Audit Report

December 31st, 2025

Andamio Platform - V2 Protocol

Contents

1 - Summary 3
1.a - Overview 3
1.b - Process 3
2 - Specification 4
2.a - UTxOs 4
2.b - Scripts 5
3 - Transactions 7
3.a - Access Token 7
3.b - IndexData Update 10
3.c - Global State 11
3.d - Local Sate Registry 15
4 - Audited Files 17
5 - Findings 19
5.a - AND-001 The IndexToken tokens can be stolen 20
5.b - AND-101 Publish purpose in IndexScript is too lax 21
5.c - AND-201 Adding a `PubKeyCredential` to the `initGSObsShList` forces the `IndexData` UTxO to be unspendable 22
5.d - AND-202 Missing checks in the `IndexData` fields related to the treasury fees output 23
5.e - AND-301 Use builtin constant `mkNil` instead of builtin function `mkNilData` 25
5.f - AND-302 Incorrect CIP-68 implementation 26
A Appendix 27
A.1 Terms and Conditions of the Commercial Agreement 27
A.2 Issue Guide 31
A.3 Revisions 32
A.4 About Us 32

1 - Summary

This report provides a thorough audit of the Andamio protocol, which allows users to create and earn credentials in a decentralized and verifiable manner.

The audit process revealed potential vulnerabilities related mostly to the safeguarding of the Index Token and the integrity of the contract datums.

The audit is conducted without warranties or guarantees of the quality or security of the code. It's important to note that this report only covers identified issues, and we do not claim to have detected all potential vulnerabilities.

1.a - Overview

The Andamio protocol helps user in creating and tracking credentials, allowing everyone to define the requirements to issue and use said credentials. The audited project consists of two set of validators:

- Access Token: A set of validators written in Plinth that allows decentralized minting of access tokens with unique token names. This is ensured by the creation of IndexUTxOs, that store valid token name ranges, excluding names from the ranges once they are minted.
- Plumbline: A set of validators written in Aiken that allow for global state tracking, that is the collection of local states that each user is enrolled. As well as a script to validate the registration of the local states.

1.b - Process

Our audit process involved a thorough examination of Andamio validators. Areas vulnerable to potential security threats were closely scrutinized, including those where attackers could exploit the validator's functions to disrupt the platform and its users. This included evaluating potential risks, such as Index token stealing, protocol halting and datum malformation. It also included evaluating common vulnerabilities, such as double satisfaction and minting policy vulnerabilities as well as optimizations techniques. Findings and feedback from the audit were communicated regularly to the Andamio team through Discord. Diagrams illustrating the necessary transaction structure for proper interaction with the Andamio protocol are attached as part of this report. The Andamio team addressed these issues in an efficient and timely manner, enhancing the overall security of the platform.

2 - Specification

2.a - UTxOs

2.a.a - Index Script UTxO

There can only be one UTxO. Identified by the token. Holds parameters used in various transactions

- Address:
  - Payment part: Hash of index_ref_script parametrized on `irpAdminCs` and `irpRefCs`
- Value:
  - 1 `index_script_token`
- Datum: `IndexData`
- RefScript: None

2.a.b - Index UTxO

There can be two or more UTxOs. Each represents a range of valid aliases to mint

- Address:
  - Payment part: Hash of index_script parameterized on `index_script_token`, `start_end_cs`, and `staking_script_hash`.
  - Staking part: `staking_script_hash`.
- Value:
  - 1 `index_script_token`
- Datum: `(BuiltinByteString, BuiltinByteString)`
- RefScript: None

2.a.c - Treasury

There can be zero or more UTxOs. Holds the collected fee from minting Access Tokens

- Address: `treasuryAddr` from `IndexData`
- Value: `mintAccessTokenValue` from `IndexData`
- Datum: `treasuryDat` from `Index Data`
- RefScript: None

2.a.d - Global State

There can be zero or more UTxOs. Hold the GlobalSate for each user, identified by the gToken with a unique alias

- Address:
  - Payment part: Hash of global_state_validator parameterized on `global_cs`, `index_ref_cs`, and `local_state_register_cs`.
  - Staking part: none
- Value: 1 `gToken`
- Datum: `GlobalStateDatum`
- RefScript: None

2.a.e - Local State registration

There can be zero or more UTxOs. Registers a LocalSate, identified by the lsRegistry token with a unique alias, holds the LocalSate script in the RefScript field.

- Address:
  - Payment part: Hash of local_state_registration parameterized on `stake_cred`, and `state_token_name`.

    - Staking part: `stake_cred` from params
- Value: 1 `lsRegistry`
- Datum: `ByteArray`
- RefScript: `ls_reference_script_hash`

2.b - Scripts

2.b.a - Index Script

- **Parameters**:
    - `referenceIndexCs`
    - `startEndCs`
    - `stakingScrHash`
- **Purpose**: Spend
    - **Redeemer**: none
    - **Datum**: `(BuiltinBytestring, BuiltinBytestring)`
- **Purpose**: Mint
    - **Redeemer**: BuiltinBytestring
- **Purpose**: Withdraw
    - **Redeemer**: none
- **Purpose**: publish
    - **Redeemer**: none

2.b.b - Init Index

- **Parameters**:
    - `txref`
- **Purpose**: Mint
    - **Redeemer**: none

2.b.c - Index Ref

- **Parameters**:
    - `irpAdminCs`
    - `irpRefCs`
- **Purpose**: Spend
    - **Redeemer**: `NewIndexData`
    - **Datum**: `IndexData`

2.b.d - Global State

- **Parameters**:
    - `global_cs`
    - `index_ref_cs`
    - `local_state_register_cs`
- **Purpose**: Spend
    - **Redeemer**: `GlobalStateAction`
    - **Datum**: `GlobalStateDatum`

2.b.e - Local State registration

- **Parameters**:
    - `stake_cred`
    - `state_token_name`
- **Purpose**: Spend
    - **Redeemer**: none

    - **Datum**: `ByteArray`
- **Purpose**: Mint
    - **Redeemer**: `List<LocalStateRegistrationData>`

