CAIRO SECURITY CLAN

ATOMIQ EXCHANGE

SECURITY ASSESEMENT REPORT

APRIL 2025

Prepared for

ATOMIQ LABS

Contents

1 About Cairo Security Clan

Cairo Security Clan is a leading force in the realm of blockchain security, dedicated to fortifying the foundations of the digital age. As pioneers in the field, we specialize in conducting meticulous smart contract security audits, ensuring the integrity and reliability of decentralized applications built on blockchain technology.

At Cairo Security Clan, we boast a multidisciplinary team of seasoned professionals proficient in blockchain security, cryptography, and software engineering. With a firm commitment to excellence, our experts delve into every aspect of the Web3 ecosystem, from foundational layer protocols to application-layer development. Our comprehensive suite of services encompasses smart contract audits, formal verification, and real-time monitoring, offering unparalleled protection against potential vulnerabilities.

Our team comprises industry veterans and scholars with extensive academic backgrounds and practical experience. Armed with advanced methodologies and cutting-edge tools, we scrutinize and analyze complex smart contracts with precision and rigor. Our track record speaks volumes, with a plethora of published research papers and citations, demonstrating our unwavering dedication to advancing the field of blockchain security.

At Cairo Security Clan, we prioritize collaboration and transparency, fostering meaningful partnerships with our clients. We believe in a customer-oriented approach, engaging stakeholders at every stage of the auditing process. By maintaining open lines of communication and soliciting client feedback, we ensure that our solutions are tailored to meet the unique needs and objectives of each project.

Beyond our core services, Cairo Security Clan is committed to driving innovation and shaping the future of blockchain technology. As active contributors to the ecosystem, we participate in the development of emerging technologies such as Starknet, leveraging our expertise to build robust infrastructure and tools. Through strategic guidance and support, we empower our partners to navigate the complexities of the blockchain landscape with confidence and clarity.

In summary, Cairo Security Clan stands at the forefront of blockchain security, blending technical prowess with a client-centric ethos to deliver unparalleled protection and peace of mind in an ever-evolving digital landscape. Join us in safeguarding the future of decentralized finance and digital assets with confidence and conviction.

2 Disclaimer

Disclaimer Limitations of this Audit:

This report is based solely on the materials and documentation provided by you to Cairo Security Clan for the specific purpose of conducting the security review outlined in the Summary of Audit and Scoped Files. The findings presented here may not be exhaustive and may not identify all potential vulnerabilities. Cairo Security Clan provides this review and report on an "as-is" and "as-available" basis. You acknowledge that your use of this report, including any associated services, products, protocols, platforms, content, and materials, occurs entirely at your own risk.

Inherent Risks of Blockchain Technology:

Blockchain technology remains in its developmental stage and is inherently susceptible to unknown risks and vulnerabilities. This review is specifically focused on the smart contract code and does not extend to the compiler layer, programming language elements beyond the reviewed code, or other potential security risks outside the code itself.

Report Purpose and Reliance:

This report should not be construed as an endorsement of any specific project or team, nor does it guarantee the absolute security of the audited smart contracts. No third party should rely on this report for any purpose, including making investment or purchasing decisions.

Liability Disclaimer:

To the fullest extent permitted by law, Cairo Security Clan disclaims all liability associated with this report, its contents, and any related services and products arising from your use. This includes, but is not limited to, implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Third-Party Products and Services:

Cairo Security Clan does not warrant, endorse, guarantee, or assume responsibility for any products or services advertised by third parties within this report, nor for any open-source or third-party software, code, libraries, materials, or information linked to, referenced by, or accessible through this report, its content, and related services and products. This includes any hyperlinked websites, websites or applications appearing on advertisements, and Cairo Security Clan will not be responsible for monitoring any transactions between you and third-party providers. It is recommended that you exercise due diligence and caution when considering any third-party products or services, just as you would with any purchase or service through any medium.

Disclaimer of Advice:

FOR THE AVOIDANCE OF DOUBT, THIS REPORT, ITS CONTENT, ACCESS, AND/OR USE, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHOULD NOT BE CONSIDERED OR RELIED UPON AS FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER PROFESSIONAL ADVICE.

3 Executive Summary

This document presents the security review performed by Cairo Security Clan on the Atomiq Exchange.

atomiq.exchange is a fully trustless cross-chain decentralized exchange (DEX) allowing you to swap between smart chains and Bitcoin, without having to trust any intermediary in the process.

All transactions are processed atomically with strong security guarantees based on bitcoin light client (leveraging bitcoin's proof-of-work security) & submarine swaps (leveraging HTLCs - hash-time locked contracts over bitcoin's lightning network). With this approach atomiq.exchange is able to offer security guarantees far exceeding those of existing bridging or cross-chain swapping solutions. Learn more from docs.

The audit was performed using

- manual analysis of the codebase,
- automated analysis tools,
- simulation of the smart contract,
- analysis of edge test cases

9 points of attention, where 0 is classified as Critical, 3 are classified as High, 0 is classified as Medium,0 is classified as Low,1 is classified as Informational and 5 are classified as Best Practices. The issues are summarized in Fig. 1.

This document is organized as follows. Section 1 About Cairo Security Clan. Section 2 Disclaimer. Section 3 Executive Summary. Section 4 Summary of Audit. Section 5 Risk Classification. Section 6 Issues by Severity Levels. Section 7 Test Evaluation.

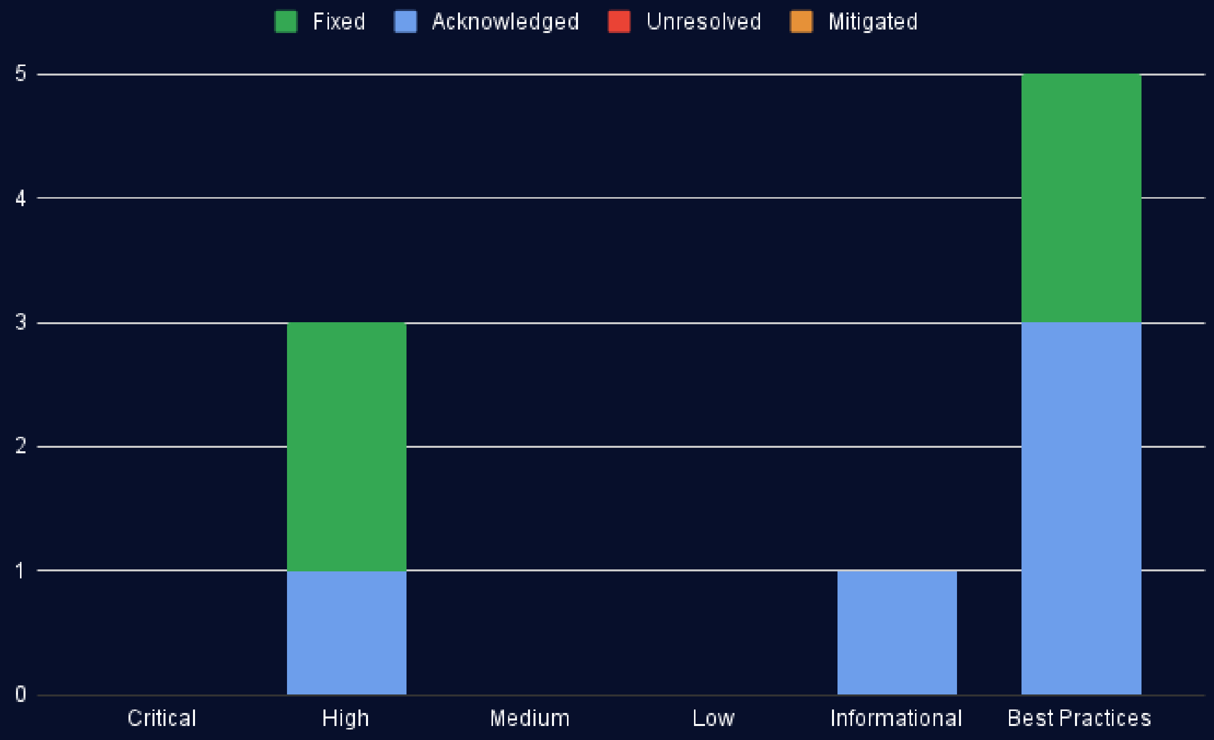

Fig 1: Distribution of issues: Critical (0), **High** (3), **Medium** (0), **Low** (0), **Informational** (1), **Best Practices** (5). **Distribution of status: Fixed** (4), **Acknowledged** (5), **Mitigated** (0), **Unresolved** (0).

