



**PALADIN**  
BLOCKCHAIN SECURITY

# Smart Contract Security Assessment

Final Report

For Cian

25 October 2022



[paladinsec.co](https://paladinsec.co)



[info@paladinsec.co](mailto:info@paladinsec.co)

# Table of Contents

Table of Contents	2
Disclaimer	5
1 Overview	6
1.1 Summary	6
1.2 Contracts Assessed	7
1.3 Findings Summary	8
1.3.1 Global Issues	9
1.3.2 ProxyWallet	9
1.3.3 ControllerLib	10
1.3.4 WalletFactory	11
1.3.5 CallProxy/CallProxyLib	11
1.3.6 ERC2612Verifier	12
1.3.7 ControllerLink	12
1.3.8 Record	12
1.3.9 ProxyCallable	13
1.3.10 AdapterManager	13
1.3.11 AdapterBase	14
1.3.12 WavaxGateway	14
1.3.13 SAVAXAdapter	14
1.3.14 TraderJoeAdapter	15
1.3.15 FeeBoxAVAX, FeeBoxSAVAX and FeeBoxToken	15
1.3.16 VerifierBasic	16
1.3.17 BankerJoeAdapter / BenqiAdapter	16
1.3.18 JoeERC3156	17
1.3.19 Timelock	17
2 Findings	18
2.1 Global Issues	18

2.1.3 Issues & Recommendations	19
2.2 ProxyWallet	22
2.2.1 Privileged Functions	22
2.2.2 Issues & Recommendations	23
2.3 ControllerLib	25
2.3.1 Privileged Functions	26
2.3.2 Issues & Recommendations	27
2.4 WalletFactory	38
2.4.1 Privileged Functions	38
2.4.2 Issues & Recommendations	39
2.5 CallProxy/CallProxyLib	43
2.5.1 Privileged Functions	43
2.5.2 Issues & Recommendations	44
2.6 ERC2612Verifier	51
2.6.1 Privileged Functions	51
2.6.2 Issues & Recommendations	52
2.7 ControllerLink	54
2.7.1 Privileged Functions	54
2.7.2 Issues & Recommendations	55
2.8 Record	59
2.8.1 Privileged Functions	59
2.8.2 Issues & Recommendations	60
2.9 ProxyCallable	62
2.9.1 Issues & Recommendations	63
2.10 AdapterManager	64
2.10.1 Privileged Functions	64
2.10.2 Issues & Recommendations	65
2.11 AdapterBase	71
2.11.1 Privileged Functions	71
2.11.2 Issues & Recommendations	72

2.12 WavaxGateway	75
2.12.1 Issues & Recommendations	76
2.13 SAVAXAdapter	77
2.13.1 Issues & Recommendations	78
2.14 TraderJoeAdapter	79
2.14.1 Privileged Functions	79
2.14.2 Issues & Recommendations	80
2.15 FeeBoxAVAX, FeeBoxSAVAX and FeeBoxToken	86
2.15.1 Privileged Functions	86
2.15.2 Issues & Recommendations	87
2.16 VerifierBasic	91
2.16.1 Issues & Recommendations	92
2.17 BankerJoeAdapter / BenqiAdapter	94
2.17.1 Privileged Functions	94
2.17.2 Issues & Recommendations	95
2.18 JoeERC3156	100
2.18.1 Issues & Recommendations	101
2.19 Timelock	105
2.19.1 Issues & Recommendations	105



# 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 advocacy 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 on the Avalaunch 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.

## 1.1 Summary

<b>Project Name</b>	Cian
<b>URL</b>	<a href="https://cian.app/">https://cian.app/</a>
<b>Network</b>	Avalanche
<b>Language</b>	Solidity

## 1.2 Contracts Assessed

Name	Contract	Live Code Match
ProxyWallet	Deployed by WalletFactory	✓ MATCH
ControllerLib	0x601954e6AfB77Dac21503DbDfA751fbef9eE5374	✓ MATCH
WalletFactory	0x15cbFF12d53e7BdE3f1618844CaaEf99b2836d2A	✓ MATCH
CallProxy	Removed; not deployed	UNUSED
CallProxyLib (Renamed to Automation)	0x056c41b8C2A2E7C6454842C9A62050fa1b5ffbAE	✓ MATCH
ERC2612Verifier	0x25440d9E199974e705a07DF6F2464291D0ba1e2f	✓ MATCH
ControllerLink	0x4792e147bCE02E5FF2b1B70416811704B5625446	✓ MATCH
Record	Removed; not deployed	UNUSED
ProxyCallable (Renamed to AutomationCallable)	Dependency	✓ MATCH
AdapterManager	0xf8fE4E5Db46D91cC30eae491363dC456e1DaF2fD	✓ MATCH
AdapterBase	Dependency	✓ MATCH
WavaxGateway	0x28F83cE214462E888787C5cfD0cc08dD439C9920	✓ MATCH
SAVAXAdapter	0x83B15AB252482E8AfB0E47460B46AaE5F145ec17	✓ MATCH
TraderJoeAdapter	0xDA7fBbDFf6225e37D349676f7b65684E96dd5C16	✓ MATCH
FeeBoxAVAX	0xec55E7cfebBE4f878E9dD998d3a038458AC3197D	✓ MATCH
FeeBoxSAVAX	0xb7ead62ca64A98b21C1212BCC82436D7E7d797c3	✓ MATCH
FeeBoxToken	Removed; not deployed	UNUSED
VeriferBasic	Dependency of FeeBoxes	✓ MATCH
BankerJoeAdapter	0x123d4F3126B0F57B86d15382ec72A444Bb6E77de	✓ MATCH
BenqiAdapter	0xe7a5b5783bee4C91c2Bdfb00FF5a34426b6b8a02	✓ MATCH
JoeERC3156	Removed; not deployed	UNUSED
Timelock	0xD3812219eb241053F9cf2b43f9B367c0b28E03DA	✓ MATCH

## 1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
● High	7	6	1	-
● Medium	14	13	1	-
● Low	23	16	-	7
● Informational	49	39	5	5
Total	93	74	7	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	LOW	Phishing: Users might have difficulties to distinguish malicious transactions if the frontend is ever compromised	ACKNOWLEDGED
02	INFO	Gas optimizations	PARTIAL
03	INFO	Typographical errors	PARTIAL

## 1.3.2 ProxyWallet

ID	Severity	Summary	Status
04	LOW	Proxy receive() function prevents ControllerLib receive() from being called	✓ RESOLVED
05	INFO	proxyAdmin can become outdated	✓ RESOLVED



### 1.3.3 ControllerLib

ID	Severity	Summary	Status
06	HIGH	SELFDESTRUCT can potentially be executed on an uninitialized implementation	✓ RESOLVED
07	HIGH	_multiCall does not validate that the _certifiedAddress is unset after the individual iterations are fulfilled allowing a malicious operator to drain and even destroy user proxies	✓ RESOLVED
08	HIGH	Privilege escalation risk: onFlashLoan multicall callback is a full privilege escalation and allows governance to potentially drain all wallets	PARTIAL
09	MEDIUM	Validation on withdrawAssetsToAccount is almost completely useless	✓ RESOLVED
10	LOW	Funds could become permanently lost if a CertifiedAddress or adapter ever contains the SELFDESTRUCT opcode as it would delete the proxy	ACKNOWLEDGED
11	LOW	ControllerLib contains unnecessary logic which makes it less generic than it could be	ACKNOWLEDGED
12	LOW	adapManager, advancedOptionEnable and CertifiedAddress are private	✓ RESOLVED
13	LOW	callBytes of _callOnAdapter unnecessarily contains costETH	✓ RESOLVED
14	LOW	onFlashLoan does not validate _multiCall parameter lengths	✓ RESOLVED
15	INFO	Unused import: Record and ProxyAdmin	✓ RESOLVED
16	INFO	Unused event: ChangeAutomation	✓ RESOLVED
17	INFO	Lack of events for setCertified, setAdapManager, setAdvancedOption and the various common functions	✓ RESOLVED
18	INFO	Typographical errors	✓ RESOLVED

## 1.3.4 WalletFactory

ID	Severity	Summary	Status
19	MEDIUM	Phishing risk: Malicious admins can be used	✓ RESOLVED
20	LOW	userDatabase and timeLock are private	✓ RESOLVED
21	INFO	userProxyAdmin may be outdated and may have a shorter length to userAccount	✓ RESOLVED
22	INFO	Unused Ownable inheritance	✓ RESOLVED
23	INFO	Lack of events for setCodeHash and createAccount	✓ RESOLVED
24	INFO	Typographical error	✓ RESOLVED
25	INFO	UI functions getUserProxyAdmin and getUserAccount can run out of gas	✓ RESOLVED

## 1.3.5 CallProxy/CallProxyLib

ID	Severity	Summary	Status
26	HIGH	Governance Issue: The implementation is a proxy	✓ RESOLVED
27	LOW	The initialize function does not call the initialize function safely	✓ RESOLVED
28	LOW	Adapter load-in allows for loading in a 32 bytes value while an address is just 20 bytes	ACKNOWLEDGED
29	INFO	Unused inheritance: OwnableUpgradeable.sol	✓ RESOLVED
30	INFO	Unused event: CallForwardSignle	✓ RESOLVED
31	INFO	Lack of events for setPublicVerifier, setAccountVerifier and setFlashLoanWhiteList	✓ RESOLVED
32	INFO	Typographical errors	✓ RESOLVED
33	INFO	multicalls lack nonzero checks	ACKNOWLEDGED
34	INFO	permit can be frontrun and cause denial of service	✓ RESOLVED

## 1.3.6 ERC2612Verifier

ID	Severity	Summary	Status
35	LOW	The current implementation limit the number of adapter to 240	ACKNOWLEDGED
36	INFO	The OperatorUpdate event lacks the approval type variable	✓ RESOLVED
37	INFO	DOMAIN_SEPARATOR can be made immutable	✓ RESOLVED
38	INFO	approvals, approve and revoke can be made external	✓ RESOLVED

## 1.3.7 ControllerLink

ID	Severity	Summary	Status
39	MEDIUM	The removeAuth function can be called multiple time with the same values	✓ RESOLVED
40	LOW	trustFactory and timeLock are private	✓ RESOLVED
41	INFO	Gas optimizations	PARTIAL
42	INFO	Typographical and minor errors	ACKNOWLEDGED

## 1.3.8 Record

ID	Severity	Summary	Status
43	MEDIUM	Any authorized address can become the only authorized address	✓ RESOLVED
44	INFO	Contract lacks an easy way for users to figure out the list of authorized addresses through explorer contract inspection	✓ RESOLVED
45	INFO	Lack of events for initAuth, enable and disable	✓ RESOLVED

