PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Cian (Ethereum)

24 September 2022

paladinsec.co   info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Paladin is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. Paladin may, at its discretion, claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for Cian's contracts on the Ethereum network. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

This audit is an extension from the Avalanche audit. All acknowledged issues remain valid in this audit.

1.1 Summary

Project Name	Cian
URL	https://cian.app/
Network	Ethereum
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
AdapterBase	Dependency	✓ MATCH
OneInchAdapter	0x601954e6AfB77Dac21503DbDfA751fbef9eE5374	✓ MATCH
WethGateway	0xc397df95d7313159b667c58A541201BD936a2aA3	✓ MATCH
AaveAdapter	0x5b465489FF729f73ec911245A84B25231b5824bA	✓ MATCH
CurvesteCRVAdapter	0xD896bf804c01c4C0Fa5C42bF6A4b15C465009481	✓ MATCH
FeeBoxETH	0x0b20d5d59E14C71a948D55439019a2Aaf74Fa7B4	✓ MATCH
FeeBoxStETH	0xC5C9953516635659e03345738D8390b7ada6351c	✓ MATCH
VerifierBasic	Dependency	✓ MATCH
LidoAdapter	0xD3812219eb241053F9cf2b43f9B367c0b28E03DA	✓ MATCH
ParaswapAdapter	0x9aa8b1998B1882008c407fbB5BF775A5E2d8e544	✓ MATCH
AdapterManager	0xc936161B3C80494172ae58734e3CE16e26D493C1	✓ MATCH
AccountManager	Not Deployed	N/A
Automation	0x53C8bF6875C66E8d7C42e30BeeF7e6241997F7e3	✓ MATCH
AutomationCallable	Dependency	✓ MATCH
ControllerLib	0x74D2Bef5Afe200DaCC76FE2D3C4022435b54CdbB	✓ MATCH
ControllerLibSub	0x68041721C81c695B72495F78BeaC4F7DFD7b19c8	✓ MATCH
ControllerLink	0xb329504622bd79329c6F82CF8c60c807dF2090c4	✓ MATCH
BalancerERC3156	0xa958090601E21A82e9873042652e35891D945a8C	✓ MATCH
ERC2612Verifier	0x045969904402F5e674ef1f27713F3230929538DF	✓ MATCH
TokenApprovalVerifier	0xfC3A513036CCD84986c1b74e2Dba471Ef417de71	✓ MATCH
Timelock	0xb39e6f93cff9Af7011810f41a4ed9b14582019b7	✓ MATCH
TimelockCallable	Dependency	✓ MATCH
AddressArrayLib	Dependency	✓ MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
🟥 High	0	-	-	-
🟧 Medium	10	7	1	2
🟨 Low	5	2	-	3
🟪 Informational	20	12	1	7
Total	**35**	**21**	**2**	**12**

Classification of Issues

Severity	Description
🟥 High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
🟧 Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
🟨 Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
🟪 Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 Global Issues

ID	Severity	Summary	Status
01	INFO	Typographical errors	ACKNOWLEDGED

1.3.2 AdapterBase

ID	Severity	Summary	Status
02	MEDIUM	Adapters will not fail if a wrong function is called	PARTIAL

1.3.3 OneInchAdapter

No issues found.

1.3.4 WethGateway

No issues found.

1.3.5 AaveAdapter

ID	Severity	Summary	Status
03	MEDIUM	The `exchangeDebt` function could be delegatecalled directly by any contract	✓ RESOLVED
04	MEDIUM	The `positionTransfer` function does not allow the transfer of stable rate borrowings	ACKNOWLEDGED
05	MEDIUM	The `positionTransfer` function does not refund enough tokens at the end of the flashloan	✓ RESOLVED
06	MEDIUM	The reentrancy check during flashloan is null and void	✓ RESOLVED
07	LOW	Ether can get stuck in the contract during a deposit	✓ RESOLVED
08	LOW	The `initialize` function could be used to add malicious bad contracts	✓ RESOLVED
09	INFO	The `positionTransfer` function can only be called using the multicall function	ACKNOWLEDGED
10	INFO	Unused imports and variables	PARTIAL
11	INFO	Typographical errors	✓ RESOLVED
12	INFO	`positionTransfer` could run out of gas	ACKNOWLEDGED

1.3.6 CurvesteCRVAdapter

ID	Severity	Summary	Status
13	INFO	Unused imports and variables	✓ RESOLVED
14	INFO	Lack of events for all functions	✓ RESOLVED

1.3.7 FeeBoxETH

ID	Severity	Summary	Status
15	INFO	Typographical errors	✓ RESOLVED
16	INFO	Gas optimizations	ACKNOWLEDGED

1.3.8 FeeBoxStETH

ID	Severity	Summary	Status
17	MEDIUM	Fees are incorrectly transferred back to the user	✓ RESOLVED
18	INFO	Typographical error	✓ RESOLVED
19	INFO	Lack of proper message during edge cases	ACKNOWLEDGED

1.3.9 VerifierBasic

ID	Severity	Summary	Status
20	MEDIUM	The `verifierBasic` contract is potentially vulnerable	✓ RESOLVED

1.3.10 LidoAdapter

ID	Severity	Summary	Status
21	MEDIUM	referral is only set within the logic contract	✓ RESOLVED

1.3.11 ParaswapAdapter

No issues found.

1.3.12 AdapterManager

No issues found.

