XCAD Contract

AUDIT REPORT

Version 1.0.0

Serial No. 20221003000012013

Presented by Fairyproof

October 3, 2022

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the XCAD contracts project.

Audit Start Time:

September 12, 2022

Audit End Time:

October 3, 2022

Audited Code's Github Repository:

https://github.com/xcademy-dev/zil-audited

Audited Code's Github Commit Number When Audit Started:

ef2f81fa5f9221b0586170eb6c62eccbbf0e30cc

Audited Code's Github Commit Number When Audit Ended:

ee0cdb3facc24121ebcf11fed9becca52115a9f6

Audited Source Files:

The calculated SHA-256 values for the audited files when the audit was done are as follows:

```
Marketplace_Auction.scilla:
0x4109479fde2b2ecd4102b3ae61cfb78069b177abdbd481f22884e99f9dd0036e
SingleAssetStaking.scilla:
0x543862bed6c2c236a0c77b83f06529846d737c8223772d67c4e78dd4bddc2b26
XcadPLAY_FungibleToken.scilla:
0xeb2c75d657db3310c3f669a08857f15164fdc4d0f14d7d82d6bbf498d58554d0
ZRC6_Xcad_Nerd_NFT.scilla:
0xd3c5d083431c614fd6ec381f7cb8c35db9de4c1c0084b776505c017866f8aa43
```

The source files audited include all the files with the extension "scilla" as follows:

```
./
├── Marketplace_Auction.scilla
├── SingleAssetStaking.scilla
├── XcadPLAY_FungibleToken.scilla
└── ZRC6_Xcad_Nerd_NFT.scilla

0 directories, 4 files
```

The goal of this audit is to review XCAD's implementation for its token issuance and NFT auction functions, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

We make observations on specific areas of the code that present concrete problems, as well as general observations that traverse the entire codebase horizontally, which could improve its quality as a whole.

This audit only applies to the specified code, software or any materials supplied by the XCAD team for specified versions. Whenever the code, software, materials, settings, environment etc is changed, the comments of this audit will no longer apply.

— Disclaimer

Note that as of the date of publishing, the contents of this report reflect the current understanding of known security patterns and state of the art regarding system security. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk.

The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. If the audited source files are smart contract files, risks or issues introduced by using data feeds from offchain sources are not extended by this review either.

Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit is recommended after the issues covered are fixed.

To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

— Methodology

The above files' code was studied in detail in order to acquire a clear impression of how the its specifications were implemented. The codebase was then subject to deep analysis and scrutiny, resulting in a series of observations. The problems and their potential solutions are discussed in this document and, whenever possible, we identify common sources for such problems and comment on them as well.

The Fairyproof auditing process follows a routine series of steps:

1. Code Review, Including:

- Project Diagnosis

Understanding the size, scope and functionality of your project's source code based on the specifications, sources, and instructions provided to Fairyproof.

- Manual Code Review

Reading your source code line-by-line to identify potential vulnerabilities.

- Specification Comparison

Determining whether your project's code successfully and efficiently accomplishes or executes its functions according to the specifications, sources, and instructions provided to Fairyproof.

2. Testing and Automated Analysis, Including:

- Test Coverage Analysis

Determining whether the test cases cover your code and how much of your code is exercised or executed when test cases are run.

- Symbolic Execution

Analyzing a program to determine the specific input that causes different parts of a program to execute its functions.

3. Best Practices Review

Reviewing the source code to improve maintainability, security, and control based on the latest established industry and academic practices, recommendations, and research.

— Structure of the document

This report contains a list of issues and comments on all the above source files. Each issue is assigned a severity level based on the potential impact of the issue and recommendations to fix it, if applicable. For ease of navigation, an index by topic and another by severity are both provided at the beginning of the report.

