- [sequentialSwapPermit2](#)
- [singleSwapUsingVault](#)
- [singleSwapPermit2](#)

Recommendations

Remove the **payable** modifier to make the function non-payable. This implicitly reverts on `msg.value > 0`, rendering the explicit check redundant.

The result is cleaner, more idiomatic Solidity code.

Preventive Measures

To enforce a zero message value, omit the **payable** modifier instead of using an explicit check like `msg.value > 0`.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-execution/pull/54>.

5. Deposit Function Is Not Pausable, Exposing Deposits During Attack Mitigation

Severity: Medium (Users can deposit into paused contract)

Difficulty: Medium (Requires paused contract)

Fix Status: Fixed

CWE: [693 - Protection Mechanism Failure](#)

Git Commit: [ebf1d0e605b795775472bb793f43f958344f219e](#)

Description

Only the `TychoRouter`'s swap-related functions are pausable:

- `splitSwap`
- `splitSwapUsingVault`
- `splitSwapPermit2`
- `sequentialSwap`
- `sequentialSwapUsingVault`
- `sequentialSwapPermit2`
- `singleSwap`
- `singleSwapUsingVault`
- `singleSwapPermit2`

No other functions are pausable.

The `deposit` function is not pausable. This exposes users to danger. See the exploit scenario below.

The `withdraw` function is not pausable and should not be pausable. Otherwise, any account with the `PAUSER_ROLE` could freeze assets.

The following public functions are currently not pausable:

- `batchGrantRole`
- `setExecutors` (recommendation: pausable)
- `removeExecutor` (recommendation: not pausable)
- `setFeeCalculator` (recommendation: pausable)
- **`receive`** (recommendation: pausable)
- **`fallback`** (recommendation: pausable)
- functions inherited from `AccessControl`, including `grantRole` and `revokeRole`

Exploit Scenario

1. An attack is in progress and vault balances are at risk.
2. An account with the `PAUSER_ROLE` calls **`pause()`** to mitigate the attack.
3. User Alice, who is unaware of the attack, calls `deposit` to deposit tokens into the router.
4. Alice's tokens get stolen.

Recommendations

Add the `whenNotPaused` modifier to `deposit`.

Decide for each item on the list of currently not pausable functions whether it should be pausable.

Add to the docstring of every public function why the function is or isn't pausable.

Preventive Measures

For each public function, decide carefully whether it should be pausable.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-execution/pull/51>.

6. Missing Expiration Block Parameter in ExecutorSet Event

Severity: Low

Difficulty: Low

Fix Status: Fixed

CWE: [778 - Insufficient Logging](#)

Git Commit: [ebf1d0e605b795775472bb793f43f958344f219e](#)

Description

To calculate the expiration block for an executor's timelock, off-chain watchers must extract `_blocksToDelayExecutorActivation` from the deployment transaction's input data and add it to the block number at which the `ExecutorSet` event was emitted.

Deriving this value manually hinders the user's ability to review new executors before their timelock expires.

Recommendations

Add the expiration block as an additional parameter to the `ExecutorSet` event.

Make `blocksToDelayExecutorActivation` public.

Preventive Measures

Include all relevant parameters in security-critical events to prevent blind spots during monitoring.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-execution/pull/49>.

7. Missing Contract Address Validation for Fee Calculator

Severity: Medium (Denial of service due to sudden failure of all swaps)

Difficulty: Medium (Accidental setting of fee calculator to non-contract address)

Fix Status: Fixed

CWE: [1287 - Improper Validation of Specified Type of Input](#)

Git Commit: [ebf1d0e605b795775472bb793f43f958344f219e](#)

Description

Neither `TychoRouter`'s [constructor](#) nor its [setFeeCalculator](#) function validates whether the supplied fee calculator is a contract.

If the fee calculator is not a contract, fee calculation during swaps will revert.

Recommendations

Add `require(feeCalculator.code.length != 0)` to `TychoRouter`'s [constructor](#) and [setFeeCalculator](#) function.

Preventive Measures

Validate that all arguments intended as contracts contain deployed code.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-execution/pull/48>.