1.3.13 AccountManager

ID	Severity	Summary	Status
22	LOW	Authorized addresses are difficult to query	ACKNOWLEDGED
23	INFO	`accountNum` is a duplicate of the length function	✓ RESOLVED
24	INFO	The constructor does not emit the expected events	✓ RESOLVED
25	INFO	Lack of validation	✓ RESOLVED
26	INFO	Unused import	✓ RESOLVED
27	INFO	Typographical errors	✓ RESOLVED

1.3.14 Automation

No issues found.

1.3.15 AutomationCallable

No issues found.

1.3.16 ControllerLib

ID	Severity	Summary	Status
28	MEDIUM	The `multicall` function might fail or send too much Ether	✓ RESOLVED
29	MEDIUM	Privilege escalation: The approve functions allow the bypassing of the `advancedTradingEnable` boolean	ACKNOWLEDGED
30	INFO	Typographical error	✓ RESOLVED

1.3.17 ControllerLibSub

ID	Severity	Summary	Status
31	INFO	`implementationAddress` must be made immutable for the value to be set correctly on the proxy	✓ RESOLVED

1.3.18 ControllerLink

No issues found.

1.3.19 BalancerERC3156

ID	Severity	Summary	Status
32	LOW	`maxFlashLoan` returns a wrong value	ACKNOWLEDGED
33	LOW	The reentrancy check is flawed	ACKNOWLEDGED
34	INFO	`vault` can be made constant	ACKNOWLEDGED

1.3.20 ERC2612Verifier

No issues found.

1.3.21 TokenApprovalVerifier

No issues found.

1.3.22 Timelock

No issues found.

1.3.23 TimelockCallable

No issues found.

1.3.24 AddressArrayLib

ID	Severity	Summary	Status
35	INFO	AddressArrayLib is unused	ACKNOWLEDGED

2 Findings

2.1 Global Issues

The issues in this section occur across multiple contracts within the protocol.

2.1.1 Issues & Recommendations

Issue #01	Typographical errors
Severity	● INFORMATIONAL

Description	We have consolidated the typographical errors into a single issue to keep the report brief and readable.
	
	`ProxyWallet::40 (example for variables)`
	`address public immutable userDatabase;`
	
	`ProxyWallet::76 (example for parameters)`
	`function proxyAdminCheck(address defaultProxyAdmin)`
	
	Throughout the codebase, tokens and other contracts are almost never cast to their correct type. This requires the developer to then explicitly cast them to `IERC20`, `IControllerLink`, `IAdapterManager`, etc. The developer should consider always immediately specifying the types as the correct types instead of using the generic "`address`" type. Although this will not affect gas usage, it heavily simplifies the codebase and also indicates to third parties that the developer has a good understanding of solidity best practice.
	
	`pragma solidity >=0.8.0 <0.9.0;`
	
	This can be simplified to `pragma solidity ^0.8.0` which restricts the version to 0.8 compatible versions as well.

Recommendation	Consider fixing the typographical errors.
Resolution	● ACKNOWLEDGED

2.2 Adapters/AdapterBase

This is the code for the `AdapterBase` contract, which is an abstract contract that defines a basic adapter template. The contract is `Ownable`, which means that it has an owner address that can be used to control access to the contract's functions. The contract is also `TimelockCallable`, which means that it can be called by a `Timelock` contract.

The contract has a `constructor` function that takes an adapter manager address, a timelock address, and a name for the adapter as input. The contract also has functions for pulling tokens from an address, approving tokens, returning assets to an address, and sweeping assets from an address.

Note that the privileged functions are present in all adapters and will not be repeated in the following adapter sections.

No significant changes were made since the Avalanche audit. Acknowledged issues from the previous audit are not listed again (as goes for all contracts within this audit).

2.2.1 Privileged Functions

- `sweep [ timelock ]`
- `transferOwnership [ owner ]`
- `renounceOwnership [ owner ]`
- `setTimelock [ timelock ]`

2.2.2 Issues & Recommendations

Issue #02	Adapters will not fail if a wrong function is called
Severity	● MEDIUM SEVERITY
Description	`AdapterBase` is inherited by all adapters and defines an empty `fallback` and an empty `receive` function. This means that any adapter can receive Ether directly, and if a function is called that was not defined in the adapter, the transaction would not fail. For example, if someone calls an Aave function using the 1inch Adapter, the function will not revert. This issue becomes annoying when calling multiple adapters at a time because you would not know which call did nothing. Additionally, any adapter can receive Ether directly even if they should not ever receive Ether directly.
Recommendation	Consider removing the `fallback/receive` functions and defining it only within the adapter that needs them.
Resolution	● PARTIALLY RESOLVED The `fallback` function was removed, but the `receive` function was kept.

2.3 Adapters/OneInchAdapter

`OneInchAdapter` inherits from the `AdapterBase` contract and allows for automation to use 1inch to swap for a wallet.

The contract also defines a public constant for the `oneInchRouter` which is hard-coded to be the address `0x1111111254fb6c44bAC0beD2854e76F90643097d`.

Finally, the contract defines a function called `swap` which can be called via delegation in order to perform a swap of tokens using the `OneInchRouter` contract. The function takes two arguments (a `bytes memory callArgs`, and a `uint256 amountETH`), and blindly uses these to call the `OneInchRouter` contract, requiring said call to succeed. Any function can therefore be called on the router.

`OneInchAdapter` is a `delegatecall` adapter.

2.3.1 Issues & Recommendations

No issues found.

2.4 Adapters/WethGateway

`WethGateway` is a simple adapter that allows for the depositing and withdrawal of WETH from and into ETH.

