



Report by ULAM LABS

Security Assessment for Deflex

Report by Ulam Labs

Deflex Contracts

Findings and Recommendations Report Presented to:

Deflex Team

April 20, 2023 Version: 0.1

Presented by:

Ulam Labs

Grabiszynska 163/502,

53-332 Wroclaw, POLAND

© Ulam Labs 2023. All Rights Reserved.

Executive Summary

Overview

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the Ulam Labs Security Teams took to identify and validate each issue, and any applicable recommendations for remediation.

Scope

The audit has been conducted on the commit **18c31806f8e60a0c10f8cefaa3d602ff6b1ffe8c** of Deflex private GitHub Repository.

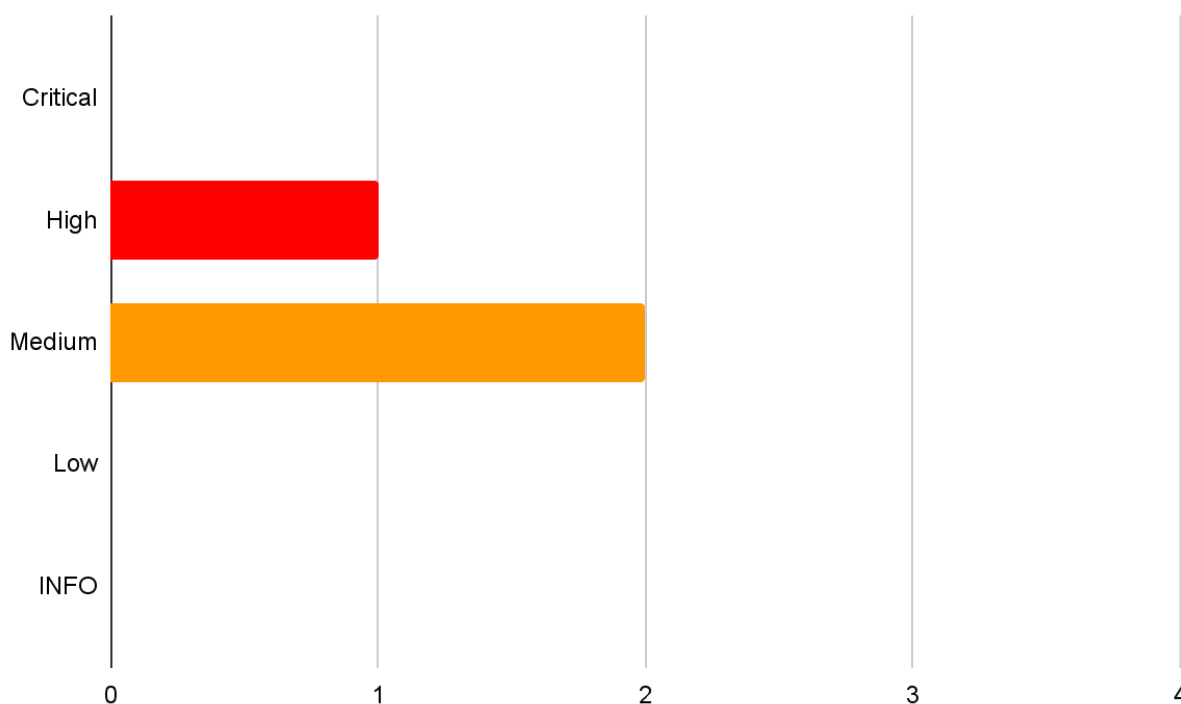


Chart 1: Findings by severity.

Key findings

During the Security Assessment, following findings have been discovered:

- 0 findings with a CRITICAL severity rating,
- 1 findings with a HIGH severity rating,
- 2 findings with a MEDIUM severity rating,
- 0 findings with a LOW severity rating.
- 0 findings with an INFO severity rating.

Disclaimer

This report does not constitute legal or investment advice. The preparers of this report present it as an informational exercise documenting the due diligence involved in the secure development of the target contract only, and make no material claims or guarantees concerning the contract's operation post-deployment. The preparers of this report assume no liability for any and all potential consequences of the deployment or use of the contract.

Smart contracts are still a nascent software arena, and their deployment and public offering carries substantial risk. This report makes no claims that its analysis is fully comprehensive, and recommends always seeking multiple opinions and audits.

This report is also not comprehensive in scope, excluding a number of components critical to the correct operation of this system.

The possibility of human error in the manual review process is very real, and we recommend seeking multiple independent opinions on any claims which impact a large quantity of funds.

Technical analysis & findings

Order router can be permanently blocked by any user

Finding ID: **DFX-1**

Contract: order_router_app.teal

Severity: **High**

Status: **Fixed**

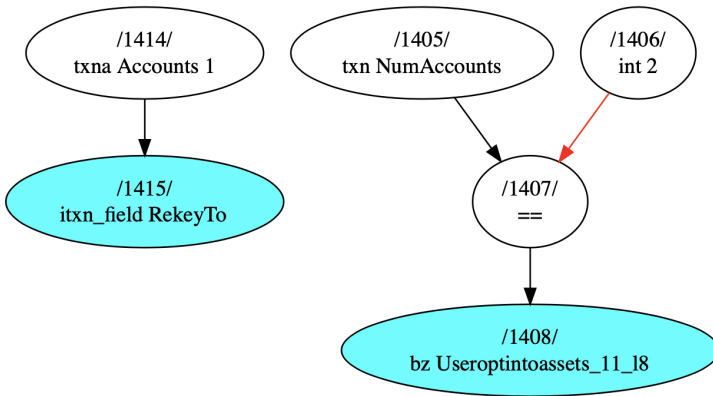
Description

To block the router contract, it is enough to make only one opcode from any method used during each swap constantly failing. There are not too many candidates, because in most cases the state is nicely cleaned up in the swap finalize method.

However, there is one exception: minimum balance.

At first look, we can see many lines of code handling asset opt-ins and close-outs, however one source of minimum balance change has been omitted.

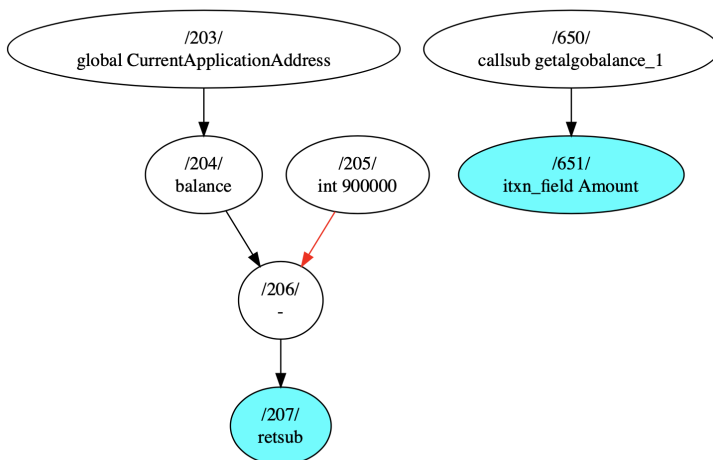
In the method opt into assets, there is an option to rekey the router contract to any address.



After rekey, we should be ready for two cases:

- Sender can spend all the funds that router has. It is not problematic, because the router does not have anything to spend.
- Min balance can be increased permanently by deploying applications without delete support.

Let's take a closer look at the second case. Normally, it can be problematic, when we want to delete a contract, but here it is not supported. Minimum balance is also important, if we want contract to send some ALGOS like in fill finalize method:



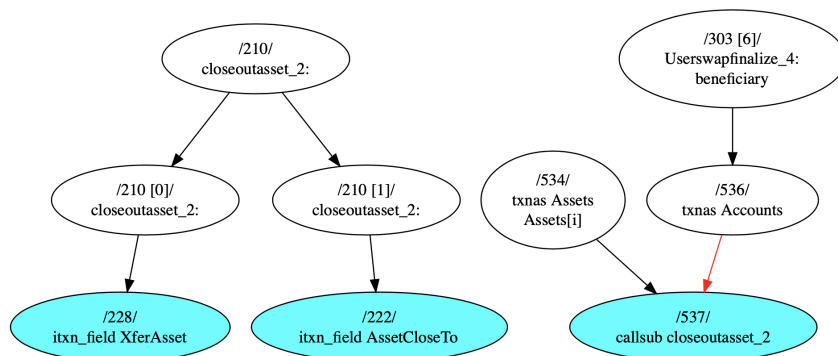
Such payment guarantees, the contract at the end will have at exactly 900000

© Ulam Labs 2023. All Rights Reserved.

ALGOS.

As we have shown previously, anyone can increase contract minimum balance above 900_000 and then make the inner transaction fail. However, there is a problem, because to set a new minimum balance for all users, an inner transaction, which must fail for others must first succeed for us.

It turns out that there is one case, when it is possible to send the whole balance regardless of minimum balance. It is possible if the sender is the same as the receiver.



Calgraph above shows that beneficiary address is also used to close assets. It would make use of the contract address as beneficiary impossible, but it turns out, to use a router, the user does not have to opt into any asset.

Impact

After the minimum balance is raised above 900_000 ALGOS, no one else can use the router anymore, because the only valid beneficiary is the router address.

Such an attack would probably cause denial of service for about 24 hours, because it would take some time to spot the problem, find the root cause and deploy the correction.

Solution

If we know that minimum balance can be changed, it should be checked at

the end of the transaction.

Comparing minimum balance to 100_000 solves the problem.

Status

The issue has been reported to the Deflex team. Correction delivered and deployed.

Anyone can manipulate registry app state

Finding ID: **DFX2-2**

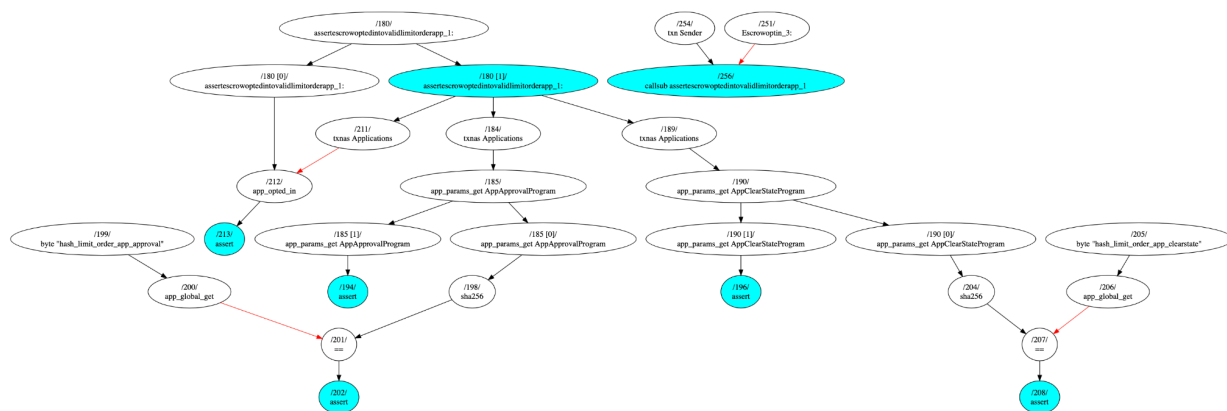
Contract: registry_app.teal

Severity: **Medium**

Status: **Open**

Description

Registry app aggregates statistics about orders. Application accepts inputs only from senders opted to valid limited order application.



Everything looks fine, but there is one problem. Users can opt in into any app and then update this application to limit order app. In such case user can omit some steps, which are normally performed during opt in.

Impact

Potential consequences of such a flaw could be disastrous, but in this case, the application does not store any funds and manipulates just two counters.

Those counters could be a source of integer overflow, but decrementation is possible only after incrementation.

After all, an attacker can just “improve” statistics by paying only transaction fees.

Solution

If approval and a clear program is used to authenticate an address, it is crucial to do it, while application is created. Then we have proof that the application will never be updated, if the approval program does not allow it.

Order matching bots cannot trust limit order apps

Finding ID: **DFX2-3**

Contract: limit_order_app.teal

Severity: **Medium**

Status: **Open**

Description

As shown in the previous issue, the state of the limit order app registered in the registry app cannot be trusted. Whole setup phase can be omitted, faked or something extra can be done.

Impact

Order matching bots should expect literally everything, while processing the orders. It makes the backend code much more complicated.

The biggest risk of such a situation is making bots paying transaction fees for transactions, which could not succeed.

As backend code is out of scope, it is hard to say if the problem really exists, but it is recommended to revisit the code and check if there are no assumptions based on contract code.

Solution

Out of scope.

General observations

General safety

Contract is designed very well. Written in Pyteal, it has all the required checks to keep user funds safe. The idea with authentication using an approval program is also very nice, but just one attack has been overlooked. Thankfully, there are no consequences of such behavior.

Other

Severity classification

We have adopted a severity classification inspired by the **Immunefi Vulnerability Severity Classification System - v2**. It can be found [here](#).