Exa App Plugin (Exact.Ly)

Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

Type	ERC-6900 Plugin
Timeline	2025-01-14 through 2025-01-27
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	README.md
Source Code	• exactly/mobile #b262356
Auditors	• Hytham Farah Auditing Engineer • Nikita Belenkov Senior Auditing Engineer • Ruben Koch Senior Auditing Engineer

Documentation quality	Medium
Test quality	High
Total Findings	11 Fixed: 11
High severity findings	3 Fixed: 3
Medium severity findings	3 Fixed: 3
Low severity findings	5 Fixed: 5
Undetermined severity findings	0
Informational findings	0

Summary of Findings

The **ExaPlugin** is a modular system designed to integrate on-chain lending with off-chain credit card issuers, enabling seamless payments backed by decentralized collateral. All users are required to deposit on-chain collateral that is used to fund the card transactions. The card offers 2 modes of payment, either credit or debit. The debit method takes the collateral as payment, whilst the credit takes a borrow position against the collateral. The account is modified for these transactions via a signed message from the payment processor, ie the issuer in this protocol. The protocol has a Proposal mechanism for withdrawals so that sufficient collateral exists and remains locked during normal operation of the system. The plugin interacts with Exactly Protocol for core lending functionality, leveraging its markets and oracles to evaluate collateral and enforce liquidity requirements.

The scope of the audit was limited to the plugin itself, excluding the underlying Exactly Protocol's backend components, such as oracles, liquidation logic, and debt management, which were assumed to function correctly (see "Operational Considerations"). Our core concern was validating the integrity of the plugin's interactions to prevent any unintended consequences of state updates or design flaws. Some significant vulnerabilities have been identified, which we strongly recommend to address. These issues revolve around insufficient restrictions on interactions with the core protocol, enabling credit card purchases to be unbacked on-chain with malicious interactions by end-users.

The Exactly team has been being very communicative and helpful in providing context around the core protocol. We would also like to highlight the thorough test suite.

Fix-Review Update

The fix-review process went through four rounds in total, concluding at `368df25`. Some new issues were introduced in earlier rounds, which have been communicated to the Exactly team and have now been sufficiently fixed. Due to the time-constrained nature of the audit, a proper writeup of these issues is not part of this report.

ID	DESCRIPTION	SEVERITY	STATUS
EXA-1	`executeBatch()` Can Bypass the Created Execution Hook Restrictions	● High	Fixed
EXA-2	Plugin Can Be Uninstalled, Removing All Checks on Plugin-Side	● High	Fixed
EXA-3	Incorrect Assumptions Enable Numerous Credit Card Purchase Race Conditions	● High	Fixed

ID	DESCRIPTION	SEVERITY	STATUS
EXA-4	Proposal Time Lock Is Skipped in the Case of Ether Withdrawal With No Swap	● Medium	Fixed
EXA-5	More Selectors Need Restrictions to Ensure Correct Plugin Behaviour	● Medium	Fixed
EXA-6	Hashes are not invalidated properly	● Medium	Fixed
EXA-7	Refunding Signature Should Differ From the Collecting Signature	● Low	Fixed
EXA-8	Malicious Plugin Can Add User Op Validation Functions to the Newly Added Selector	● Low	Fixed
EXA-9	Incorrect `operationExpiry` Is Used to Update the Expiry Window	● Low	Fixed
EXA-10	`_depositApprovedUnspent()` Not Necessarily Paired With `IMarket.enterMarket()`	● Low	Fixed
EXA-11	Approvals Should Be Reset After Each `_approveFromSender()` and Related Execution	● Low	Fixed

Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Disclaimer
Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.
The fixes included significant changes in logic outside of the identified issues. For example, a plugin replacement flow has been added, the flow of token transfers for some operations has been changed, and the execution hooks have been refactored. These additional logic changes have been reviewed to the extent possible within the constraints of the fix review, but they remain outside the defined scope.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
   1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Scope

The scope of the audit focused on the ExaPlugin, a plugin for ERC-6900 modular accounts that interacts with the Exactly Protocol backend, enabling lending and borrowing functionality with a credit/debit card for the Exa App frontend. Specifically, the audit covered contracts responsible for the plugin's integration, including collateral management, issuer validation, and refund mechanisms, ensuring the integrity of interactions between users, credit issuers, and the Exactly Protocol.

Files Included

```
contracts
├── src
│   ├── ExaAccountFactory.sol
│   ├── ExaPlugin.sol
│   ├── IExaAccount.sol
│   ├── IssuerChecker.sol
│   └── Refunder.sol
```

Files Excluded

```
contracts
├── src
│   ├── InstallmentsPreviewer.sol
│   ├── KeeperFeeModel.sol
```

Operational Considerations

1. The system assumes the Exactly Protocol acts as documented, including the behavior of its price oracles, liquidation mechanisms, and debt management. It is further assumed that the oracles have sufficient staleness checks, ensuring accurate and timely price feeds. The audit scope is limited to the integration of the plugin and the plugin itself and excludes the internal mechanics of the Exactly Protocol.
2. The underlying account system is assumed to be Modular-Account v1 and paired with a secure owner authentication plugin such as the Exactly WebAuthn Plugin to validate execution functions like `execute()` and `executeBatch()`.
3. It is assumed that other installed plugins and external contracts targeted by account calls do not use `delegateCall`. If `delegateCall` were used, it could introduce risks by routing back into a module installed on the account.
4. No further external contracts are assumed to require execution hook checks beyond `IAuditor`, `IMarket`, and `IInstallmentRouter`.
5. The account is not designed to serve as a regular wallet for holding assets accepted as collateral by the Exactly Protocol. Tokens held by the account that are underlying assets in an Exactly market will be deposited to a market by the Keeper address.
6. The `collectCredit()` function, when used without specifying the `maxRepay` parameter, should only be invoked under standard market conditions. In non-standard conditions, there is a risk of unexpectedly high owed amounts without ceiling protection, which could result in undesired behavior.
7. The `_checkLiquidity()` function may encounter gas limit issues if there are a large number of active markets. It is assumed that the system will operate with a manageable number of markets to ensure reliable liquidity checks.
8. Functions like `crossRepay()` and `repay()` depend on Balancer's flash loan mechanics, assuming sufficient USDC liquidity and zero fees. If liquidity becomes insufficient or fees increase, these functions could fail or revert unexpectedly.
9. The plugin interacts with external contracts such as `SWAPPER`, `BALANCER_VAULT`, and `IMarket`. These contracts must not introduce unguarded re-entrancy vulnerabilities that could bypass `_checkLiquidity()` or disrupt the proposal system. Changes in external protocols that add callbacks may necessitate additional re-entrancy protections.
10. The plugin interacts with a `SWAPPER` contract for token swaps via `_approveAndExecuteFromSender()` and assumes it to be a trusted or externally audited DEX aggregator. If the `SWAPPER` contract is compromised or malicious, user funds could be misrouted, bypassing slippage checks or the intended receiver.
11. The `PROPOSAL_DELAY` is set to 1 minute, and `_checkIssuer` uses an `OPERATION_EXPIRY` of 15 minutes. These intervals are assumed to be appropriate for the chain's block times and user interactions. However, changes in network conditions or user behavior may require retuning these parameters to prevent proposal spam or operational stalling.
12. The off-chain credit card infrastructure should continuously monitor active withdrawal proposals to prevent users from simultaneously borrowing on the card and withdrawing collateral. This ensures race conditions leading to under-collateralization are avoided. The off-chain system must reliably track the on-chain Proposal states to ensure no conflicting operations occur until the timelock expires or the proposal is canceled.
13. Off-chain systems must manage scenarios where users abandon or no longer need withdrawal proposals. Stale proposals should be cleared or liquidity re-checked to prevent funds from being locked unnecessarily.
14. The `allowlist` in the `ProposalManager` should be carefully managed and should only contain addresses of the underlying collateral tokens and not, for example, market tokens themselves.