4 Summary of Audit

Audit Type	Security Review
Cairo Version	2.10.1
Final Report	07/04/2025
Repository	atomiqlabs/atomiq-contracts-starknet
Initial Commit Hash	7150468429472f8ce246a51b764a5293e723f499
Final Commit Hash	50596c81bae561a1ebc161c362221800d7429396
Documentation	Website documentation
Test Suite Assessment	High

4.1 Scoped Files

	Contracts
1	packages/btc_nonced_output_claim_handler/src/lib.cairo
2	packages/btc_output_claim_handler/src/lib.cairo
3	packages/btc_relay/src/constants.cairo
4	packages/btc_relay/src/lib.cairo
5	packages/btc_relay/src/state.cairo
6	packages/btc_relay/src/structs.cairo
7	packages/btc_relay/src/utils.cairo
8	packages/btc_relay/src/state/fork.cairo
9	packages/btc_relay/src/structs/blockheader.cairo
10	packages/btc_relay/src/structs/stored_blockheader.cairo
11	packages/btc_relay/src/utils/difficulty.cairo
12	packages/btc_relay/src/utils/endianness.cairo
13	packages/btc_relay/src/utils/nbits.cairo
14	packages/btc_relay/src/utils/u256_utils.cairo
15	packages/btc_txid_claim_handler/src/lib.cairo
16	packages/btc_utils/src/bitcoin_merkle_tree.cairo
17	packages/btc_utils/src/bitcoin_tx.cairo
18	packages/btc_utils/src/byte_array.cairo
19	packages/btc_utils/src/compact_size.cairo
20	packages/btc_utils/src/lib.cairo
21	packages/common/src/handlers.cairo
22	packages/common/src/lib.cairo
23	packages/common/src/handlers/claim.cairo
24	packages/common/src/handlers/refund.cairo
25	packages/erc20_utils/src/lib.cairo
26	packages/escrow_manager/src/components.cairo
27	packages/escrow_manager/src/events.cairo
28	packages/escrow_manager/src/lib.cairo
29	packages/escrow_manager/src/sighash.cairo
30	packages/escrow_manager/src/state.cairo
31	packages/escrow_manager/src/structs.cairo
32	packages/escrow_manager/src/utils.cairo
33	packages/escrow_manager/src/components/escrow_storage.cairo
34	packages/escrow_manager/src/components/lp_vault.cairo
35	packages/escrow_manager/src/components/reputation.cairo
36	packages/escrow_manager/src/state/escrow.cairo
37	packages/escrow_manager/src/state/reputation.cairo
38	packages/escrow_manager/src/structs/escrow.cairo
39	packages/escrow_manager/src/utils/snip6.cairo
40	packages/hashlock_claim_handler/src/lib.cairo
41	packages/timelock_refund_handler/src/lib.cairo
42	packages/execution_contract/src/events.cairo
43	packages/execution_contract/src/execution_proxy.cairo
44	packages/execution_contract/src/lib.cairo
45	packages/execution_contract/src/state.cairo
46	packages/execution_contract/src/structs.cairo
47	packages/execution_contract/src/utils.cairo
48	packages/spv_swap_vault/src/events.cairo
49	packages/spv_swap_vault/src/lib.cairo
50	packages/spv_swap_vault/src/state.cairo
51	packages/spv_swap_vault/src/structs.cairo
52	packages/spv_swap_vault/src/utils.cairo

4.2 Issues

	Findings	Severity	Update
1	Invalid long fork could be merged	High	Fixed
2	Chain re-org could occur when a long fork is constructed	High	Fixed
3	Fronting Unconfirmed Bitcoin Transactions Can Cause Fronters to Lose Funds	High	Acknowledged
4	Lack of support camel-case naming ERC20 tokens	Informational	Acknowledged
5	Variable shadowing	Best Practices	Fixed
6	`claim_data` is emitted instead of `refund_data` even in `refund()` function	Best Practices	Acknowledged
7	Same swap cannot be executed twice	Best Practices	Acknowledged
8	Unused input parameter `extra_data`	Best Practices	Acknowledged
9	Unnecessary Caller Check	Best Practices	Fixed

5 Risk Classification

The risk rating methodology used by Cairo Security Clan follows the principles established by the CVSS risk rating methodology. The severity of each finding is determined by two factors: **Likelihood** and **Impact**.

Likelihood measures how likely an attacker will uncover and exploit the finding. This factor will be one of the following values:

a) **High**: The issue is trivial to exploit and has no specific conditions that need to be met;

b) **Medium**: The issue is moderately complex and may have some conditions that need to be met;

c) **Low**: The issue is very complex and requires very specific conditions to be met.

When defining the likelihood of a finding, other factors are also considered. These can include but are not limited to Motive, opportunity, exploit accessibility, ease of discovery, and ease of exploit.

Impact is a measure of the damage that may be caused if an attacker exploits the finding. This factor will be one of the following values:

a) **High**: The issue can cause significant damage such as loss of funds or the protocol entering an unrecoverable state;

b) **Medium**: The issue can cause moderate damage such as impacts that only affect a small group of users or only a particular part of the protocol;

c) **Low**: The issue can cause little to no damage such as bugs that are easily recoverable or cause unexpected interactions that cause minor inconveniences.

When defining the impact of a finding other factors are also considered. These can include but are not limited to Data/state integrity, loss of availability, financial loss, and reputation damage. After defining the likelihood and impact of an issue, the severity can be determined according to the table below.

		Likelihood		
		High	**Medium**	**Low**
Impact	**High**	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Info/Best Practices

To address issues that do not fit a High/Medium/Low severity, Cairo Security Clan also uses three more finding severities: **Informational**, **Best Practices** and **Gas**

a) **Informational** findings do not pose any risk to the application, but they carry some information that the audit team intends to formally pass to the client;

b) **Best Practice** findings are used when some piece of code does not conform with smart contract development best practices;

c) **Gas** findings are used when some piece of code uses more gas than it should be or have some functions that can be removed to save gas.

6 Issues by Severity Levels

6.1 High

6.1.1 Invalid Long Fork Could Be Merged

File(s): packages/btc_relay/src/lib.cairo

Description: In the `btc_relay` package, when a fork occurs on the Bitcoin main chain, the program allows anyone to submit fork block headers. These headers are automatically added to the main chain if their chain work surpasses that of the current main chain. For large forks (≥ 25 blocks), block headers are submitted across multiple transactions. The block headers are temporarily stored on-chain, and when the fork's chain work exceeds that of the main chain, the headers are moved to the main chain state.

Using multiple transactions, it is possible to modify the committed blocks of the fork chain. However, the current implementation does not remove old committed blocks from storage, nor does it have any checks to prevent these blocks from being used to add more blocks to the chain.

Consider the scenario:

1. Alice starts a long fork: $B_1 \rightarrow B_2 \rightarrow B_3 \rightarrow B_4$.
2. Alice modifies blocks 2 and 3 of her fork. The storage now records: $B_1 \rightarrow B_2' \rightarrow B_3' \rightarrow B_4$.
3. Alice continues to add more blocks after block 4. The storage now records: $B_1 \rightarrow B_2' \rightarrow B_3' \rightarrow B_4 \rightarrow B_5$. This long fork eventually overtakes the main chain.

As observed, the transition between $B_3' \rightarrow B_4$ is invalid. This occurs because B_4 was not removed in step 2, when Alice only modified blocks 2 and 3. As a result, B_4 retains the old data, allowing Alice to add block B_5, even though B_4 should have already been removed.

Recommendation(s): Consider maintaining the fork block height for long forks and checking if the `stored_header.block_height` is smaller or equal to the current fork height. This would help prevent invalid block transitions from occurring.

Status: Fixed

Update from the client: Fixed in this commit.

6.1.2 Chain Reorg Could Occur When a Long Fork Is Constructed

File(s): packages/btc_relay/src/lib.cairo

Description: In the `btc_relay` package, when a fork occurs on the Bitcoin main chain, the program allows anyone to submit fork block headers. These headers will automatically become part of the main chain if their chain work surpasses that of the current main chain. For large forks (≥ 25 blocks), block headers are submitted across multiple transactions. The block headers are temporarily stored on-chain, and when the fork's chain work exceeds the main chain's chain work, the headers are moved to the main chain state.

This process is handled by the function `submit_fork_blockheaders()`. In the first transaction, the program verifies that the stored header is committed to the main chain and records the `fork_start_blockheight` value. For subsequent transactions, it only verifies that the stored header is committed to the fork chain.

```
let caller = get_caller_address();
let fork_ptr = self.forks.entry(caller).entry(fork_id);

let mut fork_start_blockheight = fork_ptr.start_height.read();

if fork_start_blockheight == 0 {
    // Verify stored header is committed in the main chain
    self.verify_blockheader(stored_header);
    fork_start_blockheight = stored_header.block_height.into() + 1;
    fork_ptr.start_height.write(fork_start_blockheight);
} else {
    // Verify stored header is committed in the fork chain
    assert(fork_ptr.chain.entry(stored_header.block_height.into()).read() == stored_header.get_hash(), 'fork:
     fork block commitment');
}
```

Later, the code checks if the fork's chain work is greater than the main chain's work. If it is, the fork overtakes the main chain. However, the process does not verify whether the block at `fork_start_blockheight` is still committed to the main chain. If the main chain has been overtaken by a different fork, this block may have already been replaced.

```
// Check if this fork's chainwork is higher than main chainwork
if self.main_chainwork.read().into() < _stored_header.chain_work {
    // This fork has just overtaken the main chain in chainwork
    // Make this fork main chain
    let mut block_height = fork_start_blockheight;

    while block_height != _stored_header.block_height.into()+1 {
        self.main_chain.entry(block_height).write(fork_ptr.chain.entry(block_height).read());
        block_height += 1;
    };

    // Emit chain re-org event
    self.emit(events::ChainReorg {
        fork_submitter: caller,
        fork_id: fork_id,
        tip_commit_hash: _stored_header.get_hash(),
        tip_block_hash_poseidon: _stored_header.get_block_hash_poseidon(),
        start_height: fork_start_blockheight
    });

    // Update globals
    self.main_chainwork.write(_stored_header.chain_work.try_into().unwrap());
    self.main_blockheight.write(_stored_header.block_height.into());
}
```

Consider a scenario:

1. The current main chain is $B_1 \rightarrow B_2 \rightarrow B_3 \rightarrow B_4 \rightarrow B_5$ with a chain work of 100.
2. Alice starts a long fork $B_4 \rightarrow B_5'$ with chain work of 95. This fork is being constructed with more than one transaction.
3. Bob starts another fork with $B_3 \rightarrow B_4^* \rightarrow B_5^* \rightarrow B_6^*$ and a chain work of 105, overtaking the main chain starting from block 4.
4. The main chain becomes $B_1 \rightarrow B_2 \rightarrow B_3 \rightarrow B_4^* \rightarrow B_5^* \rightarrow B_6^*$ with chain work of 105.
5. Alice finishes her long fork $B_4 \rightarrow B_5' \rightarrow B_6' \rightarrow B_7'$ with chain work of 110, overtaking the main chain starting from block 5.
6. The main chain becomes $B_1 \rightarrow B_2 \rightarrow B_3 \rightarrow B_4^* \rightarrow B_5' \rightarrow B_6' \rightarrow B_7'$ with chain work of 110.

