PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Kalao Labs

19 November 2021

paladinsec.co

info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

1 Overview

This report has been prepared for Kalao Labs on the Avalanche network. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	Kalao Labs
URL	https://kalao.io/
Platform	Avalanche
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
KalaoDirect	https://gitlab.com/kalao.dev/marketplace-contracts/-/blob/a661d4532d39aa333e4528c0c41a32e763e3a389/contracts/KalaoDirect.sol	
KalaoAuctionFactory	https://gitlab.com/kalao.dev/marketplace-contracts/-/blob/a661d4532d39aa333e4528c0c41a32e763e3a389/contracts/KalaoAuctionFactory.sol	
KalaoAuction	https://gitlab.com/kalao.dev/marketplace-contracts/-/blob/a661d4532d39aa333e4528c0c41a32e763e3a389/contracts/KalaoAuction.sol	
KalaoSignature	https://gitlab.com/kalao.dev/marketplace-contracts/-/blob/a661d4532d39aa333e4528c0c41a32e763e3a389/contracts/KalaoSignature.sol	
KalaoAuctionHouse	https://gitlab.com/kalao.dev/marketplace-contracts/-/blob/token_check/contracts/KalaoAuctionHouse.sol	

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	6	4	2	-
Medium	8	2	3	3
Low	4	4	-	-
Informational	11	6	-	5
Total	**29**	**16**	**5**	**8**

Classification of Issues

Severity	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 KalaoDirect

ID	Severity	Summary	Status
01	HIGH	Lack of whitelists for listable ERC721 contracts can result in scams	RESOLVED
02	MEDIUM	Centralized design of requiring signer	ACKNOWLEDGED
03	MEDIUM	`buyNFT` will revert if `kalao`, `community` or `seller` is a smart contract without a fallback function or with a fallback function that consumes more than 2300 gas	PARTIAL
04	LOW	Loss of precision due to division before multiplication	RESOLVED
05	LOW	Using `transferFrom` `buyNFT` could cause transfers to a contract that does not implement `onERC721Received`	RESOLVED
06	INFO	`duration` is not used in `onERC721Received`	RESOLVED
07	INFO	Royalty does not appear to have any functionality	RESOLVED

1.3.2 KalaoAuctionFactory

ID	Severity	Summary	Status
08	HIGH	Lack of whitelists for listable ERC721 contracts can result in scams	RESOLVED
09	MEDIUM	Centralized design of requiring signer	ACKNOWLEDGED
10	MEDIUM	Lack of maximum duration check allows never ending auctions	RESOLVED
11	LOW	`getAuctions` would return an error if the auctions array is too large	RESOLVED

1.3.3 KalaoAuction

ID	Severity	Summary	Status
12	HIGH	Bidders that are smart contracts with a fallback function that consumes more gas than 2300 gas will not be able to withdraw funds	PARTIAL
13	HIGH	`claimHighestBid` will revert if highest bidder is a smart contract that implements a fallback function that costs more than 2300 gas	RESOLVED
14	MEDIUM	`close` will revert if `kalao`, `community`, `royaltyBeneficiary` or `seller` is a smart contract without a fallback function or with a fallback function that consumes more than 2300 gas	PARTIAL
15	LOW	`royaltyAmount` should be capped at 97.5% of the sale amount or the auction cannot be closed	RESOLVED
16	INFO	Anyone can close an auction that has ended	ACKNOWLEDGED
17	INFO	First bid of auction cannot be the start price	RESOLVED
18	INFO	Minimum bid increment from `highestBid` can be 1 wei	ACKNOWLEDGED
19	INFO	`started` is a redundant state variable	RESOLVED
20	INFO	Comment and code behavior do not match for `claimTime`	RESOLVED
21	INFO	Closing and bidding of the auction is possible in the same timestamp	RESOLVED

1.3.4 KalaoSignature

ID	Severity	Summary	Status
22	INFO	Signer can be set to the zero address	ACKNOWLEDGED
23	INFO	Unnecessary adding of 0 in `parseParams`	ACKNOWLEDGED