Key Actors And Their Capabilities

DEFAULT_ADMIN_ROLE

Responsibilities

The `DEFAULT_ADMIN_ROLE` is the highest privilege within the protocol, allowing holders to manage other roles, configure essential settings, and maintain overall control of the contract.

Trust Assumptions

- Will manage protocol parameters responsibly without malicious intent.
- Will only assign roles to trustworthy entities and avoid granting unnecessary access.
- Will ensure secure management of the role to prevent unauthorized access.

Exclusive Functions

1. **IssuerChecker.sol**
   - `setIssuer()` : Updates the designated issuer responsible for refund approvals.
   - `setOperationExpiry()` : Configures the expiration time for authorized operations.
   - `setPrevIssuerWindow()` : Adjusts the time window for recognizing previous issuers as valid.
2. **AccessControl**
   - `_grantRole()` : Assigns other roles, such as `KEEPER_ROLE` , to designated addresses.
3. **ExaPlugin.sol**
   - `_setCollector()` : Sets the address of the collector contract responsible for aggregating collateral or protocol funds.

KEEPER_ROLE

Responsibilities

The `KEEPER_ROLE` is responsible for executing operational functions necessary for the protocol's daily activities, such as processing refunds and managing collateral interactions.

Trust Assumptions

- Will facilitate refunds for the user when it is warranted.
- Will interact with markets (e.g., `poke()` and `collectCollateral()`) to maintain protocol functionality.

Exclusive Functions

1. **Refunder.sol**
   - `refund()` : Processes user refunds after they are approved by the issuer. This function transfers the specified amount to the user's account.
2. **ExaPlugin.sol**
   - `collectCollateral()` : Collects collateral from external markets or protocols.
   - `poke()` : Updates the state of a specified market to reflect the latest changes.
   - `pokeETH()` : Specifically updates the state of the WETH market.

Issuer Role

Responsibilities

The issuer represents the entity that provides the user with their card, and fulfils payment obligations from card payments on-chain. For the scope of this audit, together with the Keeper role, they serve as the authority for approving user refunds. They are responsible for determining whether refund requests are legitimate and ensuring compliance with the protocol's policies. Without issuer approval, refunds cannot be processed.

Trust Assumptions

- Issuers will evaluate refund requests fairly and in accordance with protocol guidelines.
- Issuers will not unjustly deny or delay refunds for users who meet the necessary criteria.
- Issuers will act in good faith, ensuring that legitimate refund claims are processed promptly and efficiently.

Exclusive Functions

1. **Refunder.sol**
   - **Refund Approval**: Issuers are responsible for approving refund requests. This approval acts as a prerequisite for the `KEEPER_ROLE` to execute the `refund()` function, ensuring that users receive refunds only when warranted by protocol rules.

Findings

EXA-1

`executeBatch()` Can Bypass the Created Execution Hook Restrictions

● High ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `5caeded5defc937e4580038dd6cc619a477abcde`.

File(s) affected: `ExaPlugin`

Description: Currently the `preExecutionHook()` is applied to the following 3 selectors: `executeFromPluginExternal()`, `execute()` and `executeBatch()`. This hook is used to enforce which addresses can be called and with what parameters. This is done by extracting the target and selector of the call. This works well for the first 2 selectors mentioned but not for `executeBatch()`, as it is made of an array of `Calls[]`. Hence in the current setup only the first entry of that array of calls is checked and if it is valid, all of the entries of the array are approved, which might include not allowed calls such as `exitMarket()` or withdrawals circumventing the `proposal` mechanic.

Recommendation: In the case of `executeBatch()`, all of the array's entries need to be verified via the same checks in the existing execution hooks.

EXA-2
Plugin Can Be Uninstalled, Removing All Checks on Plugin-Side

● High ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `f2dc87c8c4360f7c6a239963fb380e60eafff565` and `b69e45f77e46aa32863c0a603f701bd8370a9c2f`.

File(s) affected: `ExaPlugin`

Description: Currently, the ExaPlugin can simply be uninstalled by a malicious user. This would make the account lose all the restrictions that are being put in place for safe credit card usage, as well as stopping the Keeper role from withdrawing funds or creating a credit on behalf of the user. In particular, this is another vector through which a malicious user can altogether bypass the proposal mechanism, resulting in the ability to perform a double spend exploit at the expense of the issuers.

Exploit Scenario:
1. Uninstall the `ExaPlugin`, bypassing all restrictions.
2. Spend money using the credit card.
3. On-chain monitoring system determines that user is solvent, as user still has sufficient collateral and approves card payment
4. However, the issuer signed Keeper transaction to settle the purchase on-chain fails, as the plugin is uninstalled.
5. The user can withdraw the collateral, e.g. by encoding a custom call to the appropriate market within the `execute()` function, without the Keeper being able to interfere or withdraw any collateral on behalf of the user.
6. The card payment will remain unbacked, leaving the user at a profit.

Recommendation: For proper operation of the system restrictions should be put on the `installPlugin()` and `uninstallPlugin()` selectors. There are multiple ways of going about it via addition of execution hooks. A list of potential mitigation options includes:
1. Only allow uninstallation if the user is completely out of the system, i.e. no collateral and no debt.
2. `installPlugin()` and `uninstallPlugin()` should only work with a signature from the Keeper or be solely callable by the keeper. That way, malicious users can not uninstall the plugin, yet a safe upgrade process is assured.
3. An execution hook could be added to the `installPlugin()` and `uninstallPlugin()` that limits these 2 selectors to be only callable by the keeper

EXA-3
Incorrect Assumptions Enable Numerous Credit Card Purchase Race Conditions

● High ⓘ Fixed

Update
The fix-review featured an extensive update to the proposal mechanism to mitigate the race conditions:

The proposal mechanism in `ExaPlugin` implements a two-step execution pattern where operations (like borrowing, repaying, or swapping) must first be proposed and then executed after a delay period. Each proposal is identified by a nonce and contains the target market, amount, proposal type (e.g., `BORROW_AT_MATURITY`, `CROSS_REPAY_AT_MATURITY`, `ROLL_DEBT`), and type-specific encoded data. The `ProposalManager` contract validates and tracks these proposals, enforcing timelock delays and ensuring proper execution order through nonce management, while runtime validation functions enforce additional security checks based on whether the caller is a keeper or self-executing the proposal.

Update
Marked as "Fixed" by the client.
Addressed in: `bd0222d300f81b14fb3025c44f440958705044eb`.

