Audit Report

August 3, 2024

Big Blymp - Yamfore

Contents

1 - Summary

This report provides a comprehensive audit of Yamfore, a decentralized non-custodial lending protocol on Cardano that offers stablecoins loans backed by ADA.

The investigation spanned several potential vulnerabilities, including scenarios where attackers might exploit the validator to lock up or steal funds.

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 Yamfore protocol offers the possibility of taking loans of a fixed asset, e.g. a USD-based stablecoin, by locking ADA as collateral. It also allows for providing liquidity for the loans in exchange for ADA or CBLP tokens, the protocol's main asset.

The protocol is comprised of three kind of Vault UTxOs, one for each asset managed by the protocol: ADA, CBLP and the loaned asset, called USD hereafter. When users take loans, User Positions UTxOs are created to hold the ADA collateral. All these UTxOs are held under the same script address that corresponds to the Main validator, which controls the spending of all kinds of Vaults and User Positions.

Users can borrow USD from USD Vaults. When getting a loan, users must pay fees in CBLP tokens. The fee and the collateral amounts depend on the size of the loan, governance parameters and USB/CBLP price data that is obtained from oracles. This operation results in the creation of a User Position, which holds the ADA collateral and information about the loan.

When repaying a loan, users must pay back into USD Vaults the USD loan plus an interest that depends on the size of the loan, its duration, governance parameters and price data obtained from oracles.

Apart from borrowing and repaying, users can exchange USD for CBLP or ADA, by consuming CBLP or ADA Vaults and creating USD Vaults. These USD Vaults will be available for loans.

A single Governance UTxO provides global settings such as the interest rate and the fees. Another single UTxO, the Price Feed Pointer, provides the hash for the Price Feed withdraw script that validates the USB/CBLP prices given by the oracles. A privileged Admin actor controls the initialization and update of both UTxOs.

For the Price Feed, Yamfore provides a default implementation that reads the price information for USD and CBLP from two oracle UTxOs.

1.b - Process

Our audit process involved a thorough examination of Yamfore 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 unauthorized asset addition, hidden market creation, and disruptions to interoperability with other Plutus scripts. This also included the common vulnerabilities such as double satisfaction and minting policy vulnerabilities.

The audit took place over a period of several weeks, and it involved the evaluation of the protocol's mathematical model to verify that the implemented equations matched the expected behavior.

Findings and feedback from the audit were communicated regularly to the Yamfore team through Discord. Diagrams illustrating the necessary transaction structure for proper interaction with the protocol are attached as part of this report. The Yamfore 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 - User Positions

A user position UTxOs holds the collateral for a loan. It is created when the user borrows and removed when he repays the loan.

- Address: Main validator script hash
- Value
  - N ADA (collateral)
  - 1 user validity
- Datum: `Position(PositionP)` where `PositionP` is `{ created_at: Int, borrow_amt: Int, interest_rate: (Int, Int) }`

2.a.b - USD Vaults

A USD vault contains USD tokens that can be borrowed. USD vaults can be created by users than want to offer USD for loans.

- Address: Main validator script hash
- Value
  - N USD (one of the stablecoins accepted by the protocol)
- Datum: `UsdVault`

2.a.c - CBLP Vaults

CBLP vaults contain CBLP tokens. Users can exchange their USD for CBLP by consuming these vaults.

- Address: Main validator script hash
- Value
  - N CBLP
- Datum: `CblpVault(Int)`

2.a.d - ADA Vaults

ADA vaults contain ADA tokens. Users can exchange their USD for ADA by consuming these vaults.

- Address: Main validator script hash
- Value:
  - N ADA
- Datum: `AdaVault`

2.a.e - Governance UTxO

The governance UTxO contains global parameters used in the protocol.

- Address: Gov validator script hash.
- Value
  - 1 gov validity
- Datum: `GovDat`:
  - `interest_rate: (Int, Int)`
  - `cblp_fee: (Int, Int)`
  - `exchange_reward: (Int, Int)`

2.a.f - Price Feed Pointer UTxO

The Price Feed Pointer UTxO indicates the withdrawal script used to validate the price feed.

- Address: PFP validator script hash.
- Value
  - 1 PFP validity
- Datum: `PfpDat`: `Credential`

2.b - Assets

2.b.a - USD

USD is a shorthand for a stablecoin accepted by the protocol, that a user can borrow by locking ADA as collateral. It is an external token.

- Policy ID: defined with parameter `usd_hash` in Main validator
- Token name: `tokens.usd()`

2.b.b - CBLP

Users must pay in CBLP tokens the fees of taking a loan. It is an external token.

- Policy ID: defined with parameter `cblp_hash` in Main validator
- Token name: `tokens.cblp()`

2.b.c - User validity

Identifies a user Position UTxO.

- Policy ID: hash of Main validator
- Token name: `tokens.user_validity(tag)` where tag is the hash of a seed UTxO.

2.b.d - User auth

Related to a user validity token via the tag. This is held by the user who created the position.

- Policy ID: hash of Main validator
- Token name: `tokens.user_auth(tag)` where tag is the hash of a seed UTxO

2.b.e - Main auth

Meant to be held by the admin as well. Allows delegating the locked ADA.

- Policy ID: hash of Main validator
- Token name: `tokens.admin()`

2.b.f - Gov validity

Identifies the Gov UTxO.

- Policy ID: hash of Gov validator
- Token name: `tokens.gov_validity()`

2.b.g - Gov auth

Allows the holder (menat to be the admin) to update the Gov UTxO datum.

- Policy ID: hash of Gov validator
- Token name: `tokens.gov_auth()`

2.b.g.a - PFP validity

Identifies the PFP UTxO.

- Policy ID: hash of PFP validator
- Token name: `tokens.pfp_validity()`

2.b.h - PFP auth

The agent holding this token has the privilege to perform operations over the PFP UTxO. Meant to be held by the admin.

- Policy ID: hash of PFP validator
- Token name: `tokens.pfp_auth()`

2.b.i - USD oracle token

Identifies an oracle UTxO that holds a valid price of the stablecoin in ADA.

- Policy ID: defined with parameter `usd_ora_hash` in PF (Price Feed) validator
- Token name: defined with parameter `usd_ora_name` in PF validator

2.b.j - CBLP oracle token

Identifies an oracle UTxO that holds a valid price of CBLP in ADA.

- Policy ID: defined with parameter `cblp_ora_hash` in PF validator
- Token name: defined with parameter `cblp_ora_name` in PF validator

2.c - Transactions

2.c.a - Users

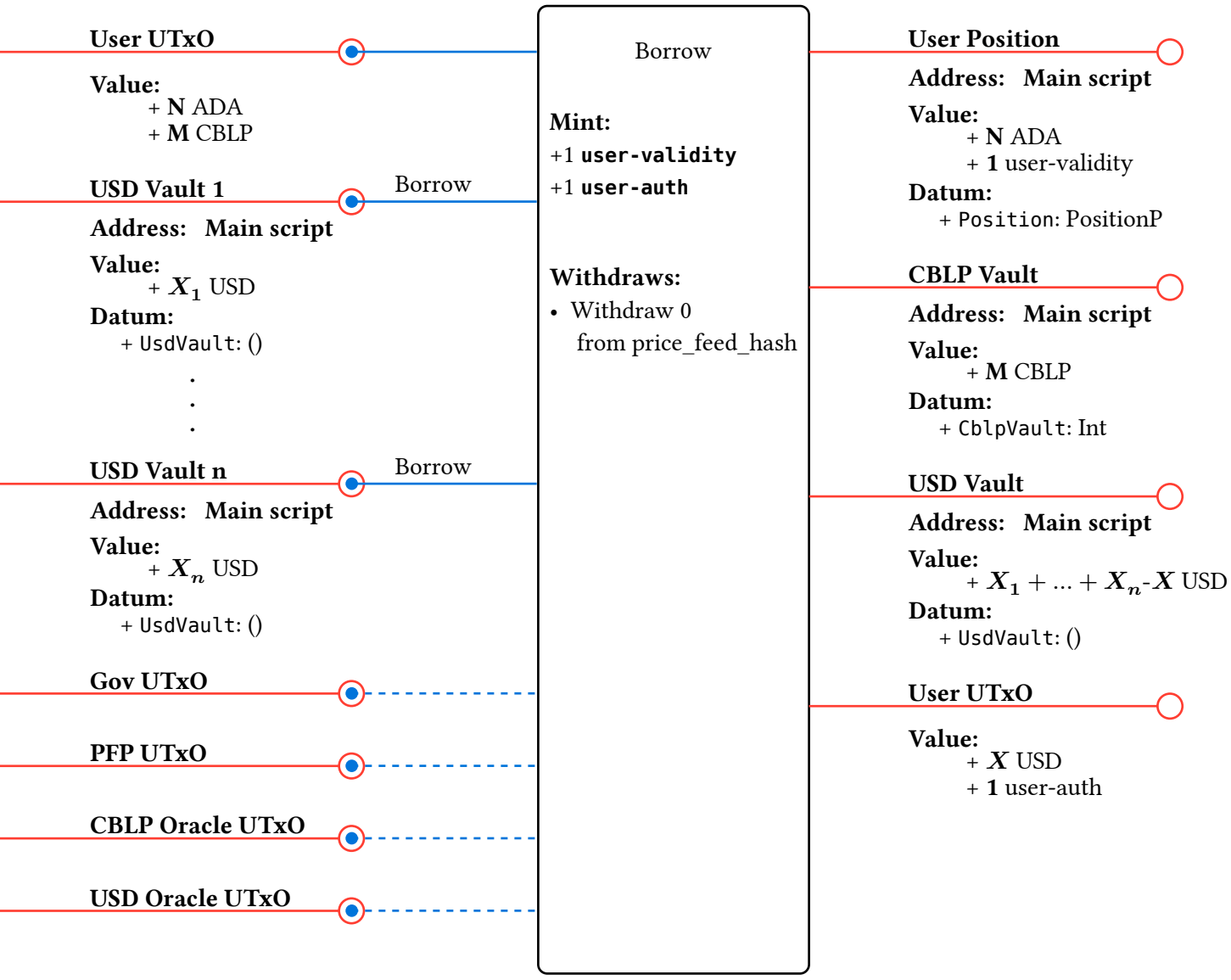
2.c.a.a - Borrow

A user borrows USD by locking ADA as collateral. The CBLPs paid in concept of fees are used to create a CBLP Vault.

There could be more than one USD Vault in the inputs for satisfying the amount of USD that the user wants to borrow. If there is a USD remainder, it is paid to a fresh vault.

Involved redeemers:

- `Borrow`: for spending a USD Vault
- `Mint`: for minting user auth and user validity tokens
- `Pf2Red`: for withdrawing Price Feed script



Note: PositionP = { created_at: Int, borrow_amt: Int, interest_rate: (Int, Int) }

Figure 1: Borrow transaction

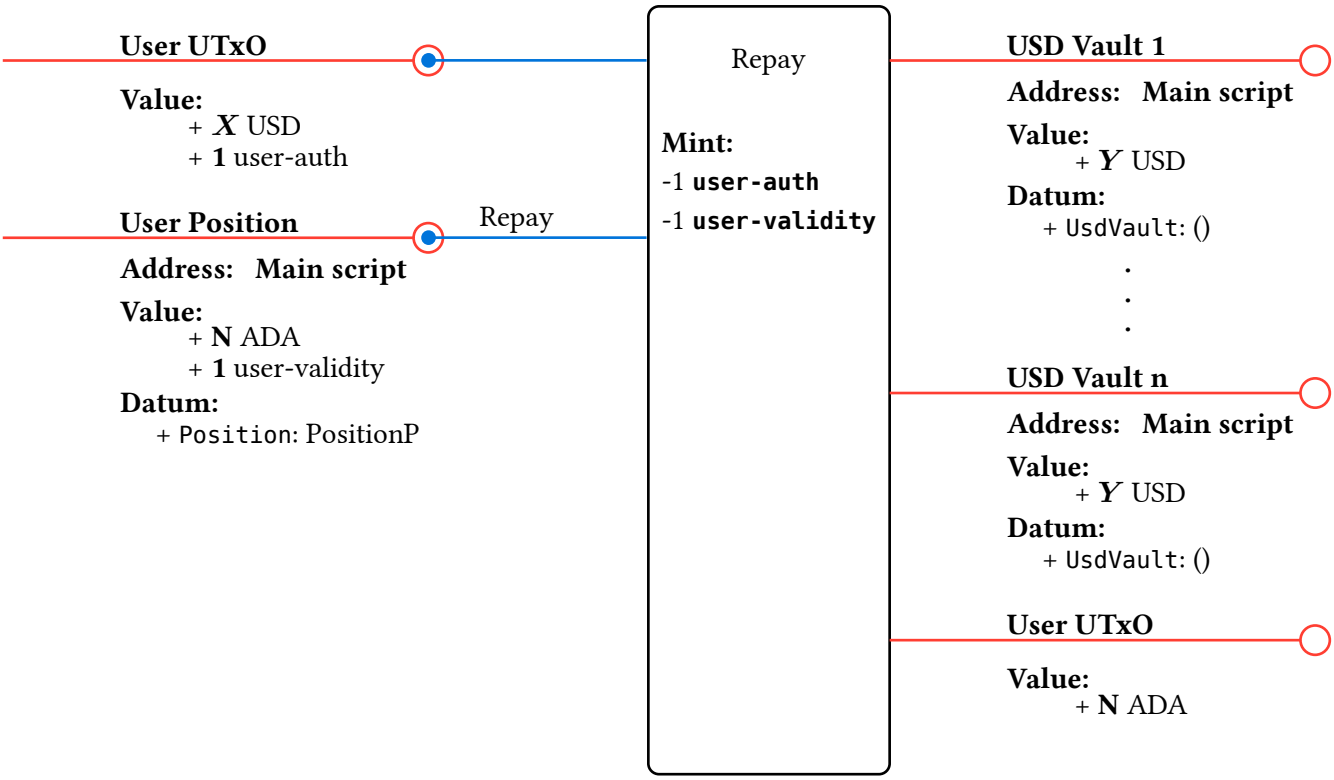
2.c.a.b - Repay

User pays back borrowed USD and gets back its locked ADA.

The repaid USD can be used to create the desired amount of USD Vaults (up to 40 aprox).

Involved redeemers:

- `Repay`: for spending a User Position
- `Burn`: for burning user auth and user validity tokens



Note: PositionP = { created_at: Int, borrow_amt: Int, interest_rate: (Int, Int) }

Y = X / n

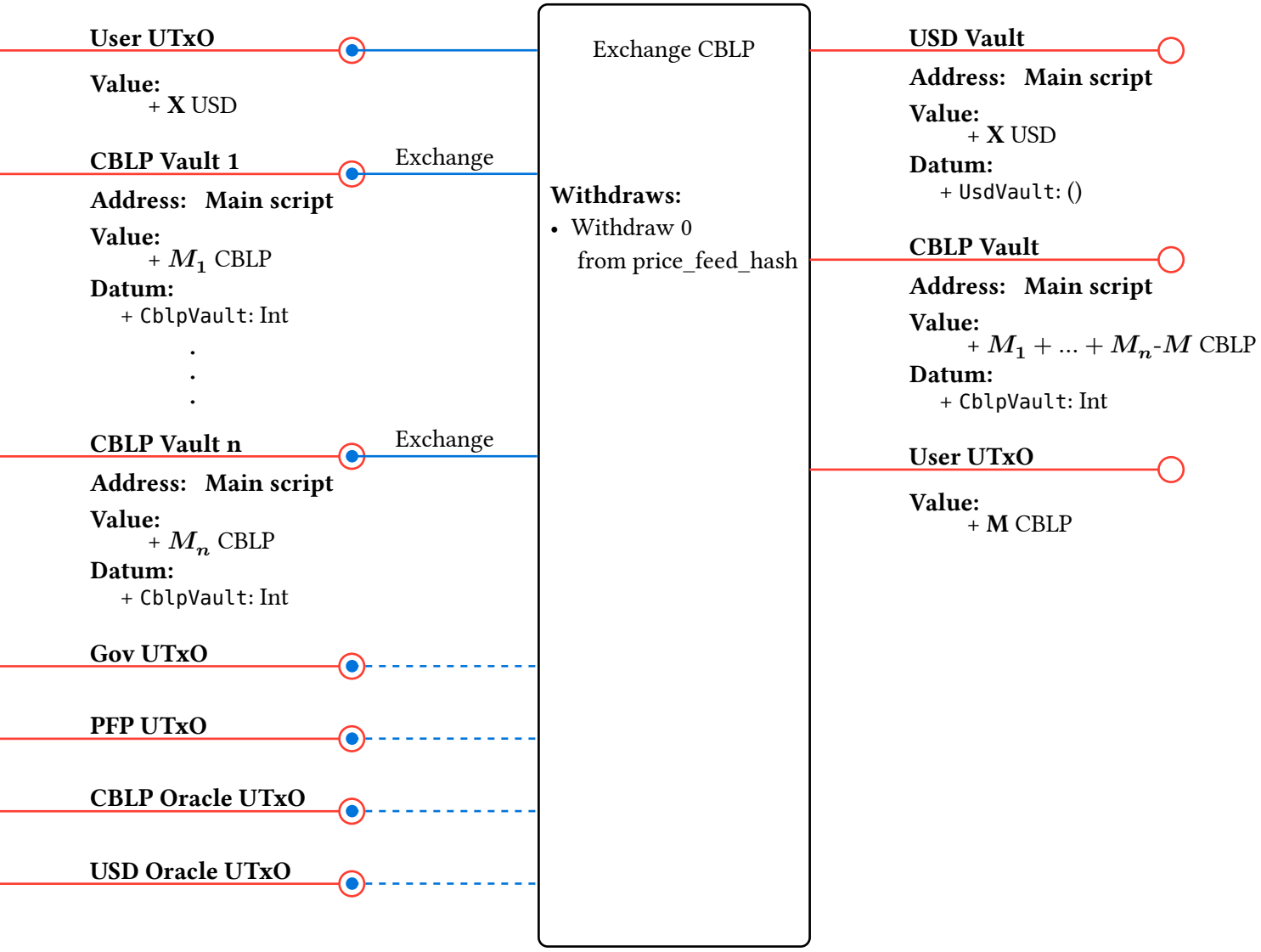
Figure 2: Repay transaction

2.c.a.c - Exchange CBLP

User exchanges USD for CBLP. CBLP can come from more than one CBLP Vault. The USD given by the user are used to create a brand new USD Vault.

Involved redeemers:

- `Exchange`: for spending a CBLP Vault
- `Pf2Red`: for withdrawing Price Feed script



Note: M = corresponding CBLP amount in exchange of X USD, based on Price Feed

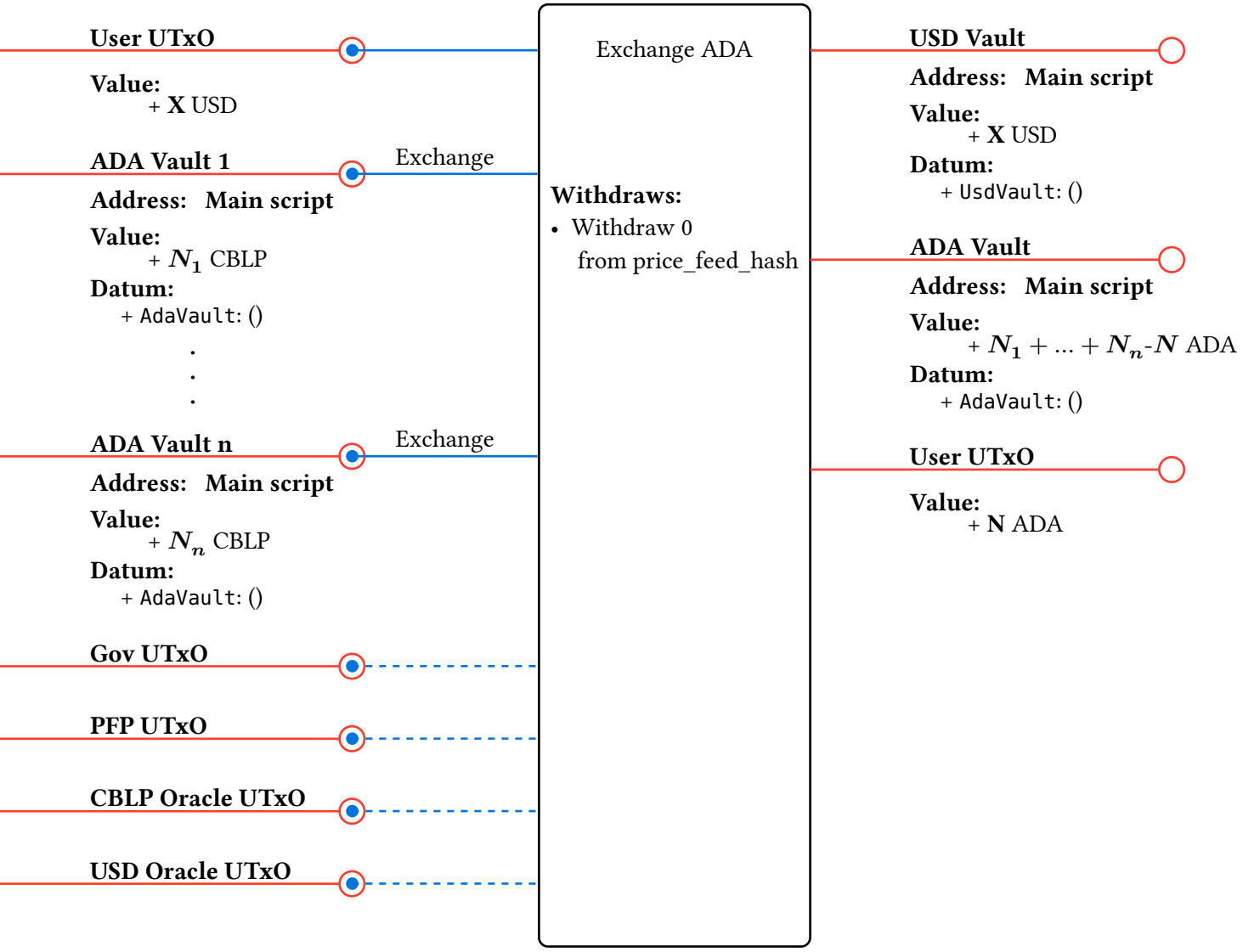
Figure 3: Exchange CBLP transaction

2.c.a.d - Exchange ADA

Analogous with Exchange CBLP, the user gives USD in exchange of ADA. ADA Vaults are involved instead of CBLP Vaults, which are spent with Exchange redeemer as well.

Involved redeemers:

- `Exchange`: for spending a ADA Vault
- `Pf2Red`: for withdrawing Price Feed script



Note: N = corresponding ADA amount in exchange of X USD, based on Price Feed

Figure 4: Exchange ADA transaction

2.c.a.e - Withdraw Main

A user performs a withdrawal on the Main script. The rewards must be used to create a ADA Vault UTxO.

Involved redeemers:

- `WithdrawFrom`: for managing Main script stake rewards

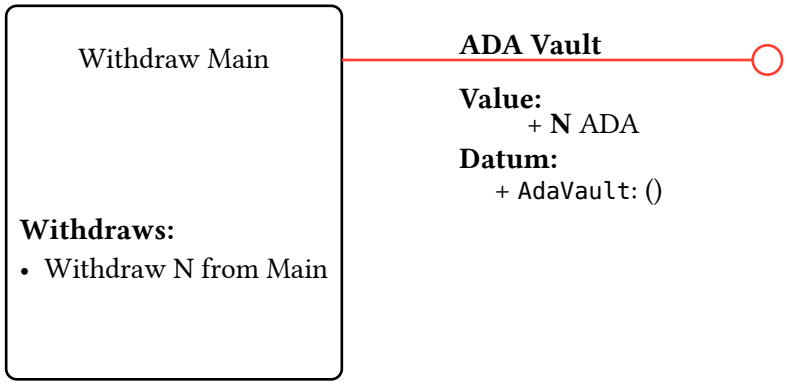


Figure 5: Withdraw Main transaction

2.c.b - Admin

2.c.b.a - Mint Main Auth token

Admin mints and receives the Main Auth token.

Involved redeemers:

- `AdminMint`: for minting the Main auth token

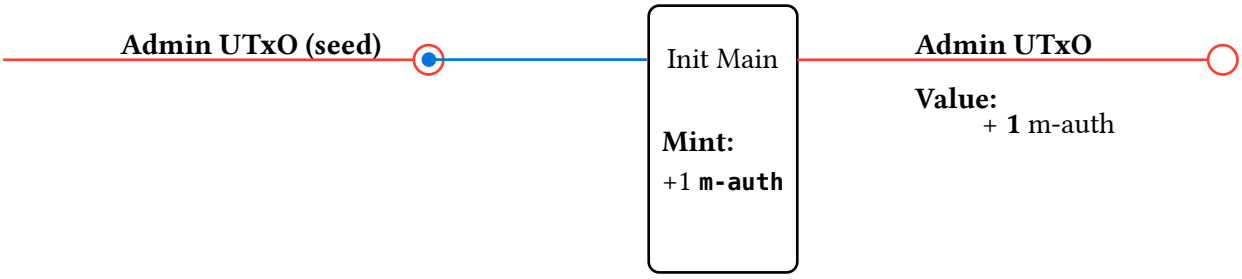


Figure 6: Mint Main Auth token transaction

2.c.b.b - Publish Main

Admin delegates the stake of the Main script address.

Involved redeemers:

- `Publish`: for managing the script certificate

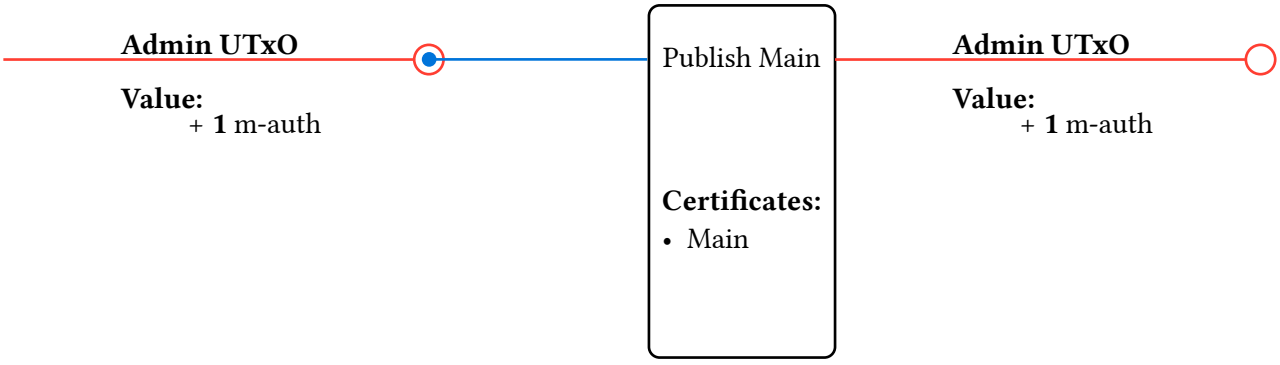


Figure 7: Publish Main transaction

2.c.b.c - Burn Main Auth token

Admin burns the Main auth token.

Involved redeemers:

- `AdminBurn`: for burning the Main auth token

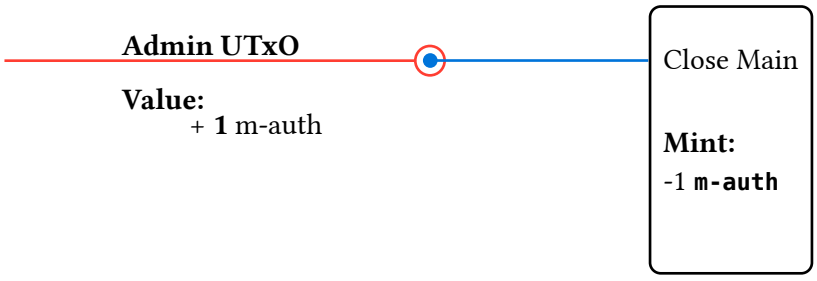


Figure 8: Burn Main Auth token transaction

2.c.c - Governance

2.c.c.a - Init Gov

Admin initializes the Governance UTxO. In the datum is stored some global parameters from the protocol, like the fees in CBLP. Is identified by a validity token.

Involved redeemers:

- `Gov2Init`: for minting both Gov validity and auth tokens

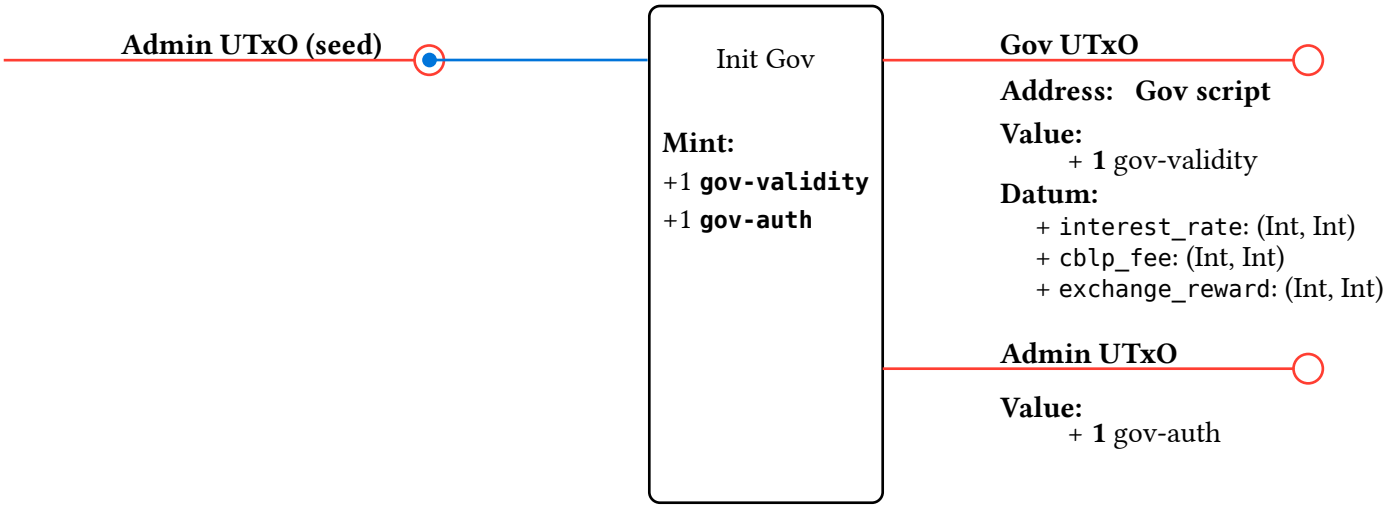


Figure 9: Init Gov transaction

2.c.c.b - Update Gov

Admin updates the Governance parameters stored in the Gov UTxO datum.

Involved redeemers:

- `Gov3Update`: for spending the Gov UTxO

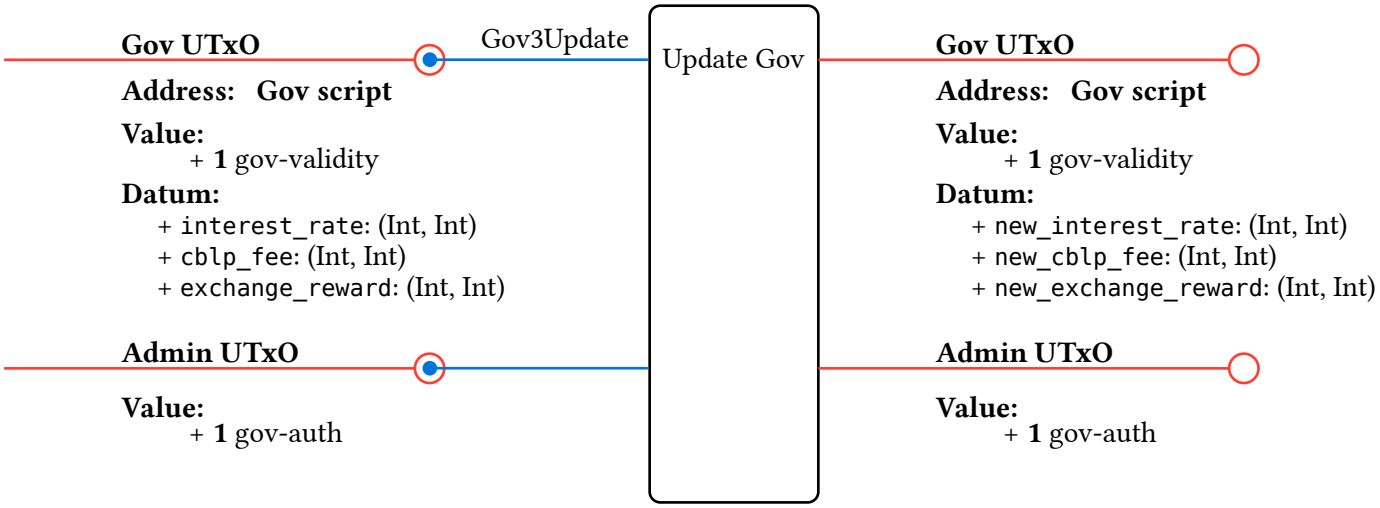


Figure 10: Update Gov transaction

2.c.c.c - Close Gov

Admin closes the Gov UTxO by burning both validity and auth tokens.

Involved redeemers:

- `Gov3Close`: for spending the Gov UTxO
- `Gov2Burn`: for burning both Gov validity and auth tokens

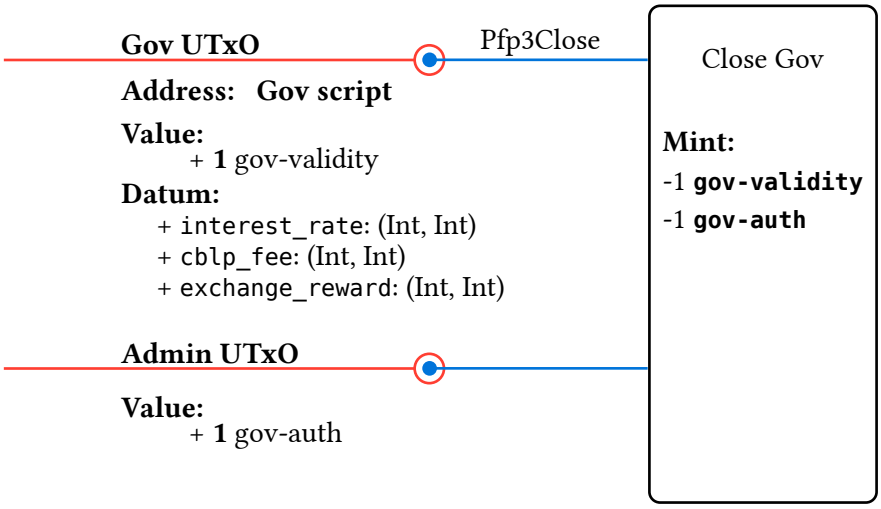


Figure 11: Close Gov transaction

2.c.d - Price Feed Pointer

2.c.d.a - Init PFP

Admin initializes the Price Feed Pointer (PFP), which will hold the current Price Feed (PF) script hash in its datum. The PFP UTxO will also hold a token to identify it (validity token) that is twinned with an auth token held by the admin.

Involved redeemers:

- `Pfp2Init`: minting of PFP validity and auth tokens

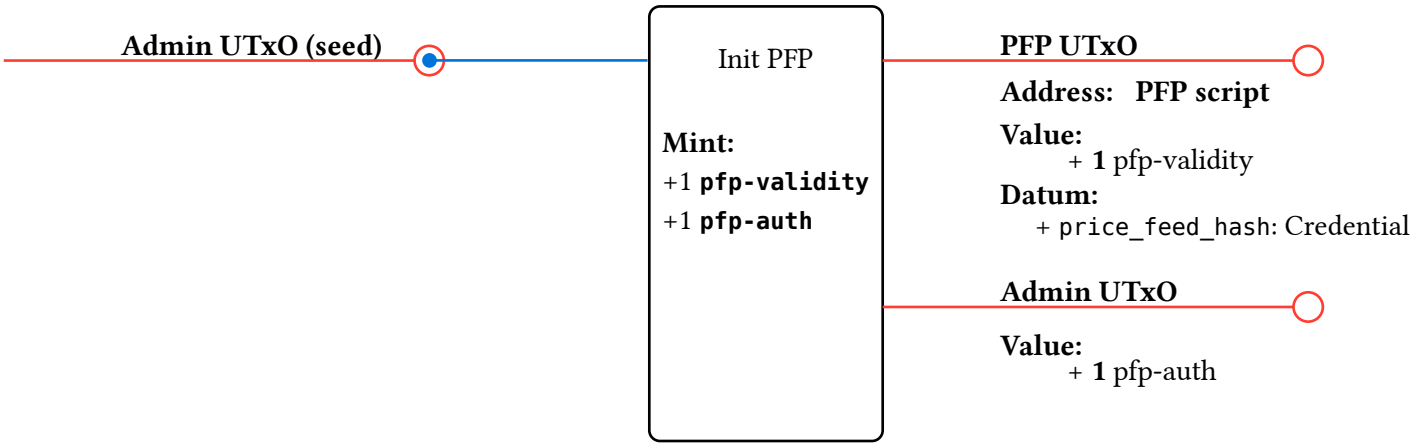


Figure 12: Init PFP transaction

2.c.d.b - Update PFP

Admin updates the Price Feed (PF) script hash from the PFP UTxO datum.

Involved redeemers:

- `Pfp3Update`: for spending the PFP UTxO

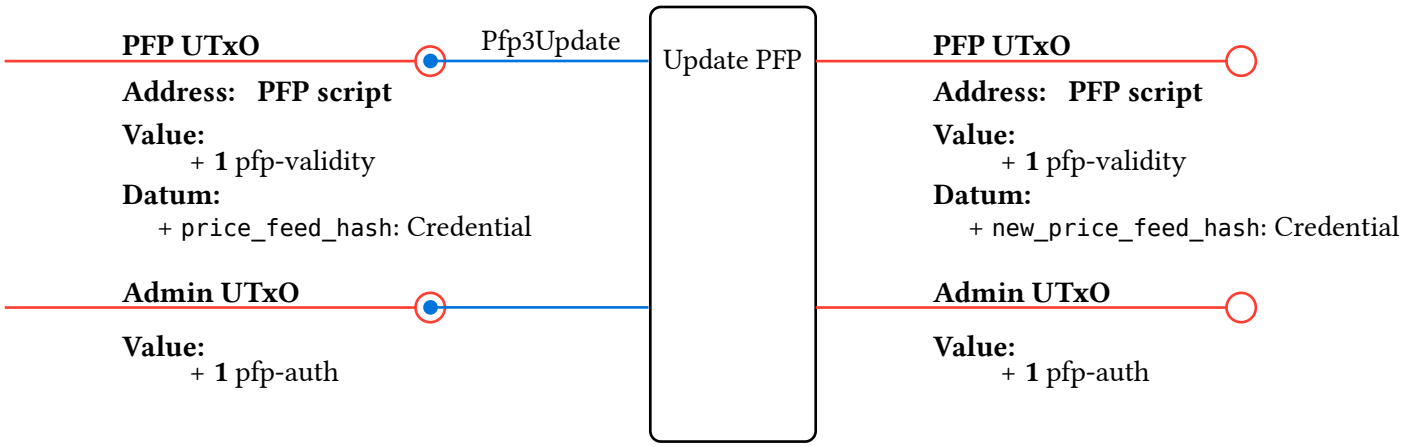


Figure 13: Update PFP transaction

2.c.d.c - Close PFP

Admin closes the PFP UTxO by burning both validity and auth PFP tokens.

Involved redeemers:

- `Pfp3Close`: for spending the PFP UTxO
- `Pfp2Burn`: for burning both PFP validity and auth tokens

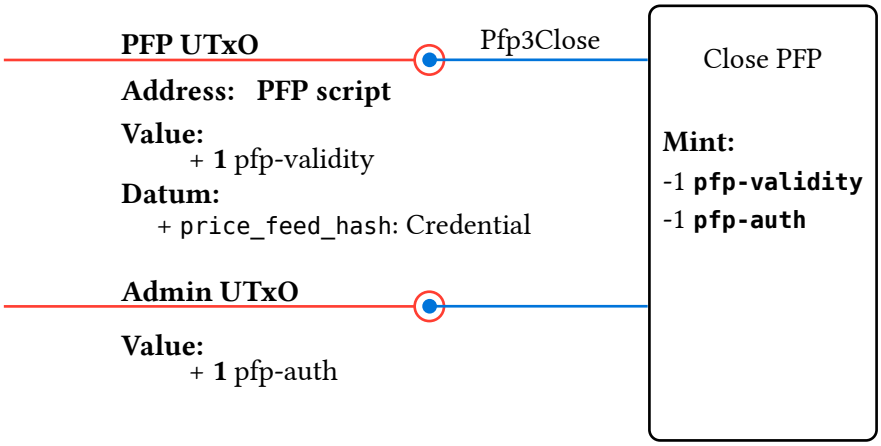


Figure 14: Close PFP transaction

3 - Audited Files

Below is a list of all files audited 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 `9671f5ae727ba87f2f0e2100e16993e7c7a68eea`.

Filename
./aik/validators/m.ak
./aik/validators/gov.ak
./aik/validators/pf.ak
./aik/validators/pfp.ak
./aik/lib/yamfore/tokens.ak
./aik/lib/yamfore/functions.ak
./aik/lib/yamfore/utils.ak
./aik/lib/yamfore/types.ak
./aik/lib/yamfore/constants.ak

4 - Findings

ID	Title	Severity	Status
YAM-001	USD can be stolen in Borrow transaction	Critical	Resolved
YAM-002	Exchange transaction: User can control staking key of output vaults	Critical	Resolved
YAM-101	Exchange transaction: Users can collect too many ADA vaults, hurting vault availability	Major	Resolved
YAM-301	Simpler check for `created_at`	Info	Resolved
YAM-302	Exchange transaction: Inefficient check for admin token	Info	Acknowledged
YAM-303	Exchange transaction: Incentive for users to collect too many CBLP vaults, hurting vault availability	Info	Resolved
YAM-304	Unneeded validity range restriction in Borrow and Exchange	Info	Resolved
YAM-305	WithdrawFrom: Risk of double satisfaction attack	Info	Acknowledged
YAM-306	No sanity checks for governance UTxO datum	Info	Acknowledged
YAM-307	No sanity checks for price feeds	Info	Acknowledged

4.a - YAM-001 USD can be stolen in Borrow transaction

Category	Commit	Severity	Status
Vulnerability	fedd8e020b25d4e2270a89df19fac1336a100c4b	Critical	Resolved

4.a.a - Description

USD can be stolen in Borrow transactions as it is not checked that USD leftovers are all paid to a new USD vault.

4.a.b - Recommendation

When there's a remainder of USD in the inputs for the amount borrowed by the user, assert the following:

```
usd_output_amt >= usd_input_amt - borrow_amt
```

4.a.c - Resolution

Resolved in PR #5

4.b - YAM-002 Exchange transaction: User can control staking key of output vaults

Category	Commit	Severity	Status
Vulnerability	fedd8e020b25d4e2270a89df19fac1336a100c4b	Critical	Resolved

4.b.a - Description

For the vault outputs (created USD, leftover CBLP/ADA), there is no check that the staking part of the address is the own hash. The only check about the staking part is that it must be equal to that of the 0th own spend, but the 0th input could be fabricated by the user to contain any staking key he wants. This way, a user can consume several ADA vaults and create a single big leftover ADA vault with a staking key under his control.

4.b.b - Recommendation

Check that all vault outputs are paid to an address with staking part equal to the own script hash.

4.b.c - Resolution

Resolved in PR #6

4.c - YAM-101 Exchange transaction: Users can collect too many ADA vaults, hurting vault availability

Category	Commit	Severity	Status
Functionality	fedd8e020b25d4e2270a89df19fac1336a100c4b	Major	Resolved

4.c.a - Description

The exchange operation allows users to exchange USD for ADA tokens. To do this, ADA vaults are consumed and USD vaults are created. If there are ADA leftovers, an ADA vault is created for them.

In current version, the validator doesn't have any enabled check about the maximum amount of ADA leftovers (only a commented check is present in line 318). The implication is that users can spend as many ADA vaults as they want. In the extreme case, if the transaction is within protocol limits, they can spend all available ADA vaults, collapsing them into a single one. This hurts ADA vaults availability, limiting the amount of concurrent users the protocol can have.

4.c.b - Recommendation

A check that limits the amount of ADA leftovers is necessary.

The commented check in line 318 can be enabled to solve the worst case scenario of consuming all ADA Vaults. Here, the leftover amount is limited by the max size of the input vaults. However, it might not be sufficient to prevent more vault collapsing than needed.

A stronger check such as the one used in the borrow operation could be used (see line 482). In this case, the amount of leftovers is limited by the average size of the input vaults, instead of the max.

4.c.c - Resolution

Resolved in commit `012b058`

4.d - YAM-301 Simpler check for `created_at`

Category	Commit	Severity	Status
Simplification	fedd8e020b25d4e2270a89df19fac1336a100c4b	Info	Resolved

4.d.a - Description

It is checked that it is at distance < 10000 from `lb`.

4.d.b - Recommendation

Just check that it is equal to `lb`.

4.d.c - Resolution

Resolved in PR #15

4.e - YAM-302 Exchange transaction: Inefficient check for admin token

Category	Commit	Severity	Status
Optimization	fedd8e020b25d4e2270a89df19fac1336a100c4b	Info	Acknowledged

4.e.a - Description

The exchange transaction can spend several CBLP vaults. For each spent vault, it is checked that the time lock is finished or the admin token is present (see `m.ak:L224`). To check for the admin token all the inputs are traversed, and this is done repeatedly, one time for each vault. However, it is unnecessary to repeat each time the search for the admin token, as the result is always the same.

4.e.b - Recommendation

Check for the presence of the admin token only once and save the result in a boolean variable.

4.e.c - Resolution

The project team decided not to resolve this finding. The optimization only applies to an admin action that is expected to be extremely uncommon.

4.f - YAM-303 Exchange transaction: Incentive for users to collect too many CBLP vaults, hurting vault availability

Category	Commit	Severity	Status
Functionality	fedd8e020b25d4e2270a89df19fac1336a100c4b	Info	Resolved

4.f.a - Description

The exchange operation allows users to exchange USD for CBLP tokens. To do this, CBLP vaults are consumed and USD vaults are created. If there are CBLP leftovers, a CBLP vault is created for them.

In current version, it is checked that the leftover amount must be smaller than the amount contained in the biggest input vault. This check is not strong enough to prevent unnecessary spending of CBLP vaults. Users can be interested in spending many CBLP vaults and collapse them into a single one so they can take the remains of the collected min ADA. At the same time, they will be hurting vault availability.

4.f.b - Recommendation

A stronger check such as the one used in the borrow operation could be used (see line 482). In this case, the amount of leftovers is limited by the average size of the input vaults, instead of the max.

4.f.c - Resolution

Resolved in commit `9eef8e6`

4.g - YAM-304 Unneeded validity range restriction in Borrow and Exchange

Category	Commit	Severity	Status
Functionality	fedd8e020b25d4e2270a89df19fac1336a100c4b	Info	Resolved

4.g.a - Description

Both Borrow and Exchange transactions have a validity range restriction of 20 minutes, which is unnecessary. Even more, this could hurt protocol responsiveness in case of network congestion (this applies to Repay too).

4.g.b - Recommendation

Relax the upper bound restriction on Borrow and Exchange. It could be a half-open interval without any vulnerability. For Repay can be considered an extension of the 20 minutes interval to one of several hours for cases of network congestion.

4.g.c - Resolution

Resolved in commit `9eef8e6`

4.h - YAM-305 WithdrawFrom: Risk of double satisfaction attack

Category	Commit	Severity	Status
Vulnerability	fedd8e020b25d4e2270a89df19fac1336a100c4b	Info	Acknowledged

4.h.a - Description

In the WithdrawFrom purpose of the main validator, there is risk for a double satisfaction attack if the ADA vault output is used also to validate some other operation that has an ADA vault as its first output.

Fortunately, current operations Borrow, Repay and Exchange doesn't have an ADA vault as first output, so the attack is not possible. But if Yamfore later introduces an operation that admits an ADA vault as first output, the attack will be inadvertently enabled.

4.h.b - Recommendation

As the attack is not possible in current version, the recommendation is to take measures to prevent involuntarily enabling it in the future.

4.h.c - Resolution

The project team acknowledged the finding and documented it as an "implicit global constraint".

4.i - YAM-306 No sanity checks for governance UTxO datum

Category	Commit	Severity	Status
Robustness	fedd8e020b25d4e2270a89df19fac1336a100c4b	Info	Acknowledged

4.i.a - Description

The governance UTxO provides three important parameters for the protocol: `interest_rate`, `cblp_fee` and `exchange_reward`.

Currently, no checks are done on the values when the UTxO is created or when it is updated. If, by accident or on purpose, some of these values are incorrectly updated, the protocol can be temporally disabled or fatally vulnerable. As the parameters are rational numbers used as rates and fees, zero or negative values must be avoided.

4.i.b - Recommendation

Add sanity checks for all the datum fields both in the creation and in the update of the Governance UTxO.

For `interest_rate` it can be checked that both the numerator and the denominator are > 0.

For `cblp_fee` it can be checked that the numerator is >= 0 and the denominator is > 0.

For `exchange_reward` it can be checked that both the numerator and the denominator are > 0. It may also be checked that the rational number is >= 1 (numerator >= denominator).

4.i.c - Resolution

The project team decided not to resolve this finding. The recommended validation is not enough to guarantee that harmful values will not be used. It is still responsibility of the admin to use reasonable values for the parameters.

4.j - YAM-307 No sanity checks for price feeds

Category	Commit	Severity	Status
Robustness	fedd8e020b25d4e2270a89df19fac1336a100c4b	Info	Acknowledged

4.j.a - Description

No validation is done in the main validator about the price values provided by the oracles. Instead, the validity check of the oracle information is delegated to the Price Feed withdraw validator. This validator is determined by the Price Feed Pointer UTxO and can be modified by the admin of the protocol.

The price information is highly sensitive and incorrect values can be fatal for the entire protocol. All possible layers of validation of these values are desired. In particular, in the main validator only a basic sanity check can be done to ensure that the provided values are > 0.

4.j.b - Recommendation

Every time the price feeds are used, add checks to ensure that for both prices the numerator and the denominator are > 0. These checks can be added to the `get_pf_red` function that is used to retrieve the price feeds from the corresponding redeemer.

4.j.c - Resolution

The project team decided not to resolve this finding. The recommended validation is not enough to guarantee that harmful values will not be used, such as extremely high or low prices. It is still responsibility of the Price Feed validator to provide values that correctly follow a price resolution protocol.

5 - Minor issues

In this section we list some issues we found that do not qualify as findings such as typos, coding style, naming, documentation, etc. We used the Github issues system to report them.

- CBLP datum comment says “slot” but is “POSIX time” #7
- Unused `n_vaults` function #8
- main minting policy: can return last check instead of True #11
- borrow: redundant check that own datum is UsdVault #12
- rename max_vault_amt to show it refers to usd #13
- token name prefixes: add comment indicating they follow CIP 68 #14

6 - Appendix

6.a - Terms and Conditions of the Commercial Agreement

6.a.a - 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.

6.a.b - 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 Big Blymp 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.

6.a.c - 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: ./aik/validators/m.ak **Hash**: 144a368f0ca55acbbebd097bd606c42f2e49435d6b56c6c2f4e438964bbbca81
Filename: ./aik/validators/gov.ak **Hash**: fde9639a7bc4385117c9a4436ae9c88faf414ec2cfa5c7e35afba088159e8487
Filename: ./aik/validators/pf.ak **Hash**: 4758a023d381e2b52f55e5bc21ea83387b3b43c1ba85ddb20e0359bc41b34a38
Filename: ./aik/validators/pfp.ak **Hash**: b41c4da76543617647d4174f7f366297dfe416eea9c189fc5ee2b96d362e31cf
Filename: ./aik/lib/yamfore/tokens.ak **Hash**: a464a1c0133828eb5668aea0edefd18411e1b3cbb2e64624714665f183a23538
Filename: ./aik/lib/yamfore/functions.ak **Hash**: cd43ef0e01f33f6860977e7a4c4df54fac7e3cc85a587a23990ffbcebfc61dea
Filename: ./aik/lib/yamfore/utils.ak **Hash**: e2412de5ce29b830d1bd44a4ce8f409d3fdf5dbf8329eae6172f6916d521e5b2
Filename: ./aik/lib/yamfore/types.ak **Hash**: 2f4328662aa2623f1aaa7d44601d14c281d39ab43c50b41cefa0cd0dc1025a6b
Filename: ./aik/lib/yamfore/constants.ak **Hash**: 99752231e4ed565e030e1748e6759e499021ad9e9947b68e314e5aead42beeb9

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.

6.b - Issue Guide

6.b.a - 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.

6.b.b - 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.

6.c - 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.

6.d - 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.

6.d.a - Links

- Website
- Email
- Twitter