1.3.5 KalaoAuctionHouse

ID	Severity	Summary	Status
24	HIGH	Lack of whitelists for listable ERC721 contracts can result in scams	RESOLVED
25	HIGH	Bidders that are smart contracts with a fallback function that consumes more gas than 2300 gas will not be able to withdraw funds	PARTIAL
26	MEDIUM	Centralized design of requiring signer	ACKNOWLEDGED
27	MEDIUM	`close` will revert if `kalao`, `community`, `royaltyBeneficiary` or `seller` is a smart contract without a fallback function or with a fallback function that consumes more than 2300 gas	PARTIAL
28	MEDIUM	Lack of maximum duration check allows never ending auctions	RESOLVED
29	INFO	Minimum bid increment from `highestBid` can be 1 wei	ACKNOWLEDGED

2 Findings

2.1 KalaoDirect

The KakaoDirect contract is a ERC721 sale contract that allows users to list their NFT for sale at a fixed price. To create a listing, a user will have to call the `safeTransferFrom` function for the ERC721 token contract, with the KalaoDirect's address as the recipient.

To purchase a listed NFT, the buyer will have to send an exact amount of native ETH as the listing's price while calling the `buyNFT` function. Listed NFTs can be cancelled by the user who listed it.

2.1.1 Privileged Roles

The following functions can be called by the owner of the contract:

- `changeCommunity`
- `changeKalao`
- `setSigner`

2.1.3 Issues & Recommendations

Issue #01	Lack of whitelists for listable ERC721 contracts can result in scams
Severity	HIGH SEVERITY
Description	The current listing process does not have any whitelist for ERC721 contracts which can be listed. As such, any ERC721 contract can be listed. A malicious user could modify the transfer functionality in an ERC721 contract and list it. When a buyer attempts to purchase it, the malicious modified transfer function would not actually transfer the token to the buyer, but the seller would still get the sale proceeds.
Recommendation	Consider adding a whitelist for ERC721 token contract addresses which can be listed for sale. This whitelist should be maintained by the Kalao team, and the team has to do due diligence to ensure that the whitelisted addresses do not contain any malicious behavior. Alternatively, whitelisting of vetted collections can be done off-chain, and displayed in the user interface. In such a case, it will be the user's responsibility to verify the NFT they are purchasing.
Resolution	RESOLVED Whitelisting is done by the signer on the backend. A check in `onERC721Received` has also been added to ensure that the msg.sender is the collection contract address encoded in the data payload.

Issue #02	Centralized design of requiring signer
Severity	MEDIUM SEVERITY
Description	A centralized signer is required to sign the data payload used in the transfer of the ERC721 token. If the centralized signer ever becomes unavailable (e.g. denial of service attacks), users will be unable to list their ERC721 tokens for sale as there is no way to get a valid signature for the data payload. Also, as it is a centralized service, the signer could modify the payload's data (e.g. price) before signing and return the modified signed payload. When the user calls the transfer of the ERC721 token with that payload, they would have to verify the payload to ensure that the parameters are as they have originally submitted for signing. The centralized service could also deny certain users from certain behavior, such as having to list above a certain price, even if such restrictions are not done in the contract.
Recommendation	Consider switching to a decentralized model where the user calls a function to list an item for sale and the contract will call the ERC721 contract to transfer the user's ERC721 token to the contract. This will require 1 more transaction before the listing for the user to approve the token for the KalaoDirect contract.
Resolution	ACKNOWLEDGED The Kalao team has acknowledged that this centralized design is required for their project.

Issue #03	**buyNFT will revert if `kalao`, `community` or `seller` is a smart contract without a fallback function or with a fallback function that consumes more than 2300 gas**
Severity	MEDIUM SEVERITY
Description	An NFT will be unpurchasable if the `kalao`, `community` or `seller` `address` is either of the following: • A smart contract that does not implement the fallback function • A smart contract that implements the fallback function, but the fallback function costs more than 2300 gas.
Recommendation	Consider using wrapped ERC20 ETH instead of native ETH for transfers to the community and seller address in buyNFT.
Resolution	PARTIALLY RESOLVED `call` is now used to send the native ETH. This issue has been partially resolved as long as the seller is a smart contract that has implemented the fallback function. Otherwise, the call will still fail. This is mitigated as the user will have to interact with the centralized API to sign the data for the transaction, so it is expected that the seller will be an EOA. Kalao and community will still use `transfer`, as the project team have stated that they will be set to addresses which will not cause the transfer to fail.