It should be noted that withdrawing WETH straight into a proxy is generally discouraged due to the fallback logic of a proxy costing potentially too much gas for the gas-limited transfer to succeed. However, as the wallet proxy presently has a `receive()` override, this should not cause a problem for now. Generally and informationally speaking, a non-upgradeable helper contract is used to withdraw WETH instead of the approach which is taken here.

`WethGateway` is a `delegationcall` adapter.

2.4.1 Issues & Recommendations

No issues found.

2.5 Adapters/AaveAdapter

`AaveAdapter` allows users to deposit and withdraw tokens from the Aave lending pool. The contract also allows users to borrow and repay tokens. The contract includes a flash loan function that allows users to borrow tokens from the Balancer Vault and repay them using the Aave lending pool. Finally, the contract allows users to claim rewards from the Aave Incentives Controller.

The `deposit` and `positionTransfer` functions are meant to be called by the `AdapterManger`.

All the other functions should be called by a `delegatecall` from the user's `ProxyWallet`.

2.5.1 Privileged Functions

- `initialize [ timelock ]`

2.5.2 Issues & Recommendations

Issue #03	**The `exchangeDebt` function could be delegatecalled directly by any contract**
Severity	● MEDIUM SEVERITY
Description	The `exchangeDebt` function does not check that the call was initiated by the `positionTransfer` function. This could incur losses to an account if that function were delegatecalled directly.
Recommendation	Consider checking that `IAaveAdapter(ADAPTER_ADDRESS).executor() != address(0)` to make sure the call was initiated by the `positionTransfer` function.
Resolution	✓ RESOLVED

Issue #04	**The `positionTransfer` function does not allow the transfer of stable rate borrowings**
Severity	● MEDIUM SEVERITY
Description	The `positionTransfer` function only considers variable rate borrowings. This function will fail if the user borrows with a stable rate market.
Recommendation	Consider also transferring stable rate borrowings by checking the user's balance of the `stableDebtToken` and using `rateMode = 1` to borrow and repay to transfer the user's position if this behavior was not expected.
Resolution	● ACKNOWLEDGED As `rateMode = 1` is not widely adopted, Cian prefers not to add cases that are virtually unused.

Issue #05	**The `positionTransfer` function does not refund enough tokens at the end of the flashloan**
Severity	MEDIUM SEVERITY
Location	L281 `_tokens[i].safeTransfer(address(flashLoanVault), _amounts[i]);`
Description	Currently, Aave has no flashloan fee but it could change in the future. If that happens, the function will revert as the amount transferred back does not consider the fee.
Recommendation	Consider refunding amount + fee at the end of the flashloan if this is not expected. Additionally, the balance check at line 278 should then be moved into the `executeFlashLoan` function surrounding the `flashLoan` call at line 253. One could argue that this function should not be used if there was a flashloan fee. If that is the case, consider clarifying by adding comments.
Resolution	✓ RESOLVED The balance check was moved accordingly. Additionally, if the flashloan had fees, it would not be able to call this function and CIAN's front end will not display this function.

Issue #06	The reentrancy check during flashloan is null and void
Severity	● MEDIUM SEVERITY
Location	L263 `require(executor != address(0), "Reentrant call!");`
Description	Contrary to what is stated, the reentrant check does not prevent any reentrancy. However, this check is essential as it ensures the flashloan was initiated by the `positionTransfer` function. The function also expects that all arrays are of length 1, which could be incorrect during a reentrant call.
Recommendation	Consider adding a reentrancy check and add a safety check that all arrays are of length 1. Additionally, consider removing the `for` loop at lines 280-282 as the lengths are expected and should be of length 1 everywhere else in the function.
Resolution	✓ RESOLVED

Issue #07	Ether can get stuck in the contract during a deposit
Severity	● LOW SEVERITY
Description	The `deposit` function does not check that `msg.value` is 0 when adding a token other than Ether. A deposit with bad parameters could lock Ether in that contract.
Recommendation	Consider checking that `msg.value` is 0 during a deposit of a token other than Ether.
Resolution	✓ RESOLVED

Issue #08	**The `initialize` function could be used to add malicious bad contracts**
Severity	● LOW SEVERITY
Description	The `initialize` function only checks that the underlying asset matches the tokens provided. A contract that was not added to Aave could be added as long as this check passes.
Recommendation	Consider asserting that the token was added to Aave. One way could be to query the `getReserveTokenAddresses` function from the `AaveProtocolDataProvider` contract with each `tokenAddr` and assert that the `aToken` provided and returned are the same and non-zero. This will ensure that the contract has been added to Aave.
Resolution	✓ RESOLVED

Issue #09	**The `positionTransfer` function can only be called using the `multicall` function**
Severity	● INFORMATIONAL
Description	`positionTransfer` needs a callback in order to work properly. As the `executeOnAdapter` function does not allow the user to set if a callback is required, this specific function will revert.
Recommendation	Consider fixing this issue if this behavior is not expected and is causing issues.
Resolution	● ACKNOWLEDGED

Issue #10	Unused imports and variables
Severity	INFORMATIONAL

Description	Imports and variables defined in a contract but not used within said contract could confuse third-party auditors. They also increase the contract length unnecessarily.
	