As illustrated, the transition from $B_4^* \rightarrow B_5'$ is invalid because these blocks belong to different forks. However, the current system allows this transition, leading to incorrect transaction verification.

Recommendation(s): To ensure proper fork integration, verify that the original block header at `fork_start_blockheight` is still committed to the main chain when the long fork is merged.

Status: Fixed

Update from the client: Fixed in this commit.

6.1.3 Fronting Unconfirmed Bitcoin Transactions Can Cause Fronters to Lose Funds

File(s): packages/spv_swap_vault/src/lib.cairo, packages/btc_utils/src/bitcoin_tx.cairo

Description: The `SpvVaultManager` contract enables users to front funds for unconfirmed Bitcoin transactions via the `front()` function, allowing the receiver to withdraw funds on Starknet without waiting for Bitcoin confirmations. Once the transaction confirms on-chain, the fronter can reclaim their funds (plus fees) via `claim()`.

However, the Bitcoin transaction parser (`from_byte_array()`) explicitly disallows witness data and supports only non-SegWit transactions:

```
fn from_byte_array(data: @ByteArray) -> BitcoinTransaction {
    //...
    // Check that segwit flag is not set (we only accept non-segwit transactions, or transactions with segwit
     data stripped)
    if input_count == 0 && bytes_read == 1 && data.at(5).unwrap() == 0x01 {
        panic(array!['bitcointx: witness not stripped']);
    }
    //...
}
```

Non-SegWit Bitcoin transactions are vulnerable to signature malleability. That is, the transaction hash (`tx_hash`) can be altered post-broadcast without modifying the core transaction data. This creates the risk that a fronter fronts a valid-looking transaction, which is later malleated before being mined.

The result: the malleated version gets confirmed, but its `tx_hash` differs from the one originally fronted. When `claim()` is called, the contract fails to match the `fronting_id`:

```
let fronting_id = tx_data.get_hash(btc_tx_hash_u256);
let fronting_address: ContractAddress = self.liquidity_fronts.entry(owner).entry(vault_id).entry(fronting_id).
     read();
if !fronting_address.is_zero() {
    // Pay back the fronter
} else {
    // Process as if not fronted and pay directly to recipient
}
```

Since the `tx_hash` mismatch prevents identifying the original fronter, the contract processes the transaction as if it was not fronted and sends funds directly to the receiver. If the attacker controls the receiver address, they could maliciously collect double the amount: once from the fronters, and once again after confirmation.

While fronting is inherently risky, this vulnerability amplifies that risk.

Recommendation(s): Introduce mitigations against signature malleability when fronting unconfirmed Bitcoin transactions. Consider enforcing SegWit support or restricting fronting only to transactions that are already confirmed, or at minimum implement replay protection by verifying transaction structure or signature uniqueness.

Status: Acknowledged

Update from the client: Fronting of unconfirmed transaction anyway caries the risk of double-spending, therefore this is not just an issue with signature malleability (which also is a kind of double-spending).

It is also important to note that while `from_byte_array()` in `bitcoin_tx` doesn't support decoding Segwit transactions directly, it still supports decoding Segwit transactions with witness data stripped - this is done to save on amount of data that needs to be posted on Starknet for verification.

An economically rational fronter will always wait for at least 1 bitcoin confirmation before fronting, such that he can minimize the risk of the user double-spending or malleating the signature. The verification of the tx confirmation is outside the scope of the contract because the only party that stands to lose from this is the fronter himself, and it is ultimately a decision of the fronter to front. Therefore the fronting logic will be kept as-is - without an explicit verification of transaction confirmation.

6.2 Info

6.2.1 Lack of Support for Camel-Case Naming in ERC20 Tokens

File(s): packages/erc20_utils/src/lib.cairo

Description: The ERC20 library used by Atomiq contracts is designed to interact with tokens whose function names follow a *snake_-case* convention. However, many ERC20 tokens within the StarkNet ecosystem still use *camelCase* naming conventions. The current implementation of the contract assumes all token function names are in *snake_case*, meaning any calls to tokens that use *camelCase* naming will result in a revert.

```
// Transfer ERC20 tokens to the current contract using transfer_from function
pub fn transfer_in(token: ContractAddress, src: ContractAddress, amount: u256) {
    let erc20_dispatcher = IERC20Dispatcher { contract_address: token };
    // @audit did not support legacy transferFrom tokens
    assert(erc20_dispatcher.transfer_from(src, get_contract_address(), amount), 'transfer_in: transfer_from');
}

// Gets the balance of the specific owner
pub fn balance_of(token: ContractAddress, owner: ContractAddress) -> u256 {
    let erc20_dispatcher = IERC20Dispatcher { contract_address: token };
    erc20_dispatcher.balance_of(owner)
}
```

Recommendation(s): Consider adding support for both *camelCase* and *snake_case* naming conventions for ERC20 tokens within the library.

Status: Acknowledged

Update from the client: Most of the top tokens by marketcap do use snake_case and we are not looking to support legacy, nor low market cap tokens, therefore the increased complexity of the contract is unwarranted.

6.3 Best Practices

6.3.1 Variable Shadowing

File(s): packages/execution_contract/src/execution_proxy.cairo

Description: In the `execution_proxy.cairo` file, there is an instance of variable shadowing. The variable `token` is redefined within the for-loop scope, which may lead to confusion or maintenance difficulties in the future. Although this does not have any immediate impact on the functionality of the code, it can reduce code clarity and make it harder to track the original variable.

```
fn drain_tokens(ref self: ContractState, token: ContractAddress, other_tokens: Span<ContractAddress>, recipient:
     ContractAddress) {
    let balance = erc20_utils::balance_of(token, get_contract_address());
    if balance != 0 {
        erc20_utils::transfer_out(token, recipient, balance);
    }
    for token in other_tokens { // @audit variable shadowing
        let balance = erc20_utils::balance_of(*token, get_contract_address());
        if balance != 0 {
            erc20_utils::transfer_out(*token, recipient, balance);
        }
    }
}
```

Recommendation(s): Consider renaming the inner `token` variable to a more descriptive name to avoid shadowing the outer `token` and improve code readability.

Status: Fixed

Update from the client: Fixed in this commit.. `token` renamed as `main_token` and other tokens to `other_token`

6.3.2 `claim_data` is Emitted Instead of `refund_data` Even in `refund()` Function

File(s): packages/escrow_manager/src/lib.cairo

Description: The `claim_data` event is always emitted instead of `refund_data`. Even in the `refund()` function, which is intended to handle refund logic, `refund_data` is not emitted but rather `claim_data`. This inconsistency could lead to confusion and inaccurate tracking of events related to refunds.

```
// Refund funds
self._pay_out(escrow.offerer, escrow.token, escrow.amount, escrow.is_pay_in());

// Emit event
self.emit(events::Refund {
    offerer: escrow.offerer,
    claimer: escrow.claimer,
    claim_data: escrow.claim_data, // @audit Function refund but does not emit refund_data but claim_data
    escrow_hash: escrow_hash,
    witness_result: refund_result,
    refund_handler: escrow.refund_handler
});
```

Recommendation(s): Consider emitting both `claim_data` and `refund_data` to accurately track the action being performed.

Status: Acknowledged

Update from the client: Emitting refund data is not important, as it currently is just blockheight/timestamp of the escrow expiry. Emitting claim data increases security of HTLC swaps, since user can quickly check if a certain payment hash was already used, making sure to not re-use a hash for which the pre-image is already known (i.e. case of 2 people paying the same lightning network invoice).

6.3.3 Same Swap Cannot Be Executed Twice

File(s): packages/escrow_manager/src/structs/escrow.cairo

Description: Users may need to execute the same swap again using identical parameters (e.g., swap amount, input token, and output token). This could occur if the initial swap encountered an error and was canceled or refunded.

