MELD Token Migration

Audit Report, v1.0

May 4, 2023

Contents

Revision table

Report version	Report name	Date	Report URL
1.0	Meld token migration audit	2023-05-04	Full report link

1 Executive summary

Project overview

The project migrates old Meld tokens into new Meld tokens, converting 1 old into 1 new. As the original token policy does not allow burning of tokens, the old tokens are stored in a smart contract from which they cannot be taken out, effectively freezing them forever.

Anyone who has old tokens can mint the same number of new Meld tokens, provided they lock the old tokens in a ***Locker*** UTxO, together with a Locker fee and additional Ada used as a min-Ada for the UTxO. Later, anyone can collect (batch) multiple Locker UTxOs and one ***Archive*** UTxO, collecting all old tokens from all inputs into one Archive UTxO. This is called the ***Cleanup*** transaction. The Locker fee belongs to the creator of the collect transaction but the Ada deposited on top of Locker fee is returned back to the creators of the Locker UTxOs.

Anyone can create an Archive UTxO and write their public key hash into its datum. Then each transaction spending it must be signed by the owner of the key written in the datum. A *common Archive UTxO* does not have the public key hash filled in and can be used by anyone. Typically, there will be only one common Archive UTxO (created by Meld at the start of the protocol) and the rest of Archive UTxOs will contain a public key hash. During periods of high contention for the Archive UTxO, new Archives can be created to speed up the collecting.

Old tokens are bound to the Archive script, they can only be moved from one Archive UTxO to another. This allows merging of multiple Archive UTxOs. Typical use case is the merge of a public-key-hash Archive with the common Archive, retrieving the underlying min-Ada from the public-key-hash Archive. Note that this particular merge is only possible with the signature of the owner of the public-key-hash Archive.

Methodology

Our audit collaboration consisted of a design review followed by a deeper code audit. Our manual process focused on several types of attacks, including but not limited to:

1. Double satisfaction
2. Stealing of funds
3. Violating business requirements
4. Token uniqueness attacks

5. Faking timestamps
6. Locking funds forever
7. Denial of service
8. Unauthorized minting
9. Loss of staking rewards

The audit lasted from 20 April 2023 to 4 May 2023. We interacted on Slack and gave feedback in GitHub pull requests. Most of the reported issues were fully fixed.

Summary of Findings

From the six findings which are detailed in the Findings section two were fixed fully, the rest was resolved partially to the point where their severity is only informational.

The design changed during the audit to allow the creation of additional Archive UTxOs with a public key hash in their datums. This change combats possible Denial of Service attacks. However, the Archives created by the user need to be created correctly – they need to contain a public key hash in their datum and have an empty staking address. The user can still create invalid Archive UTxOs to their own detriment as they won't be able to change them or retrieve the underlying Ada as intended. For this reason, the status of MTOK-201 and MTOK-402 is not resolved.

The finding MTOK-202 is partially resolved, because it still allows the creation of invalid Locker UTxOs. Once again, these UTxOs pose no danger to the overall design, they are only to the detriment of the user who created them incorrectly.

The finding MTOK-401 is acknowledged because its solution is off-chain which is not in the scope of this audit.

