SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

Carbonstarter

AUDIT

SECURITY ASSESSMENT

26. October, 2023

FOR

CARBONSTARTER

SolidProof_io

@solidproof_io

Introduction

SolidProof.io is a brand of the officially registered company MAKE Network GmbH, based in Germany. We're mainly focused on Blockchain Security such as Smart Contract Audits and KYC verification for project teams.
Solidproof.io assess potential security issues in the smart contracts implementations, review for potential inconsistencies between the code base and the whitepaper/documentation, and provide suggestions for improvement.

Disclaimer

SolidProof.io reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc'...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof's position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of the security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	Carbon Starter
Website	https://carbonstarter.xyz
About the project	CarbonStarter is a fully decentralized, community-governed launchpad designed to foster the growth and launch of early-stage projects. Built on Arbitrum, Carbonstarter provides a safe, trusted and verified place for innovations to launch and individuals to invest and grow.
Chain	Arbitrum
Language	Solidity
Codebase Link	Provided as Files (Private Repo)
Commit	N/A
Unit Tests	Partially Provided

Social Medias

Telegram	https://t.me/Carbonstarter
Twitter	https://twitter.com/Carbonstarter_
Facebook	N/A
Instagram	N/A
Github	N/A
Reddit	N/A
Medium	https://medium.com/@Carbonstarter_
Discord	https://discord.gg/carbonstarter-1087098740931821799
Youtube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Changelog
v1.0	25. October 2023	• Layout Project • Automated- /Manual-Security Testing • Summary
v1.1	26. October 2023	• Reaudit

Note - The following audit report presents a comprehensive security analysis of the smart contracts utilized in the project that includes outside manipulation of the contract's functions in a malicious way. This analysis did not include functional testing (or unit testing) of the contract/s logic. We cannot guarantee 100% logical correctness of the contract as we did not functionally test it. This includes internal calculations in the formulae used in the contract.

File Overview

The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name	SHA-1 Hash
contracts/TestERC20.sol	2bfdb3e0aa366a77a767676e1aef6f74777586bb
contracts/dex/interfaces/ICarbonPair.sol	53713eb40dd9afc14b9d39855930e5d3995340ca
contracts/dex/interfaces/ICarbonFactory.sol	7a7276bc82e9a53085844d43a0149e4c3ffcc856
contracts/dex/interfaces/ICarbonERC20.sol	41713df4bf3c4bcc85ccc88be73856a93eb0c331
contracts/dex/interfaces/IWETH.sol	f38b78bc02631c83b321016f4cb723aad1ff525b
contracts/dex/interfaces/ICarbonRouter02.sol	53c2d684bf0882b92aae81abf709d9f8d471256d
contracts/dex/interfaces/ICarbonRouter01.sol	7f8282dba90e0feadfdcfd05d582d03733b5afa2
contracts/dex/interfaces/ICarbonCallee.sol	2f3555b9eba65b97e37cef90bc71e726f555778a
contracts/dex/interfaces/IERC20.sol	6b80210e4d9adfa64eaaa71209082d9fd8b13504
contracts/dex/interfaces/V1/IUniswapV1Exchange.sol	ba992548a32038fcca1cebdcfd4b11f81f726391
contracts/dex/interfaces/V1/IUniswapV1Factory.sol	41777838b683a6861b8f41d91a9b418eafd42526
contracts/dex/CarbonERC20.sol	2d1f7d866da23ceede3551491ad9eadc8c10c6f1
contracts/dex/lib/ICarbonFactory.sol	7a83eac2e2376735fce55ecf1cc426ad9da4f6a7
contracts/dex/lib/ERC20Detailed.sol	265b6046bc8814fd18298e4e9cf781454fd65fa8
contracts/dex/lib/TransferHelper.sol	c7d9065085f3837ef93edcc7057ede2bb5f091e5