File(s) affected: `ExaPlugin`

Description: The plugin attempts to guard changes to the collateral from instant changes in order to avoid a crucial race condition between credit card payments and potential on-chain collateral withdrawals. However, the assumptions around interactions with the collateral are incorrect in numerous cases:

- The transfer of share tokens of deposited collateral directly affects the collateral a user has at their disposal. While on a core protocol level, the share transfer has to not lead to unhealthy positions, it does enable the race condition with credit card purchases the `proposal` mechanism attempts to mitigate. This includes `transfer()`, but also a two step process of creating approvals and invoking `transferFrom()` at the time of a credit card purchase. This also includes the `swap()` function, which could swap collateral shares for some other token.
- Plugin-native functions such as `rollDebt()` directly affect the collateral and debt of a user. Therefore, they too should go through the `proposal` mechanic to avoid the race condition. Furthermore, a call to `_checkLiquidity()` should be appended to all of these operations. This includes calls to `withdraw()` and `redeem()`.
- Loan repayment through swaps via `crossRepay()`, should also go through the proposal mechanic, as with the uncontrolled `route` calldata, complex multi-swaps could be performed that swap out the whole collateral deposited for some other token, leaving just enough USDC for the repayment of the loan, effectively withdrawing all the collateral in one go. A proposal including the `proposal` with `amount = maxRepay` should be a necessary step in between loan repayments.

Recommendation: Generally, all functions that affect the collateral of a user's position in any way should invoke the `_checkLiquidity()` at the end and go through the `proposals` mechanic.

Furthermore, we recommend a strong whitelist restriction of the `approve()` selector to only be used on the necessary contracts as part of intended protocol flows. While this affects UX slightly, we believe it is the only way to properly protect the `approve()` + `transferFrom()` flow from credit card purchase race conditions.

EXA-4
Proposal Time Lock Is Skipped in the Case of Ether Withdrawal With No Swap

• Medium ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `05d44af389a081b3fb9d74dfbd6ea28372a61e7b`.

File(s) affected: `ExaPlugin`

Description: Currently, to `withdraw()` any tokens from the system, a user needs to make a proposal first and wait out the proposal period. This is done to avoid race conditions with card payments. This is enforced via a pre-execution hook, but not when `receiver == address(this)`, which occurs in the case where `market == EXA_WETH && proposal.swapData.length == 0` during the withdrawal. Hence, by having no swap data and `EXA_WETH` as the market, the recipient will be the plugin so that the `WETH` can be unwrapped. This action would cause the pre-execution hook to skip the time lock check and allow the withdrawal to occur straight after the proposal is created.

Recommendation: Make sure the time lock is checked during a withdrawal with `market == EXA_WETH && proposal.swapData.length == 0`.

EXA-5
More Selectors Need Restrictions to Ensure Correct Plugin Behaviour

• Medium ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `a04eb0fd7eb422bdd148edb62a96fd55477e5f52`.
The client provided the following explanation:

```
withdraw at maturity doesn't handle collateral
```

File(s) affected: `ExaPlugin`

Description: The plugin has a restriction on what selectors the users can call via the `preExecutionHook()` on all function that can make arbitrary external calls. But there should be more selectors that are covered by those restrictions:

1. `IMarket.withdrawAtMaturity()` to go through `proposal` mechanic.
2. `IInstallmentsRouter.borrow()` to restrict `recipient` to `collector`.

This is, however, not an exhaustive list, and there could be more functions that should also have restrictions.

Recommendation: Calls to these selectors should be verified to be behave as expected.

To make sure the plugin only interacts with the intended functions of the core protocol, we recommend a strict whitelist of both target addresses and selectors that can be invoked on the target address in combination with the restrictions imposed on the specific selectors.

EXA-6 Hashes are not invalidated properly

● Medium ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `024544db95b480f0d16ccca61a5d4f1065d378e5`.

File(s) affected: `IssuerChecker`

Description: The `IssuerChecker` contract is responsible for validating the signed operations that have been performed on the card so that the wallet can be charged correctly. This is done via a call to the `checkIssuer()` with a signed operation that includes the timestamp, the amount, and the address of the wallet. When an operation has been used, it should be invalidated, but currently, only the last operation's hash is stored and is not allowed to be used. This leads to an ability to replay any valid operation that is older than the most recent one.

Note the severity of the issue is slightly mitigated by the fact that only the `Keeper` can call `Refunder.refund()`

Recommendation: Make sure that the operation's hash is invalidated correctly. Such as through the following:

```
    if (issuerOperations[hash] == account) revert AlreadyUsed();

....

     issuerOperations[hash] = account;
```

EXA-7 Refunding Signature Should Differ From the Collecting Signature

● Low ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `95f4ca1b1fc7ba94c14733115561f458a1b7da5a`.

File(s) affected: `IssuerChecker`

Description: The `IssuerChecker` contract can validate operations that perform both collection calls and refund calls on the contract. They are both checked via the `checkIssuer()` call and have the same structure of an `operation` with amount, timestamp, and account. This can lead to 2 issues:

1. A signed operation created to reflect a refund can be used to actually charge the account, as there is no distinction between a collection and a refund in the signed operation structure.
2. If a collection and a refund are issued at the same time, which will depend on the internal setup of the payment processor, there can be a case where the refund cannot be used as this hash has already been invalidated for the collection call, as these 2 calls with have the same hash. Hence, the user will not be able to receive a refund.

Recommendation: We recommend to make these 2 types of signed entities distinct, such as:

1. `Collection(address account,uint256 amount,uint40 timestamp)`
2. `Refund(address account,uint256 amount,uint40 timestamp)`

EXA-8 Malicious Plugin Can Add User Op Validation Functions to the Newly Added Selector

● Low ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `7348bced2a76469493954c8ded5f2a61a2156d5c`.

File(s) affected: `ExaPlugin`

Description: ERC-6900 validation functions have 2 independent flows, one via the user op and the other via runtime. The plugin supports the runtime validation flow; hence, the `runtimeValidationFunction()` is implemented. While the plugin is not expected to be used within the user op validation flow, a malicious plugin could install a validation of this type on the execution selectors that the Exactly plugin has provided, circumventing restrictions imposed on the intended runtime validation, such as making sure that the caller is the Keeper address. This would cause unexpected user flows and could put funds at risk.

Recommendation: Consider adding a `preUserOpValidationHook()` of type `ManifestAssociatedFunctionType.PRE_HOOK_ALWAYS_DENY` on all exposed execution selectors.

EXA-9 Incorrect `operationExpiry` Is Used to Update the Expiry Window

• Low ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `c9a3aa36c8626630d5fd508e11d5b741fde5fe12`.

File(s) affected: `IssuerChecker`

Description: When operations are verified by the `IssuerChecker`, they are checked not to have expired via `timestamp + OPERATION_EXPIRY < block.timestamp`. The contract also has the option to update this window via `_setOperationExpiry(),` but this function updates the `operationExpiry` variable but not the variable that is used, hence doing nothing.

Recommendation: Make sure that the correct `operationExpiry` variable is used in the comparison check.

EXA-10 `_depositApprovedUnspent()` Not Necessarily Paired With `IMarket.enterMarket()`

• Low ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `172fa932d5043964e6c7ec6f4c4eef2e9cc30051`.
The client provided the following explanation:

```
when there's USDC debt, the user already has entered the market
```

File(s) affected: `ExaPlugin`

Description: The `_depositApprovedUnspent()` is used to deposit left-over USDC from swaps by the user back in to the protocol. However, the user might not have a position in that market. Hence, such calls should be combined with `enterMarket()` to make sure that collateral deposited here is included in calculations of overall health of a user's account.

Recommendation: Combine calls to `_depositApprovedUnspent()` with `IMarket.enterMarket()`.

EXA-11 Approvals Should Be Reset After Each `_approveFromSender()` and Related Execution

• Low ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `be037cf918ee77df5bc7c0dbdfefab52463b1c03`.