	L6-8
	`import "../../../interfaces/aave/v2/IProtocolDataProvider.sol";`
	`import "../../../interfaces/aave/v2/IIncentivesController.sol";`
	`import "../../../interfaces/aave/v2/ILendingPool.sol";`
	
	L12
	`import "../../../interfaces/aave/v2/IOracle.sol";`
	
	L73-74
	`address public constant stethTokenAddr = 0xae7ab96520DE3A18E5e111B5EaAb095312D7fE84;`
	
	L76-77
	`address public constant aaveProviderAddr = 0xB53C1a33016B2DC2fF3653530bfF1848a515c8c5;`
	
	L85-86
	`address public constant aaveOracleAddr = 0xA50ba011c48153De246E5192C8f9258A2ba79Ca9;`

Recommendation	Consider removing the unused imports and variables to keep the contract short and simple.
Resolution	PARTIALLY RESOLVED

Issue #11	Typographical errors
Severity	INFORMATIONAL

Location	L213-216
	`function getReward(`
	`    address[] memory assertAddress,`
	`    uint256 amount`
	`) external onlyDelegation`

Description	`getReward` should be renamed to `claimRewards` as it claims rewards and is not a getter. Additionally, `assertAddress` should be renamed as `assetAddress`.
Recommendation	Consider fixing the typographical errors.
Resolution	✓ RESOLVED

Issue #12	`positionTransfer` could run out of gas
Severity	INFORMATIONAL
Description	`positionTransfer` iterates over all the Aave's markets to transfer the user's position to its smart account. This loop could run out of gas if too many markets were to be added.
Recommendation	Consider redesigning the functionality mentioned above if this behavior is not expected.
Resolution	● ACKNOWLEDGED If the function ever runs out of gas, it would be turned off in the interface.

2.6 Adapters/CurvesteCRVAdapter

`CurvesteCRVAdapter` allows CIAN users to interact with the curve protocol. They can exchange `stEth` to `Eth`, add `stEth` and `Eth` liquidity, remove `stEth` and `Eth` liquidity with the options to either receive both tokens back or to accumulate just one of both.

Additionally, the adapter allows CIAN users to `deposit`, `withdraw`, and `claimRewards` from `CurveLiquidityGaugev2`.

The `exchange`, `addLiquidity`, `removeLiquidity` and `removeLiquidityOneCoin` functions are meant to be called by the `AdapterManager`.

The `deposit`, `withdraw` and `claimRewards` functions are meant to be delegatecalled by the user's `ProxyWallet`.

2.6.1 Issues & Recommendations

Issue #13	Unused imports and variables
Severity	● INFORMATIONAL

Description	Imports and variables defined in a contract but not used within said contract could confuse third-party auditors. They also increase the contract length unnecessarily.
	
	L9
	`import "../../interfaces/curve/ICurveLpToken.sol";`
	
	L27-28
	`address public constant crvAddr = 0xD533a949740bb3306d119CC777fa900bA034cd52;`

Recommendation	Consider removing the unused variables and imports.
Resolution	✓ RESOLVED

Issue #14	Lack of events for all functions
Severity	● INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for all functions in the contract.
Resolution	✓ RESOLVED

2.7 Adapters/FeeBoxETH

`FeeBoxETH` is responsible for taking fees from users' wallets to subsidize gas and management costs for the operators that execute automation jobs for them.

All functions are meant to be called by the `AdapterManager`.

2.7.1 Privileged Functions

- `initialize [ timelock ]`
- `setAdapterManager [ timelock ]`
- `paymentCheck [ balanceController ]`
- `setBalance [ balanceController ]`

2.7.2 Issues & Recommendations

Issue #15	Typographical errors
Severity	● INFORMATIONAL
Location	L77 `mapping(address => uint256) public wethBlance;`
Description	`wethBlance` should be renamed to `wethBalance`. Additionally, as the contract stores ETHER and not wrapped ETHER, the variable should be renamed to `ethBalance`.
Recommendation	Consider fixing the typographical errors.
Resolution	✓ RESOLVED

Issue #16	Gas optimizations
Severity	● INFORMATIONAL

Description	We have consolidated the sections which can be further optimized for gas usage below.
	
	L136 - 145
	`require(`
	`    wethBlance[account] + amount + msg.value >=`
	`consumedAmount,`
	`    "Insolvent!"`
	`);`
	
	`wethBlance[account] =`
	`    wethBlance[account] +`
	`    amount +`
	`    msg.value -`
	`    consumedAmount;`
	
	`wethBalance` can be cached to save some gas. It also does not make much sense to deposit if the consumed amount is greater than the `amount + msg.value` as it would only decrease the user’s balance. Consider checking that `amount + msg.value > consumedAmount` instead. This check will also prevent user from depositing 0 ether.
	
	L180 - 181
	`require(wethBlance[account] >= consumedAmount + amount,`
	`"Insolvent!");`
	`wethBlance[account] = wethBlance[account] - amount -`
	`consumedAmount;`
	
	`wethBalance` can be cached to save some gas.

Recommendation	Consider implementing the gas optimizations mentioned above.
Resolution	● ACKNOWLEDGED

2.8 Adapters/FeeBoxStETH

`FeeBoxStETH` is responsible for taking fees from users' wallets to subsidize gas and management costs for the operators that execute automation jobs for them.

All functions are meant to be called by the `AdapterManager`.

