zkHoldem Smart Contract
Audit Report

Wed Dec 20 2023

✉ contact@scalebit.xyz https://twitter.com/scalebit_

ScaleBit

zkHoldem Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	On-chain Texas Hold'em, powered by ZKP
Type	Game
Auditors	ScaleBit
Timeline	Mon Dec 04 2023 - Fri Dec 15 2023
Languages	Solidity
Platform	zkSync Era
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/zkHoldem/zkHoldem-contract
Commits	d267644088522ce643972533be0b117cef7d709b

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
BMV	contracts/BoardManagerView.sol	f9ac311d4e6bf89b28b973833372b43ae5f5fbdd
LIB	contracts/library.sol	d5ad88ffaf44136d17cb25bd64e2ace726ec2a9c
ISH	contracts/shuffle/IShuffle.sol	f42671ea06bb30c1901f565a2541668d913a492a
IAM	contracts/account/IAccountManager.sol	7458f3c8bb032cf6f45e52862e0823b8bd8cd8b6
TYP	contracts/Types.sol	4778119e2f78939edac7607a479a99066dd79de5
ICM	contracts/chip/IChipManager.sol	3ea75d6f7140b821cd3bc8baaa82d07150afcc4c
ZKT	contracts/ZKT.sol	6a6efd2d59bf33c58437479b1f20f83cdaec2a60
IPE	contracts/pokerEvaluator/IPokerEvaluator.sol	b9059488a81428191dd3e341495c4d383c5f11fd
EV7	contracts/pokerEvaluator/Evaluator7.sol	a45b1e84e8581793cc042068fda4720876aaff8f
FL2	contracts/pokerEvaluator/flush/Flush2.sol	7ea46efdfce5777f640cde66db916a6e11c07aea
FL1	contracts/pokerEvaluator/flush/Flush1.sol	f105bac633a1ad49a00fa3da9fe5f2ca1480495c

FL3	contracts/pokerEvaluator/flush/Flush3.sol	57b85909cc530808fac9884acec3a72428dc5a57
NF6	contracts/pokerEvaluator/noFlush/NoFlush6.sol	5f33598d46a4c25706eb23014f59787e469edd23
NF4	contracts/pokerEvaluator/noFlush/NoFlush4.sol	2f2749acb406c65a80ec1fbf1456095e5747f24b
NF1	contracts/pokerEvaluator/noFlush/NoFlush1.sol	1a3ae76525bd98454e6d0238a5ce583645d6165a
NF1	contracts/pokerEvaluator/noFlush/NoFlush11.sol	f0ab0f04cee174c1e41564dc8921ea3aead2419d
NF2	contracts/pokerEvaluator/noFlush/NoFlush2.sol	8c03247d3d88968a6d2881ebbaaca6c96a07330c
NF5	contracts/pokerEvaluator/noFlush/NoFlush5.sol	4642b258c1ab3bb4406dcbf330b47254017f2f39
NF1	contracts/pokerEvaluator/noFlush/NoFlush10.sol	be066fbab2030629ba367f4e48e2d79de85d115f
NF1	contracts/pokerEvaluator/noFlush/NoFlush12.sol	1d5e5a1fef487c8bd115b7003c87cfa9104e11d5
NF8	contracts/pokerEvaluator/noFlush/NoFlush8.sol	e79e2679acd22ef5c7d73a0284d9a827caeb494c
NF1	contracts/pokerEvaluator/noFlush/NoFlush14.sol	971e9d77a101a117bdad31c6b8924fc5587519f9
NF1	contracts/pokerEvaluator/noFlush/NoFlush17.sol	ed1431c209f807c4becd0eeb42f1c70bd9d098d8
NF1	contracts/pokerEvaluator/noFlush/NoFlush13.sol	63e0669f97530c754424f4286816ab9311e9055c

NF1	contracts/pokerEvaluator/noFlush/NoFlush15.sol	7df38a53371e58a9c64e24c69725789ffd398a23
NF7	contracts/pokerEvaluator/noFlush/NoFlush7.sol	636014135341365ec067125eaa09d792139c43de
NF1	contracts/pokerEvaluator/noFlush/NoFlush16.sol	cde24f6ee09ac8f861bc100b4dd325a492916ae4
NF3	contracts/pokerEvaluator/noFlush/NoFlush3.sol	68ce7e00fcc1a6f0463296635da328050d73bd70
NF9	contracts/pokerEvaluator/noFlush/NoFlush9.sol	f43f2977aeab7a2a76497e49e9b9309317fe8770
DTA	contracts/pokerEvaluator/DpTables.sol	a8293990cc2af275d5ebacc0104d5e65cb59b184
UTI	contracts/utility/Utility.sol	b867b573834f491436ea5d55c5a93b42ff316850
IUT	contracts/utility/IUtility.sol	c0bda92e13815b47384dfba842185a9b3d4cdb24
SHU	contracts/shuffle/Shuffle.sol	6b729a43c4638ffef2e6dc70acf30fb348c3a928
CHI	contracts/account/Chip.sol	648a9f409b1fdd901cbe31196b37c5ec1e1b3654
ICH	contracts/account/IChip.sol	8e64234fa0889fd08ec3412dc9b656f69f17df14
CMA	contracts/chip/ChipManager.sol	a4c7cd243019d910f66389e627028278a13d25e5
MUL	contracts/multisig/Multisig.sol	71c6b2369341e437b4a199f6e48be7eeef3d43ac