## 1.3.9 ProxyCallable

ID	Severity	Summary	Status
46	INFO	Typographical error	ACKNOWLEDGED

## 1.3.10 AdapterManager

ID	Severity	Summary	Status
47	MEDIUM	delegatecalls are still possible even if the AdapterManager is paused	PARTIAL
48	LOW	unregisterAdapters does not reset adaptersIndex	✓ RESOLVED
49	LOW	The registerAdapters function allows the addition of too many adapters	✓ RESOLVED
50	LOW	_paused and suspendPermissions are private	✓ RESOLVED
51	INFO	Non-transferrable timelock address might be limiting if the client ever wants to move to a new governance structure	✓ RESOLVED
52	INFO	Unused imports, functionality and typographical errors	✓ RESOLVED
53	INFO	Gas usage: getRegisteredAdapters might run out of gas	✓ RESOLVED
54	INFO	Lack of events for setPauseWhiteList	✓ RESOLVED
55	INFO	setPauseWhiteList check can be simplified	✓ RESOLVED

## 1.3.11 AdapterBase

ID	Severity	Summary	Status
56	INFO	Governance privilege: The timelock can potentially sweep wallets through reentrancy during delegatecalls	✓ RESOLVED
57	INFO	ADAPTER_NAME is unaccessible during delegate calls	ACKNOWLEDGED
58	INFO	Unused import: IWAVAX.sol	✓ RESOLVED
59	INFO	Typographical errors	PARTIAL
60	INFO	pullTokensIfNeeded might not pull in enough tokens for tokens with a fee on transfer	✓ RESOLVED

## 1.3.12 WavaxGateway

ID	Severity	Summary	Status
61	INFO	Typographical error	✓ RESOLVED

## 1.3.13 SAVAXAdapter

ID	Severity	Summary	Status
62	LOW	SAVAXAdapter lacks various functions	ACKNOWLEDGED

### 1.3.14 TraderJoeAdapter

ID	Severity	Summary	Status
63	MEDIUM	User can get sandwiched and suffer high slippage	✓ RESOLVED
64	LOW	The adapter does not allow users to call emergencyWithdraw	✓ RESOLVED
65	LOW	JoeBar is now deprecated	✓ RESOLVED
66	LOW	addLiquidityAVAX contains a seemingly redundant transfer	✓ RESOLVED
67	LOW	depositLpToken and withdrawLpToken can deposit into and withdraw from Masterchefs other than Trader Joe	✓ RESOLVED
68	INFO	The JTokenSnapshot structure and IJToken import are unused	✓ RESOLVED
69	INFO	router can be made constant	✓ RESOLVED
70	INFO	SafeMath is unnecessary starting from Solidity version 0.8	✓ RESOLVED
71	INFO	Gas optimizations	✓ RESOLVED

### 1.3.15 FeeBoxAVAX, FeeBoxSAVAX and FeeBoxToken

ID	Severity	Summary	Status
72	HIGH	SAVAX redemptions might eventually relock or run out of gas	✓ RESOLVED
73	MEDIUM	Governance privilege: balanceController can take funds from users' accounts	✓ RESOLVED
74	MEDIUM	Approval flow does not contain necessary safeguards	✓ RESOLVED
75	LOW	balanceController and feeReceiver are private	✓ RESOLVED
76	INFO	Lack of events for initialize	✓ RESOLVED
77	INFO	Typographical and minor errors	✓ RESOLVED

## 1.3.16 VerifierBasic

ID	Severity	Summary	Status
78	HIGH	recoverSigner allows anyone to fake signatures for the zero address	✓ RESOLVED
79	INFO	Unused imports	✓ RESOLVED
80	INFO	Typographical error	ACKNOWLEDGED

## 1.3.17 BankerJoeAdapter / BenqiAdapter

ID	Severity	Summary	Status
81	MEDIUM	Governance privilege: The timelock could be used to steal users' tokens	✓ RESOLVED
82	MEDIUM	repay may use an outdated value	✓ RESOLVED
83	MEDIUM	Operations do not revert if underlying protocols are paused	✓ RESOLVED
84	LOW	Governance could delist a market making redemption of tokens complicated	ACKNOWLEDGED
85	LOW	GetUserDepositPosition returns wrong values	✓ RESOLVED
86	INFO	BankerJoeAdapter: Unused event — TraderJoeStake and TraderJoeUnstake	✓ RESOLVED
87	INFO	BankerJoeAdapter: Unused variables — joeBarAddr	✓ RESOLVED
88	INFO	Typographical errors	PARTIAL



## 1.3.18 JoeERC3156

ID	Severity	Summary	Status
89	HIGH	Lack of authentication on flashLoan call	✓ RESOLVED
90	MEDIUM	The initiator address of the onFlashLoan is not reliable	✓ RESOLVED
91	MEDIUM	onFlashLoan callback is vulnerable to reentrancy	✓ RESOLVED
92	MEDIUM	Governance privilege: The timelock could be used to steal users' tokens	✓ RESOLVED
93	INFO	Typographical errors	✓ RESOLVED

## 1.3.19 Timelock

No issues found.



## 2 Findings



---

### 2.1 Global Issues

The issues in this section apply to the protocol as a whole. We have consolidated the global issues to simplify the report.



## 2.1.3 Issues & Recommendations

Issue #01	<b>Phishing: Users might have difficulties to distinguish malicious transactions if the frontend is ever compromised</b>
Severity	 LOW SEVERITY
Description	Many transactions within the Cian ecosystem make it extremely difficult for the user to figure out what they are executing on their wallet. This introduces the risk that if a frontend is ever hacked, the user might not know they are actually allowing the hacker to drain their wallet.
Recommendation	Consider very carefully safeguarding the frontend. Ideally the system should be designed with transaction inspection in mind but we understand this is difficult to accomplish.
Resolution	 ACKNOWLEDGED The client has indicated they will consider front-end security with the utmost level of care. However, users should still be careful of the transactions they make as there is no way to guarantee the safety of a frontend.

**Issue #02****Gas optimizations****Severity** INFORMATIONAL**Description**

The contract contains multiple sections of code that could be further optimized for gas efficiency. We've enumerated these in a single issue in an effort to keep the report brief and readable.

Throughout the codebase, the memory type is used for external functions and their variable type arguments. This uses unnecessary gas as the parameter is needlessly copied into memory. If the parameter is never changed, the client can keep the type as `calldata` to save gas. On a low level, this causes the contract to directly fetch the values from the `calldata` storage instead of copying them over to memory first. As `calldata` storage is immutable, the variables cannot be changed if they are marked as `calldata`, which is why Solidity allows you to specify parameters as memory.

**Recommendation**

Consider implementing the gas optimizations mentioned above.

**Resolution** PARTIALLY RESOLVED

This is resolved in certain locations.



**Issue #03****Typographical errors****Severity** INFORMATIONAL**Description**

The contract contains a number of typographical errors which we have consolidated below in a single issue in an effort to keep the report size reasonable.

AdapterBase::94 (example)

```
function sweep(address[] memory tokens, address receiver)
```

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... 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 furthermore indicates to third parties that the developer has a good understanding of solidity best practice.

ControllerLib::67 (example)

```
address private CertifiedAddress;
```

proxyWallet::11 (example)

```
contract proxyWallet is TransparentUpgradeableProxy {
```