2.8.1 Privileged Functions

- `initialize [ timelock ]`
- `setAdapterManager [ timelock ]`
- `paymentCheck [ balanceController ]`
- `setBalance [ balanceController ]`

2.8.2 Issues & Recommendations

Issue #17	Fees are incorrectly transferred back to the user
Severity	● MEDIUM SEVERITY

Location	L110 - 116
	
	`function _paymentCheck(address account, uint256 consumedAmount) internal {`
	`    if (consumedAmount != 0) {`
	`        require(tokenBlance[account] >= consumedAmount, "Insolvent!");`
	`        tokenBlance[account] -= consumedAmount;`
	`        IERC20(stETH).safeTransfer(account, consumedAmount);`
	`    }`
	`}`

Description	`consumedAmount` should be transferred to the `feeReceiver` and not back to the account.
Recommendation	Consider transferring `consumedAmount` to the `feeReceiver`.
Resolution	✓ RESOLVED

Issue #18	Typographical error
Severity	● INFORMATIONAL
Description	L77 `mapping(address => uint256) public tokenBlance;` `tokenBlance` should be renamed as `tokenBalance`.
Recommendation	Consider fixing the typographical error.
Resolution	✓ RESOLVED

Issue #19	Lack of proper message during edge cases
Severity	● INFORMATIONAL
Description	Deposits can fail without a proper message if `consumedAmount` is greater than the user's balance. Withdrawals can fail without a proper message as well if the user's balance is greater than `amount`, but lower than `amount + consumedAmount`.
Recommendation	Consider reverting with a proper message. For deposits, check that the `consumedAmount` is greater than the deposited amount. For withdrawals, check that the user's balance is greater than the consumed and withdrawn amounts.
Resolution	● ACKNOWLEDGED

2.9 Adapters/VerifierBasic

`VerifierBasic` is used by the various FeeBoxes to validate signatures.

2.9.1 Issues & Recommendations

Issue #20	The `verifierBasic` contract is potentially vulnerable
Severity	● MEDIUM SEVERITY
Description	The `ecrecover` EVM opcode allows for malleable (non-unique) signatures. The `ecrecover` EVM opcode can also return a random address for wrong signatures (though this happens with any signature library), and it will return `address(0)` on wrong signatures. Those edge cases are not checked and enforced to ensure the safety of those signatures. This issue specifically poses a threat if the signer is ever set to address zero, as at this point anyone can submit seemingly valid signatures with nonsense data.
Recommendation	Consider removing this contract entirely and use OpenZeppelin's ECDSA directly in the Verifier contract instead. The signatures should follow the EIP-712 standard to allow users to know exactly what they are signing,
Resolution	✓ RESOLVED The contract now uses OpenZeppelin's ECDSA.

2.10 Adapters/LidoAdapter

`LidoAdapter` allows CIAN users to interact with the Lido protocol. Users can call `submit` and `submitWeth` which transfers either `ether` or `wEther` from the user to the Lido protocol and then mints `stEth` to the user. Users can also wrap and unwrap their `stEth` tokens.

`LidoAdapter` is a `delegatecall` adapter.

2.10.1 Privileged Functions

- `initialize [ timelock ]`

2.10.2 Issues & Recommendations

Issue #21	`referral` is only set within the logic contract
Severity	● MEDIUM SEVERITY
Description	Currently, `referral` is set within the `initialize` function. However, since it is used in various functions which are called by a `delegatecall`, the proxy contract will try to find the `referral` variable in its own storage, and it will probably take the variable from its storage place 0. This of course does not work and could cause seriously unintended results as unexpected storage is returned instead.
Recommendation	Consider using `constant` or `immutable` for the `referral` variable.
Resolution	✓ RESOLVED `referral` is now always `address(0)`.

2.11 Adapters/ParaswapAdapter

`ParaswapAdapter` is an adapter that allows the user to use the Paraswap protocol. It simply allows the user to swap tokens. It uses a simple call pattern to call any function on the Paraswap contract just like the 1inch adapter.

`ParaswapAdapter` is a `delegatecall` adapter.

2.11.1 Issues & Recommendations

No issues found.

2.12 Adapters/AdapterManager

`AdapterManager` is the main registry for all Cian adapters. An adapter is a smart contract that Cian operators can use to execute functionality for users on their wallets.

The manager can also be paused by various Cian approved pause guardians. This prevents operators from executing calls on user wallets and can be used as an emergency safeguard if an adapter has a vulnerability.

2.12.1 Privileged Functions

- `execute [ user proxies ]`
- `registerAdapters [ timelock]`
- `unregisterAdapters [ timelock ]`
- `setPauseWhiteList [ timelock ]`
- `setPause [ suspend permissioned accounts & owner can pause, timelock can unpause ]`

2.12.2 Issues & Recommendations

No issues found.

2.13 Core/AccountManager

`AccountManager` is a helper contract that is deployed for each user that aims to increase the comfort when handling an arbitrary amount of `Accounts`. The owner of this contract can add various `Accounts` to the `AccountManager` and grant arbitrary addresses privileged rights to execute the following functions for a `userAccount` (`IAccount`) on the previously added `Accounts`:

- `createSubAccount`
- `executeOnAdapter`
- `executeMulticall`
- `setAdvancedOption`
- `callOnSubAccount`
- `withdrawAssets`
- `approveTokens`

It also allows privileged addresses to call `approve` on the `ERC2612Verifier` as well on the `tokenApprovalVerifier` contract.

As mentioned above, the owner of this contract has the privilege to add and delete accounts via `addAccounts` and `delAccounts`. Before any accounts can be added, the ownership of this account must be transferred to the `AccountManager`.

The most privileged function is the `setAuthorization` function which allows the owner to set any address as executor for specific operations for any account within a certain deadline.