BMA	contracts/BoardManager.sol	55a563da9b3fc1032864321c748374c51c6d60f0
IBM	contracts/IBoardManager.sol	e4bda3e50efb96e1082be7276c4dee5db67a7fd0

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	13	0	12
Informational	4	0	4
Minor	2	0	2
Medium	3	0	3
Major	3	0	3
Critical	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by zkHoldem to identify any potential issues and vulnerabilities in the source code of the zkHoldem smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 13 issues of varying severity, listed below.

ID	Title	Severity	Status
CHI-1	Stable Token May Have Different Value	Medium	Acknowledged
HNF-1	`Initialize` Could Be Front-Run	Major	Acknowledged
HNF-2	Pseudo-random in `mint()`	Major	Acknowledged
HNF-3	Potential Gas Waste Due to Unoptimized State Modifications	Medium	Acknowledged
HNF-4	Lack of Events Emit	Minor	Acknowledged
HNF-5	Lack of Validation for Zero Address	Informational	Acknowledged
HNF-6	Unused Constant	Informational	Acknowledged
HNF-7	Same TokenURI Applied to Different TokenIDs	Discussion	Acknowledged
MUL-1	Use `abi.encode` instead of `abi.encodePacked`	Medium	Acknowledged
MUL-2	Unused Global Variables	Minor	Acknowledged
MUL-3	Lack `indexed` In Event	Informational	Acknowledged

MUL-4	Use Calldata Instead of Memory for Function Arguments That Do not Get Mutated	Informational	Acknowledged
ZKT-1	ZKT Can Be Minted Infinitely	Major	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the zkHoldem Smart Contract:

Admin

- `registerContract(address addr)` : Registers a contract.
- `unregisterContract(address addr)` : Unregisters a contract.
- `updateBoardManagerSettings` : Update the board settings.
- `updateAccountManagerSettings` :Update `AccountManager` settings.
- `setConfigs` :Set the nft configs.
- `setBaseURI` :Customize the base URI of the whole set of the NFT.
- `add/removeBlacklist` : Manage the Chip Blacklist.
- `updateRatio` : Update the ratio.

User

- `deposit(uint256 tokenAmount)` : Deposits ERC20 tokens for chips.
- `withdraw(uint256 chipAmount)` : Withdraws chips for ERC20 tokens. Note that `withdraw` takes `chipAmount` but `deposit` takes `tokenAmount` .
- `claim() -> (uint256)` : Claims matured `withhold` s to `chipEquity` and returns the amount of unmatured chips.
- `authorize(address ephemeralAccount)` : Authorizes an ephemeral address.
- `hasAuthorized(address permanentAccount, address ephemeralAccount) -> bool` : Checks if `permanentAccount` has authorized `ephemeralAccount` .
- `getChipEquityAmount(address player) -> uint256` : Gets the amount of chip equity.
- `getCurGameId(address player) -> uint256` : Gets the current game id of `player` .
- `getLargestGameId() -> uint256` : Gets the largest game id.
- `join(address player, uint256 gameId, uint256 buyIn, bool isNewGame)` : Joins a game with `gameId` , `buyIn` , and `isNewGame` on whether joining a new game or an existing game.

- `settle(address player, uint256 gameId, uint256 amount, bool isPositive, bool removeDelay)`: Settles chips for `player` and `gameId` by adding `amount` if `isPositive` and subtracting `amount` otherwise. Chips are immediately repaid to `chipEquity` if `removeDelay`.

4 Findings

CHI-1 Stable Token May Have Different Value

Severity: Medium

Status: Acknowledged

Code Location:

contracts/account/Chip.sol#105

Descriptions:

Actually is that not all stablecoins have a value that corresponds to $1, and they sometimes have problems with decoupling causing a token to be worth less than 1$. Also due to the fact that `buy` always uses the same `stableTokenBuyRatio` , it can lead to a different actual price for the purchase of the chip.

Suggestion:

It is recommended to use different `stableTokenBuyRatio` for different stablecoins or need to keep updating `stableTokenBuyRatio` .

HNF-1 `Initialize` Could Be Front-Run

Severity: Major

Status: Acknowledged

Code Location:

contracts/nft/HoldemNFT.sol#25;

contracts/multisig/Multisig.sol#32

Descriptions:

In the contract, by calling the `initialize` function to initialize the contracts, there is a potential issue that malicious attackers preemptively call the initialize function to initialize and there is no access control verification for the initialize functions.

