Indigo
Follow

Nov 11, 2021
.
9 min read
.

Indigo Protocol: Tweag Security Audit Report

We are pleased to announce that Indigo just finished the first one-month audit before going to testnet. This audit reviews Indigo's potential risks to guarantee good code quality. As transparency is an important characteristic of blockchain, we believe that it is necessary to disclose all our activities to the community.

Our partner in this audit is Tweag — a Software Information Lab, founded by compiler writers, mathematicians, and distributed systems experts. Their works span across different areas, including Software Engineering, Infrastructure, Statistics, and Applied Research. They are one of the largest contributors to the Haskell programming language, upon which Plutus is built. Tweag is also the creator of Ormolu which is a popular formatter for Haskell source code.

Indigo is aiming to be the first synthetic application that is stable and powerful on the Cardano platform. With many years of audit experience, Tweag helped us achieve that goal by finding the vulnerabilities in our on-chain validation code and giving us useful recommendations to improve our code.

Audit Process, Scope & Methodology

. . .

Tweag analyzed the smart-contract code provided by Indigo Protocol, focusing on on-chain validator scripts. These scripts are documented in our Yellow paper.

They look exclusively at the on-chain validation code provided by Indigo. This means they exclude all off-chain code, test cases, and relevant infrastructure. Tweag manually inspected the code contained in the respective files and attempted to locate potential problems in one of these categories:

- Unclear or wrong specifications that might allow for fringe behavior.
- The implementation doesn't follow the specification.
- Vulnerabilities an attacker could exploit if the code were deployed as-is, including:
- Race conditions or denial-of-service attacks block other users from using the contract.
- Incorrect dust collection and arithmetic calculations (including due to overflow or underflow).
- Incorrect minting, burning, locking, and allocation of tokens.
- Authorization issues.

- Code quality and lack of documentation might create confusion for the developers.

After inspecting every line of code, Tweag pointed out 14 issues in the on-chain implementation: 3 issues are critical severity, 2 issues are high severity, 3 issues are medium severity, 5 issues are low severity issues, and 1 issue is in the lowest severity. Those issues, however, do not make us nervous. In fact, we are glad to detect them early, even one of them is our original intention when coding. We provide more details in the section below.

Tweag uses the following criteria to classify severity:

- **CRITICAL** : The issues could prevent the contract to be used.
- **HIGH**: The issues indicate unexpected behavior of the contract. However, they alone do not defeat the purpose of the contract.
- **MEDIUM**: The issues, in the current state, do not modify the behavior of the contract but remain dangerous since any modification of apparently unrelated functions could break the contract.
- **LOW** or **LOWEST**: The issues are concerns regarding “good coding and documentation practices”, without any impact, but the simplicity for the reader to understand the code.

#1 — Unauthorized Manipulation of StakingManager

Severity: **CRITICAL**

An attacker can counterfeit a **StakingPosition**(1) output and use **StakingToken** (*issue #4*) to make it seem authentic. The attacker then can un-stake his counterfeit **StakingPosition** output and decreases `totalStake` in **StakingManager**(2) output. If the `totalStake` is low enough, the attacker can propose, pass, and execute any proposals he or she wants.
This problem will be resolved by fixing issue #4 (**StakingToken** can be duplicated). With the release of Plutus V2.0, we are able to view all redeemers from the Script Context, which will help us simplify our Validator Scripts(3) and Minting Policies(4) while avoiding more security problems. The better design will be implemented immediately after that.

#2 — Stability Pool reward cannot be withdrawn

Severity: **CRITICAL**

The validator responsible for the Stability Pool(5) always fails, not allowing users to withdraw their rewards. There is confusion when checking to block users withdraw a negative amount.

Disallowing users to withdraw a negative amount fixes this issue completely. We also prepared the testing plan for this scenario.

#3 — Protocol parameters cannot be modified

Severity: **CRITICAL**

One line of validator code requires the transaction must burn one $INDY token. But $INDY tokens can not be burned so users have no way to execute the proposals to modify protocol parameters.
While this issue is critical severity, the solution is not complicated. We simply changed one comparison operator to fix the issue.

#4 — StakingToken can be duplicated

Severity: **HIGH**

It is possible to redeem the **StakingManager** with **View** and mint a **StakingToken**, which can then be sent to any address. This can lead to vulnerabilities for the system .e.g issue #1.
We added a condition to forbid minting **StakingToken** to fix this problem.

#5 — Modifying `proposalDeposit` interacts with existing proposals

Severity: **HIGH**

`proposalDeposit` is the number of $INDY tokens that need to be deposited to open a

new proposal. If the `proposalDeposit` is updated, existing proposals will still hold the old `proposalDeposit` $INDY. However, when ending an existing proposal, the validator checks that the transaction must send the new `proposalDeposit` $INDY to the proposal's creator or **TreasuryPool**(6).
We have addressed this inconsistency and added the relevant test cases.
Moreover, we already defined the rules for changing protocol parameters in a private document but accidentally left them off the official specification:

- If `votingPeriod`(7) or `proposalDeposit`(8) is changed, it will not affect the ongoing proposals.
- If `quorum`(9) or `threshold`(10) is changed, it will affect to `inProgress` proposals, will not be applied to passed/rejected proposals.
- If `effectiveDelay`(11) or `expirationPeriod`(12) is changed, it will be applied to all ongoing proposals.