File(s) affected: `ExaPlugin`

Description: Approvals made on behalf of the account should be reset to zero after the related execution, in case the approval was not fully consumed. This is not the case in `ExaPlugin.repay()` and `crossRepay()`, as potentially less shares are burnt than initially calculated.

Recommendation: Reset the approvals after their intended usage.

Auditor Suggestions

S1 `IMarket` Should Be Used Instead of `IERC4626`

Acknowledged

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Not possible without redeclaring the needed IERC4626 functions to IMarket
```

File(s) affected: `ExaPlugin`

Description: Throughout the codebase, `IMarket` and `IERC4626` interfaces are used interchangeably. To provide consistency in reading the code, we recommend using just `IMarket` as it inherits from `IERC4626`.

Recommendation: Consider using just `IMarket` for selectors in the codebase.

S2 Contract Layout Does Not Follow Solidity Style Guide

Acknowledged

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
The order of events, structs and errors is a style choice to be consistent with the rest of Exactly's
code
```

File(s) affected: `ExaPlugin`, `ExaAccountFactory`, `IssuerChecker`, `Refunder`

Description: The project contract's organization deviates from the standard Solidity style guide, with functions appearing before structures, errors, events, enums, and interfaces. This makes the code harder to read and maintain.

Recommendation: Reorganize contract elements according to the style guide order.

S3 ERC-6900 Gas Optimisations

Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `5ef8ac9455df04a60716bc6660b6272eefd9b0a2`.

File(s) affected: `ExaPlugin`

Description: Currently, certain aspects of the ERC-6900 plugin are not implemented in the most efficient way or are not used. Below is a non-exhaustive list of such places of improvement:

1. A runtime validation function is added to the `receiveFlashLoan()` selector that restricts access only from the Balancer pool. However, this validation function will never be used, as the function is not directly accessed via the account itself but by the plugin. So it can be removed.
2. `preExecutionHook()` has 2 hooks that are applied to the same selectors and are executed one after another, so they can be merged into one.
3. `PRE_EXEC_VALIDATION_AUDITOR_CALL` post-execution hook does not do anything, so it can be removed. Simply providing the pre-execution hook is sufficient.

Recommendation: Consider adding these changes to the codebase to save gas.

S4 Inconsistencies in Variables Shared Between `IssuerChecker` and `ExaPlugin`

Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `c9a3aa36c8626630d5fd508e11d5b741fde5fe12`, `ec9fa429920c665fbc7dc834dfee586cb3296d8b`.

File(s) affected: `ExaPlugin`, `IssuerChecker`

Description: Both `IssuerChecker` and `ExaPlugin` contracts have a variable called `OPERATION_EXPIRY`, which signifies how long a signed operation should be valid for. In `ExaPlugin`, it's a constant hardcoded at 15 minutes, but in `IssuerChecker`, this variable can be updated, hence leading to potential confusion.

`PROPOSAL_DELAY` is also a variable featured in `ExaPlugin`, where it is hardcoded to 1 minute. This constant also appears in `IssuerChecker` but as a magic constant hardcoded to 1 minute.

Recommendation: Consider aligning the variables so it does not lead to confusions. Use proper constants for magic values.

S5 No Reentrancy Guard Around `_swap()` External Call

Fixed

Update
Marked as "Fixed" by the client.
Addressed in the fix review process.

File(s) affected: `ExaPlugin`

Description: The `_swap()` function in `ExaPlugin.sol` invokes `SWAPPER.functionCall(route)`, which executes arbitrary external code. This call lacks a `nonReentrant` modifier or validaiton for the `route` data passed.

Recommendation: Consider adding `nonReentrant` modifiers to sensitive functions, or validate the `route` data passed into the swap function.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Files

- `a96...c4e ./src/Refunder.sol`
- `474...42f ./src/ExaAccountFactory.sol`
- `e24...b6c ./src/ExaPlugin.sol`
- `3dc...552 ./src/IssuerChecker.sol`
- `486...57b ./src/IExaAccount.sol`

Tests

- `ba9...573 ./test/MockSwapper.t.sol`
- `10d...fa1 ./test/ExaAccountFactory.t.sol`
- `ea8...903 ./test/IssuerChecker.t.sol`
- `c60...755 ./test/ExaPlugin.t.sol`
- `e59...f99 ./test/InstallmentsPreviewer.t.sol`

- `c7b...d46 ./test/Refunder.t.sol`
- `e9e...f13 ./test/Fork.t.sol`
- `14f...5b9 ./test/mocks/Protocol.s.sol`
- `221...2ce ./test/mocks/MockVelodromeFactory.sol`
- `e0e...239 ./test/mocks/MockSwapper.sol`
- `816...be9 ./test/mocks/Bob.s.sol`
- `fb4...90b ./test/mocks/Account.s.sol`
- `26e...fc4 ./test/mocks/Mocks.s.sol`

Test Suite Results

All tests are passing with a total of 93 tests. More extensive integration tests involving the interaction between the plugin and the Exactly protocol may have been helpful in uncovering ways to bypass the proposal mechanism.