However, the same set of `EscrowData` parameters can only be created and used once in the escrow manager. After being called in `_commit()` and `_finalize()`, it cannot be reused.

```
pub struct EscrowData {
  // Account funding the escrow
  pub offerer: ContractAddress,
  // Account entitled to claim the funds from the escrow
  pub claimer: ContractAddress,
  // Token of the escrow
  pub token: ContractAddress,
  // Address of the IRefundHandler deciding if this escrow is refundable
  pub refund_handler: ContractAddress,
  // Address of the IClaimHandler deciding if this escrow is claimable
  pub claim_handler: ContractAddress,

  // Misc escrow data flags, currently defined: payIn, payOut, reputation
  pub flags: u128,

  // Data provided to the claim handler along with the witness to check claimability
  pub claim_data: felt252,
  // Data provided to the refund handler along with the witness to check for refundability
  pub refund_data: felt252,

  // Amount of tokens in the escrow
  pub amount: u256,

  // Gas/fee token of the swap
  pub fee_token: ContractAddress,
  // Security deposit taken by the offerer if swap expires without claimer claiming (i.e. options premium)
  pub security_deposit: u256,
  // Claimer bounty that can be claimed by a 3rd party claimer if he were to claim this swap on behalf of claimer
  pub claimer_bounty: u256
}
```

Recommendation(s): Consider adding a unique salt to the `EscrowData`. This would allow the system to process the same swap multiple times while maintaining security and integrity.

Status: Acknowledged

Update from the client: Additional salt data is already used in the `flags` data, currently most significant 64-bits are randomized. Added comment clarifying this fact to the code at 6863808.

6.3.4 Unused Input Parameter `extra_data`

File(s): packages/escrow_manager/src/structs/escrow.cairo

Description: The `initialize()` function includes an input parameter, `extra_data`, which is not utilized within the function. This unused parameter may create unnecessary clutter and confusion in the code.

```
// @audit extra_data is unused inside function
fn initialize(ref self: ContractState, escrow: EscrowData, signature: Array<felt252>, timeout: u64, extra_data:
     Span<felt252>) {
    // ...
}
```

Recommendation(s): Consider removing the `extra_data` parameter if it is not required for any logic within the function.

Status: Acknowkedged

Update from the client: `extra_data` parameter is used to save additional data on-chain as calldata for on-chain data-availability/propagation. Added additional comment to clarify this to the code at 1cfaf30.

6.3.5 Unnecessary Caller Check

File(s): packages/spv_swap_vault/src/lib.cairo

Description: The `front()` function includes an explicit check to assert that the caller address is not zero:

```
fn front(
    ref self: ContractState, owner: ContractAddress, vault_id: felt252,
    withdraw_sequence: u32, btc_tx_hash: u256, data: BitcoinVaultTransactionData
) {
    //...
    let caller = get_caller_address();
    assert(!caller.is_zero(), 'front: caller is 0');
    //...
}
```

However, on both Starknet mainnet and testnet, `get_caller_address()` will never return a zero address under normal execution. The check is therefore redundant and adds unnecessary code complexity.

Recommendation(s): Consider removing the `assert(!caller.is_zero())` statement to reduce code noise and improve clarity.

Status: Fixed

Update from the client: Fixed in this commit.

7 Test Evaluation

7.1 Compilation Output

```
scarb build
 Downloading snforge_std v0.38.2
 Downloading snforge_scarb_plugin v0.38.2
   Compiling snforge_scarb_plugin v0.38.2
    Updating crates.io index
     Locking 121 packages to latest compatible versions
      Adding cairo-lang-debug v2.10.0 (available: v2.11.2)
      Adding cairo-lang-diagnostics v2.10.0 (available: v2.11.2)
      Adding cairo-lang-filesystem v2.10.0 (available: v2.11.2)
      Adding cairo-lang-macro v0.1.0 (available: v0.1.1)
      Adding cairo-lang-parser v2.10.0 (available: v2.11.2)
      Adding cairo-lang-syntax v2.10.0 (available: v2.11.2)
      Adding cairo-lang-syntax-codegen v2.10.1 (available: v2.11.2)
      Adding cairo-lang-utils v2.10.0 (available: v2.11.2)
      Adding indoc v2.0.5 (available: v2.0.6)
      Adding smol_str v0.2.2 (available: v0.3.2)
 Downloading crates ...
  Downloaded once_cell v1.21.2
  Downloaded icu_locid_transform_data v1.5.1
  Downloaded icu_normalizer_data v1.5.1
  Downloaded icu_properties_data v1.5.1
   Compiling proc-macro2 v1.0.94
   Compiling unicode-ident v1.0.18
   Compiling serde v1.0.219
   Compiling stable_deref_trait v1.2.0
   Compiling autocfg v1.4.0
   Compiling smallvec v1.14.0
   Compiling cfg-if v1.0.0
   Compiling once_cell v1.21.2
   Compiling version_check v0.9.5
   Compiling hashbrown v0.15.2
   Compiling equivalent v1.0.2
   Compiling libc v0.2.171
   Compiling zerocopy v0.7.35
   Compiling litemap v0.7.5
   Compiling ahash v0.8.11
   Compiling num-traits v0.2.19
   Compiling lock_api v0.4.12
   Compiling parking_lot_core v0.9.10
   Compiling writeable v0.5.5
   Compiling icu_locid_transform_data v1.5.1
   Compiling icu_properties_data v1.5.1
   Compiling scopeguard v1.2.0
   Compiling either v1.15.0
   Compiling allocator-api2 v0.2.21
   Compiling quote v1.0.40
   Compiling itertools v0.12.1
   Compiling tracing-core v0.1.33
   Compiling syn v2.0.100
   Compiling icu_normalizer_data v1.5.1
   Compiling heck v0.4.1
   Compiling winnow v0.7.4
   Compiling pin-project-lite v0.2.16
   Compiling semver v1.0.26
   Compiling num-integer v0.1.46
   Compiling parking_lot v0.12.3
   Compiling thiserror v1.0.69
   Compiling oorandom v11.1.5
   Compiling genco-macros v0.17.10
   Compiling rustc-hash v1.1.0
   Compiling utf8_iter v1.0.4
   Compiling write16 v1.0.0
   Compiling linkme-impl v0.3.32
 Compiling path-clean v1.0.1
 Compiling utf16_iter v1.0.5
 Compiling relative-path v1.9.3
 Compiling xshell-macros v0.2.7
 Compiling xxhash-rust v0.8.15
 Compiling data-encoding v2.8.0
 Compiling cairo-lang-primitive-token v1.0.0
 Compiling percent-encoding v2.3.1
 Compiling lazy_static v1.5.0
 Compiling colored v2.2.0
 Compiling form_urlencoded v1.2.1
 Compiling cairo-lang-macro-stable v1.0.0
 Compiling xshell v0.2.7
 Compiling scarb-stable-hash v1.0.0
 Compiling indoc v2.0.5
 Compiling synstructure v0.13.1
 Compiling serde_derive v1.0.219
 Compiling zerofrom-derive v0.1.6
 Compiling yoke-derive v0.7.5
 Compiling zerovec-derive v0.10.3
 Compiling displaydoc v0.2.5
 Compiling icu_provider_macros v1.5.0
 Compiling tracing-attributes v0.1.28
 Compiling rust-analyzer-salsa-macros v0.17.0-pre.6
 Compiling thiserror-impl v1.0.69
 Compiling cairo-lang-macro-attributes v0.1.0
 Compiling tracing v0.1.41
 Compiling zerofrom v0.1.6
 Compiling yoke v0.7.5
 Compiling genco v0.17.10
 Compiling zerovec v0.10.4
 Compiling linkme v0.3.32
 Compiling unescaper v0.1.5
 Compiling cairo-lang-macro v0.1.0
 Compiling cairo-lang-syntax-codegen v2.10.1
 Compiling tinystr v0.7.6
 Compiling icu_collections v1.5.0
 Compiling icu_locid v1.5.0
 Compiling icu_provider v1.5.0
 Compiling icu_locid_transform v1.5.0
 Compiling icu_properties v1.5.1
 Compiling indexmap v2.8.0
 Compiling hashbrown v0.14.5
 Compiling serde_spanned v0.6.8
 Compiling toml_datetime v0.6.8
 Compiling num-bigint v0.4.6
 Compiling triomphe v0.1.14
 Compiling smol_str v0.2.2
 Compiling toml_edit v0.22.24
 Compiling rust-analyzer-salsa v0.17.0-pre.6
 Compiling cairo-lang-utils v2.10.0
 Compiling cairo-lang-debug v2.10.0
 Compiling icu_normalizer v1.5.0
 Compiling idna_adapter v1.2.0
 Compiling idna v1.0.3
 Compiling url v2.5.4
 Compiling toml v0.8.20
 Compiling cairo-lang-filesystem v2.10.0
 Compiling cairo-lang-diagnostics v2.10.0
 Compiling cairo-lang-syntax v2.10.0
 Compiling cairo-lang-parser v2.10.0
 Compiling snforge_scarb_plugin v0.38.2 (/scarb/registry/src/scarbs.xyz-9djtpev4jug5q/snforge_scarb_plugin
   -0.38.2)
  Finished `release` profile [optimized] target(s) in 1m 09s
 Compiling lib(atomiq_contracts) atomiq_contracts v0.1.0 (/Scarb.toml)
 Compiling starknet-contract(atomiq_contracts) atomiq_contracts v0.1.0 (/Scarb.toml)
  Finished `dev` profile target(s) in 2 minutes
```

