safepress

Smart Contract Audit

Solana Endpoint of the Multi-Chain NFT Bridge

CUSTOMER: XP.NETWORK

DATE: Oct 18th, 2022

Code review and security report

IMPORTANT: This document likely contains critical information about the Client's software and hardware systems, security susceptibilities, descriptions of possible exploits and attack vectors. The document shall remain undisclosed until any significant vulnerabilities are remedied.

CLIENT:	XP.NETWORK	START DATE:	Oct 3th, 2022
TYPE, SUBTYPE:	NFT Bridge	END DATE:	Oct 18th, 2022

Scope

REPOSITORY:	https://github.com/XP-NETWORK/solana-bridge
DOCUMENTATION:	No documentation
TESTS:	Passing
AUDITORS:	Zain Franci, Brandon Botosh
REVIEW $ APPROVAL:	Ryan Rhiel Madsen
SMART CONTRACT AUDITED;	programs / xp_bridge / src / lib.rs

Commit hashes:

BASE:	D4DB409B8E29E080AB349031C7481478E0993AF6 (INCL.)
UPDATE 1:	c1e1a9b83f9b039cdae068c4968c35043fc459d8
UPDATE 2:	a502e8b86617f5669f3cfee0cce61699fd2bd0db

Definitions of vulnerability classification

CRITICAL

Bug / Logic failures in the code that cause loss of assets / data manipulation.

HIGH

Difficult to exploit problems which could result in elevated privileges, data loss etc.

MEDIUM

Bug / Logic failures in the code which need to be fixed but cannot lead to loss of assets / data manipulation.

LOW

Mostly related to unused code, style guide violations, code snippets with low effect etc.

Findings

0	0	7	1	0
Critical	High	Medium	Low	Informational

Summary

XPSOL-01	Improper implementation for enforcing uniqueness	Medium	Fixed
XPSOL-02	Possible underflow error	Medium	Fixed
XPSOL-03	Unimplemented TODOs in code	Medium	Fixed
XPSOL-04	Calculate bump value off-chain to reduce cost	Medium	Fixed
XPSOL-05	Move looping done over data.creators off-chain.	Medium	Fixed
XPSOL-06	Hard to read and error prone calculations	Medium	Fixed
XPSOL-07	Code duplication	Medium	Fixed
XPSOL-08	Unused variables	Low	Fixed

Finding: XPSOL-01

Improper implementation for enforcing uniqueness.

Base	Function	`create_action`	Lines 41 - 44	Medium	Fixed

Description

There is no need for the function `create_action` to ensure uniqueness of the `action_id` .

PDA's created with `init` , seeds and `bump` enforce uniqueness. [1]

Recommendation

The implementation using `create_action` can be changed. The function should be removed since Solana - Anchor will automatically ensure that the `action_id` is unique. This is because `action_id` is being used as seed value to calculate the PDA and since PDA's are deterministically calculated, uniqueness can automatically be enforced.

Finding: XPSOL-02

Improper implementation for enforcing uniqueness.

Update 1	Function	`validate_withdraw_fees`	Line 93	Medium	Fixed

Description

There is a possible underflow error in `validate_withdraw_fees` when calculating total amount to withdraw in the expression:

```
let amount = bridge_balance - min_balance;
```

which can arise if `bridge_balance` is less that the `minimum_balance`

Recommendation

Add a check above the calculator to ensure that `bridge_balance` is always greater than `min_balance` .

Finding: XPSOL-03

Unimplemented TODOs in code

Base	Function	`validate_withdraw_fees`	Line 116	Medium	Fixed
Base	Function	`withdraw_nft`	Line 266	Medium	Fixed
Base	Function	`freeze_nft`	Line 323	Medium	Fixed

Description

There are unimplemented TODOs in the code for checks that should be implemented, such as:

- “TODO: set correct amount” in the function `validate_withdraw_fees` . The variable `amount` is hardcoded to 1000.
- “TODO: check if lamports are greater than zero” in functions `withdraw_nft` and `freeze_nft` .

Recommendation

The variable `amount` in `validate_withdraw_fees` samountnamic by passing it to the system program. Checks should also be added to ensure that the correct value is always sent.

Checks should be added to `withdraw_nft` and `wfreeze_nft` to ensure that 0 lamports are never passed into the functions.

Finding: XPSOL-04

Calculate bump values off-chain to reduce cost

Base	In Account Context Structs:	**ValidatePause**	Line 443	Medium	Fixed
Base	In Account Context Structs:	**ValidateUnpause**	Line 470	Medium	Fixed
Base	In Account Context Structs:	**ValidateUpdateGroupkey**	Line 497	Medium	Fixed
Base	In Account Context Structs:	**ValidateTransferNft**	Line 525, 530	Medium	Fixed
Base	In Account Context Structs:	**WithdrawFees**	Line 684	Medium	Fixed
Base	In Account Context Structs:	**WithdrawNft**	Line 800	Medium	Fixed
Base	In Account Context Structs:	**FreezeNft**	Line 825	Medium	Fixed
Base	In Account Context Structs:	**ValidateUnfreezeNft**	Line 839	Medium	Fixed

Description

PDAs are calculated by hashing seeds, id of the program and bump value (a random number, usually starts from 255 and goes down). There is a 50% chance that the hash function returns a PDA with a public key. If it has a public key, the bump value is reduced by 1 and the PDA is calculated again. This happens till we get a PDA that does not have a public key [2]. The loop to calculate the correct PDA has costs associated with it.

The loop can be avoided if we calculate the PDA off-chain and send the bump to the program for which we got the correct PDA off-chain.

Recommendation

It is recommended to pass the bump value as an argument to the program function in order to reduce costs and pass that as bump in account contexts

Finding: XPSOL-05

Move looping done over `data.creators` off-chain

Base	Function	`validate_transfer_nft`	Line 190 -196	Medium	Fixed

Description

In `validate_transfer_nft` , there is unnecessary looping done over `data.creators` which is creating an instance of the struct `AnchorCreator` and pushing it in an array.

Recommendation

The looping over `data.creators` is unnecessary and can be avoided by performing this loop off chain and sending the data to the program function in the format in which it is to be consumed in the function.

Finding: XPSOL-06

Hard to read and error prone calculations

Base	In Account Context Structs:	**Initialize**	Line 421	Medium	Fixed
Base	In Account Context Structs:	**ValidatePause**	Line 449	Medium	Fixed
Base	In Account Context Structs:	**ValidateUnpause**	Line 476	Medium	Fixed
Base	In Account Context Structs:	**ValidateUpdateGroupkey**	Line 503	Medium	Fixed
Base	In Account Context Structs:	**ValidateTransferNft**	Line 551	Medium	Fixed
Base	In Account Context Structs:	**WithdrawFees**	Line 690	Medium	Fixed
Base	In Account Context Structs:	**ValidateUnfreezeNft**	Line 850	Medium	Fixed

Description

The size of the account is being explicitly defined in account context. This makes the code difficult to read and most importantly, there is room for error when performing the calculations manually to calculate the space that is needed for the account.

Recommendation

It is recommended to calculate the space needed for when we are initializing an account using `size_of` from `std::mem::size_of` as there are less chances of mistakes and it makes the code more readable.

It is to be noted that the space for the address will still have to be explicitly defined. Also, `size_of` only works when dealing with types that have predefined spaces.

Finding: XPSOL-07

Code duplication

Base	Function	`validate_pause`	Line 47-61	Medium	Fixed
Base	Function	`validate_unpause`	Line 70-84	Medium	Fixed
Base	Function	`validate_withdraw_fees`	Line 96-110	Medium	Fixed
Base	Function	`validate_update_groupkey`	Line 126-140	Medium	Fixed
Base	Function	`validate_transfer_nft`	Line 153-157, 163-171	Medium	Fixed

Description

The code to verify that the bridge is not paused and the action is not duplicated is being repeated in most functions.

Recommendation

The checks can be extracted from the program functions and added as constraints in account contexts.

Finding: XPSOL-08

Unused variables

Base	Function initialize	Line 34-35	Low	Fixed

Description

`cost_in_lamports` and `tx_fees` are initialized in the initialize function but they are never used in the program

Recommendation

Remove unused variables from the initialize function.

Executive Summary

Based on the audit findings the Client's contracts are: Well Secured

Not Secure	Insufficiently Secured	Secured	**Well Secured**

Disclaimers

SafePress Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions). The audit makes no statements or warranties on the security of the code. It also cannot be considered a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit cannot guarantee the explicit security of the audited smart contracts.

References

[1] Anchor-Lang Book, Enforcing uniqueness
https://book.anchor-lang.com/anchor_in_depth/PDAs.html#enforcing-uniqueness

[2] Anchor-Lang Book, Creation of a PDA
https://book.anchor-lang.com/anchor_in_depth/PDAs.html#creation-of-a-pda