Fix Review Update

The test suite now stands at 175 tests, all of which pass. It now includes some integration testing.

```
Ran 8 tests for test/IssuerChecker.t.sol:IssuerCheckerTest
[PASS] test_setIssuer_emits_IssuerSet() (gas: 50233)
[PASS] test_setIssuer_reverts_whenNotAdmin() (gas: 18171)
[PASS] test_setIssuer_reverts_whenZeroAddress() (gas: 15580)
[PASS] test_setOperationExpiry_emits_OperationExpirySet() (gas: 25687)
[PASS] test_setOperationExpiry_reverts_whenNotAdmin() (gas: 19897)
[PASS] test_setOperationExpiry_reverts_whenZeroValue() (gas: 17426)
[PASS] test_setPrevIssuerWindow_emits_PrevIssuerWindowSet() (gas: 25756)
[PASS] test_setPrevIssuerWindow_reverts_whenNotAdmin() (gas: 19986)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 31.15ms (29.77ms CPU time)

Ran 1 test for test/InstallmentsPreviewer.t.sol:InstallmentsPreviewerTest
[PASS] test_preview_returns() (gas: 165841)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 38.71ms (532.63µs CPU time)

Ran 2 tests for test/MockSwapper.t.sol:MockSwapperTest
[PASS] test_swapExactAmountIn_swaps() (gas: 171436)
[PASS] test_swapExactAmountOut_swaps() (gas: 170661)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 39.36ms (968.33µs CPU time)

Ran 7 tests for test/Refunder.t.sol:RefunderTest
[PASS] test_refund_refunds() (gas: 225494)
[PASS] test_refund_reverts_whenExpired() (gas: 51132)
[PASS] test_refund_reverts_whenNotKeeper() (gas: 38927)
[PASS] test_refund_reverts_whenReplay() (gas: 238294)
[PASS] test_refund_reverts_whenTimelocked() (gas: 46688)
[PASS] test_refund_reverts_whenWrongSignature() (gas: 54338)
[PASS] test_refund_toleratesTimeDrift() (gas: 225668)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 40.50ms (24.70ms CPU time)

Ran 2 tests for test/ExaAccountFactory.t.sol:ExaAccountFactoryTest
[PASS] testFuzz_createAccount_EOAOwners(uint256,address[63]) (runs: 256, µ: 3928678, ~: 3770411)
[PASS] test_deploy_deploysToSameAddress() (gas: 35307304)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 9.78s (11.22s CPU time)

Ran 73 tests for test/ExaPlugin.t.sol:ExaPluginTest
[PASS] testFork_collectCollateral_collects() (gas: 17656029)
[PASS] testFork_crossRepay_repays() (gas: 17759242)
[PASS] testFork_swap_swaps() (gas: 14955839)
[PASS] test_borrowAtMaturity_reverts_withUnauthorized_whenReceiverNotCollector() (gas: 431865)
[PASS] test_borrow_reverts_withUnauthorized_whenReceiverNotCollector() (gas: 431026)
[PASS] test_collectCollateral_collects() (gas: 870891)
[PASS] test_collectCredit_collects() (gas: 945816)
[PASS] test_collectCredit_collects_whenHealthFactorHigherThanOne() (gas: 773530)
[PASS] test_collectCredit_collects_withEnoughSlippage() (gas: 771528)
[PASS] test_collectCredit_collects_withPrevIssuerSignature() (gas: 997205)
[PASS] test_collectCredit_passes_whenProposalLeavesEnoughLiquidity() (gas: 959799)
[PASS] test_collectCredit_reverts_asNotKeeper() (gas: 386237)
[PASS] test_collectCredit_reverts_whenDisagreement() (gas: 541572)
```

```
[PASS] test_collectCredit_reverts_whenExpired() (gas: 384990)
[PASS] test_collectCredit_reverts_whenHealthFactorLowerThanOne() (gas: 946392)
[PASS] test_collectCredit_reverts_whenPrevSignatureNotValidAnymore() (gas: 464106)
[PASS] test_collectCredit_reverts_whenProposalCausesInsufficientLiquidity() (gas: 955063)
[PASS] test_collectCredit_reverts_whenReplay() (gas: 777004)
[PASS] test_collectCredit_reverts_whenTimelocked() (gas: 380480)
[PASS] test_collectCredit_toleratesTimeDrift() (gas: 773811)
[PASS] test_collectDebit_collects() (gas: 608363)
[PASS] test_collectDebit_collects_whenProposalLeavesEnoughLiquidity() (gas: 789926)
[PASS] test_collectDebit_reverts_asNotKeeper() (gas: 386145)
[PASS] test_collectDebit_reverts_whenExpired() (gas: 384717)
[PASS] test_collectDebit_reverts_whenProposalCausesInsufficientLiquidity() (gas: 779783)
[PASS] test_collectDebit_reverts_whenReplay() (gas: 611354)
[PASS] test_collectDebit_reverts_whenTimelocked() (gas: 380276)
[PASS] test_collectDebit_toleratesTimeDrift() (gas: 608515)
[PASS] test_collectInstallments_collects() (gas: 1432219)
[PASS] test_collectInstallments_revertsWhenNoSlippage() (gas: 1265123)
[PASS] test_collectInstallments_reverts_asNotKeeper() (gas: 386584)
[PASS] test_collectInstallments_reverts_whenExpired() (gas: 386811)
[PASS] test_collectInstallments_reverts_whenReplay() (gas: 1040537)
[PASS] test_collectInstallments_reverts_whenTimelocked() (gas: 382454)
[PASS] test_collectInstallments_toleratesTimeDrift() (gas: 1225340)
[PASS] test_crossRepay_repays() (gas: 1407525)
[PASS] test_crossRepay_repays_whenKeeper() (gas: 1400320)
[PASS] test_crossRepay_reverts_whenDisagreement() (gas: 1410015)
[PASS] test_crossRepay_reverts_whenNotKeeper() (gas: 964087)
[PASS] test_exitMarket_reverts() (gas: 414677)
[PASS] test_marketWithdraw_transfersAsset_asOwner() (gas: 736950)
[PASS] test_onUninstall_uninstalls() (gas: 402698)
[PASS] test_poke() (gas: 364657)
[PASS] test_pokeETH_deposits() (gas: 452651)
[PASS] test_proposeSwap_emitsSwapProposed() (gas: 375757)
[PASS] test_propose_emitsProposed() (gas: 214542)
[PASS] test_refund_refunds() (gas: 378848)
[PASS] test_repay_partiallyRepays() (gas: 1210707)
[PASS] test_repay_partiallyRepays_whenKeeper() (gas: 1140465)
[PASS] test_repay_repays() (gas: 1051097)
[PASS] test_repay_repays_whenKeeper() (gas: 981630)
[PASS] test_rollDebt_rolls() (gas: 1147604)
[PASS] test_rollDebt_rolls_asKeeper() (gas: 1145184)
[PASS] test_setCollector_emitsCollectorSet() (gas: 20265)
[PASS] test_setCollector_reverts_whenAddressZero() (gas: 11559)
[PASS] test_setCollector_reverts_whenNotAdmin() (gas: 14041)
[PASS] test_setCollector_sets_whenAdmin() (gas: 19597)
[PASS] test_swap_reverts_withDisagreement() (gas: 294099)
[PASS] test_swap_swaps() (gas: 310155)
[PASS] test_withdraw_reverts_whenNoProposal() (gas: 436300)
[PASS] test_withdraw_reverts_whenNoProposalKeeper() (gas: 378649)
[PASS] test_withdraw_reverts_whenNotKeeper() (gas: 377898)
[PASS] test_withdraw_reverts_whenReceiverIsContractAndMarketNotWETH() (gas: 590233)
[PASS] test_withdraw_reverts_whenTimelocked() (gas: 244145)
[PASS] test_withdraw_reverts_whenTimelockedKeeper() (gas: 270809)
[PASS] test_withdraw_reverts_whenWrongAmount() (gas: 246235)
[PASS] test_withdraw_reverts_whenWrongMarket() (gas: 249858)
[PASS] test_withdraw_reverts_whenWrongReceiver() (gas: 245859)
[PASS] test_withdraw_swapsProposed() (gas: 1532793)
[PASS] test_withdraw_swapsWETH() (gas: 1579583)
[PASS] test_withdraw_transfersAsset_asKeeper() (gas: 745403)
[PASS] test_withdraw_transfersETH() (gas: 824987)
[PASS] test_withdraw_withdrawsProposed() (gas: 737912)
Suite result: ok. 73 passed; 0 failed; 0 skipped; finished in 112.97s (113.41s CPU time)

Ran 6 test suites in 113.10s (122.90s CPU time): 93 tests passed, 0 failed, 0 skipped (93 total tests)

**Fix Review**

Ran 8 tests for test/IssuerChecker.t.sol:IssuerCheckerTest
[PASS] test_setIssuer_emits_IssuerSet() (gas: 70595)
[PASS] test_setIssuer_reverts_whenNotAdmin() (gas: 37043)
[PASS] test_setIssuer_reverts_whenZeroAddress() (gas: 35466)
[PASS] test_setOperationExpiry_emits_OperationExpirySet() (gas: 46742)
[PASS] test_setOperationExpiry_reverts_whenNotAdmin() (gas: 38913)
```

```
[PASS] test_setOperationExpiry_reverts_whenZeroValue() (gas: 37400)
[PASS] test_setPrevIssuerWindow_emits_PrevIssuerWindowSet() (gas: 46777)
[PASS] test_setPrevIssuerWindow_reverts_whenNotAdmin() (gas: 38890)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 222.88ms (336.18ms CPU time)

Ran 2 tests for test/MockSwapper.t.sol:MockSwapperTest
[PASS] test_swapExactAmountIn_swaps() (gas: 220971)
[PASS] test_swapExactAmountOut_swaps() (gas: 220939)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 229.55ms (524.25µs CPU time)

Ran 7 tests for test/ExaPreviewer.t.sol:ExaPreviewerTest
