onemind

ONEMIND

SMART CONTRACT AUDIT

zokyo

November 16th 2022 | v. 1.0

Security Audit Score

PASS

Zokyo Security has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

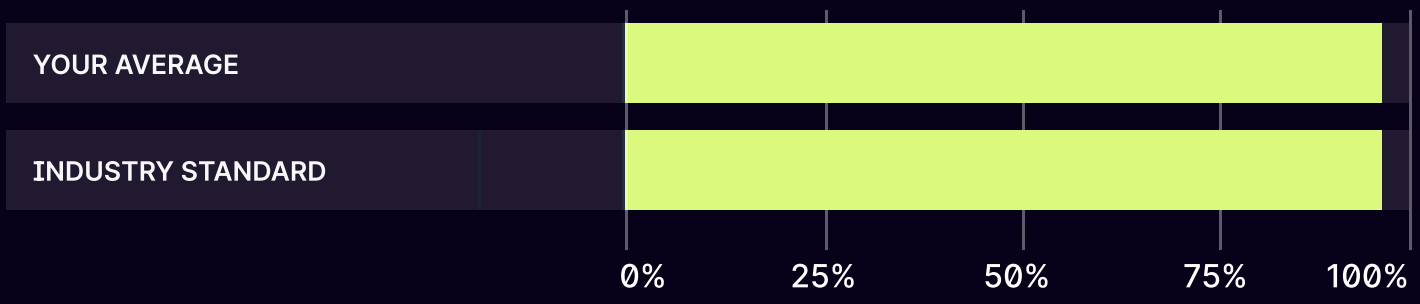
TECHNICAL SUMMARY

This document outlines the overall security of the Onemind smart contracts evaluated by the Zokyo Security team.

The scope of this audit was to analyze and document the Onemind smart contract codebase for quality, security, and correctness.

Contract Status

Testable Code



The testable code verified by Zokyo Security is sufficient to cover the industry standard.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract but rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that can withstand the Ethereum network's fast-paced and rapidly changing environment, we recommend that the Onemind team put in place a bug bounty program to encourage further active analysis of the smart contract.

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the Onemind repository: https://github.com/monterail/tokenyard-sol

Initial commit: main branch, 52ccf1db81cc7374efd0e7cca8023945c517bfd9

Final commit: main branch, 3cf4839a8f2dd837ca6dfbfd48fcbeab1954a50f

The contracts are written in Rust, prepared for the Solana blockchain and have tests prepared via the Anchor testing suite.

Within the scope of this audit, the team of auditors reviewed the following contract(s):

- **programs/auction_house**
  - auction_house/src/auction_house.rs
  - auction_house/src/auction.rs
  - auction_house/src/lib.rs
  - auction_house/src/methods/cancel_auction.rs
  - auction_house/src/methods/change_fee.rs
  - auction_house/src/methods/close_auction.rs
  - auction_house/src/methods/configure_auction.rs
  - auction_house/src/methods/create_auction.rs
  - auction_house/src/methods/initialize.rs
  - auction_house/src/methods/make_bid.rs
  - auction_house/src/methods/mod.rs
  - auction_house/src/methods/pay_creator_fee.rs
  - auction_house/src/methods/pay_platform_fee.rs
  - auction_house/src/methods/warp.rs
- **programs/nft_factory**
  - nft_factory/src/factory.rs
  - nft_factory/src/lib.rs
  - nft_factory/src/methods/create_nft.rs
  - nft_factory/src/methods/create_single_nft.rs
  - nft_factory/src/methods/freeze_nft.rs
  - nft_factory/src/methods/initialize.rs
  - nft_factory/src/methods/mod.rs
  - nft_factory/src/methods/update_factory.rs
  - nft_factory/src/methods/update_nft.rs
  - nft_factory/src/methods/verify_nft_collection.rs

During the audit, Zokyo Security ensured that the contract:

- Implements and adheres to the existing standards appropriately and effectively;
- The documentation and code comments match the logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices, efficiently using resources without unnecessary waste;
- Uses methods safe from denial of service attacks;
- Is not affected by the most resent vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of Onemind smart contracts. To do so, the code was reviewed line by line by our smart contract developers, who documented even minor issues as they were discovered. Part of this work includes writing a test suite using the Anchor testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01	Due diligence in assessing the overall code quality of the codebase.
02	Cross-comparison with other, similar smart contracts by industry leaders.
03	Testing contract logic against common and uncommon attack vectors.
04	Thorough manual review of the codebase line by line.

Executive Summary

During the audit, Zokyo Security reviewed the entire set of contracts provided by the Onemind team. The factory program is responsible for NFT minting, manipulation, and interaction with NFTs, while the Auction House program provides an NFT auction function for various types of NFTs with the ability to take commissions.

