Cyberscope

A TAC Security Company

Audit Report

Plutus escrow

July 2025

Repository : https://github.com/PlutusDao/xPlutusToken

Commit : e95b73fbf8a7de063d2bb891873d77a755d82e5d

Audited by © cyberscope

Table of Contents

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
Critical	Highly Likely / High Impact
Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
Minor / Informative	Unlikely / Low to no Impact

Review

Repository	https://github.com/PlutusDao/xPlutusToken
Commit	e95b73fbf8a7de063d2bb891873d77a755d82e5d

Audit Updates

Initial Audit	27 Jun 2025 https://github.com/cyberscope-io/audits/blob/main/1-plutus/v1/audit.pdf
Corrected Phase 2	09 Jul 2025

Source Files

Filename	SHA256
XPlutusToken.sol	77d328e9f9a6677124d731ef96beaae5c0197a0467044f62a3e5cad138bd592d
interfaces/IXPlutusToken.sol	c32e47797c94520bcc155a2890ef1ba11192338bf01e3c2b4d18d6d63f0eb528
interfaces/IPlutusToken.sol	6ea1f61b9b1cd7369cbf37ec69d31a140c7a1e48805dea50e6e43be006131db3

Overview

`XPlutusToken` is an upgradeable ERC20 token, designed to enable token conversion, vesting, and redemption for the `xPLUTUS` token in conjunction with the `PLUTUS` token. The contract allows users to convert `PLUTUS` tokens to `xPLUTUS` tokens and vest `xPLUTUS` tokens for a defined period to redeem `PLUTUS` tokens at a variable redemption ratio based on vesting duration. Vesting details, including ownership and status (active, redeemed, or canceled), are tracked efficiently. Users can cancel vesting to recover `xPLUTUS` tokens or redeem matured vests to receive `PLUTUS` tokens, with excess tokens sent to a designated receiver address. Token transfers are restricted to authorized addresses via a whitelist, managed by an admin role that also handles pausing, unpausing, and updating redemption settings.

Findings Breakdown

Critical	0
Medium	0
Minor / Informative	9

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	1	8	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	MC	Missing Check	Unresolved
	CCR	Contract Centralization Risk	Acknowledged
	MIM	Missing Incentivization Mechanism	Acknowledged
	BC	Blacklists Addresses	Acknowledged
	CO	Code Optimization	Acknowledged
	MT	Mints Tokens	Acknowledged
	PTAI	Potential Transfer Amount Inconsistency	Acknowledged
	ST	Stops Transactions	Acknowledged
	L04	Conformance to Solidity Naming Conventions	Acknowledged

MC - Missing Check

Criticality	Minor / Informative
Location	XPlutusToken.sol#L90,132
Status	Unresolved

Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues. Specifically, `updateRedeemSettings` does not ensure that the difference between `maxRatio` and `minRatio` is large enough to avoid small fractional values in `getPlutusByVestingDuration`'s ratio calculation, which can lead to truncation and return only the `minRatio`.

```
function updateRedeemSettings(RedeemSettings memory redeemSettings_) external
onlyRole(DEFAULT_ADMIN_ROLE) {
if (redeemSettings_.minRatio > redeemSettings_.maxRatio ||
redeemSettings_.maxRatio > MAX_FIXED_RATIO) {
revert XPlutusToken_WrongRatioValues();
}
if (redeemSettings_.minDuration > redeemSettings_.maxDuration) {
revert XPlutusToken_WrongDurationValues();
}
_redeemSettings = redeemSettings_;
emit RedeemSettingsUpdated(redeemSettings_);
}
function getPlutusByVestingDuration(uint256 _xPlutusAmount, uint256 _duration)
public view returns (uint256) {
//...
uint256 ratio = redeemSettings_.minRatio + (((_duration -
redeemSettings_.minDuration) * (redeemSettings_.maxRatio -
redeemSettings_.minRatio)) / (redeemSettings_.maxDuration -
redeemSettings_.minDuration));
//...
}
```

Recommendation

The team is advised to properly check the variables according to the required specifications.

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	XPlutusToken.sol#L83,90,103,113,117,310
Status	Acknowledged

Description

The contract's functionality and behavior are heavily dependent on external parameters or configurations. While external configuration can offer flexibility, it also poses several centralization risks that warrant attention. Centralization risks arising from the dependence on external configuration include Single Point of Control, Vulnerability to Attacks, Operational Delays, Trust Dependencies, and Decentralization Erosion.

```
function updateExcessReceiver(address _excessReceiver) external
onlyRole(DEFAULT_ADMIN_ROLE) {...}
function updateRedeemSettings(RedeemSettings memory redeemSettings_) external
onlyRole(DEFAULT_ADMIN_ROLE) {...}
function updateWhitelist(address _account, bool _whitelisted) external
onlyRole(DEFAULT_ADMIN_ROLE) {...}
function updateWhitelist(address _account, bool _whitelisted) external
onlyRole(DEFAULT_ADMIN_ROLE) {...}
function pause() external onlyRole(DEFAULT_ADMIN_ROLE) {...}
function unpause() external onlyRole(DEFAULT_ADMIN_ROLE) {...}
function _authorizeUpgrade(address newImplementation) internal override
onlyRole(DEFAULT_ADMIN_ROLE) { }
```

Recommendation

To address this finding and mitigate centralization risks, it is recommended to evaluate the feasibility of migrating critical configurations and functionality into the contract's codebase itself. This approach would reduce external dependencies and enhance the contract's self-sufficiency. It is essential to carefully weigh the trade-offs between external configuration flexibility and the risks associated with centralization.

MIM - Missing Incentivization Mechanism

Criticality	Minor / Informative
Location	XPlutusToken.sol#L155
Status	Acknowledged

Description

The contract allows users to exchange Plutus tokens for xPlutus tokens at a 1:1 ratio through the `_convert` function. The minted xPlutus tokens can then be vested using the `vest` function, and later redeemed for Plutus tokens by calling the `redeem` function, which burns the xPlutus tokens. However, the amount of Plutus tokens received during redemption is always less than or equal to the amount of tokens initially deposited.

Therefore, the contract does not provide any economic incentive for users to convert their Plutus tokens into xPlutus. Economic incentives encourage participation in the protocol.

```
function convert(uint256 _amount, address _to) external nonReentrant
{
_convert(_amount, _to);
}
```

Recommendation

The team could consider introducing an economic incentive that encourages users to convert Plutus tokens into xPlutus. Providing such incentives for vesting will help increase participation and support the intended use of the mechanism.

BC - Blacklists Addresses

Criticality	Minor / Informative
Location	XPlutusToken.sol#L103
Status	Acknowledged

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `updateWhitelist` function.

```
function updateWhitelist(address _account, bool _whitelisted) external
onlyRole(DEFAULT_ADMIN_ROLE) {
if (_account == address(this)) {
revert XPlutusToken_InvalidWhitelistAddress();
}
if (_account == address(0)) revert XPlutusToken_InvalidAddress();
whitelist[_account] = _whitelisted;
emit WhitelistUpdated(_account, _whitelisted);
}
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

CO - Code Optimization

Criticality	Minor / Informative
Location	XPlutusToken.sol#L261,268,273,289
Status	Acknowledged

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations. In particular, the contract performs array operations in an inefficient approach, that hinders code readability and future maintenance.

```
function _addVestToOwnerEnumeration(address to, uint256 vestId) private {
...
}

function _addVestToAllVestsEnumeration(uint256 vestId) private {
...
}

function _removeVestFromOwnerEnumeration(address from, uint256 vestId) private
{
...
}

function _removeVestFromAllVestsEnumeration(uint256 vestId) private {
...
}
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

MT - Mints Tokens

Criticality	Minor / Informative
Location	XPlutusToken.sol#L222
Status	Acknowledged

Description

The contract mints tokens. Tokens can be minted by calling the `_convert` function. As a result, the contract tokens will be inflated.

```
function _convert(uint256 _amount, address _to) internal {
if (_amount == 0) revert XPlutusToken_AmountZero();
if (_to == address(0)) revert XPlutusToken_InvalidAddress();
plutus.safeTransferFrom(msg.sender, address(this), _amount);
_mint(_to, _amount);
emit Converted(msg.sender, _to, _amount);
}
```

Recommendation

The team should be aware that allowing tokens to be minted through the `_convert` function without any form of supply limitation introduces a significant risk of token inflation. Since this function mints tokens equivalent to the input amount, repeated use could drastically increase the total supply, potentially undermining the token's value.

PTAI - Potential Transfer Amount Inconsistency

Criticality	Minor / Informative
Location	XPlutusToken.sol#L222
Status	Acknowledged

Description

The `transfer()` and `transferFrom()` functions are used to transfer a specified amount of tokens to an address. The fee or tax is an amount that is charged to the sender of an ERC20 token when tokens are transferred to another address. According to the specification, the transferred amount could potentially be less than the expected amount. This may produce inconsistency between the expected and the actual behavior.

The following example depicts the diversion between the expected and actual amount.

Tax	Amount	Expected	Actual
No Tax	100	100	100
10% Tax	100	100	90

```
function _convert(uint256 _amount, address _to) internal {
if (_amount == 0) revert XPlutusToken_AmountZero();
if (_to == address(0)) revert XPlutusToken_InvalidAddress();
plutus.safeTransferFrom(msg.sender, address(this), _amount);
_mint(_to, _amount);
emit Converted(msg.sender, _to, _amount);
}
```

Recommendation

The team is advised to take into consideration the actual amount that has been transferred instead of the expected.

It is important to note that an ERC20 transfer tax is not a standard feature of the ERC20 specification, and it is not universally implemented by all ERC20 contracts. Therefore, the contract could produce the actual amount by calculating the difference between the transfer call.

```
Actual Transferred Amount = Balance After Transfer - Balance Before Transfer
```

ST - Stops Transactions

Criticality	Minor / Informative
Location	XPlutusToken.sol#L113,117
Status	Acknowledged

Description

The contract owner has the authority to stop transactions for all users. The owner may take advantage of it by calling the `pause` function.

```
function pause() external onlyRole(DEFAULT_ADMIN_ROLE) {
_pause();
}

function unpause() external onlyRole(DEFAULT_ADMIN_ROLE) {
_unpause();
}
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	XPlutusToken.sol#L40,47
Status	Acknowledged

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
uint256[50] private __gap
address _plutus
address _initialAuthority
```

Recommendation

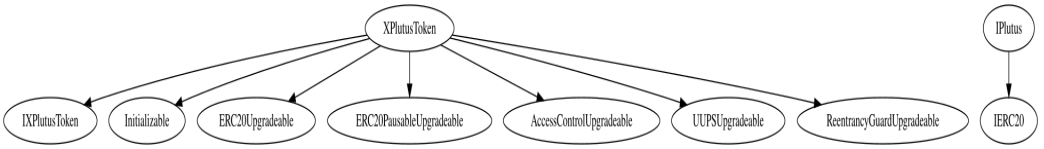
By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.
Find more information on the Solidity documentation
https://docs.soliditylang.org/en/stable/style-guide.html#naming-conventions.

Functions Analysis

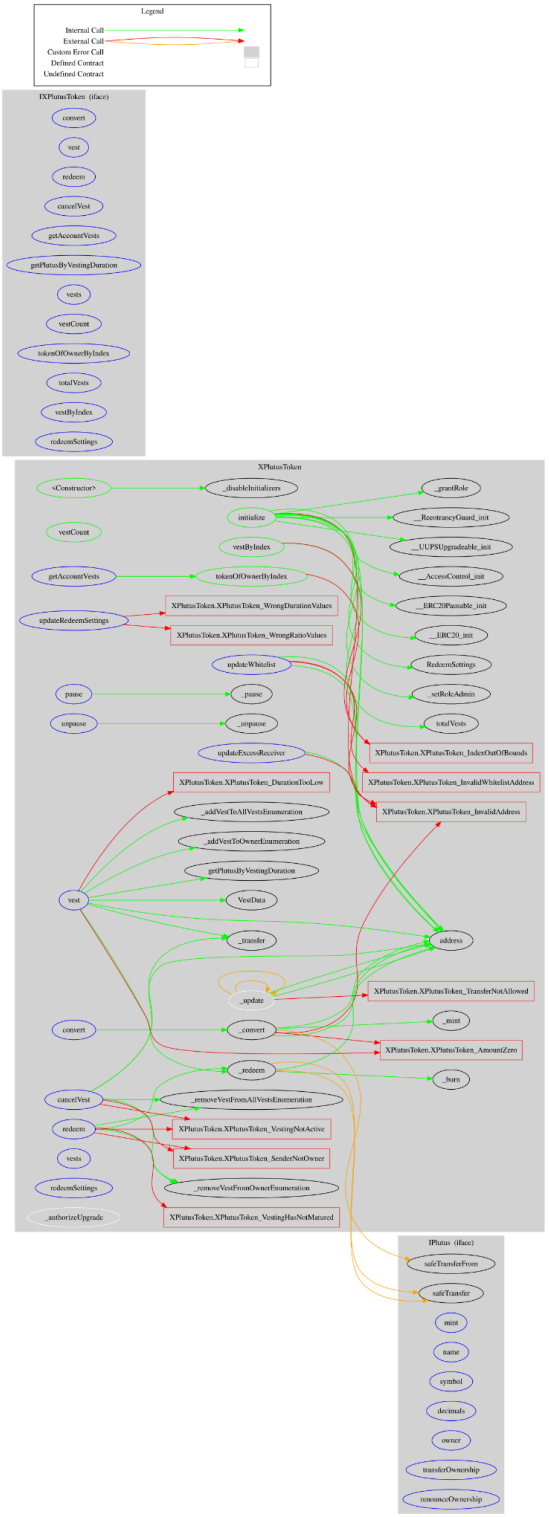
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
XPlutusToken	Implementation	IXPlutusToken, Initializable, ERC20Upgradeable, ERC20PausableUpgradeable, AccessControlUpgradeable, UUPSUpgradeable, ReentrancyGuardUpgradeable		
		Public	✓	-
	initialize	Public	✓	initializer
	vestCount	Public		-
	tokenOfOwnerByIndex	Public		-
	totalVests	Public		-
	vestByIndex	Public		-
	updateExcessReceiver	External	✓	onlyRole
	updateRedeemSettings	External	✓	onlyRole
	updateWhitelist	External	✓	onlyRole
	pause	External	✓	onlyRole
	unpause	External	✓	onlyRole

	getPlutusByVestingDuration	Public		-
	getAccountVests	External		-
	convert	External	✓	nonReentrant
	vest	External	✓	nonReentrant
	redeem	External	✓	nonReentrant
	cancelVest	External	✓	nonReentrant
	_convert	Internal	✓	
	_redeem	Internal	✓	
	_update	Internal	✓	
	_addVestToOwnerEnumeration	Private	✓	
	_addVestToAllVestsEnumeration	Private	✓	
	_removeVestFromOwnerEnumeration	Private	✓	
	_removeVestFromAllVestsEnumeration	Private	✓	
	vests	External		-
	redeemSettings	External		-
	_authorizeUpgrade	Internal	✓	onlyRole

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Custom Error Call
Defined Contract
Undefined Contract
IXPlutusToken (iface)
convert
vest
redeem
cancelVest
getAccountVests
getPlutusByVestingDuration
vests
vestCount
tokenOfOwnerByIndex
totalVests
vestByIndex
redeemSettings
XPlutusToken
<Constructor>
_disableInitializers
initialize
vestCount
vestByIndex
getAccountVests
tokenOfOwnerByIndex
updateRedeemSettings
XPlutusToken.XPlutusToken_WrongDurationValues
XPlutusToken.XPlutusToken_WrongRatioValues
updateWhitelist
pause
_pause
unpause
_unpause
updateExcessReceiver
XPlutusToken.XPlutusToken_DurationTooLow
_addVestToAllVestsEnumeration
_addVestToOwnerEnumeration
getPlutusByVestingDuration
vest
VestData
_transfer
_update
convert
_convert
_redeem
cancelVest
_removeVestFromAllVestsEnumeration
redeem
XPlutusToken.XPlutusToken_VestingNotActive
vests
XPlutusToken.XPlutusToken_SenderNotOwner
redeemSettings
_removeVestFromOwnerEnumeration
_authorizeUpgrade
XPlutusToken.XPlutusToken_VestingHasNotMatured
_grantRole
__ReentrancyGuard_init
__UUPSUpgradeable_init
__AccessControl_init
__ERC20Pausable_init
__ERC20_init
RedeemSettings
setRoleAdmin
totalVests
XPlutusToken.XPlutusToken_IndexOutOfBounds
XPlutusToken.XPlutusToken_InvalidWhitelistAddress
XPlutusToken.XPlutusToken_InvalidAddress
address
XPlutusToken.XPlutusToken_TransferNotAllowed
_mint
XPlutusToken.XPlutusToken_AmountZero
_burn
IPlutus (iface)
safeTransferFrom
safeTransfer
mint
name
symbol
decimals
owner
transferOwnership
renounceOwnership

Summary

XPlutusToken contract implements a token and vesting mechanism. This audit investigates security issues, business logic concerns and potential improvements.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment 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 Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

A **TAC Security** Company

The Cyberscope team

cyberscope.io