Files audited

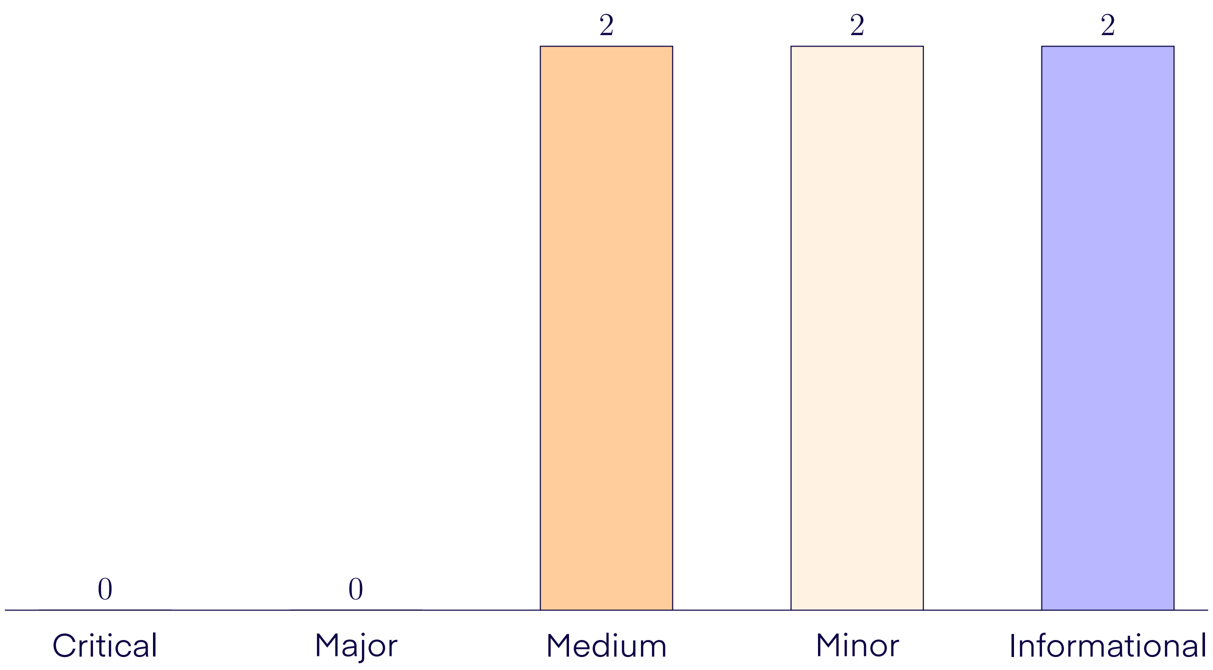
The files and their hashes reflect the final state at commit `b4737c133aba1fb84b1cbb99907c41bb62c26e8b` after all the fixes have been implemented. Note that *we did not audit* the part of `token-contracts/src/Token/Contracts/Meld.hs` handling the *PBridge* redeemer. In other words, we did not audit the bridge functionality.

SHA256 hash	Filename
92ae0...890f7	token-contracts/src/Token/Contracts/Archive.hs
16eef...54cff	token-contracts/src/Token/Contracts/Common.hs
cd414...5b283	token-contracts/src/Token/Contracts/Locker.hs
afe8c...4f3b1	token-contracts/src/Token/Contracts/Meld.hs (partial)
406e6...c7cda	token-contracts/src/Token/Contracts/Orphans.hs
1cb2b...c9333	token-contracts/src/Token/Contracts/Types/Archive.hs
d8425...28894	token-contracts/src/Token/Contracts/Types/Locker.hs
96ec2...b582d	token-contracts/src/Token/Contracts/Types/Meld.hs

The code uses multiple functions from Liqwid-Plutarch-Extra package. We have audited functions used in the files listed above.

Please note that we did not audit the files not listed above that are a part of the commit hash. We also did not assess the security of Plutarch itself. The assessment builds on the assumption that Plutarch is secure and delivers what it promises.

2 Severity overview



Findings

ID	TITLE	SEVERITY	STATUS
MTOK-201	Merging of Archive UTxOs leads to unwarranted gains	MEDIUM	PARTIALLY RESOLVED
MTOK-202	Filling up the UTxOs with arbitrary tokens	MEDIUM	PARTIALLY RESOLVED
MTOK-301	High value of `ldRefundAmount` freezes min-Ada	MINOR	RESOLVED
MTOK-302	Denial of service of the Archive	MINOR	RESOLVED
MTOK-401	Locker fee can disincentivize the use of the standard workflow	INFORMATIONAL	ACKNOWLEDGED
MTOK-402	Staking credential of an Archive disallows Archive merging	INFORMATIONAL	ACKNOWLEDGED

MTOK-201 Merging of Archive UTxOs leads to unwarranted gains

Category	Vulnerable commit	Severity	Status
Design Issue	`32d2347c5c`	MEDIUM	PARTIALLY RESOLVED

Description

Multiple Archive UTxOs can be created to increase the throughput, each containing min-Ada. It is also allowed to merge Archive UTxOs into one Archive output, collecting all old Meld tokens into one UTxO. Whoever performs this transaction can claim the min-Ada from the inputs for themselves.

Recommendation

Similarly to Locker UTxO, add information about the original creator of the Archive UTxO datum and make sure the min-Ada in the Archive UTxO is distributed back to the original creator when the Archive UTxO is merged with another one.

Resolution

Partially fixed in pull request number 98. The creator of the Archive UTxO can write their pubkey hash into its datum. Then each transaction spending that datum needs to be signed by the creator.

Archive UTxOs without the pubkey hash in the datum are still vulnerable to this attack. However, no one has the incentive to create additional Archives without pubkey hash.

MTOK-202 Filling up the UTxOs with arbitrary tokens

Category	Vulnerable commit	Severity	Status
Logical Issue	32d2347c5c	MEDIUM	PARTIALLY RESOLVED

Description

The Archive UTxO can be a part of any transaction, as long as the amount of old MELD tokens bound to it doesn't decrease. There are no other checks present. Therefore, an attacker can deposit any amount of arbitrary tokens into the Archive UTxO which increases the size of the transaction.

The attacker can add some tokens (e.g. minted using custom minting policies) into the Archive UTxO through one simple transaction in such a way that the following transaction is as close as possible to hitting transaction limits. This effectively blocks the use of Archive UTxO with the Locker as the resulting transaction would surpass the transaction limits due to the additional inputs, outputs, and the Locker script.

It is still possible to remove the additional tokens from the Archive UTxO.

You can also fill the Locker UTxO with arbitrary tokens, so that it can not be batched into any Archive UTxO. Moreover, it is not possible to retrieve these tokens from the UTxO so the Locker UTxO is locked forever. This is most likely an intentional action of the user and any loss of funds is only theirs. Apart from locking them out of Ada deposited in the Locker UTxO, it also prevents collecting all old Meld tokens in one Archive UTxO.

Recommendation

Check, that the Archive UTxOs contain only Ada and old MELD tokens. Also, limiting the use of the Archive UTxO in arbitrary transactions would help with this issue, see MTOK-302 for more information.

Resolution

Partially fixed in pull request number 97, the Archive UTxOs can be only filled with old tokens and Ada.

The less severe filling up of the Locker UTxOs has not been addressed.

MTOK-301 High value of `ldRefundAmount` freezes min-Ada

Category	Vulnerable commit	Severity	Status
Code Issue	32d2347c5c	MINOR	RESOLVED

Description

The `ldRefundAmount` datum field in `PLockerDatum` determines the amount of Ada that is refunded during the Cleanup transaction. However, if the amount is set higher than the amount of Ada present, whoever does the Cleanup transaction has to provide extra Ada for it to be sucessful. A very high value of `ldRefundAmount` might be impossible to fulfill.

The difference between the deposited Ada and the value of `ldRefundAmount` is the Locker fee. According to the specification, if the user specifies an invalid Locker fee amount, they should be able to cleanup after themselves. However, this might be impossible if the `ldRefundAmount` is not filled in correctly.

Recommendation

When minting new Meld tokens, check that the value of `ldRefundAmount` in the Locker datum is at most the amount of Ada present in the Locker.

Resolution

Fixed in pull request number 93.

MTOK-302 Denial of service of the Archive

Category	Vulnerable commit	Severity	Status
Logical Issue	32d2347c5c	MINOR	RESOLVED

Description

An Archive UTxO is important in claiming the Ada locked in Locker which can be claimed only when the Archive is present. However, the Archive validator doesn't check if there is at least one Locker in the transaction. Therefore, the Archive UTxO can be added to an arbitrary transaction with only a minimal increase of the transactions fees, making the Archive UTxO impossible to use for its designated purpose and locking Ada in Locker for prolonged periods.

Recommendation

As there can be an arbitrary number of Archive UTxOs, a higher amount of Archive UTxOs would mitigate this attack vector as it becomes too costly to block all of them. Also, the Archive validator could check whether at least one old MELD token was added to the Archive UTxO during the transaction limiting it to only reasonable transactions that collect old tokens.

Resolution

Fixed in pull request number 98. The creator of the Archive UTxO can write their pubkey hash into its datum. Then each transaction spending that datum needs to be signed by the creator. This enables the creation of additional Archive UTxOs without loss as the min-Ada that's locked in Archive UTxO is returned back to its original owner. See MTOK-201 for more context. Also, this new Archive can be used only by the owner of pubkey hash, protecting it from DoS attacks.

MTOK-401 Locker fee can disincentivize the use of the standard workflow

Category	Vulnerable commit	Severity	Status
Design Issue	32d2347c5c	INFORMATIONAL	ACKNOWLEDGED

Description

The documentation mentions that the standard Locker fee and Ada deposited are both equal to 2 Ada. In total, the user deposits 4 Ada. From the deposited Locker fee and additional Ada, they eventually get back the additional Ada but the Locker fee is lost. In addition, they pay for the transaction fees. The standard workflow leads to a total cost to the user equal to the Locker fee (currently 2 Ada) plus transaction fees.

Alternatively, a user can deposit min-Ada into a Locker, without paying any Locker fee. As there's no Locker fee, the min-Ada might stay locked in there forever. The cost of this approach for the user is just the min-Ada and transaction fees.

The Locker fee setting resides offchain and could change. When set too high, users will not have an incentive to use the intended standard workflow, as they could use the alternative without any Locker fees paid and they will save costs, as Locker fee is much bigger than min-Ada. The min-Ada will possibly stay there locked forever and the old Meld tokens will never reach the Archive UTxO as intended.

We think it's unlikely that the users would build these transactions themselves but someone might create a simple web just for the single purpose of saving Locker fees for users.

Recommendation

The recommendation is regarding the offchain Locker fee setting: make sure that it is not set too high.

Resolution

Acknowledged by the team. The Locker fee will be set to 1.8 Ada initially, and an additional Ada that is returned upon merging to the Archive will be 2 Ada.

MTOK-402 Staking credential of an Archive disallows Archive merging

Category	Vulnerable commit	Severity	Status
Code Issue	`32d2347c5c`	INFORMATIONAL	ACKNOWLEDGED

Description

The staking part of the Archive address can be set to anything by the creator of the Archive UTxO. The Archive script expects the output to be at the exact same address as the input, so two Archive UTxOs with different staking credentials cannot be merged and their staking address cannot change.

As a result, the min-Ada deposited into an Archive UTxO might not be retrievable and multiple Archive UTxOs might exist forever. The merging of additional Archive UTxOs is still possible as long as their staking address is the same as a common Archive UTxO staking address which will be empty.

The Locker validator requires that the staking address of the input Archive is empty, therefore no old Meld tokens can be deposited to the incorrectly created Archive through the standard workflow.

Recommendation

Skip the staking credential when comparing the Archive UTxOs addresses. That will also allow the change of the staking address in the Archive UTxO which should be disabled for the common Archive.

Resolution

Acknowledged by the team without change due to the low severity and no potential risk to the overall protocol.

A Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the agreement between VacuumLabs Bohemia s.r.o. (Vacuumlabs) and Sogo Labs (Client) (the Agreement), or the scope of services, and terms and conditions provided to the Client in connection with the Agreement, and shall be used only subject to and to the extent permitted by such terms and conditions. This report may not be transmitted, disclosed, referred to, modified by, or relied upon by any person for any purposes without Vacuumlabs's prior written consent.

This report is not, nor should be considered, an endorsement, approval or disapproval of any particular project, team, code, technology, asset or anything else. This report is not, nor should be considered, an indication of the economics or value of any technology, product or asset created by any team or project that contracts Vacuumlabs to perform a smart contract assessment. This report does not provide any warranty or guarantee regarding the quality or nature of the technology analysed, nor does it provide any indication of the technology's proprietors, business, business model or legal compliance.

To the fullest extent permitted by law, Vacuumlabs disclaims all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. This report is provided on an as-is, where-is, and as-available basis. Vacuumlabs does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by Client or any third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services, assets and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and Vacuumlabs will not be a party to or in any way be responsible for monitoring any transaction between you and client and/or any third-party providers of products or services.

This report should not be used in any way by anyone to make decisions around investment or involvement with any particular project, services or assets, especially not to make decisions to buy or sell any assets or products. This report provides general information and is not tailored to anyone's specific situation, its content, access, and/or usage thereof, including any associated services or materials, shall not be considered or relied upon as any form of financial, investment, tax, legal, regulatory, or other advice.

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Vacuumlabs prepared this report as an informational exercise documenting the due diligence involved in the course of development of the Client's smart contract only, and THIS REPORT MAKES NO CLAIMS OR GUARANTEES CONCERNING THE SMART CONTRACT'S OPERATION ON DEPLOYMENT OR POST-DEPLOYMENT. This report provides no opinion or guarantee on the security of the code, smart contracts, project, the related assets or anything else at the time of deployment or post deployment. Smart contracts can be invoked by anyone on the internet and as such carry substantial risk. VACUUMLABS HAS NO DUTY TO MONITOR CLIENT'S OPERATION OF THE PROJECT AND UPDATE THE REPORT ACCORDINGLY.

THE INFORMATION CONTAINED IN THIS REPORT MAY NOT BE COMPLETE NOR INCLUSIVE OF ALL VULNERABILITIES. This report is not comprehensive in scope, it excludes a number of components critical to the correct operation of this system. You agree that your access to and/or use of, including but not limited to, any associated services, products, protocols, platforms, content, assets, and materials will be at your sole risk. On its own, it cannot be considered a sufficient assessment of the correctness of the code or any technology. This report represents an extensive assessing process intending to help Client increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology, however blockchain technology and cryptographic assets present a high level of ongoing risk, including but not limited to unknown risks and flaws.

While Vacuumlabs has conducted an analysis to the best of its ability, it is Vacuumlabs's recommendation to commission several independent audits, a public bug bounty program, as well as continuous security auditing and monitoring and/or other auditing and monitoring in line with the industry best practice. The possibility of human error in the manual review process is highly real, and Vacuumlabs recommends seeking multiple independent opinions on any claims which impact any functioning of the code, project, smart contracts, systems, technology or involvement of any funds or assets. VACUUMLABS'S POSITION IS THAT EACH COMPANY AND INDIVIDUAL ARE RESPONSIBLE FOR THEIR OWN DUE DILIGENCE AND CONTINUOUS SECURITY.

B Issue classification

Severity levels

The following table explains the different severities.

Severity	Impact
CRITICAL	Theft of user funds, permanent freezing of funds, protocol insolvency, etc.
MAJOR	Theft of unclaimed yield, permanent freezing of unclaimed yield, temporary freezing of funds, etc.
MEDIUM	Smart contract unable to operate, partial theft of funds/yield, etc.
MINOR	Contract fails to deliver promised returns, but does not lose user funds.
INFORMATIONAL	Best practices, code style, readability, documentation, etc.

Resolution status

The following table explains the different resolution statuses.

Resolution status	Description
RESOLVED	Fix applied.
PARTIALLY RESOLVED	Fix applied partially.
ACKNOWLEDGED	Acknowledged by the project to be fixed later or out of scope.
PENDING	Still waiting for a fix or an official response.

C Report revisions

This appendix contains the changelog of this report. Please note that the versions of the reports used here do not correspond with the audited application versions.

v1.0: Meld token migration audit

Revision date:	2023-05-04
Final commit:	`b4737c133aba1fb84b1cbb99907c41bb62c26e8b`

We conducted the audit of the Meld token migration smart contracts. To see the files audited, see Executive Summary.

Full report for this revision can be found at url.

D About Us

Vacuumlabs has been building crypto projects since the day they became possible on the Cardano blockchain.

- Helped create the decentralized exchange on Cardano – WingRiders, currently the second largest exchange on Cardano (based on TVL).
- We are the group behind the popular AdaLite wallet. It was later improved into a multichain wallet named NuFi which also integrates a decentralised exchange.
- We built the Cardano applications for hardware wallets Ledger and Trezor.
- We built the first version of the cutting-edge decentralized NFT marketplace Jam on Bread on Cardano with truly unique features and superior speed of both the interface & transactions.

Our auditing team is chosen from the best.

- Ex-WingRiders, ex-NuFi developers
- Experience from Google, Spotify, traditional finance, trading and ethical hacking
- Medals and awards from programming competitions: ACM ICPC, TopCoder, International Olympiad in Informatics
- Passionate about Program correctness, Security, Game theory and Blockchain

We are a trusted Cardano ecosystem development partner

Contact us:

audit@vacuumlabs.com