Issue #04	Loss of precision due to division before multiplication
Severity	LOW SEVERITY

Location	Line 125
	`uint256 kalaoPart = 15*(price/ (800)); // 75% of 2.5% of fees`
	
	Line 128:
	`uint256 communityPart = 5*(price/(800)); // 25% of 2.5% of fees`

Description	For the calculation of fees, division is done before multiplication, which can result in a loss of precision.
Recommendation	Do multiplication before division.
Resolution	RESOLVED Multiplication is now done before division.

Issue #05	**Using `transferFrom` buyNFT could cause transfers to a contract that does not implement `onERC721Received`**
Severity	LOW SEVERITY
Description	If the buyer of a ERC721 token listing is a smart contract that does not have `onERC721Received` implemented, the ERC721 token will still be transferred.
Recommendation	Consider using `safeTransferFrom` instead to ensure that the receiving address has `onERC721Received` implemented if it is a contract. This should only be done for `buyNFT` and not cancel, as it is unclear if the seller address has `onERC721Received` implemented during listing.
Resolution	RESOLVED `safeTransferFrom` is used for KalaoDirect. The buyer will need to implement `onERC721Received` if it is a contract.

Issue #06	**`duration` is not used in `onERC721Received`**
Severity	INFORMATIONAL
Description	It is observed that the `duration` variable is not used in `onERC721Received`.
Recommendation	Clarify if the `duration` should be used in KalaoDirect. If it is not used, there is no need to declare a local variable to store it.
Resolution	RESOLVED `duration` has been removed.

Issue #07	Royalty does not appear to have any functionality
Severity	INFORMATIONAL
Description	Although the royalty field is stored as a boolean in the Sale struct, there does not appear to be any usage for it such as determining if there are royalty fees, as the name suggests.
Recommendation	Clarify the usage for the royalty field in the Sale struct.
Resolution	RESOLVED Royalties have a functionality.

2.2 KalaoAuctionFactory

The KalaoAuctionFactory is a contract that will allow users to list ERC721 tokens for auctioning, and will store the ERC721 tokens during the course of the auction. To create an auction, a user will have to call the `safeTransferFrom` function for the ERC721 token contract, with the KalaoAuctionFactory's address as the recipient. A new contract is created by the factory for each auction.

2.2.1 Privileged Roles

The following functions can be called by the owner of the contract:

- `changeCommunity`
- `changeKalao`
- `setSigner`

2.2.2 Issues & Recommendations

Issue #08	Lack of whitelists for listable ERC721 contracts can result in scams
Severity	HIGH SEVERITY
Description	The current listing process does not have any whitelist for ERC721 contracts which can be listed. As such, any ERC721 contract can be listed. A malicious user could modify the transfer functionality in an ERC721 contract and list it. When a buyer attempts to purchase it, the malicious modified transfer function would not actually transfer the token to the buyer, but the seller would still get the sale proceeds.
Recommendation	Consider adding a whitelist for ERC721 token contract addresses which can be listed for sale. This whitelist should be maintained by the Kalao team, and the team has to do due diligence to ensure that the whitelisted addresses do not contain any malicious behavior. Alternatively, whitelisting of vetted collections can be done off-chain, and displayed in the user interface. In such a case, it will be the user's responsibility to verify the NFT they are purchasing.
Resolution	RESOLVED Whitelisting is done by the signer on the backend. A check in `onERC721Received` has also been added to ensure that the msg.sender is the collection contract address encoded in the data payload.

Issue #09	Centralized design of requiring signer
Severity	MEDIUM SEVERITY
Description	A centralized signer is required to sign the data payload used in the transfer of the ERC721 token. If the centralized signer ever becomes unavailable (e.g. denial of service attacks), users will be unable to list their ERC721 tokens for auction as there is no way to get a valid signature for the data payload. Also, as it is a centralized service, the signer could modify the payload's data (e.g. price) before signing and return the modified signed payload. When the user calls the transfer of the ERC721 token with that payload, they would have to verify the payload to ensure that the parameters are as they have originally submitted for signing. The centralized service could also deny certain users from certain behavior, such as having to list above a certain price, even if such restrictions are not done in the contract.
Recommendation	Consider switching to a decentralized model where the user calls a function to list an item for auction and the contract will call the ERC721 contract to transfer the user's ERC721 token to the contract. This will require 1 more transaction before the listing for the user to approve the token for the KalaoDirect contract.
Resolution	ACKNOWLEDGED The Kalao team has acknowledged that this centralized design is required for their project.