The team has checked contracts from different points of view:

- Access control, roles distribution, and the correct flow for roles assignment
- Funds flow, correct tokens transfer through the system, correct deposits and withdrawals
- Math calculations, weights for governance, correct funds calculations through the staking process
- A sequence of events during the proposals creation and approval
- A sequence of events during the Club creation and its lifetime
- Correct initialization parameters
- Main userflows within the protocol
- Edgecases analysis

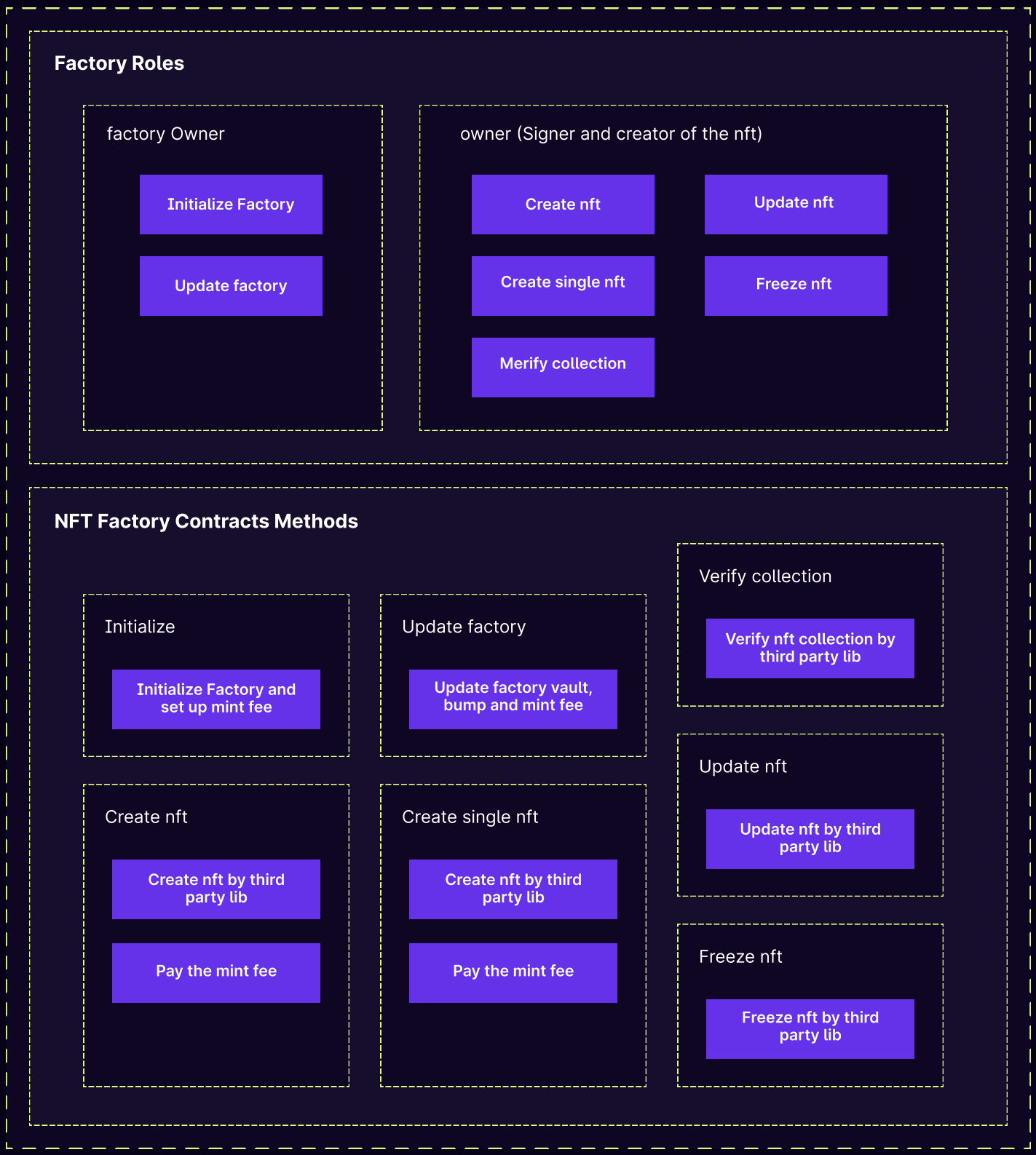
All checks were supported with additional manual tests or unit tests.