Suggestion:

It is suggested that the `initialize` function can be called only by privileged addresses or be called in the same transaction immediately after the contract is created to avoid being maliciously called by the attacker.

HNF-2 Pseudo-random in `mint()`

Severity: Major

Status: Acknowledged

Code Location:

contracts/nft/HoldemNFT.sol#146

Descriptions:

In the `mint()` method, the value returned by the `random()` method is not a truly random number; instead, it is a deterministic value calculated based on input values such as salt1, salt2, and block number.

Suggestion:

It is recommended to confirm if aligns with the design.

HNF-3 Potential Gas Waste Due to Unoptimized State Modifications

Severity: Medium

Status: Acknowledged

Code Location:

contracts/nft/HoldemNFT.sol#173,179,192,196

Descriptions:

When modifying certain states, the current state is not considered, which may result in a waste of gas.

Suggestion:

It is recommended to modify the state only when the state is changed.

HNF-4 Lack of Events Emit

Severity: Minor

Status: Acknowledged

Code Location:

contracts/nft/HoldemNFT.sol#192,196;

contracts/account/Chip.sol#53-85

Descriptions:

The smart contract lacks appropriate events for monitoring sensitive operations, which could make it difficult to track sensitive actions or detect potential issues.

Suggestion:

It is recommended to emit events for those sensitive functions.

HNF-5 Lack of Validation for Zero Address

Severity: Informational

Status: Acknowledged

Code Location:

contracts/nft/HoldemNFT.sol#38

Descriptions:

There is no check for the zero address.

Suggestion:

It is recommended to add a check for the zero address.

HNF-6 Unused Constant

Severity: Informational

Status: Acknowledged

Code Location:

contracts/nft/HoldemNFT.sol#94

Descriptions:

The linked constants are not used throughout the entire contract.

Suggestion:

It is recommended to remove unused constants if there's no further design.

HNF-7 Same TokenURI Applied to Different TokenIDs

Severity: Discussion

Status: Acknowledged

Code Location:

contracts/nft/HoldemNFT.sol#55

Descriptions:

In the function `tokenURI()`, if the `unifiedTokenUrl` is true, the return value will always be the same.

Suggestion:

It is recommended to confirm if it aligns with the design.

MUL-1 Use `abi.encode` instead of `abi.encodePacked`

Severity: Medium

Status: Acknowledged

Code Location:

contracts/multisig/Multisig.sol#98

Descriptions:

Use `abi.encode()` instead which will pad items to 32 bytes, which will prevent hash collisions (e.g. `abi.encodePacked(0x123,0x456)` => `0x123456` => `abi.encodePacked(0x1,0x23456)`, but `abi.encode(0x123,0x456)` => `0x0...1230...456`). Unless there is a compelling reason, `abi.encode` should be preferred.

Suggestion:

It is recommended to use `abi.encode` as preferred.

Resolution:

The client followed the suggestion and fixed this issue.

MUL-2 Unused Global Variables

Severity: Minor

Status: Acknowledged

Code Location:

contracts/multisig/Multisig.sol#23

Descriptions:

There are unused global variables in the contract.

Suggestion:

It is suggested to remove it to reduce gas consumption.

MUL-3 Lack `indexed` In Event

Severity: Informational

Status: Acknowledged

Code Location:

contracts/multisig/Multisig.sol#25

Descriptions:

Index event fields make the field more quickly accessible to off-chain tools that parse events. However, note that each index field costs extra gas during emission, so it's not necessarily best to index the maximum allowed per event (three fields). Each event should use three indexed fields if there are three or more fields and gas usage is not particularly of concern for the events in question. If there are fewer than three fields, all of the fields should be indexed.

Suggestion:

It is recommended to add `indexed` modifier in the event.

MUL-4 Use Calldata Instead of Memory for Function Arguments That Do not Get Mutated

Severity: Informational

Status: Acknowledged

Code Location:

contracts/multisig/Multisig.sol#55,85;

contracts/account/Chip.sol#85

Descriptions:

Mark data types as `calldata` instead of memory where possible. This makes it so that the data is not automatically loaded into memory. If the data passed into the function does not need to be changed (like updating values in an array), it can be passed in as `calldata`. The one exception to this is if the argument must later be passed into another function that takes an argument that specifies memory storage.

Suggestion:

It is recommended to use `calldata` instead of `memory`.

ZKT-1 ZKT Can Be Minted Infinitely

Severity: Major

Status: Acknowledged

Code Location:

contracts/ZKT.sol#13

Descriptions:

There is a `facuet` function in the `ZKT` contract that doesn't have any permissions, and any user can call `facuet` to mint 10000 `ZKT` to themself.

Suggestion:

It is recommended that the `facuet` function be controlled with the appropriate permissions.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.