contracts/dex/lib/math/SafeMath.sol	104a955c7e6f68c5fe44a163c058b78 69090d49d
contracts/dex/lib/utils/Memory.sol	c683b012185620bc2e21b56e67519aa b162469c4
contracts/dex/lib/utils/Create2.sol	e6d8b477de54cfb67f64a5d537ebff6 6bd67ecbe
contracts/dex/lib/utils/ SafeBEP20Namer.sol	382de47a02427351d97c7e37d479e3 4e5e5bf01b
contracts/dex/lib/utils/EnumerableSet.sol	92055ef21f2e73edb53b4ed44446b1a 640babcd4
contracts/dex/lib/utils/PairNamer.sol	4c0abfc2e65b5cbae92030cf0cfe242 54d596b95
contracts/dex/lib/utils/Address.sol	686d3ffdbc7e834c21fb2816fd3d6db 935bed799
contracts/dex/lib/utils/FixedPoint.sol	de9da9a1a40befc1f5d595e4565bdf1 b0e91aa42
contracts/dex/lib/utils/ AddressStringUtil.sol	844c63853652c00dd59eec96726ad c047eaf8d80
contracts/dex/lib/utils/ ReentrancyGuard.sol	6ac6c1c983529faf11c6aea60b72905a 47158c3d
contracts/dex/lib/GSN/Context.sol	9cd6389ec1e6258456c2724194fcb90 2d0bc46dc
contracts/dex/lib/access/Manageable.sol	ab8c501445cf2dad8b0999ae88a961 f94242cd09
contracts/dex/lib/access/Ownable.sol	b9789ee48641755a7937952738e5f82 3ecd84d93
contracts/dex/lib/proxy/Proxy.sol	d2d73225a6496433d4e7e68b57e8fe db23fd67fe
contracts/dex/lib/proxy/ TransparentUpgradeableProxy.sol	9965e41a9efcedccbeaf99764e50f3a b14ac65bf
contracts/dex/lib/proxy/ UpgradeableProxy.sol	93271f8a0f5bdd09275ca56085657c1 d63c631ee
contracts/dex/lib/proxy/ProxyAdmin.sol	9a280f3c7dc0f1988def024004ae0e b535651510
contracts/dex/lib/proxy/Initializable.sol	4835b2d08397f8bd91376c8cd68de2 b9c8c7243f

contracts/dex/lib/ERC20.sol	0293906ca758522a4a029e48a932bd995f05757d
contracts/dex/lib/IERC20.sol	72c15b6a16b7dc92e69ff97ccfe1958d9948e200
contracts/dex/lib/token/BEP20/IBEP20.sol	59c10db734afa4e875505ab81e0b62f3bb645a29
contracts/dex/lib/token/BEP20/BEP20.sol	a4818b987d49da3b0dfc69321885ba9c33f2f5f8
contracts/dex/lib/token/BEP20/SafeBEP20.sol	9018680775f4115c41cf523f8a225450d3ab9157
contracts/dex/CarbonRouter.sol	beafd69b153fa2da2d31549898dea4936930e016
contracts/dex/CarbonFactory.sol	a154f78fda0e4f8b1fbf2589c036be039d6147ea
contracts/dex/libraries/ICarbonPair.sol	53713eb40dd9afc14b9d39855930e5d3995340ca
contracts/dex/libraries/CarbonLibrary.sol	7fbf7843d46aa01fcabe802adb41757fd554bacd
contracts/dex/libraries/UQ112x112.sol	5c0f96357914f9f80b6d616b79ece099d5f91ec4
contracts/dex/libraries/SafeMath.sol	f32b3f13eb0959de2a73dbd737df1073595e6531
contracts/dex/libraries/mathUpdated/SafeMath.sol	91cc035ffff449d7c25fb15ca709c579768a3019
contracts/dex/CarbonPair.sol	b721744edab1c82c12ce915ad2ae305384db12d9
contracts/governance/IncentivisedVotingLockup.sol	8ca2e63ab57b26360ee7f3b8e98f4e5bb77da28a
contracts/TestCarbonStarterToken.sol	6c28e19fd48ba719ecaa8d88c39bbcab7ae4bc2a
contracts/CarbonStarterToken.sol	8119de7fb9d77f4f3c0737eb397c37d9e9c6daaa
contracts/yield-farm/rewarders/MultipleRewards.sol	ced10b792bb4262327edd1d73188cf45f4f83525
contracts/yield-farm/rewarders/IMultipleRewards.sol	4309797bfbaa5c277553e4726496865033aa0fc3