Issue #10	Lack of maximum duration check allows never ending auctions
Severity	MEDIUM SEVERITY
Description	In `onERC721Received`, when the auction is to be created, there is only a minimum check that the duration is non zero, but a lack of a maximum duration check. Hence, an auction can be set to go on forever by submitting a huge duration value. E.g. 315360000 will be 10 years. The highest bidder of such an auction will have his funds stuck in the contract waiting for the auction to end. The seller will also have his NFT stuck in the contract until it ends.
Recommendation	Consider adding a reasonable maximum duration (e.g. 7 days).
Resolution	RESOLVED KalaoAuction and KalaoAuctionFactory will be merged, and KalaoAuctionHouse will be used instead. A check of a maximum of 10 days has been added when creating an auction.

Issue #11	**getAuctions would return an error if the auctions array is too large**
Severity	LOW SEVERITY
Description	Although `getAuctions` is a `view` function, it can still revert with an error if the array it returns is too large. As this function is most likely used to return information about auctions in the dapp frontend, it will cause issues in doing so as time passes and the auction array keeps growing with each auction created. This is because the auction array will never decrease in length.
Recommendation	Use pagination for the `getAuctions` to limit the number of items in the array returned.

```
function getAuctions(
        uint256 cursor,
        uint256 size
    )
        external
        view
        returns (
           address[] memory,
           uint256
        )
    {
        uint256 length = size;

        if (length > auctions.length - cursor) {
            length = auctions.length - cursor;
        }

        address[] memory values = new address[](length);

        for (uint256 i = 0; i < length; i++) {
            values[i] = auctions[cursor + i];
        }

        return (values, cursor + length);
    }
```

Resolution	RESOLVED This function has been removed.

2.3 KalaoAuction

Users can bid on a listed auction for an ERC721 token by calling the `bid` function and sending native ETH greater than 0.01 ETH. The address' bid will then be incremented by the amount of ETH sent. The cumulative bid will have to be higher than the current `highestBid`, or starting price, whichever is higher. If a successful bid is done in the last 5 minutes before the ending time, the end and claim time will both be extended by 5 minutes. Users will be able to withdraw their bids as long as they are not the highest bidder.

After the ending time of the auction has passed, the seller will be able to close it, obtain the proceeds of the auction and send the NFT to the highest bidder. If there are no bids, the seller will receive the NFT back.

If the `claimTime` time passes and the auction has not been closed, the highest bidder can withdraw the funds used for bidding. Similarly, the seller can withdraw the NFT from the auction.

2.3.1 Issues & Recommendations

Issue #12	**Bidders that are smart contracts with a fallback function that consumes more gas than 2300 gas will not be able to withdraw funds**
Severity	HIGH SEVERITY
Description	A bidder will not be able to withdraw funds used for bidding if it is a smart contract that implements a fallback function that costs more than 2300 gas. This is because send will return `false` due to failing as it does not have enough gas forwarded. This will result in such bidders having their funds stuck once they are outbidded. Reference: https://consensys.net/diligence/blog/2019/09/stop-using-solidityss-transfer-now/
Recommendation	Consider using wrapped ERC20 ETH instead of native ETH for transfers. Alternatively, use `call` instead, and add the `nonReentrant` modifier on all `external` functions.
Resolution	PARTIALLY RESOLVED `call` is now used to send the native ETH. This issue has been partially resolved as long as the seller is a smart contract that has implemented the fallback function. Otherwise, the call will still fail. Users who intend to bid by interacting with the auction contract using a smart contract are advised to implement a fallback function.

Issue #13	**`claimHighestBid` will revert if highest bidder is a smart contract that implements a fallback function that costs more than 2300 gas**
Severity	HIGH SEVERITY
Description	If the auction is not closed and the `closeTime` has passed, the highest bidder will not be able to withdraw funds used for bidding if it is a smart contract that implements a fallback function that costs more than 2300 gas. This is because `transfer` will revert as it does not have enough gas forwarded. This will result in such highest bidders having their funds stuck for unclosed auctions. Reference: https://consensys.net/diligence/blog/2019/09/stop-using-soliditys-transfer-now/
Recommendation	Consider using wrapped ERC20 ETH instead of native ETH for transfers. Alternatively, use `call` instead, and add the `nonReentrant` modifier on all `external` functions.
Resolution	RESOLVED `call` is now used to send the native ETH and the `nonReentrant` modifier has been added. Users who intend to bid by interacting with the auction contract using a smart contract are advised to implement a fallback function.