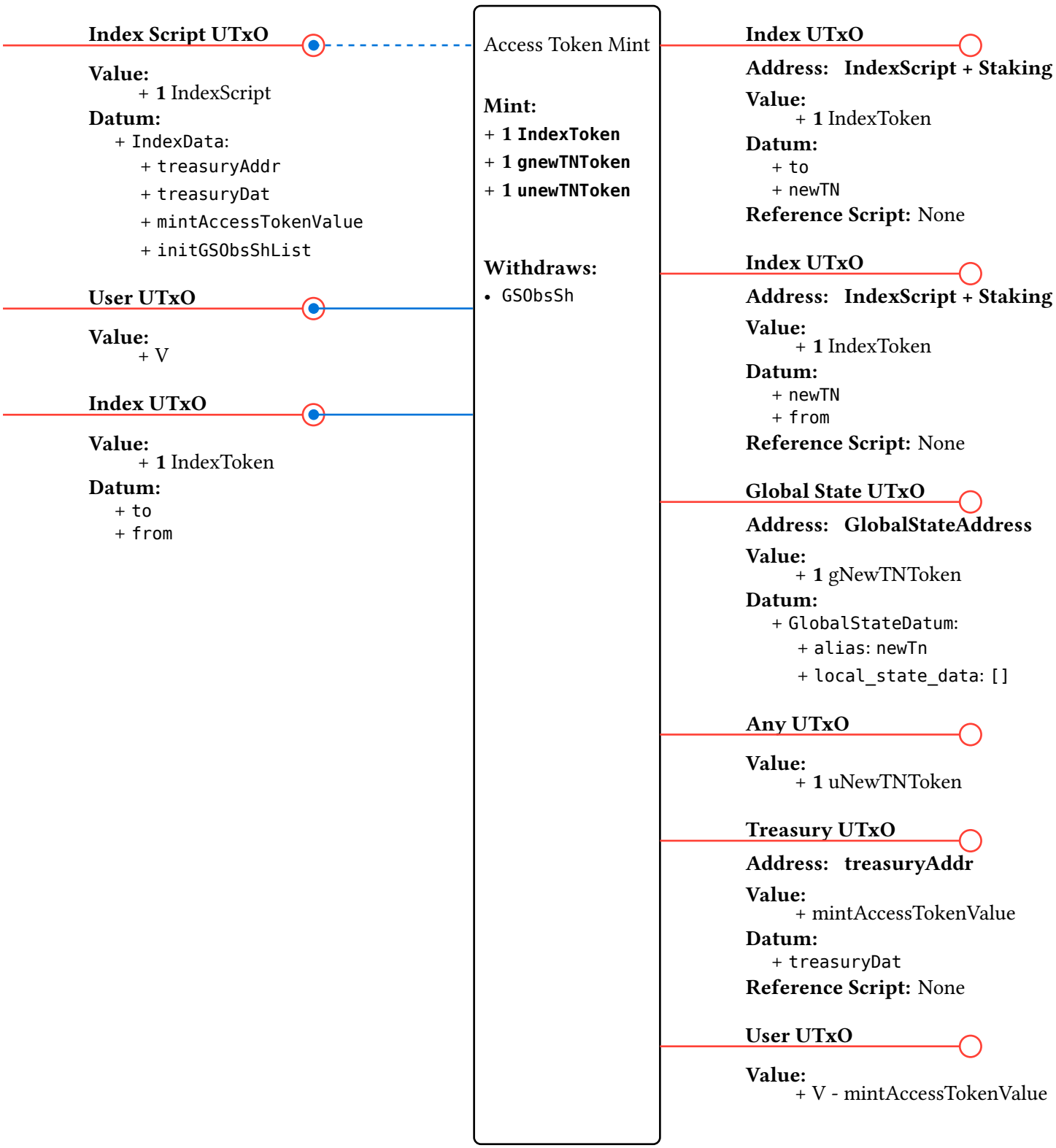
2.b.f - Init Global State

- **Parameters**:
    - `global_policy_id`
    - `init_index_policy_id`
    - `global_state_addr`
- **Purpose**: Withdraw
    - **Redeemer**: `ByteArray`
- **Purpose**: Publish
    - **Redeemer**: none

3 - Transactions

3.a - Access Token

3.a.a - Mint



Note:

IndexToken = IndexScript + " "

gNewTNToken = IndexScript + gnewTN

uNewTNToken = IndexScript + unewTN

to < newTN < from

GSObsSh ∈ initGSObsShList

Figure 1: Access Token mint transaction

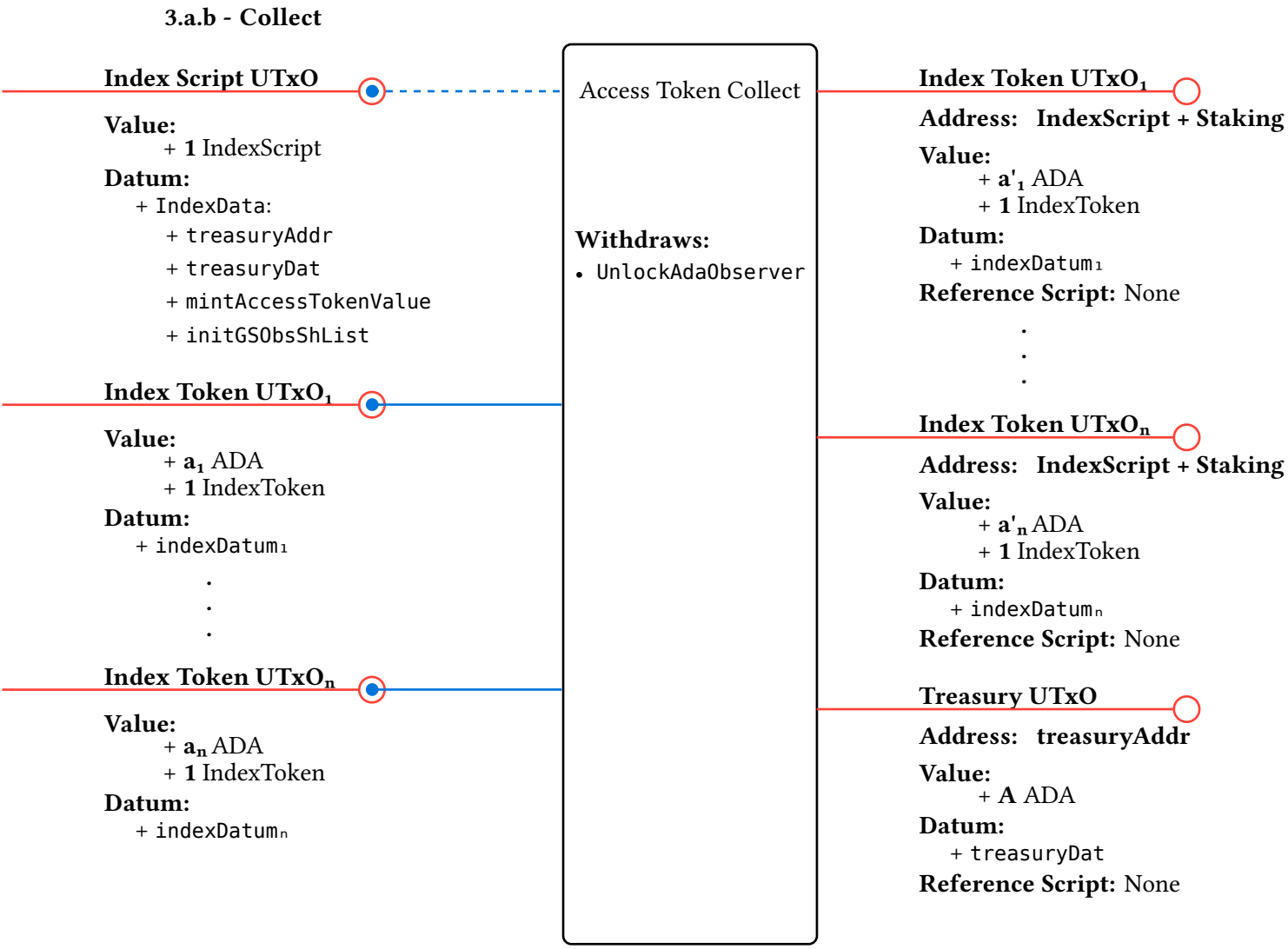
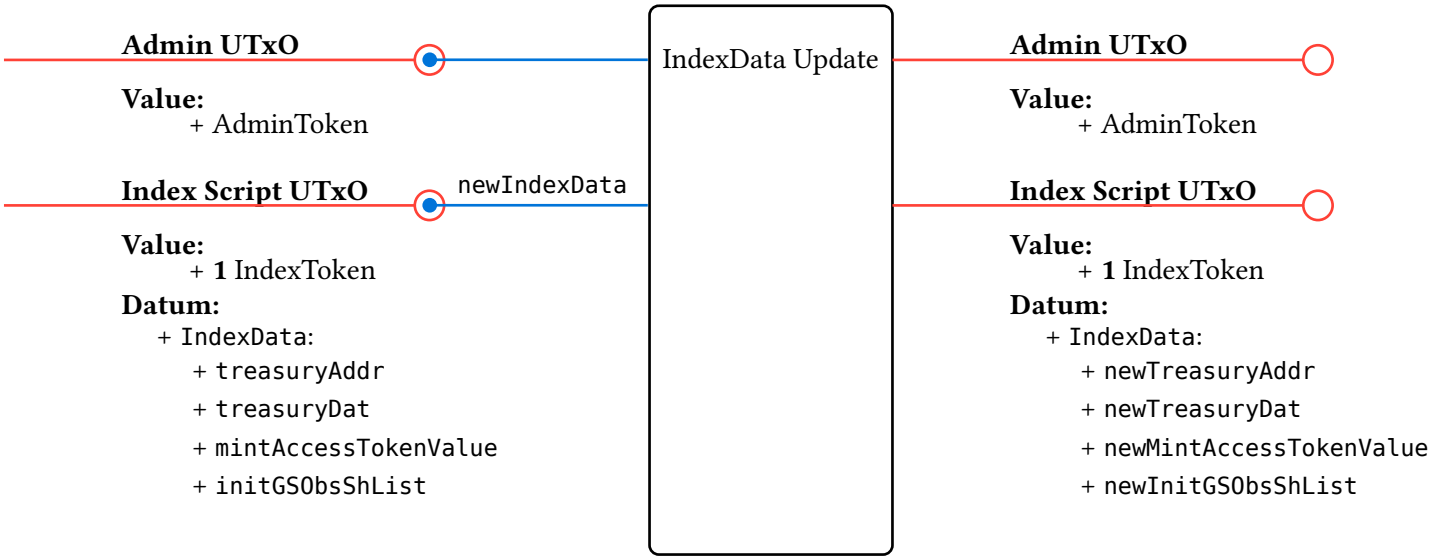


Figure 2: Access Token Collect ADA transaction

3.b - IndexData Update



Note:

AdminToken = `"admin one-shot minting policy"` + any

The amount of AdminToken is not checked, but it can be assumed to always be 1.

`newTreasuryAddr = newIndexData.newTreasuryAddr`

`newTreasuryDat = newIndexData.newTreasuryDat`

`newMintAccessTokenValue = newIndexData.newMintAccessTokenValue`

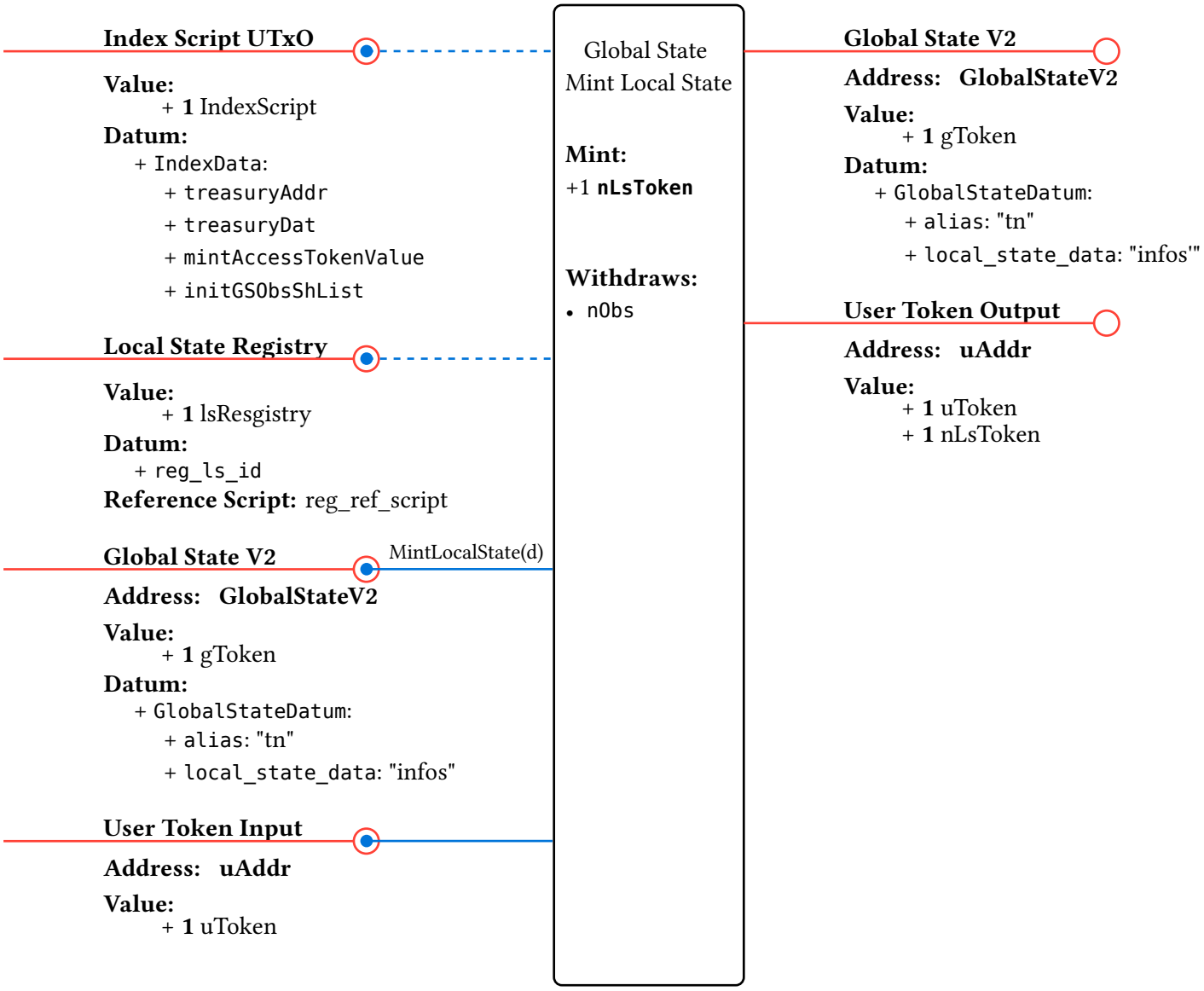
`newInitGSObsShList = newIndexData.newInitGSObsSh` ▶ `initGSObsShList`

`newIndexData.newInitGSObsSh` must be a `ScriptCredential`

Figure 3: IndexData Update transaction

3.c - Global State

3.c.a - Mint



Note:

gToken = IndexScript + 'g' + tn

uToken = IndexScript + 'u' + tn

nLsToken = local_state_cs + tn

d = {local_state_id, local_state_index, local_state_data_hash, local_state_cs}

reg_ls_id == local_state_id

reg_ref_script == local_state_cs

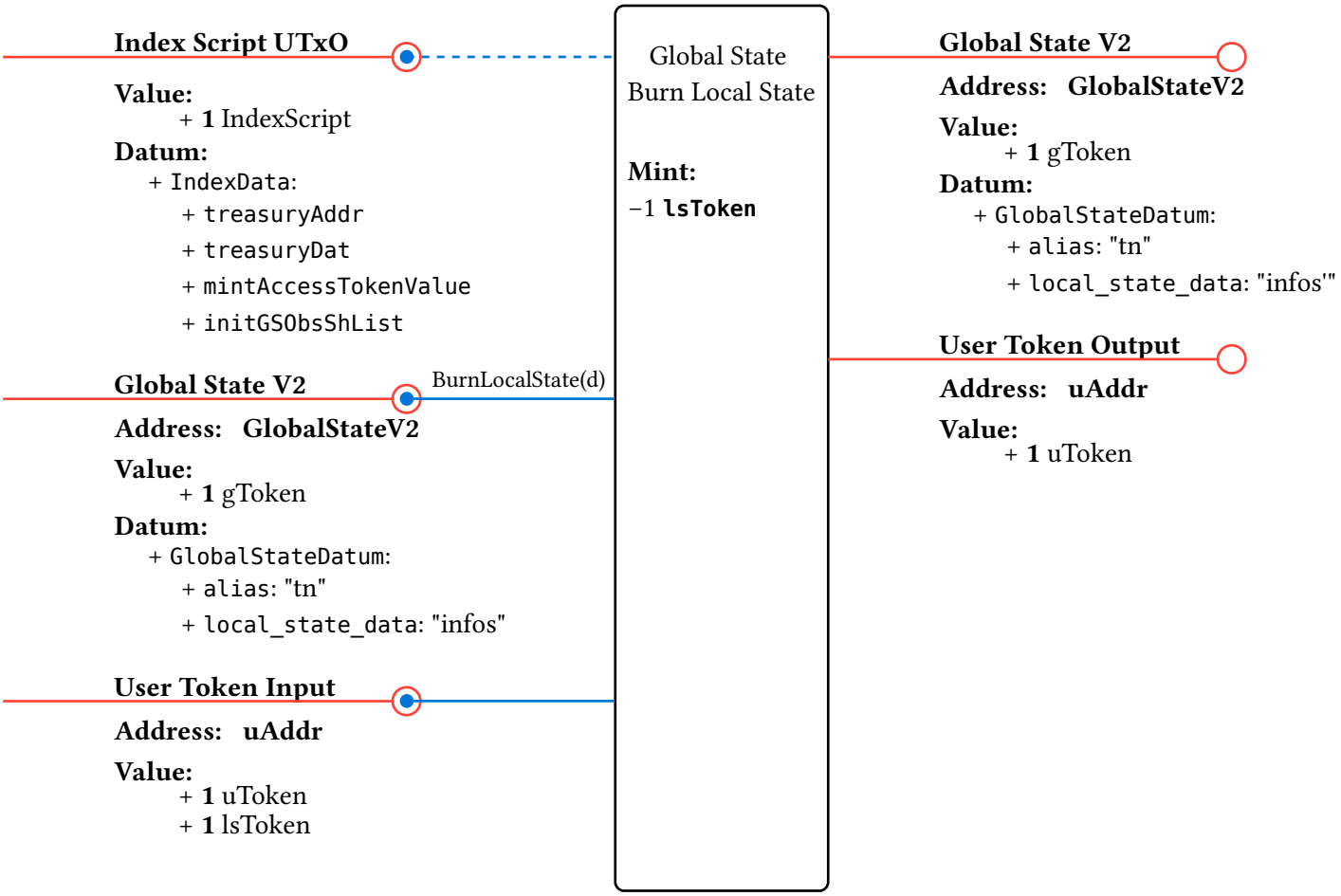
newLsInfo = hash_local_state(local_state_data_hash, local_state_cs)

infos' = append newLsInfo to infos and delete (local_state_data_hash, "") if local_state_data_hash ∈ infos

nObs ∈ initGSObsShList

Figure 4: Global State Mint Local State transaction

3.c.b - Burn



Note:

`gToken` = `IndexScript` + `'g'` + `tn`

`uToken` = `IndexScript` + `'u'` + `tn`

`lsToken` = `local_state_cs` + `tn`

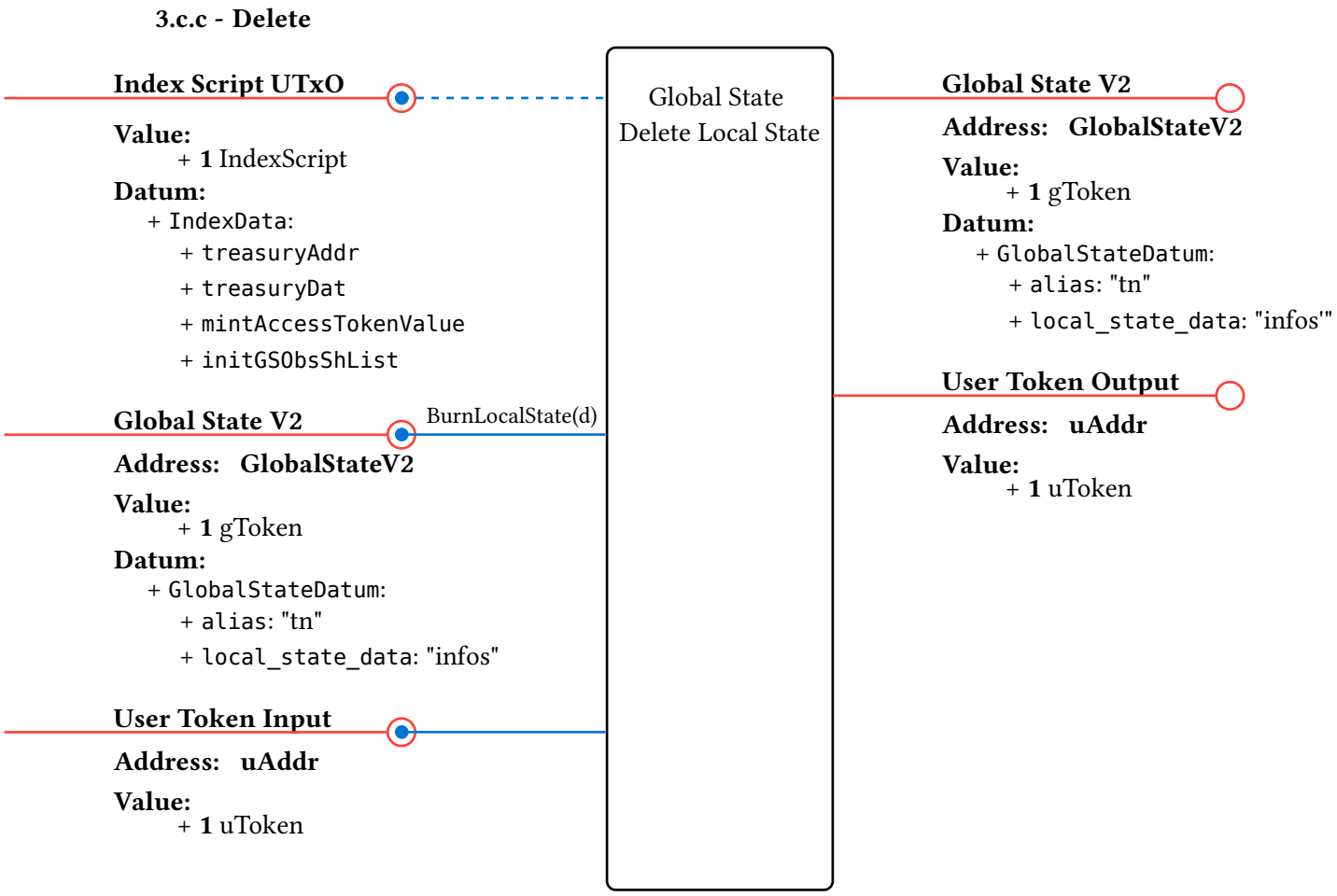
d = {local_state_id, local_state_index, local_state_data_hash, local_state_cs, new_data_hash}

`reg_ls_id` == `local_state_id`

`oldLsInfo` = (local_state_id, hash_local_state(local_state_data_hash, local_state_cs=)

`infos'` = delete `oldLsInfo` from `infos` and append (local_state_id, hash_local_state(`local_state_data_hash`, “”))

Figure 5: Global State Burn Local State transaction



Note:

`gToken` = `IndexScript` + `'g'` + `tn`

`uToken` = `IndexScript` + `'u'` + `tn`

`lsToken` = `local_state_cs` + `tn`

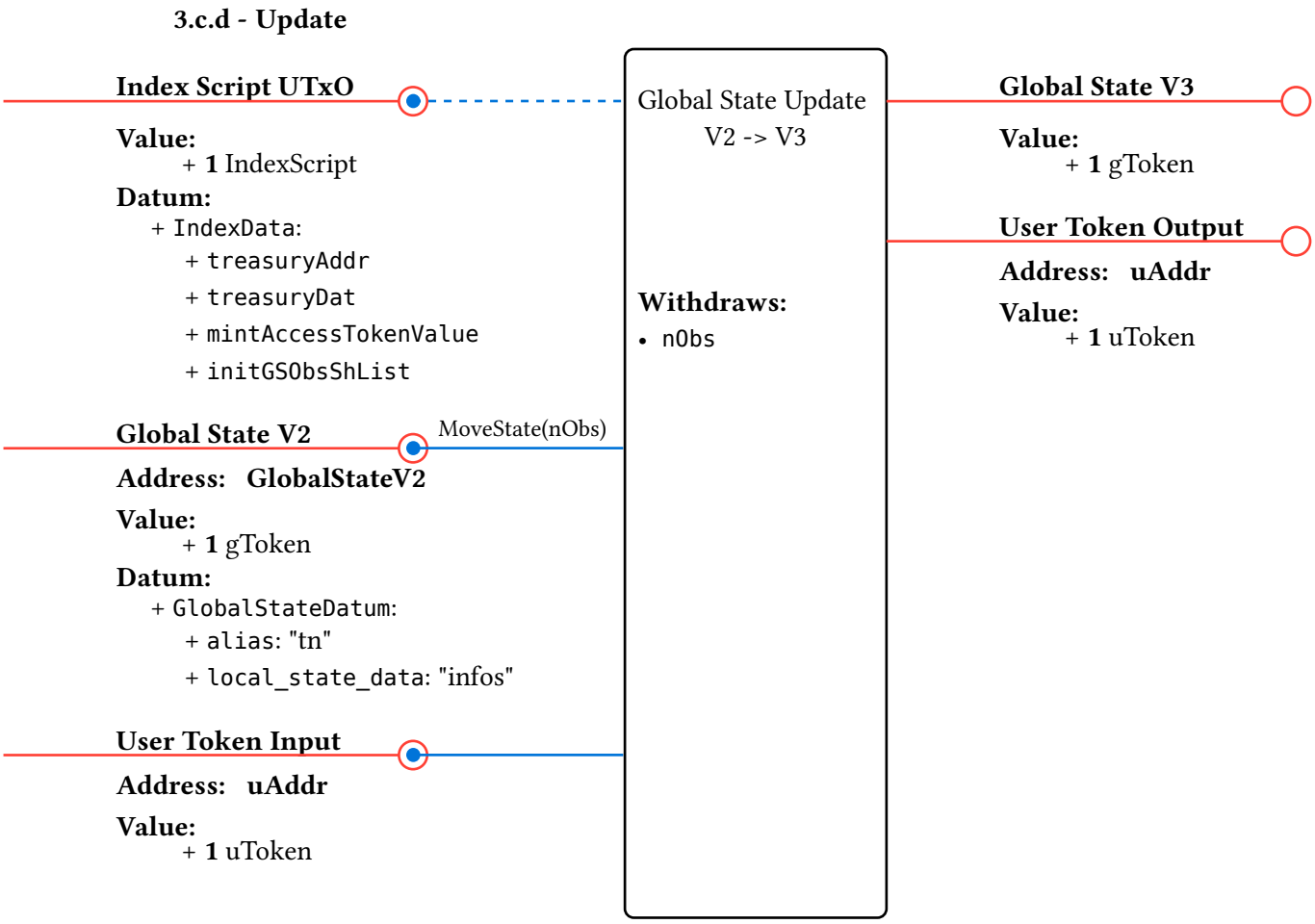
d = {local_state_id, local_state_index, local_state_data_hash}

`reg_ls_id` == `local_state_id`

`oldLsInfo` = (local_state_id, hash_local_state(local_state_data_hash, “”)

`infos'` = delete `oldLsInfo` from `infos`

Figure 6: Global State Delete Local State transaction



Note:

`gToken = IndexScript + 'g' + tn`

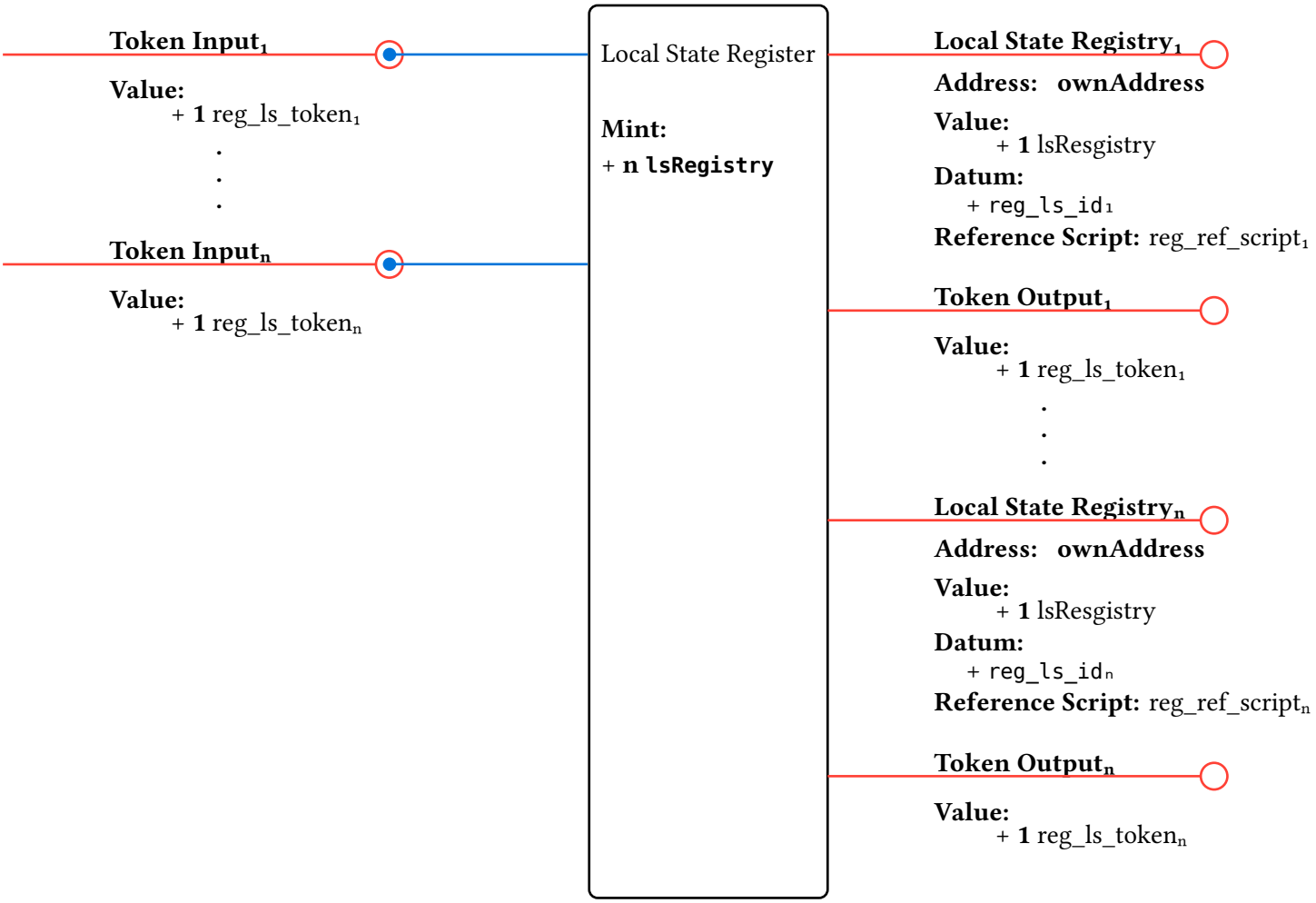
`uToken = IndexScript + 'u' + tn`

`nObs ∈ initGSObsShList`

Figure 7: Global State Update transaction

3.d - Local Sate Registry

3.d.a - Register



Note:

ownAddress = own_script + stake_cred

stake_cred is a parameter of the script
lsRegistry token name is a parameter of the script

lsRegistry minting redeemer = [lsRegData$_1$, ... ,lsRegData$_n$]

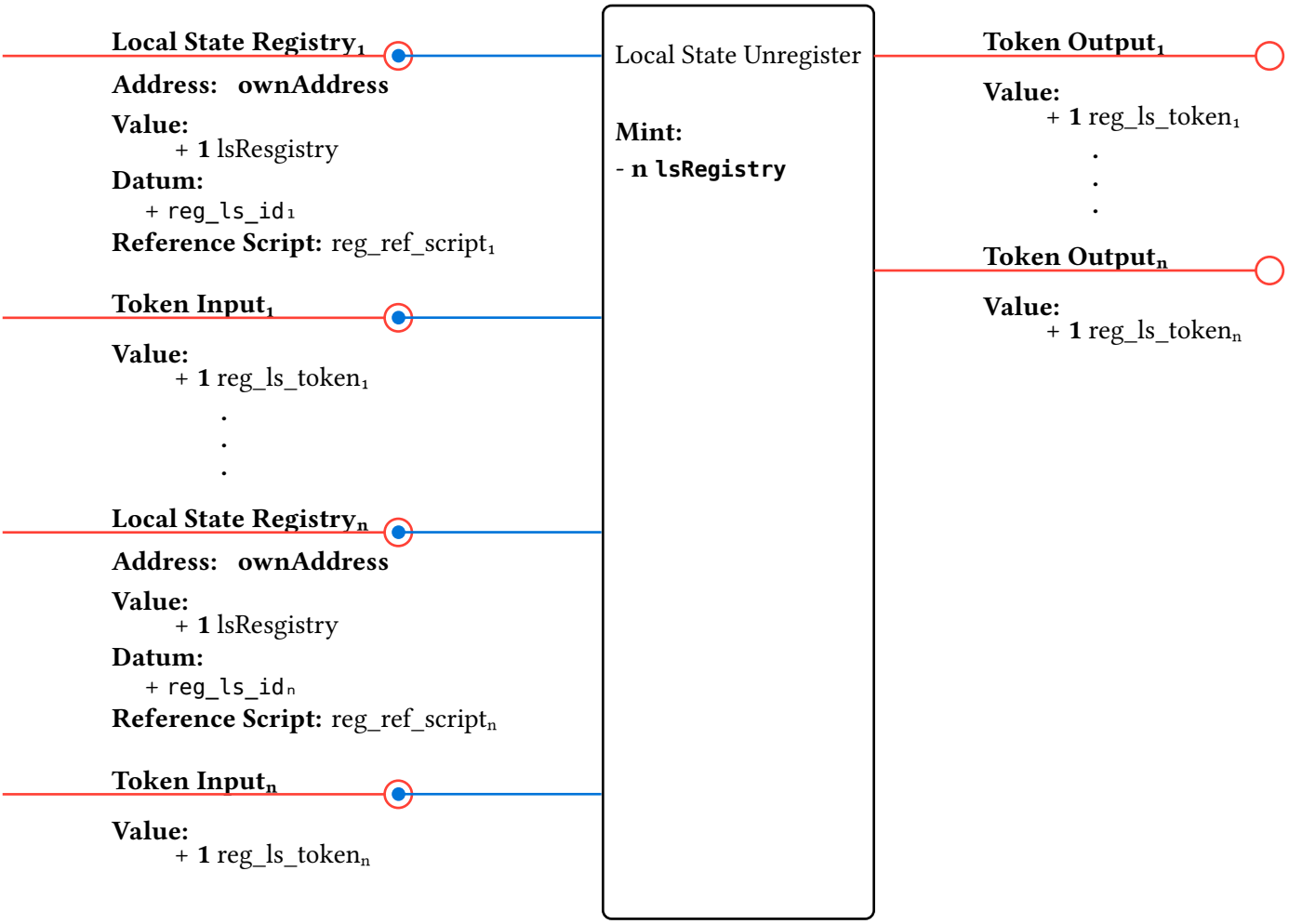
$$\text{lsRegData}_i = \{\text{reg\_ls\_id}_i, \text{reg\_ref\_script}_i\}$$

$$\text{lsRegData}_i \neq \text{lsRegData}_j \forall i, j$$

reg_ls_token$_i$ has policy reg_ls_id$_i$
reg_ls_tokens can be in any output

Figure 8: Local State Register transaction

3.d.b - Unregister



Note:

ownAddress = own_script + stake_cred

stake_cred is a parameter of the script
lsRegistry token name is a parameter of the script

lsRegistry minting redeemer = []

$\text{lsRegData}_i = \{\text{reg\_ls\_id}_i, \text{reg\_ref\_script}_i\}$

$\text{lsRegData}_i \neq \text{lsRegData}_j \forall i, j$

reg_ls_token$_i$ has policy reg_ls_id$_i$
reg_ls_tokens can be in any output

Figure 9: Local State Unregister transaction

4 - Audited Files

Below is a list of all audited files in this report. Any files **not** listed here were **not** audited. The final state of the files for the purposes of this report is considered to be commit `2e67b3c848c600ad9e5aaf6ae057a845767b7f97` in the `acces-token` repo and commit `16f76214ec98ced575be2eb253fa9ed13156402b` in the `plumbline-global-state` repo.

Filename
./validators/global/init_global_state_obs.ak
./validators/global/local_state_registration.ak
./validators/global/global_state.ak
./lib/types/index_data.ak
./lib/plumbline/plumb_output.ak
./lib/plumbline/plumb_bytearray.ak
./lib/plumbline/plumb_address.ak
./lib/plumbline/plumb_value.ak
./lib/plumbline/plumb_input.ak
./lib/plumbline/types/local_state_registration_data.ak
./lib/plumbline/types/flat_value.ak
./lib/plumbline/types/global_state.ak
./lib/plumbline/plumb_redeemers.ak
./lib/plumbline/plumb_global_state.ak
./andamio-access-token/src/Index/OnChain/IndexScripts.hs
./andamio-access-token/src/Index/OnChain/InitIndexPolicy.hs
./andamio-access-token/src/Index/OnChain/IndexRefScript.hs
./andamio-access-token/src/Index/OnChain/IndexRef/IndexRefParams.hs
./andamio-access-token/src/Index/OnChain/IndexRef/IndexData.hs
./andamio-access-token/src/Index/OnChain/IndexScripts/SpendingScript.hs

./andamio-access-token/src/Index/OnChain/IndexScripts/IndexParams.hs

./andamio-access-token/src/Index/OnChain/IndexScripts/MintingScript.hs

./andamio-access-token/src/Index/OnChain/IndexScripts/UnlockAdaObserver.hs

./andamio-utility-on-chain/src/Andamio/Utility/Cbors.hs

./andamio-utility-on-chain/src/Andamio/Utility/LazyContextV3.hs

./andamio-utility-on-chain/src/Andamio/Utility/ValidatorOnChainNames.hs

./andamio-utility-on-chain/src/Andamio/Utility/TxOut.hs

./andamio-utility-on-chain/src/Andamio/Utility/Address.hs

./andamio-utility-on-chain/src/Andamio/Utility/SampleData.hs

./andamio-utility-on-chain/src/Andamio/Utility/TxInInfo.hs

./andamio-utility-on-chain/src/Andamio/Utility/Datum.hs

./andamio-utility-on-chain/src/Andamio/Utility/OnChain.hs

./andamio-utility-on-chain/src/Andamio/Utility/Redeemers.hs

./andamio-utility-on-chain/src/Andamio/Utility/BuiltinByteString.hs

./andamio-utility-on-chain/src/Andamio/Utility/Value.hs

./andamio-utility-on-chain/src/Andamio/Utility/Validators/TokenNamePolicy.hs

./andamio-utility-on-chain/src/Andamio/Utility/Validators/AlwaysTrue.hs

./andamio-utility-on-chain/src/Andamio/Utility/Validators/NFTPolicyTnStrict.hs

5 - Findings

ID	Title	Severity	Status
AND-001	The IndexToken tokens can be stolen	Critical	Resolved
AND-101	Publish purpose in IndexScript is too lax	Major	Resolved
AND-201	Adding a `PubKeyCredential` to the `initGSObsShList` forces the `IndexData` UTxO to be unspendable	Minor	Resolved
AND-202	Missing checks in the `IndexData` fields related to the treasury fees output	Minor	Acknowledged
AND-301	Use builtin constant `mkNil` instead of builtin function `mkNilData`	Info	Resolved
AND-302	Incorrect CIP-68 implementation	Info	Resolved

5.a - AND-001 The IndexToken tokens can be stolen

Category	Commit	Severity	Status
Vulnerability	`f4257ad502abbc2ad31c3ee09c40878c1ee71e67`	Critical	Resolved

5.a.a - Description

The mint access token operation enables any user of the protocol to mint their own access token. An important characteristic of this token is its uniqueness. The policy of the tokens is shared among all access tokens (including Index tokens), but each has a unique name. This is ensured by consuming an Index UTxO that, in its datum, has a range of valid token names. This UTxO is identified by its Index token, and the minting policy validation ensures that the new token name selected by the user falls within the corresponding range. Additionally, the validation checks that two Index UTxOs are created, each with its Index token and its datum with the valid range considering the new token name.

The validation allows the consumption of more than one Index UTxO, but the described validations are only performed over the first Index UTxO. This allows any user to steal Index tokens.

This is catastrophic to the protocol because anyone could create an Index UTxO with a valid Index token and its datum with a valid but fake range of token names. That can then be used to mint access tokens with already selected names. Thus, losing the uniqueness characteristic of the access tokens.

5.a.b - Recommendation

Improve the whole validation of the operation so it only allows to include one Index UTxO input.

5.a.c - Resolution

Resolved in commit `ea1e0294bdf8e3ce0cf1cae9ffea600e552f00c0`.

5.b - AND-101 Publish purpose in IndexScript is too lax

Category	Commit	Severity	Status
Robustness	`40ebfece6353f6e5090abc3a494efdd5c5472ac9`	Major	Resolved

5.b.a - Description

The IndexScript validator can be run with multiple purposes, namely the `spend`, `mint` and `withdraw` purposes have interesting validation implemented. To allow withdrawing from this validator, it must be registered. For this reason the publish purpose also had to be implemented.

However, the current implementation is just `True`, allowing anyone to register the credential, but also anyone to unregister it. Considering that registering requires a deposit of at least 5 ADAs and that unregistering unlocks those funds, this would allow an attacker to constantly unregister the credential, stalling the protocol and constantly pocketing the funds.

5.b.b - Recommendation

We recommend updating the validation so that only registering can be done, all other certificates should be disallowed.

5.b.c - Resolution

Resolved in commit `2e67b3c848c600ad9e5aaf6ae057a845767b7f97`.

5.c - AND-201 Adding a `PubKeyCredential` to the `initGSObsShList` forces the `IndexData` UTxO to be unspendable

Category	Commit	Severity	Status
Bug	`ea1e0294bdf8e3ce0cf1cae9ffea600e552f00c0`	Minor	Resolved

5.c.a - Description

The transaction that allows the minting of an access token includes a reference input whose datum is `IndexData`. The information there is relevant because it determines the amount and where the fees must be paid. More interestingly, it includes a list of script hashes `initGSObsShList` from which only one must be run by performing a withdraw.

The `IndexData` information can be updated by a simpler transaction that consumes the corresponding UTxO, under the `IndexRef` script, and produces a new output with the updated `IndexData`. Related to the updated `initGSObsShList` list, it must contain the current credentials and only one new credential can be added. Other check that are performed over this list is that each current credential in the list is a `ScriptCredential`, and not a `PubKeyCredential`.

Now, there isn't any restriction to add a new `PubKeyCredential` to the list. Thus, this blocks the `IndexData` UTxO from future updates.

5.c.b - Recommendation

Review the update logic of the `initGSObsShList` field of the `IndexData` datum.

5.c.c - Resolution

Resolved in commit `0e848f053620ee00a0aaaf091ea2205943ed470a`

5.d - AND-202 Missing checks in the `IndexData` fields related to the treasury fees output

Category	Commit	Severity	Status
Vulnerability	`ea1e0294bdf8e3ce0cf1cae9ffea600e552f00c0`	Minor	Acknowledged

5.d.a - Description

The transaction that allows the minting of an access token includes a reference input whose datum is `IndexData`. The information there is relevant because it determines the amount and where the fees must be paid. In fact, it precisely describes how this output, called treasury where the fees are paid, must look: the address (`treasuryAddr`), the amount of tokens (`mintAccessTokenValue`), and the datum (`treasuryDat`).

With this in mind, it is important to check that each part of the `IndexData`, which will constitute the treasury output, is consistent with the ledger rules. Otherwise, there are two possible outcomes: The access token can't be created because the treasury output is wrong, or the treasury output is created, but then it can't be consumed.

We acknowledge the following cases:

- The `treasuryAddr` could be unknown, either `PubKey` or `Script`. This blocks (possibly) forever the treasury UTxOs with that address.
- The `mintAccessTokenValue` can be updated to contain only non-ADA tokens. This momentarily blocks the protocol, but it's solvable by updating the mintAccessTokenValue field again.
- The `treasuryDat` could be a datum represented by a datum hash, but by mistake, it might be a hash from an unknown datum. This blocks (possibly) forever the treasury UTxOs with that datum. Those UTxO can be created because you don't need to provide the pre-image of the datum hash. But of course, you need that later when consuming.

5.d.b - Recommendation

The general solution would be to enforce (as much as possible) the ledger rules for each IndexData field that describes the treasury output.

- For the `treasuryAddr`, if it is a `PubKey`, one solution is to enforce that the `PubKey` signs the transaction that updates the IndexData. For the `Script`, something similar could be applied by checking a withdraw action over that `Script`. Of course, this implies the `Script` needs to implement some kind of mechanism for when it's run for this particular operation.
- For the `mintAccessTokenValue`, it is easy to enforce that ADA must be always part of the list. But there are more difficult scenarios where the included token doesn't even exist.
- For the `treasuryDat`, the datum type could be limit to be only: NoOutputDatum or OutputDatum (inline).

5.d.c - Resolution

The **Project team** decided not to address this finding. The team's main argument for not resolving it is related to future ledger updates that the Cardano protocol may have. The goal is to maintain the current level of flexibility resulting from not including the checks.

As the **Audit team**, we advise implementing the aforementioned validations. However, we acknowledged the reasons and, more importantly, the creation and update of the IndexData UTxO can only be performed by the Project protocol administrator, which is represented onchain as an NFT locked in a multisig wallet. Thus, we encourage the Project team to implement additional off-chain measures to ensure that no "corrupted" or "malformed" information is included in the IndexData UTxO.

5.e - AND-301 Use builtin constant `mkNil` instead of builtin function `mkNilData`

Category	Commit	Severity	Status
CodeQuality	`ea1e0294bdf8e3ce0cf1cae9ffea600e552f00c0`	Info	Resolved

5.e.a - Description

Module locations where the `mkNil` constant can be used instead of `mkNilData`.

Access Token
- `Index/OnChain/IndexScripts/UnlockAdaObserver.hs:75`
- `Index/OnChain/IndexScripts/UnlockAdaObserver.hs:85`
- `Index/OnChain/IndexScripts/MintingScript.hs:46`
- `Index/OnChain/IndexScripts/SpendingScript.hs:22`
- `Index/OnChain/IndexScripts/SpendingScript.hs:23` (two occurrences)

Utility OnChain
- `Andamio/Utility/Address.hs:19`
- `Andamio/Utility/Address.hs:23`
- `Andamio/Utility/Address.hs:27`
- `Andamio/Utility/Address.hs:31` (two occurrences)
- `Andamio/Utility/Address.hs:35`
- `Andamio/Utility/Address.hs:39`
- `Andamio/Utility/Datum.hs:20`
- `Andamio/Utility/Datum.hs:24`

5.e.b - Recommendation

Replace all occurrences of `mkNilData` with `mkNil`.

5.e.c - Resolution

Resolved in commit `e15a03de83500ddab71fbf710dbe3573d2f87eda`

5.f - AND-302 Incorrect CIP-68 implementation

Category	Commit	Severity	Status
CodeQuality	`ea1e0294bdf8e3ce0cf1cae9ffea600e552f00c0`	Info	Resolved

5.f.a - Description

The protocol intends to use CIP-68 for the Access tokens, assigning the `222` prefix to user access tokens and the `100` prefix to the global state access token. But the current implementation is just prepending `222` or `100` to each token name. Instead of following CIP-67 for the token name enconding. Furthermore, the datum where the `100` token is stored, does not support the correct format to hold CIP-68 metadata.

5.f.b - Recommendation

After talking with the team we came to the conclusion that CIP-68 is not needed, as these tokens won't hold any metadata. So we recommend changing the prefixes to avoid confusion to `u` and `g` denoting user and global access tokens respectively.

5.f.c - Resolution

The finding should be implemented across both repositories:

Resolved in commit `access-token:7488503b4b79d9a28cf378ab6878de3dbaf4cf33`

Resolved in commit `plumbline-global-state:a2d2cc72f2a0c654a61c15b4019f263f87bb642a`

A Appendix

A.1 Terms and Conditions of the Commercial Agreement

A.1.1 Confidentiality

Both parties agree, within a framework of trust, to discretion and confidentiality in handling the business. This report cannot be shared, referred to, altered, or relied upon by any third party without Txpipe LLC, 651 N Broad St, Suite 201, Middletown registered at the county of New Castle, written consent.

The violation of the aforementioned, as stated supra, shall empower TxPipe to pursue all of its rights and claims in accordance with the provisions outlined in Title 6, Subtitle 2, Chapter 20 of the Delaware Code titled "Trade Secrets,", and to also invoke any other applicable law that protects or upholds these rights.

Therefore, in the event of any harm inflicted upon the company's reputation or resulting from the misappropriation of trade secrets, the company hereby reserves the right to initiate legal action against the contractor for the actual losses incurred due to misappropriation, as well as for any unjust enrichment resulting from misappropriation that has not been accounted for in the calculation of actual losses.

A.1.2 Service Extension and Details

This report does not endorse or disapprove any specific project, team, code, technology, asset or similar. It provides no warranty or guarantee about the quality or nature of the technology/code analyzed.

This agreement does not authorize the client Andamio Platform to make use of the logo, name, or any other unauthorized reference to Txpipe LLC, except upon express authorization from the company.

TxPipe LLC shall not be liable for any use or damages suffered by the client or third-party agents, nor for any damages caused by them to third parties. The sole purpose of this commercial agreement is the delivery of what has been agreed upon. The company shall be exempt from any matters not expressly covered within the contract, with the client bearing sole responsibility for any uses or damages that may arise.

Any claims against the company under the aforementioned terms shall be dismissed, and the client may be held accountable for damages to reputation or costs resulting from non-compliance with the aforementioned provisions. **This report provides general information and is not intended to constitute financial, investment, tax, legal, regulatory, or any other form of advice.**

Any conflict or controversy arising under this commercial agreement or subsequent agreements shall be resolved in good faith between the parties. If such negotiations do not result in a conventional agreement, the parties agree to submit disputes to the courts of Delaware and to the laws of that jurisdiction under the powers conferred by the Delaware Code, TITLE 6, SUBTITLE I, ARTICLE 1, Part 3 § 1-301. and Title 6, SUBTITLE II, chapter 27 §2708.

A.1.3 Disclaimer

The audit constitutes a comprehensive examination and assessment as of the date of report submission. The company expressly disclaims any certification or endorsement regarding the subsequent performance, effectiveness, or efficiency of the contracted entity, post-report delivery, whether resulting from modification, alteration, malfeasance, or negligence by any third party external to the company.

The company explicitly disclaims any responsibility for reviewing or certifying transactions occurring between the client and third parties, including the purchase or sale of products and services.

This report is strictly provided for ***informational purposes*** and reflects solely the due diligence conducted on the following files and their corresponding hashes using sha256 algorithm:

Filename: `./validators/global/init_global_state_obs.ak` **Hash**: `3b42450f614648b5aa92c2cdcef145030ff9627e0df46aad572f225fcddcb28f`
Filename: `./validators/global/local_state_registration.ak` **Hash**: `2c189861d60312aa64269779aee94fb88593c0430b1c0bd89fc323fdfc609a72`
Filename: `./validators/global/global_state.ak` **Hash**: `5689c9305ea73f5f966961dd4169926ecd3ba673f8c3fdc844d5894409946f49`
Filename: `./lib/types/index_data.ak` **Hash**: `54bca5b156026c30d13a4761a13974da11997dde00f718f6ccdb8742a3d4ed2b`
Filename: `./lib/plumbline/plumb_output.ak` **Hash**: `6f94217df5b245b664eb9d16ee0844d55cba2fd3697b4bd53166e1d2977b5001`
Filename: `./lib/plumbline/plumb_bytearray.ak` **Hash**: `27b5f1b0c2aada742c389613ba98fab456a6dd36f3b58732f8b002d92eeffec8`
Filename: `./lib/plumbline/plumb_address.ak` **Hash**: `311d28ddb5c728c7bc6766d1944d510e51c46adf92774edb36bb0b5b32bbc665`
Filename: `./lib/plumbline/plumb_value.ak` **Hash**: `48b45a0215bd4e76cb4d246c1ec98057849b72ff9cf67685c62beaddaf16d831`
Filename: `./lib/plumbline/plumb_input.ak` **Hash**: `c0a519908f682dad114b00faa83b7200b877d7bff1ba03894968a0ddcc7c5d43`
Filename: `./lib/plumbline/types/local_state_registration_data.ak` **Hash**: `ba1ddd18f4f2c10ea0c3821a69b166635d8cbfaf9df98e6ec50aa74ff0cf45db`
Filename: `./lib/plumbline/types/flat_value.ak` **Hash**: `bf30223cae5f01cc528809df55b8b900f2fce5a70608f7b1b430c3061d893808`
Filename: `./lib/plumbline/types/global_state.ak` **Hash**: `8e3556d2f42514ccb17549479e01fa514568c5c37da6fc4098d9daa5406b2c59`
Filename: `./lib/plumbline/plumb_redeemers.ak` **Hash**: `1d3debea2979f250ec2b73fa8ef930208db6265d5a36540568a265d7ba0b659d`
Filename: `./lib/plumbline/plumb_global_state.ak` **Hash**: `5680a4af9c764d07fdc592e5cf605cef9afa23bc5e015a73c622cfc2f0ea491a`
Filename: `./andamio-access-token/src/Index/OnChain/IndexScripts.hs` **Hash**: `8c773bb92dbc610e7eca98008125ae3a9e72da3f725f6906569499f95c8719ce`
Filename: `./andamio-access-token/src/Index/OnChain/InitIndexPolicy.hs` **Hash**: `6ae27eb662ecd12b699ed1b7ec85d289f371ca196af178c45afd6bc9ee165660`
Filename: `./andamio-access-token/src/Index/OnChain/IndexRefScript.hs` **Hash**: `a7a46edbabd745856d37e91f3e197e4e4a16c9733aca9e718e15b637514ffeb3`
Filename: `./andamio-access-token/src/Index/OnChain/IndexRef/IndexRefParams.hs` **Hash**: `6eb886c487c7d3ca70a4ee8efa18c51b6ca21f85076f2a442ada349f8a1829c5`
Filename: `./andamio-access-token/src/Index/OnChain/IndexRef/IndexData.hs` **Hash**: `72451c2924b514173640f0aff42d6b12ba6fb61ca9e408f53e66b5af5619e83c`

Filename: ./andamio-access-token/src/Index/OnChain/IndexScripts/SpendingScript.hs **Hash**: 0ac325933b178f1bf8fb4b931f8fa3598ff6fbb9fc24971f2c5bc4e1528e7d63
Filename: ./andamio-access-token/src/Index/OnChain/IndexScripts/IndexParams.hs **Hash**: eaf82dd43e59fc4f0e8f9c6a2077aaad1d6189789d700fce5e544b512271a7ab
Filename: ./andamio-access-token/src/Index/OnChain/IndexScripts/MintingScript.hs **Hash**: 4e0fdd3b9aabe4023f92c23d447dd1279ee35f32c1944f66211bb9a69af1ded5
Filename: ./andamio-access-token/src/Index/OnChain/IndexScripts/UnlockAdaObserver.hs **Hash**: c59995e165e47d7dd72b616f7c4a3293ce0cf78d53e22abcb9c6fc82536d2692
Filename: ./andamio-utility-on-chain/src/Andamio/Utility/Cbors.hs **Hash**: 92dd55f51b11b8fd69b59da81887c5ff4f9691293d0dff450b5b5a78c064c4a0
Filename: ./andamio-utility-on-chain/src/Andamio/Utility/LazyContextV3.hs **Hash**: 83674837c401a773943503d9186d661f70c9a1f5bd94defff163ba5f04dc0546
Filename: ./andamio-utility-on-chain/src/Andamio/Utility/ValidatorOnChainNames.hs **Hash**: 0c923e3bd729c288f4665d3dd288bd5210638970c9cce76333d3ca849a8ffbdb
Filename: ./andamio-utility-on-chain/src/Andamio/Utility/TxOut.hs **Hash**: d9d0bde0cb587192069675c7b8d17ee995a9079af49fe4c8b82d50f98399f6fc
Filename: ./andamio-utility-on-chain/src/Andamio/Utility/Address.hs **Hash**: 2cf73dc0a03af46be8ec6ab599b4c92a2eca563ad21ea684e329dcb7a0a66daa
Filename: ./andamio-utility-on-chain/src/Andamio/Utility/SampleData.hs **Hash**: a87479ec8472efa15fb7d822c08104cbf12e2cf0a1504d7ca168dd518afc0cdc
Filename: ./andamio-utility-on-chain/src/Andamio/Utility/TxInInfo.hs **Hash**: 664a6cee09e8bddbe0d9954c3cb40b56ba821292a79e5f2f03284e2ba888c47f
Filename: ./andamio-utility-on-chain/src/Andamio/Utility/Datum.hs **Hash**: a26bfcbe61cb209a0ede01052c085bc495d706d99086ef8ff50c0597fcda0b9d
Filename: ./andamio-utility-on-chain/src/Andamio/Utility/OnChain.hs **Hash**: 2828a79641742e3a8c7d418a081b3a46d30322afaa8f6403b5b8f5762831d192
Filename: ./andamio-utility-on-chain/src/Andamio/Utility/Redeemers.hs **Hash**: cf38fdfe4b8300be0be4eed87457a223f8afdd0768cb2bd72016a01a33c2c3a2
Filename: ./andamio-utility-on-chain/src/Andamio/Utility/BuiltinByteString.hs **Hash**: ca6339681c2b247590f8050e38f0a058e3eb60b0f25f14527dd852cd8eab689b
Filename: ./andamio-utility-on-chain/src/Andamio/Utility/Value.hs **Hash**: 31ee133e366ff3cdc32b39ccda71be3ae58c36e746fe7aaf9677e169a78c7ccf
Filename: ./andamio-utility-on-chain/src/Andamio/Utility/Validators/TokenNamePolicy.hs **Hash**: 4ef5401d173cfbfe146ed688f570ae6aaf9dea43d52d863db0173afd32774663
Filename: ./andamio-utility-on-chain/src/Andamio/Utility/Validators/AlwaysTrue.hs **Hash**: 8b4feaffce82daab3cd0328db844ffcde27aef8b2324a65ef5862120a3c44ecd
Filename: ./andamio-utility-on-chain/src/Andamio/Utility/Validators/NFTPolicyTnStrict.hs **Hash**: 5f307ea7fc9342e985420de12c44b2c8ed63d339c9bdb210ed1b5af9df6de1c9

TxPipe advocates for the implementation of multiple independent audits, a publicly accessible bug bounty program, and continuous security auditing and monitoring. Despite the diligent manual

review processes, the potential for errors exists. TxPipe strongly advises seeking multiple independent opinions on critical matters. It is the firm belief of TxPipe that every entity and individual is responsible for conducting their own due diligence and maintaining ongoing security measures.

A.2 Issue Guide

A.2.1 Severity

Severity	Description
Critical	Critical issues highlight exploits, bugs, loss of funds, or other vulnerabilities that prevent the dApp from working as intended. These issues have no workaround.
Major	Major issues highlight exploits, bugs, or other vulnerabilities that cause unexpected transaction failures or may be used to trick general users of the dApp. dApps with Major issues may still be functional.
Minor	Minor issues highlight edge cases where a user can purposefully use the dApp in a non-incentivized way and often lead to a disadvantage for the user.
Info	Info are not issues. These are just pieces of information that are beneficial to the dApp creator. These are not necessarily acted on or have a resolution, they are logged for the completeness of the audit.

A.2.2 Status

Status	Description
Resolved	Issues that have been **fixed** by the **project** team.
Acknowledged	Issues that have been **acknowledged** or **partially fixed** by the **project** team. Projects can decide to not **fix** issues for whatever reason.
Identified	Issues that have been **identified** by the **audit** team. These are waiting for a response from the **project** team.

A.3 Revisions

This report was created using a git based workflow. All changes are tracked in a github repo and the report is produced using typst. The report source is available here. All versions with downloadable PDFs can be found on the releases page.

A.4 About Us

TxPipe is a blockchain technology company responsible for many projects that are now a critical part of the Cardano ecosystem. Our team built Oura, Scrolls, Pallas, Demeter, and we're the original home of Aiken. We're passionate about making tools that make it easier to build on Cardano. We believe that blockchain adoption can be accelerated by improving developer experience. We develop blockchain tools, leveraging the open-source community and its methodologies.

A.4.1 Links

- Website
- Email
- Twitter