These are the following operations that can be assigned to the executor:

- `CREATE_SUBACCOUNT`
- `EXECUTE_ON_ADAPTER`
- `MULTICALL`
- `SET_ADVANCED_OPTION`
- `CALL_ON_SUBACCOUNT`
- `WITHDRAW_ASSETS`
- `APPROVE_TOKENS`
- `APPROVE_ERC2612_VERIFIER`
- `APPROVE_TOKEN_VERIFIER`

If an address was set as executor for an account with the correct operation, it can execute the function which was assigned to the operation arbitrarily often within the determined deadline.

The owner can also freely define the `ERC2612Verifier` and the `TokenApprovalVerified` as well as change the `minDelay` and `maxDelay` which is used for granting the authorization.

2.13.1 Privileged Functions

- `transferOwnership`
- `renounceOwnership`
- `setDelay`
- `setVerifier`
- `addAccounts`
- `delAccounts`
- `setAuthorization`

2.13.2 Issues & Recommendations

Issue #22	Authorized addresses are difficult to query
Severity	● LOW SEVERITY
Description	An account could forget which address they authorized. They would need to query events to get them, which is complicated, and not all users may be able to do that.
Recommendation	Consider adding the authorized address to a set so the user can query which address was authorized for which accounts more easily.
Resolution	● ACKNOWLEDGED

Issue #23	`accountNum` is a duplicate of the `length` function
Severity	● INFORMATIONAL
Description	`accountNum` will always be equal to the length of account that is returned by the `getAccountsLength` function.
Recommendation	Consider removing the `accountNum` variables to save some gas, as the `getAccountsLength` function already returns this information.
Resolution	✓ RESOLVED

Issue #24	The constructor does not emit the expected events
Severity	● INFORMATIONAL
Description	The constructor sets sensitive variables and should emit the same event as the setters.
Recommendation	Consider emitting events even inside the constructor.
Resolution	✓ RESOLVED

Issue #25	Lack of validation
Severity	● INFORMATIONAL
Description	The contract lacks validation for multiple functions. We have consolidated them below. Within `setVerifier`, there is no validation that `_erc2612Verifier` and `_tokenApprovalVerifier` is not `address(0)`. Within `setAuthorization`, there is no validation for `_authorization`. If this is desired to be able to revoke operations, we recommend adding a default operation for revoking purposes. `setAuthorization` does not check that the approved operations exist. This could be done by asserting that `_authorization < (1 << (max(enum) + 1)`. Additionally, it would be cleaner to assert that the `_authorization` is non-zero and use a dedicated function to revoke authorizations.
Recommendation	Consider validating the above functions properly.
Resolution	✓ RESOLVED

Issue #26	Unused import
Severity	● INFORMATIONAL
Description	Imports defined in a contract but not used within said contract could confuse third-party auditors. They also increase the contract length unnecessarily. L7 `import "@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol";` The contract does already import Ownable, so there is no need to also import the upgradeable version.
Recommendation	Consider removing the unused import.
Resolution	✓ RESOLVED

Issue #27	Typographical errors
Severity	INFORMATIONAL

Description	We have consolidated the typographical errors into a single issue to keep the report brief and readable.
	
	L6
	`uint256 public constant MINIMUM_DELAY = 1 days;`
	
	This should be named `MINIMUM_DEADLINE`.
	
	L7
	`uint256 public constant MAXIMUM_DELAY = 365 days;`
	
	This should be named `MAXIMUM_DEADLINE`.
	
	L119
	`OwnableUpgradeable(_accounts[i]).owner() == address(this),`
	
	Since we recommended removing the `OwnableUpgradeable` import, this should be casted to either `Ownable` or `IAccount`.

Recommendation	Consider fixing the above errors.
Resolution	✓ RESOLVED

2.14 Core/Automation

`Automation` contract is the core authorization contract used by all wallets. Operators must go through the `Automation` contract if they wish to execute automation tasks on a user wallet. `Automation` will then call the `ERC2612Verifier` to check if the operator has permission to execute the specific action for the user.

`TokenApprovalVerifier` will be queried if the action deals with tokens.

The user can also set a `LoanProvider` that will be used for flashloans, and if none are defined, the default one will be used.

2.14.1 Privileged Functions

- `setLoanProvider [ only account owner ]`
- `autoExecute [ only approved adapters ]`
- `autoExecuteMultiCall [ only approved adapters ]`
- `autoApprove [ only if 0 was approved and spender needs to have been approved ]`
- `autoApproveWithPermit [ only if 0 was approved and owner has signed a message to permit ]`
- `doFlashLoan [ only if 1 was approved ]`
- `autoExecuteOnSubAccount [ only if 2 was approved ]`
- `doFlashLoanOnSubAccount [ only if 3 was approved ]`

2.14.2 Issues & Recommendations

No issues found.

2.15 Core/AutomationCallable

`AutomationCallable` is a contract that needs to be inherited to allow the contract to set an `autoExecutor` which allows it to execute tasks on the contract.

2.15.1 Issues & Recommendations

No issues found.

2.16 Core/ControllerLib

`ControllerLib` represents the core contract of CIAN architecture — it is the implementation of the user's `ProxyWallet`, which is their virtual wallet. `ControllerLib`, therefore, contains all core logic for the user and other system components to manage the user's virtual wallet.

It allows the user to force their virtual wallet to execute arbitrary logic through either calls or delegatecalls. It also allows the user to approve various controllers to execute logic on adapters for them. These controllers do this by calling the `CallProxy` (called the "automation" in this contract) which is also described within this audit. The `CallProxy` then validates the request and forwards it to the user's virtual wallet.

