**Comdex CosmWasm Contracts**

**Executive Summary**

The Comdex CosmWasm Contract comprises of Governance contract loosely based on

CosmWasm cw-plus/cw3 fixed-multisig contract.Bindings are written under packages/bindings that interact with the Comdex modules through wasm bindings defined in comdex/app/wasm.

**Code Base**

| Github Repo | https://github.com/comdex-official/gov-contracts |
| --- | --- |
| Commit Hash |  |
| Branch |  |

**Dev Point of Contact**

| **Name** | **Email** |
| --- | --- |
| Dheeraj Dubey | dheeraj@comdex.one |
| Subham | shubham@comdex.one |

**Scope**

**Specifications**

The main specifications which are to be audited are listed below:

* Inconsistency between the specification and Implementation
* Flaws in design, logic or access control
* Arithmetic Overflows and Underflows
* Function Visibility
* Compiler Warnings
* Limits on ByteCode and Gas Usage
* Reentrancy, Code Injection and Denial of Service Attacks
* Call stack Depth
* Time Manipulation, Unbounded Loops and
* External Calls and Validation of inputs for public/external functions.
* Authentication mechanism
* Race Conditions and other known attacks
* Failure States, Speed Bumps, and Circuit Breakers

**High-Risk Areas**

* External and Public Functions
* Assembly and Low-Level Calls
* Superuser privileges
* Timing or network congestion
* Payments and Withdrawals

**Files in Scope**

Below mentioned are CosmWasm contracts/files to be covered for audit

| **S.No** | **File(s)** | **Location** | **Description** |
| --- | --- | --- | --- |
| 1 | governance | contracts/governance/\* | The base governance contract |
| 2 | bindings | packages/bindings/\* | Has structs and enums for Comdex bindings |

**Files Out of Scope**

This audit is only for the smart contracts, the following listed below are not part of the scope:

* packages/cw3 (directly used from cw-plus/packages of CosmWasm) [Link](https://github.com/CosmWasm/cw-plus/tree/main/packages)
* packages/cw4 (directly used from cw-plus/packages of CosmWasm) [Link](https://github.com/CosmWasm/cw-plus/tree/main/packages)
* packages/utils (directly used from cw-plus/packages of CosmWasm) [Link](https://github.com/CosmWasm/cw-plus/tree/main/packages)
* packages/storage-plus (directly used from cw-plus/packages of CosmWasm) [Link](https://github.com/CosmWasm/cw-plus/tree/main/packages)

**Depth of Scope**

The goal of the smart contract audit is to meticulously go through all the smart contracts and identify security flaws and vulnerabilities.

Depth of the Audit Includes:

* Overall Analysis of Smart Contract Code and Documentation
* Code Review, including third party interaction, library structure, functionality and cryptography
* Automated and Manual analysis of application, nested components, input fields, actions and all the requests.
* Bug Scanning, both at binary and source code level. Potential deviations from coding guidelines and security practices.
* Scanning Result Verification, false positives and false negatives which might affect the application.

**Tools used by Comdex**

Rust, CosmWasm ,Keplr, llmv-cov ( for coverage test in mac m1),

**Deliverables**

The following deliverables are expected from the audit team:

* Executive summary of Audit
* Project approach
  + Rules of Engagement
  + Description of security audit methodology
  + Scope description in detail
* Vulnerability analysis, findings and recommendations
* Workflow of security audit
* Further information on findings and detailed recommendations
* Conclusion
* Summary recommendations and further steps

**Smart Contracts System Overview**

**Governance**

The contract serves the executive governance for Harbor protocol.The execution messages required for proposals require custom bindings defined in packages/binding containing msg.rs for execution messages for comdex modules and query.rs for getting states.The contract includes the following hierarchy

* artifacts (contains the .wasm file)
* examples
  + schema.rs (to generate schema files in schema dir.)
* schema
  + execute\_msg.json (json structure for execution msgs)
  + Instantiate\_msg.json (json structure for instantiation msgs)
  + query\_msg.json (json structure for query msgs)
* src
  + coin\_helpers.rs ( helper function to asset correct coin from user in cli in –amount flag)
  + contract.rs (main governance contract logic with test cases)
  + error.rs ( all error variants used in the contract)
  + lib.rs ( defines crates in scope for the contract)
  + msg.rs (msgs struct and enum for the contracts)
  + state.rs (state variables and state change implementations)
  + validation.rs (calls binding queries to get states from module state [not included in contract test cases])
* cargo.toml (contract lib dependency and versions maintained)

**Instantiation**:

To create a contract we must first instantiate the contract. The instantiate msg is provided below:

pub struct InstantiateMsg {

pub threshold: Threshold,

pub target: String,

}

pub enum Threshold {

/// Declares that a fixed weight of Yes votes is needed to pass.

/// See `ThresholdResponse.AbsoluteCount` in the cw3 spec for details.

AbsoluteCount { weight: u128 },

/// Declares a percentage of the total weight that must cast Yes votes in order for

/// a proposal to pass.

/// See `ThresholdResponse.AbsolutePercentage` in the cw3 spec for details.

AbsolutePercentage { percentage: Decimal },

/// Declares a `quorum` of the total votes that must participate in the election in order

/// for the vote to be considered at all.