#6 — Two incompatible instances of $INDY token

Severity: **MEDIUM**

The initialization contract creates two currencies with tokens named “INDY”. This means that one could not use the $INDY they receive from the Liquidity contract to vote for the proposal.
So from Indigo’s point of view, we think this issue is only at **LOW** severity.
This issue belongs to the logic of off-chain code, and not with the real $INDY token. $INDY tokens used within the off-chain code are simulated versions of the $INDY token and used exclusively for testing. As we officially launch the platform, we will use a single $INDY token as the parameter for all contracts including Liquidity and Staking.

#7 — Minting Policies are too loose

Severity: **MEDIUM**

The minting policy in the Indigo ecosystem is that transactions consuming outputs that contain some tokens are allowed to mint the token associated with it. It is the conjunction of different contracts that is often necessary to stop tokens from being

duplicated. In other terms, a minor refactoring could cause token duplication bugs due to the complexity outlined below.

In our design, the Minting Policy scripts only check the condition in which we can mint or burn the tokens. For example, we can mint a kind of token in a transaction that spends a specific NFT or token. All logics for validating the exact minted or burned amounts are delegated to the relevant Validator scripts.

Plutus V2 with new features and improvements to refactoring and redesign all the Validator Scripts and Minting Policies. For instance, being able to view redeemers from the Script Context will help us simplify both Validator and Minting Policy scripts.

#8 — Dangerous reliance on `findInputWithToken`

Severity: **MEDIUM**

As feedback from Tweag, this function's name will cause misleading for other programmers. This is a function that expects to find exactly one input with the token from the transaction. In another case, we will raise an exception.

We agree that using a better naming convention is the needed step to improve the code readability.

#9 — Underspecified OpenPosition output

Severity: **LOW**

When creating a Collateralized Debt Position (CDP) and minting some synthetic assets, our design enables the user to send them to any address to save fees, but the specification does not mention an explicit receiver. This is the problem of lacking documentation, which is now updated in our Yellow paper.

#10 — Underspecified CloseCDP output

Severity: **LOW**

The same issue as #9 — OpenPosition, as the withdrawal value, can be sent to any address the user want. We have updated our Yellow paper and documentation to reflect this concern.

#11 — $INDY Tokens can be deposited back into TeamVesting

Severity: **LOW**

It is possible to redeem a **TeamVesting** by withdrawing a negative amount, essentially enabling us to deposit $INDY tokens back into the relevant **TeamVesting**.
We already know that it is possible to withdraw a negative amount from **TeamVesting** output before the audit, it causes no bad effect on the protocol, but to increase user experience we fixed it.

#12 — Documentation is partial

Severity: **LOW**

Some fields of datums and parameters are undocumented. This confuses and increases the chances of introducing a bug by misusing one of those fields.
We already improved our specification and design documents, also added more comments to the code.

#13 — Comments are lacking

Severity: **LOW**

The general code based is poorly commented, which makes the subtle interactions go unnoticed and could be broken by a seemingly harmless refactoring.
As mentioned in the above issues, we updated the source code with detailed comments.

#14 — Long lines harm readability

Severity: **LOWEST**

The validator scripts have long lines and this harms readability and, coupled with the lack of documentation can lead to editing mistakes.
We are using the Ormolu formatter to format the code. Ormolu does not add line breaks for the long lines automatically. Thus, we are planning to use Stylish Haskell instead to make the code more readable. Stylish Haskell is a simple Haskell code prettifier. It supports converting line endings and the line length is customizable.

Conclusion

. . .

Indigo Protocol wants to thank Tweag for their rigorous analysis of our smart-contract code base while going above and beyond providing insightful thoughts on our overall code readability and structure. The concerns Tweag has found have all been resolved, and we continue to work towards deploying our contracts on testnet soon. We are delighted to have undergone this audit and look forward to working with Tweag again in the future.

About Indigo

. . .

Indigo is an algorithmic, autonomous synthetics protocol for on-chain price exposure to real-world assets, built on Cardano. Plutus Smart Contracts enable the creation of synthetic assets called iAssets (Indigo Assets).

The minting of iAssets is decentralized and is undertaken by users throughout the network by opening a CDP and depositing collateral. Indigo ensures that there is always sufficient collateral within the protocol to cover iAssets. In the case a CDP is undercollateralized, the tokens are burnt to repay the debt.

The iAssets are economically pegged to their real-world value due to the following properties:

1. The system is designed to keep iAssets over-collateralized at all times, meaning the dollar value of the collateral exceeds the dollar value of the iAsset in CDP.
2. The iAssets are fully redeemable — users can always swap $x worth of iAsset for $x worth of their collateral (minus fees), directly with the system.
3. The system algorithmically controls the generation of iAssets through a variable issuance fee.

iAssets are freely exchangeable, meaning anyone with a Cardano wallet can send or receive iAssets, whether or not they have a CDP open.

The Indigo system also includes a decentralized governance system (Governance), as well as a decentralized data feed for prices (Oracle). The governance system can be used to vote on the minimum collateralization ratio (MCR) of any particular iAsset, as well as much more. The oracle is then used to feed prices to help the system determine if the CDP is subject to liquidation due to under-collateralization.

. . .

(1) ***StakingPosition*** *— The Unspent Transaction Output (UTXO) represents the user's staking position.*

(2) ***StakingManager*** *— The Unspent Transaction Output (UTXO) manages the creation of StakingPosition.*

(3) ***Validator Scripts*** *— The authentication mechanism in Cardano, can be used to lock the outputs on the blockchain.*

(4) ***Minting Policies*** *— The mechanism to control the minting and burning of assets.*

(5) ***StabilityPool*** *— The Stability Pool acts as the source of liquidity to repay debt from liquidated CDPs, replaces auctions, and maintains system solvency.*

(6) ***TreasuryPool*** *— The Unspent Transaction Output (UTXO) represents the treasury funds of the protocol.*

(7) ***votingPeriod*** *— The period (in blocks) in which stakers can vote on a proposal.*

(8) ***proposalDeposit*** *— The number of INDY tokens need to be deposited to open a new proposal.*

(9) ***quorum*** *— The minimum percentage of stake required to pass a proposal.*

(10) ***threshold*** *— The minimum percentage of yes votes required to pass a proposal.*

(11) ***effectiveDelay*** *— The period (in blocks) the system waits before executing a passed proposal (after a proposal passes)*

(12) ***expirationPeriod*** *— The period (in blocks) that a proposal has to be executed before being invalidated (after a proposal's voting period)*

Cardano

Crypto

Defi

Ada

Cryptocurrency

More from Indigo

Follow

Indigo is a decentralized synthetic asset issuance protocol built on Cardano

Oct 29, 2021

Indigo Is Integrating Chainlink Price Feeds to Bring Mirrored Assets to Cardano

Today, we're excited to announce that we are integrating Chainlink Price Feeds as the primary oracle solution for Indigo Protocol, a synthetic asset protocol being built on Cardano. By integrating the industry-leading oracle solution, we will have access to high-quality, tamper-proof price data needed to secure the minting, swapping, redemption...

3 min read

Oct 21, 2021

Partnership Announcement: Indigo is Accepting Ardana's Stablecoin dUSD as Collateral!

About the Partnership Today we are very pleased to announce our latest partnership with Indigo. Indigo is an autonomous synthetics protocol built on Cardano that allows users to create fungible assets known as "synthetics" that mirror the value of real-world assets. Indigo synthetics are designed to be utilized as critical components in smart...

Cardano

4 min read

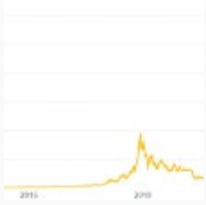
Aug 13, 2021

Indigo Protocol: The Story of Blue and Violet

Introduction Meet Blue, an avid investor who has recently become interested in synthetic assets. Blue believes that decentralization and the blockchain are powerful tools for the modern trader. ...

Cyptocurrency

8 min read



Jul 29, 2021

Indigo Protocol Tokenomics and Fair Launch

Indigo Protocol has approached the issue of both tokenomics and a fair launch from a new perspective. Our focus for Indigo has always been to gain community trust, first and foremost, building with a vision of fairness and ease of use within the protocol for the native token itself...

Cardano

5 min read

Jul 14, 2021

Indigo x SundaeSwap: Synthetic Assets, Organic DeFi ����

Today, SundaeSwap Labs and Indigo Protocol are excited to announce that we have formed a development partnership. This partnership is not only a major step forward for both protocols, but we expect that it will add great value to the growth of DeFi as a whole on Cardano by encouraging...

Crypto

2 min read

Recommended from Medium

Jasmine Jackson

@RealTryHackMe #AdventOfCyber Series: Challenge 9—Dock The Halls #TisTheSeasonForHacking

Pratik Raj

SOLID principles of OOPS

Alex Henegar

Ruby on Rails — Entry 5: Editing Resources and Creating Profiles

Amit Prajapati

15 Top web development trends in 2020

Raviya Technical

Laravel Advance | Customize pagination templates example from scratch

Vincent Blanchon

in

A Journey With Go

Go: Discovery of the Trace Package

Sumanth Samala

Docker Fundas (My Version ⁇)

Auntor Acharja

Network Scanning by Nmap