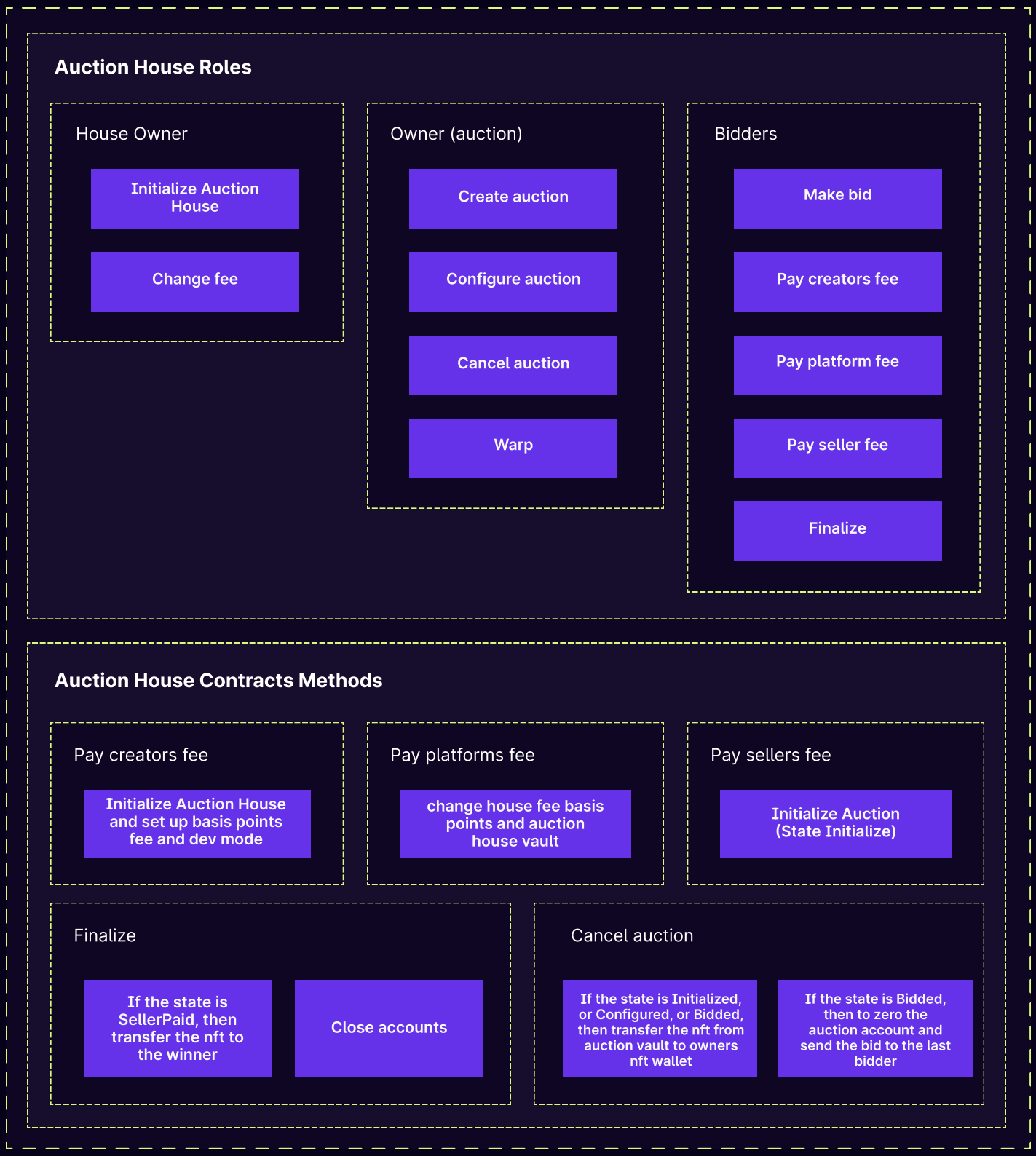
The purpose of the audit was to test the contract security against a list of common security vulnerabilities to ensure that contracts interact in the most secure and optimized way possible and to verify that smart contracts employ Rust's best practices.