/// See `ThresholdResponse.ThresholdQuorum` in the cw3 spec for details.

ThresholdQuorum { threshold: Decimal, quorum: Decimal },

}

* Only the ThresholdQuorum variant is used for this contract.
* threshold\_quorum.threshold: a decimal value denoting the minimum percentage of yes vote to pass the proposal
* threshold\_quorum.quorum: a decimal value denoting the minimum percentage of total vote weight required to pass the proposal
* target: grpc server default port (used for fetching the token balance from of an address at a specific block height)

**Messages:**

* *Propose*: The proposer creates a proposal
  + Message Struct:

struct Propose {

pub title: String,

pub description: String,

pub msgs: Vec<ComdexMessages>,

// note: we ignore API-spec'd earliest if passed, always opens immediately

pub latest: Option<Expiration>,

pub app\_id\_param: u64,

}

1. title: Title of proposal
2. description: Description of proposal
3. msgs: Cosmos Custom msgs to be executed once the proposal is passed . Comdex Messages variants defined in packages/binding/msgs.
4. latest :Optional Expiration enum for setting up expiration duration apart from default Expiration for the application.
5. app\_id\_param: app id for which proposal is raised.

* *Vote*: Users can vote on a specific proposal id
  + Message Struct:

Vote { proposal\_id: u64, vote: Vote }

pub enum Vote {

/// Marks support for the proposal.

Yes,

/// Marks opposition to the proposal.

No,

/// Marks participation but does not count towards the ratio of support / opposed

Abstain,

/// voters to be able to Veto, or them to be counted stronger than No.

Veto,

}

1. proposal\_id: the proposal id for which the vote is being casted
2. vote: on of vote enum is yes to cast vote

* *Execute*: Execute the Custom message (Comdex Message) in a proposal once proposal is passed .
  + Message Struct:

Execute { proposal\_id: u64 }

1. proposal\_id: the proposal id for which the vote is being casted

* *Refund*: Proposal depositors claim refund if proposal stands passed or executed.
  + Message Struct:

Refund { proposal\_id: u64 }

1. proposal\_id: the proposal id for which the vote is being casted

* *Deposit*: Users can deposit on “Pending” or “Open” proposals.
  + Message Struct:

Deposit { proposal\_id: u64 }

1. proposal\_id: the proposal id for which the vote is being casted

* *Slash*: Burn proposal deposit if the proposal is vetoed.
  + Message Struct:

Slash { proposal\_id: u64 }

1. proposal\_id: the proposal id for which the vote is being casted

**States:**

* *CONFIG*:
  + Description:
  + Type: item<Config>
  + Config State Struct:

pub struct Config {

pub threshold: Threshold,

pub target: String,

}

* *PROPOSAL\_COUNT*
  + Description:Global Counter for proposals
  + Type: item<u64>
* *BALLOTS*
  + Description:Voter’s Vote on a proposal
  + Type: Map<(u64, &Addr), Ballot>
    - key: proposal\_id:u64 , voters address: &Addr
    - value:Ballot
  + Ballot State Struct:

pub struct Ballot {

pub weight: u128,

pub vote: Vote,

}

* *PROPOSALSBYAPP*
  + Description: List of proposals for an app\_id
  + Type: Map<u64, Vec<u64>>
    - Key:app\_id
    - Value: array for proposal id’s
* *PROPOSALS*
  + Description: Proposal details for an proposal\_id
  + Type:Map<u64,Proposal>
    - Key: proposal\_id:u64
    - Value: Proposals
  + Proposal State Struct:

pub struct Proposal {

pub title: String,

pub start\_time: Timestamp,

pub description: String,

pub start\_height: u64,

pub expires: Expiration,

pub msgs: Vec<ComdexMessages>,

pub status: Status,

pub duration: Duration,

/// pass requirements

pub threshold: Threshold,

// the total weight when the proposal started (used to calculate percentages)

pub total\_weight: u128,

// summary of existing votes

pub votes: Votes,

pub deposit: Vec<Coin>,

pub proposer: String,

pub token\_denom: String,

pub min\_deposit: u64,

pub current\_deposit: u128,

pub app\_mapping\_id: u64,

pub is\_slashed: bool,

}

* *VOTERDEPOSIT*
  + Description: User deposit for a proposal\_id
  + Type:Map<(u64, &Addr), Vec<Coin>>
    - key:<proposal\_id:u64,voter\_address:&Addr>
    - Value: Vec<Coin>
* *APPPROPOSALS*
  + Description: Array of Proposals details for an app\_id
  + Type:Map<u64, Vec<AppProposalConfig>>
    - Key:app\_id: u64
    - Value:Vec<AppProposalConfig>
  + AppProposalConfig State Struct
  + pub struct AppProposalConfig {
  + pub proposal\_id: u64,
  + pub proposal: Proposal,
  + }

* *APPGOVCONFIG*
  + Description: Proposal all up info for and app\_id
  + Type:Map<u64, AppGovConfig>
    - key:app\_id:u64
    - value:AppGovConfig
  + AppGovConfig State Struct:

pub struct AppGovConfig {

pub proposal\_count: u64,

}

pub current\_supply: u128,

pub active\_participation\_supply: u128,

}