V1.0

AdEx

Audit

Dean Eigenmann - ZK Labs

November 10, 2017

Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bugfree status. The audit documentation is for discussion purposes only.

Overview

The audit was performed on all smart contracts found the AdEx core repository (https://github.com/AdExBlockchain/adex-core), the contracts are found in the directory: contracts. The commit hash this audit was performed upon is: 6ecc86b2a3c3594569e3df9936f6b356b0f42d1e

The following contracts were ignored in this project. This is because they are assumed to be secure as they stem from previously audited projects or they are mocks:

- Migrations
- ADXMock

Audit Results

The smart contracts developed by the AdEx team are kept to a very high standard, due to the logical flow of how the program works, no security related bugs could be uncovered. However minor code style issues were found which the team is advised to change.

Suggestions

1. Lock the smart contracts to a specific version. Suggested is one higher than the current one being used, which is known to have fixed older compiler bugs.
2. With later planned versions of solidity, many of the getter functions for structs implemented can be eliminated, it is suggested to do this at a later stage.

Drainable.sol

The Drainable contract allows the owner to transfer all ERC20 tokens or ethereum tokens out of the smart contract.

Suggestions

1. Considering changing to spaces instead of tabs, this conforms with the language style.
2. Consider adding a new line between contract definition and first function.

withdrawToken

This function allows the withdrawal of the entire balance for a specified ERC20 token, the address is passed as a function parameter.

Suggestions

1. Move the onlyOwner modifier and the opening bracket to the same line as the function definition, this increases legibility.
2. Instead of passing an address, consider already type hinting the function parameter as ERC20.
3. Token balance does not really need to be stored in a variable
4. Consider wrapping the transfer in a require, for those ERC20 contracts that work with booleans instead of exceptions.

withdrawEther

This function allows the withdrawal of all ether stored in the contract.

Suggestions

1. Move the onlyOwner modifier and the opening bracket to the same line as the function definition, this increases legibility.
2. Consider replacing send with transfer, require can then be omitted.

ADXExchange.sol

This contract contains the core logic of the Adex platform, it implements the entire logic for bidding.

Suggestions

1. Variable “name” can most likely be omitted. If it is really required, consider a natspec comment on the contract definition.
2. Clean up ordering of functions to match solidity style guide.
3. Considering changing to spaces instead of tabs, this conforms with the language style.
4. Comments listing where function, event and modifier definitions are seem rather useless.
5. Consider summarizing “Links on publisher” and “Links on advertiser” into a common struct, so the struct Bid contains advertiser and publisher that are both a struct containing the specific info. This can be done as the info on both sides is the same, except of slot and unit, but for this the enum can be used to specify what type of data it is.
6. Consider changing the description comments above functions to natspec conforming comments.

Modifiers

onlyRegisteredAcc

Checks that msg.sender is registered in the registry.
Suggestion: Consider changing Acc to its full meaning, Account. All though generally understandable, naming full wording.

onlyBidOwner

Checks that msg.sender is listed as advertiser in bid.
Suggestion: Consider changing Owner to Advertiser, as this is actually what it checks.

onlyBidAceptee

Checks that msg.sender is listed as publisher in bid.
Suggestion: Consider changing Aceptee to Publisher, as this is actually what it checks.

onlyBidState

Checks that bid is in required state.

onlyExistingBid

Checks that bid exists by ensuring id is not equal to 0.

unonlyExistingBid

Checks that bid does not exist by ensuring id is equal to 0.
Suggestion: Consider changing the name of the modifier to something more intuitive, like onlyNotExisitingBid.

Constructor

Sets token and registry from the passed variables.

Suggestions

1. Move function opening braces to the same line as definition, conforms to solidity coding style.

placeBid

Creates a new bid with the passed data and adds it to the various bid mappings. (bidById & bidByAdunit)

Suggestions

1. Consider wrapping the token transferFrom function in a require.
2. Underscores to parameter names can be omitted.
3. If underscores are omitted, modifier and opening bracket can be moved to same line as function definition.

cancelBid

Cancels an open bid.

Suggestions

1. Change var to an explicit type, increases legibility and ease of understanding.

acceptBid

Sets bid state to accepted, adds publisher information and pushes the bid into the array found in the mapping named bidsByAdslot.

Suggestions

1. Change var to an explicit type, increases legibility and ease of understanding.

2. Remove the underscores from parameter names.
3. There is nothing preventing the advertiser and publisher from being the same person, maybe this should be considered.

giveupBid

This function sets the bid status to Cancelled, and transfers the tokens back to the advertiser from the smart contract.

Suggestions

1. The logic is identical to that found in cancelBid, consider expanding the permissions in the cancelBid function to allow for both the advertiser and the publisher to cancel, eliminating the need for this function.

verifyBid

Confirms the bid by either the publisher or advertiser, depending on who called the function. If both have called the function, the bid status is to completed.

Suggestions

1. Change var to an explicit type, increases legibility and ease of understanding.
2. The require method can be summarized in a modifier that can then be used to combine both the giveup and cancelbid methods.

claimBid

Sets the bid status as claimed, and transfers the tokens to the publisher address.

Suggestions

1. Change var to an explicit type, increases legibility and ease of understanding.

refundBid

Refunds a bid if it has an expires time set, this sets its status to expired and transfers the tokens back to the advertiser.

Suggestions

1. Change var to an explicit type, increases legibility and ease of understanding.

getBidsFromArr

Returns bids for a particular state.

Suggestions

1. Consider passing _state as Bid.State rather than uint.
2. Modifiers, return type definitions and opening brackets can all be moved to function definition line.

getAllBidsByAdunit

Returns the values from mapping bidsByAdunit.

Suggestions

1. Consider making the mapping public, then this function can be removed.

getAllBidsByAdslot

Returns the values from mapping bidsByAdslot.

Suggestions

1. Consider making the mapping public, then this function can be removed.

getBidsByAdslot

Returns the values from mapping bidsByAdslot that have been filtered by their state using the getBidsFromArr function.

Suggestions

1. Modifiers, return type definitions and opening brackets can all be moved to function definition line.

getBid

Returns a specific bid identified by the passed id.

Suggestions

1. Consider adding natspec to function to replace current comments.

getBidReports

Returns a specific bid reports identified by the passed id.

Suggestions

1. Consider adding natspec to function to replace current comments.

ADXRegistry.sol

This contract stores most of the data used in the ADXExchange.

Modifiers

onlyRegistered

Ensures that the value of the addr in the mapping accounts is not equal to 0.

register

Creates or modifies an account with the passed data.

Suggestions

1. Consider adding brackets to the if else clauses found, to increase legibility.

registerItem

Creates or modifies an item with the passed data.

Suggestions

1. Consider adding brackets to the if else clauses found, to increase legibility.

isRegistered

Ensures that the value of the addr in the mapping accounts is not equal to 0.

Suggestions

1. Modifiers, return type definitions and opening brackets can all be moved to function definition line.

getAccount

Returns account information for the passed address.

Suggestions

1. Modifiers, return type definitions and opening brackets can all be moved to function definition line.

getAccountItems

Returns account items for the passed address.

Suggestions

1. Modifiers, return type definitions and opening brackets can all be moved to function definition line.

getItem

Returns item for the passed id.

Suggestions

1. Modifiers, return type definitions and opening brackets can all be moved to function definition line.

hasItem

Ensures that the stored id for the passed id is not equal to 0.

Suggestions

1. Modifiers, return type definitions and opening brackets can all be moved to function definition line.