Next, the contract often deviates from the casing Solidity standard practice. Contracts should always be capitalized while all variables except for constants should start with a lowercase character.

```
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** PARTIALLY RESOLVED

Some of these errors have been resolved throughout the codebase.

---

## 2.2 ProxyWallet

ProxyWallet is the proxy contract for a user's wallet. It will hold the user's token and important information related to the user's activity. Each user has their own proxyWallet. The implementation of the proxyWallet is the ControllerLib which is also covered within this audit.



It should be noted that the ProxyWallet is the absolute centerpiece contract for users. It represents their virtual wallet that owns all assets for the user. The user should therefore be very careful with calling the three privileged functions: `changeAdmin`, `upgradeTo` and `upgradeToAndCall`. As this contract is a proxy that points to the ControllerLib, users should understand that all logic and issues described in the ControllerLib section also effectively apply to this contract, as the ControllerLib code is used to handle any call to the ProxyWallet.



### 2.2.1 Privileged Functions

The following functions can be called by the owner of the contract:

- `changeAdmin`
- `upgradeTo`
- `upgradeToAndCall`

## 2.2.2 Issues & Recommendations

<b>Issue #04</b>	<b>Proxy receive() function prevents ControllerLib receive() from being called</b>
<b>Severity</b>	 LOW SEVERITY
<b>Description</b>	<p>The proxy implementation, the ControllerLib, has a receive() function to receive gas tokens. However, as the proxy also has a receive() function, this causes the ControllerLib receive() to never be called.</p>
<b>Recommendation</b>	<p>We understand the desire to have a proxy level receive() function as implementation level receive() functions often use too much gas to allow for .transfer() (used for example in WAVAX.withdraw()) calls to work.</p> <p>However, this does mean that the contract cannot execute fallback logic when it receives AVAX (which is not that big of a deal) but more importantly that the ControllerLib receive() function is redundant. The client should therefore consider removing that function from ControllerLib.</p>
<b>Resolution</b>	 RESOLVED
	The underlying ControllerLib receive() has been removed.

Issue #05	proxyAdmin can become outdated
Severity	 INFORMATIONAL
Location	<u>Line 22</u> <code>address private proxyAdmin;</code>
Description	<p>As a proxy traditionally only exposes the <code>admin()</code> function to the actual admin, the client has added a <code>getProxyAdmin()</code> function to allow for the admin to be visible within the explorer.</p> <p>However, if the admin is ever transferred, the <code>proxyAdmin</code> variable would not be updated as it is just a local variable not linked to the actual proxy admin logic.</p>
Recommendation	<p>Consider removing the <code>proxyAdmin</code> logic and changing it to a dynamic <code>getProxyAdmin</code> implementation:</p> <pre>function getProxyAdmin() external view returns (address) {     return _getAdmin(); }</pre>
Resolution	 RESOLVED <p>The recommendation has been implemented.</p>





---

## 2.3 ControllerLib

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

It allows for 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. The controllers do this by calling CallProxy, (called the "automation" in this contract) which is also covered within this audit. CallProxy then validates the request and forwards it to the user's virtual wallet.

**!** Disclaimer: During the course of this audit, CertifiedAddress logic was replaced with callback logic which allows an adapter to escalate to delegatecalling code within the wallet's context. subAccount logic was also added (with several flaws) during the course of the audit. This logic and its flaws are not within the scope of this audit, all though Paladin did try to guide the Cian team through several critical flaws found in this logic, regardless of it being out of scope.





## 2.3.1 Privileged Functions

The following functions can be called by the various privileged roles of the contract:

- `setAdapManager [ owner ]`
- `setAdvancedOption [ owner ]`
- `withdrawAssets [ owner ]`
- `approve [ owner / CallProxy ]`
- `approveTokens [ owner / CallProxy ]`
- `executeOnAdapter [ owner / CallProxy ]`
- `multiCall [ owner / CallProxy ]`
- `callDirectly [ owner ]`
- `callback [ certified: set by owner / CallProxy ]`
- `transferOwnership [ owner ]`
- `renounceOwnership [ owner ]`
- `InitAuth [ auth ]`
- `enable [ auth ]`
- `disable [ auth ]`



## 2.3.2 Issues & Recommendations

Issue #06	SELFDESTRUCT can potentially be executed on an uninitialized implementation
Severity	 HIGH SEVERITY
Description	<p>Solidity has a special opcode to delete a contract from the blockchain. It is commonly used to clean-up temporary contracts as it gives a gas rebate. However, since the ControllerLib often delegatecalls to other contracts, a malicious party could initialize the implementation that is shared between all proxies and cause it to self destruct.</p>
Recommendation	<p>Consider initializing the implementation and burning ownership. Next, consider adding an onlyProxy modifier to all functions which execute delegatecalls:</p> <pre>address public immutable implementationAddress;  constructor () {     implementationAddress = address(this); }  modifier onlyProxy() {     require(address(this) != implementationAddress, "! proxy");     _; }</pre> <p>This modifier uses a little trick as it executes code in the constructor of the implementation, which is ignored at the proxy level. However, since immutable variables are in fact directly stored in the on-chain bytecode, the implementationAddress actually becomes available at the proxy level and still references the implementation address. By ensuring the current context (address(this)) does not equal the implementation address, we effectively lock usage of functions with this modifier to proxy calls exclusively.</p>
Resolution	 RESOLVED

**Issue #07**

**`_multiCall` does not validate that the `_certifiedAddress` is unset after the individual iterations are fulfilled allowing a malicious operator to drain and even destroy user proxies**

**Severity** HIGH SEVERITY**Description**

The contract allows the controllers to provide a `certifiedAddress` which can execute arbitrary code if the adapter calls back to the user proxy.

Not only does this allow for significant privilege escalation if the operators are only trusted by the fact that the adapters are restricted, right now the codebase also does not unset the certified address.

If an operator is only trusted because the adapter is sufficiently safeguarded, an operator can bypass this safety mechanism completely by providing a malicious certified address contract. Once they execute `ForwardExecuteMultiCall` with the malicious contract, they can then at a later point in time execute callback to steal all user funds or even selfdestruct the proxy.

It should be noted that even if the `CertifiedAddress` is always unset, a malicious operator could do this if they somehow are able to execute any code during the adapter execution.



**Recommendation**

Consider whether the `_certifiedAddress` logic is strictly necessary. We are not huge fans of it as it seems to needlessly complicate the proxy. In case it is not strictly necessary, we recommend removing it for now as it can always be re-introduced through a proxy upgrade (which the client should of course be careful with as well).

In case the `CertifiedAddress` is really necessary, the client will need to treat it with a lot more care as it is a large security vulnerability. These addresses should be carefully validated and always unset after a multicall is finished.

**Resolution** RESOLVED

`CertifiedAddresses` have been replaced with a new type of logic. This new type of logic does allow for privilege escalation from call adapters to delegatcalling anything. users should therefore carefully acknowledge this privilege escalation.

<b>Issue #08</b>	<b>Privilege escalation risk: onFlashLoan multicall callback is a full privilege escalation and allows governance to potentially drain all wallets</b>
<b>Severity</b>	 HIGH SEVERITY
<b>Description</b>	<p>The <code>_multiCall</code> call within the <code>onFlashLoan</code> function allows for complete privilege escalation. Any operator that can execute flashloans can therefore drain the contract.</p> <p>Secondly, and the reason why this vector is marked as high severity, the <code>onFlashLoan</code> function can be called directly by any contract that is whitelisted by governance. Using the vector above, the governance can therefore whitelist a malicious contract and drain all of its users' wallets. The user approval flow which is intended to occur can therefore be completely circumvented through this avenue.</p>
<b>Recommendation</b>	Consider whether the <code>onFlashLoan</code> hook is strictly necessary. We recommend extracting flashloan logic into helper contracts managed by adapters and to keep this logic out of the core.
<b>Resolution</b>	 PARTIALLY RESOLVED
	<p>The governance risk is no longer present as the user now needs to explicitly approve the flashloan provider. Each user starts by approving the default flashloan provider.</p>

**Issue #09****Validation on `withdrawAssetsToAccount` is almost completely useless****Severity** MEDIUM SEVERITY**Description**

The `withdrawAssetsToAccount` function is supposed to validate that the destination of the withdrawal is another user-owned proxy wallet. However, the checks to validate this are extremely insufficient.

The code presently compares the codehashes of both contracts (the origin and destination) to be equal in an attempt to validate that the destination is in fact a wallet. Then it checks that the wallet proxy owner is in fact the current owner.

What the developer failed to realize is that the code of both the contracts is in fact just the proxy bytecode, hence it does not say anything about the implementation. The implementation of the receiver could therefore be a terribly malicious contract that exposes a fake owner() function indicating it is owned by the current owner.

**Recommendation**

Consider keeping a registry in the `WalletFactory` and simply validating that the `_account` parameter was deployed by the current owner. It should be reiterated that one cannot rely on the owner of the `_account` as a malicious user can upgrade their proxy to return a fake owner, even if it was deployed by the `WalletFactory`.

The main valid solution is to only allow transferring to wallets you are the "creator" of.

It should be noted that such a registry actually exists in the `ControllerLink` contract, which could be used for this purpose.


**Resolution** RESOLVED

The function has been removed.

**Issue #10**

**Funds could become permanently lost if a CertifiedAddress or adapter ever contains the SELFDESTRUCT opcode as it would delete the proxy**

**Severity**

 LOW SEVERITY

**Description**

Solidity has a special opcode to delete a contract from the blockchain. It is commonly used to clean-up temporary contracts as it gives a gas rebate. However, since the ControllerLib often delegatecalls to other contracts, it could accidentally delegatecall to a contract with a SELFDESTRUCT opcode.

Of course, all contracts that are delegatecalled to should be validated, audited and well-tested. In theory there should never be a scenario where the proxy self-destructs. However, since there is a way to actually recover the proxy if this ever were to happen, we have included this as an explicit issue.

**Recommendation**


Consider using deterministic deployment in the `upgradableWalletCreate` function within the `WalletFactory`. This can be done with almost minimal change:

WalletFactory::98

```
bytes32 salt = keccak256(abi.encode(msg.sender,
walletName));
proxyWallet newAccount = new proxyWallet(salt: salt)(logic,
admin, data);
```

If the wallet is ever deleted and the user is allowed to call `createAccount` again with the same `walletName`, it would deploy the same proxy to the same address. In other words, if the proxy wallet is ever selfdestructed by accident, this logic would allow you to recover it.

**Resolution**

 ACKNOWLEDGED

**Description**

The ControllerLib controls various specific functions that could be provided as adapters or be called directly by the user:

- `withdrawAsset`
- `withdrawAssets`
- `withdrawAssetsToAccount`
- `approve`
- `approveTokens`

These functions are limited because if the user wishes to approve an ERC-721 or ERC-1155, they would still need to resort to the traditional generic way of calling them. The only merit we see in having specific functions for these is that they reduce phishing risk, but this merit might not outweigh the downside of these functions bloating the absolutely most core component of the codebase.

Secondly, the ControllerLib contains various specific callback logic which seems highly restrictive:

- `callback` function
- `onFlashLoan` function

These are restricted to specific callback types. For example, Uniswap flashloan callbacks are not supported (so are many others).

Usually, a helper contract is used to execute the callback logic. This way, the logic does not need to occur in the proxy itself (this is how this problem is generally solved).

However, if the developer prefers to have a generic proxy callback functionality, the cleanest way to do it would be something similar to the following:



---

```

address public currentAdapter;

function delegatecallonAdapter(...) internal {
    currentAdapter = adapter;
    adapter.delegatecall(...);
    ...;
    currentAdapter = address(0);
}