2.16.1 Privileged Functions

- `createSubAccount [ owner ]`
- `executeOnAdapter [ automation / owner ]`
- `multiCall [ automation / owner ]`
- `callDirectly [ owner ]`
- `callOnSubAccount [ automation / owner ]`
- `setAdvancedOption [ owner ]`
- `withdrawAssets [ owner ]`
- `approve [ automation / owner ]`
- `approveTokens [ automation / owner ]`
- `transferOwnership [ owner ]`
- `renounceOwnership [ owner ]`
- `reinitialize [ owner ]`

2.16.2 Issues & Recommendations

Issue #28	The `multicall` function might fail or send too much Ether
Severity	● MEDIUM SEVERITY

Location	L167 - 169
	`returnData = IAdapterManager(adapterManager).execute{`
	`    value: costETH + msg.value`
	`}(_callBytes);`

Description	During a call to the `multicall` function, `msg.value` will be the forwarder each time, meaning that it will try to send `msg.value` over and over. This will either send too much Ether, or revert because of the insufficient balance.
Recommendation	Consider redesigning those functions to avoid this issue. One way could be to send a number of Ether that was given as a parameter while verifying that `msg.value` and the parameter are set accordingly.
Resolution	✓ RESOLVED `multicall` is no longer `payable`. It should be noted that this fix comes at the cost of not being able to send Ether to any function or adapter during a `multicall`.

Issue #29	Privilege escalation: The approve functions allow the bypassing of the `advancedTradingEnable` boolean
Severity	● MEDIUM SEVERITY
Location	L299 - 339

```
function withdrawAssets(
    address[] memory _tokens,
    address _receiver,
    uint256[] memory _amounts
) external onlyOwner {
    if (_receiver != owner() && !isSubAccount[_receiver]) {
        require(advancedOptionEnable, "Not allowed!");
    }
    _transferAssets(_tokens, _amounts, _receiver);

    emit WithdrawAssets(_tokens, _receiver, _amounts);
}

function approve(
    IERC20 _token,
    address _spender,
    uint256 _amount
) external onlyAutomationOrOwner {
    _token.safeApprove(_spender, 0);
    _token.safeApprove(_spender, _amount);

    emit ApproveToken(_token, _spender, _amount);
}
```

```
function approveTokens(
    IERC20[] memory _tokens,
    address[] memory _spenders,
    uint256[] memory _amounts
) external onlyAutomationOrOwner {
    require(
        _tokens.length == _amounts.length &&
            _spenders.length == _amounts.length,
        "approve length error."
    );
    for (uint256 i = 0; i < _tokens.length; i++) {
        _tokens[i].safeApprove(_spenders[i], 0);
        _tokens[i].safeApprove(_spenders[i], _amounts[i]);
    }

    emit ApproveTokens(_tokens, _spenders, _amounts);
}
```

Description	In order to withdraw tokens to an external address, the owner needs to allow the `advancedOptionEnable`. A privilege escalation can occur by approving an external address as the spender. This spender can then call `transferFrom` to withdraw the tokens. This can be done by the owner or the automation.
Recommendation	Consider whether this is an issue. If so, consider preventing these functions from being called when that bool is set to false
Resolution	● ACKNOWLEDGED The Cian team does not consider this to be an issue since the advanced mode is to let users execute arbitrary operation rather than withdrawing funds.

Issue #30	Typographical error
Severity	● INFORMATIONAL

Location	L124 - 128
	`require(`
	`    // autoExecutor or owner`
	`    autoExecutor == msg.sender || owner() == msg.sender,`
	`    "Permit: caller is not the Permit"`
	`);`

Description	The error message seems outdated.
Recommendation	Consider fixing the typographical error.
Resolution	✓ RESOLVED

2.17 Core/ControllerLibSub

`ControllerLibSub` represents a sub-wallet of the main `ControllerLib` wallet with less strict permission controls. The main wallet has full authorization over this sub wallet as well as the main wallet owner.

Most of the issues from `ControllerLib` are present here as well.

2.17.1 Privileged Functions

- `reinitialize [ eoa owner ]`
- `withdrawAssets [ eoa owner ]`
- `approveTokens [ eoa owner ]`
- `executeOnAdapter [ owner: parent wallet ]`
- `multiCall [ owner: parent wallet ]`

2.17.2 Issues & Recommendations

Issue #31	**`implementationAddress` must be made immutable for the value to be set correctly on the proxy**
Severity	● INFORMATIONAL
Description	`implementationAddress` is the zero address on the proxy. The value should be made `immutable`.
Recommendation	Consider making the variable `immutable`.
Resolution	✓ RESOLVED

2.18 Core/ControllerLink

`ControllerLink` is a helper contract that will behave like a user database. Every time a new `ProxyWallet` is created, it is added to the `ControllerLink` mappings.

2.18.1 Privileged Functions

- `addAuth [ factory ]`
- `removeAuth [ owner ]`
- `transferOwnership [ owner ]`
- `renounceOwnership [ owner ]`

2.18.1 Privileged Functions

No issues found.

2.19 Core/BalancerERC3156

`BalancerERC3156` is a simple user interface for executing flashloans. The user can request a flashloan from the `vault` with an arbitrary `borrower` address as receiver.

The `vault` will then send the tokens to the contract and these tokens will then be sent to the `borrower` to execute its logic with the tokens.

After the logic is executed, `BalancerERC3156` will take the tokens + fee from the `borrower` and send it back to the `vault`.