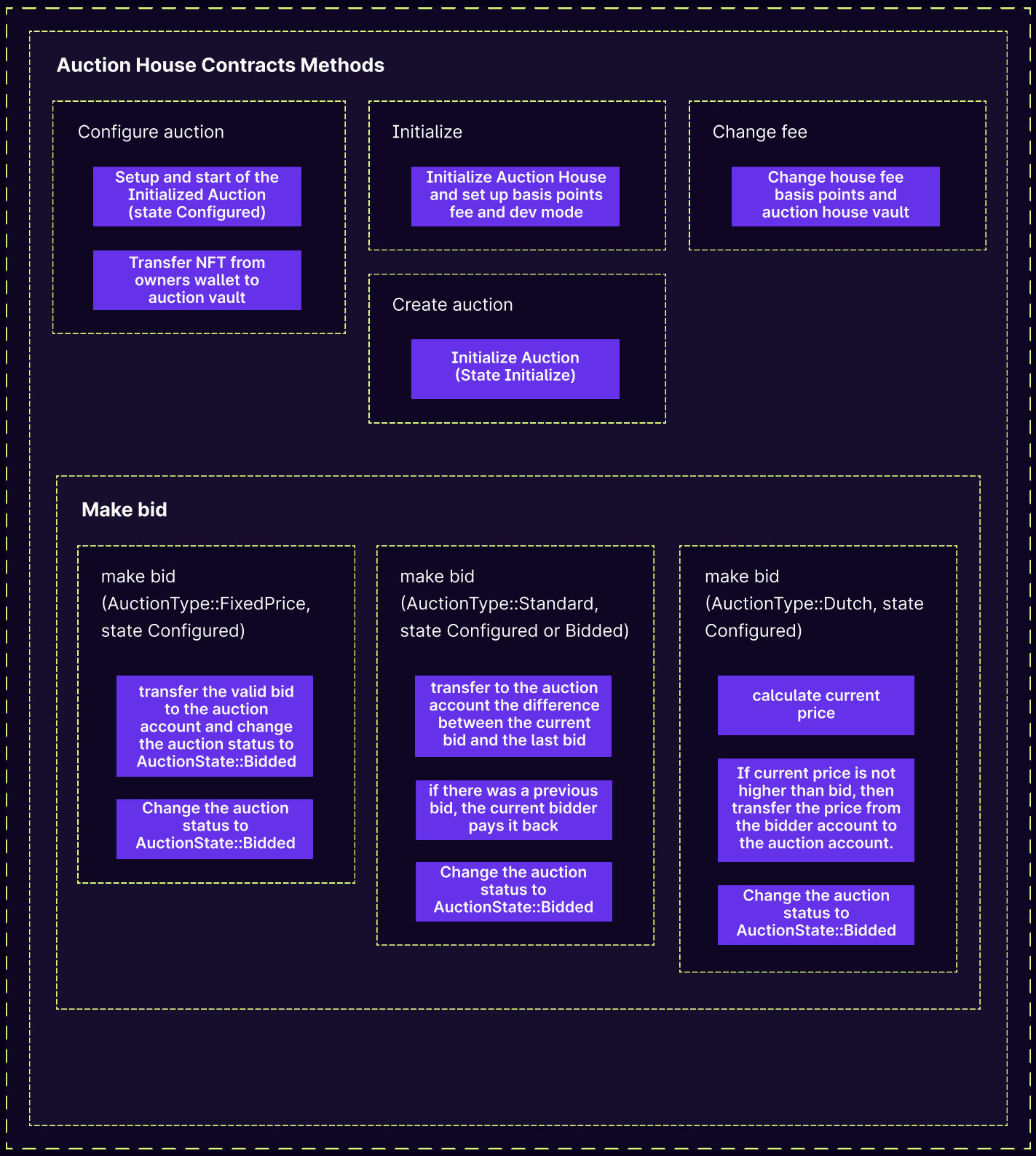
During the audit, Zokyo Security found a few issues, one of which was of high severity. Most of the problems were related to account closure and funds transfer issues, as well as the functionality of the platform owner to assign a fee, but they were successfully and promptly fixed by the Onemind team.

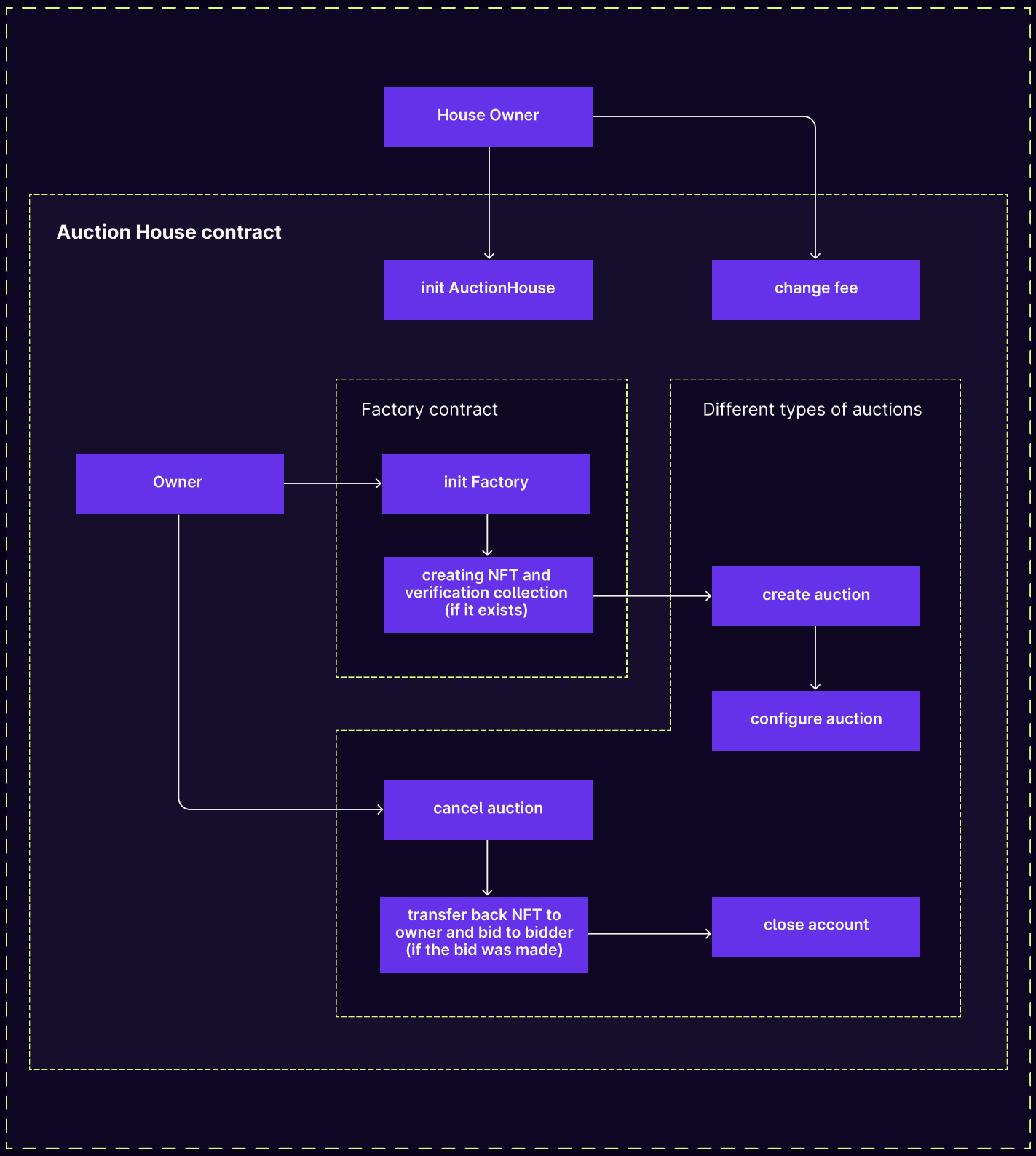
The overall security of the contracts is quite high. The contracts are rather easy to understand; they contain good documentation and comments and are well-tested by the Onemind team. Zokyo Security has prepared its own set of tests that test specific options for interacting with contracts. No critical issues were found either, and all tests passed the security check after the issues were fixed.