[PASS] test_collectCollateral_collects() (gas: 1115414)
[PASS] test_collectInstallments_collects() (gas: 1153587)
[PASS] test_collect_reverts_whenProposalsHaveTooMuchDebt() (gas: 2069795)
[PASS] test_collect_reverts_whenProposalsHaveTooMuchRollDebt() (gas: 874362)
[PASS] test_collect_reverts_whenProposalsLeaveNoLiquidity() (gas: 1378233)
[PASS] test_pendingProposals_returnsPendingProposals() (gas: 1867848)
[PASS] test_utilizations_returns() (gas: 135008)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 304.53ms (192.00ms CPU time)

Ran 9 tests for test/Refunder.t.sol:RefunderTest
[PASS] test_refund_refunds() (gas: 246451)
[PASS] test_refund_reverts_whenExpired() (gas: 73735)
[PASS] test_refund_reverts_whenNotKeeper() (gas: 56679)
[PASS] test_refund_reverts_whenReplay() (gas: 285538)
[PASS] test_refund_reverts_whenTimelocked() (gas: 63242)
[PASS] test_refund_reverts_whenWrongSignature() (gas: 95260)
[PASS] test_refund_toleratesTimeDrift() (gas: 246582)
[PASS] test_withdraw_reverts_whenNotAdmin() (gas: 35768)
[PASS] test_withdraw_withdraws() (gas: 68535)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 304.86ms (316.41ms CPU time)

Ran 2 tests for test/ExaAccountFactory.t.sol:ExaAccountFactoryTest
[PASS] testFuzz_createAccount_EOAOwners(uint256,address[63]) (runs: 256, µ: 4009198, ~: 3894500)
[PASS] test_deploy_deploysToSameAddress() (gas: 30084541)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 2.78s (3.40s CPU time)
^C
root@0b46573d38ff:~/workspace/projects/Exactly/exactly-mobile-master-github~full/contracts# forge test
[·] Compiling...
No files changed, compilation skipped

Ran 8 tests for test/IssuerChecker.t.sol:IssuerCheckerTest
[PASS] test_setIssuer_emits_IssuerSet() (gas: 70595)
[PASS] test_setIssuer_reverts_whenNotAdmin() (gas: 37043)
[PASS] test_setIssuer_reverts_whenZeroAddress() (gas: 35466)
[PASS] test_setOperationExpiry_emits_OperationExpirySet() (gas: 46742)
[PASS] test_setOperationExpiry_reverts_whenNotAdmin() (gas: 38913)
[PASS] test_setOperationExpiry_reverts_whenZeroValue() (gas: 37400)
[PASS] test_setPrevIssuerWindow_emits_PrevIssuerWindowSet() (gas: 46777)
[PASS] test_setPrevIssuerWindow_reverts_whenNotAdmin() (gas: 38890)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 247.20ms (430.56ms CPU time)

Ran 2 tests for test/MockSwapper.t.sol:MockSwapperTest
[PASS] test_swapExactAmountIn_swaps() (gas: 220971)
[PASS] test_swapExactAmountOut_swaps() (gas: 220939)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 255.96ms (851.08µs CPU time)

Ran 7 tests for test/ExaPreviewer.t.sol:ExaPreviewerTest
[PASS] test_collectCollateral_collects() (gas: 1115414)
[PASS] test_collectInstallments_collects() (gas: 1153587)
[PASS] test_collect_reverts_whenProposalsHaveTooMuchDebt() (gas: 2069795)
[PASS] test_collect_reverts_whenProposalsHaveTooMuchRollDebt() (gas: 874362)
[PASS] test_collect_reverts_whenProposalsLeaveNoLiquidity() (gas: 1378233)
[PASS] test_pendingProposals_returnsPendingProposals() (gas: 1867848)
[PASS] test_utilizations_returns() (gas: 135008)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 320.39ms (154.38ms CPU time)

Ran 9 tests for test/Refunder.t.sol:RefunderTest
[PASS] test_refund_refunds() (gas: 246451)
[PASS] test_refund_reverts_whenExpired() (gas: 73735)
[PASS] test_refund_reverts_whenNotKeeper() (gas: 56679)
[PASS] test_refund_reverts_whenReplay() (gas: 285538)
```

```
[PASS] test_refund_reverts_whenTimelocked() (gas: 63242)
[PASS] test_refund_reverts_whenWrongSignature() (gas: 95260)
[PASS] test_refund_toleratesTimeDrift() (gas: 246582)
[PASS] test_withdraw_reverts_whenNotAdmin() (gas: 35768)
[PASS] test_withdraw_withdraws() (gas: 68535)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 320.71ms (288.23ms CPU time)

Ran 2 tests for test/ExaAccountFactory.t.sol:ExaAccountFactoryTest
[PASS] testFuzz_createAccount_EOAOwners(uint256,address[63]) (runs: 256, μ: 4118789, ~: 4100597)
[PASS] test_deploy_deploysToSameAddress() (gas: 30084541)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 1.31s (1.69s CPU time)

Ran 175 tests for test/ExaPlugin.t.sol:ExaPluginTest
[PASS] testFork_claimAndVestEscrowedEXA_claimsAndVests() (gas: 35963079)
[PASS] testFork_collectCollateral_collects() (gas: 30184071)
[PASS] testFork_crossRepay_repays() (gas: 31912715)
[PASS] testFork_swap_swaps() (gas: 26284675)
[PASS] test_allowPlugin_emitsPluginAllowed() (gas: 58612)
[PASS] test_allowPlugin_reverts_whenAddressZero() (gas: 32427)
[PASS] test_allowPlugin_reverts_whenNotAdmin() (gas: 34005)
[PASS] test_allowPlugin_sets_whenAdmin() (gas: 57883)
[PASS] test_allowTarget_emitsTargetAllowed() (gas: 83073)
[PASS] test_allowTarget_reverts_whenAddressZero() (gas: 32410)
[PASS] test_allowTarget_reverts_whenNotAdmin() (gas: 38404)
[PASS] test_allowTarget_sets_whenAdmin() (gas: 54664)
[PASS] test_approveMarket_reverts_whenNotAllowlisted() (gas: 95197)
[PASS] test_auditorCalls_passes() (gas: 117775)
[PASS] test_borrowAtMaturity_borrows() (gas: 1134210)
[PASS] test_borrowAtMaturity_incrementsNonce() (gas: 1151086)
[PASS] test_borrowAtMaturity_reverts_whenProposalTypeIsWithdraw() (gas: 638357)
[PASS] test_borrowAtMaturity_reverts_whenTimelocked() (gas: 692627)
[PASS] test_borrowAtMaturity_reverts_withNoProposal_whenDataDiffers() (gas: 696950)
[PASS] test_borrowAtMaturity_reverts_withNoProposal_whenReceiverNotCollector() (gas: 389580)
[PASS] test_borrowToCollector_borrows() (gas: 696444)
[PASS] test_borrow_reverts_withUnauthorized_whenReceiverNotCollector() (gas: 384578)
[PASS] test_collectCollateral_collects() (gas: 755228)
[PASS] test_collectCollateral_reverts_withDisagreement() (gas: 797511)
[PASS] test_collectCredit_collects() (gas: 843028)
[PASS] test_collectCredit_collects_whenHealthFactorHigherThanOne() (gas: 736217)
[PASS] test_collectCredit_collects_whenProposalCausesInsufficientLiquidity() (gas: 997448)
[PASS] test_collectCredit_collects_whenProposalLeavesHealthFactorLowerThanOne() (gas: 992001)
[PASS] test_collectCredit_collects_withEnoughSlippage() (gas: 734732)
[PASS] test_collectCredit_collects_withPrevIssuerSignature() (gas: 919252)
[PASS] test_collectCredit_passes_whenProposalLeavesEnoughLiquidity() (gas: 1002055)
[PASS] test_collectCredit_reverts_asNotKeeper() (gas: 340493)
[PASS] test_collectCredit_reverts_whenDisagreement() (gas: 537653)
[PASS] test_collectCredit_reverts_whenExpired() (gas: 344004)
[PASS] test_collectCredit_reverts_whenPrevSignatureNotValidAnymore() (gas: 463926)
[PASS] test_collectCredit_reverts_whenReplay() (gas: 779942)
[PASS] test_collectCredit_reverts_whenTimelocked() (gas: 332829)
[PASS] test_collectCredit_toleratesTimeDrift() (gas: 738153)
[PASS] test_collectDebit_collects() (gas: 593266)
[PASS] test_collectDebit_collects_whenProposalCausesInsufficientLiquidity() (gas: 852734)
[PASS] test_collectDebit_collects_whenProposalLeavesEnoughLiquidity() (gas: 854153)
[PASS] test_collectDebit_reverts_asNotKeeper() (gas: 340236)
[PASS] test_collectDebit_reverts_whenExpired() (gas: 343800)
[PASS] test_collectDebit_reverts_whenReplay() (gas: 634864)
[PASS] test_collectDebit_reverts_whenTimelocked() (gas: 332562)
[PASS] test_collectDebit_toleratesTimeDrift() (gas: 593330)
[PASS] test_collectInstallments_collects() (gas: 1230889)
[PASS] test_collectInstallments_revertsWhenNoSlippage() (gas: 1174042)
[PASS] test_collectInstallments_reverts_asNotKeeper() (gas: 341322)
[PASS] test_collectInstallments_reverts_whenExpired() (gas: 346207)
[PASS] test_collectInstallments_reverts_whenReplay() (gas: 978207)
[PASS] test_collectInstallments_reverts_whenTimelocked() (gas: 335088)
[PASS] test_collectInstallments_toleratesTimeDrift() (gas: 1100000)
[PASS] test_collect_collects_whenProposalsLeaveNoLiquidity() (gas: 1721038)
[PASS] test_collect_collects_whenTooMuchProposedDebt() (gas: 1801798)