8. Permit2 Can Be Set to Non-Contract Address

Severity: Low (Permit2 swaps fail and contract must be redeployed)

Difficulty: High (Accidental deployment with non-contract Permit2 address)

Fix Status: Fixed

CWE: [1287 - Improper Validation of Specified Type of Input](#)

Git Commit: [ebf1d0e605b795775472bb793f43f958344f219e](#)

Description

Dispatcher's `constructor` does not check whether its `_permit2` parameter is a contract.

If `_permit2` is not a contract, Permit2-related swaps will fail. The router would be non-functional and would have to be redeployed.

Recommendations

Add `require(_permit2.code.length != 0)` to Dispatcher's `constructor`.

Preventive Measures

Validate that all arguments intended as contracts contain deployed code.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-execution/pull/48>.

9. Inconsistent Naming for State Variables and Colliding Parameters

Severity: Informational

Difficulty: Not applicable

Fix Status: Fixed

CWE: [1099 - Inconsistent Naming Conventions for Identifiers](#)

Git Commit: [ebf1d0e605b795775472bb793f43f958344f219e](#)

Description

Given the name `fooBarBaz`, the Solidity style guide recommends using `fooBarBaz` for public entities, `_fooBarBaz` for [non-public entities](#) and `fooBarBaz_` for [function parameters that clash with existing names](#).

Recommendations

Adopt the naming convention consistently.

Incomplete list of inconsistencies:

- `uint256 private immutable blocksToDelayExecutorActivation`
- `uint256 private constant ALLOWED_DUST = 2`
- `address _permit2`

Preventive Measures

- Use leading underscore (`_foo`) for [non-public entities](#).
- Use trailing underscore (`foo_`) for [function parameters that clash with existing names](#).

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-execution/pull/55>.

10. Low Delay Accepted for Executor Timelock During Deployment

Severity: Low

Difficulty: High (Deployment with low timelock)

Fix Status: Fixed

CWE: 1188 - Initialization of a Resource with an Insecure Default

Git Commit: [ebf1d0e605b795775472bb793f43f958344f219e](https://github.com/propeller-heads/tycho-execution/pull/49)

Description

The executor timelock is an important security control. Without the timelock, a compromised `DEFAULT_ADMIN_ROLE` or `EXECUTOR_SETTER_ROLE` can be used **immediately** to execute arbitrary code, which in turn can **steal all of the router's assets**. The timelock gives the team and `TychoRouter`'s users time to mitigate the consequences of such a compromise.

`TychoRouter`'s [constructor](#) takes `_blocksToDelayExecutorActivation` as an argument. Passing `0` or a sufficiently low value disables the timelock immediately upon deployment.

Recommendations

In the constructor, validate that `_blocksToDelayExecutorActivation` corresponds to at least three days.

Preventive Measures

Hardcode or validate constructor arguments that are essential to security, such as fixed addresses (Permit2, WETH) and timelock durations.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-execution/pull/49>.

11. Incorrect Cyclic Swap Accounting Enables Asset Theft

Severity: High (Attacker can steal all of the router's assets)

Difficulty: Medium (Requires deployment of specific exploit contract)

Fix Status: Fixed

CWE: [682 - Incorrect Calculation](#)

Git Commit: [c711715039a5266fd4f9db3f4ee5357d388c634a](#)

Description

By choosing the right inputs, including `SlipstreamsExecutor` and a cyclic swap, an attacker can cause `singleSwapUsingVault` to arbitrarily increase their vault balance of a chosen token. They can then withdraw the balance. Repeating this process can drain all assets from the router.

The initial variant of this finding was found through vulnerability research.

Additional variants were found via [modeling](#):

- Using `UniswapV3Executor` and accruing balance.
- Using `CurveExecutor` and accruing allowance.
- Using `UniswapV2Executor` and transferring assets to attacker.
- Using `MaverickV2Executor` and transferring assets to attacker.

The recommendation and fix apply to all variants.