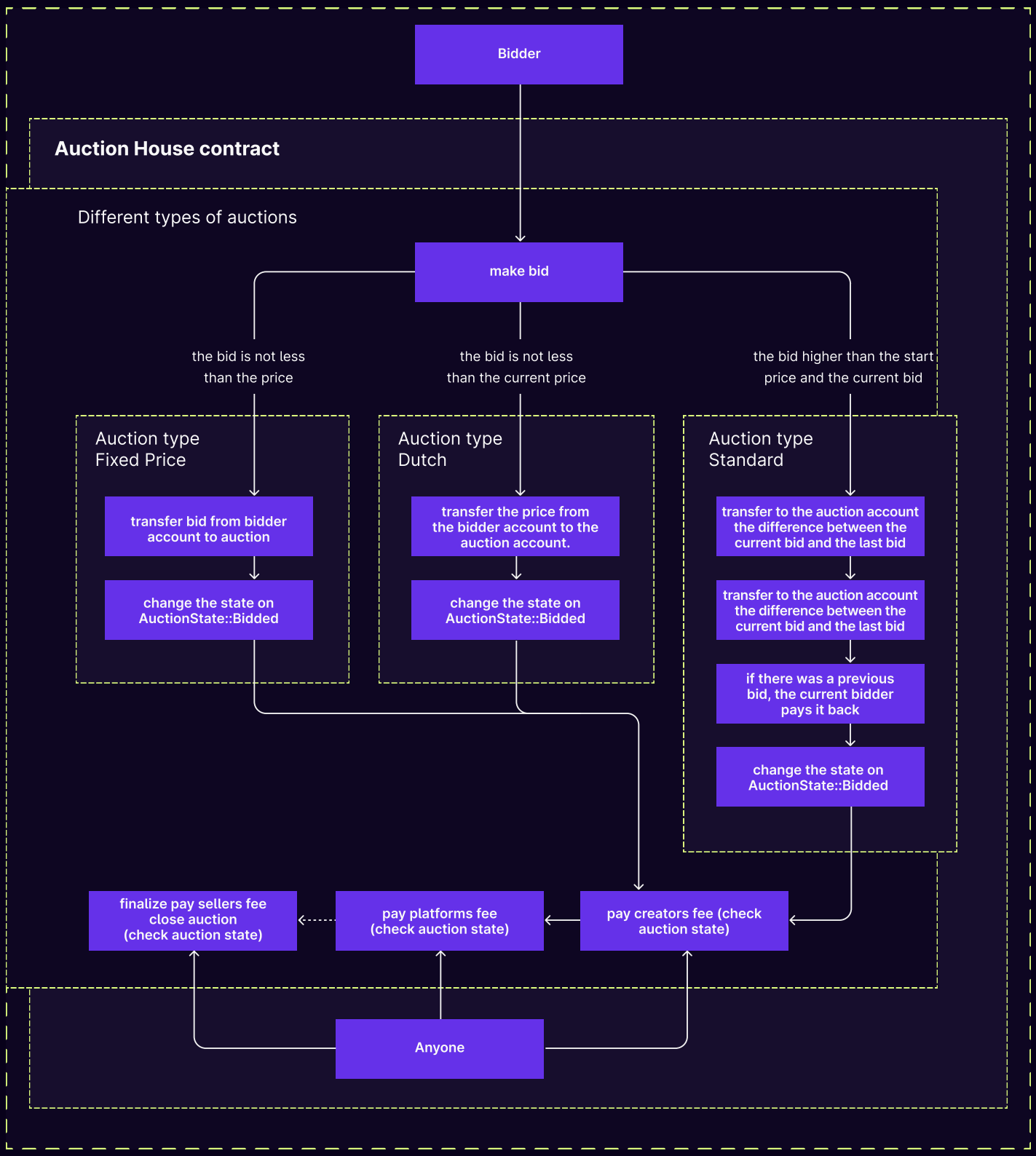
PROTOCOL OVERVIEW




Auction House Roles
House Owner
Initialize Auction House
Change fee
Owner (auction)
Create auction
Configure auction
Cancel auction
Warp
Bidders
Make bid
Pay creators fee
Pay platform fee
Pay seller fee
Finalize
Auction House Contracts Methods
Pay creators fee
Initialize Auction House and set up basis points fee and dev mode
Pay platforms fee
change house fee basis points and auction house vault
Pay sellers fee
Initialize Auction (State Initialize)
Finalize
If the state is SellerPaid, then transfer the nft to the winner
Close accounts
Cancel auction
If the state is Initialized, or Configured, or Bidded, then transfer the nft from auction vault to owners nft wallet
If the state is Bidded, then to zero the auction account and send the bid to the last bidder


Auction House Contracts Methods
Configure auction
Setup and start of the Initialized Auction (state Configured)
Transfer NFT from owners wallet to auction vault
Initialize
Initialize Auction House and set up basis points fee and dev mode
Change fee
Change house fee basis points and auction house vault
Create auction
Initialize Auction (State Initialize)
Make bid
make bid (AuctionType::FixedPrice, state Configured)
transfer the valid bid to the auction account and change the auction status to AuctionState::Bidded
Change the auction status to AuctionState::Bidded
make bid (AuctionType::Standard, state Configured or Bidded)
transfer to the auction account the difference between the current bid and the last bid
if there was a previous bid, the current bidder pays it back
Change the auction status to AuctionState::Bidded
make bid (AuctionType::Dutch, state Configured)
calculate current price
If current price is not higher than bid, then transfer the price from the bidder account to the auction account.
Change the auction status to AuctionState::Bidded


House Owner
Auction House contract
init AuctionHouse
change fee
Factory contract
Different types of auctions
Owner
init Factory
creating NFT and verification collection (if it exists)
create auction
configure auction
cancel auction
transfer back NFT to owner and bid to bidder (if the bid was made)
close account


Bidder
Auction House contract
Different types of auctions
make bid
the bid is not less than the price
the bid is not less than the current price
the bid higher than the start price and the current bid
Auction type Fixed Price
transfer bid from bidder account to auction
change the state on AuctionState::Bidded
Auction type Dutch
transfer the price from the bidder account to the auction account.
change the state on AuctionState::Bidded
Auction type Standard
transfer to the auction account the difference between the current bid and the last bid
transfer to the auction account the difference between the current bid and the last bid
if there was a previous bid, the current bidder pays it back
change the state on AuctionState::Bidded
finalize pay sellers fee close auction (check auction state)
pay platforms fee (check auction state)
pay creators fee (check auction state)
Anyone

STRUCTURE AND ORGANIZATION OF DOCUMENT

For the ease of navigation, sections are arranged from the most to the least critical one. Issues are tagged as “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Customer’s side or remains disregarded by the Customer. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

HIGH-1 | RESOLVED

The auction account may be closed before the NFT is transferred to the winner
If the user does not pay fees with a single transaction, but, for example, pays a fee via a separate transaction and calls the finalize method separately, all lamports from the auction account will be transferred. After the transfer, no lamports will remain on the auction account and the auction account will be closed. Once the auction account is closed, it becomes impossible to call the finalize method (the auction account in this case will be uninitialized), and, therefore, the winner will be left without an NFT.
The isue is marked as High since it is directly connected to the funds flow.
programs/auction_house/src/methods/pay_seller_fee.rs

Line. `40`.

Recommendation:
Merge two instructions ‘pay_seller_fee’ and ‘close_auction’ to make it impossible to execute them separately.

Post-audit:
The team of auditors has checked the business logic update and reflected that on the schemeafter the fix.

MEDIUM-1 | RESOLVED

The fee may be changed during the auction.
The platform owner can change the fee of the platform at any time without any restrictions at any moment during the existence of the auction. As a result, if this functionality is used in bad faith, you can change the fee of the auction by 100% and leave the creator of the auction with nothing.
programs/auction_house/src/methods/change_fee.rs

Recommendation:
Make the platform fee not changeable by the auction platform owner after the auction is created.