[PASS] test_crossRepay_consumesProposal() (gas: 2029187)
[PASS] test_crossRepay_repays() (gas: 2042444)
[PASS] test_crossRepay_repays_whenKeeper() (gas: 2041536)
[PASS] test_crossRepay_reverts_whenDisagreement() (gas: 2127408)
```

```
[PASS] test_crossRepay_reverts_whenNotKeeper() (gas: 1360975)
[PASS] test_debtManagerCalls_revert_withUnauthorized_whenSelectorNotRollFixed() (gas: 74434)
[PASS] test_exaPluginConstructor_reverts_whenSwapperIsZeroAddress() (gas: 618160)
[PASS] test_executeBatch_checksEveryCall() (gas: 243145)
[PASS] test_executeBatch_reverts_whenUnauthorizedCallsNested() (gas: 152657)
[PASS] test_executeBatch_reverts_whenWithdrawingMultipleWithOneProposal() (gas: 720406)
[PASS] test_executeBatch_supportsMultipleProposals() (gas: 1589959)
[PASS] test_executeProposal_reverts_whenInvalidNonce() (gas: 298670)
[PASS] test_exitMarket_reverts() (gas: 367428)
[PASS] test_forceUninstall_reverts_withUnauthorized_withoutExecute() (gas: 72031)
[PASS] test_install_installs_whenNotRelatedPlugin() (gas: 1481657)
[PASS] test_install_reverts_whenAlreadyInstalled() (gas: 5920452)
[PASS] test_installmentsRouterCalls_revert_withUnauthorized_whenSelectorNotBorrow() (gas: 75802)
[PASS] test_issuerChecker_emits_collected() (gas: 835951)
[PASS] test_issuerChecker_emits_refunded() (gas: 355148)
[PASS] test_marketWithdraw_transfersAsset_asOwner() (gas: 894018)
[PASS] test_pluginMetadata() (gas: 12946)
[PASS] test_poke() (gas: 293526)
[PASS] test_pokeETH_deposits() (gas: 349875)
[PASS] test_poke_reverts_withNoBalance() (gas: 126884)
[PASS] test_poke_reverts_withNotMarket() (gas: 63342)
[PASS] test_postExecutionHook_reverts_withNotImplemented() (gas: 30739)
[PASS] test_preExecutionHook_reverts_withNotImplemented() (gas: 33588)
[PASS] test_proposalManagerConstructor_reverts_whenSwapperIsZeroAddress() (gas: 296731)
[PASS] test_proposeRepay_reverts_whenWrongType() (gas: 53632)
[PASS] test_proposeSwap_emitsSwapProposed() (gas: 399376)
[PASS] test_proposeWithdraw_reverts_whenKeeper() (gas: 331936)
[PASS] test_proposeZeroAmount_reverts_withZeroAmount() (gas: 40895)
[PASS] test_propose_emitsProposed() (gas: 281993)
[PASS] test_propose_incrementsQueueNonces() (gas: 178311)
[PASS] test_propose_reverts_whenNotMarket() (gas: 895212)
[PASS] test_propose_reverts_whenNotProposer() (gas: 39603)
[PASS] test_redeem_reverts_whenNoProposalAndReceiverIsProposer() (gas: 437168)
[PASS] test_redeem_reverts_whenProposalTypeIsWithdraw() (gas: 641598)
[PASS] test_redeem_withdraws() (gas: 901314)
[PASS] test_redeem_withdraws_whenExecuteProposal() (gas: 946214)
[PASS] test_refund_refunds() (gas: 365416)
[PASS] test_refund_refunds_whenCollectedAtSameTime() (gas: 907671)
[PASS] test_repay_consumesProposal() (gas: 1303328)
[PASS] test_repay_partiallyRepays() (gas: 1351006)
[PASS] test_repay_partiallyRepays_whenKeeper() (gas: 1325195)
[PASS] test_repay_repaysX() (gas: 1327760)
[PASS] test_repay_repays_whenKeeper() (gas: 1291144)
[PASS] test_rollDebt_rollsX() (gas: 1507718)
[PASS] test_rollDebt_rolls_asKeeper() (gas: 1509901)
[PASS] test_rollFixed_reverts__withNoProposal_proposalTypeDiffers() (gas: 340571)
[PASS] test_rollFixed_reverts_whenTimelocked() (gas: 342720)
[PASS] test_rollFixed_reverts_withNoProposal_whenRollDataDiffers() (gas: 421790)
[PASS] test_runtimeValidationFunction_reverts_withNotImplemented() (gas: 9503)
[PASS] test_setCollector_emitsCollectorSet() (gas: 40604)
[PASS] test_setCollector_reverts_whenAddressZero() (gas: 32182)
[PASS] test_setCollector_reverts_whenNotAdmin() (gas: 33758)
[PASS] test_setCollector_sets_whenAdmin() (gas: 39863)
[PASS] test_setDelay_emitsDelaySet() (gas: 39529)
[PASS] test_setDelay_reverts_whenHigherThanOneHours() (gas: 32148)
[PASS] test_setDelay_reverts_whenNotAdmin() (gas: 38180)
[PASS] test_setDelay_reverts_whenZero() (gas: 32111)
[PASS] test_setDelay_sets_whenAdmin() (gas: 39131)
[PASS] test_setFlashLoaner_emitsFlashLoanerSet() (gas: 40562)
[PASS] test_setFlashLoaner_reverts_whenAddressZero() (gas: 32160)
[PASS] test_setFlashLoaner_reverts_whenNotAdmin() (gas: 33746)
[PASS] test_setFlashLoaner_sets_whenAdmin() (gas: 39791)
[PASS] test_setNonce_reverts_whenNonceTooHigh() (gas: 41660)
[PASS] test_setNonce_reverts_whenNonceTooLow() (gas: 39402)
[PASS] test_setNonce_reverts_whenNotProposer() (gas: 35883)
[PASS] test_setProposalManager_emitsProposalManagerSet() (gas: 40559)
[PASS] test_setProposalManager_reverts_whenAddressZero() (gas: 32183)
[PASS] test_setProposalManager_reverts_whenNotAdmin() (gas: 33716)
[PASS] test_setProposalManager_sets_whenAdmin() (gas: 39821)
[PASS] test_setProposalNonce_emitsEvent() (gas: 324565)
[PASS] test_setProposalNonce_sets() (gas: 1176306)
[PASS] test_setSwapper_emitsSwapperSet() (gas: 40605)
```

```
[PASS] test_setSwapper_reverts_whenAddressZero() (gas: 32185)
[PASS] test_setSwapper_reverts_whenNotAdmin() (gas: 33716)
[PASS] test_setSwapper_sets_whenAdmin() (gas: 39798)
[PASS] test_skippingProposal_reverts_withNoProposal() (gas: 1997037)
[PASS] test_swap_reverts_whenAssetInIsMarket() (gas: 374101)
[PASS] test_swap_reverts_whenCallerIsNotSelf() (gas: 52147)
[PASS] test_swap_reverts_withDisagreement() (gas: 300487)
[PASS] test_swap_swaps() (gas: 274621)
[PASS] test_targetNotAllowed_reverts_withUnauthorized() (gas: 1085945)
[PASS] test_transferShares_reverts_whenNoRightProposal() (gas: 634771)
[PASS] test_transferShares_transfers_whenRedeemProposalPresent() (gas: 781852)
[PASS] test_uninstallAndInstall_installs_whenPluginIsAllowed() (gas: 648560)
[PASS] test_uninstall_reverts_whenExecuteBatchInsideExecute() (gas: 107003)
[PASS] test_uninstall_reverts_whenInsideExecuteOfExecuteBatch() (gas: 148957)
[PASS] test_uninstall_reverts_whenPendingProposals() (gas: 555987)
[PASS] test_uninstall_reverts_whitUnauthorized_whenNextCallIsNotInstall() (gas: 74596)
[PASS] test_uninstall_reverts_withUnauthorized_whenExecute() (gas: 63529)
[PASS] test_uninstall_reverts_withUnauthorized_whenInsideBatch() (gas: 66535)
[PASS] test_uninstall_reverts_withUnauthorized_whenNotAllowedPlugin() (gas: 149007)
[PASS] test_uninstall_reverts_withUnauthorized_withoutExecute() (gas: 70602)
[PASS] test_uninstall_uninstalls_whenBatchedWithAnotherPlugin() (gas: 931200)
[PASS] test_uninstall_uninstalls_whenExecuteWithAnotherPlugin() (gas: 926343)
[PASS] test_uninstall_uninstalls_whenItsAnotherPlugin() (gas: 900043)
[PASS] test_uninstall_uninstalls_whenWrongProposalManager() (gas: 632218)
[PASS] test_userOpValidationFunction_reverts_withBadPlugin() (gas: 1806391)
[PASS] test_withdrawProposed_emits_proposalNonceSet() (gas: 892298)
[PASS] test_withdrawProposed_incrementsNonce() (gas: 891090)
[PASS] test_withdraw_reverts_whenNoProposal() (gas: 442853)
[PASS] test_withdraw_reverts_whenNoProposalAndReceiverIsProposer() (gas: 439465)
[PASS] test_withdraw_reverts_whenNoProposalKeeper() (gas: 399831)
[PASS] test_withdraw_reverts_whenNotKeeper() (gas: 337887)
[PASS] test_withdraw_reverts_whenProposalTypeIsBorrowAtMaturity() (gas: 644108)
[PASS] test_withdraw_reverts_whenProposalTypeIsRedeem() (gas: 644118)
[PASS] test_withdraw_reverts_whenReceiverIsContractAndMarketNotWETH() (gas: 696205)
[PASS] test_withdraw_reverts_whenTimelocked() (gas: 402042)
[PASS] test_withdraw_reverts_whenTimelockedETH() (gas: 647510)
[PASS] test_withdraw_reverts_whenTimelockedKeeper() (gas: 345324)
[PASS] test_withdraw_reverts_whenWrongAmount() (gas: 403697)
[PASS] test_withdraw_reverts_whenWrongMarket() (gas: 401838)
[PASS] test_withdraw_reverts_whenWrongReceiver() (gas: 405657)
[PASS] test_withdraw_swapsProposed() (gas: 1637750)
[PASS] test_withdraw_swapsWETH() (gas: 1652263)
[PASS] test_withdraw_transfersETH() (gas: 882606)
[PASS] test_withdraw_withdrawsProposed() (gas: 866946)
Suite result: ok. 175 passed; 0 failed; 0 skipped; finished in 57.29s (57.20s CPU time)
```