contracts/yield-farm/ICarbonChef.sol	8f40f078411c1018751676efc2d9c1e13853ea24
contracts/yield-farm/libraries/IBoringERC20.sol	77fca667e47e4023e9bc40482286fb731ce1352e
contracts/yield-farm/libraries/BoringERC20.sol	36c34d54767e185dec1358539c5ef24a9fe40767
contracts/yield-farm/CarbonChef.sol	eceb71c441dc2016e68e42c2f93e629e77965761
contracts/launchpad/interfaces/IWETH.sol	7f3183f4a0d743811e5ed22941c91e76778c7cc2
contracts/launchpad/LaunchpadVesting.sol	4d671f1f386013c78296ecb5cbb255670c52392c
contracts/launchpad/ArbsCappedIDO.sol	f02a1916a927a2e9c8a555a7b1ef6488d4e246fa
contracts/launchpad/ArbsCappedIDOVested.sol	96392e2160fb2a554a4f65c931b0050754dbfb99
contracts/launchpad/ArbsOverflowVesting.sol	5ad74ce11a91784582bb6be308bb368bbf5c8889
contracts/shared/IERC20WithCheckpointing.sol	970a54c531a9a56cee05a8026784c408a6e30cbc
contracts/shared/Root.sol	9aa1cd3597d4cd6ef59c709009fbfa2b0f879320
contracts/shared/StableMath.sol	361d3d0450380ee0b683c02e4439917e58dd62cc
contracts/shared/IBasicToken.sol	c4436fb31af77e053c1dfcea64cb4cf2d5af718e
contracts/rewards/RewardsDistributionRecipient.sol	5006f34872059a34ad0c0b455faa6f6a319720c3
contracts/rewards/RewardsDistributor.sol	a7a6053c55286c3e743bbdb572313f1b92953b4f
contracts/XCarbonStarterToken.sol	e82035f803f2f00bfe298a4d4ad703b50025ede5

Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) indicate a changed state or potential vulnerability that was not the subject of this scan.

Imported packages

Used code from other Frameworks/Smart Contracts (direct imports).

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	12
@openzeppelin/contracts/security/ReentrancyGuard.sol	7
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	5
@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol	2
@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol	2
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	8
@openzeppelin/contracts/utils/Address.sol	2
@openzeppelin/contracts/utils/math/Math.sol	1
@openzeppelin/contracts/utils/math/SafeCast.sol	1
@openzeppelin/contracts/utils/structs/EnumerableSet.sol	1

Note for Investors: We only audited contracts mentioned in the scope above. All contracts related to the project apart from that are not a part of the audit, and we cannot comment on its security and are not responsible for it in any way

Audit Information

Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit vulnerability and the impact of that event on the organization or system. The risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to check the repository for security-related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pen-testers and smart contract developers reviewed the code line by line and documented any issues discovered.

We check every file manually. We use automated tools only so that they help us achieve faster and better results.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    a. Reviewing the specifications, sources, and instructions provided to
    SolidProof to ensure we understand the size, scope, and functionality of the
    smart contract.
    b. Manual review of the code, i.e., reading the source code line by line to identify potential vulnerabilities.
    c. Comparison to the specification, i.e., verifying that the code does what is described in the specifications, sources, and instructions provided to SolidProof.

2. Testing and automated analysis that includes the following:
    a. Test coverage analysis determines whether test cases cover code and how much code is executed when those test cases are executed.
    b. Symbolic execution, which is analysing a program to determine what inputs cause each part of a program to execute.

3. Review best practices, i.e., review smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on best practices, recommendations, and research from industry and academia.

4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.

Overall Security

Upgradeability

Contract is not an upgradeable	✅ Deployer cannot update the contract with new functionalities
Description	The contract is not an upgradeable contract. The deployer is not able to change or add any functionalities to the contract after deploying.
Comment	N/A

Ownership

The ownership is not renounced	❌ **The owner is not renounce**
Description	The owner has not renounced the ownership that means that the owner retains control over the contract's operations, including the ability to execute functions that may impact the contract's users or stakeholders. This can lead to several potential issues, including: • Centralizations • The owner has significant control over contract's operations
Comment	N/A

Note - If the contract is not deployed, consider the ownership not renounced. Moreover, ownership is automatically considered renounced if there are no ownership functionalities.