7.2 Tests Output

```
scarb test
     Running test atomiq_contracts (cd scripts; npm i; npm run integration; cd ..; snforge test -w;)
added 67 packages, and audited 68 packages in 6s
14 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
> scripts@1.0.0 integration
> node ./generate_test_data.js
Generated test data for: btc_txid_claim_handler
Generated test data for: btc_output_claim_handler
Generated test data for: btc_nonced_output_claim_handler
Generated test data for: timelock_refund_handler
Generated test data for: hashlock_claim_handler
Generated test data for: btc_relay
   Compiling snforge_scarb_plugin v0.38.0
    Finished `release` profile [optimized] target(s) in 0.05s
   Compiling test(atomiq_contracts_unittest) atomiq_contracts v0.1.0 (/contracts/Scarb.toml)
   Compiling test(btc_nonced_output_claim_handler_unittest) btc_nonced_output_claim_handler v0.1.0 (/contracts/
     packages/btc_nonced_output_claim_handler/Scarb.toml)
   Compiling test(btc_nonced_output_claim_handler_integrationtest)
     btc_nonced_output_claim_handler_integrationtest v0.1.0 (/contracts/packages/btc_nonced_output_claim_handler/
     Scarb.toml)
   Compiling test(btc_output_claim_handler_unittest) btc_output_claim_handler v0.1.0 (/contracts/packages/
     btc_output_claim_handler/Scarb.toml)
   Compiling test(btc_output_claim_handler_integrationtest) btc_output_claim_handler_integrationtest v0.1.0 (/
     contracts/packages/btc_output_claim_handler/Scarb.toml)
   Compiling test(btc_relay_unittest) btc_relay v0.1.0 (/contracts/packages/btc_relay/Scarb.toml)
   Compiling test(btc_relay_integrationtest) btc_relay_integrationtest v0.1.0 (/contracts/packages/btc_relay/
     Scarb.toml)
   Compiling test(btc_txid_claim_handler_unittest) btc_txid_claim_handler v0.1.0 (/contracts/packages/
     btc_txid_claim_handler/Scarb.toml)
   Compiling test(btc_txid_claim_handler_integrationtest) btc_txid_claim_handler_integrationtest v0.1.0 (/
     contracts/packages/btc_txid_claim_handler/Scarb.toml)
   Compiling test(btc_utils_unittest) btc_utils v0.1.0 (/contracts/packages/btc_utils/Scarb.toml)
   Compiling test(common_unittest) common v0.1.0 (/contracts/packages/common/Scarb.toml)
   Compiling test(erc20_utils_unittest) erc20_utils v0.1.0 (/contracts/packages/erc20_utils/Scarb.toml)
   Compiling test(escrow_manager_unittest) escrow_manager v0.1.0 (/contracts/packages/escrow_manager/Scarb.toml)
   Compiling test(escrow_manager_integrationtest) escrow_manager_integrationtest v0.1.0 (/contracts/packages/
     escrow_manager/Scarb.toml)
   Compiling test(execution_contract_unittest) execution_contract v0.1.0 (/contracts/packages/execution_contract/
     Scarb.toml)
   Compiling test(execution_contract_integrationtest) execution_contract_integrationtest v0.1.0 (/contracts/
     packages/execution_contract/Scarb.toml)
   Compiling test(hashlock_claim_handler_unittest) hashlock_claim_handler v0.1.0 (/contracts/packages/
     hashlock_claim_handler/Scarb.toml)
   Compiling test(hashlock_claim_handler_integrationtest) hashlock_claim_handler_integrationtest v0.1.0 (/
     contracts/packages/hashlock_claim_handler/Scarb.toml)
   Compiling test(spv_swap_vault_unittest) spv_swap_vault v0.1.0 (/contracts/packages/spv_swap_vault/Scarb.toml)
   Compiling test(spv_swap_vault_integrationtest) spv_swap_vault_integrationtest v0.1.0 (/contracts/packages/
     spv_swap_vault/Scarb.toml)
   Compiling test(timelock_refund_handler_unittest) timelock_refund_handler v0.1.0 (/contracts/packages/
     timelock_refund_handler/Scarb.toml)
   Compiling test(timelock_refund_handler_integrationtest) timelock_refund_handler_integrationtest v0.1.0 (/
     contracts/packages/timelock_refund_handler/Scarb.toml)
    Finished `dev` profile target(s) in 4 minutes
Collected 0 test(s) from atomiq_contracts package
Running 0 test(s) from src/
Tests: 0 passed, 0 failed, 0 skipped, 0 ignored, 0 filtered out
Collected 12 test(s) from btc_nonced_output_claim_handler package
Running 0 test(s) from src/
Running 12 test(s) from tests/
[PASS] btc_nonced_output_claim_handler_integrationtest::test::test_invalid_commitment (gas: ~5071)
[PASS] btc_nonced_output_claim_handler_integrationtest::test::test_invalid_blockheader (gas: ~5081)
[PASS] btc_nonced_output_claim_handler_integrationtest::test::test_invalid_confirmations (gas: ~5108)
[PASS] btc_nonced_output_claim_handler_integrationtest::test::test_invalid_empty_ins (gas: ~5072)
[PASS] btc_nonced_output_claim_handler_integrationtest::test::test_invalid_empty_witness (gas: ~4986)
[PASS] btc_nonced_output_claim_handler_integrationtest::test::test_invalid_locktime_too_low (gas: ~5073)
[PASS] btc_nonced_output_claim_handler_integrationtest::test::test_invalid_nsequence_bits (gas: ~5090)
```

```
[PASS] btc_nonced_output_claim_handler_integrationtest::test::test_invalid_merkle (gas: ~5131)
[PASS] btc_nonced_output_claim_handler_integrationtest::test::test_invalid_nsequence_match (gas: ~5100)
[PASS] btc_nonced_output_claim_handler_integrationtest::test::test_invalid_output (gas: ~5097)
[PASS] btc_nonced_output_claim_handler_integrationtest::test::test_invalid_vout_out_of_bounds (gas: ~5089)
[PASS] btc_nonced_output_claim_handler_integrationtest::test::test_valid_random (gas: ~53518)
Tests: 12 passed, 0 failed, 0 skipped, 0 ignored, 0 filtered out
Collected 9 test(s) from btc_output_claim_handler package
Running 0 test(s) from src/
Running 9 test(s) from tests/
[PASS] btc_output_claim_handler_integrationtest::test::test_invalid_commitment (gas: ~5090)
[PASS] btc_output_claim_handler_integrationtest::test::test_invalid_blockheader (gas: ~5080)
[PASS] btc_output_claim_handler_integrationtest::test::test_invalid_output (gas: ~5090)
[PASS] btc_output_claim_handler_integrationtest::test::test_invalid_vout_out_of_bounds (gas: ~5070)
[PASS] btc_output_claim_handler_integrationtest::test::test_valid_real (gas: ~57620)
[PASS] btc_output_claim_handler_integrationtest::test::test_valid_random (gas: ~78140)
[PASS] btc_output_claim_handler_integrationtest::test::test_invalid_empty_witness (gas: ~4986)
[PASS] btc_output_claim_handler_integrationtest::test::test_invalid_confirmations (gas: ~5127)
[PASS] btc_output_claim_handler_integrationtest::test::test_invalid_merkle (gas: ~5134)
Tests: 9 passed, 0 failed, 0 skipped, 0 ignored, 0 filtered out
Collected 50 test(s) from btc_relay package
Running 29 test(s) from src/
[PASS] btc_relay::structs::blockheader::tests::hash_random_blockheaders (gas: ~5352)
[PASS] btc_relay::structs::blockheader::tests::hash_real_blockheaders (gas: ~5352)
[PASS] btc_relay::utils::difficulty::tests::get_chainwork_random_data (gas: ~7)
[PASS] btc_relay::structs::stored_blockheader::tests::random_fail_block_invalid_pow (gas: ~542)
[PASS] btc_relay::utils::endianness::tests::manual (gas: ~4)
[PASS] btc_relay::utils::difficulty::tests::get_chainwork_real_data (gas: ~24)
[PASS] btc_relay::utils::nbits::tests::bitcoin_core_negative_nbits_test_vector_1 (gas: ~2)
[PASS] btc_relay::utils::endianness::tests::random (gas: ~5)
[PASS] btc_relay::utils::nbits::tests::test_nbits_to_target (gas: ~2)
[PASS] btc_relay::utils::nbits::tests::bitcoin_core_test_vectors (gas: ~83)
[PASS] btc_relay::utils::nbits::tests::bitcoin_core_negative_nbits_test_vector_2 (gas: ~3)
[PASS] btc_relay::utils::nbits::tests::test_target_to_nbits (gas: ~3)
[PASS] btc_relay::utils::nbits::tests::bitcoin_core_overflow_nbits_test_vector (gas: ~2)
[PASS] btc_relay::utils::u256_utils::tests::block_hashes (gas: ~44)
[PASS] btc_relay::utils::u256_utils::tests::manual (gas: ~39)
[PASS] btc_relay::utils::u256_utils::tests::random (gas: ~48)
[PASS] btc_relay::structs::stored_blockheader::tests::random_fail_block_timestamp_median (gas: ~542)
[PASS] btc_relay::structs::stored_blockheader::tests::random_success_block_pow_retarget_slow_limit (gas: ~544)
[PASS] btc_relay::structs::stored_blockheader::tests::random_fail_block_wrong_nbits (gas: ~2)
[PASS] btc_relay::structs::stored_blockheader::tests::random_fail_block_wrong_nbits_diff_adjustment (gas: ~8)
[PASS] btc_relay::structs::stored_blockheader::tests::random_success_block_pow_retarget_fast_limit (gas: ~544)
[PASS] btc_relay::structs::stored_blockheader::tests::random_success_block_pow_retarget (gas: ~544)
[PASS] btc_relay::structs::stored_blockheader::tests::random_success_block_timestamp_median (gas: ~542)
[PASS] btc_relay::structs::stored_blockheader::tests::real_success_block_pow_retarget (gas: ~545)
[PASS] btc_relay::structs::stored_blockheader::tests::real_success_block_update (gas: ~542)
[PASS] btc_relay::structs::stored_blockheader::tests::random_success_block_updates (gas: ~542)
[PASS] btc_relay::structs::stored_blockheader::tests::real_success_block_pow_retarget_fast_limit (gas: ~544)
[PASS] btc_relay::utils::difficulty::tests::compute_new_target_real_adjustments (gas: ~48)
[PASS] btc_relay::structs::stored_blockheader::tests::random_fail_block_invalid_prev_blockhash (gas: ~1)
Running 21 test(s) from tests/
[PASS] btc_relay_integrationtest::blocks::block_invalid_future_timestamp (gas: ~321)
[PASS] btc_relay_integrationtest::blocks::block_invalid_median_timestamp (gas: ~417)
[PASS] btc_relay_integrationtest::blocks::block_invalid_nbits (gas: ~321)
[PASS] btc_relay_integrationtest::blocks::block_invalid_nbits_diff_adjustment (gas: ~417)
[PASS] btc_relay_integrationtest::blocks::block_invalid_prev_block (gas: ~321)
[PASS] btc_relay_integrationtest::blocks::block_invalid_pow (gas: ~321)
[PASS] btc_relay_integrationtest::chains::long_fork_chain (gas: ~26633)
[PASS] btc_relay_integrationtest::chains::long_fork_chain_chainwork (gas: ~173735)
[PASS] btc_relay_integrationtest::chains::long_fork_not_enough_chainwork (gas: ~159566)
[PASS] btc_relay_integrationtest::chains::long_fork_stored_header_not_committed_in_fork_data (gas: ~19806)
[PASS] btc_relay_integrationtest::chains::long_fork_not_enough_length (gas: ~16585)
[PASS] btc_relay_integrationtest::chains::main_chain (gas: ~13269)
[PASS] btc_relay_integrationtest::chains::long_fork_stored_header_not_committed (gas: ~13269)
[PASS] btc_relay_integrationtest::chains::main_chain_stored_header_not_committed (gas: ~325)
[PASS] btc_relay_integrationtest::chains::main_chain_stored_header_not_tip (gas: ~6829)
[PASS] btc_relay_integrationtest::chains::main_chain_diff_adjustment (gas: ~19709)
[PASS] btc_relay_integrationtest::chains::short_fork_chain (gas: ~24612)
[PASS] btc_relay_integrationtest::chains::short_fork_not_enough_chainwork (gas: ~79688)
[PASS] btc_relay_integrationtest::chains::short_fork_stored_header_not_committed (gas: ~13269)
```

```
[PASS] btc_relay_integrationtest::chains::short_fork_not_enough_length (gas: ~13269)
[PASS] btc_relay_integrationtest::chains::short_fork_chain_chainwork (gas: ~160741)
Tests: 50 passed, 0 failed, 0 skipped, 0 ignored, 0 filtered out
Collected 7 test(s) from btc_txid_claim_handler package
Running 0 test(s) from src/
Running 7 test(s) from tests/
[PASS] btc_txid_claim_handler_integrationtest::test::test_invalid_commitment (gas: ~5061)
[PASS] btc_txid_claim_handler_integrationtest::test::test_invalid_blockheader (gas: ~5072)
[PASS] btc_txid_claim_handler_integrationtest::test::test_invalid_empty_witness (gas: ~4986)
[PASS] btc_txid_claim_handler_integrationtest::test::test_invalid_confirmations (gas: ~5100)
[PASS] btc_txid_claim_handler_integrationtest::test::test_invalid_merkle (gas: ~5094)
[PASS] btc_txid_claim_handler_integrationtest::test::test_valid_real (gas: ~61013)
[PASS] btc_txid_claim_handler_integrationtest::test::test_valid_random (gas: ~58872)
Tests: 7 passed, 0 failed, 0 skipped, 0 ignored, 0 filtered out
Collected 58 test(s) from btc_utils package
Running 58 test(s) from src/
[PASS] btc_utils::bitcoin_merkle_tree::tests::test_invalid_change_leaf (gas: ~8031)
[PASS] btc_utils::bitcoin_merkle_tree::tests::test_invalid_change_proof_len (gas: ~7496)
[PASS] btc_utils::bitcoin_merkle_tree::tests::test_invalid_change_root (gas: ~8031)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_27b (gas: ~24)
[PASS] btc_utils::bitcoin_merkle_tree::tests::test_valid_random (gas: ~62102)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_28b (gas: ~6)
[PASS] btc_utils::bitcoin_merkle_tree::tests::test_valid_real (gas: ~50860)
[PASS] btc_utils::bitcoin_merkle_tree::tests::test_invalid_change_proof (gas: ~8031)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_29b (gas: ~8)
[PASS] btc_utils::bitcoin_tx::tests::test_invalid_data_extended (gas: ~23)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_2b (gas: ~2)
[PASS] btc_utils::bitcoin_tx::tests::test_invalid_tx_length_64 (gas: ~1)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_30b (gas: ~3)
[PASS] btc_utils::bitcoin_tx::tests::test_invalid_witness_not_stripped (gas: ~5)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_31b (gas: ~17)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_32b (gas: ~10)
[PASS] btc_utils::bitcoin_tx::tests::test_real_txs (gas: ~19446)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_3b (gas: ~3)
[PASS] btc_utils::byte_array::tests::test_invalid_bytes31 (gas: ~8)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_4b (gas: ~2)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252 (gas: ~16)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_5b (gas: ~2)
[PASS] btc_utils::bitcoin_tx::tests::test_random_txs (gas: ~27296)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_10b (gas: ~5)
[PASS] btc_utils::bitcoin_merkle_tree::tests::test_invalid_change_position (gas: ~8031)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_7b (gas: ~4)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_8b (gas: ~2)
[PASS] btc_utils::compact_size::tests::test_manual (gas: ~25)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_9b (gas: ~4)
[PASS] btc_utils::compact_size::tests::test_manual_out_of_bounds_u16 (gas: ~3)
[PASS] btc_utils::byte_array::tests::test_invalid_u16_le (gas: ~1)
[PASS] btc_utils::compact_size::tests::test_manual_out_of_bounds_u32 (gas: ~3)
[PASS] btc_utils::compact_size::tests::test_manual_out_of_bounds_u64 (gas: ~3)
[PASS] btc_utils::byte_array::tests::test_invalid_u256 (gas: ~9)
[PASS] btc_utils::byte_array::tests::test_invalid_u32_le (gas: ~2)
[PASS] btc_utils::compact_size::tests::test_manual_out_of_bounds_u8 (gas: ~3)
[PASS] btc_utils::byte_array::tests::test_invalid_u64_le (gas: ~3)
[PASS] btc_utils::compact_size::tests::test_random (gas: ~111)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_6b (gas: ~3)
[PASS] btc_utils::byte_array::tests::test_random_access (gas: ~1518)
[PASS] btc_utils::byte_array::tests::test_random (gas: ~8920)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_11b (gas: ~2)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_12b (gas: ~6)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_13b (gas: ~8)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_14b (gas: ~4)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_15b (gas: ~6)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_16b (gas: ~8)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_21b (gas: ~18)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_22b (gas: ~18)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_17b (gas: ~2)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_23b (gas: ~16)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_24b (gas: ~5)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_18b (gas: ~2)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_25b (gas: ~14)
```

```
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_19b (gas: ~16)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_26b (gas: ~10)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_1b (gas: ~1)
[PASS] btc_utils::byte_array::tests::test_invalid_felt252_20b (gas: ~2)
Tests: 58 passed, 0 failed, 0 skipped, 0 ignored, 0 filtered out
Collected 0 test(s) from common package
Running 0 test(s) from src/
Tests: 0 passed, 0 failed, 0 skipped, 0 ignored, 0 filtered out
Collected 0 test(s) from erc20_utils package
Running 0 test(s) from src/
Tests: 0 passed, 0 failed, 0 skipped, 0 ignored, 0 filtered out
Collected 45 test(s) from escrow_manager package
Running 8 test(s) from src/
[PASS] escrow_manager::sighash::tests::random_refund_sighash (gas: ~8)
[PASS] escrow_manager::sighash::tests::random_init_sighash (gas: ~8)
[PASS] escrow_manager::state::escrow::tests::test_packing (gas: ~15)
[PASS] escrow_manager::state::reputation::tests::test_packing (gas: ~16)
[PASS] escrow_manager::structs::escrow::tests::parse_flags (gas: ~4)
[PASS] escrow_manager::state::reputation::tests::test_updates (gas: ~10)
[PASS] escrow_manager::structs::escrow::tests::total_deposit (gas: ~1)
[PASS] escrow_manager::state::reputation::tests::test_updates_overflowing (gas: ~10)
Running 37 test(s) from tests/
[PASS] escrow_manager_integrationtest::escrow_claim::invalid_claim_uninitialized (gas: ~70601)
[PASS] escrow_manager_integrationtest::escrow_claim::invalid_claim_double (gas: ~103785)
[PASS] escrow_manager_integrationtest::escrow_claim::valid_claim (gas: ~101380)
[PASS] escrow_manager_integrationtest::escrow_claim::valid_claim_invert_deposits (gas: ~29597)
[PASS] escrow_manager_integrationtest::escrow_claim::invalid_claim_handler (gas: ~81971)
[PASS] escrow_manager_integrationtest::escrow_claim::valid_claim_success_action (gas: ~101380)
[PASS] escrow_manager_integrationtest::escrow_init::invalid_initialize_commit_twice (gas: ~81152)
[PASS] escrow_manager_integrationtest::escrow_init::invalid_initialize_expired (gas: ~69780)
[PASS] escrow_manager_integrationtest::escrow_refund_coop::invalid_coop_refund_sign_diff_timeout (gas: ~81994)
[PASS] escrow_manager_integrationtest::escrow_init::invalid_initialize_not_enough_allowance (gas: ~35491)
[PASS] escrow_manager_integrationtest::escrow_refund_coop::invalid_coop_refund_sign_random_msg (gas: ~81962)
[PASS] escrow_manager_integrationtest::escrow_init::invalid_initialize_not_enough_balance (gas: ~76163)
[PASS] escrow_manager_integrationtest::escrow_refund_coop::invalid_coop_refund_timed_out (gas: ~81939)
[PASS] escrow_manager_integrationtest::escrow_refund_coop::invalid_coop_refund_uninitialized (gas: ~70569)
[PASS] escrow_manager_integrationtest::escrow_init::invalid_initialize_not_enough_gas_allowance (gas: ~52130)
[PASS] escrow_manager_integrationtest::escrow_init::invalid_initialize_not_enough_gas_balance (gas: ~61053)
[PASS] escrow_manager_integrationtest::escrow_refund_coop::invalid_coop_refund_wrong_signer (gas: ~82004)
[PASS] escrow_manager_integrationtest::escrow_init::invalid_initialize_sign_different_timeout (gas: ~75979)
[PASS] escrow_manager_integrationtest::escrow_refund_coop::invalid_refund_double (gas: ~96256)
[PASS] escrow_manager_integrationtest::escrow_init::invalid_initialize_wrong_sender (gas: ~75934)
[PASS] escrow_manager_integrationtest::lp_vault::invalid_deposit_no_allowance (gas: ~1015)
[PASS] escrow_manager_integrationtest::escrow_refund_coop::valid_coop_refund (gas: ~93887)
[PASS] escrow_manager_integrationtest::lp_vault::invalid_deposit_no_balance (gas: ~1201)
[PASS] escrow_manager_integrationtest::escrow_init::invalid_initialize_wrong_sign_message (gas: ~75946)
[PASS] escrow_manager_integrationtest::lp_vault::invalid_withdraw_no_balance (gas: ~1326)
[PASS] escrow_manager_integrationtest::escrow_init::invalid_initialize_wrong_signer (gas: ~75989)
[PASS] escrow_manager_integrationtest::lp_vault::valid_deposit (gas: ~1324)
[PASS] escrow_manager_integrationtest::escrow_init::valid_initialize (gas: ~79557)
[PASS] escrow_manager_integrationtest::lp_vault::valid_withdraw_full (gas: ~1153)
[PASS] escrow_manager_integrationtest::escrow_refund::invalid_refund_handler (gas: ~81971)
[PASS] escrow_manager_integrationtest::escrow_refund::invalid_refund_double (gas: ~97082)
[PASS] escrow_manager_integrationtest::escrow_refund::invalid_refund_uninitialized (gas: ~70601)
[PASS] escrow_manager_integrationtest::escrow_refund::valid_refund_invert_deposits (gas: ~26306)
[PASS] escrow_manager_integrationtest::lp_vault::valid_withdraw_full_external (gas: ~1252)
[PASS] escrow_manager_integrationtest::lp_vault::valid_withdraw_partial_external (gas: ~1444)
[PASS] escrow_manager_integrationtest::lp_vault::valid_withdraw_partial (gas: ~1345)
[PASS] escrow_manager_integrationtest::escrow_refund::valid_refund (gas: ~94677)
Tests: 45 passed, 0 failed, 0 skipped, 0 ignored, 0 filtered out
Collected 25 test(s) from execution_contract package
Running 4 test(s) from src/
[PASS] execution_contract::state::tests::clear (gas: ~1)
[PASS] execution_contract::state::tests::clear_all (gas: ~1)
[PASS] execution_contract::utils::tests::span_hash (gas: ~6)
[PASS] execution_contract::state::tests::test_packing (gas: ~20)
Running 21 test(s) from tests/
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
[PASS] execution_contract_integrationtest::create::invalid_execution_hash_0 (gas: ~1913)
```

```
[PASS] execution_contract_integrationtest::create::invalid_execution_already_initiated (gas: ~2344)
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
[PASS] execution_contract_integrationtest::create::invalid_execution_not_enough_allowance (gas: ~2393)
[PASS] execution_contract_integrationtest::refund::invalid_already_processed (gas: ~2294)
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
[PASS] execution_contract_integrationtest::refund::invalid_not_initiated (gas: ~1613)
[PASS] execution_contract_integrationtest::create::invalid_execution_not_enough_funds (gas: ~2297)
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
[PASS] execution_contract_integrationtest::create::valid_create (gas: ~2334)
[PASS] execution_contract_integrationtest::refund::valid_refund_expired (gas: ~2003)
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
[PASS] execution_contract_integrationtest::refund::valid_refund_not_expired (gas: ~2003)
[PASS] execution_contract_integrationtest::execute::invalid_already_processed (gas: ~2437)
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
[PASS] execution_contract_integrationtest::execute::invalid_calls (gas: ~2349)
[PASS] execution_contract_integrationtest::refund_expired::invalid_already_processed (gas: ~2410)
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
[PASS] execution_contract_integrationtest::refund_expired::invalid_not_expired (gas: ~2349)
[PASS] execution_contract_integrationtest::execute::invalid_drain_tokens (gas: ~2348)
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
[PASS] execution_contract_integrationtest::refund_expired::invalid_not_initiated (gas: ~1616)
[PASS] execution_contract_integrationtest::execute::invalid_not_initiated (gas: ~1614)
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
[PASS] execution_contract_integrationtest::refund_expired::valid_refund (gas: ~2133)
[PASS] execution_contract_integrationtest::execute::valid_execute_empty (gas: ~2190)
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
[FAIL] execution_contract_integrationtest::execute::valid_execute_panic_test_contract
Failure data:
    0x70616e6963206865726521 ('panic here!')
note: run with `SNFORGE_BACKTRACE=1` environment variable to display a backtrace
5b4644a8bc1cafc820a7937b77af9cdf11472c764fe6636c0eaeef2a08032ca
[PASS] execution_contract_integrationtest::execute::valid_execute_test_contract (gas: ~2437)
[PASS] execution_contract_integrationtest::execute::valid_execute_test_contract_drain (gas: ~2665)
Tests: 24 passed, 1 failed, 0 skipped, 0 ignored, 0 filtered out
Collected 8 test(s) from hashlock_claim_handler package
Running 8 test(s) from tests/
[PASS] hashlock_claim_handler_integrationtest::test::test_invalid_empty_witness (gas: ~103)
[PASS] hashlock_claim_handler_integrationtest::test::test_invalid_wrong_hash_random (gas: ~131)
[PASS] hashlock_claim_handler_integrationtest::test::test_valid_manual (gas: ~1167)
[PASS] hashlock_claim_handler_integrationtest::test::test_invalid_wrong_secret_random (gas: ~131)
[PASS] hashlock_claim_handler_integrationtest::test::test_valid_random (gas: ~1880)
[PASS] hashlock_claim_handler_integrationtest::test::test_invalid_wrong_witness_overflow (gas: ~103)
[PASS] hashlock_claim_handler_integrationtest::test::test_invalid_wrong_witness_large (gas: ~103)