Exploit Scenario

1. Attacker deploys contract `EXP`.
2. Attacker calls `singleSwapUsingVault`.
3. Attacker chooses `SlipstreamsExecutor`.
4. Since the attacker controls the swap's `tokenIn` and `tokenOut`, they control `isCyclic` and make it `true`.
5. `_transfer` is a no-op because `SlipstreamsExecutor` sets `transferType` to `None`.
6. The attacker has set the executor's `pool` variable (bytes 43 to 63 of the swap's protocol data) to `EXP`.
7. Since the attacker controls `pool`, `pool.swap(...)` executes successfully without effect.
8. Since the attacker has caused `isCyclic` to be `true`, the problematic `balanceAfterSwap += amount` operation increases the balance that is measured after the swap by the attacker-controlled `amount`, without grounds to do so.
9. `amountOut = balanceAfterSwap - balanceBeforeSwap = amount - 0 = amount`
10. Since the attacker set `receiver` to `address(this)`, the `_updateDeltaAccounting(tokenOut, int256(amountOut))` operation is executed, which later causes an increase in the attacker's vault balance of `tokenOut` by the attacker-controlled `amount`.
11. Finally, the attacker withdraws their balance.

Recommendations

Disallow single-hop cyclic swaps and remove the dangerous `balanceAfterSwap += amount` operation.

Preventive Measures

To reduce attack surface, compute parameters from context if possible instead of relying on user input.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-execution/pull/114>. After the model was updated to reflect the fixes, the vulnerability was no longer detected.

12. Zero Input Amount Bypasses Vault Delta Validation Allowing Asset Theft

Severity: High (Attacker can steal all of the router's assets)

Difficulty: Medium (Requires deployment of specific exploit contract)

Fix Status: Fixed

CWE: [1023 - Incomplete Comparison with Missing Factors](#)

Git Commit: [c711715039a5266fd4f9db3f4ee5357d388c634a](#)

Description

This finding was discovered using [modeling](#).

`TychoRouter` uses a delta accounting mechanism.

For example, if assets are transferred out of `TychoRouter`, a corresponding delta is stored in transient storage:

```
_updateDeltaAccounting(tokenIn, -int256(amount));  
amount = _transferOut(tokenIn, receiver, amount);
```

Source: [TransferManager.sol](#)

Negative deltas can be thought of as temporary debts, which are owed to `TychoRouter`.

At the end of the swap, `_finalizeBalances` verifies the remaining nonzero deltas as follows:

```
if (useVault) {
    if (nonZeroCount > 1) {
        revert();
    } else if (nonZeroCount == 1) {
        int256 inputDelta = _getDelta(inputToken);
        if (inputDelta != -int256(inputAmount)) {
            revert();
        }
        uint256 id = _toId(inputToken);
        _burn(user, id, inputAmount);
    }
} else {
    if (nonZeroCount > 0) {
        revert();
    }
}
```

Source: [Vault.sol](#).

If the swap input is not taken from the vault and any nonzero deltas remain, execution reverts.

However, if the swap input is taken from the vault and a single nonzero delta remains, then the input delta is checked against the input amount. When both values are zero, the check passes, leaving the nonzero delta unvalidated.

By setting the input amount to zero, a nonzero delta can be accrued that is not checked.

Such a nonzero delta is accrued during the transfer triggered by callbacks from either `SlipstreamsExecutor` or `UniswapV3Executor`.

The transfer can drain a specific asset from the router. Repeating this process can drain all assets from the router.

Exploit Scenario

1. Attacker deploys contract EXP.
2. Attacker calls `singleSwapUsingVault`
3. Attacker sets `amountIn` to `0` to later cause the `inputDelta` of `0`.
4. `minAmountOut` is set to `1` so execution doesn't revert.
5. `clientFeeParams.maxClientContribution` is set to `1` and `clientFeeParams.clientFeeReceiver` is set to some address that owns `1` `tokenOut`. This ensures that execution succeeds later despite an output amount of `0`.
6. Attacker chooses `SlipstreamsExecutor` or `UniswapV3Executor`. Exploits using either executor were discovered using [modeling](#).
7. `_transfer` is a no-op because `SlipstreamsExecutor` sets `transferType` to `None`.
8. The attacker sets the executor's `pool` variable (bytes 43 to 63 of the swap's protocol data) to EXP.
9. Since the attacker controls `pool`, `pool.swap(...)` succeeds and executes a callback into `TychoRouter`.
10. Since the attacker controls the callback's calldata, they control some return values of `SlipstreamsExecutor.getCallbackTransferData`:
 1. The attacker sets `tokenIn` to USDC or some other token they want to steal.
 2. The attacker sets `amount` to the amount they want to steal.
11. During the `_transfer` triggered by the callback:
 1. EXP (i.e., the attacker) receives `amount` of `tokenIn`
 2. The transaction also [accrues a negative delta of](#) `amount` of `tokenIn`.
12. `_finalizeBalances` [ignores](#) the negative delta accrued by the previous step due to the `inputDelta` of `0`.