Alleviation - We will use a multisig and/or combined with a timelock contract to secure this.

Ownership Privileges

These functions can be dangerous. Please note that abuse can lead to financial loss. We have a guide where you can learn more about these Functions.

Minting tokens

Minting tokens refers to the process of creating new tokens in a cryptocurrency or blockchain network. This process is typically performed by the project's owner or designated authority, who has the ability to add new tokens to the network's total supply.

Contract owner can mint new tokens	❌ **The owner able to mint new tokens**
Description	Owners who have the ability to mint new tokens can reward themselves or other stakeholders, who can then sell the newly minted tokens on a cryptocurrency exchange to raise funds. However, there is a risk that the owner may abuse this power, leading to a decrease in trust and credibility in the project or platform. If stakeholders perceive that the owner is using their power to mint new tokens unfairly or without transparency, it can result in decreased demand for the token and a reduction in its value.
Example	If investors drive up the token price, the owner may choose to mint new tokens and sell them on a cryptocurrency exchange to raise funds. It is essential for investors to carefully research the project and its developers and exercise caution before investing in any cryptocurrency or DeFi project.
Comment	The owner can set minter addresses and those addresses/wallets can mint tokens till the maximum supply is reached.

Alleviation - We will use a multisig and/or combined with a timelock contract to secure this.

Burning tokens

Burning tokens is the process of permanently destroying a certain number of tokens, reducing the total supply of a cryptocurrency or token. This is usually done to increase the value of the remaining tokens, as the reduced supply can create scarcity and potentially drive up demand.

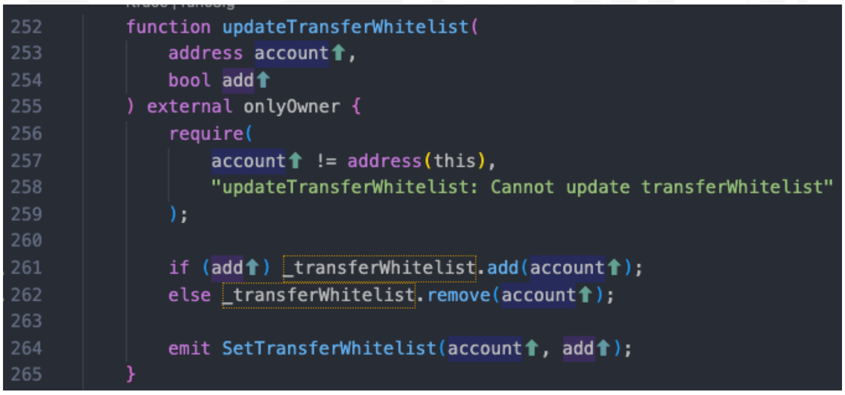
Contract owner cannot burn tokens	✅ **The owner cannot burn tokens**
Description	The owner is not able burn tokens without any allowances.
Comment	N/A

Blacklist addresses

Blacklisting addresses in smart contracts is the process of adding a certain address to a blacklist, effectively preventing them from accessing or participating in certain functionalities or transactions within the contract. This can be useful in preventing fraudulent or malicious activities, such as hacking attempts or money laundering.

Contract owner can blacklist addresses	❌ **The owner able to blacklist addresses**
Description	If the owner or developers of the smart contract abuse their power to blacklist addresses without proper justification or transparency, it can lead to a decrease in trust and credibility in the project. For example, suppose an owner or developer blacklists an address without proper explanation or communication to stakeholders. In that case, it can create speculation and uncertainty among investors, potentially causing them to sell their tokens and decreasing the token's value. Furthermore, if the owner or developers have a significant number of tokens themselves and use their power to blacklist competitors or manipulate the market, it can lead to an unfair advantage and concentration of power, potentially harming the interests of other stakeholders.
Comment	The wallets can be blacklisted from transferring their “XCarbonStarter” tokens if the owner removes them from the whitelist.

File: XCarbonStarter.sol
Codebase:



```
function updateTransferWhitelist(
    address account,
    bool add
) external onlyOwner {
    require(
        account != address(this),
        "updateTransferWhitelist: Cannot update transferWhitelist"
    );

    if (add) _transferWhitelist.add(account);
    else _transferWhitelist.remove(account);

    emit SetTransferWhitelist(account, add);
}
```

Alleviation - This is intentional, as the X token contract is not meant to be transferable by the public.