[PASS] hashlock_claim_handler_integrationtest::test::test_invalid_wrong_witness_small (gas: ~103)
Running 0 test(s) from src/
Tests: 8 passed, 0 failed, 0 skipped, 0 ignored, 0 filtered out
Collected 84 test(s) from spv_swap_vault package
Running 46 test(s) from tests/
[PASS] spv_swap_vault_integrationtest::claim::close_amounts_overflow_0_0 (gas: ~4955)
[PASS] spv_swap_vault_integrationtest::claim::close_amounts_overflow_0_1 (gas: ~4955)
[PASS] spv_swap_vault_integrationtest::claim::close_fronting_fee_1_overflow (gas: ~4955)
[PASS] spv_swap_vault_integrationtest::claim::close_amounts_overflow_0_2 (gas: ~4955)
[PASS] spv_swap_vault_integrationtest::claim::close_withdraw_too_much_token0 (gas: ~4955)
[PASS] spv_swap_vault_integrationtest::claim::close_amounts_overflow_1_0 (gas: ~4955)
[PASS] spv_swap_vault_integrationtest::claim::close_withdraw_too_much_token1 (gas: ~4955)
[PASS] spv_swap_vault_integrationtest::claim::close_amounts_overflow_1_1 (gas: ~4955)
[PASS] spv_swap_vault_integrationtest::claim::invalid_btc_tx_confirmations (gas: ~3617)
[PASS] spv_swap_vault_integrationtest::claim::close_btc_input_1_notfound (gas: ~4419)
[PASS] spv_swap_vault_integrationtest::claim::invalid_btc_tx_empty_inputs (gas: ~3259)
[PASS] spv_swap_vault_integrationtest::claim::close_btc_invalid_recipient (gas: ~4955)
[PASS] spv_swap_vault_integrationtest::claim::invalid_btc_tx_in_0_utxo (gas: ~3617)
```

```
[PASS] spv_swap_vault_integrationtest::claim::close_btc_output_1_empty_script (gas: ~4419)
[PASS] spv_swap_vault_integrationtest::claim::invalid_btc_tx_merkle_proof (gas: ~3617)
[PASS] spv_swap_vault_integrationtest::claim::close_btc_output_1_length (gas: ~4420)
[PASS] spv_swap_vault_integrationtest::claim::invalid_closed_vault (gas: ~2544)
[PASS] spv_swap_vault_integrationtest::claim::close_btc_output_1_not_found (gas: ~4419)
[PASS] spv_swap_vault_integrationtest::claim::close_btc_output_1_not_opreturn (gas: ~4419)
[PASS] spv_swap_vault_integrationtest::claim::valid_claim (gas: ~24856)
[PASS] spv_swap_vault_integrationtest::claim::close_caller_fee_0_overflow (gas: ~4955)
[PASS] spv_swap_vault_integrationtest::deposit::invalid_allowance_token0 (gas: ~3094)
[PASS] spv_swap_vault_integrationtest::claim::valid_claim_fronted (gas: ~29078)
[PASS] spv_swap_vault_integrationtest::claim::close_caller_fee_1_overflow (gas: ~4955)
[PASS] spv_swap_vault_integrationtest::deposit::invalid_allowance_token1 (gas: ~3000)
[PASS] spv_swap_vault_integrationtest::claim::close_execution_fee_0_overflow (gas: ~4955)
[PASS] spv_swap_vault_integrationtest::deposit::invalid_balance_token0 (gas: ~2998)
[PASS] spv_swap_vault_integrationtest::deposit::invalid_balance_token1 (gas: ~2904)
[PASS] spv_swap_vault_integrationtest::front::invalid_vault_not_opened (gas: ~2918)
[PASS] spv_swap_vault_integrationtest::deposit::invalid_deposit_not_opened (gas: ~2104)
[PASS] spv_swap_vault_integrationtest::front::valid_front (gas: ~3822)
[PASS] spv_swap_vault_integrationtest::deposit::valid_deposit_multiple (gas: ~3077)
[PASS] spv_swap_vault_integrationtest::front::valid_front_exec_hash (gas: ~4488)
[PASS] spv_swap_vault_integrationtest::deposit::valid_deposit_single (gas: ~2963)
[PASS] spv_swap_vault_integrationtest::front::invalid_allowance_token0 (gas: ~4004)
[PASS] spv_swap_vault_integrationtest::front::valid_front_exec_hash_only_token0 (gas: ~4389)
[PASS] spv_swap_vault_integrationtest::front::invalid_allowance_token1 (gas: ~4004)
[PASS] spv_swap_vault_integrationtest::front::valid_front_only_token0 (gas: ~3723)
[PASS] spv_swap_vault_integrationtest::open::invalid_already_opened (gas: ~2471)
[PASS] spv_swap_vault_integrationtest::front::invalid_balance_token0 (gas: ~4004)
[PASS] spv_swap_vault_integrationtest::open::invalid_zero_utxo (gas: ~1779)
[PASS] spv_swap_vault_integrationtest::open::valid_create (gas: ~2467)
[PASS] spv_swap_vault_integrationtest::front::invalid_balance_token1 (gas: ~4004)
[PASS] spv_swap_vault_integrationtest::front::invalid_front_already_fronted (gas: ~5769)
[PASS] spv_swap_vault_integrationtest::front::invalid_front_already_claimed (gas: ~6541)
[PASS] spv_swap_vault_integrationtest::claim::close_fronting_fee_0_overflow (gas: ~4955)
Running 38 test(s) from src/
[PASS] spv_swap_vault::state::tests::close (gas: ~1)
[PASS] spv_swap_vault::state::tests::from_raw_overflow_0 (gas: ~2)
[PASS] spv_swap_vault::state::tests::from_raw_overflow_1 (gas: ~3)
[PASS] spv_swap_vault::state::tests::from_raw (gas: ~6)
[PASS] spv_swap_vault::state::tests::from_raw_token0 (gas: ~3)
[PASS] spv_swap_vault::structs::tests::parse_caller_fee_1_overflow (gas: ~84)
[PASS] spv_swap_vault::state::tests::from_raw_token0_overflow (gas: ~2)
[PASS] spv_swap_vault::structs::tests::parse_execution_fee_0_overflow (gas: ~75)
[PASS] spv_swap_vault::state::tests::from_raw_token1 (gas: ~3)
[PASS] spv_swap_vault::structs::tests::parse_fronting_fee_0_overflow (gas: ~74)
[PASS] spv_swap_vault::state::tests::from_raw_token1_overflow (gas: ~2)
[PASS] spv_swap_vault::structs::tests::parse_fronting_fee_1_overflow (gas: ~84)
[PASS] spv_swap_vault::state::tests::is_opened (gas: ~2)
[PASS] spv_swap_vault::structs::tests::parse_invalid_recipient (gas: ~354)
[PASS] spv_swap_vault::state::tests::test_packing (gas: ~23)
[PASS] spv_swap_vault::structs::tests::parse_no_inputs (gas: ~41)
[PASS] spv_swap_vault::state::tests::withdraw (gas: ~1)
[PASS] spv_swap_vault::state::tests::withdraw_too_much_token0 (gas: ~1)
[PASS] spv_swap_vault::structs::tests::parse_output_empty_script (gas: ~46)
[PASS] spv_swap_vault::state::tests::withdraw_too_much_token1 (gas: ~1)
[PASS] spv_swap_vault::structs::tests::parse_output_invalid_len (gas: ~52)
[PASS] spv_swap_vault::structs::tests::get_full_amounts (gas: ~1)
[PASS] spv_swap_vault::structs::tests::parse_output_not_opreturn (gas: ~46)
[PASS] spv_swap_vault::structs::tests::get_full_amounts_overflow_0_0 (gas: ~1)
[PASS] spv_swap_vault::structs::tests::parse_single_input (gas: ~47)
[PASS] spv_swap_vault::structs::tests::get_full_amounts_overflow_0_1 (gas: ~1)
[PASS] spv_swap_vault::structs::tests::parse_single_output (gas: ~41)
[PASS] spv_swap_vault::structs::tests::get_full_amounts_overflow_0_2 (gas: ~1)
[PASS] spv_swap_vault::structs::tests::parse_valid (gas: ~389)
[PASS] spv_swap_vault::structs::tests::get_full_amounts_overflow_1_0 (gas: ~1)
[PASS] spv_swap_vault::utils::tests::test_fee_amount (gas: ~3)
[PASS] spv_swap_vault::utils::tests::test_fee_amount_overflow (gas: ~2)
[PASS] spv_swap_vault::structs::tests::get_full_amounts_overflow_1_1 (gas: ~1)
[PASS] spv_swap_vault::utils::tests::to_u256 (gas: ~3)
[PASS] spv_swap_vault::structs::tests::get_hash (gas: ~4)
[PASS] spv_swap_vault::utils::tests::tuple_add_u64 (gas: ~1)
```

```
[PASS] spv_swap_vault::structs::tests::parse_caller_fee_0_overflow (gas: ~73)
[PASS] spv_swap_vault::state::tests::deposit (gas: ~1)
Tests: 84 passed, 0 failed, 0 skipped, 0 ignored, 0 filtered out
Collected 10 test(s) from timelock_refund_handler package
Running 0 test(s) from src/
Running 10 test(s) from tests/
[PASS] timelock_refund_handler_integrationtest::test::test_fail_not_expired_1_manual (gas: ~104)
[PASS] timelock_refund_handler_integrationtest::test::test_fail_not_expired_2_manual (gas: ~104)
[PASS] timelock_refund_handler_integrationtest::test::test_fail_not_expired_3_manual (gas: ~104)
[PASS] timelock_refund_handler_integrationtest::test::test_fail_overflow_2_manual (gas: ~104)
[PASS] timelock_refund_handler_integrationtest::test::test_fail_not_expired_4_manual (gas: ~104)
[PASS] timelock_refund_handler_integrationtest::test::test_success_manual (gas: ~124)
[PASS] timelock_refund_handler_integrationtest::test::test_success_random (gas: ~148)
[PASS] timelock_refund_handler_integrationtest::test::test_fail_not_expired_random (gas: ~142)
[PASS] timelock_refund_handler_integrationtest::test::test_fail_overflow_1_manual (gas: ~104)
[PASS] timelock_refund_handler_integrationtest::test::test_fail_non_empty_witness (gas: ~103)
Tests: 10 passed, 0 failed, 0 skipped, 0 ignored, 0 filtered out
Failures:
    execution_contract_integrationtest::execute::valid_execute_panic_test_contract
```