MEDIUM-2 | RESOLVED

Clarification for potentially incorrect check
During the testing stage, the team of auditors got failing conditions after the bid was placed. The team needs clarification in determining whether the statement condition is correct. Presumably, the passed bidders_wallet parameter should correspond to the bidder account, but the actual owner of the account is the System Program. Thus, the condition will always be false because the public keys of the System Program and the bidder will always differ. In this regard, if the state of the auction is in the Bidded state, it will be impossible to cancel the auction.

Line. 78, programs/auction_house/src/methods/cancel_auction.rs,
function 'handle'
The issue is marked as Medium since there is a chance to disable the Cancel functionality (though it makes no threat for user's money).

Recommendation:
We need verification from the Onemind team that this is the correct business logic. Review and clarify if the 'ctx.accounts.bidders_wallet.owner.key() != bidder' condition should be changed to 'ctx.accounts.bidders_wallet.key() != bidder'.

MEDIUM-3 | RESOLVED

Incorrect CPI and transfer call sequence
During the testing phase, the team discovered an issue with the cancellation of the auction where the bid was placed. We received a runtime error "UnbalancedInstruction". It occurred because there is a transfer of lamports before all Cross-Program Invocations are finished. In this regard, the seller is deprived of the opportunity to cancel the auction, which has already made a bid.

Line. 75-92, programs/auction_house/src/methods/cancel_auction.rs, function ‘handle’

Recommendation:
Swap out the call to the close_account function and the 'if' block.

LOW-1 | VERIFIED

There is no time check.
In the auction configuration, the team of auditors set the start and end time of the auction, but when the user makes a bid, there is no time check, therefore, it does not matter when it starts.
programs/auction_house/src/methods/make_bid.rs

Recommendation:
The team of auditors needs verification from the Onemind team that this is the correct business logic and that there is no need of time checking.

From the client:
Onemind team stated that they don't want to limit that functionality. If the user wants to create an auction starting in the past, Onemind don't want to prevent them from doing that. In the future, more distinct logic can be put on the web2 layer, but it's not necessary to limit that on the contract layer.

Post-audit.
The issue was verified by the client. Zokyo Security concluded that such functionality has no negative effect on the program.

LOW-2 | RESOLVED

Incorrect fees calculation
'ChangeFee' functionality does not have validation of passed values. Actually, the 'house_fee_basis_points' variable could have the value much more than '10000', which means that the changed fee could be up to 655% ('u16::MAX').
programs/auction_house/src/methods/change_fee.rs

Recommendation:
Create a validation that 'house_fee_basis_points' cannot be more than '10000'.

Post-audit.
It was established that 'house_fee_basis_points' is also set during the initialization of the auction and there is no validation for the passed values there. The problem was resolved in another issue.

LOW-3 | RESOLVED

Auction account is not closed
After the cancel_auction method call (in case there is no bid yet), the 'auction' account is not closed.
programs/auction_house/src/methods/cancel_auction.rs, function 'handle'

Recommendation:
The "auction" account should be closed.

INFO-1 | VERIFIED

3rd party used
In the Factory contract, all except init and update of the contract are using third-party lib, which is out of scope.
programs/nft_factory

Recommendation:
We need verification from the Onemind team that out-of-scope libs can be trusted.

From client:
The only lib contracts are using is the common library for token and metaplex helpers - included in the git repository under `libraries/common` and attached to both contracts in Cargo.toml.

Post-audit.
The 3rd-party code has been verified by the client.

INFO-2 | VERIFIED

Incorrect price transferred
In the Auction House contract, in a Dutch auction, if the user places a bid, then the current calculated price is transferred, not their bid.
programs/auction_house/src/methods/make_bid.rs

Recommendation:
We need verification from the Onemind team that this is the correct business logic.

From client:
That is intended. The calculation of the price is based on the Dutch bid. We only take enough funds to cover the need for the Dutch bid at the given point of time.

Post-audit.
The functionality has been verified by the client.

INFO-3 | RESOLVED

Missed check for variable
During the initialization of the auction, as it was said in the postaudit for low-2, there is no check for the passed ‘house_fee_basis_points’ variable.
programs/auction_house/src/methods/initialize.rs

Line. 35.

Recommendation:
Create a validation that ‘house_fee_basis_points’ cannot be more than ‘10000’.

INFO-4 | RESOLVED

Invalid funds recipient
Potentially invalid recipient of funds from closing the auction account. In the Auction House contract, at the time of closing the auction, the close_account function specifies ‘payer’ as the
destination account. Probably, the seller's account should be specified here.
programs/auction_house/src/methods/close_auction.rs

Line. 79.

Recommendation:
We need verification from the Onemind team that this is the correct business logic.