Code Coverage

Initial Coverage

Coverage is exceptionally high in all metrics, with the only recommendation being to improve the branch coverage of `ExaPlugin` to at least 90%.

```
File                           | Lines      | Stmts      | Branches   | Funcs
-------------------------------|------------|------------|------------|----------
ExaAccountFactory.sol          | 100%       | 100%       | 100%       | 100%
ExaPlugin.sol                  | 95.27%     | 94.17%     | 85.11%     | 97.44%
IssuerChecker.sol              | 100%       | 100%       | 100%       | 100%
Refunder.sol                   | 100%       | 100%       | 100%       | 100%
```

Fix Review Update

Coverage remained very high (at least 90%) and `ExaPlugin` coverage has been improved.

```
File                           | Lines      | Stmts      | Branches   | Funcs
-------------------------------|------------|------------|------------|----------
ExaAccountFactory.sol          | 100%       | 100%       | 100%       | 100%
ExaPlugin.sol                  | 99.66%     | 99.73%     | 98.08%     | 100%
ExaPreviewer.sol               | 100%       | 100%       | 100%       | 100%
IssuerChecker.sol              | 100%       | 100%       | 100%       | 100%
ProposalManager.sol            | 100%       | 100%       | 100%       | 100%
Refunder.sol                   | 100%       | 100%       | 100%       | 100%
```

Changelog

- 2025-01-27 - Initial report
- 2025-04-01 - Final report

Quantstamp