Recommendations

Ensure that the input amount is not zero.

Ensure that the input delta is not zero if the input is taken from the vault and a single nonzero delta remains.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-execution/commit/91ccd07b9b4146b1b683394da5ef2bc2ec88f69d>.

After the model was updated to reflect the fixes, the vulnerability was no longer detected.

13. List of Sequential Swaps Allowed to Be Empty

Severity: Informational

Difficulty: Not applicable

Fix Status: Fixed

CWE: [1284 - Improper Validation of Specified Quantity in Input](#)

Git Commit: [c711715039a5266fd4f9db3f4ee5357d388c634a](#)

Description

None of the sequential swap functions revert if the list of swaps is empty.

While no vulnerabilities were found that can make use of an empty swap list, allowing empty swaps is undesirable.

Recommendations

Ensure that the list of sequential swaps is not empty.

Preventive Measures

For every user input array, check whether an empty array is acceptable.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-execution/pull/105>.

14. Duplicated Single Swap Logic

Severity: Informational

Difficulty: Not applicable

Fix Status: Risk Accepted by Propeller Heads

CWE: [1041 - Use of Redundant Code](#)

Git Commit: [c711715039a5266fd4f9db3f4ee5357d388c634a](#)

Description

A single swap is equivalent to a sequential swap with one swap.

Dedicated single swap functions exist in order to achieve a small decrease in gas usage.

This results in around 170 lines of additional duplicate code.

If a bug is found in the duplicated code, it has to be fixed in two places instead of one. If a change is made to the duplicated code, it has to be made in two places instead of one. There's a chance one place is forgotten.

Tests and security assessments have to cover all duplicates, costing additional time.

Recommendations

Remove the single swap functions in order to remove the duplicate code.

Preventive Measures

Avoid nontrivial duplicate logic. Duplicates create the risk that fixes are not applied to all copies. Duplicates multiply testing and review effort of the logic in question.

15. Zero Address Represents Both Missing Address and Native Token

Severity: Informational

Difficulty: Not applicable

Fix Status: Fixed

CWE: [1024 - Comparison of Incompatible Types](#)

Git Commit: [fc8f17e97bd47676e107c95d1a51912af1d89917](#)

Description

The project uses `address(0)` sometimes to represent the absence of an address and other times to represent the native token (Ether).

For example, when `CurveExecutor.getTransferData` returns `tokenIn = address(0)`, `address(0)` means the native token, but when `EkuboExecutor.getTransferData` returns `receiver = address(0)`, `address(0)` means that the `receiver` field is empty.

Since Solidity doesn't have an option type, `address(0)` must be used to represent an address that is not present. If `address(0)` is also used to represent the native token, it can lead to ambiguity and bugs.

`tstore(_SWAP_INPUT_TOKEN_SLOT, 0)` is intended to unset the input token address in transient storage. As an unintended side effect, this operation sets the input token address to the native token address. While no related vulnerabilities were found, the chance that one is accidentally introduced is too high.

Recommendations

Use a unique marker address to represent the native token.

Curve, for example, uses the marker address

`0xEeeeeEeeeEeEeeEeEeEEEEEEEEEEEEEEEEEEEE` for this purpose.

Preventive Measures

Avoid giving a type's zero value any meaning other than zero. For example, avoid having `address(0)` mean the native token and avoid having `0` mean that an error occurred.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-indexer/pull/984>.

16. Incomplete Signature Input Enables Replay and Solver Balance Theft

Severity: High (Allows theft of solver balance)

Difficulty: Medium (Requires solver to sign sufficiently high contribution)

Fix Status: Partially fixed

CWE: [345 - Insufficient Verification of Data Authenticity](#)

Git Commit: [c711715039a5266fd4f9db3f4ee5357d388c634a](#)

Description

All of `TychoRouter`'s swap functions take a `clientFeeParams` argument of type `ClientFeeParams`.

```
struct ClientFeeParams {
    uint16 clientFeeBps;
    address clientFeeReceiver;
    uint256 maxClientContribution;
    uint256 deadline;
    bytes clientSignature; // 65-byte EIP-712 ECDSA sig
    by clientFeeReceiver
}
```

Source: [TychoRouter.sol](#).

The `maxClientContribution` field allows solvers to sponsor swaps that would otherwise fail due to slippage.

The solver provides their signature in the `clientSignature` field, allowing their router balances to be used as slippage compensation in order to increase the number of settlements:

```
if (amountOut < minAmountOut) {
    uint256 requiredContribution = minAmountOut -
amountOut;
    if (requiredContribution > maxClientContribution) {
        revert();
    }
    // Debit the client's vault balance
    _debitVault(client, tokenOut, requiredContribution);
    int256 outputDelta = _getDelta(tokenOut);
```

Source: [TychoRouter.sol](#)

Once a swap transaction is included in a block, the signed `ClientFeeParams` object is visible to anyone.

Only the domain separator and `ClientFeeParams`'s fields are signature input:

```
bytes32 digest = _hashTypedDataV4(
    keccak256(
        abi.encode(
            CLIENT_FEE_TYPEHASH,
            p.clientFeeBps,
            p.clientFeeReceiver,
            p.maxClientContribution,
            p.deadline
        )
    )
);
```

Source: [TychoRouter.sol](#)

For `_hashTypedDataV4` see [EIP712.sol](#).

Nonces are not used for replay protection.

Since the swap itself is not part of the signature input, the signed `ClientFeeParams` object can be reused for any swap.

This can be used to steal solver's balances.

Exploit Scenario

1. Solver Charlie deposits 50000 USDC into `TychoRouter`.
2. Solver Charlie deposits 100 DAI into `TychoRouter`.
3. User Alice uses Charlie's solver to swap WETH into DAI. To increase chances of settlement, Charlie signs a contribution of 5 DAI. Since DAI uses 18 decimals, 5 DAI corresponds to a `uint256` of $5 * 10^{**18}$.
4. The swap is included in a block and the signed contribution is visible to anyone.
5. Bob prepares a swap that produces a slippage of 50000 USDC. This is possible through flashloans and/or specific executors.
6. Having seen Charlie's contribution in the block, Bob uses it in the swap transaction. Since USDC has 6 decimals, the slippage corresponds to a `uint256` of $50000 * 10^{**6}$.
7. Because the `tokenOut` is not included in the signed message and the logic doesn't take decimals into account, Charlie's 5 DAI contribution corresponds to a $5 * 10^{**18} / 10^{**6}$ USDC contribution. More than enough to pay the 50000 USDC slippage.
8. `TychoRouter` sends 50000 USDC of Charlie's balance to Bob, to cover the slippage.

Recommendations

1. Include a unique nonce in the signature, remember used digests (not signatures, due to [malleability](#)), and allow each digest to be used only once. [Here's an example implementation](#).
2. Include `tokenOut` in the signed message, so it cannot be reused for tokens with different decimals.
3. Include `msg.sender` in the signed message, so it cannot be reused by other callers.
4. Include `receiver` in the signed message, so none other than the original receiver can benefit from the contribution.
5. Include the swaps in the signed message.

Fix Review

<https://github.com/propeller-heads/tycho-execution/pull/116> includes `amountIn`, `tokenIn`, `tokenOut`, `minAmountOut` and `receiver` in the signed message.

<https://github.com/propeller-heads/tycho-execution/pull/119> includes the swaps themselves in the signed message.

The exploit scenario is no longer possible.