2.19.1 Issues & Recommendations

Issue #32	`maxFlashLoan` returns a wrong value
Severity	● LOW SEVERITY
Description	Currently, the `maxFlashLoan` returns `uint256(max);` however, the function name indicates that the goal for this function is to return the maximum flashloan amount.
Recommendation	Consider removing this function or adding logic that returns the maximum possible amount of a flashloan for a specific token, i.e. the token's balance of the vault.
Resolution	● ACKNOWLEDGED

Issue #33	The reentrancy check is flawed
Severity	● LOW SEVERITY
Location	L70 `require(executor != address(0), "reEntrance");`
Description	This requirement is not a reentrancy check, but it ensures that the flashloan was initiated by this contract with the `flashLoan` function. A reentrancy could in theory still be made, though we are sure the balancer implementation protects against this.
Recommendation	Consider reverting with a more accurate message. Also, if a reentrancy check was needed, there can be a check that `executor` is `address(0)` at line 50.
Resolution	● ACKNOWLEDGED

Issue #34	`vault` can be made `constant`
Severity	INFORMATIONAL
Description	Variables that are never modified can be indicated as such with the constant keyword. This is considered best practice since it makes the code more accessible for third-party reviewers and saves gas.
Recommendation	Consider making the variable explicitly `constant`.
Resolution	ACKNOWLEDGED

2.20 Core/ERC2612Verifier

`ERC2612Verifier` allows users to specify if they approve basic operations and/or specific adapters. Those approvals are represented using ids. If a user wants to allow a specific id, they need to call approve with `2^id` as the `approvalType`.

In addition, an user can sign a message to approve an adapter without ever calling the function themselves.

Currently the basic operations are:

- (2^0): approve a token.
- (2^1): allow flashloans on `BankerJoe`.

The `id` of the different adapters will be chosen by the team.

Note that any approval will overwrite all previous approvals. This means that the user must be extremely careful with their transaction bytes, as it will be exceptionally difficult to figure out which adapter they are approving.

2.20.1 Privileged Functions

- `approve [only owner of that account]`
- `revoke [only owner of that account]`

2.20.1 Issues & Recommendations

No issues found.

2.21 Core/TokenApprovalVerifier

`TokenApprovalVerifier` allows users to approve different addresses to use the tokens that are in their proxies. They can also sign a message that can be used to approve on behalf of the user.

2.21.1 Privileged Functions

- `approve [ proxies owner ]`

2.21.2 Issues & Recommendations

No issues found.

2.22 Timelock

`Timelock` is a clean fork of Compound Finance's timelock. This is the most common contract used in DeFi to time lock governance access and is thus compatible with most third-party tools.

`Timelock` allows an administrator to set a delay before transactions are executed, which must be between 12 hours and 30 days. This prevents the administrator from executing transactions without first announcing them beforehand. Transactions can be queued by the administrator, and they will be executed after the delay has passed. If a transaction is not executed within the grace period, it is considered stale and will not be executed. This ensures that only transactions which have been properly announced and queued will be executed, preventing the administrator from executing unauthorized or malicious transactions.

The admin is the account which has been designated as the owner of the Timelock contract.

Parameter	Value	Description
Delay	12 hours	The `delay` indicates the time the administrator has to wait after queuing a transaction to execute it.
Minimum Delay	12 hours	The `minDelay` indicates the lowest value that the `delay` can minimally be set. Sometimes, projects will queue a transaction that sets the `delay` to zero with the hope that nobody notices it. However, because of the minimum delay parameter, the value of `delay` can never be lower than that of the `minDelay` value. Note that the administrator could still queue a transaction to simply transfer the ownership back to their own account so it is still important to inspect every transaction carefully.
Grace Period	14 days	After the delay has expired after queueing a transaction, the administrator can only execute it within the grace period. This is to prevent them from hiding a malicious transaction among much earlier transactions, hoping that it goes unnoticed or buried, which can be executed in the future.

2.22.1 Privileged Functions

- `setDelay [ timelock itself ]`
- `setPendingAdmin [ timelock itself ]`
- `acceptAdmin [ new owner ]`
- `queueTransaction [ owner ]`
- `cancelTransaction [ owner ]`
- `executeTransaction [ owner ]`

2.22.2 Issues & Recommendations

No issues found.

2.23 TimelockCallable

`TimelockCallable` is an abstract contract that is meant to be inherited by various contracts. It contains logic that allows certain functions to get only executed by the `Timelock`.

The timelock can be changed by the timelock by calling the setTimelock function.

2.23.1 Privileged Functions

- `setTimelock (onlyTimelock)`

2.23.2 Issues & Recommendations

No issues found.

2.24 AddressArrayLib

`AddressArrayLib` is a utility library that can be used to add, remove, or fill values in an address array. The library includes functions to add an item to an array, add an item to an array only if it is not already in the array, verify if an array contains a particular value, reassign all items in an array with a specified value, verify if array is a set of unique values, and remove items from an array.

The gas cost of the `contains` function alongside many of the other functions increases linearly and sometimes quadratically with the size of the array, since they have to loop through the entire array to check if the target value is present.

2.24.1 Issues & Recommendations

Issue #35	`AddressArrayLib` is unused
Severity	● INFORMATIONAL
Description	The contract is unused even though it is imported.
Recommendation	Consider removing it to make the code base more readable.
Resolution	● ACKNOWLEDGED

PALADIN

BLOCKCHAIN SECURITY