Issue #14	**`close` will revert if `kalao`, `community`, `royaltyBeneficiary` or `seller` is a smart contract without a fallback function or with a fallback function that consumes more than 2300 gas**
Severity	MEDIUM SEVERITY
Description	An auction with a valid bid will not be closable if the `kalao`, `community`, `royaltyBeneficiary` or `seller` address is either of the following: • A smart contract that does not implement the fallback function • A smart contract that implements the fallback function, but the fallback function costs more than 2300 gas.
Recommendation	Consider using wrapped ERC20 ETH instead of native ETH for transfers of the fees and proceeds.
Resolution	PARTIALLY RESOLVED `call` is now used to send the native ETH to the seller and `royaltyBeneficiary`. This issue has been partially resolved as long as the receiver is a smart contract that has implemented the fallback function. Otherwise, the call will still fail. For the `royaltyBeneficiary`, if the call fails, no royalties will be deducted from the sale. For the seller, this is mitigated as the user will have to interact with the centralized API to sign the data for the transaction, so it is expected that the seller will be an EOA. Kalao and community will still use `transfer`, as the project team have stated that they will be set to addresses which will not cause the transfer to fail.

Issue #15	**`royaltyAmount` should be capped at 97.5% of the sale amount or the auction cannot be closed**
Severity	LOW SEVERITY
Location	Line 173 `uint256 amount = highestBid - kalaoPart - royaltyAmount - communityPart;`
Description	The sale proceeds that will be sent to the seller is calculated as the above, subtracting all the fees from the `highestBid` amount. As `kalaoPart + communityPart` are fixed at 2.5% in total, if `royaltyAmount` exceeds 97.5%, it will result in a revert due to underflow.
Recommendation	Add a check to ensure that the `royaltyAmount` is at most 97.5% of the `highestBid`.
Resolution	RESOLVED Royalties are now capped at 97%.

Issue #16	**Anyone can close an auction that has ended**
Severity	INFORMATIONAL
Description	Anyone can call the `close` function to close the auction, sending the funds to the seller and the NFT to the highest bidder.
Recommendation	Verify if the auction should be closable by anyone, or if only the seller should be able to close it.
Resolution	ACKNOWLEDGED

Issue #17	First bid of auction cannot be the start price
Severity	INFORMATIONAL
Location	Line 105 `require(newBid > startingPrice, "Starting price is higher");`
Description	The above line enforces that the first bid has to be larger than the starting price.
Recommendation	Verify if the intention is to make the first bid always be greater than the starting price. If it should allow bidding at the starting price, then the > should be changed to >=.
Resolution	RESOLVED Bidding at the starting price is now allowed, by changing > to >=.

Issue #18	Minimum bid increment from `highestBid` can be 1 wei
Severity	INFORMATIONAL
Description	There is only a check that a user's bid will be greater than the `highestBid`, but as long as the user's bid is 1 wei greater than the `highestBid`, it can be accepted. **Example:** User A has bidded 1 ETH for the auction. Some time has passed and the highest bid is 2 ETH. User A will just have to bid and send 1 ETH and 1 wei as the transaction's value, and will be able to outbid the highest bid. The highest bid is now 2 ETH and 1 wei.
Recommendation	Verify if there should be a minimum bid increment from the `highestBid`.
Resolution	ACKNOWLEDGED

Issue #19	`started` is a redundant state variable
Severity	INFORMATIONAL
Description	`started` will always be true once the auction is deployed by the factory as it is set to `true` in the constructor.
Recommendation	Remove usages of `started`.
Resolution	RESOLVED `started` has been removed.

Issue #20	Comment and code behavior do not match for `claimTime`
Severity	INFORMATIONAL
Location	Line 82 `claimTime = endTime + 1209600; // 3 days`
Description	Although the comment states 3 days, the added time to the `endTime` is 14 days.
Recommendation	Clarify the actual amount of time after `endTime` the `claimTime` will be, and update the code or comment to the correct value.
Resolution	RESOLVED KalaoAuction and KalaoAuctionFactory will be merged, and KalaoAuctionHouse will be used instead. The comment in KalaoAuctionHouse has been updated.