fallback() external payable {
    if (currentAdapter != address(0)) {
        currentAdapter.delegatecall(...);
    }
}

```

This would of course need to be extremely carefully considered as there are a lot of security perspectives to think about (what if a malicious party can execute your fallback while an adapter is set...). The larger question is: Should `callback/onFlashLoan` occur at all within the proxy if the generic approach is not used. Perhaps at this point it is cleaner and simpler to use the traditional helper contract approach.

---



<b>Recommendation</b>	Consider removing this logic in favor of having it be executed at the adapter level.
-----------------------	--



---



<b>Resolution</b>	<div style="display: inline-block; border: 1px solid #ccc; border-radius: 10px; padding: 2px 5px; background-color: #f0f0f0;"> <div style="display: inline-block; width: 10px; height: 10px; background-color: #555; border-radius: 50%; margin-right: 5px;"></div>             ACKNOWLEDGED         </div>
-------------------	---



---







<b>Issue #12</b>	<b>adapManager, advancedOptionEnable and CertifiedAddress are private</b>
<b>Severity</b>	 LOW SEVERITY
<b>Description</b>	Important variables that third-parties might want to inspect should be marked as public so that these third-parties can easily inspect them through the explorer, web3 and derivative contracts.
<b>Recommendation</b>	Consider marking the variables as public.
<b>Resolution</b>	 RESOLVED

<b>Issue #13</b>	<b>callBytes of _callOnAdapter unnecessarily contains costETH</b>
<b>Severity</b>	 LOW SEVERITY
<b>Location</b>	<u>Line 95</u> costETH := mload(add(add(_callBytes, 32), 32))
<b>Description</b>	<p>The contract unnecessarily encodes the costETH in the callBytes, which requires it to do a low-level decoding of this parameter.</p> <p>As the costETH is actually never used by the adapter manager or adapter itself, it is not strictly relevant to the callBytes.</p>
<b>Recommendation</b>	Consider simply providing costEth as a parameter to multiCall and executeOnAdapter. It should be noted that msg.value kind of becomes meaningless in multiCall and the client should be careful with relying on it within that function.
<b>Resolution</b>	 RESOLVED

Issue #14	onFlashLoan does not validate _multiCall parameter lengths
Severity	 LOW SEVERITY
Description	The onFlashLoan function does not validate the _multiCall parameter lengths to be equal. This is inconsistent with how the multicall is called with validated parameters in the callProxy.
Recommendation	Consider validating that the _multiCall parameters are equal.
Resolution	 RESOLVED

Issue #15	Unused import: Record and ProxyAdmin
Severity	 INFORMATIONAL
Location	<p><u>Line 8</u></p> <pre>import "@openzeppelin/contracts/proxy/transparent/ProxyAdmin.sol";</pre>
Description	<p>The Record contract is inherited but the isAuth method is never used as the contract uses the ControllerLink contract to store users.</p> <p>Files that are imported in a contract but not used within said contract could confuse third-party auditors. They also increase the contract length unnecessarily.</p>
Recommendation	Consider removing the import to keep the contract short and simple.
Resolution	 RESOLVED

<b>Issue #16</b>	<b>Unused event: ChangeAutomation</b>
<b>Severity</b>	 INFORMATIONAL
<b>Description</b>	Events which are defined in a contract but remain unused could confuse third-party auditors. They also increase the contract length unnecessarily.
<b>Recommendation</b>	Consider removing the event to keep the contract short and simple.
<b>Resolution</b>	 RESOLVED

<b>Issue #17</b>	<b>Lack of events for setCertified, setAdapManager, setAdvancedOption and other various common functions</b>
<b>Severity</b>	 INFORMATIONAL
<b>Description</b>	<p>Functions that affect the status of sensitive variables should emit events as notifications.</p> <p>All other commonly called functions should also emit events, such as multical1, executeOnAdapter, withdrawAsset, etc.</p>
<b>Recommendation</b>	Add events for the functions.
<b>Resolution</b>	 RESOLVED



Issue #18	Typographical errors
Severity	<div data-bbox="456 203 485 232"></div> INFORMATIONAL
Description	<p>The contract contains a number of typographical errors which we have consolidated below in a single issue in an effort to keep the report size reasonable.</p> <p><u>L50</u>  <code>modifier onlyPermit()</code></p> <p>The onlyPermit modifier is in fact onlyAutomationOrOwner.</p> <p><u>L67</u>  <code>address private CertifiedAddress</code></p> <p>Variables should start with a lowercase letter.</p> <p><u>L95</u>  <code>costETH := mload(add(add(_callBytes, 32), 32))</code></p> <p>Consider doing <code>add(_callBytes, 64)</code> to save some gas.</p> <p><u>L185</u>  <code>function _transferAsset()</code></p> <p>The contract uses <i>transfer</i> instead of <i>transfer</i> in various function names.</p> <p>Finally, automation would be more adequately called callProxy (or CallProxy should be called ProxyAutomation). It took a few minutes during our first architectural meeting to figure out that CallProxy was actually the automation address.</p>
Recommendation	Consider fixing the typographical errors.
Resolution	<div data-bbox="456 1693 485 1722"></div> RESOLVED

---

## 2.4 WalletFactory

WalletFactory creates the users' proxy wallets. These wallets are "virtual users" managed by the user. Essentially they are identical to a regular wallet controlled by the user, but they can be controlled programmatically by other adapters as well. This means that the user could for example give approval to a secondary system to execute a limit order for them once the price of AVAX/USDC dips below a certain threshold on Trader Joe.

When the user calls `createAccount`, the contract will create a new `ProxyWallet` and a `ProxyAdmin` contract. The `ProxyWallet` represents the user's virtual wallet while the `ProxyAdmin` is a contract which the user can use to upgrade the proxy to a new implementation if they ever want to.



The wallet created needs to have the same bytecode as the one set by admins.



### 2.4.1 Privileged Functions



The following functions can be called by the various privileged roles of the contract:

- `setTrustLogic [ sub-account section added post-audit ]`
- `renounceOwnership`
- `transferOwnership`



## 2.4.2 Issues & Recommendations



Issue #19	Phishing risk: Malicious admins can be used
Severity	 MEDIUM SEVERITY
Location	<u>L34</u> mapping(address => address[]) public userProxyAdmin;
Description	<p>Consider only allowing the user to recycle an admin to avoid phishing risk. Right now there is no code validation done on the admin contract address that is provided by the user and will be allowed to upgrade their virtual wallet.</p> <p>To do so, consider using an enumerableSet instead of an array of addresses to be able to check that the provided admin was created by the user.</p> <p><u>L95</u> address admin,</p> <p>A malicious front end could provide a malicious admin.</p> <p>Although an owner ( ) check is done, this is insufficient given that a malicious contract could misrepresent this.</p> <p>In theory, the _data parameter in createAccount could also be maliciously changed as a phishing vector. We however have no clean recommendation to resolve this vector so the issue will still be resolved regardless of this being addressed.</p>
Recommendation	Consider using an EnumerableSet and checking that the provided admin was created by the user. By using an EnumerableSet instead of an array, the client can check in $O(1)$ that the admin address was in fact created by the contract for that user at some point in the past. Therefore, the contract can enforce that this parameter solely uses recycled admin contracts.
Resolution	 RESOLVED Only wallet admins that match the bytecode of the trusted admin contract can be provided.

<b>Issue #20</b>	<b>userDatabase and timelock are private</b>
<b>Severity</b>	 LOW SEVERITY
<b>Description</b>	Important variables that third-parties might want to inspect should be marked as public so that these third-parties can easily inspect them through the explorer, web3 and derivative contracts.
<b>Recommendation</b>	Consider marking the variables as public.
<b>Resolution</b>	 RESOLVED



<b>Issue #21</b>	<b>userProxyAdmin may be outdated and may have a shorter length to userAccount</b>
<b>Severity</b>	 INFORMATIONAL
<b>Location</b>	<u>L83</u> userProxyAdmin[msg.sender].push(newProxyAdminAddr);
<b>Description</b>	<p>userProxyAdmin is not updated when an admin is recycled. This means that the userProxyAdmin array might have a different length to the userAccount array.</p> <p>Additionally, proxyAdmin can still be changed after deployment so this array may be outdated if the ownership is transferred.</p>
<b>Recommendation</b>	<p>Consider adding the proxyAdmin to the array even if it is reused unless this is explicitly desired. Also, an enumerableSet would be more secure against phishing attacks as discussed in a previous issue (it should be noted that once an EnumerableSet is used, this recommendation cannot be implemented).</p> <p>The transferred owner issue could be fixed by dynamically generating the userProxyAdmin array by looping over all user proxies, however, this might cause some gas issues.</p>
<b>Resolution</b>	 RESOLVED This has been removed completely.





Issue #22 Unused Ownable inheritance	
Severity	 INFORMATIONAL
Location	<u>Line 33</u> contract WalletFactory is Ownable, Basic
Description	Even though the Ownable contract is inherited, the onlyOwner modifier is never used. This also increases the deployment gas cost and the contract length unnecessarily.
Recommendation	Consider removing the inheritance to keep the contract short and simple.
Resolution	 RESOLVED The ownable dependency has been removed completely.

Issue #23 Lack of events for setCodeHash and createAccount	
Severity	 INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for the functions.
Resolution	 RESOLVED



Issue #24	Typographical error
Severity	 INFORMATIONAL
Location	<u>L48</u> codeHash := extcodehash(_trustLogic)
Description	Consider using the getCodeHash function instead.
Recommendation	Consider fixing the typographical error.
Resolution	 RESOLVED The logic has been removed.

Issue #25	UI functions getUserProxyAdmin and getUserAccount can run out of gas
Severity	 INFORMATIONAL
Location	<u>Lines 117, 125</u> function getUserProxyAdmin(address owner) function getUserAccount(address owner)
Description	The contract contains functionality that can revert due to the nature of their implementation nature. As the state of the contract expands, this functionality might become so expensive that the gas cost does not fit in a single block and would become impossible to call. As RPCs also have various rate limiting methods, the functionality might become inaccessible even sooner.
Recommendation	Consider adding a length function to those arrays so they can be called one by one if those functions ever run out of gas.
Resolution	 RESOLVED The functions have been removed.

---

## 2.5 CallProxy/CallProxyLib



CallProxy is an upgradeable contract that uses the CallProxyLib as its implementation. It is the core authorization contract used by all wallets. Operators need to go through CallProxy if they wish to execute automation tasks on a user wallet. CallProxy will then call ERC2612Verifier to check if the operator has the required permission to execute the specific action for the user.

### 2.5.1 Privileged Functions



The following functions can be called by the various privileged roles of the contract:



- `setFlashLoanWhiteList [ TimeLock ]`
- `transferOwnership [ owner ]`
- `renounceOwnership [ owner ]`
- `setAccountVerifier [ owner of the proxyWallet ]`



## 2.5.2 Issues & Recommendations



<b>Issue #26</b>	<b>Governance Issue: The implementation is a proxy</b>
<b>Severity</b>	 HIGH SEVERITY
<b>Description</b>	<p>A malicious governance could change the implementation to steal users' tokens. Additionally, a malicious governance could set a bad publicVerifier to bypass the isTxPermitted check and steal users' token.</p>
<b>Recommendation</b>	<p>Consider not making the ControllerLib a proxy as adding the burden on the user to potentially re-approve a contract is definitely desired over adding governance risk.</p> <p>The timelock should however become transferable at this point, as it cannot be changed through an upgrade.</p>
<b>Resolution</b>	 RESOLVED
	<p>The contract is no longer deployed as a proxy. Timelock is still not transferable.</p>



  

<b>Issue #27</b>	<b>The initialize function does not call the initialize function safely</b>
<b>Severity</b>	 LOW SEVERITY
<b>Description</b>	<p>The contract calls the unchained initializer and calling all the parent initializer by hands — this is not recommended as one could be forgotten.</p> <p>Initializer functions are not linearized by the compiler like constructors. Because of this, each <code>__{ContractName}_init</code> function embeds the linearized calls to all parent initializers.</p>
<b>Recommendation</b>	<p>Consider using the <code>init</code> function that embeds the parent initializers.</p>
<b>Resolution</b>	 RESOLVED
	<p>The contract is no longer a proxy.</p>

<b>Issue #28</b>	<b>Adapter load-in allows for loading in a 32 bytes value while an address is just 20 bytes</b>
<b>Severity</b>	 LOW SEVERITY
<b>Location</b>	<u>Line 90</u> <code>adapter := mload(add(add(callBytes, 12), 20))</code>
<b>Description</b>	<p>The adapter load-in loads in a whole word in the adapter address slot. This would allow a malicious party to hide bits into the last 12 bytes of the second word of the <code>callBytes</code>. This gives a malicious user excessive control over the memory as they are traditionally not allowed to do this (Solidity reverts if this is attempted with high level <code>abi.decode</code> code).</p> <p>This issue is marked as low severity as we could not find a way to exploit the contract with those last 12 bytes, however, we still highly recommend rectifying it.</p>
<b>Recommendation</b>	Consider using high level <code>abi.decode</code> code instead. Within a system where security is as crucial as it is here, our opinion is that the team should not over optimize for gas with clever solutions.
<b>Resolution</b>	 ACKNOWLEDGED

Issue #29 Unused inheritance: OwnableUpgradeable.sol	
Severity	 INFORMATIONAL
Location	<u>L12</u> contract CallProxyLib is Initializable, OwnableUpgradeable, Basic
Description	Files that are imported in a contract but not used within the contract could confuse third-party auditors. They also increase the contract length unnecessarily.
Recommendation	Consider removing the inheritance to keep the contract short and simple.
Resolution	 RESOLVED

Issue #30 Unused event: CallForwardSignle	
Severity	 INFORMATIONAL
Description	Events which are defined in a contract but remain unused could confuse third-party auditors. They also increase the contract length unnecessarily. In addition, the event is also misspelled.
Recommendation	Consider removing the event to keep the contract short and simple.
Resolution	 RESOLVED This event has been removed.

<b>Issue #31</b>	<b>Lack of events for <code>setPublicVerifier</code>, <code>setAccountVerifier</code> and <code>setFlashLoanWhiteList</code></b>
<b>Severity</b>	 INFORMATIONAL
<b>Description</b>	Functions that affect the status of sensitive variables should emit events as notifications.
<b>Recommendation</b>	Add events for the above functions.
<b>Resolution</b>	 RESOLVED



**Description**

The contract contains a number of typographical errors which we have consolidated below in a single issue in an effort to keep the report size reasonable.

L8-9

```
import "../controller/ControllerLib.sol";  
import "../verifier/ERC2612Verifier.sol";
```

Importing an interface here instead of the whole implementation would have been sufficient. This would reduce the verified code size significantly in the explorer.

L48

```
address owner_,
```

This parameter can be removed as it must be `msg.sender`.

L69

```
function _excuteVerifyBasic(
```

L84

```
function _excuteVerifyAdapter(address account, bytes memory  
callBytes)
```

The functions should be named *execute* instead of *excute*.

L84

```
function _excuteVerifyAdapter(address account, bytes memory  
callBytes)
```

This function is inconsistent with `_excuteVerifyBasic` as `_excuteVerifyBasic` contains an `operator_` parameter. Consider being consistent and removing the parameter from `_excuteVerifyBasic`.



---

L90

```
adapter := mload(add(add(callBytes, 12), 20))
```

Consider doing `mload(add(callBytes, 32))`.

L184-190

```
function doFlashLoan(
    address loanProvider,
    address account_,
    address token,
    uint256 amount,
    bytes calldata payload
) public {
```

doFlashLoan can be made external, although we believe it should ideally be removed.

---

<b>Recommendation</b>	Consider fixing the typographical errors.
-----------------------	---

---

<b>Resolution</b>
-------------------



Most of the errors have been fixed.

---

<b>Issue #33</b>	<b>multicall functions lack non-zero checks</b>
------------------	---

<b>Severity</b>
-----------------



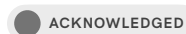
<b>Description</b>
--------------------


Users can execute a multicall with zero length parameters on user accounts. This might not be desired as it could put off some users.

<b>Recommendation</b>
-----------------------

Consider validating that the length of the parameters are greater than zero.

<b>Resolution</b>
-------------------



**Issue #34****permit can be frontrun and cause denial of service****Severity** INFORMATIONAL**Description**

If permit is executed twice, the second execution will be reverted. It is thus in theory possible for a bot to pick up permit transactions in the mempool and execute them before a contract can.

The implications of this issue is that a bad actor could prevent a user from using the permit flow. It is a denial-of-service attack which is present in most permit contracts.

**Recommendation**

Consider this if the permit flow ever stops working. If the client wishes to deal with it explicitly, they can decide to ignore the permit signature if the permission was already granted.

**Resolution** RESOLVED

The client will consider this.



---

## 2.6 ERC2612Verifier

ERC2612Verifier will allow 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. A user can also 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.



### 2.6.1 Privileged Functions



The following functions can be called by the various privileged roles of the contract:



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

## 2.6.2 Issues & Recommendations

Issue #35      The current implementation limit the number of adapter to 240	
Severity	 LOW SEVERITY
Description	The first 16 bits are reserved for basic operations while the last 240 bits will be allocated to the adapters. Because of this design, it is not possible to add more than 240 adapters.
Recommendation	<p>Consider using an enumerableSet for approval. If the goal was to be able to approve more than one adapter in one transaction, consider using a for loop as the increase in gas cost will not be too noticeable as the approval lasts in perpetuity.</p> <p>To combat phishing, it might be especially valuable to explicitly validate the adapter addresses instead of using the current gas optimized solution. Validating the approval transaction is terribly difficult right now for a user.</p>
Resolution	 ACKNOWLEDGED The client has indicated they will fix this nearer to the limit.

Issue #36      The OperatorUpdate event lacks the approval type variable	
Severity	 INFORMATIONAL
Description	The OperatorUpdate event lacks important information: the approval type variable. Additionally, there is no way to differentiate them as the approvalType is not a parameter of this event since it used as the event for approval and revocation.
Recommendation	Consider adding the approvalType to the event.
Resolution	 RESOLVED

<b>Issue #37</b>	<b>DOMAIN_SEPARATOR can be made immutable</b>
<b>Severity</b>	 INFORMATIONAL
<b>Description</b>	Variables that are only set in the constructor but never modified can be indicated as such with the immutable keyword. This is considered best practice since it makes the code more accessible for third-party reviewers and saves gas.
<b>Recommendation</b>	Consider making the variable explicitly immutable.
<b>Resolution</b>	 RESOLVED

<b>Issue #38</b>	<b>approvals, approve and revoke can be made external</b>
<b>Severity</b>	 INFORMATIONAL
<b>Description</b>	Functions that are not used within the contract but only externally can be marked as such with the external keyword. Apart from being a best practice when the function is not used within the contract, this can lead to a lower gas usage in certain cases.
<b>Recommendation</b>	Consider marking the functions mentioned above as external.
<b>Resolution</b>	 RESOLVED



---

## 2.7 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.



### 2.7.1 Privileged Functions



The following functions can be called by the various privileged roles of the contract:

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



## 2.7.2 Issues & Recommendations

<b>Issue #39</b>	<b>The removeAuth function can be called multiple time with the same values</b>
<b>Severity</b>	 MEDIUM SEVERITY
<b>Description</b>	<p>The removeAuth only checks that the accountId was set, but fails to reset that value back to 0. This allows users to call removeAuth multiple times with the same account, artificially reducing the total count. This may prevent other users to be able to call removeAuth as the function may revert because the count would be lower than the total amount of wallets.</p> <p>A malicious party will simply call removeAuth as many times as there are accounts to prevent any further removal of accounts due to the count subtraction underflow reverting.</p>
<b>Recommendation</b>	Consider resetting the accountID[_account] value to 0, so users will not be able to call removeAuth multiple times with an already deleted account.
<b>Resolution</b>	 RESOLVED

<b>Issue #40</b>	<b>trustFactory and timeLock are private</b>
<b>Severity</b>	 LOW SEVERITY
<b>Description</b>	Important variables that third-parties might want to inspect should be marked as public so that these third-parties can easily inspect them through the explorer, web3 and derivative contracts.
<b>Recommendation</b>	Consider marking the variables as public.
<b>Resolution</b>	 RESOLVED

**Description**

The contract contains multiple sections of code that could be further optimized for gas efficiency. We have consolidated these issues into a single issue in an effort to keep the report brief and readable.

L17

```
address private timeLock;
```

timeLock can be marked as immutable to save gas whenever it is used.

L76-79

```
accountID[_account] = accounts; // @audit gas store values /  
use param values  
accountAddr[accounts] = _account;  
addAccount(_owner, accountID[_account]);  
addUser(_owner, accountID[_account]);
```

Consider caching the accountID[\_account] value to reduce gas cost.

L104

```
function addUser(address _owner, uint64 _account) internal
```

L126

```
function removeUser(address _owner, uint64 _account)  
internal
```

The accountLink and accountList behave like a linked list but the account added is always new as it is a counter that only increases. Using such a storage unnecessarily increases the gas cost while making it harder for users to access the important values.

Consider using a simple mapping of uint256 to the user to make the contract simpler and shorter. This will furthermore reduce gas cost.



---

L137

```
function add(uint64 x, uint64 y) internal pure returns  
(uint64 z)
```

L141

```
function sub(uint64 x, uint64 y) internal pure returns  
(uint64 z)
```

Checking for overflow and underflow is unnecessary when using Solidity >= v0.8 as it is now natively checked. Consider using normal math instead of SafeMath.

---

**Recommendation** Consider implementing the gas optimizations mentioned above.

**Resolution**

 PARTIALLY RESOLVED

add and sub have been removed.

---



Issue #42	Typographical and minor errors
Severity	<div data-bbox="456 203 485 232"></div> INFORMATIONAL
Description	<p data-bbox="448 286 1370 416">The contract contains a number of typographical or minor errors which we have consolidated below in a single issue in an effort to keep the report size reasonable.</p> <p data-bbox="448 506 533 535"><u>L5, 8</u></p> <pre data-bbox="448 551 1286 624">import "@openzeppelin/contracts/access/Ownable.sol"; contract ControllerLink is Ownable {</pre> <p data-bbox="448 667 1072 701">Ownable is unused throughout the contract.</p> <p data-bbox="448 790 501 819"><u>L48</u></p> <pre data-bbox="448 835 1272 909">function initialize(address _trustFactory) external onlyTimeLock {</pre> <p data-bbox="448 952 1390 1032">This function would be more adequately named setTrustFactory as it is not just an initializer.</p>
Recommendation	Consider fixing the typographical and minor errors.
Resolution	<div data-bbox="456 1144 485 1173"></div> ACKNOWLEDGED



---

## 2.8 Record

Record is a simple RBAC contract that allows various addresses to be marked as “authorized”. Any authorized account can add and remove other authorized accounts.



### 2.8.1 Privileged Functions



The following functions can be called by the owner of the contract:



- `enable`
- `disable`



## 2.8.2 Issues & Recommendations

<b>Issue #43</b>	<b>Any authorized address can become the only authorized address</b>
<b>Severity</b>	 MEDIUM SEVERITY
<b>Description</b>	Any authorized address can disable any authorized address, including themselves. A malicious user that is authorized can become the only authorized address by disabling the other ones.
<b>Recommendation</b>	Consider only allowing an owner to enable and disable authorization. Ideally one should resort to OpenZeppelin's RBAC solutions.
<b>Resolution</b>	 RESOLVED The client has removed this contract.

<b>Issue #44</b>	<b>Contract lacks an easy way for users to figure out the list of authorized addresses through explorer contract inspection</b>
<b>Severity</b>	 INFORMATIONAL
<b>Description</b>	<p>The Record contract keeps track of a set of authorized wallets. However, there is no way for users to easily see the full list of authorized wallets. If such wallets ever have important functionality that could affect user funds, this might frustrate investors.</p> <p>Currently the only way to figure out who is authorized is to go back over all transactions of all authorized addresses, which is terribly difficult to organize.</p>
<b>Recommendation</b>	Consider using OpenZeppelin's enumerableSet to allow the users to iterate through the addresses they have authorized.
<b>Resolution</b>	 RESOLVED The client has removed this contract.

<b>Issue #45</b>	<b>Lack of events for <code>initAuth</code>, <code>enable</code> and <code>disable</code></b>
<b>Severity</b>	 INFORMATIONAL
<b>Description</b>	Functions that affect the status of sensitive variables should emit events as notifications.
<b>Recommendation</b>	Add events for the above functions, a single event with an address and boolean parameter would suffice for all three locations.
<b>Resolution</b>	 RESOLVED The client has removed this contract.





---

## 2.9 ProxyCallable

The ProxyCallable contract is a dependency used by the ControllerLib (the user wallet) to store the address of the CallProxy which is used for operators to interact with the wallet.



## 2.9.1 Issues & Recommendations

<b>Issue #46</b>	<b>Typographical error</b>
<b>Severity</b>	 INFORMATIONAL
<b>Location</b>	<u>L9</u> event AutomationTransferred( 
<b>Description</b>	This event is in fact an AutomationInitialised event and can be simplified as such. The previousAutomation value will always be address(0) so it can be removed.
<b>Recommendation</b>	Consider fixing the typographical error.
<b>Resolution</b>	 ACKNOWLEDGED

---

## 2.10 AdapterManager

AdapterManager is the main registry for all Cian adapters. An adapter is a smart contract which can be used by Cian operators to execute functionality for users on their wallets.

The manager can also be paused by various Cian-approved pause guardians. This prevents calls from being executed by operators on user wallets and can be used as an emergency safeguard if an adapter turns out to have a vulnerability.



### 2.10.1 Privileged Functions



The following functions can be called by the various privileged roles of the contract:



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





## 2.10.2 Issues & Recommendations

<b>Issue #47</b>	<b>delegatecalls are still possible even if the AdapterManager is paused</b>
<b>Severity</b>	 MEDIUM SEVERITY
<b>Description</b>	The ControllerLib presently does not check whether the AdapterManager is paused, which still allows operators to execute delegatecalls (and approval and flashloans) even when the AdapterManager is paused. If ever an adapter has a vulnerability that allows it to drain wallets, it might be insufficient to pause the AdapterManager and there might be nothing the Cian team can do to stop it.
<b>Recommendation</b>	Consider checking whether the AdapterManager is paused on all operator interactions.
<b>Resolution</b>	 PARTIALLY RESOLVED The paused state is presently checked in all delegatecall operations but not all adapter manager operations (eg. normal calls).

<b>Issue #48</b>	<b>unregisterAdapters does not reset adaptersIndex</b>
<b>Severity</b>	 LOW SEVERITY
<b>Description</b>	adaptersIndex is not reset during the unregisterAdapters function. Approval will therefore appear to remain granted to certain adapters.
<b>Recommendation</b>	Consider setting adaptersIndex back to zero whenever an adapter is unregistered.
<b>Resolution</b>	 RESOLVED

<b>Issue #49</b>	<b>The registerAdapters function allows the addition of too many adapters</b>
<b>Severity</b>	 LOW SEVERITY
<b>Description</b>	The registerAdapters function currently allows the index of the new adapter to be set to values greater than 256. Those adapters will never be able to be called because of current implementation, as any uint256 shifted by 256 bits or more will always return 0.
<b>Recommendation</b>	Consider asserting that the index of the adapter is lower or equal to 255 or change the approval logic.
<b>Resolution</b>	 RESOLVED

<b>Issue #50</b>	<b>_paused and suspendPermissions are private</b>
<b>Severity</b>	 LOW SEVERITY
<b>Description</b>	Important variables that third-parties might want to inspect should be marked as public so that these third-parties can easily inspect them through the explorer, web3 and derivative contracts.
<b>Recommendation</b>	Consider marking the variables as public.
<b>Resolution</b>	 RESOLVED



**Issue #51**

**Non-transferrable timelock address might be limiting if the client ever wants to move to a new governance structure**

**Severity**

 INFORMATIONAL

**Description**

Presently the timelock ownership address is immutable. If the governance ever wants to move to a different governance structure this might complicate such a matter.

**Recommendation**

Consider whether it is desired to allow the timelock to change the timelock address.

**Resolution**

 RESOLVED

The client added a new `TimelockCallable` contract that allows for a mutable timelock contract.



**Issue #52****Unused imports, functionality and typographical errors****Severity** INFORMATIONAL**Description**L7

```
import "../utils/AddressArrayLib.sol";
```

L13

```
using AddressArrayLib for address[];
```

Files that are imported in a contract but not used within said contract could confuse third-party auditors. They also increase the contract length unnecessarily.

The client likely included this at first and then moved to the better suited EnumerableSet structure which is currently used. We appreciate them not using this library as it is indeed not as ideal compared to the much more suitable EnumerableSet.

L24-27

```
event AdapterDeregistered(  
    address indexed adapter,  
    string indexed identifier  
);
```

This event is not used.

L133



```
require(adapterIsRegistered(_adapters[i]), "Adapter is not  
exist");
```



The error message should be "Adapter does not exist".

**Recommendation**

Consider resolving the above issues to keep the contract short and simple.

**Resolution** RESOLVED

Issue #53 Gas usage: getRegisteredAdapters might run out of gas	
Severity	 INFORMATIONAL
Description	getRegisteredAdapters increases in gas cost as more adapters are registered. Eventually an RPC might reject executing it because the gas cost is too expensive.
Recommendation	Consider adding a length and index-specific getter function.
Resolution	 RESOLVED The client has opted for an index-specific getter function.

Issue #54 Lack of events for setPauseWhiteList	
Severity	 INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for the function.
Resolution	 RESOLVED



<b>Issue #55</b>	<b>setPauseWhiteList check can be simplified</b>
<b>Severity</b>	 INFORMATIONAL
<b>Location</b>	<u>L169-173</u> <pre> if (val == false) {     require(suspendPermissions[partner], "No change."); } else {     require(!suspendPermissions[partner], "No change."); } </pre>
<b>Description</b>	This check can be simplified.
<b>Recommendation</b>	Consider simplifying this check to: <pre>require(suspendPermissions[partner] != val, "No change.");</pre>
<b>Resolution</b>	 RESOLVED



---

## 2.11 AdapterBase

AdapterBase contract contains the core functionality for any adapter. All of the adapters extend it.

Note that the privileged functions are present in all adapters but will not be repeated from here on out.



### 2.11.1 Privileged Functions



The following functions can be called by the various privileged roles of the contract:

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







## 2.11.2 Issues & Recommendations



<b>Issue #56</b>	<b>Governance privilege: The timelock can potentially sweep wallets through reentrancy during delegatecalls</b>
<b>Severity</b>	 INFORMATIONAL
<b>Description</b>	AdapterBase contains a function sweep that allows the timelock to take out any tokens in the adapter. However, if a <code>delegatecall</code> would be made to sweep by the timelock on a user proxy, it would allow the timelock to sweep funds from the user wallet
<b>Recommendation</b>	Consider disabling delegatecalls to sweep by using a modifier exactly opposite to <code>onlyDelegation</code> .
<b>Resolution</b>	 RESOLVED sweep can no longer be called through delegation.

<b>Issue #57</b>	<b>ADAPTER_NAME is inaccessible during delegate calls</b>
<b>Severity</b>	 INFORMATIONAL
<b>Description</b>	ADAPTER_NAME is only accessible through normal calls to the contract, unlike the other variables like ADAPTER_ADDRESS which are available during delegatecalls.
<b>Recommendation</b>	Consider moving ADAPTER_NAME to an abstract pure function that must be overridden by all adapters. This would allow the variable to become accessible even during delegatecalls.
<b>Resolution</b>	 ACKNOWLEDGED



Issue #58	Unused import: IWAVAX.sol
Severity	 INFORMATIONAL
Location	<u>L9</u> <code>import "../interfaces/IWAVAX.sol";</code>
Description	Files that are imported in a contract but not used within said contract could confuse third-party auditors. They also increase the contract length unnecessarily.
Recommendation	Consider removing the import to keep the contract short and simple.
Resolution	 RESOLVED

Issue #59	Typographical errors
Severity	 INFORMATIONAL
Description	<p>The contract contains a number of typographical errors which we have consolidated below in a single issue in an effort to keep the report size reasonable.</p> <p><u>L33</u>  <code>require(ADAPTER_ADDRESS != address(this), "Only For delegatecall.");</code></p> <p>"For" should not be capitalized in this error.</p> <p><u>L59</u>  <code>require(_token != address(0) &amp;&amp; _token != avaxAddr);</code></p> <p>This requirement lacks an explicit revert message.</p>
Recommendation	Consider fixing the typographical errors.
Resolution	 PARTIALLY RESOLVED

<b>Issue #60</b>	<b>pullTokensIfNeeded might not pull in enough tokens for tokens with a fee on transfer</b>
<b>Severity</b>	 INFORMATIONAL
<b>Description</b>	pullTokensIfNeeded might not pull in enough tokens for tokens as a fee on transfer as the contract would receive less tokens than requested.
<b>Recommendation</b>	Consider this behavior carefully. No changes need to be made as it's inherent to token pulling behavior. If tokens with a fee on transfer ever needs to be supported, a before-after pattern should be considered.
<b>Resolution</b>	 RESOLVED <p>The client has indicated that they will consider this carefully. The likelihood of this function not pulling in sufficient tokens has even increased after the audit, as the function no longer reverts if there were insufficient tokens. We discussed this with the client and they have indicated that this behavior is desired.</p>



---

## 2.12 WavaxGateway



WavaxGateway is a simple adapter that allows for the depositing and withdrawal of WAVAX from and into AVAX.

It should be noted that withdrawing WAVAX straight into a proxy is generally a disliked practice 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 WAVAX instead of the approach which is taken here.

The WavaxGateway is a `delegationcall` adapter.



## 2.12.1 Issues & Recommendations



Issue #61	Typographical error
Severity	 INFORMATIONAL
Location	<u>L10</u> AdapterBase(_adapterManager, _timeLock, "WavxGateway")
Description	The adapter's name should be WavaxGateway.
Recommendation	Consider fixing the typographical error.
Resolution	 RESOLVED

---

## 2.13 SAVAXAdapter

SAVAXAdapter is an adapter to stake into and unstake from Benqi's liquid staking SAVAX (Implementation) solution. SAVAXAdapter is a delegation adapter.

## 2.13.1 Issues & Recommendations

<b>Issue #62</b>	<b>SAVAXAdapter lacks various functions</b>
<b>Severity</b>	 LOW SEVERITY
<b>Description</b>	The SAVAXAdapter lacks functionality to retrieve requested unlocks that are relocked. It also lacks functionality to redeem() and retrieve relocked unlocks by index as might be required if the looped versions consume too much gas.
<b>Recommendation</b>	Consider adding the missing functions.
<b>Resolution</b>	 ACKNOWLEDGED <p>The client has indicated the contract is not responsible for SAVAX unwrapping and that users should simply take it out to unwrap it. However, there are still redeem functions which means that the functionality is still somewhat there. We are therefore marking the issue as acknowledged.</p>

---

## 2.14 TraderJoeAdapter

TraderJoeAdapter is an adapter that allows users to use the TraderJoe DEX and farms within their wallets. It allows for the swapping of tokens, adding of liquidity and removing of liquidity. The CIAN team added a zap function to add liquidity from one token to a pair, and it also implements a way to optimally add liquidity when 2 tokens are provided in an unbalanced fashion.



TraderJoeAdapter is a delegation adapter for farming and a call adapter for swaps.



### 2.14.1 Privileged Functions

The following functions can be called by the various privileged roles of the contract:



- `swapTokensForExactTokens [ adapterManager ]`
- `swapExactTokensForTokens [ adapterManager ]`
- `addLiquidity [ adapterManager ]`
- `removeLiquidity [ adapterManager ]`
- `addLiquidityAVAX [ adapterManager ]`
- `removeLiquidityAVAX [ adapterManager ]`
- `depositLpToken [ only delegation ]`
- `withdrawLpToken [ only delegation ]`
- `enter [ only delegation ]`
- `leave [ only delegation ]`
- `addLiquidityCustomized [ adapterManager ]`
- `addLiquidityFromOneToken [ adapterManager ]`

## 2.14.2 Issues & Recommendations



<b>Issue #63</b>	<b>User can get sandwiched and suffer high slippage</b>
<b>Severity</b>	 MEDIUM SEVERITY
<b>Description</b>	The autoswap and the autoSwapFromOneToken internal functions do not set a minimum amount of tokens to receive. Users may get sandwiched and suffer high slippage.
<b>Recommendation</b>	Consider adding the slippage values within the encodedData to reduce the impact of a sandwich attack.
<b>Resolution</b>	 RESOLVED The client will be using a 1inch adaptor, however, it should be noted that this adaptor is not within the scope of this audit. Users should still be aware that this function is vulnerable to high slippage.



<b>Issue #64</b>	<b>The adapter does not allow users to call emergencyWithdraw</b>
<b>Severity</b>	 LOW SEVERITY
<b>Description</b>	The adapter does not allow users to call emergencyWithdraw. This should be added in case users ever need to call this during an emergency situation.
<b>Recommendation</b>	Consider adding a way to call the emergencyWithdraw function.
<b>Resolution</b>	 RESOLVED A function for emergencyWithdraw has been added.







<b>Issue #65</b>	<b>JoeBar is now deprecated</b>
<b>Severity</b>	 LOW SEVERITY
<b>Description</b>	<p>The adapter allows users to enter and leave from JoeBar or xJOE. This feature has now been deprecated as users no longer receive incentives to stake their JOE tokens inside JoeBar.</p> <p>These days, TraderJoe allows users to stake their JOE in 3 different ways: rJoe, veJoe and sJoe.</p>
<b>Recommendation</b>	Consider removing the functions related to JoeBar. Consider whether the new ways of staking JOE should be added. If so, it might make sense to add them as separate adapters to keep adapters short and modular.
<b>Resolution</b>	 RESOLVED The JoeBar logic was removed.





Issue #66	addLiquidityAVAX contains a seemingly redundant transfer
Severity	 LOW SEVERITY
Location	<p><u>L230-238</u></p> <pre> if (addInfo.amountTokenDesired &gt; _amountToken) {     IERC20(addInfo.tokenAddr).safeTransfer(         account,         addInfo.amountTokenDesired - _amountToken     ); } if (msg.value == _amountAVAX) {     IERC20 token = IERC20(addInfo.tokenAddr);     token.safeTransfer(account,         token.balanceOf(address(this))); } </pre>
Description	The second transfer seems to be redundant with the first.
Recommendation	Consider whether there is supposed to be any token dust after the first transfer executes. If so, consider whether either of the transfers can be removed.
Resolution	<p> RESOLVED</p> <p>The redundant transfer has been removed.</p>

<b>Issue #67</b>	<b>depositLpToken and withdrawLpToken can deposit into and withdraw from Masterchefs other than Trader Joe</b>
<b>Severity</b>	 LOW SEVERITY
<b>Description</b>	depositLpToken and withdrawLpToken can deposit tokens into and withdraw tokens from and into any Masterchef — this might be undesirable in case operators are not fully trusted. It should be noted that this contract in general should not allow untrusted operators on in any case.
<b>Recommendation</b>	Consider hardcoding the masterchefAddr as a variable.
<b>Resolution</b>	 RESOLVED There is now a list of trusted Masterchefs that the user can use. It should be noted that as this check is also done during a withdraw — if the list was to be updated and a Masterchef was removed, users that deposited to that specific contract may not be able to withdraw anymore.

<b>Issue #68</b>	<b>The JTokenSnapshot structure and IJToken import are unused</b>
<b>Severity</b>	 INFORMATIONAL
<b>Description</b>	The JTokenSnapshot structure and IJToken import are unused throughout the contract.
<b>Recommendation</b>	Consider removing the unused structure and import to keep the contract short and simple.
<b>Resolution</b>	 RESOLVED

<b>Issue #69</b>	<b>router can be made constant</b>
<b>Severity</b>	 INFORMATIONAL
<b>Description</b>	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.
<b>Recommendation</b>	Consider making the variable explicitly constant.
<b>Resolution</b>	 RESOLVED

<b>Issue #70</b>	<b>SafeMath is unnecessary starting from Solidity version 0.8</b>
<b>Severity</b>	 INFORMATIONAL
<b>Description</b>	Throughout the contract, the developers use SafeMath to protect the contracts against integer overflow. However, starting from Solidity version 0.8, such protections are baked into the standard math operators of Solidity. Therefore, it is no longer necessary to use SafeMath in the current version of the codebase. Using SafeMath in this version will slightly increase gas usage.
<b>Recommendation</b>	Consider removing the SafeMath dependency and reverting to standard math operators throughout all contracts starting from version 0.8.
<b>Resolution</b>	 RESOLVED

**Issue #71****Gas optimizations****Severity** INFORMATIONAL**Description**

We have consolidated the sections of code that could be further optimized for gas efficiency into a single issue in an effort to keep the report brief and readable.

- Throughout this adapter, the deadline of the swaps are set to be `block.timestamp + TIME_INTERVAL`. This is unnecessary as the swaps will be done within the same `block.timestamp`.
- Consider setting the deadline to `block.timestamp` (or `type(uint256).max` for even higher gas efficiency) or, even better, consider whether the deadline should be added to the `encodedData`.

**Recommendation**

Consider implementing the gas optimizations mentioned above.

**Resolution** RESOLVED

---

## 2.15 FeeBoxAVAX, FeeBoxSAVAX and FeeBoxToken

The FeeBox contracts are responsible for taking fees from the users' wallets to subsidize gas and management costs for the operators that execute automation jobs for them.



### 2.15.1 Privileged Functions



The following functions can be called by the various privileged roles of the contract:



- `initialize [ timelock ]`
- `setAdapterManager [ timelock, added after audit ]`
- `paymentCheck [ balanceController ]`
- `setBalance [ balanceController, added after audit ]`





## 2.15.2 Issues & Recommendations



<b>Issue #72</b>	<b>SAVAX redemptions might eventually relock or run out of gas</b>
<b>Severity</b>	 HIGH SEVERITY
<b>Location</b>	<code>FeeBoxSAVAX : :98</code> <code>ISAVAX(sAVAX).redeem();</code>
<b>Description</b>	<p>The redeem function of Benqi's sAVAX does not use constant gas. This means that as unlock requests accumulate, the gas usage of redeem increases, and this might cause redemptions to eventually run out of gas.</p> <p>More severely, if an unlock request is not redeemed in time, it will relock. When this happens, the user needs to explicitly call a method to receive sAVAX shares for those relocked tokens. However, the contract presently does not allow the calling of that method which means all the relocked AVAX tokens will be permanently lost under the current design.</p>
<b>Recommendation</b>	Consider removing the FeeBoxSAVAX contract in favor of a simple ERC20 FeeBox (essentially FeeBoxAVAX for ERC20 tokens) that transfers the sAVAX tokens as is to the feeReceiver. There is no advantage to handling the redemption at the protocol level like this as we are generally fans of the "less is more" principle.
<b>Resolution</b>	 RESOLVED The client now transfers the sAVAX directly to the fee receiver.

<b>Issue #73</b>	<b>Governance privilege: balanceController can take funds from users' accounts</b>
<b>Severity</b>	 MEDIUM SEVERITY
<b>Description</b>	The balanceController can freely take funds from users' accounts if they ever approve a feebox. We are unsure why this flow is desired or if users will give infinite approval to feeboxes. If they give infinite approval to feeboxes, this is a serious governance risk.
<b>Recommendation</b>	Consider whether it is the user that should actually sign the approval.
<b>Resolution</b>	 RESOLVED A tx.origin check is now done which validates that any transaction to deposit originated from the user wallet owner.

<b>Issue #74</b>	<b>Approval flow does not contain necessary safeguards</b>
<b>Severity</b>	 MEDIUM SEVERITY
<b>Description</b>	The approval flow does not use OpenZeppelin's ECDSA library, instead, it uses a low level ecrecover call to validate the signature. As ecrecover lacks various safeguards, this could be risky. For example, while initialize is not called and balanceController is still address(0), any signature can be faked as ecrecover simply returns address(0) when it fails to recover a signature due to bad inputs.
<b>Recommendation</b>	Consider moving to ECDSA by OpenZeppelin.
<b>Resolution</b>	 RESOLVED The contracts now use the ECDSA function.



Issue #75 <b>balanceController and feeReceiver are private</b>	
Severity	 LOW SEVERITY
Description	Important variables that third-parties might want to inspect should be marked as public so that these third-parties can easily inspect them through the explorer, web3 and derivative contracts.
Recommendation	Consider marking the variables as public.
Resolution	 RESOLVED This has been fixed within all contracts.

Issue #76 <b>Lack of events for initialize</b>	
Severity	 INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for the function.
Resolution	 RESOLVED



**Issue #77****Typographical and minor errors****Severity** INFORMATIONAL**Description**

The contract contains a number of typographical and minor errors which we have consolidated below into a single issue in an effort to keep the report size reasonable.

```
mapping(address => uint256) public tokenBlance;
```

The various balance mappings are misspelled as "blance".

```
function initialize(  
    address public sAVAX =  
    0x2b2C81e08f1Af8835a78Bb2A90AE924ACE0eA4bE;  
    sAVAX can be marked as constant.
```

This function is not an initializer as it can be called multiple times. Consider renaming it.

```
FeeBoxAVAX::88  
require(wavaxBlance[account] >= consumedAmount);
```

This requirement lacks an explicit return value.

```
FeeBoxSAVAX::74  
address public sAVAX =  
0x2b2C81e08f1Af8835a78Bb2A90AE924ACE0eA4bE;  
sAVAX can be marked as constant.
```

```
address public sAVAX =  
0x2b2C81e08f1Af8835a78Bb2A90AE924ACE0eA4bE;  
sAVAX can be marked as constant.
```

sAVAX can be marked as constant.

**Recommendation**

Consider fixing the errors above.



**Resolution** RESOLVED



---



## 2.16 VerifierBasic

VerifierBasic is used by the various FeeBoxes to validate signatures.

## 2.16.1 Issues & Recommendations

<b>Issue #78</b>	<b>recoverSigner allows anyone to fake signatures for the zero address</b>
<b>Severity</b>	 HIGH SEVERITY
<b>Location</b>	<u>Line 35</u> <code>return ecrecover(_ethSignedMessageHash, v, r, s);</code>
<b>Description</b>	The VerifierBasic contract uses a low level ecrecover call to validate a signature. This call is known to be vulnerable to various attacks like the fact that it returns 0 if any of the parameters are wrong. This allows anyone to fake signatures for the zero address which might be abused in contracts that inherit this Verifier.
<b>Recommendation</b>	Consider removing this contract and moving to OpenZeppelin's ECDSA.sol.
<b>Resolution</b>	 RESOLVED The client now use OpenZeppelin's ECDSA library.

Issue #79	Unused imports
Severity	 INFORMATIONAL
Location	<u>L5, 6</u> <pre>import "../base/AdapterBase.sol"; import {ISAVAX} from "../../interfaces/benqi/ISAVAX.sol";</pre>
Description	Files that are imported in a contract but not used within said contract could confuse third-party auditors. They also increase the contract length unnecessarily.
Recommendation	Consider removing the unused imports to keep the contract short and simple.
Resolution	 RESOLVED ISAVAX has been removed but AdapterBase has not.

Issue #80	Typographical error
Severity	 INFORMATIONAL
Location	<u>Line 18</u> <pre>"\x19Ethereum Signed Message\n" + len(msg) + msg</pre>
Description	This is not actually what's happening on the return statement under this comment.
Recommendation	Consider fixing this typographical error.
Resolution	 ACKNOWLEDGED

---

## 2.17 BankerJoeAdapter / BenqiAdapter

BankerJoeAdapter and BenqiAdapter are adapters that allows CIAN users to use the lending platform managed by the TraderJoe team, BankerJoe, and the one managed by Benqi. It allows users to lend and borrow assets to receive incentives from the lending platforms.



The two adapters are delegation adapters for deposits, borrows, repays, and claiming rewards, while it is a call adapter for withdrawals.

### 2.17.1 Privileged Functions



The following functions can be called by the various privileged roles of the contract:



- `initialize [ timelock ]`
- `deposit [ only delegation ]`
- `withdraw [ adapterManager ]`
- `enterMarkets [ only delegation ]`
- `exitMarket [ only delegation ]`
- `borrow [ only delegation ]`
- `repay [ only delegation ]`
- `claimReward / claimRewards (BankerJoeAdapter / BenqiAdapter) [ only delegation ]`



## 2.17.2 Issues & Recommendations

<b>Issue #81</b>	<b>Governance privilege: The timelock could be used to steal users' tokens</b>
<b>Severity</b>	 MEDIUM SEVERITY
<b>Description</b>	Governance could steal users' tokens by adding malicious tokens to the trusted list if they control the operator.
<b>Recommendation</b>	Consider checking that the assets added to the trusted list are registered within their respective comptroller. A call to <code>comptroller.markets(cToken).isListed</code> suffices (or <code>joetroller.isMarketListed</code> ).
<b>Resolution</b>	 RESOLVED The client has added the check to the respective contracts.



  



<b>Issue #82</b>	<b>repay may use an outdated value</b>
<b>Severity</b>	 MEDIUM SEVERITY
<b>Description</b>	The repay function uses <code>borrowBalanceStored</code> to get the total amount of AVAX needed to be sent to repay the user's loan. This value is likely outdated as it does not call <code>accrueInterest</code> , and this will result in users not being able to repay their entire loan.
<b>Recommendation</b>	Consider using <code>borrowBalanceCurrent</code> to have the current <code>borrowBalance</code> and allow users to repay their entire loan.
<b>Resolution</b>	 RESOLVED The client now uses the current borrow balance.



Issue #83 Operations do not revert if underlying protocols are paused	
Severity	 MEDIUM SEVERITY
Description	Both underlying protocols are based upon the Compound codebase which is notorious for not reverting on errors. This causes the overall operation to potentially not revert even if Banker Joe is for example paused.
Recommendation	Consider requiring the return values of all Banker Joe and Benqi interactions to return the zero value.
Resolution	 RESOLVED

Issue #84 Governance could delist a market making redemption of tokens complicated	
Severity	 LOW SEVERITY
Description	The governance could use the <code>initialize</code> function to delist a token, making the redemption of the underlying tokens complicated as the user should use a low-level call to do it, or transfer the tokens to one of its EOA.
Recommendation	Consider whether this is wanted or always allows users to withdraw or repay assets even if the governance delists them.
Resolution	 ACKNOWLEDGED



<b>Issue #85</b>	<b>GetUserDepositPosition returns wrong values</b>
<b>Severity</b>	 LOW SEVERITY
<b>Description</b>	GetUserDepositPosition returns wrong values as it should not be divided by token **decimals. Hopefully, as all lending tokens have the same decimals (i.e. 8), the returned value would simply not have the right number of decimals.
<b>Recommendation</b>	Consider fixing the function by using the following code:  <pre>assetValue += (tokenBalance * price * exchangeRateStored) / 1e36;</pre>
<b>Resolution</b>	 RESOLVED

<b>Issue #86</b>	<b>BankerJoeAdapter: Unused events — TraderJoeStake and TraderJoeUnstake</b>
<b>Severity</b>	 INFORMATIONAL
<b>Description</b>	Events which are defined in a contract but remain unused could confuse third-party auditors. They furthermore increase the contract length for no reason.
<b>Recommendation</b>	Consider removing the events to keep the contract short and simple.
<b>Resolution</b>	 RESOLVED

<b>Issue #87</b>	<b>BankerJoeAdapter: Unused variable — joeBarAddr</b>
<b>Severity</b>	 INFORMATIONAL
<b>Description</b>	Variables defined in a contract but not used within said contract could confuse third-party auditors. They also increase the contract length and bytecode size unnecessarily.
<b>Recommendation</b>	Consider removing the variable to keep the contract short and simple.
<b>Resolution</b>	 RESOLVED The variable was removed.





Issue #88	Typographical errors
Severity	<div data-bbox="456 203 485 232"></div> INFORMATIONAL
Description	<p data-bbox="448 286 1378 416">The contract contains a number of typographical errors which we have consolidated below in a single issue in an effort to keep the report size reasonable.</p> <p data-bbox="448 506 501 535"><u>L69</u></p> <pre data-bbox="448 548 772 580">function initialize(</pre> <p data-bbox="448 620 1385 703">This function is not an initializer as it can be called multiple times. Consider renaming it.</p> <p data-bbox="448 792 676 822"><u>BankerJoe::126</u></p> <pre data-bbox="448 835 892 866">//todo delete jtokenAddr arg</pre> <p data-bbox="448 907 1378 1086">The comment should be removed to keep the contract short and simple. Additionally, jTokenAddr should indeed be removed from the event so there is consistency between BenqiAdapter and BankerJoeAdapter.</p> <p data-bbox="448 1176 676 1205"><u>BenqiAdapter::</u></p> <pre data-bbox="448 1218 1385 1294">event BenqiWithdraw(address token, uint256 amount, address account);</pre> <p data-bbox="448 1335 1082 1366">The event should be named BenqiWithdraw.</p> <p data-bbox="448 1456 517 1485"><u>L245</u></p> <pre data-bbox="448 1498 983 1529">/// @return The calaculated value.</pre> <p data-bbox="448 1570 938 1608">The spelling should be <i>calculated</i>.</p>
Recommendation	Consider fixing the typographical errors.
Resolution	<div data-bbox="456 1718 485 1747"></div> PARTIALLY RESOLVED



---



## 2.18 JoeERC3156



The JoeERC3156 is a helper contract to perform flashloans using the BankerJoe lending platform.

## 2.18.1 Issues & Recommendations

Issue #89	Lack of authentication on flashLoan call
Severity	 HIGH SEVERITY
Description	<p>Anyone can initiate a flashloan, and if the goal is for this contract to ever be whitelisted as a flashloan provider (in flashLoanWhiteList) this would be a huge problem as it would allow for privilege escalation.</p> <p>We are however unsure whether this contract is supposed to be a flashloan provider or what its purpose is. If we follow the operation flow, it does not seem to be able to be used as a call-based adapter as the onFlashLoan call would be made with the adapter manager as the initiator. User wallets only allow such calls to be made if the initiator is in fact the CallProxy, which has a different interface for the flashloan management.</p>
Recommendation	<p>Consider explaining to us how this contract is supposed to be used. Consider adding validation to flashloan if it is supposed to be an adapter, and in that case, consider redesigning the flashloan hook as we believe it currently would not work.</p> <p>As discussed in the core section of this report, we generally dislike the way flashloans are managed within the core. If this is simply supposed to be a utility contract used by call adapters we do understand the merit, but it might make more sense to not allow the specification of a receiver in that case and always execute the callback on the originInitiator.</p>
Resolution	 RESOLVED JoeERC3156 has been removed.

<b>Issue #90</b>	<b>The initiator address of the onFlashLoan is not reliable</b>
<b>Severity</b>	 MEDIUM SEVERITY
<b>Description</b>	The BankerJoe implementation of flashloans diverges from the original EIP-3156 by allowing anyone to bypass the initiator authentication. Therefore, it cannot be relied on to be the <code>msg.sender</code> .
<b>Recommendation</b>	Consider not trusting the initiator of the flashloan and carefully check that no harm can be done.
<b>Resolution</b>	 RESOLVED JoeERC3156 has been removed.

<b>Issue #91</b>	<b>onFlashLoan callback is vulnerable to reentrancy</b>
<b>Severity</b>	 MEDIUM SEVERITY
<b>Location</b>	<u>L115-116</u> <pre>originTarget = address(0); originInitator = address(0);</pre>
<b>Description</b>	Unsetting the origin this late is not a good idea as this would allow onFlashLoan to be called twice in a reentrancy attack.
<b>Recommendation</b>	Consider immediately unsetting it before the callback.
<b>Resolution</b>	 RESOLVED JoeERC3156 has been removed.

<b>Issue #92</b>	<b>Governance privilege: The timelock could be used to steal users' tokens</b>
<b>Severity</b>	 MEDIUM SEVERITY
<b>Description</b>	The governance could steal users' tokens by setting a malicious address instead of the jTokens address in order to steal user's tokens when they call the flashLoan function.
<b>Recommendation</b>	Consider checking that the assets added to the trusted list are registered within the JoeTroller.
<b>Resolution</b>	 RESOLVED JoeERC3156 has been removed.



Issue #93	Typographical errors
Severity	<div data-bbox="456 203 485 232"></div> INFORMATIONAL
Description	<p data-bbox="448 286 1366 416">The contract contains a number of typographic mistakes which we've enumerated below in a single issue in an effort to keep the report size reasonable.</p> <p data-bbox="448 506 501 535"><u>L30</u></p> <p data-bbox="448 551 1417 622">* @dev From ERC-3156. The amount of currency available to be lended.</p> <p data-bbox="448 667 836 696">This should be "to be lent".</p> <p data-bbox="448 786 501 815"><u>L31</u></p> <p data-bbox="448 831 1155 860">* @param token The loan currency, <b>in</b> jToken.</p> <p data-bbox="448 904 1046 934">This is the ordinary token, not the jToken.</p> <p data-bbox="448 1023 501 1052"><u>L74</u></p> <p data-bbox="448 1068 1401 1189">* @param receiver The contract receiving the tokens, needs to implement the onFlashLoan(<b>address</b> user, <b>uint256</b> amount, <b>uint256</b> fee, <b>bytes</b> calldata) interface.</p> <p data-bbox="448 1234 1417 1305">The interface to implement is in fact different, as it also contains the token address as seen within the callback within onFlashLoan.</p> <p data-bbox="448 1395 501 1424"><u>L87</u></p> <p data-bbox="448 1440 900 1469">originInitator = msg.<b>sender</b>;</p> <p data-bbox="448 1514 1166 1543">Throughout the contract, initiator is misspelled.</p>
Recommendation	Consider fixing the typographical errors.
Resolution	<div data-bbox="456 1671 485 1700"></div> RESOLVED

JoeERC3156 has been removed.



---

## 2.19 Timelock

The Timelock contract 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.

Parameter	Value	Description
<b>Delay</b>	12 hours	The delay indicates the time the administrator has to wait after queuing a transaction to execute it.
<b>Minimum Delay</b>	12 hours	<p>The minDelay indicates the lowest value that the delay can minimally be set.</p> <p>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.</p>
<b>Grace Period</b>	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.

### 2.19.1 Issues & Recommendations

No issues found.



**PALADIN**  
BLOCKCHAIN SECURITY