Fees and Tax

In some smart contracts, the owner or creator can set fees for certain actions or operations within the contract. These fees can be used to cover the contract's cost, such as paying for gas fees or compensating the contract's owner for their time and effort in developing and maintaining the contract.

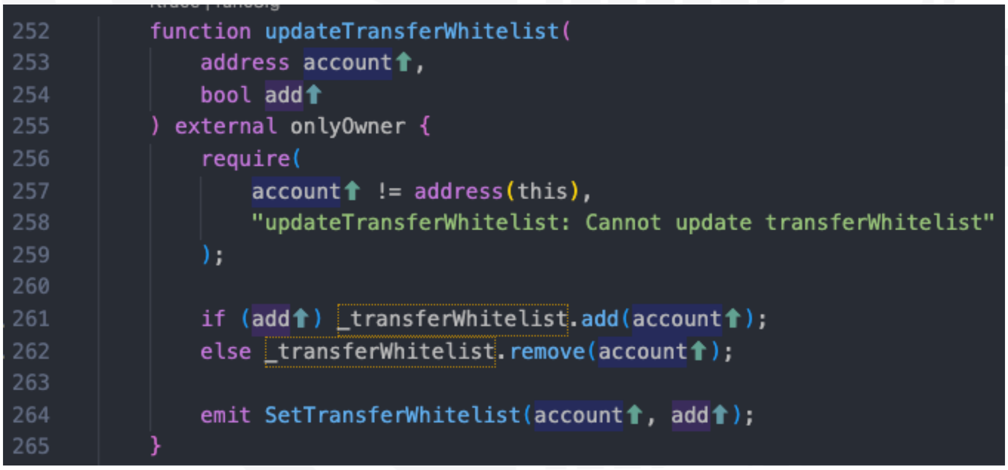
Contract owner cannot set fees more than 25%	✅ **The owner cannot levy unfair taxes**
Description	The owner is not able to set the fees above 25%
Comment	N/A

Lock User Funds

In a smart contract, locking refers to restricting access to certain tokens or assets for a specified period. When tokens or assets are locked in a smart contract, they cannot be transferred or used until the lock-up period has expired or certain conditions have been met.

Contract owner can lock the user funds	❌ **The owner is able to lock the tokens**
Description	Locking the contract means that the owner is able to lock any funds of addresses that they are not able to transfer bought tokens anymore.
Example	An example of locking is by pausing the contract or blacklisting any addresses. That causes that the blacklisted address is not able to transfer (buy/sell) anymore.
Comment	N/A

File: XCarbonStarter.sol
Codebase:



```
function updateTransferWhitelist(
    address account,
    bool add
) external onlyOwner {
    require(
        account != address(this),
        "updateTransferWhitelist: Cannot update transferWhitelist"
    );

    if (add) _transferWhitelist.add(account);
    else _transferWhitelist.remove(account);

    emit SetTransferWhitelist(account, add);
}
```

Alleviation - This is intentional, as the X token contract is not meant to be transferable by the public.

External/Public functions

External/public functions are functions that can be called from outside of a contract, i.e., they can be accessed by other contracts or external accounts on the blockchain. These functions are specified using the function declaration's external or public visibility modifier.

State variables

State variables are variables that are stored on the blockchain as part of the contract's state. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be defined with a visibility modifier, such as public, private, or internal, which determines the access level of the variable.

Components

📝Contracts	📚Libraries	🎨Abstract
33	19	5

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐Public	💰Payable
413	22

External	Internal	Private	Pure	View
278	470	23	97	181

StateVariables

Total	🌐Public
206	151

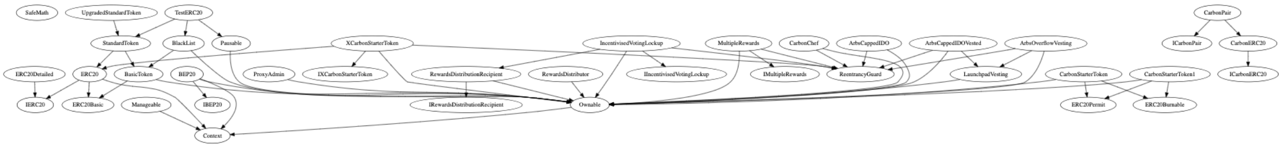
Capabilities

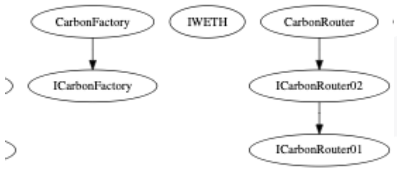
Solidity Versions observed	DelegateCall	Can Receive Funds	Uses Assembly	Transfers ETH

`^0.4.17`
`>=0.5.0`
`>=0.6.2`
`=0.5.16`
`>=0.5.0`
`<=0.6.12`
`>=0.6.0`
`^0.5.0`
`=0.6.6`
`0.8.6`
`^0.8.0`
`0.8.18`
`^0.8.9`
`=0.8.18`
	Yes	Yes	Yes	Yes

Inheritance Graph

An inheritance graph is a graphical representation of the inheritance hierarchy among contracts. In object-oriented programming, inheritance is a mechanism that allows one class (or contract, in the case of Solidity) to inherit properties and methods from another class. It shows the relationships between different contracts and how they are related to each other through inheritance.





Centralization Privileges

Centralization can arise when one or more parties have privileged access or control over the contract's functionality, data, or decision-making. This can occur, for example, if a single entity controls the contract or if certain participants have special permissions or abilities that others do not.

In the project, some authorities have access to the following functions:

File	Privileges
IncentivisedVotingLockup	• onlyOwner ◦ Manually End the contract's functionality to stake which will result in unlocking all the stakes. However, withdraw and claim will still work. ◦ Set/Update the X Token reward rate upto 100% at any time
ArbsCappedIDO	• onlyOwner ◦ The owner can withdraw or burn the unsold tokens from the contract once the sale has ended
ArbsCappedIDOVested	• onlyOwner ◦ Extend(Delay) the time of withdraw ◦ Set the linear vesting end time but it cannot exceed the max limit of 10 years ◦ The owner can withdraw or burn the unsold tokens from the contract once the sale has ended ◦ Set cliff period (claim time) before the sale has ended ◦ The owner can manually enable the claim only once which will result in the tokens being claimable without the dependency on withdraw time. Moreover, this action cannot be undone

File	Privileges
ArbsOverflowVesting	• onlyOwner • Extend(Delay) the time of withdraw • Set the linear vesting end time but it cannot exceed the max limit of 10 years • The owner can withdraw the unsold tokens from the contract once the sale has ended. Burn functionality is absent in this contract • Set cliff period (claim time) before the sale has ended • The owner can manually enable the claim only once which will result in the tokens being claimable without the dependency on withdraw time. Moreover, this action cannot be undone
RewardDistributor	• onlyOwner • Add/Remove fund managers • onlyFundManager • Distribute Rewards
CarbonChef	• onlyOwner • Start Farming but cannot stop it • Add new LP to the pool • Set any pool's carbon token allocation point and deposit fee rate (upto 10%) • Set Fee Address • Set the Reward ratio of XCarbon token to be distributed as rewards. This ratio can go upto a 100% and if it is set to that value then all the rewards will distributed in XCarbon Tokens
XCarbonStarterToken	• onlyOwner • Update redeem settings which includes maximum and minimum values of the redeem ratios • Include/Exclude wallets/addresses from the transfer whitelist • Set Excess Ratio and Excess Address

Recommendations

To avoid potential hacking risks, it is advisable for the client to manage the private key of the privileged account with care. Additionally, we recommend enhancing the security practices of centralized privileges or roles in the protocol through a decentralized mechanism or smart-contract-based accounts, such as multi-signature wallets.

Here are some suggestions of what the client can do:

- Consider using multi-signature wallets: Multi-signature wallets require multiple parties to sign off on a transaction before it can be executed, providing an extra layer of security e.g. Gnosis Safe
- Use of a timelock at least with a latency of e.g. 48-72 hours for awareness of privileged operations
- Introduce a DAO/Governance/Voting module to increase transparency and user involvement
- Consider Renouncing the ownership so that the owner can no longer modify any state variables of the contract. Make sure to set up everything before renouncing.