If a solver were to sign a significant contribution that is higher than the transaction cost, the swap's `receiver` could still replay the swap to get the contribution until the solver's balance is exhausted.

Recommendation 1 would prevent replay attacks entirely.

17. Input Parsing Ambiguity Allows Divergent Transfer Type in CurveExecutor's getTransferData Function

Severity: Informational

Difficulty: Not applicable

Fix Status: Fixed

CWE: [1287 - Improper Validation of Specified Type of Input](#)

Git Commit: [c5b5c4209bc7cb560e9c662d264d499f972527f1](#)

Description

CurveExecutor's `getTransferData` function sets `tokenIn` to `address(0)` if it matches `nativeToken`. In this case `transferType` is set to `TransferNativeInExecutor`.

However, the caller can also directly set `tokenIn` to `address(0)`. In this case `transferType` is set to `ProtocolWillDebit`.

This could lead to unexpected behavior in downstream logic.

```

function getTransferData(bytes calldata data)
    external
    payable
    returns (
        TransferManager.TransferType transferType,
        address receiver,
        address tokenIn,
        address tokenOut,
        bool outputToRouter
    )
{
    tokenIn = address(bytes20(data[0:20]));
    tokenOut = address(bytes20(data[20:40]));
    if (tokenIn == nativeToken) {
        tokenIn = address(0);
        transferType =
TransferManager.TransferType.TransferNativeInExecutor;
    } else {
        transferType =
TransferManager.TransferType.ProtocolWillDebit;
    }
    if (tokenOut == nativeToken) {
        tokenOut = address(0);
    }
    receiver = address(bytes20(data[40:60]));
    outputToRouter = true;
}

```

Source: [CurveExecutor.sol](#)

Recommendations

In `CurveExecutor.getTransferData` enforce that `tokenIn` and `tokenOut` cannot be `address(0)`.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-indexer/pull/943>.

18. Delegatecall Where Staticcall Would Suffice

Severity: Medium (Code is unnecessarily executed with permission to transfer any of the router's assets)

Difficulty: High (Requires malicious or vulnerable executor)

Fix Status: Fixed

CWE: [250 - Execution with Unnecessary Privileges](#)

Git Commit: [bcd04d6d8fd1ab2b193dfc56d247bcb2512d82a7](#)

Description

An executor's `getCallbackTransferData` function is called via `delegatecall`, despite being a getter function that is not supposed to have side effects.

This creates unnecessary danger, since the `delegatecall` allows the executor's code to transfer any of the router's assets and write to any of the router's storage slots.

Additionally, `delegatecalls` must be verbose low-level calls while `staticcalls` can be high-level calls. Also see finding [20](#).

Recommendations

Call `getCallbackTransferData` via high-level `staticcall`.

Preventive Measures

- `Delegatecall` is one of Solidity's most dangerous operations. Never use `delegatecall` without a strong justification. Document the justification.
- Never use `delegatecall` to call functions that have no side effects (`view` or `pure`). `Staticcall` suffices and is far safer.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-indexer/commit/ee120f5bd2209c85bf3a6a3e4e58b94a83103ba2>.

19. Fee Calculator Update Lacks Timelock

Severity: Medium (Fee calculator output can put assets at risk)

Difficulty: High (Requires compromise of `ROUTER_FEE_SETTER_ROLE` multi-sig)

Fix Status: Fixed

CWE: [654 - Reliance on a Single Factor in a Security Decision](#)

Git Commit: [fc8f17e97bd47676e107c95d1a51912af1d89917](#)

Description

When the fee calculator is updated via `TychoRouter`'s [setFeeCalculator](#) function, the new fee calculator is used immediately for the next swap.

The data returned by the fee calculator can put assets at risk in downstream logic. This was confirmed by the team.

Exploit Scenario

1. Attacker Bob manages to compromise an account with `ROUTER_FEE_SETTER_ROLE`
2. Bob deploys a malicious fee calculator contract `EXP`
3. Bob makes `TychoRouter` use the malicious fee calculator by calling `TychoRouter.setFeeCalculator(EXP)`
4. Alice uses `TychoRouter` to make a swap
5. `TychoRouter` uses the malicious fee calculator `EXP`
6. The data returned by `EXP` causes Alice to lose assets

Neither Alice nor the team had a chance to react to step 1, since step 3 immediately enables step 5.

Recommendations

Add timelock to `TychoRouter`'s [setFeeCalculator](#) function.

Preventive Measures

Timelock all authorized operations that change the system's behavior. This gives the team and users time to detect and react to malicious changes.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-indexer/commit/00913377dd3a53f4951307d625027ca32c4143ac>.

20. Unnecessary Low-Level Calls

Severity: Informational

Difficulty: Not applicable

Fix Status: Fixed

CWE: [695 - Use of Low-Level Functionality](#)

Git Commit: [fc8f17e97bd47676e107c95d1a51912af1d89917](#)

Description

Several instances of `staticcall` are unnecessarily performed using low-level calls. Replacing them with high-level calls would result in shorter and safer code.

For example, in [Dispatcher.sol](#):

```
(bool transferDataSuccess, bytes memory transferData) =
executor.staticcall(

abi.encodeWithSelector(IExecutor.getTransferData.selector,
data)
);

if (!transferDataSuccess) {
    revert(
        string(
            transferData.length > 0
                ? transferData
                : abi.encodePacked("Getting transfer data
failed")
        )
    );
}

(
    TransferManager.TransferType transferType,
```

```

        address transferReceiver,
        address tokenIn,
        address tokenOut,
        bool outputToRouter
    ) = abi.decode(
        transferData,
        (TransferManager.TransferType, address, address,
address, bool)
    );

```

The above can be replaced by the following:

```

(
    TransferManager.TransferType transferType,
    address transferReceiver,
    address tokenIn,
    address tokenOut,
    bool outputToRouter
) = IExecutor(executor).getTransferData(data);

```

Since `IExecutor.getTransferData` is marked as **view**, `IExecutor(executor).getTransferData` will compile into a `staticcall`.

The second code snippet has several advantages over the first:

- Achieves in 7 lines of code what previously needed 24.
- Automated revert on failure ensures that failure handling is not forgotten.
- Automated bubbling up of error.
- Automated ABI-encoding of parameters ensures parameter encoding matches the interface.
- Automated ABI-decoding of return data ensures return data decoding matches the interface.
- Automatically ensures that target is a contract. Not performed because unnecessary if call returns data. Important because calls to EOAs always silently succeed.

If a function in an interface is **view** or **pure**, then Solidity will compile it into a `staticcall` when called via a [high-level call](#).

For more information, read the relevant Solidity documentation on [view functions](#) and [pure functions](#).

Recommendations

Replace the following low-level `staticcall`s by high-level calls, which automatically compile into `staticcall`s as long as the called functions are marked as **view** or **pure**:

- <https://github.com/propeller-heads/tycho-indexer/blob/fc8f17e97bd47676e107c95d1a51912af1d89917/crates/tycho-execution/contracts/src/Dispatcher.sol#L113>
- <https://github.com/propeller-heads/tycho-indexer/blob/fc8f17e97bd47676e107c95d1a51912af1d89917/crates/tycho-execution/contracts/src/Dispatcher.sol#L231>
- <https://github.com/propeller-heads/tycho-indexer/blob/fc8f17e97bd47676e107c95d1a51912af1d89917/crates/tycho-execution/contracts/src/Dispatcher.sol#L305>
- <https://github.com/propeller-heads/tycho-indexer/blob/fc8f17e97bd47676e107c95d1a51912af1d89917/crates/tycho-execution/contracts/src/Dispatcher.sol#L344>
- <https://github.com/propeller-heads/tycho-indexer/blob/fc8f17e97bd47676e107c95d1a51912af1d89917/crates/tycho-execution/contracts/src/Dispatcher.sol#L374>

Preventive Measures

Prefer [high-level calls](#) (`Interface(address).functionName(param1, param2)`) over [low-level calls](#) (`address.call(raw_calldata)`). High-level calls require less code and provide automatic safeguards.

Fix Review

Fixed in <https://github.com/propeller-heads/tycho-indexer/commit/15c2a0c571772f1cf12f780cb8be2a370c41791a>.