U T x O C O M P A N Y

AUDIT REPORT

Blink Labs

Nabu VPN

2025-11-19

https://github.com/utxo-company/audit-blink-vpn-contracts

Contents

1 - Summary

Blink Labs engaged UTxO Company to audit the Aiken validators that secure Nabu VPN, a subscription-based VPN service that relies on Cardano for payments. The contract mints a per-user access token, locks it inside the script, and enforces ADA payments according to on-chain plan and region reference data. Our review focused on risks such as unauthorized asset creation, incorrect expiration handling, tampering with pricing information, and premature token recovery. As always, this report documents the issues we found during the engagement, but it cannot guarantee the absence of undiscovered vulnerabilities.

1.a - Overview

Nabu VPN keeps all subscription information on-chain. A bootstrap transaction mints a "provider" NFT, locking the pricing/region reference datum at the VPN validator address. Subsequent signups read the provider NFT via a reference input, validate the user's plan selection, and mint a unique access token whose name is derived from the funding UTxO. That token stays at the contract address alongside the subscriber's datum, which records the owner's payment credential, chosen region, and an expiration stored in milliseconds. Renewals reuse the same token and either add time to the current expiration or restart the clock for expired accounts, along with an optional new owner which allows controlled transfers. Burn logic lets the owner reclaim the token after it expires and grants the provider a delayed backstop burn path. Together, these constraints align the on-chain state with the public pricing advertised by the service and the privacy goals described on nabuvpn.com.

1.b - Process

We performed a manual review of the Aiken code in `validators/vpn.ak`, the helper modules in `lib/`, the minting policy in `validators/nft.ak`, and the repository's accompanying documentation. The audit reconstructed expected flows from the "How It Works" product description, verified that each validator branch matched the project's README specification, and followed the provider's payment paths across example transactions. Particular attention was given to the derivation of token names, validation of reference inputs, expiration arithmetic, and the gating logic that governs burns initiated by either the owner or the provider. Findings were communicated to the team as they were discovered and iterated on with UTxO Company's internal review checklist.

2 - Specification

2.a - UTxOs

2.a.a - VPN Reference Data UTxO

- **Address**: VPN validator script address parameterized by a wait time, a one shot policy id, and the provider's `Address`.
- **Value**: Min ADA and the reference NFT minted under policy `reference_policy_id` which is the one shot policy id.
- **Datum**:

```
type VPNReferenceData {
  pricing: List<Pricing>,
  regions: List<ByteArray>,
}
```

- **Usage**: A reference input for signup and renewal transactions and must either be re-created with identical locking conditions when the provider updates pricing or available regions. It may be spent when burning the NFT which essentially vanishes the UTxO and reference data.

2.a.b - VPN Access UTxO

- **Address**: Same VPN validator script address that holds a unique access token for the customer.
- **Value**: Min ADA and a unique VPN access token minted by the validator.
- **Datum**:

```
type VPNData {
  owner: VerificationKeyHash,
  region: ByteArray,
  expiration_time: Int
}
```

- **Notes**: The access token name is derived from some input in the minting transaction using the BLAKE2b-256 of the input. Renewals "reuse" the VPN access UTxO, by spending it and re-creating it at the VPN validator script address updating only the datum fields.

2.b - Assets

2.b.a - Reference NFT

- **Minting policy**: `validators/nft.ak` enforces single-token minting tied to the provider's bootstrap transaction output reference.
- **Asset name**: Does not matter, can be anything.
- **Role**: Anchors the `VPNReferenceData` datum and must remain at the contract address for the pricing data to remain valid; burning it intentionally retires the reference input.

2.b.b - VPN Access Token

- **Minting policy**: `validators/vpn.ak` has a redeemer `MintVPNAccess` for minting unique tokens per subscription.
- **Properties**: Quantity is limited to 1 per transaction, and the token stays inside the script until explicitly burned.
- **Burn path**: The same validator allows either the owner (after expiry) or the provider (after `wait_time` + expiration) to remove the token via `BurnVPNAccess`.

2.c - Transactions

2.c.a - Setup

The provider bootstraps the system by minting the "provider" reference NFT, locking the initial pricing datum at the VPN script address.

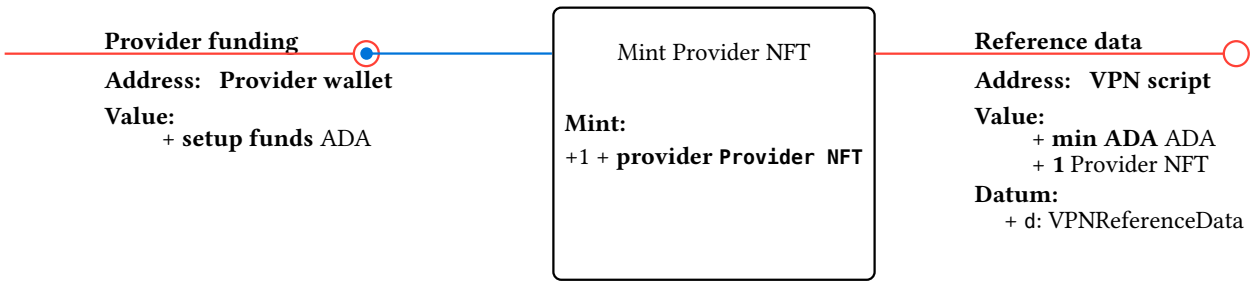

Figure 1: Mint provider NFT transaction

2.c.b - Signup (MintVPNAccess)

When a wallet purchases VPN time, it references the provider's pricing datum and the on-chain reference script, pays the advertised ADA amount, and mints a unique access token whose datum locks the owner, region, and computed expiration inside the contract.

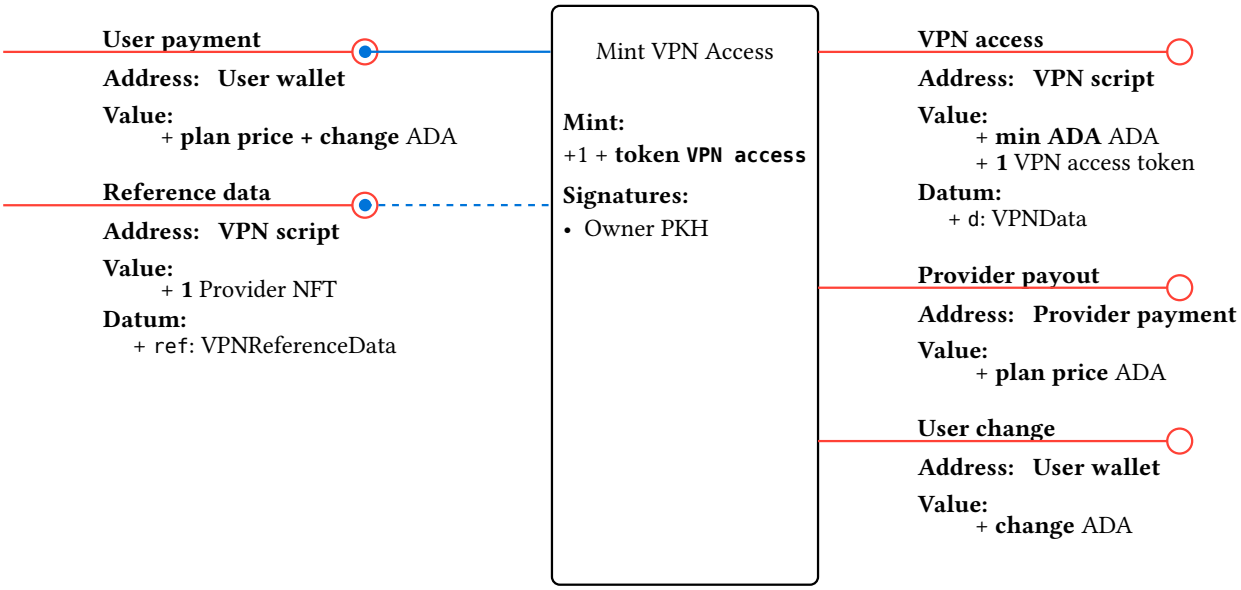

Figure 2: Mint VPN access transaction

2.c.c - Renewal or Transfer (ExtendVPNAccess)

An existing subscription can add time by selecting a pricing tier and pays the provider again. The owner may also authorize a new payment key to take control in the same transaction, while the datum's

expiration is recomputed from either the current expiry or "now" depending on whether the plan had already lapsed. The transaction reuses both the reference datum and the reference script deployed during setup. Passing `selection = -1` lets the owner perform a transfer-only update without sending new ADA to the provider.

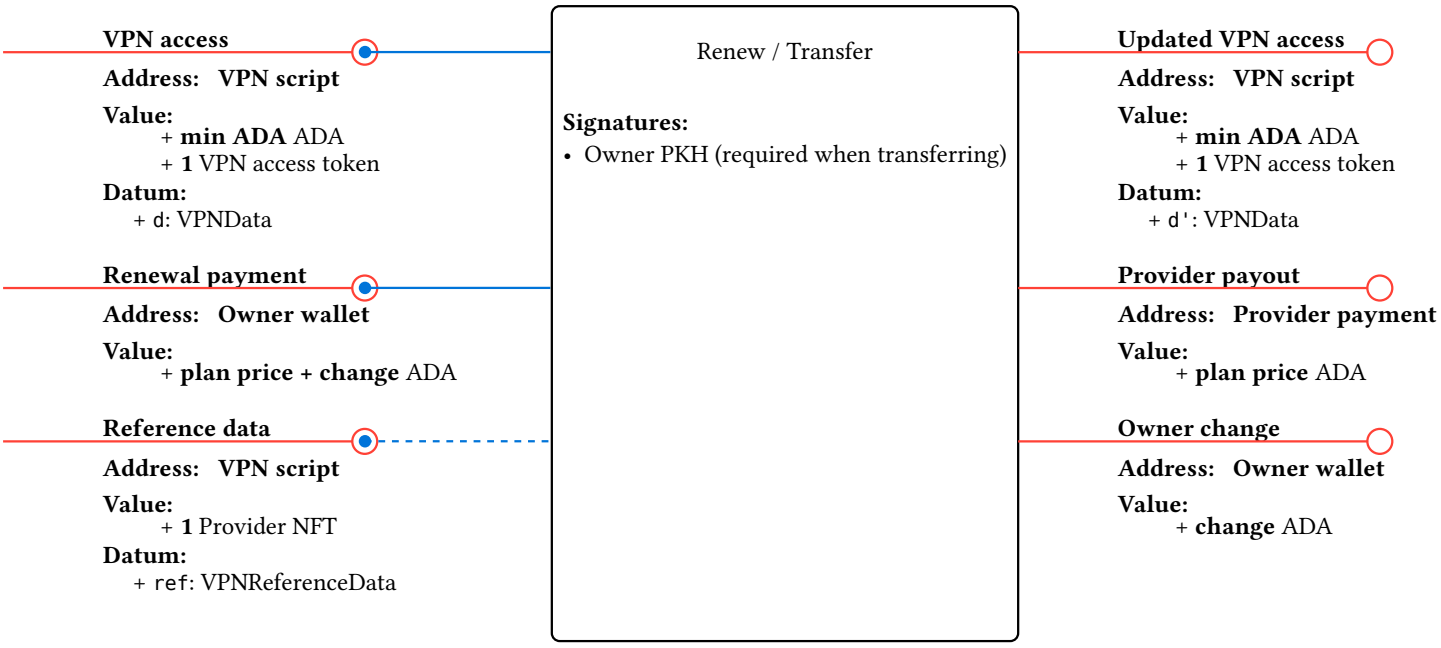

Figure 3: Renew or transfer transaction

2.c.d - Owner Burn (BurnVPNAccess)

Once a plan has expired, the token can be burned to free the script UTxO and reclaim min ADA. The owner may burn it immediately after the deadline.

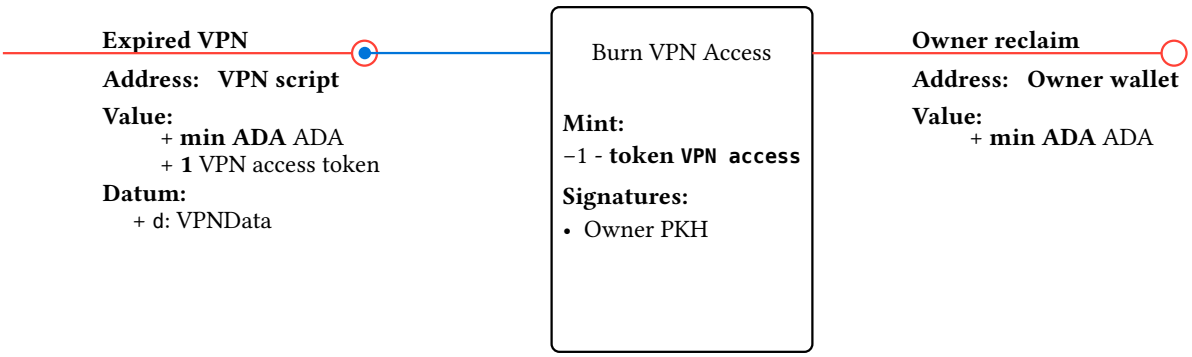

Figure 4: Owner-initiated burn transaction

2.c.e - Provider Burn (BurnVPNAccess)

If the owner does not act, the provider can reclaim the access token after the configured wait time. The provider-signed transaction mirrors the owner burn but routes min-ADA back to any wallet and requires the validity range to start after the extra wait period post expiration.

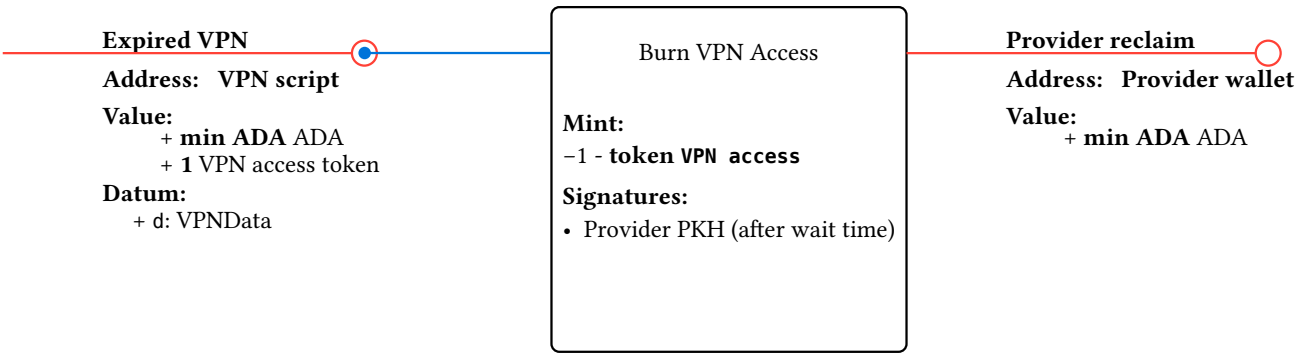

Figure 5: Provider-initiated burn transaction

2.c.f - Reference Data Update (UpdateReferenceData)

The provider can update pricing or regions by spending the reference UTxO and re-locking it with the same NFT. The validator requires the provider's signature and accepts an updated inline datum.

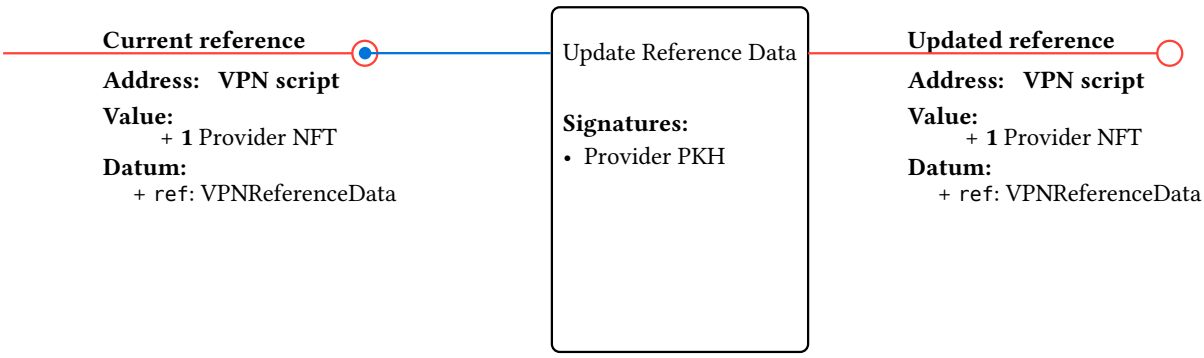

Figure 6: Update reference data transaction

2.c.g - Retire Reference Data (UpdateReferenceData)

When pricing data is no longer needed, the provider can spend the reference UTxO and burn the "provider" NFT, removing the reference input entirely and reclaiming the min-ADA deposit using the same redeemer as the update flow.

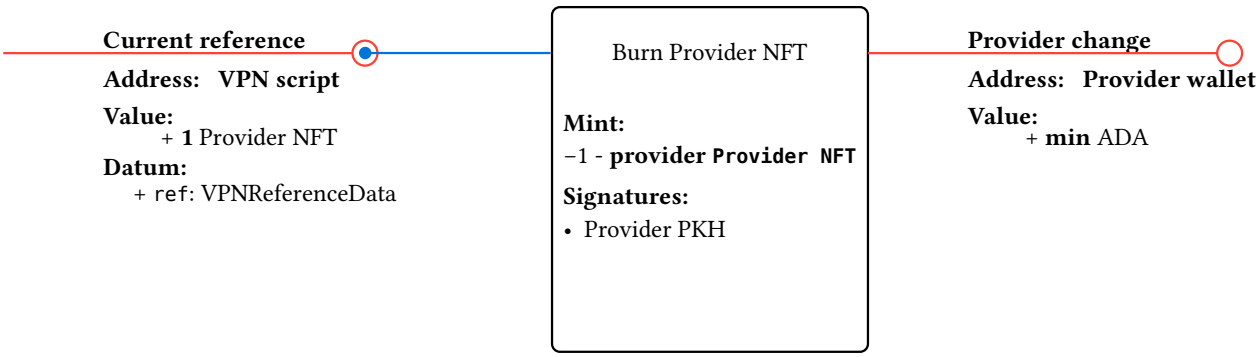

Figure 7: Burn provider NFT transaction

3 - 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 `c3768121415eab902352ecf680efda7f4ea82885`.

Filename
./lib/types.ak
./lib/utilities.ak
./validators/nft.ak
./validators/vpn.ak

4 - Findings

ID	Title	Severity	Status
BLV-001	Extend flow can double-count a single provider payout	Critical	Resolved
BLV-101	Provider burn path captures the user's ADA deposit	Major	Acknowledged
BLV-201	Minted expirations do not account for slot lag	Minor	Acknowledged
BLV-301	Reference configuration should capture provider credentials	Info	Acknowledged
BLV-302	Simplify provider address handling	Info	Resolved
BLV-303	Streamline reference update input destructuring	Info	Resolved
BLV-304	UpdateReferenceData redeemer carries redundant fields	Info	Acknowledged
BLV-305	Reference-NFT burn check relies on implicit mint semantics	Info	Acknowledged
BLV-306	Extend redeemer carries derivable fields	Info	Resolved
BLV-307	Extend path input destructuring can be simplified	Info	Resolved
BLV-308	Burn redeemer carries derivable token name	Info	Resolved
BLV-309	`get_expiration_time_extend` uses boolean guard instead of `expect`	Info	Resolved

BLV-310	`output_reference_to_tn` can hash serialized data directly	Info	Resolved
BLV-311	`only_2_tokens` could validate the exact asset	Info	Resolved
BLV-312	Confusing precedence in `validators/nft.ak`	Info	Acknowledged

5 - BLV-001 Extend flow can double-count a single provider payout

Category	Commit	Severity	Status
Bug	c3768121415eab902352ecf680efda7f4ea82885	Critical	Resolved

5.a - Description

`get_expiration_time_extend` relies on `ada_paid_to` to confirm that the provider received the renewal fee, but `ada_paid_to` only checks that **any** output pays the expected amount. The same provider payout output can therefore satisfy multiple `ExtendVPNAccess` redeemers in one transaction, letting a user extend several subscriptions while paying for just one (vpn-contracts/lib/utilities.ak:34, vpn-contracts/validators/vpn.ak:131).

5.b - Recommendation

Enforce that the total ADA paid to the provider equals the sum of every renewal's price (or otherwise tie each extension to a unique provider payment) instead of using an `any` check that can be satisfied repeatedly by the same output.

5.c - Resolution

Resolved in `e5388ade38fd0703cfe0e94ac496b357384fc419`

6 - BLV-101 Provider burn path captures the user's ADA deposit

Category	Commit	Severity	Status
Bug	c3768121415eab902352ecf680efda7f4ea82885	Major	Acknowledged

6.a - Description

When the provider invokes `BurnVPNAccess`, the validator does not require the min-ADA deposit to be returned to the user. After `wait_time` elapses, the provider can burn the token and keep the ADA, causing a guaranteed loss of funds for expired subscriptions (vpn-contracts/validators/vpn.ak:170).

6.b - Recommendation

Require the provider burn path to output at least the min-ADA back to the original owner (or to an owner-controlled address) before permitting the token to be burned.

6.c - Resolution

Acknowledged by team "There is already trust in the provider to deliver the service or issue a refund if not, so we see no need to distrust sending the min-ADA value back to the correct address".

UTxO Company accepts this acknowledgement as legitimate.

7 - BLV-201 Minted expirations do not account for slot lag

Category	Commit	Severity	Status
Robust	c3768121415eab902352ecf680efda7f4ea82885	Minor	Acknowledged

7.a - Description

The contract sets a subscription's expiration to `get_now(validity_range) + duration`. Because the lower bound of the validity range can lag real time by 20 seconds, users receive less service than paid for and a token can be burned earlier than expected (vpn-contracts/validators/vpn.ak:59, vpn-contracts/lib/utilities.ak:34).

7.b - Recommendation

Add a buffer (e.g., 20–40 seconds) when computing the new expiration or enforce a narrow validity range so that chain lag does not shorten the subscription period.

7.c - Resolution

Acknowledged by team "Clients are removed from access in our database every hour, so the 20-second slot lag is not a concern".

UTxO Company accepts this acknowledgement as legitimate.

8 - BLV-301 Reference configuration should capture provider credentials

Category	Commit	Severity	Status
Perf	c3768121415eab902352ecf680efda7f4ea82885	Info	Acknowledged

8.a - Description

The validator receives `provider_pkh` and `provider_stake_pkh` as parameters. Storing this information inside `VPNReferenceData` would allow the provider to rotate credentials as part of the on-chain configuration update flow (vpn-contracts/validators/vpn.ak:20).

8.b - Recommendation

Include the provider payment and stake credentials in the reference datum so they can be updated without redeploying the validator.

8.c - Resolution

Acknowledged by team “We see no need to rotate provider credentials without redeploying new/different validators”.

UTxO Company accepts this acknowledgement as legitimate.

9 - BLV-302 Simplify provider address handling

Category	Commit	Severity	Status
Perf	c3768121415eab902352ecf680efda7f4ea82885	Info	Resolved

9.a - Description

`MintVPNAccess` and other redeemers reconstruct the provider address from separate payment and stake credentials. Passing a single provider address parameter would simplify the code and avoid repeatedly rebuilding the same address (vpn-contracts/validators/vpn.ak:53).

9.b - Recommendation

Accept a full provider address parameter and reuse it wherever the validator needs to compare against provider outputs.

9.c - Resolution

Resolved in `e47f33d9e9ede1ff4eb25c93e78461ebb92cd8a7`

10 - BLV-303 Streamline reference update input destructuring

Category	Commit	Severity	Status
Perf	c3768121415eab902352ecf680efda7f4ea82885	Info	Resolved

10.a - Description

`UpdateReferenceData` retrieves the consumed input via `expect Some(own_input) = ...` without unpacking the inner `Input`. Pattern matching the structure would reduce budget consumption and clarify the intent (vpn-contracts/validators/vpn.ak:82).

10.b - Recommendation

Use `expect Some(Input { output, .. }) = ...` to destructure the input in a single step.

10.c - Resolution

Resolved in `6f2f2a184ceb54dea91fcbf4e211a590e295e2e0`

11 - BLV-304 UpdateReferenceData redeemer carries redundant fields

Category	Commit	Severity	Status
Perf	c3768121415eab902352ecf680efda7f4ea82885	Info	Acknowledged

11.a - Description

The `UpdateReferenceData` redeemer includes pricing and region data that is immediately revalidated against the output datum. Removing the parameters would shrink the redeemer and avoid duplicate comparisons (vpn-contracts/validators/vpn.ak:90).

11.b - Recommendation

Drop the pricing/region fields from the redeemer and verify the new datum directly on the output.

11.c - Resolution

Acknowledged by team “Providing the new data in the redeemer seems cheaper as it allows upcasting instead of downcasting”.

UTxO Company accepts this acknowledgement as legitimate.

12 - BLV-305 Reference-NFT burn check relies on implicit mint semantics

Category	Commit	Severity	Status
Robust	c3768121415eab902352ecf680efda7f4ea82885	Info	Acknowledged

12.a - Description

The validator infers that the reference NFT was burned by checking that `policies(tx.mint) == [reference_policy_id]`. Although valid today, the logic depends on ledger semantics that forbid minting zero-value tokens (vpn-contracts/validators/vpn.ak:97).

12.b - Recommendation

Consider explicitly asserting that the reference policy is minted with quantity `-1` to avoid relying on ledger-level assumptions.

12.c - Resolution

Acknowledged by team “We assume many contracts would break if that change were to happen”.

UTxO Company accepts this acknowledgement as legitimate.

13 - BLV-306 Extend redeemer carries derivable fields

Category	Commit	Severity	Status
Perf	c3768121415eab902352ecf680efda7f4ea82885	Info	Resolved

13.a - Description

The `ExtendVPNAccess` redeemer passes `new_owner` and `token_name`, both of which can be derived from the consumed input and outputs. Removing them would shrink the redeemer and reduce off-chain coordination (vpn-contracts/validators/vpn.ak:104).

13.b - Recommendation

Limit the redeemer to the selection index and derive the owner/token name on-chain.

13.c - Resolution

Resolved in `5988629a838e16970ce7cb599509c0c812e61cab`

14 - BLV-307 Extend path input destructuring can be simplified

Category	Commit	Severity	Status
Perf	c3768121415eab902352ecf680efda7f4ea82885	Info	Resolved

14.a - Description

Similar to the reference update path, `ExtendVPNAccess` retrieves the consumed input without destructuring it. Pattern matching the `Input` structure would reduce budget consumption and improve readability (vpn-contracts/validators/vpn.ak:113).

14.b - Recommendation

Use `expect Some(Input { output, .. }) = ...` to unpack the input in one statement.

14.c - Resolution

Resolved in `6f2f2a184ceb54dea91fcbf4e211a590e295e2e0`

15 - BLV-308 Burn redeemer carries derivable token name

Category	Commit	Severity	Status
Perf	c3768121415eab902352ecf680efda7f4ea82885	Info	Resolved

15.a - Description

`BurnVPNAccess` receives the token name in the redeemer even though the token can be identified directly from the input value (vpn-contracts/validators/vpn.ak:157).

15.b - Recommendation

Remove the token-name field from the burn redeemer and verify it from the consumed input instead.

15.c - Resolution

Resolved in `3a08b3620d5d783faa5367c8532e15cba5771a74`

16 - BLV-309 `get_expiration_time_extend` uses boolean guard instead of `expect`

Category	Commit	Severity	Status
Perf	c3768121415eab902352ecf680efda7f4ea82885	Info	Resolved

16.a - Description

The helper returns `fail` after checking `if provider_paid { ... }`. Replacing the boolean guard with `expect ada_paid_to(...)` would reduce the branching and make the intent clearer (vpn-contracts/lib/utilities.ak:35).

16.b - Recommendation

Replace the boolean guard with an `expect` and refactor the subsequent conditionals accordingly.

16.c - Resolution

Resolved in `0a7eea6738ef455935e39ae180ac3f087a1a43f8`

17 - BLV-310 `output_reference_to_tn` can hash serialized data directly

Category	Commit	Severity	Status
Perf	c3768121415eab902352ecf680efda7f4ea82885	Info	Resolved

17.a - Description

The helper manually concatenates the transaction ID and output index before hashing. Using `builtin.serialize_data(output_reference)` followed by `blake2b_256` would be shorter and less error-prone (vpn-contracts/lib/utilities.ak:46).

17.b - Recommendation

Replace the manual concatenation with serialization + hash.

17.c - Resolution

Resolved in `57d7beb2207f376b780385b720738fbb9969f0e4`

18 - BLV-311 `only_2_tokens` could validate the exact asset

Category	Commit	Severity	Status
Perf	c3768121415eab902352ecf680efda7f4ea82885	Info	Resolved

18.a - Description

`only_2_tokens` counts tokens but does not verify that the non-ADA token is the expected asset. Accepting the policy ID and asset name would let the helper check both properties at once (vpn-contracts/lib/utilities.ak:124).

18.b - Recommendation

Parameterize the helper with the expected policy/asset name to avoid a follow-up iteration elsewhere in the code.

18.c - Resolution

Resolved in `0d5592629dd54b927a567b2a62004d7ffb72f6a7`

19 - BLV-312 Confusing precedence in `validators/nft.ak`

Category	Commit	Severity	Status
Read	04ede7c7944d36995bd44848524964cb260e3777	Info	Acknowledged

19.a - Description

In `validators/nft.ak`, it seems the intention is for the `&&` operator to group the first two conditions. The way precedence works in Aiken does mean that this code is correct. But that said, it is more confusing to read and potentially error prone to chain logical operators like that.

19.b - Recommendation

Prefer using the special `and/or` keywords in Aiken to completely disambiguate the precedence visually.

19.c - Resolution

Acknowledged by team “The code is correct as is and there is little value in changing”.

UTxO Company accepts this acknowledgement as legitimate.

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 UTxO Company's written consent.

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 Blink Labs to make use of the logo, name, or any other unauthorized reference to UTxO Company, except upon express authorization from the company.

UTxO Company 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.

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: `./lib/types.ak` **Hash**: `6001f7e5f227824d3cb028753eae6008e0d198a923a761435f08efa595414f83`
Filename: `./lib/utilities.ak` **Hash**: `fa1cf398f7b6d2e940bb2fe3a64ad4589b7d7aca38709d27e6e8cbc9b4fb075d`

Filename: `./validators/nft.ak` **Hash**: `d535cd560a08487c30290be250ba35207bba8113d787b6a2dfb8a3015dd5f0d5`
Filename: `./validators/vpn.ak` **Hash**: `3dfec4decd3ab109bf379261039ad054015286b9374c249670bc4c71bfda210c`

UTxO Company 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. UTxO Company strongly advises seeking multiple independent opinions on critical matters. It is the firm belief of UTxO Company 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

UTxO Company is a Blockchain-native innovation studio delivering end-to-end blockchain engineering, product design, and security reviews. We build Sigil, advance open tooling through Aiken, and continue shipping audited smart contracts across Cardano, Bitcoin, and leading EVM ecosystems. Our team pairs protocol research with production-grade infrastructure to help ambitious teams launch, scale, and govern with confidence.

A.4.1 Links

- Website
- Email
- Twitter