Audit Results

Critical issues

No critical issues

High issues

No high issues

Medium issues

#1 | Wrong Implementation of the defined NatSpec

File	Severity	Location	Status
ArbsCappedIDO	Medium	L297, 316	ACK
ArbsCappedIDOVested	Medium	L343, 362	ACK

Description - The NatSpec states that the owner will burn the unsold tokens if the minimum threshold of raised capital is not reached. Still, according to the contract's logic, the owner can withdraw the presale tokens after it has ended and will burn only the necessary amount of tokens they decide, not all unsold tokens.

Remediation - We recommend putting a check in the 'emergencyWithdrawFunction' that the unsold tokens will not be available for withdrawal if the minimum threshold of the presale is reached.

Alleviation - This functionality is part of the intentional behaviour of the project.

#2 | Funds Locked

File	Severity	Location	Status
XCarbonStarterToken	Medium	L253	ACK

Description - The owner can include/exclude wallets in the whitelist, and all the addresses that are removed from the whitelist won't be able to transfer their reward tokens.

Remediation - We recommend not to lock the transfer function entirely for non-whitelisted users.

Alleviation - This is intentional as the X token contract is not meant to be transferable by the public.

#3 | Owner can mint tokens

File	Severity	Location	Status
CarbonStarterToken	Medium	L39	ACK

Description - The contract owner can set minter addresses, and these addresses can mint tokens until the maximum supply of 50_000_000 is reached, which is not recommended in an uncontrolled manner as it can be used to manipulate the price of the tokens.

Remediation - Make sure to regulate minting periodically or limit it to only a few addresses.

Alleviation - We will use a multisig and/or combined with a timelock contract to secure this.

.

Low issues

#1 | Missing Events

File	Severity	Location	Status
CarbonChef	Low	L158, 742	ACK
CarbonFactory	Low	L23, 54, 59	ACK
IncentivisedVotingLockup	Low	L1051	ACK

Description - Make sure to emit events for all the critical parameter changes in the contract to ensure the transparency and trackability of all the state variable changes.

#2 | Missng “isContract” check

File	Severity	Location	Status
ArbsCappedIDO	Low	L50	ACK
ArbsCappedIDOVested	Low	L61	ACK
ArbsOverflowVesting	Low	L42	ACK

Description - The contract has no checks to verify whether EOAs or contract addresses are being set in the constructor. Since these addresses cannot be changed/updated we recommend putting a check to avoid passing wrong addresses in the constructor.

Remediation - We recommend putting a check to verify that the Tokens and Sale addresses must be a Smart Contract.

#3 | Old Compiler version

File	Severity	Location	Status
CarbonFactory	Low	L1	ACK
CarbonRouter	Low	L1	ACK
CarbonPair	Low	L1	ACK

Description - The contracts use outdated compiler versions, which are not recommended for deployment as they may be susceptible to known vulnerabilities.

Remediation - Use a newer pragma version. At least use the 0.8.18 version.

Informational issues

#1 | Missing Error Message

File	Severity	Location	Status
CarbonRouter	Informational	L29, 70, 136, 431	ACK

Description - Make sure to return revert (error) messages in the "require" and "assert" statements

#2 | Local Variable Shadowing

File	Severity	Location	Status
ArbsCappedIDO	Informational	L327	ACK
ArbsOverflowVesting	Informational	L261	ACK

Description - The 'cliffPeriod' variable shadows the component from the launchpad vesting contract. We recommend renaming the variable that shadows another component of the imported contract.

#3 | General Recommendations

File	Severity	Location	Status
All	Informational	N/A	ACK

Description - We recommend the project team either remove or implement the commented code in the smart contracts to enhance the code's readability and also remove all other duplications in the code. Moreover, we'd recommend creating local variables when the same calculations occur in the code to avoid unnecessary gas usage.

#4 | Contract doesn't import npm packages from source (like OpenZeppelin etc.)

File	Severity	Location	Status
All	Informational	N/A	ACK

Description - We recommend importing all packages from npm directly without flattening the contract. Functions could be modified or can be susceptible to vulnerabilities. We have observed some libraries used which are not sourced from reputable sources.

Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY