vacuumlabs

Ardana dUSD

Design Review Report, v1.0

May 11, 2023

Contents

Revision table

Report version	Report name	Date	Report URL
1.0	Specification design review	2023-05-11	Full report link

1 Executive summary

Project overview

The *Ardana dUSD project* aims to create a stablecoin on Cardano named dUSD. It consists of four main on-chain components – the Price Module, Vaults, the Buffer and the Protocol Parameters Module. In this overview, we will introduce each of these four parts and describe the most essential interactions. The dUSD stablecoin is backed by ADA collateral.

Vault

The *Vault* is the central component of the dUSD design. It holds the collateral backing the stablecoin and allows a user to take a loan based on this collateral by minting dUSD tokens. There can be multiple vaults, any user can create an arbitrary number of vaults that belong to her.

The vault is represented by a single UTxO that holds information about the owner of the vault, debt taken, and the size of the collateral which is also the value of this UTxO.

The owner of the vault can at any time deposit or withdraw ADA, take a loan, or pay back the loan. This can happen multiple times, e.g. the vault owner can increase the size of the debt or pay back only a portion of the debt.

When a user takes a loan, dUSD tokens are minted and are available to the owner. These tokens are backed by the collateral in the vault.

The vault is trying to balance the amount of collateral and debt that are bound to it and stay in *a healthy state*. The health of the vault is determined by the *collateralization ratio* which is the ratio between the value of the collateral and the value of the loaned dUSD. The vault is deemed healthy if this collateralization ratio is bigger than a parameter named *liquidation ratio*. Taking out a further loan or withdrawing ADA is only possible if the vault stays healthy even after this operation.

However, due to the fluctuation of the price of both ADA and dUSD on the market, the collateralization ratio of the vault can drop below the liquidation ratio, at which point the vault is no longer healthy and a liquidation can take place to make the vault healthy again – note that in some situations a liquidation can not make a vault healthy again.

Liquidation allows anyone to pay back a part of the debt and obtain an appropriate amount of collateral in return. Moreover, the liquidator buys the collateral ADA at a discount, and not all dUSD paid by the liquidator is used to repay debt. There is a liquidation fee that the owner of the vault needs to pay. The vault can be liquidated all at once or

in parts until the collateralization ratio is bigger than the liquidation ratio and the vault is healthy once again.

Note that the vault cannot be liquidated the moment it becomes unhealthy. The vault needs to be unhealthy for at least one hour to allow liquidation. This gives an opportunity to the owner to deposit more ADA into it and reach a healthy state of the vault and also protects the owner from the temporary fluctuation of prices.

Buffer

The *Buffer* is an on-chain component that is responsible for keeping the value of dUSD token stable at 1 USD. It accomplishes this by creating surplus and debt auctions. The buffer starts with some initial value that acts as base collateral. Afterward, the buffer is filled by auctions, some of them indirectly triggered by users that need to pay stability fees. Stability fee is paid to the vault but the need for it increases the price of dUSD and thus can trigger an auction.

The surplus and debt auctions can be triggered by anyone, once the corresponding conditions are met. The surplus action can be run when the dUSD token depegs from USD by more than 1% in a positive direction. In such a case, some amount of dUSD is minted in the buffer and sold for ADA which is gathered in the buffer.

The debt auction is triggered when the depeg of dUSD from USD is at least 1% in a negative direction. This auction buys and burns dUSD in exchange for ADA from Buffer's reserves.

The value that accumulates in the buffer can be controlled by an additional mechanism. The admin can run special auctions that sell or buy the project's own DANA tokens. The selling of DANA fills the buffer as it is sold for some value of ADA. These tokens can be sold back to the buffer during the opposite auction to claim ADA rewards, emptying the buffer.

Price Module

For all the calculations in the dUSD stablecoin model, it is important to know the current price of dUSD and ADA. For this purpose, the on-chain Price Module is implemented. It is a UTxO that congregates the last 48 hours of price data. These price data are collected through off-chain and are put into the Price Module by the admin. Only the admin can change the UTxO datum, other users can use it as a read-only input of their transactions.

The off-chain bot should collect the prices once an hour, obtaining the average price from at least three different sources and taking the median of them.

Protocol Parameters Module

There are multiple parameters that affect the whole dUSD design – e.g. liquidation discount, liquidation fee, stability fee, and others. These parameters are held in the Protocol Parameters Module. It is a UTxO that contains these parameters in its datum. These parameters can be subject to change, but only the admin can change them.

This, however, means that values can change at will of the administrator to the detriment of the user. For example, the stability fee (which is basically the interest rate of the loan) is a sensitive parameter as it determines how much more money the user needs to pay back. To circumvent this problem and offer some stability, the last 20 values of the stability fee are held in the datum with their timestamps at the moment of change. Therefore, the total interest can be calculated more precisely with historical data.

Governance

The provided design specification is unclear about the governance of the whole system. It mentions three admin keys that are hard-wired into the protocol on the compilation. These keys then have the rights to update protocol parameters, the price module and run DANA tokens auctions. They have no control over the vaults and the value inside of the vaults.

However, it is not specified if these keys are independent and any one of them has full rights on its own, or if they are part of a multisignature schema in which a multiple of them need to be present for any given update.

There is also a mention of a more community based approach to governance, although it is stated that this is a plan for further versions of the protocol.

Methodology

The review of Ardana's dUSD stablecoin was *design-only* – our goal was to read the design specification, find possible vulnerabilities and suggest improvements early on, so it would serve as a better basis for the developers implementing it.

The collaboration started on 12 September 2022. We went through the provided specification and identified several issues and unclear parts. We asked for clarifications which were partially answered during a call with one of the project members. According to the client, the design specification was not finished and changes to it were expected.

Over the following months, the project was put on hold. It was announced that the project halted operation. We have not received the assistance necessary to continue.

However, later, the remaining team members wanted to finish the review with the specification as-is. We resumed the work on 15 March 2023. All found vulnerabilities and inconsistencies were reported to the client and can be found in this report in detail.

There were no changes to the design document during our collaboration. Furthermore, there was lack of any response from the client, especially later in the process. As a result, the reported issues remain pending and *unresolved*. Further collaboration is necessary in case the work would be continued.

Our manual process focused on several types of attacks, including but not limited to:

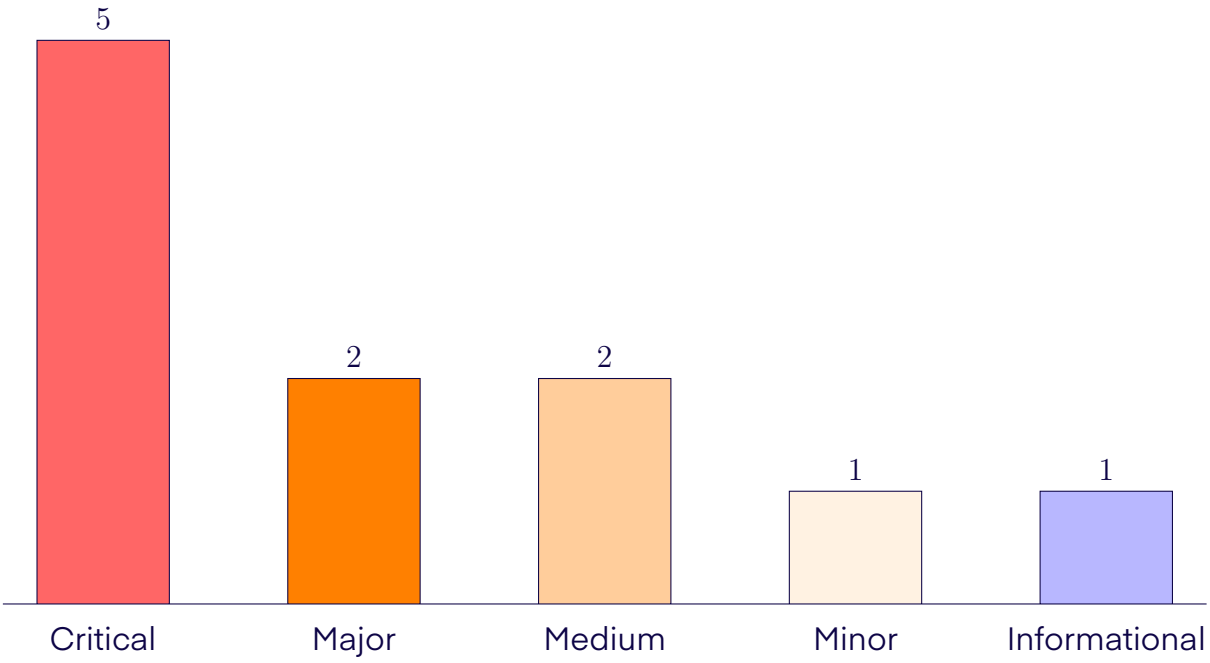
1. Ambiguous or unclear specification
2. Violating business requirements
3. Unexpected loss of value
4. Faking timestamps
5. Denial of service
6. Loss of staking rewards
7. Double satisfaction
8. Locking funds forever
9. Stealing of funds

Files audited

The files and their hashes reflect the final state at commit `4e336507945f470494421f094a1e1c19150bf8be` .

SHA256 hash	Filename
`fede0...46eba`	`dUSD/docs/dusd-spec.tex`

2 Severity overview



Findings

ID	TITLE	SEVERITY	STATUS
ARD-001	Unclear specification of Buffer and auctions	CRITICAL	PENDING
ARD-002	Liquidation burns too little, duplicating value	CRITICAL	PENDING
ARD-003	Fixed exchange rate slows down liquidations	CRITICAL	PENDING
ARD-004	Unclear specification of the governance system	CRITICAL	PENDING
ARD-005	Use of fixed admin keys	CRITICAL	PENDING
ARD-101	Deposited ADA is not staked	MAJOR	ACKNOWLEDGED

Continued on next page

ID	TITLE	SEVERITY	STATUS
ARD-102	Limited storage of historical stability fees decreases costs	MAJOR	PENDING
ARD-201	Timestamp is a lower bound of 12-hours-long validity windows	MEDIUM	PENDING
ARD-202	Liquidations not batched, slowing the process down	MEDIUM	PENDING
ARD-203	Unprofitable small liquidations can make dUSD undercollateralized	MEDIUM	PENDING
ARD-301	Unfulfillable acceptance criterium for vault	MINOR	PENDING
ARD-401	Price module contention	INFORMATIONAL	PENDING

ARD-001 Unclear specification of Buffer and auctions

Category	Vulnerable commit	Severity	Status
Design Issue	4e33650794	CRITICAL	PENDING

Description

One of the core parts of the dUSD model is the Buffer which serves as the way to correct price deviations of dUSD from USD. It accomplishes this task by creating debt or surplus auctions. However, the design of the Buffer and auctions is unclear and vague. The Buffer depends on several important parameters (stability fee, amount of ADA in it, amount sold in an auction...) which are not set and it is not defined how they will be calculated.

We therefore cannot give any guarantees or verify assumptions about this part of the design.

Recommendation

Improve the specification by describing the more precise workings of Buffer and its auctions.

The precise setup of the Buffer is crucial for the whole design, the specification should contain more details about how it is set up (e.g. the initial amount of underlying ADA) and some analysis of the impact of the stability fee on the size of the Buffer. Also, some strategies for dealing with possible situations (e.g. emptying out the buffer) should be outlined.

For auctions, some design details on the used UTxOs and possible actions are needed.

ARD-002 Liquidation burns too little, duplicating value

Category	Vulnerable commit	Severity	Status
Design Issue	4e33650794	CRITICAL	PENDING

Description

The specification claims that 10% of the dUSD used during liquidation is burned. It is unclear what happens to the rest (i.e. 90%), as the specification mentions that "[it] is used to pay back debt". Due to ambiguity, whoever implements the protocol based on the specification might choose the less secure interpretation. Hence, we assume that the 90% is not burned but stays in the protocol, e.g. in the vault for the vault owner to collect.

After a successful liquidation, most of the collateral is taken away from the vault but the original dUSD amount still exists as well as the 90% used during liquidation. As collateral is removed from sick vaults during liquidations but the 90% of dUSD tokens minted in the sick vaults stay around, gradually, as more liquidations are executed, the stablecoin becomes undercollateralized, causing a depegged value of dUSD. The only fix is depositing more collateral, but no one has an incentive to do so.

Recommendation

Burn all dUSD used to liquidate a sick vault. Write 10% of the paid dUSD as remaining debt into the vault datum. This has to be paid by the vault owner before they can retrieve all remaining collateral from the vault.

ARD-003 Fixed exchange rate slows down liquidations

Category	Vulnerable commit	Severity	Status
Design Issue	4e33650794	CRITICAL	PENDING

Description

When liquidating a vault, the user pays 97 dUSD for 100 USD worth of ADA in the vault, ignoring the fact that the exchange rate of dUSD can leave the target exchange rate of $1:1$. This leads to liquidations being unprofitable when 1 dUSD is worth more than 1 USD and more profitable when 1 dUSD is worth less.

For example, when dUSD is worth 1.05 USD, users would not be motivated to do the liquidations, as they could pay more than 100 USD ($105 \cdot 0.97 = 101.85$) for something only worth 100 USD. When the depeg is in the opposite direction, i.e. when dUSD is worth less than 1 USD, the problem is the opposite: liquidations are too lucrative.

Even if this leads to dUSD eventually rising in price, as market actors buy up dUSD to make profitable liquidations with it, the liquidators might artificially wait for larger depegs which are even more profitable for them. This behavior in turn leads to speculative behavior and is slowing the liquidations down.

Recommendation

Use the dUSD/ADA price to determine the amount of ADA that liquidators get during liquidations, using the price from an oracle. Note that liquidations would be still profitable for liquidators due to the 3% discount on the collateral.

ARD-004 Unclear specification of the governance system

Category	Vulnerable commit	Severity	Status
Design Issue	4e33650794	CRITICAL	PENDING

Description

A major part of the proposed dUSD model is its form of governance. It dictates how the important parameters such as the stability fee or the liquidation fee are updated, and also who has control over the Buffer and can run DANA auctions. A safe and trustworthy set of rules for governance is important for the trust of the users.

The design specification mentions three admin keys that are hard-wired into the system but fails to give details about the use of these keys in regard to the mentioned actions. The most important part is whether any single one of these keys has full access rights or whether there is some more complicated schema in place.

Recommendation

Specify the governance of the model in more detail, focus on describing the process that is used to approve changes to the model.

We recommend using a multisignature schema that requires multiple signatures for each transaction.

ARD-005 Use of fixed admin keys

Category	Vulnerable commit	Severity	Status
Design Issue	4e33650794	CRITICAL	PENDING

Description

The script addresses of the Price module and the Buffer depend on the three admin keys, so the keys are very hard or impossible to rotate. In case any admin key is compromised, it should be rotated as fast as possible but this design choice prevents that.

Moreover, the design specification doesn't state how many keys are needed to sign admin transactions (e.g. a change of protocol parameters, updating the price module), see the finding ARD-004. We must assume that any single admin key is sufficient. Therefore, if any admin key is compromised, the attacker gains absolute control over the protocol without any possibility of removing his access rights.

Recommendation

Use a multisignature schema that ideally can update its admin keys.

Also, we suggest that admin transactions need to be signed by multiple admin keys (e.g. the majority of them) and the number of multisignature schema participants is increased as the three-member schema can be easily compromised with only two keys.

ARD-101 Deposited ADA is not staked

Category	Vulnerable commit	Severity	Status
Design Issue	4e33650794	MAJOR	ACKNOWLEDGED

Description

Deposited ADA tokens in the vaults smart contracts are not staked. The users are losing on potential gains by depositing into a vault.

Recommendation

Allow staking in the vaults. Some staking rewards could go back to the protocol, e.g. to the Buffer.

Resolution

The team acknowledged the issue, as staking was not planned to be a part of the first version of the protocol.

ARD-102 Limited storage of historical stability fees decreases costs

Category	Vulnerable commit	Severity	Status
Design Issue	4e33650794	MAJOR	PENDING

Description

Protocol parameters are stored in the Protocol Parameters Module UTxO. The design states that the stability fee parameter is stored in a map based on timestamps of stability fee changes. The main reason is that vaults are meant to operate with parameters that were valid during the whole time period when they were used. However, due to the limited size of the datum only the last 20 changes are kept.

Ultimately, some historical stability fees will be lost, leading to stability fee payments that might be too low. For example, only the stability from the last year might be considered when paying back a five-year loan. Hence, users might wait out periods of high interest rates to make their payments low.

Recommendation

One way to solve the issue could be to use the concept of prefix sums. Store the accrued interest since the beginning of the protocol (called *compounded interest*) in the Protocol Parameters. Also store the value of the compounded interest in the vault, using the valid value from when it was created. For example, if the first year the interest rate is 1% p.a. and the second year it is 2% p.a., the total accrued interest since the beginning of the protocol is a bit over 3%, so the compounded interest is approximately 1.03.

For a vault created at the end of the second year, the compounded interest of 1.03 is recorded in its datum. One year later, during which the interest was also 2%, the compounded interest is $1.03 \cdot 1.02 = 1.0506$. If the vault is then fully paid back, the user pays interest of $1.0506/1.03 = 1.02$, so exactly 2%.

ARD-201 Timestamp is a lower bound of 12-hours-long validity windows

Category	Vulnerable commit	Severity	Status
Design Issue	4e33650794	MEDIUM	PENDING

Description

The design mentions that the current timestamp used for calculating various prices is set as a lower bound of `txInfoValidRange`. It is stated that the longest possible validity window will be 12 hours. This, however, leads to speculations about the prices stored in the Price module as the lower bound can be set arbitrarily (within 12 hour window) during the transaction creation.

Recommendation

Use the upper bound of `txInfoValidRange` when calculating the current timestamp instead. Doing so also removes the need to bound the length of the validity window.

ARD-202 Liquidations not batched, slowing the process down

Category	Vulnerable commit	Severity	Status
Design Issue	4e33650794	MEDIUM	PENDING

Description

The health of the stablecoin relies heavily on the speed of the liquidation process. If the underlying collateral asset, in this case ADA, drops quickly in value and the liquidators are not able to react quickly enough, the reserves in the vaults will drop too much and the dUSD will depeg from USD.

The design document disallows batching but depending on the implementation and the nature of market participants, it might be needed. Especially batching of multiple liquidations done by one liquidator might be more efficient than doing them one by one.

Recommendation

Allow batching of multiple liquidations by one liquidator in one transaction. Note that if the transaction fees and liquidation throughput in the final implementation end up good enough, batching might not be needed. However, the design specification should not forbid it before these are empirically verified.

The result of the batching should be the same as if the vaults were liquidated one by one. Make sure that the batching code is executed exactly once, in order to save transaction fees. Be also mindful of burning the appropriate amount of dUSD and of double-satisfaction attacks. A possible solution to double-satisfaction is to make sure that the NFTs identifying vaults on the inputs are also present in the outputs, one per every vault output.

ARD-203 Unprofitable small liquidations can make dUSD undercollateralized

Category	Vulnerable commit	Severity	Status
Logical Issue	4e33650794	MEDIUM	PENDING

Description

Small liquidations can be unprofitable, as the liquidator earns 3% of the collateral but they have to pay transaction fees. The liquidator can liquidate the loan only until the vault is healthy again. Over time, vaults that are not healthy but are also too expensive to liquidate can accumulate in the protocol and after a fall in the price of Ada, they become undercollateralized. The whole dUSD can thus become undercollateralized.

Recommendation

Introduce a *minimal liquidation size* parameter s, meaning that the liquidator can always liquidate at least s Ada of collateral from an unhealthy vault. Note that there also has to be a mechanism in place to make all vaults of size at least s during normal operation, e.g. a withdrawal cannot make the vaults smaller than that. Furthermore, vaults of size below $2 \cdot s$ Ada have to be liquidated fully, as the remaining vault would be too small.

ARD-301 Unfulfillable acceptance criterium for vault

Category	Vulnerable commit	Severity	Status
Design Issue	4e33650794	MINOR	PENDING

Description

The design specification defines one of the acceptance criteria for the vault as "Unless in the case of resource contention, any vault is liquidated within 1 hour of becoming sick.".

Vault can become sick due to the price change of the underlying token. This change is updated in the Price module once per hour. Moreover, liquidation is possible only when at least the last two records from the Price module report the vault as sick. Hence, it can take almost two hours, before the liquidation of the sick vault is even possible.

Recommendation

We recommend more frequent updates of the Price module or updating this assumption to better reflect the rest of the design and update all parts that are dependent on it.

ARD-401 Price module contention

Category	Vulnerable commit	Severity	Status
Implementation suggestion	4e33650794	INFORMATIONAL	PENDING

Description

The Price module, the UTxO congregating the last 48 hours of price data, is central to the presented design. It is used by both admins (to update the price data) and users (to perform various operations on the model). Therefore, this UTxO will be heavily contested and can lead to low throughput of transactions.

The design states that the Price module should be read-only for the users. However, this can be achieved in multiple ways (e.g. reference inputs or spending and recreating of the UTxO).

Recommendation

The Price module can be used as a read-only reference input for the users' transactions to allow concurrency of user transactions. Using reference inputs requires the V2 script functionality[1].

[1] https://github.com/cardano-foundation/CIPs/pull/159

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 Ardana Labs, Inc (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: Specification design review

Revision date: 2023-05-11
Final commit: `4e336507945f470494421f094a1e1c19150bf8be`

We conducted the design review of the dUSD specification. To see the files reviewed, see the 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

Contact us:
audit@vacuumlabs.com