Issue #21	Closing and bidding of the auction is possible in the same timestamp
Severity	INFORMATIONAL

Location	Line 101
	`require(block.timestamp <= endTime, "Bidding is ended");`
	
	Line 140
	`require(block.timestamp >= endTime, "Auction is not ended yet");`

Description	As bidding and closing both are allowed if the timestamp is the same as the `endTime`, it would be possible to do both in the same block. This could cause a valid bid to be unsuccessful if it happens to be in the same block with a timestamp equal to that of the `endTime`, but with the close transaction being executed first.
Recommendation	Consider allowing bidding only before `endTime` has passed. `require(block.timestamp < endTime, "Bidding is ended");`
Resolution	RESOLVED KalaoAuction and KalaoAuctionFactory will be merged, and KalaoAuctionHouse will be used instead. The change will be made to the close from `>= endTime` to `> endTime`.

2.4 KalaoSignature

This contract is inherited by the other Kalao contracts, and used for its utility functions to deal with parsing parameters, checking signatures, and generating hashes.

2.4.1 Issues & Recommendations

Issue #22	Signer can be set to the zero address
Severity	INFORMATIONAL
Description	There is a lack of zero address check in `setSigner`, which allows the signer to be set to the zero address.
Recommendation	Add a check to ensure that the signer parameter passed to `_setSigner` is not the zero address.
Resolution	ACKNOWLEDGED

Issue #23	Unnecessary adding of 0 in `parseParams`
Severity	INFORMATIONAL
Location	Line 54 `seller := mload(add(add(data, 32), 0))`
Description	The following addition of 0 is unnecessary as the result is the same without it.
Recommendation	The unnecessary addition can be removed. `seller := mload(add(data, 32))`
Resolution	ACKNOWLEDGED

2.5 KalaoAuctionHouse

The KalaoAuctionHouse is a contract that will allow users to list ERC721 tokens for auctioning, and will store the ERC721 tokens during the course of the auction. To create an auction, a user will have to call the safeTransferFrom function for the ERC721 token contract, with the KalaoAuctionHouse's address as the recipient.

Users can bid on a listed auction for an ERC721 token by calling the bid function and sending native ETH greater than 0.01 ETH. The address' bid will be incremented by the amount of ETH sent. The cumulative bid will have to be higher than the current highestBid, or starting price, whichever is higher. If a successful bid is done in the last 5 minutes before the ending time, the end and claim time will both be extended by 5 minutes. Users will be able to withdraw their bids as long as they are not the highest bidder.

After the ending time of the auction has passed, the seller will be able to close it, obtain the proceeds of the auction and send the NFT to the highest bidder. If there are no bids, the seller will receive the NFT back.

If the claimTime time passes and the auction has not been closed, the highest bidder can withdraw the funds used for bidding. Similarly, the seller can withdraw the NFT from the auction.

2.5.1 Privileged Roles

The following functions can be called by the owner of the contract:

- `changeCommunity`
- `changeKalao`
- `setSigner`

2.5.2 Issues & Recommendations

Issue #24	Lack of whitelists for listable ERC721 contracts can result in scams
Severity	HIGH SEVERITY
Description	The current listing process does not have any whitelist for ERC721 contracts which can be listed. As such, any ERC721 contract can be listed. A malicious user could modify the transfer functionality in an ERC721 contract and list it. When a buyer attempts to purchase it, the malicious modified transfer function would not actually transfer the token to the buyer, but the seller would still get the sale proceeds.
Recommendation	Consider adding a whitelist for ERC721 token contract addresses which can be listed for sale. This whitelist should be maintained by the Kalao team, and the team has to do due diligence to ensure that the whitelisted addresses do not contain any malicious behavior. Alternatively, whitelisting of vetted collections can be done off-chain, and displayed in the user interface. In such a case, it will be the user's responsibility to verify the NFT they are purchasing.
Resolution	RESOLVED Whitelisting is done by the signer on the backend. A check in `onERC721Received` has also been added to ensure that the `msg.sender` is the collection contract address encoded in the data payload.

Issue #25	**Bidders that are smart contracts with a fallback function that consumes more gas than 2300 gas will not be able to withdraw funds**
Severity	HIGH SEVERITY
Description	A bidder will not be able to withdraw funds used for bidding if it is a smart contract that implements a fallback function that costs more than 2300 gas. This is because send will return `false` due to failing as it does not have enough gas forwarded. This will result in such bidders having their funds stuck once they are outbidded. Reference: https://consensys.net/diligence/blog/2019/09/stop-using-soliditys-transfer-now/
Recommendation	Consider using wrapped ERC20 ETH instead of native ETH for transfers. Alternatively, use `call` instead, and add the `nonReentrant` modifier on all external functions.
Resolution	PARTIALLY RESOLVED `call` is now used to send the native ETH. This issue has been partially resolved as long as the seller is a smart contract that has implemented the fallback function. Otherwise, the call will still fail. Users who intend to bid by interacting with the auction contract using a smart contract are advised to implement a fallback function.

Issue #26	Centralized design of requiring signer
Severity	MEDIUM SEVERITY
Description	A centralized signer is required to sign the data payload used in the transfer of the ERC721 token. If the centralized signer ever becomes unavailable (e.g. denial of service attacks), users will be unable to list their ERC721 tokens for auction as there is no way to get a valid signature for the data payload. Also, as it is a centralized service, the signer could modify the payload's data (e.g. price) before signing and return the modified signed payload. When the user calls the transfer of the ERC721 token with that payload, they would have to verify the payload to ensure that the parameters are as they have originally submitted for signing. The centralized service could also deny certain users from certain behavior, such as having to list above a certain price, even if such restrictions are not done in the contract.
Recommendation	Consider switching to a decentralized model where the user calls a function to list an item for auction and the contract will call the ERC721 contract to transfer the user's ERC721 token to the contract. This will require 1 more transaction before the listing for the user to approve the token for the KalaoAuctionHouse contract.
Resolution	ACKNOWLEDGED The Kalao team has acknowledged that this centralized design is required for their project.

Issue #27	**`close` will revert if `kalao`, `community`, `royaltyBeneficiary` or `seller` is a smart contract without a fallback function or with a fallback function that consumes more than 2300 gas**
Severity	MEDIUM SEVERITY
Description	An auction with a valid bid will not be closable if the kalao or community, is either of the following: - A smart contract that does not implement the fallback function - A smart contract that implements the fallback function, but the fallback function costs more than 2300 gas. Also, an auction with a valid bid will not be closable if the `royaltyBeneficiary` or seller address is the following: - A smart contract that does not implement the fallback function
Recommendation	Consider using wrapped ERC20 ETH instead of native ETH for transfers of the fees and proceeds.
Resolution	PARTIALLY RESOLVED `call` is now used to send the native ETH to the seller and `royaltyBeneficiary`. This issue has been partially resolved as long as the receiver is a smart contract that has implemented the fallback function. Otherwise, the call will still fail. For the `royaltyBeneficiary`, if the call fails, no royalties will be deducted from the sale. For the seller, this is mitigated as the user will have to interact with the centralized API to sign the data for the transaction, so it is expected that the seller will be an EOA. Kalao and community will still use transfer, as the project team have stated that they will be set to addresses which will not cause the transfer to fail.

Issue #28	Lack of maximum duration check allows never ending auctions
Severity	MEDIUM SEVERITY
Description	In `onERC721Received`, when the auction is to be created, there is only a minimum check that the duration is non zero, but a lack of a maximum duration check. Hence, an auction can be set to go on forever by submitting a huge duration value. E.g. 315360000 will be 10 years. The highest bidder of such an auction will have his funds stuck in the contract waiting for the auction to end. The seller will also have his NFT stuck in the contract until it ends.
Recommendation	Consider adding a reasonable maximum duration (e.g. 7 days).
Resolution	RESOLVED A check of a maximum of 10 days has been added when creating an auction.

Issue #29	Minimum bid increment from `highestBid` can be 1 wei
Severity	INFORMATIONAL
Description	There is only a check that a user's bid will be greater than the `highestBid`, but as long as the user's bid is 1 wei greater than the highestBid, it can be accepted. Example: User A has placed a bid of 1 ETH for the auction. Some time has passed and the highest bid is 2 ETH. User A will just have to bid and send 1 ETH and 1 wei as the transaction's value, and will be able to outbid the highest bid. The highest bid is now 2 ETH and 1 wei.
Recommendation	Verify if there should be a minimum bid increment from the `highestBid`.
Resolution	ACKNOWLEDGED


PALADIN
BLOCKCHAIN SECURITY