ASSURE DEFI®

THE VERIFICATION GOLD STANDARD

SECURITY ASSESSMENT REPORT

NAME:	ZYGOSWAP
STATUS:	FAIL
DATE:	12/02/2026

Risk Analysis

Vulnerability summary

Classification	Description
🔴 High	Vulnerabilities that can lead to **loss or theft of funds**, **permanent locking of assets**, **unauthorized minting/burning**, **protocol insolvency**, or **full control of the contract or protocol**. These issues are typically **exploitable on-chain** and require immediate remediation.
🟠 Medium	Vulnerabilities that **weaken protocol security or trust assumptions** but do not directly enable immediate loss of funds under normal conditions. Exploitation may require **specific conditions, privileged roles, or external failures**.
🟡 Low	Issues that have **limited direct impact** on funds or protocol integrity but may cause **unexpected behavior**, **reduced resilience**, or **future exploitability** if combined with other weaknesses.
🟢 Informational	Findings that **do not represent security vulnerabilities**, but highlight **code quality, clarity, maintainability, or best-practice improvements** that may enhance long-term safety and auditability.

SCOPE

Target Code And Revision

Project	Zygo DEX contracts [Router, claim and swaptoken]
Codebase	https://github.com/goran9999/zygo-dex Commit [a6542bd6a0e7441d680d76a70f777b9b314637c2]
Audit Methodology	Static, Manual

Detailed Technical Report

1. Reentrancy enables repeated claims and token drain

Location:

ZygoClaim.sol - claimToken(string deviceId, bytes[] signatures, uint256[] amounts)

(Indirectly via) external calls: IERC20(rewardToken).safeTransfer(...)

Issue:

claimToken() performs external token transfers before it marks the deviceId as claimed. The state change:

tokenClaims[deviceId] = 1;

happens only after the loop of transfers finishes.

If any reward token in rewardTokens is callback-capable (e.g., ERC777-style hooks, malicious ERC20 with reentrant transfer logic, or a token with a custom transfer that calls back), it can re-enter claimToken() during safeTransfer() while tokenClaims[deviceId] is still 0.

The re-entered call passes the “not claimed yet” require and triggers another set of transfers.

This repeats until the contract’s balances are drained.

Remediation:

Apply Checks-Effects-Interactions:

Set tokenClaims[deviceId] = 1; before any external transfers.

Add nonReentrant (OpenZeppelin ReentrancyGuard) to claimToken().

Optionally redesign as pull-based withdrawals (record claim amounts then withdraw per token) to reduce external-call surface.

2. Signatures are bearer tokens: not bound to recipient / contract / chain (replay & theft)

Location:

ZygoClaim.sol : _verifySignature(bytes32 messageHash, bytes signature), claimToken(...)

Signed payload: keccak256(abi.encodePacked(deviceId, amount))

Issue:

The signed message only includes (deviceId, amount) and omits critical domain/context binding such as:

- intended recipient (msg.sender or an explicit recipient)
- verifying contract address (address(this))
- block.chainid
- nonce and/or expiry (deadline)
- reward token address (see next finding)

As a result, any party who obtains a valid signature can redeem it to themselves (signature theft), and signatures can also be replayed across:

- different deployments of the same contract (different addresses)
- different chains (same message hash)

This converts signatures into transferable bearer coupons rather than authorization for a specific user on a specific deployment.

Remediation:

Switch to EIP-712 typed structured data and bind at minimum:

deviceId, recipient, token, amount, nonce, deadline, chainId, verifyingContract

Store and enforce nonces (per deviceId or per recipient+deviceId).

Enforce expiry (deadline) to limit leakage impact.

Add the msg.sender verification in the signature

3. Signatures are not bound to token address/index (cross-token value theft)

Location:

ZygoClaim.sol - claimToken(...) loop over rewardTokens[i]

Signed payload: keccak256(abi.encodePacked(deviceId, amount)) (no token)

Issue:

Inside the claim loop, the contract verifies the signature over (deviceId, amount) and then transfers amount of rewardTokens[i]. Because the token address (or index) is not included in the signed message, a signature intended for one token is valid for any token in rewardTokens at the same amount.

Exploit scenario (if contract holds multiple tokens with different values):

Admin signs claim for “10,000 units” intended for a low-value token (for example USDC units).

Claimer reuses the same signature against the index corresponding to a higher-value token (like WETH) if balances allow, draining higher-value assets.

Remediation:

Include token (and optionally tokenIndex) in the signed payload (EIP-712 strongly preferred):

for example hash deviceId + recipient + token + amount + nonce + deadline + chainId + verifyingContract

Validate that the provided signature matches the specific token being claimed.

4. ERC20 swap path charges fee twice and traps funds in router

Location:

ZygoRouter.sol - swap(address fromToken, address toToken, uint256 inputAmount, uint256 minOutputAmount)

ERC20 input path (non-ETH)

Issue:

On ERC20 swaps, the router pulls inputAmount from the user into the router, then separately pulls fee from the user to feeWallet, and also subtracts fee from inputAmount before swapping:

- User pays inputAmount to router
- User pays fee additionally to feeWallet
- Router swaps only (inputAmount - fee)

This results in fee tokens remaining in the router contract (since router holds inputAmount but only uses inputAmount - fee), creating trapped funds with no retrieval path shown (unless another method exists). Users effectively pay the fee twice (once to feeWallet, once as stranded value in the router).

Remediation:

Pick one coherent fee model:

Fee taken from input (typical):

- Pull inputAmount once
- Transfer fee from router to feeWallet
- Swap inputAmount - fee

Fee charged on top (less typical):

- Pull inputAmount and swap full amount
- Separately pull fee (don’t subtract from inputAmount), and clearly document UX

Also add an explicit accounting/rescue approach if the router can ever retain balances.

MEDIUM

1. Raw ERC20 calls (no SafeERC20) create swap failures / inconsistent behavior on non-standard tokens

Location:

ZygoRouter.sol - swap(...) uses transfer, transferFrom, approve directly

Any function performing raw ERC20 operations without SafeERC20

Issue:

The router uses raw IERC20.transfer/transferFrom/approve. Many real-world tokens are non-standard or have edge-case rules:

- Some return false instead of reverting on failure
- Some require setting allowance to 0 before changing it (like, USDT-style behavior)
- Some have fee-on-transfer behavior that breaks assumptions

This can cause:

- unexpected reverts/DoS for specific assets
- silent failures (if return values aren't handled)
- allowance-update issues when swapping repeatedly

Remediation:

Replace raw calls with OpenZeppelin SafeERC20:

safeTransfer, safeTransferFrom, safeApprove/safeIncreaseAllowance

For allowance changes, prefer safeIncreaseAllowance or set-to-zero-then-set patterns where required.

If supporting fee-on-transfer tokens, measure actual received amounts and adapt logic accordingly.

LOW

1. Missing validation on rewardTokens configuration (zero addresses, unbounded array)

Location:

ZygoClaim.sol — constructor(...) (initial rewardTokens setup, if applicable)

ZygoClaim.sol — setRewardTokens(address[] calldata _rewardTokens) (owner-only)

Issue:

rewardTokens can be set with:

- address(0) entries, and/or
- an excessively large array length.

Because claimToken() iterates over rewardTokens, misconfiguration can cause:

- reverts (for example, attempting to safeTransfer to/from address(0) token address),
- unexpectedly high gas usage (large array), making claims impractical.

Remediation:

Add input validation in constructor and setRewardTokens():

Reject zero addresses:

- require(_rewardTokens[i] != address(0), "zero token");

Enforce a reasonable maximum length:

- require(_rewardTokens.length <= MAX_REWARD_TOKENS, "too many tokens");

Optionally prevent duplicates to avoid repeated transfers to the same token address.

INFORMATIONAL

1. Reentrancy guard recommendation for swap() (defense-in-depth)

Location:

ZygoRouter.sol - swap(address fromToken, address toToken, uint256 inputAmount, uint256 minOutputAmount)

Issue:

There is no clear, reproducible exploit showing that swap() is currently reentrancy-vulnerable in a way that allows theft or state corruption, because the function does not maintain per-user accounting/state that can be manipulated across an external call (unlike typical vault/claim patterns), and the critical transfer logic is not “state-after-external-call” in a way that benefits an attacker.

However, swap() does interact with external tokens and external router contracts (via approvals and router calls). Adding nonReentrant can reduce the blast radius of:

unexpected token callback behavior,

malicious/non-standard token implementations,

future code changes that introduce state updates after external calls.

So this is best categorized as defense-in-depth, not a confirmed vulnerability.

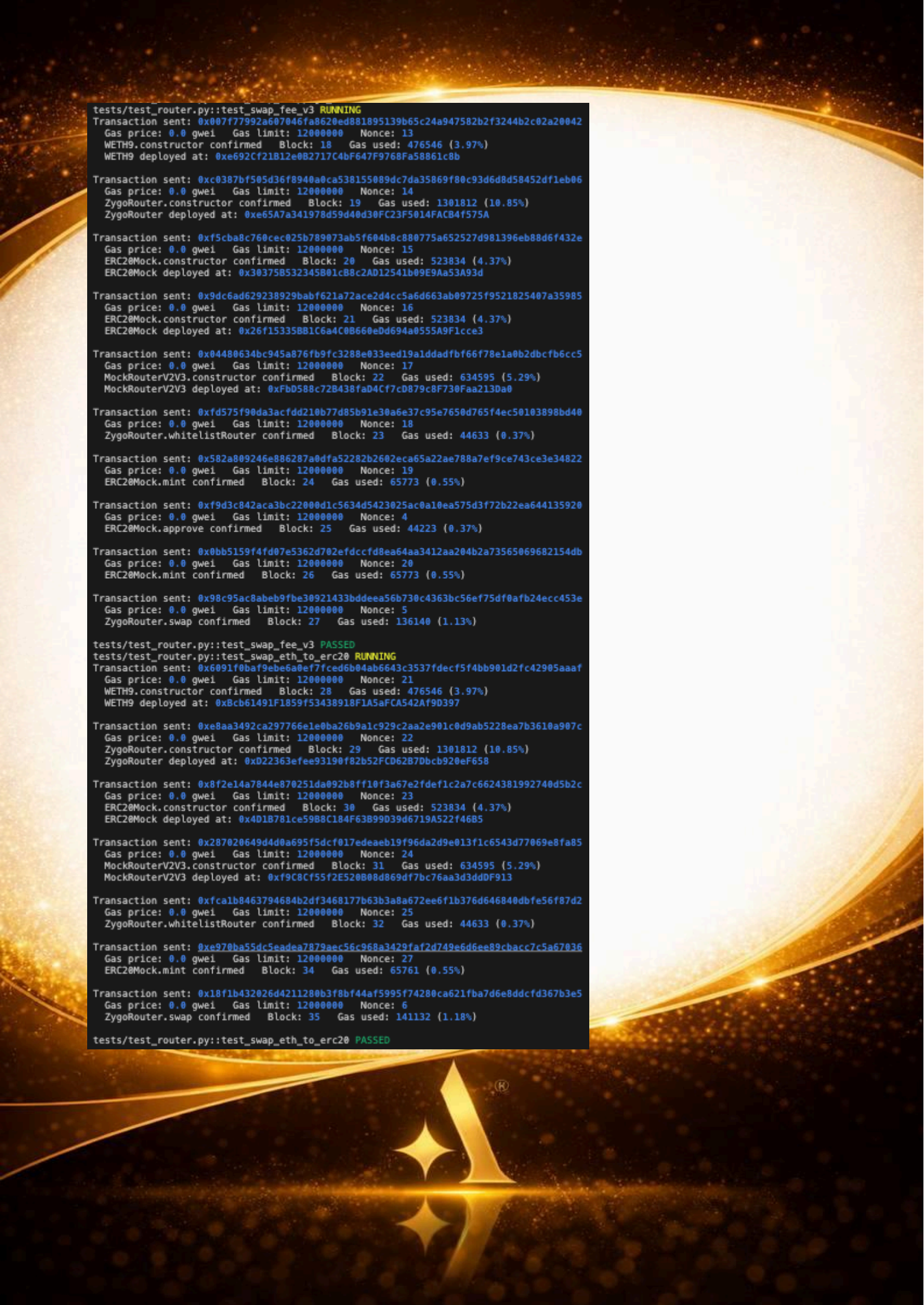
Remediation:

Add OpenZeppelin ReentrancyGuard and mark swap() as nonReentrant if you want an extra safety layer.

Testing coverage

During the testing phase, custom use cases were written to cover all the logic of contracts. *Check “Annexes” to see the testing code.*

```
tests/test_router.py::test_swap_fee_v3 RUNNING
Transaction sent: 0x007f77992a607046fa8620ed881895139b65c24a947582b2f3244b2c02a20042
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 13
  WETH9.constructor confirmed   Block: 18   Gas used: 476546 (3.97%)
  WETH9 deployed at: 0xe692Cf21B12e0B2717C4bF647F9768Fa58861c8b

Transaction sent: 0xc0387bf505d36f8940a0ca538155089dc7da35869f80c93d6d8d58452df1eb06
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 14
  ZygoRouter.constructor confirmed   Block: 19   Gas used: 1301812 (10.85%)
  ZygoRouter deployed at: 0xe65A7a341978d59d40d30FC23F5014FACB4f575A

Transaction sent: 0xf5cba8c760cec025b789073ab5f604b8c880775a652527d981396eb88d6f432e
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 15
  ERC20Mock.constructor confirmed   Block: 20   Gas used: 523834 (4.37%)
  ERC20Mock deployed at: 0x30375B532345B01cB8c2AD12541b09E9Aa53A93d

Transaction sent: 0x9dc6ad629238929babf621a72ace2d4cc5a6d663ab09725f9521825407a35985
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 16
  ERC20Mock.constructor confirmed   Block: 21   Gas used: 523834 (4.37%)
  ERC20Mock deployed at: 0x26f15335BB1C6a4C0B660eDd694a0555A9F1cce3

Transaction sent: 0x04480634bc945a876fb9fc3288e033eed19a1ddadfbf66f78e1a0b2dbcfb6cc5
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 17
  MockRouterV2V3.constructor confirmed   Block: 22   Gas used: 634595 (5.29%)
  MockRouterV2V3 deployed at: 0xFbD588c72B438faD4Cf7cD879c8F730Faa213Da0

Transaction sent: 0xfd575f90da3acfdd210b77d85b91e30a6e37c95e7650d765f4ec50103898bd40
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 18
  ZygoRouter.whitelistRouter confirmed   Block: 23   Gas used: 44633 (0.37%)

Transaction sent: 0x582a809246e886287a0dfa52282b2602eca65a22ae788a7ef9ce743ce3e34822
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 19
  ERC20Mock.mint confirmed   Block: 24   Gas used: 65773 (0.55%)

Transaction sent: 0xf9d3c842aca3bc22000d1c5634d5423025ac0a10ea575d3f72b22ea644135920
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 4
  ERC20Mock.approve confirmed   Block: 25   Gas used: 44223 (0.37%)

Transaction sent: 0x0bb5159f4fd07e5362d702efdccfd8ea64aa3412aa204b2a735650696821 54db
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 20
  ERC20Mock.mint confirmed   Block: 26   Gas used: 65773 (0.55%)

Transaction sent: 0x98c95ac8abeb9fbe30921433bddeea56b730c4363bc56ef75df0afb24ecc453e
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 5
  ZygoRouter.swap confirmed   Block: 27   Gas used: 136140 (1.13%)

tests/test_router.py::test_swap_fee_v3 PASSED
tests/test_router.py::test_swap_eth_to_erc20 RUNNING
Transaction sent: 0x6091f0baf9ebe6a0ef7fced6b04ab6643c3537fdecf5f4bb901d2fc42905aaaf
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 21
  WETH9.constructor confirmed   Block: 28   Gas used: 476546 (3.97%)
  WETH9 deployed at: 0xBcb61491F1859f53438918F1A5aFCA542Af9D397

Transaction sent: 0xe8aa3492ca297766e1e0ba26b9a1c929c2aa2e901c0d9ab5228ea7b3610a907c
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 22
  ZygoRouter.constructor confirmed   Block: 29   Gas used: 1301812 (10.85%)
  ZygoRouter deployed at: 0xD22363efee93190f82b52FCD62B7Dbcb920eF658

Transaction sent: 0x8f2e14a7844e870251da092b8ff10f3a67e2fdef1c2a7c6624381992740d5b2c
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 23
  ERC20Mock.constructor confirmed   Block: 30   Gas used: 523834 (4.37%)
  ERC20Mock deployed at: 0x4D1B781ce59B8C184F63B99D39d6719A522f46B5

Transaction sent: 0x287020649d4d0a695f5dcf017edeaeb19f96da2d9e013f1c6543d77069e8fa85
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 24
  MockRouterV2V3.constructor confirmed   Block: 31   Gas used: 634595 (5.29%)
  MockRouterV2V3 deployed at: 0xf9C8Cf55f2E520B08d869df7bc76aa3d3ddDF913

Transaction sent: 0xfca1b8463794684b2df3468177b63b3a8a672ee6f1b376d646840dbfe56f87d2
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 25
  ZygoRouter.whitelistRouter confirmed   Block: 32   Gas used: 44633 (0.37%)

Transaction sent: 0xe970ba55dc5eadea7879aec56c968a3429faf2d749e6d6ee89cbacc7c5a67036
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 27
  ERC20Mock.mint confirmed   Block: 34   Gas used: 65761 (0.55%)

Transaction sent: 0x18f1b432026d4211280b3f8bf44af5995f74280ca621fba7d6e8ddcfd367b3e5
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 6
  ZygoRouter.swap confirmed   Block: 35   Gas used: 141132 (1.18%)

tests/test_router.py::test_swap_eth_to_erc20 PASSED
```



```
tests/test_claim.py::test_claim_success RUNNING
Transaction sent: 0x9eba0e7d0e3079f9ff9f71c8a0821cea3c5119f37e55928ed529395edbe3dcdd
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  ERC20Mock.constructor confirmed   Block: 1   Gas used: 523846 (4.37%)
  ERC20Mock deployed at: 0x3194cBDC3dbcd3E11a07892e7bA5c3394048Cc87

Transaction sent: 0xc4153bef9f001206d4479c5606dab9ca96ef07413afc25d9aa2e0c439add950f
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 1
  ZygoClaim.constructor confirmed   Block: 2   Gas used: 927111 (7.73%)
  ZygoClaim deployed at: 0x602C71e4DAC47a042Ee7f46E0aee17F94A3bA0B6

Transaction sent: 0x0ee174a3dd8a872337ca66b4487aaabe3e5a5404fa9be58b765e44f3f90be2f2
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  ZygoClaim.transferOwnership confirmed   Block: 3   Gas used: 30141 (0.25%)

Transaction sent: 0x40c6840b0c4377f6540c485bc2055190348c4a48277b761b186d0cf2372d7913
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 3
  ERC20Mock.mint confirmed   Block: 4   Gas used: 65821 (0.55%)

Transaction sent: 0xb05b50d3088e3c0ca6413493e80a0ce05598288df2b149992fd8f2ff9ea57b9b
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 4
  ERC20Mock.transfer confirmed   Block: 5   Gas used: 51060 (0.43%)

Transaction sent: 0x38718de7bbad63a8c01b402f70007e53d66be0c367a70889d2010790d11c9f64
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  ZygoClaim.claimToken confirmed   Block: 6   Gas used: 73820 (0.62%)

tests/test_claim.py::test_claim_success PASSED
tests/test_claim.py::test_cannot_claim_twice RUNNING
Transaction sent: 0x414158665595d9a448fd300ff84b79c3e245b9bd792f7f63911e18d9457e665f
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 5
  ERC20Mock.constructor confirmed   Block: 7   Gas used: 523846 (4.37%)
  ERC20Mock deployed at: 0x6b4BDe1086912A6Cb24ce3dB43b3466e6c72AFd3

Transaction sent: 0xb941d6ae2e54c783e239c199cf81a09f1f408b584eb68b6bd5252f6f6e7e57e7
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 6
  ZygoClaim.constructor confirmed   Block: 8   Gas used: 927111 (7.73%)
  ZygoClaim deployed at: 0x9E4c14403d7d9A8A782044E86a93CAE09D7B2ac9

Transaction sent: 0xdcae69481b8343fc6282c90b7a9b0d826ef8089a7c5fb64766a147ccf764f403
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 7
  ZygoClaim.transferOwnership confirmed   Block: 9   Gas used: 30141 (0.25%)

Transaction sent: 0x01bfde86b37970f6182abaf1087844ce593c6d3b2cd925c7c441716f1fdebbaa
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 8
  ERC20Mock.mint confirmed   Block: 10   Gas used: 65821 (0.55%)

Transaction sent: 0x3cf68f49a3cc49c14d652410a28dfa5cdbd43e42b6d40e7ff3c1e019deba8ae7
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 9
  ERC20Mock.transfer confirmed   Block: 11   Gas used: 51060 (0.43%)

Transaction sent: 0xa779758905l7e60b318f3beb994f50d33645905f5096a53ce69acla6dbeb3de1
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 1
  ZygoClaim.claimToken confirmed   Block: 12   Gas used: 88820 (0.74%)

Transaction sent: 0x02d96b4f33f435f0158d4988bb0ae861225f53f4c6a85ef5530feac03fdd5c49
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  ZygoClaim.claimToken confirmed (reverted)   Block: 13   Gas used: 26967 (0.22%)

tests/test_claim.py::test_cannot_claim_twice PASSED
```

```
tests/test_claim.py::test_front_running_attack RUNNING
Transaction sent: 0xb2a8e87523790485ec6fc26efbea62cb6bbeb31ef0eaa3ab47d4d8419f051f7f
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 10
  ERC20Mock.constructor confirmed   Block: 14   Gas used: 523846 (4.37%)
  ERC20Mock deployed at: 0xb6286fAFd0451320ad6A8143089b216C2152c025

Transaction sent: 0x6d82d141d5bdb019ad06709dac1a84c79e43707b6a18fb749f99543781150604
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 11
  ZygoClaim.constructor confirmed   Block: 15   Gas used: 927111 (7.73%)
  ZygoClaim deployed at: 0x7a3d735ee6873f17Dbdcab1d51B604928dc10d92

Transaction sent: 0x789bb6e6e9e1da5d8c4e850fc91be2d58dc687566dff0b11ca8070783a6ebaa1
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 12
  ZygoClaim.transferOwnership confirmed   Block: 16   Gas used: 30141 (0.25%)

Transaction sent: 0x3681b39ec233c33e4ba1060b8924378016ad1ba37092045849bafda736111abd
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 13
  ERC20Mock.mint confirmed   Block: 17   Gas used: 65821 (0.55%)

Transaction sent: 0x6892e0132f75c5838d7d75702f6aefdef4f5ade78fe7fbccf642ecb73485100d
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 14
  ERC20Mock.transfer confirmed   Block: 18   Gas used: 51060 (0.43%)

Transaction sent: 0xe7e85b6bc3fc5d69e68d8a395edfe01cf56ec82b11532410868940b69f4260cf
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  ZygoClaim.claimToken confirmed   Block: 19   Gas used: 73820 (0.62%)

Transaction sent: 0x4e947605ba34d663a5b036a8bb2469b95ac1e2eca3252dec7b721a94255a7923
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 3
  ZygoClaim.claimToken confirmed (reverted)   Block: 20   Gas used: 26967 (0.22%)

tests/test_claim.py::test_front_running_attack PASSED
tests/test_claim.py::test_invalid_signature RUNNING
Transaction sent: 0x60e3a9203d797acb5d21019179ffcc0deacde13dd3d87263b370fce9a311ac16
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 15
  ERC20Mock.constructor confirmed   Block: 21   Gas used: 523846 (4.37%)
  ERC20Mock deployed at: 0x30375B532345B01cB8c2AD12541b09E9Aa53A93d

Transaction sent: 0x1524dea3f9b89aa36d2ea9cd9624422e6422c748c752bbc82dc09c8027cdab26
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 16
  ZygoClaim.constructor confirmed   Block: 22   Gas used: 927111 (7.73%)
  ZygoClaim deployed at: 0x26f15335BB1C6a4C0B660eDd694a0555A9F1cce3

Transaction sent: 0x3e57ec09fd33c5c4455b504aff8420bdd84ce51a2aac5465044bd283fad47daa
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 17
  ZygoClaim.transferOwnership confirmed   Block: 23   Gas used: 30141 (0.25%)

Transaction sent: 0x28dd673bd37afed35660bfd6e5b6a2925239f926a1abc6048ee67e1924d7ad1d
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 18
  ERC20Mock.mint confirmed   Block: 24   Gas used: 65821 (0.55%)

Transaction sent: 0x746a13fea8547c2cccb8591ce2c78fe9de92f728cdefea9bd9165ac627672a64
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 19
  ERC20Mock.transfer confirmed   Block: 25   Gas used: 51060 (0.43%)

Transaction sent: 0x0d21b7a940fb6b3fc0c1678c463d4a651be9f4574f2eb59c56bf484d0551f474
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 4
  ZygoClaim.claimToken confirmed (reverted)   Block: 26   Gas used: 34578 (0.29%)

tests/test_claim.py::test_invalid_signature PASSED
```

```
tests/test_claim.py::test_ignores_extra_signatures RUNNING
Transaction sent: 0x3438aa158f8cecccf2e718f22b1c1c3ebd8cf865c2476d0499652e47e51587e
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 20
  ERC20Mock.constructor confirmed   Block: 27   Gas used: 523846 (4.37%)
  ERC20Mock deployed at: 0xdCF93F11ef216cEC9C07fd31dD801c9b2b39Afb4

Transaction sent: 0x7f71d9f5170dc9b5f2011fad9f5acdefbf83e8eb14dfe0f4132da3ce117116e8
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 21
  ZygoClaim.constructor confirmed   Block: 28   Gas used: 927111 (7.73%)
  ZygoClaim deployed at: 0xBcb61491F1859f53438918F1A5aFCA542Af9D397

Transaction sent: 0x4002d931195c7ee4735ed5fefaefe46e553f538cf4701e04104da0d0038f3497
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 22
  ZygoClaim.transferOwnership confirmed   Block: 29   Gas used: 30141 (0.25%)

Transaction sent: 0x31fd108d1022613d3e41639ca35b764144fc21b5e8bad6f374ec3f9c53d4188d
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 23
  ERC20Mock.mint confirmed   Block: 30   Gas used: 65821 (0.55%)

Transaction sent: 0x1b639879c4d183204502a1b2abe9db0ab11432aecccd02a81fbf8c10496d37aa
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 24
  ERC20Mock.transfer confirmed   Block: 31   Gas used: 51060 (0.43%)

Transaction sent: 0xb9ec883cd13c6e4bc4e73c9adda53dfba2a5e6a09ed7d1b51c1681449c7c48af
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 5
  ZygoClaim.claimToken confirmed   Block: 32   Gas used: 75721 (0.63%)

Amounts length: 1
Signatures sent: 2
Balance increment: 100
tests/test_claim.py::test_ignores_extra_signatures PASSED
tests/test_claim.py::test_multiple_reward_tokens RUNNING
Transaction sent: 0xf75ae05de1a2951f92d01b7592c86dffaa6efa7bfbae6e0bbaacfc402ab20841
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 25
  ERC20Mock.constructor confirmed   Block: 33   Gas used: 523834 (4.37%)
  ERC20Mock deployed at: 0x654f70d8442EA18904FA1AD79114f7250F7E9336

Transaction sent: 0xdbb8b00f3cae1dc6535fe7a1ebceb2ad3095c36599f939dc95567daf6da81e6a
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 26
  ERC20Mock.constructor confirmed   Block: 34   Gas used: 523834 (4.37%)
  ERC20Mock deployed at: 0xADeD61D42dE86f9058386D1D0d739d20C7eAfC43

Transaction sent: 0x9962fc7b2a15d5a564ba9e30859267beb5f3b40d4f4d590dacdbe6ccff0d4369
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 27
  ZygoClaim.constructor confirmed   Block: 35   Gas used: 948552 (7.90%)
  ZygoClaim deployed at: 0x832698Daec363C9A7aB036C224Af5B21280b3AC6

Transaction sent: 0x97ead1ada7e4fc8a7bf9b133d5cfc3a42a83d0d899b1adf76b506057cf75f5a8
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 28
  ZygoClaim.transferOwnership confirmed   Block: 36   Gas used: 30141 (0.25%)

Transaction sent: 0xd907ea3992d0c03a32c9566e4b6507796054e8ced9943e2ff6dff0212bf4d1a1
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 29
  ERC20Mock.mint confirmed   Block: 37   Gas used: 65821 (0.55%)

Transaction sent: 0x761f54ef03ee1b763e329c0067937e6ffb82bd8b95c0ca6808b12bcc02c43edb
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 30
  ERC20Mock.mint confirmed   Block: 38   Gas used: 65821 (0.55%)

Transaction sent: 0xd4419363714555c573b62860fdc71c22d66e825edf4a1011710052b8d0eb4d965
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 31
  ERC20Mock.transfer confirmed   Block: 39   Gas used: 51060 (0.43%)

Transaction sent: 0xa9e0ad181f6cb62f17da5c1e7e27f1ad89b9b45840cd95223b27075b18d64865
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 32
  ERC20Mock.transfer confirmed   Block: 40   Gas used: 51060 (0.43%)

Transaction sent: 0x870cb8c10e06e569ea62bb5b8ea228343f10026148dd1cee3236a59fe2977e75
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 6
  ZygoClaim.claimToken confirmed   Block: 41   Gas used: 102487 (0.85%)

tests/test_claim.py::test_multiple_reward_tokens PASSED
```

```
tests/test_claim.py::test_only_owner_can_set_reward_tokens RUNNING
Transaction sent: 0x6c27c8275f3e92987189d334a23d5204ca6422848f3c44cc811fb1b43a8ceb73
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 33
  ERC20Mock.constructor confirmed   Block: 42   Gas used: 523834 (4.37%)
  ERC20Mock deployed at: 0x34b97ffa01dc0DC959c5f1176273D0de3be914C1

Transaction sent: 0x3f1bbabb563563b78602a03e1e46971bc507269338bf8eef6e60eb48677261d0
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 34
  ERC20Mock.constructor confirmed   Block: 43   Gas used: 523834 (4.37%)
  ERC20Mock deployed at: 0xc830Ad2FDfCC2f368fE5DeC93b1Dc72ecABb3691

Transaction sent: 0x65482004282e4e0761dd42c59ff0c7764b15c677bc6d7710ffbb0c10fe359a8e
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 35
  ZygoClaim.constructor confirmed   Block: 44   Gas used: 927111 (7.73%)
  ZygoClaim deployed at: 0xbc8eCccb89650c3E796e803CB009BF9b898CB359

Transaction sent: 0x8e7a2185d43c94d52ebdbce3f2d3217c04724ef52bd115cb1e16a85b5ee07373
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 36
  ZygoClaim.setRewardTokens confirmed   Block: 45   Gas used: 52576 (0.44%)

Transaction sent: 0x048006b2f55bbbeb89bd7ba408f3a371edbb84ce2dde5738d786c781b3d1bf60
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 7
  ZygoClaim.setRewardTokens confirmed (reverted)   Block: 46   Gas used: 23393 (0.19%)

tests/test_claim.py::test_only_owner_can_set_reward_tokens PASSED
tests/test_claim.py::test_set_reward_tokens_to_empty_array RUNNING
Transaction sent: 0x47e83b1b905c830383ad02cfa835bf963f0d0371418085df0e0680991befa2f9
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 37
  ERC20Mock.constructor confirmed   Block: 47   Gas used: 523834 (4.37%)
  ERC20Mock deployed at: 0x4B0FccF53589c1F185B35db88bB315a0bBF9a3e0

Transaction sent: 0xc78953d99100cb24414d63b66c070ea48c76739a9d6d77ef9083aa4aec0aa164
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 38
  ZygoClaim.constructor confirmed   Block: 48   Gas used: 927111 (7.73%)
  ZygoClaim deployed at: 0xFE0F4Cf81B5c0a6Fd65a610FD9488F33aE9095cB

Transaction sent: 0x9e957596e3ab80fe4857847a831eaaeb7c8b54ef79b4d05e76d24c92a5277df3
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 39
  ZygoClaim.setRewardTokens confirmed   Block: 49   Gas used: 16935 (0.14%)

tests/test_claim.py::test_set_reward_tokens_to_empty_array PASSED
tests/test_claim.py::test_multiple_set_reward_tokens RUNNING
Transaction sent: 0x4dbaa90260c253c9baaee3543f6c4a54cf5d1db7357264a0821aeea2b1ece299
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 40
  ERC20Mock.constructor confirmed   Block: 50   Gas used: 523834 (4.37%)
  ERC20Mock deployed at: 0x0AC45e945A008D3fc19da8f591be8601C1F63130

Transaction sent: 0x040b3af6df97e8ae9872e2d87fe9555ed663311beea3538006630ccc9fc329cb
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 41
  ERC20Mock.constructor confirmed   Block: 51   Gas used: 523834 (4.37%)
  ERC20Mock deployed at: 0x5847798CE8c89e3Fff59AE5fA30BEC0d406b5687

Transaction sent: 0x928f5a984961a54fb8ae23a711d72b19ea556cb5f405ec066a939d98e08f0ef0
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 42
  ERC20Mock.constructor confirmed   Block: 52   Gas used: 523834 (4.37%)
  ERC20Mock deployed at: 0x0C60536783db9ED5A2B216970B10FF2243d317dD

Transaction sent: 0xec9ddd2e1501ad7d1c01094b2b881d90c1dc4f0a21a5a574aff857bc7b33e6be
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 43
  ZygoClaim.constructor confirmed   Block: 53   Gas used: 927099 (7.73%)
  ZygoClaim deployed at: 0x0e19E87b363DD7a3af5c45C95ab6885367251B94

Transaction sent: 0xa1d587bc0fff637986bded684d22f7430799ed3cbb6c4d4b137c868365573109
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 44
  ZygoClaim.setRewardTokens confirmed   Block: 54   Gas used: 52564 (0.44%)

Transaction sent: 0x9abf87ac98d1c385def5ff2cde6079e60da104194d4693d5175ff0904e4c7148
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 45
  ZygoClaim.setRewardTokens confirmed   Block: 55   Gas used: 37576 (0.31%)

tests/test_claim.py::test_multiple_set_reward_tokens PASSED
```

Annexes

Testing code:

Claim:

```
from brownie import (
    reverts,
)
from eth_account import Account
from eth_account.messages import encode_defunct
from web3 import Web3

from scripts.helpful_scripts import (
    ZERO_ADDRESS,
    DAY_TIMESTAMP,
    get_account,
    get_timestamp,
    get_chain_number,
    increase_timestamp
)

from scripts.deploy import (
    deploy_erc,
    deploy_claim
)

backend = Account.create()
owner_address = backend.address

def sign_claim(device_id, amount):
    message_hash = Web3.solidityKeccak(
        ["string", "uint256"],
        [device_id, amount]
    )
    message = encode_defunct(primitive=message_hash)
    signed = backend.sign_message(message)
```

```python
    return signed.signature

def test_claim_success(only_local):
    # Arrange
    owner = get_account(0)
    other = get_account(1)
    extra = get_account(2)

    reward_token = deploy_erc(owner, "Reward", "RWD")
    claim = deploy_claim(owner, reward_token.address)

    claim.transferOwnership(owner_address, {"from": owner})

    # mint some tokens
    reward_token.mint(owner, 1e18)

    device_id = "device123"
    amount = 100

    reward_token.transfer(claim.address, amount, {"from": owner})
    signature = sign_claim(device_id, amount)
    claim.claimToken(
        device_id,
        [signature],
        [amount],
        {"from": other}
    )
    assert reward_token.balanceOf(other) == amount

def test_cannot_claim_twice(only_local):
    owner = get_account(0)
    other = get_account(1)

    reward_token = deploy_erc(owner, "Reward", "RWD")
    claim = deploy_claim(owner, reward_token.address)
    claim.transferOwnership(owner_address, {"from": owner})

    reward_token.mint(owner, 2e18)

    device_id = "device123"
    amount = 100

    reward_token.transfer(claim.address, amount * 2, {"from": owner})
    signature = sign_claim(device_id, amount)
```

```
    # SUCCESS
    claim.claimToken(device_id, [signature], [amount], {"from": other})
    assert reward_token.balanceOf(other) == amount

    # FAIL
    with reverts("ETokensAlreadyClaimed: "):
        claim.claimToken(device_id, [signature], [amount], {"from": other})

def test_front_running_attack(only_local):
    owner = get_account(0)
    victim = get_account(1)
    attacker = get_account(2)

    reward_token = deploy_erc(owner, "Reward", "RWD")
    claim = deploy_claim(owner, reward_token.address)
    claim.transferOwnership(owner_address, {"from": owner})

    reward_token.mint(owner, 1e18)

    device_id = "device123"
    amount = 100
    reward_token.transfer(claim.address, amount, {"from": owner})

    signature = sign_claim(device_id, amount)

    # Attacker
    claim.claimToken(device_id, [signature], [amount], {"from": attacker})
    assert reward_token.balanceOf(attacker) == amount

    # Victim FAIL
    with reverts("ETokensAlreadyClaimed: "):
        claim.claimToken(device_id, [signature], [amount], {"from": victim})

def test_invalid_signature(only_local):
    owner = get_account(0)
    other = get_account(1)

    reward_token = deploy_erc(owner, "Reward", "RWD")
    claim = deploy_claim(owner, reward_token.address)
    claim.transferOwnership(owner_address, {"from": owner})

    reward_token.mint(owner, 1e18)

    device_id = "device123"
    amount = 100
```

```
    reward_token.transfer(claim.address, amount, {"from": owner})

    # Invalid sign
    fake_backend = Account.create()
    message_hash = Web3.solidityKeccak(
        ["string", "uint256"],
        [device_id, amount]
    )
    message = encode_defunct(primitive=message_hash)
    fake_signature = fake_backend.sign_message(message).signature

    with reverts("EInvalidAdminSignature: "):
        claim.claimToken(device_id, [fake_signature], [amount], {"from": other})


def test_ignores_extra_signatures(only_local):
    # Arrange
    owner = get_account(0)
    claimer = get_account(1)

    # Deploy reward token and claim contract
    reward_token = deploy_erc(owner, "Reward", "RWD")
    claim = deploy_claim(owner, reward_token.address)

    # Transfer ownership to independent backend signer
    claim.transferOwnership(owner_address, {"from": owner})

    reward_token.mint(owner, 1e18)

    device_id = "device123"
    amounts = [100]  # Only one amount
    reward_token.transfer(claim.address, amounts[0], {"from": owner})

    # Create two signatures but only one amount
    signatures = [
        sign_claim(device_id, 100),
        sign_claim(device_id, 200)  # Extra signature that should be ignored
    ]

    # Act
    balance_before = reward_token.balanceOf(claimer)
    claim.claimToken(device_id, signatures, amounts, {"from": claimer})
    balance_after = reward_token.balanceOf(claimer)

    # Assert
```

```
    print("Amounts length:", len(amounts))
    print("Signatures sent:", len(signatures))
    print("Balance increment:", balance_after - balance_before)

    assert balance_after - balance_before == amounts[0], "Extra signature should be
ignored"

def test_multiple_reward_tokens(only_local):
    owner = get_account(0)
    other = get_account(1)

    r1 = deploy_erc(owner, "Token1", "T1")
    r2 = deploy_erc(owner, "Token2", "T2")

    claim = deploy_claim(owner, [r1.address, r2.address])
    claim.transferOwnership(owner_address, {"from": owner})

    r1.mint(owner, 1e18)
    r2.mint(owner, 1e18)

    r1.transfer(claim.address, 100, {"from": owner})
    r2.transfer(claim.address, 200, {"from": owner})

    device_id = "device_multi"
    amounts = [100, 200]

    signatures = [sign_claim(device_id, amt) for amt in amounts]

    claim.claimToken(device_id, signatures, amounts, {"from": other})

    assert r1.balanceOf(other) == 100
    assert r2.balanceOf(other) == 200

def test_only_owner_can_set_reward_tokens(only_local):
    owner = get_account(0)
    other = get_account(1)

    reward_token1 = deploy_erc(owner, "Token1", "T1")
    reward_token2 = deploy_erc(owner, "Token2", "T2")

    claim = deploy_claim(owner, reward_token1.address)

    # Owner updates the reward tokens
    claim.setRewardTokens([reward_token1.address, reward_token2.address], {"from": owner})
    assert claim.rewardTokens(0) == reward_token1.address
```

```
    assert claim.rewardTokens(1) == reward_token2.address

    # Non-owner tries to update reward tokens
    with reverts():
        claim.setRewardTokens([reward_token1.address], {"from": other})

def test_set_reward_tokens_to_empty_array(only_local):
    owner = get_account(0)

    reward_token = deploy_erc(owner, "Token1", "T1")
    claim = deploy_claim(owner, reward_token.address)

    # Set reward tokens to empty array
    claim.setRewardTokens([], {"from": owner})

def test_multiple_set_reward_tokens(only_local):
    owner = get_account(0)

    r1 = deploy_erc(owner, "Token1", "T1")
    r2 = deploy_erc(owner, "Token2", "T2")
    r3 = deploy_erc(owner, "Token3", "T3")

    claim = deploy_claim(owner, r1.address)

    claim.setRewardTokens([r1.address, r2.address], {"from": owner})
    assert claim.rewardTokens(0) == r1.address
    assert claim.rewardTokens(1) == r2.address

    claim.setRewardTokens([r2.address, r3.address], {"from": owner})
    assert claim.rewardTokens(0) == r2.address
    assert claim.rewardTokens(1) == r3.address
```

Router:

```
from brownie import (
    reverts,
    Wei
)

from scripts.helpful_scripts import (
    ZERO_ADDRESS,
    DAY_TIMESTAMP,
    get_account,
    get_timestamp,
    get_chain_number,
    increase_timestamp
)

from scripts.deploy import (
    deploy_erc,
    deploy_weth,
    deploy_router,
    deploy_mock_router,
)

def test_swap_empty_path(only_local):
    owner = get_account(0)
    claimer = get_account(1)
    fee_wallet = get_account(2)

    fee_bps = 500  # 5% fee
    wrapped_token = deploy_weth(owner)
    zygo_router = deploy_router(owner, fee_bps, wrapped_token.address, fee_wallet.address)

    with reverts("EEmptySwapPath: "):
        zygo_router.swap([], 100, 50, {"from": claimer})

def test_swap_same_token(only_local):
    owner = get_account(0)
    claimer = get_account(1)
    fee_wallet = get_account(2)
```

```
    fee_bps = 500
    wrapped_token = deploy_weth(owner)
    zygo_router = deploy_router(owner, fee_bps, wrapped_token.address, fee_wallet.address)

    token = deploy_erc(owner, "Token", "TKN")
    hop = make_hop(token, token, 0, get_account(3))
    hops = [hop]

    with reverts("EFromAndToTokenCantBeSame: "):
        zygo_router.swap(hops, 100, 50, {"from": claimer})

def _test_swap_router_not_whitelisted(only_local):
    owner = get_account(0)
    claimer = get_account(1)
    fee_wallet = get_account(2)

    fee_bps = 500
    wrapped_token = deploy_weth(owner)
    zygo_router = deploy_router(owner, fee_bps, wrapped_token.address, fee_wallet.address)

    # deploy a mock router
    mock_router = deploy_mock_router(owner)

    token_in = deploy_erc(owner, "Token1", "T1")
    token_out = deploy_erc(owner, "Token2", "T2")

    hop = make_hop(token_in, token_out, 0, mock_router)  # use mock router
    hops = [hop]

    token_in.mint(claimer, 1000)
    token_in.approve(zygo_router.address, 1000, {"from": claimer})

    # Do NOT whitelist the router → should revert
    with reverts("EInvalidRouter: "):
        zygo_router.swap(hops, 100, 50, {"from": claimer})

def _test_swap_fee_erc20(only_local):
    owner = get_account(0)
    claimer = get_account(1)
    fee_wallet = get_account(2)

    fee_bps = 500  # 5%
    wrapped_token = deploy_erc(owner, "WrappedETH", "WETH")
    zygo_router = deploy_router(owner, fee_bps, wrapped_token.address, fee_wallet.address)
```

```
    token_in = deploy_erc(owner, "Token1", "T1")
    token_out = deploy_erc(owner, "Token2", "T2")
    mock_router = deploy_mock_router(owner)

    zygo_router.whitelistRouter(mock_router.address, {"from": owner})

    hop = make_hop(token_in, token_out, 0, mock_router)
    hops = [hop]

    # Input amount + fee
    input_amount = 1000
    fee = input_amount * fee_bps // 10000

    # Mint token_in to claimer and approval
    token_in.mint(claimer, input_amount + fee)
    token_in.approve(zygo_router.address, input_amount + fee, {"from": claimer})

    # Mint token_out to router mock to swap
    token_out.mint(mock_router.address, input_amount)

    fee_balance_before = token_in.balanceOf(fee_wallet)
    claimer_balance_before = token_out.balanceOf(claimer)

    # Swap
    zygo_router.swap(hops, input_amount, input_amount - fee, {"from": claimer})

    fee_balance_after = token_in.balanceOf(fee_wallet)
    claimer_balance_after = token_out.balanceOf(claimer)

    # Validate fee
    assert fee_balance_after - fee_balance_before == fee
    # Validate the claimer
    assert claimer_balance_after - claimer_balance_before == input_amount - fee

def test_swap_slippage_exceeded(only_local):
    owner = get_account(0)
    claimer = get_account(1)
    fee_wallet = get_account(2)

    fee_bps = 500  # 5%
    wrapped_token = deploy_weth(owner)
    zygo_router = deploy_router(owner, fee_bps, wrapped_token.address, fee_wallet.address)

    token_in = deploy_erc(owner, "Token1", "T1")
```

```
    token_out = deploy_erc(owner, "Token2", "T2")
    mock_router = deploy_mock_router(owner)
    zygo_router.whitelistRouter(mock_router.address, {"from": owner})

    hop = make_hop(token_in, token_out, 0, mock_router)
    hops = [hop]

    # ----------- Input amount and fee -----------
    input_amount = 100
    fee = input_amount * fee_bps // 10000  # 5% = 5
    total_approval = input_amount + fee

    # ----------- Mint token_in to claimer and approve ZygoRouter -----------
    token_in.mint(claimer, total_approval)
    token_in.approve(zygo_router.address, total_approval, {"from": claimer})

    # ----------- Mint token_out in mock router for swap -----------
    # Must cover currentAmount = input_amount
    token_out.mint(mock_router.address, input_amount)

    # ----------- Prepare minAmountOut deliberately higher to trigger slippage -----------
    min_amount_out = input_amount + 1  # higher than what the mock can deliver

    # ----------- Record balances before -----------
    fee_balance_before = token_in.balanceOf(fee_wallet)
    claimer_balance_before = token_out.balanceOf(claimer)

    # ----------- Execute swap and expect revert due to slippage -----------
    with reverts("ESlippageExceeded: "):
        zygo_router.swap(hops, input_amount, min_amount_out, {"from": claimer})

    # Fee should NOT be taken since swap reverted
    fee_balance_after = token_in.balanceOf(fee_wallet)
    assert fee_balance_after - fee_balance_before == 0

    # ----------- Claimer should not receive token_out since swap reverted -----------
    claimer_balance_after = token_out.balanceOf(claimer)
    assert claimer_balance_after - claimer_balance_before == 0


def test_swap_fee_v3(only_local):
    owner = get_account(0)
    claimer = get_account(1)
    fee_wallet = get_account(2)
```

```
    fee_bps = 500
    wrapped_token = deploy_weth(owner)
    zygo_router = deploy_router(owner, fee_bps, wrapped_token.address, fee_wallet.address)

    token_in = deploy_erc(owner, "Token1", "T1")
    token_out = deploy_erc(owner, "Token2", "T2")
    mock_router = deploy_mock_router(owner)

    zygo_router.whitelistRouter(mock_router.address, {"from": owner})

    hop = make_hop(token_in, token_out, 1, mock_router)  # V3
    hops = [hop]

    input_amount = 1000
    fee = input_amount * fee_bps // 10000

    token_in.mint(claimer, input_amount + fee)
    token_in.approve(zygo_router.address, input_amount + fee, {"from": claimer})

    # Mint token_out in mock to cover swap
    token_out.mint(mock_router.address, input_amount)

    fee_balance_before = token_in.balanceOf(fee_wallet)
    claimer_balance_before = token_out.balanceOf(claimer)

    zygo_router.swap(hops, input_amount, input_amount - fee, {"from": claimer})

    fee_balance_after = token_in.balanceOf(fee_wallet)
    claimer_balance_after = token_out.balanceOf(claimer)

    assert fee_balance_after - fee_balance_before == fee
    assert claimer_balance_after - claimer_balance_before == input_amount - fee

def test_swap_eth_to_erc20(only_local):
    owner = get_account(0)
    claimer = get_account(1)
    fee_wallet = get_account(2)

    fee_bps = 500
    wrapped_token = deploy_weth(owner)
    zygo_router = deploy_router(owner, fee_bps, wrapped_token.address, fee_wallet.address)

    token_out = deploy_erc(owner, "Token1", "T1")
    mock_router = deploy_mock_router(owner)
```

```
    zygo_router.whitelistRouter(mock_router.address, {"from": owner})

    hop = make_hop(wrapped_token, token_out, 0, mock_router)  # V2 ETH swap
    hops = [hop]

    input_amount = 100
    fee = input_amount * fee_bps // 10000

    # Mint ETH to claimer
    owner.transfer(claimer, Wei(f"{input_amount + fee} ether"))

    # Mint token_out to mock to simulate swap
    token_out.mint(mock_router.address, input_amount)

    fee_balance_before = wrapped_token.balanceOf(fee_wallet)
    claimer_balance_before = token_out.balanceOf(claimer)

    zygo_router.swap(
        hops,
        input_amount,
        input_amount - fee,
        {"from": claimer, "value": input_amount}
    )

    claimer_balance_after = token_out.balanceOf(claimer)
    assert claimer_balance_after - claimer_balance_before == input_amount - fee

#-------------------------------------
# Helper: Create a SwapHop dict usable in tests
#-------------------------------------

# SwapHop = (address tokenIn, address tokenOut, uint24 swapType, uint24 fee, address
router, bool hasDeadline)
def make_hop(token_in, token_out, swap_type, router_address, fee=3000, has_deadline=True):
    return (
        token_in.address,
        token_out.address,
        swap_type,       # 0 = V2, 1 = V3
        fee,
        router_address.address if hasattr(router_address, "address") else router_address,
        has_deadline
    )

#-------------------------------------
# Mock Router V2
```

```python
#-------------------------------------
class RouterMockV2:
    def __init__(self):
        self.swap_called = False

    def swapExactTokensForTokens(self, amountIn, amountOutMin, path, to, deadline):
        self.swap_called = True
        # simulate output: returns amounts array
        # simple: output = amountIn * 95% to simulate fee/slippage
        amountOut = amountIn * 95 // 100
        return [amountIn, amountOut]



#-------------------------------------
# Mock Router V3
#-------------------------------------
class RouterMockV3:
    def __init__(self):
        self.swap_called = False

    def exactInputSingle(self, params):
        self.swap_called = True
        # output = 98% of input to simulate swap
        return params.amountIn * 98 // 100
```

Technical Findings Summary

Findings

Vulnerability Level	Total	Pending	Not Apply	Acknowledged	Partially Fixed	Fixed
High	4					
Medium	1					
Low	1					
Informational	1					

Assessment Results

Score Results

Review	Score
Global Score	**70/100**
Assure KYC	Not completed
Audit Score	70/100

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum score is 100, however to attain that value the project must pass and provide all the data needed for the assessment. Our Passing Score has been changed to 84 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below. The Global Score is a combination of the evaluations obtained between having or not having KYC and the type of contract audited together with its manual audit.

Audit FAIL

Following our comprehensive security audit of the smart contract for the **ZYGADEX** project, the project did not fulfill the necessary criteria required to pass the security audit.

Disclaimer

Assure Defi has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocating for the Project, and users relying on this report should not consider this as having any merit for financial Token in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies. All information provided in this report does not constitute financial or investment in Token, nor should it be used to signal that any person reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided ‘as is, and Assure Defi is under no covenant to audit completeness, accuracy, or solidity of the contracts. In no event will Assure Defi or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report. The assessment of Tokens provided by Assure Defi are subject to dependencies and are under continuing development. You agree that your access or use, including but not limited to any Tokens, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The Token may access, and depend upon, multiple layers of third parties.