— Documentation

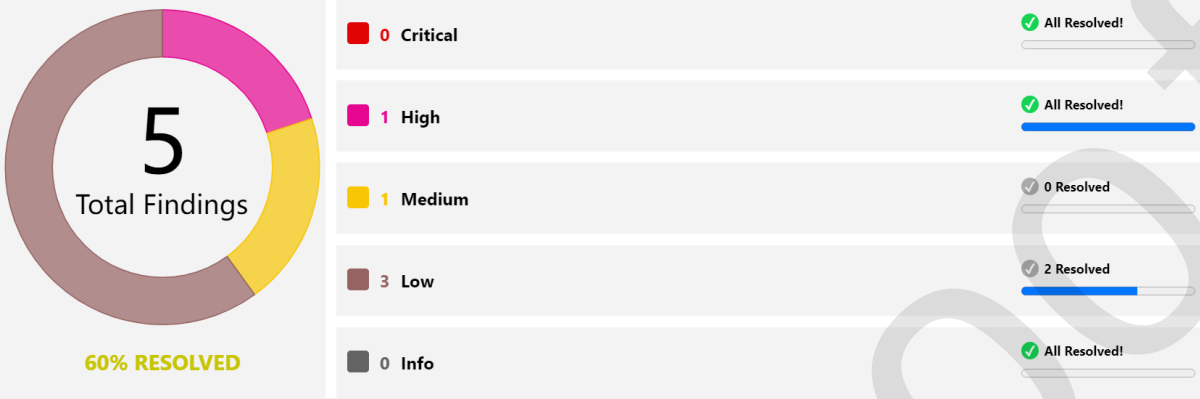
For this audit, we used the following sources of truth about how the token issuance and NFT auction functions should work:

Smart Contract Files

These were considered the specification, and when discrepancies arose with the actual code behavior, we consulted with the XCAD team or reported an issue.

— Comments from Auditor

Serial Number	Auditor	Audit Time	Result
2022100300012013	Fairyproof Team	Sep 12, 2022 - Oct 3, 2022	Medium Risk



Summary:

The Fairyproof security team used its auto analysis tools and manual work to audit the project. During the audit, one issue of high-severity, one issue of medium-severity and three issues of low-severity were uncovered. The XCAD team fixed the issue of high-severity and two issues of low-severity, and acknowledged the remaining issues.

02. About Fairyproof

Fairyproof is a leading technology firm in the blockchain industry, providing consulting and security audits for organizations. Fairyproof has developed industry security standards for designing and deploying blockchain applications.

03. Major functions of audited code

The audited code mainly implements the following functions:

1. `XcadPLAY_FungibleToken.scilla`

Issuance of ZRC-2 Fungible Token Similar to ERC-20 Tokens

- Token Name: to be determined on deployment
- Token Symbol: to be determined on deployment
- Decimals: to be determined on deployment
- Initial Supply: to be determined on deployment
- Max Supply: Uncapped
- Subsequent Minting: Yes
- Misc: No

Note:

The admin can mint tokens infinitely, therefore its access control should be managed with great care.

When the whitelist is enabled, only the addresses in the whitelist are allowed to call `IncreaseAllowance`, `DecreaseAllowance` and `TransferFrom`. When the whitelist is disabled, all addresses are allowed to call these functions.

2. `ZRC6_Xcad_Nerd_NFT.scilla`

   Issuance of ZRC-6 Non-fungible Token Similar to ERC-721 Tokens

- Token Name: to be determined on deployment
- Token Symbol: : to be determined on deployment
- Royalty_fee Charged: Yes
- Token Transfer Pausable: Yes

Note:

The NFT can be issued in three options: it can be issued by the XCAD team, it should be purchased by users with both types tokens or it can only be purchased by specific whitelisted users at a discount.

The whitelist function is enabled or disabled by the `whitelist_check_enabled` variable.

The whitelist feature does not apply to token owners. Token Owners can always calls `TransferFrom` and there is no restriction. Spenders are subjected to the whitelist function.

3. `Marketplace_Auction.scilla`

   NFT Auction Contract

   A seller can list their NFTs for auction with a specific token, set the auction's ending date or cancel the auction. After the auction ends, the seller will be able to get the auction revenue excluding the service fee and royalty fee and the bidder will get the NFT.

   Note: the implementation has a pause function and an emergency withdrawal function. When the system is paused, the XCAD team is able to withdraw all the assets (ZRC-2 tokens and NFTs). Users should be aware of this.

4. `SingleAssetStaking.scilla`

   A Single Token Staking Contract

   Users can stake single tokens to get rewards in another token.

   Note:

   The staking mechanism has a locking period. If users withdraw their staked assets before the locking period ends they will be charged with some fees as penalty.

   The implementation has a pause function and a emergency withdrawal function. When the system is paused, the XCAD team is able to withdraw all the assets (ZRC-2 tokens). Users should be aware of this.

04. Coverage of issues

The issues that the Fairyproof team covered when conducting the audit include but are not limited to the following ones:

- Replay Attack
- Reordering Attack
- Miner's Advantage
- Rollback Attack
- DoS Attack
- Injection Attack
- Transaction Ordering Attack
- Race Condition
- Access Control
- Timestamp Attack
- Gas Consumption
- Inappropriate Callback Function
- Function Visibility
- Implementation Vulnerability
- Uninitialized Storage Pointer
- Arithmetic Precision
- Fake Deposit
- Shadow Variable
- Design Vulnerability
- Token Issuance
- Admin Rights
- Asset Security
- Contract Upgrade/Migration
- Code Improvement
- Misc

05. Severity level reference

Every issue in this report was assigned a severity level from the following:

Critical severity issues need to be fixed as soon as possible.

High severity issues will probably bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Informational is not an issue or risk but a suggestion for code improvement.

06. Major areas that need attention

Based on the provided source code the Fairyproof team focused on the possible issues and risks related to the following functions or areas.

- Function Implementation

We checked whether or not the functions were correctly implemented.

We found some issues, for more details please refer to FP-1 and FP-5 in "08. Issue description".

- Access Control

We checked each of the functions that could modify a state, especially those functions that could only be accessed by "owner".

We didn't find issues or risks in these functions or areas at the time of writing.

- Token Issuance & Exchange

We checked whether or not the contract files could mint tokens at will.

We found some issues, for more details please refer to FP-3 and FP-4 in "08. Issue description".

- State Update

We checked some key state variables which should only be set at initialization.

We didn't find issues or risks in these functions or areas at the time of writing.

- Asset Security

We checked whether or not all the functions that transfer assets were safely handled.

We found one issue, for more details please refer to FP-2 in "08. Issue description".

- Contract Migration/Upgrade

We checked whether or not the contract files would introduce issues or risks associated with contract migration/upgrade.

We didn't find issues or risks in these functions or areas at the time of writing.

- Miscellaneous

We didn't find issues or risks in other functions or areas at the time of writing.

07. List of issues by severity

Index	Title	Issue/Risk	Severity	Status
FP-1	Deny of Service Attack	DoS Attack	High	✓ Fixed
FP-2	Admin Could Withdraw All Assets	Asset Security	Medium	Acknowledged
FP-3	Uncapped Supply	Token Issuance	Low	Acknowledged
FP-4	Normal Functions Restricted	Design Vulnerability	Low	✓ Fixed
FP-5	Contradictory Logic	Design Vulnerability	Low	✓ Fixed

08. Issue descriptions

[FP-1] Deny of Service Attack High ✓ Fixed

Issue/Risk: DoS Attack

Description:

In `Marketplace_Auction.scilla`, if a bidder won the the item and the previous bidder's assets (ZRC-2 Token) would be returned. In Zilliqa, when a ZRC-2 Token executes a `transfer` (transition), a callback function will be called. In this implementation a `rouge` user could deploy a malicious smart contract to deny any ZRC-2 tokens sent from `Market`. This could turn out to be a DoS attack. This resulted in that the previous losing bidder's bid would still be effective, new bidders couldn't bid and the `rouge` user would eventually win the bid with a low price.

Recommendation:

Consider implementing a token withdrawal function.

Update:

DONE.

Status:

DONE

[FP-2] Admin Could Withdraw All Assets Medium Acknowledged

Issue/Risk: Asset Security

Description:

There are emergency withdrawal functions in both `Marketplace_Auction.scilla` and `SingleAssetStaking.scilla`. The admin can call these functions to withdraw all assets. In a case in which the admin's access control is compromised, all assets can be stolen. Users should be aware of this issue. It is strongly recommended to manage this access control with great care.

Recommendation:

Consider changing the EmergencyWithdraw function such that `owner` cannot withdraw assets, but only the users can manually withdraw assets and give up their rewards. When the users perform emergency withdrawals the application will check if they will be punished and their staked assets will be fully refunded due to actual locking time.

If for some reason the function needs to be kept, consider transferring `owner`'s access control to a multi-sig wallet or DAO.

Update:

For security reasons the XCAD will keep the emergency withdraw function.

Status:

The XCAD team has acknowledged this issue.

[FP-3] Uncapped Supply Low Acknowledged

Issue/Risk: Token Issuance

Description:

In `XcadPLAY_FungibleToken.scilla`, `Mint` doesn't define a cap for token's max supply. In a case in which the admin's access control is compromised, this function can be called to mint tokens infinitely thus possibly hurting token holders' interest. This access control should be managed with great care.

Recommendation:

Consider adding a cap

Update:

The max supply hasn't been determined yet. The access control will be managed with great care.

Status:

The XCAD team has acknowledged this issue.

[FP-4] Normal Functions Restricted Low ✓ Fixed

Issue/Risk: Design Vulnerability

Description:

In `ZRC6_Xcad_Nerd_NFT.scilla`, the caller of `TransferFrom` or `SetSpender`(transitions) may be restricted by the whitelist. This may affect contracts' or users' normal transfer operations.

In `XcadPLAY_FungibleToken.scilla`, the caller of `IncreaseAllowance`, `DecreaseAllowance` or `TransferFrom` may be restricted by the whitelist. This may affect contracts' or users' these operations.

This may need to be notified to users

Recommendation:

Consider managing the access control and calling these functions with great care.

Update:

The XCAD team removed some restrictions. Normal NFT transfers will not be affected.

Status:

The XCAD team has fixed this issue.

[FP-5] Contradictory Logic Low ✓ Fixed

Issue/Risk: Design Vulnerability

Description:

In `ZRC6_Xcad_Nerd_NFT.scilla` , `TransferFrom` is restricted by `RequireMarketplaceIsActive _sender;`, however `BatchTransferFrom` is not. These two implementations are contradictory.

Recommendation:

Consider removing `RequireMarketplaceIsActive _sender;` in `TransferFrom` or adding this restriction in `BatchTransferFrom`.

Update:

Consider changing the code such that the implemented logic in `BatchTransferFrom` is the same as the logic in `TransferFrom`

Status:

The XCAD team has fixed this.

09. Recommendations to enhance the overall security

We list some recommendations in this section. They are not mandatory but will enhance the overall security of the system if they are adopted.

- Consider updating `minter`'s access control after updating `owner` in ' `XcadPLAY_FungibleToken.scilla` and `ZRC6_Xcad_Nerd_NFT.scilla` .

Status: Fixed

- Consider checking whether `default_nft_type` falls between `min_nft_type` and `max_nft_type` after `SetNFTTypeRange` is done in `ZRC6_Xcad_Nerd_NFT.scilla`. If `default_nft_type` doesn't, `SetDefaultNFTType` should be called to reset `default_nft_type`.

  Status: Fixed

- Consider requiring `new_treasury_address` to be a non-zero address when calling `SetTreasuryAddress` in `ZRC6_Xcad_Nerd_NFT.scilla` .

  Status: Fixed

- Consider transferring all `owner`'s access control to multi-sig wallets.
- Consider avoiding the reward token to be the same as the staking token. If for some reason the reward token has to be the same as the staking token, the XCAD team should keep sufficient reward tokens in the vault to maintain its solvency

Appendices

Transition Check

File:Marketplace_Auction.scilla

Index	Transition	Permission Check	Notes
1	Start	No Need	RequireNotPaused
2	BatchStart	No Need	RequireNotPaused
3	Bid	No Need	RequireNotPaused,RequireBidderNotSmartContract
4	Cancel	RequireAccessToCancel	RequireNotExpired
5	End	RequireSellerOrContractOwner Or RequireSellerOrBuyerOrContractOwner	RequireNotPaused
6	WithdrawPaymentTokens	No Need	RequireNotPaused
7	WithdrawAsset	No Need	RequireNotPaused
8	WithdrawPaymentTokensEmergency	RequireContractOwner	RequirePaused,Risk
9	WithdrawAssetEmergency	RequireContractOwner	RequirePaused,Risk
10	Pause	RequireContractOwner	RequireNotPaused
11	Unpause	RequireContractOwner	RequirePaused
12	SetServiceFeeBPS	RequireContractOwner	
13	SetBidIncrementBPS	RequireContractOwner	
14	SetServiceFeeRecipient	RequireContractOwner	
15	AllowPaymentTokenAddress	RequireContractOwner	
16	DisallowPaymentTokenAddress	RequireContractOwner	
17	ClearAllowList	RequireContractOwner	
18	SetAllowlist	RequireContractOwner	
19	SetContractOwnershipRecipient	RequireContractOwner	
20	AcceptContractOwnership	is_recipient	
21	ZRC6_RecipientAcceptTransferFrom		Empty Callback
22	ZRC6_TransferFromCallback		Empty Callback
23	RecipientAcceptTransferFrom		Empty Callback
24	TransferSuccessCallBack		Empty Callback
25	TransferFromSuccessCallBack		Empty Callback

File:SingleAssetStaking.scilla

Index	Transition	Permission Check	Notes
1	UpdateOwner	RequireOwner	
2	ClaimOwner	staging_owner	
3	Pause	RequireOwner	
4	Unpause	RequireOwner	
5	UpdateStartBlock	RequireOwner	
6	UpdateEndBlock	RequireOwner	
7	UpdateTokenRewards	RequireOwner	
8	RemoveTokenRewards	RequireOwner	
9	UpdatePenaltyRate	RequireOwner	
10	UpdateLockupCycle	RequireOwner	
11	Deposit		RequireNotPaused
12	Claim		RequireNotPaused,RequireValidUser
13	CheckRewards		RequireNotPaused
14	Withdraw		RequireNotPaused,RequireValidUser
15	WithdrawByLoss		RequireNotPaused,RequireValidUser
16	WithdrawPenalty	RequireOwner	
17	EmergencyWithdraw	RequireOwner	RequirePaused
18	TransferSuccessCallBack		Callback
19	RecipientAcceptTransfer		Callback
20	TransferFromSuccessCallBack		Callback
21	RecipientAcceptTransferFrom		Callback

File:XcadPLAY_FungibleToken.scilla

Index	Transition	Permission Check	Notes
1	Mint	IsMinter	Need Cap
2	Burn		Burn Self
3	TransferOwnership	ThrowUnlessSenderIsOwner	
4	AcceptPendingOwnership	pending_owner	
5	IncreaseAllowance	ThrowIfDexIsInActive	IsNotSender
6	DecreaseAllowance	ThrowIfDexIsInActive	IsNotSender
7	Transfer		
8	BatchTransfer		
9	TransferFrom		ThrowIfDexIsInActive
10	EnableDexCheck	ThrowUnlessSenderIsOwner	
11	DisableDexCheck	ThrowUnlessSenderIsOwner	
12	AddDex	ThrowUnlessSenderIsOwner	
13	DisableDex	ThrowUnlessSenderIsOwner	
14	RemoveDex	ThrowUnlessSenderIsOwner	
15	SetMinter	ThrowUnlessSenderIsOwner	
16	RemoveMinter	ThrowUnlessSenderIsOwner	
17	SetSellFeeBPS	ThrowUnlessSenderIsOwner	
18	SetSellFeeRecipient	ThrowUnlessSenderIsOwner	

File:ZRC6_Xcad_Nerd_NFT.scilla

Index	Transition	Permission Check	Notes
1	Pause	RequireContractOwner	
2	Unpause	RequireContractOwner	
3	SetRoyaltyRecipient	RequireContractOwner	
4	SetRoyaltyFeeBPS	RequireContractOwner	
5	SetBaseURI	RequireContractOwner	
6	Mint	IsMinter	RequireNotPaused
7	PublicMint		RequireNotPaused
8	DiscountMint	RequireMinterInTierWhitelist	RequireNotPaused
9	BatchMint	IsMinter	RequireNotPaused
10	Burn	RequireOwnerOrOperator	RequireNotPaused
11	BatchBurn	RequireOwnerOrOperator	RequireNotPaused
12	AddMinter	RequireContractOwner	
13	RemoveMinter	RequireContractOwner	
14	SetSpender		RequireMarketplaceIsActive
15	AddOperator		RequireNotSelf
16	RemoveOperator		has_operator
17	TransferFrom	RequireMarketplaceIsActive	RequireNotPaused
18	BatchTransferFrom	RequireMarketplaceIsActive	RequireNotPaused
19	SetContractOwnershipRecipient	RequireContractOwner	
20	AcceptContractOwnership		is_recipient
21	EnableWhitelistCheck	RequireContractOwner	
22	DisableWhitelistCheck	RequireContractOwner	
23	AddMarketplace	RequireContractOwner	
24	DisableMarketplace	RequireContractOwner	
25	RemoveMarketplace	RequireContractOwner	
26	SetTreasuryAddress	RequireContractOwner	
27	SetDefaultNFTType	RequireContractOwner	
28	SetNFTTypeRange	RequireContractOwner	
29	SetMintPrice	RequireContractOwner	
30	SetPaymentToken1Address	RequireContractOwner	
31	SetPaymentToken2Address	RequireContractOwner	
32	SetDiscountBPS	RequireContractOwner	
33	AddMinterByTier	RequireContractOwner	
34	RemoveMinterByTier	RequireContractOwner	
35	TransferFromSuccessCallBack		Callback

FAIRYPROOF

https://medium.com/@FairyproofT

https://twitter.com/FairyproofT

https://www.linkedin.com/company/fairyproof-tech

https://t.me/Fairyproof_tech

Reddit: https://www.reddit.com/user/FairyproofTech