From the client:
The Client marked this as a valid issue, but by the time of reporting, it was already resolved by the Onemind team. The team changed it to send the assets back to the seller instead of the payer.

Post-audit.
The team of auditors has verified the provided update.

INFO-5 | RESOLVED

Unused auction state
Unused auction state. In the enum ‘AuctionState’, the ‘Finished’ field isn’t used anywhere further.
programs/auction_house/src/auction.rs

Line. 78.

Recommendation:
The ‘Finished’ auction state should be used further or removed.

INFO-6 | RESOLVED

Unfinished functionality
Commented functionality that isn't currently implemented.
programs/auction_house/src/lib.rs

Line. 65-94.

Recommendation:
The functionality must be implemented or comments must be removed.

Typos
Typos may confuse the user. The ‘royalities’ field has bad naming.
programs/nft_factory/src/lib.rs

Line. 28.

Recommendation:
‘royalities’ should be replaced with ‘royalties’.

	programs/auction_house	programs/nft_factory
Correct Accounts Usage Flow / Solana Accounts Confusion	Pass	Pass
Access Management Hierarchy	Pass	Pass
Types Conformity, Over/Under Flows	Pass	Pass
Unexpected Tokens	Pass	Pass
Cross-contract calls / Redeployment with cross-instance confusion	Pass	Pass
Public Visibility Methods Access	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Incorrect Parameters Attack	Pass	Pass
Unchecked Return Values	Pass	Pass
Race Conditions/Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Unsafe Rust Code/Outdated dependencies	Pass	Pass
Floating Points and Precision	Pass	Pass
Authentication	Pass	Pass
Missing signer checks / Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying token)	Pass	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by the Onemind team

As a part of our work assisting Onemind in verifying the correctness of their contract code, our team has checked the complete set of tests prepared by the Onemind team.

We need to mention that the original code has a significant original coverage with testing scenarios provided by the Onemind team. All of them were also carefully checked by the team of auditors.

Auction House

tests/auction-house-test.ts

- ✓ Auction house → changing fees
- ✓ Auction creation and fixprice buy
- ✓ Auction cancelation
- ✓ Standard auction bidding
- ✓ Standard auction bidding → failing claiming price lower than minimum price
- ✓ Auction buy (bid for dutch bid) → success after half time

NFT Factory

tests/nft-factory-test.ts

- ✓ Create NFT
- ✓ List all NFTs
- ✓ Update existing NFT
- ✓ Freezed NFT should be unchangable
- ✓ Create a collection with one NFT
- ✓ Check that we have ONE collection with one NFT in it
- ✓ Check if a collection properly identifies 3 NFTs inside

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Security

As a part of our work assisting Onemind in verifying the correctness of their contract code, our team was responsible for writing tests (including integration tests via Anchor framework).

The tests were based on the functionality of the code, as well as a review of the Onemind contract requirements for details about issuance amounts and how the system handles these.

Auction House

tests/auction-house-test.ts

✓ # Cancellation of the auction with a bid
✓ # Verification of debit and credit convergence
✓ # Change fee by more than 100%
✓ # Initialization of an auction house with a fee of more than 100%
✓ # Closing the auction account on cancellation
✓ # Changing the fee during the auction
✓ # First seller getting royalties for second NFT sell

Auxiliary functions

tests/auction-house-test.ts

getNewNFTWithRoyaltiesAndAuction

interfaces/AuctionHouse.ts

makeBid
createWallet
payPlatformsFee
finalize
cancelBiddedAuction
getAuctionFee

Tests description

✓ # Cancellation of the auction with a bid
- The test checks the cancellation of the auction in which the bid was made by the bidder.
✓ # Verification of debit and credit convergence
- Verification of convergence of funds of all actors (seller and buyer of NFT, auction house owner, buyer2 in the role of a third party, causing the payment of the platform fee and the finalization of the auction).
✓ # Change fee by more than 100%
- A fee change of more than 100% should not be possible. The test checks that if you try to set the fee above 100% by using 'change_fee', the transaction fails.

```
✓ # Initialization of an auction house with a fee of more than 100%
- The test checks that if you try to set the fee above 100% by using 'initialize', the transaction fails.
✓ # Closing the auction account on cancellation
- The test checks if the auction account is closed by using 'cancel_auction'.
✓ # Change the fee during the auction
- Checking the impossibility of changing the fee of the platform after the creation of the auction.
✓ # Seller getting royalties for the second NFT sell
- The test checks if the specified royalties are paid correctly to the original seller after the NFT is resold.
```

Each test contains additional functionality for the edge case setting up and tuning the parameters of the protocol for checking the particular case. Therefore, within each edge case test, Zokyo Security has verified the core functionality as well.

We are grateful for the opportunity to work with the Onemind team.

The statements made in this document should not be interpreted as an investment or legal advice, nor should its authors be held accountable for the decisions made based on them.

Zokyo Security recommends the Onemind team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo