// Cosmos Security Assessment

09.16.2024 - 09.20.2024

SetAdmin & Config Files

Kima Finance

HALBORN

SetAdmin & Config Files - Kima Finance

Prepared by: HALBORN

Last Updated 10/31/2024

Date of Engagement by: September 16th, 2024 - September 20th, 2024

Summary

100% ⓘ OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
8	1	1	0	2	4

TABLE OF CONTENTS

1. Introduction

`Kima Finance` engaged Halborn to conduct a security assessment on their config files, `go.mod`, `app/app.go` and `SetAdmin` function beginning on 2024-09-16 and ending on 2024-09-20. The security assessment was scoped to the `kima-blockchain` and config files provided to the Halborn team.

2. Assessment Summary

The team at Halborn was provided five days for the engagement and assigned one full-time security engineer to assessment the security of the merge requests. The security engineers are blockchain and smart-contract security experts with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this assessment is to:

- Ensure that the **Golang components** operate as intended.
- Identify potential security issues with the Golang application.
- Detect bad configurations that could pose any security risk to the protocol.
- Shared config files were properly created for a local test environment.

Based on the current scope provided, we are limited in our ability to fully evaluate the overall performance of the chain and identify any issues that might impact its functionality. Expanding the scope would enable us to conduct a more thorough assessment and address any potential risks.

In summary, Halborn identified one relevant issue and some recommendations that some of them were addressed by `Kima Finance team`.

3. Test Approach And Methodology

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the custom modules. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of structures and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the assessment:

- Research into architecture and purpose.
- Static Analysis of security for scoped repository, and imported functions. (e.g., `staticcheck`, `gosec`, `unconvert`, `codeql`, `ineffassign` and `semgrep`)
- Manual Assessment for discovering security vulnerabilities on the codebase.
- Ensuring the correctness of the codebase.
- Dynamic Analysis of files and modules in scope.

4. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient.** This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope.** These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

4.1 EXPLOITABILITY

ATTACK ORIGIN (AO):

Captures whether the attack requires compromising a specific account.

ATTACK COST (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

ATTACK COMPLEXITY (AX):

Describes the conditions beyond the attacker’s control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

METRICS:

EXPLOITABILIY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

4.2 IMPACT

CONFIDENTIALITY (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

INTEGRITY (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

AVAILABILITY (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

DEPOSIT (D):

Measures the impact to the deposits made to the contract by either users or owners.

YIELD (Y):

Measures the impact to the yield generated by the contract for either users or owners.

METRICS:

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Confidentiality (C)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Integrity (I)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Availability (A)	None (A:N) Low (A:L) Medium (A:M) High (A:H) Critical (A:C)	0 0.25 0.5 0.75 1
Deposit (D)	None (D:N) Low (D:L) Medium (D:M) High (D:H) Critical (D:C)	0 0.25 0.5 0.75 1

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Yield (Y)	None (Y:N) Low (Y:L) Medium (Y:M) High (Y:H) Critical (Y:C)	0 0.25 0.5 0.75 1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

4.3 SEVERITY COEFFICIENT

REVERSIBILITY (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

SCOPE (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

METRICS:

SEVERITY COEFFICIENT (C)	COEFFICIENT VALUE	NUMERICAL VALUE
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25

SEVERITY COEFFICIENT (C)	COEFFICIENT VALUE	NUMERICAL VALUE
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

SEVERITY	SCORE VALUE RANGE
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9

SEVERITY	SCORE VALUE RANGE
Low	2 - 4.4
Informational	0 - 1.9

5. SCOPE

FILES AND REPOSITORY

(a) Repository: kima-blockchain

(b) Assessed Commit ID: bbca9c7

(c) Items in scope:

- kima-blockchain/x/kima/keeper/msg_server_set_admin.go
- kima-blockchain/go.mod
- kima-blockchain/app/app.go
- genesis.json (shared in Slack)
- config.toml (shared in Slack)
- app.toml (shared in Slack)
- IBC relayer config (shared in Slack)

Out-of-Scope: Third party dependencies and economic attacks.

Out-of-Scope: New features/implementations after the remediation commit IDs.

6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
1	1	0	2	4

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
LACK OF ACCESS CONTROL FOR PRIVILEGED FUNCTION	CRITICAL	SOLVED - 10/30/2024
MINIMUM GAS PRICE IS SET TO ZERO LEADS NETWORK TO SPAM ATTACKS	HIGH	SOLVED - 10/29/2024
RETURN NIL, ERR IN CASE OF ERRORS	LOW	RISK ACCEPTED - 10/29/2024
INCORRECT DENOMINATION SUGGESTION IN CONFIGURATION	LOW	SOLVED - 10/29/2024
TLS IS DISABLED IN RPC SERVER	INFORMATIONAL	ACKNOWLEDGED - 10/29/2024
DISABLE CONFIRMATION FOR RELAYED PACKETS	INFORMATIONAL	ACKNOWLEDGED - 10/29/2024
GAS USAGE CAN BE OPTIMIZED IN RELAYED PACKETS	INFORMATIONAL	ACKNOWLEDGED - 10/29/2024

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
DEFINE GLOBAL LABELS FOR TELEMETRY ANALYTICS	INFORMATIONAL	ACKNOWLEDGED - 10/29/2024

7. FINDINGS & TECH DETAILS

7.1 LACK OF ACCESS CONTROL FOR PRIVILEGED FUNCTION

// CRITICAL

Description

The `SetAdmin` function in the Cosmos chain sets a super admin for the chain by allowing any incoming message to assign the `msg.Creator` as the super admin, without enforcing any form of access control or permission checks. Although the function includes a check to prevent the super admin from being set more than once, there is no restriction on who can attempt the initial admin setup. This introduces a significant security vulnerability, as any user can front-run the transaction and set themselves as the super admin, gaining control over critical functionalities of the blockchain.

https://github.com/kima-finance/kima-blockchain/blob/7c1a9c7f9b784a21ee3d698c27cc643fce151175/x/kima/keeper/msg_server_set_admin.go#L36-L62

```go
func (k msgServer) SetAdmin(goCtx context.Context, msg *types.MsgSetAdmin) (*types.MsgSetAdminResponse, error) {
	ctx := sdk.UnwrapSDKContext(goCtx)

	// Check if super admin already exists
	_, bFound := k.GetSuperAdmin(ctx)
	if bFound {
		return &types.MsgSetAdminResponse{Code: "301", Msg: "Already exists!"}, status.Error(codes.AlreadyExists
	}

	// set super admin
	k.SetSuperAdminAddress(ctx, msg.Creator)

	// Emits event
	ctx.EventManager().EmitEvents(sdk.Events{
		sdk.NewEvent(
			sdk.EventTypeMessage,
            sdk.NewAttribute(sdk.AttributeKeyModule, types.EventTypeSuperAdminSet),
            sdk.NewAttribute(sdk.AttributeKeySender, msg.Creator),
        ),
        sdk.NewEvent(
            types.EventTypeSuperAdminSet,
            sdk.NewAttribute(sdk.AttributeKeySender, msg.Creator),
        ),
    })

    return &types.MsgSetAdminResponse{Code: "200", Msg: "Successfully set"}, nil
}
```

Given that this operation is intended for specific, trusted entities, the absence of authentication or role verification allows malicious actors to exploit the function by being the first to invoke it. This could lead to unauthorized individuals gaining administrative privileges and compromising the integrity of the blockchain.

BVSS

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:C/I:C/D:N/Y:N (10.0)

Recommendation

Ensure that only a predefined list of trusted addresses can call the `SetAdmin` function. The initial admin setup should be restricted to addresses that have been verified or authorized beforehand, such as through governance proposals or multi-signature wallets. For example, setting an `authority` address during the module's initialization, allowing this function only to be called by `authority` account.

Remediation

SOLVED: The **Kima team** solved this issue guaranteeing that super admin assignment will be done in first transactions prior to making it public.

7.2 MINIMUM GAS PRICE IS SET TO ZERO LEADS NETWORK TO SPAM ATTACKS

// HIGH

Description

In the app.toml configuration file, the `minimum-gas-prices` parameter is set to "0stake", effectively allowing transactions with no fees:

```
minimum-gas-prices = "0stake"
```

With no minimum gas price, the network becomes vulnerable to spam attacks, where malicious actors can flood the network with zero-fee transactions.

BVSS

AO:A/AC:L/AX:L/C:N/I:M/A:M/D:N/Y:N/R:N/S:C (7.8)

Recommendation

Set an appropriate minimum gas price based on the native token of the Kima network. For example:

```
minimum-gas-prices = "0.01ukima"
```

Remediation

SOLVED: The **Kima team** solved this issue by setting an `minimum-gas-prices` other than zero.

7.3 RETURN NIL, ERR IN CASE OF ERRORS

// LOW

Description

In the current implementation, the code checks if a `SuperAdmin` already exists and returns a custom response along with a status error if it does:

https://github.com/kima-finance/kima-blockchain/blob/7c1a9c7f9b784a21ee3d698c27cc643fce151175/x/kima/keeper/msg_server_set_admin.go#L39-L43

```
// Check if super admin already exists
_, bFound := k.GetSuperAdmin(ctx)
if bFound {
    return &types.MsgSetAdminResponse{Code: "301", Msg: "Already exists!"}, status.Error(codes.AlreadyExists, "A
}
```

However, this return format deviates from the common standard, where functions are expected to return `nil` and an error in case of failure or an invalid request. By returning a custom response in the form of `&types.MsgSetAdminResponse` instead of `nil` along with the error, the code may introduce inconsistency in the error handling. This could lead to further issues when handling errors or processing transactions, as the expected pattern is not followed.

BVSS

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:L/I:N/D:N/Y:N (2.5)

Recommendation

It is recommended to return a `nil` instead of the custom response.

Remediation

RISK ACCEPTED: The **Kima team** accepted the risk of this issue.

7.4 INCORRECT DENOMINATION SUGGESTION IN CONFIGURATION

// LOW

Description

In the app.toml configuration file, under the **config.toml**, the `denom-to-suggest` parameter is incorrectly set to "uatom" instead of "kima". This error could lead to incorrect fee suggestions for transactions on the Kima network.

```
# DenomToSuggest defines the defult denom for fee suggestion.
# Price must be in minimum-gas-prices.
denom-to-suggest = "uatom"
```

BVSS

AO:A/AC:L/AX:M/C:N/I:L/A:N/D:N/Y:N/R:N/S:C (2.1)

Recommendation

Update the app.toml file, changing the `denom-to-suggest` value from "uatom" to "ukima"

Remediation

SOLVED: The **Kima team** solved this issue by setting `denom-to-suggest` to `ukima`.

7.5 TLS IS DISABLED IN RPC SERVER

// INFORMATIONAL

Description

In the provided `config.toml`, the TLS-related fields `tls_cert_file` and `tls_key_file` are left empty:

`config.toml`:

```
# The path to a file containing certificate that is used to create the HTTPS server.
# Might be either absolute path or path related to CometBFT's config directory.
# If the certificate is signed by a certificate authority,
# the certFile should be the concatenation of the server's certificate, any intermediates,
# and the CA's certificate.
# NOTE: both tls_cert_file and tls_key_file must be present for CometBFT to create HTTPS server.
# Otherwise, HTTP server is run.
tls_cert_file = ""

# The path to a file containing matching private key that is used to create the HTTPS server.
# Might be either absolute path or path related to CometBFT's config directory.
# NOTE: both tls-cert-file and tls-key-file must be present for CometBFT to create HTTPS server.
# Otherwise, HTTP server is run.
tls_key_file = ""
```

Without configuring these fields, the node will default to running an HTTP server instead of a more secure HTTPS server. This exposes the network communication to potential attacks, such as eavesdropping, man-in-the-middle (MITM) attacks, and data tampering. Using HTTP means the communication between the node and its clients is unencrypted.

BVSS

AO:A/AC:M/AX:M/R:N/S:U/C:L/A:N/I:N/D:N/Y:N (1.1)

Recommendation

To mitigate the risk of insecure communication, it's strongly recommended to enable TLS by configuring the `tls_cert_file` and `tls_key_file` fields.

Remediation

ACKNOWLEDGED: The **Kima team** acknowledged the risk of this issue.

7.6 DISABLE CONFIRMATION FOR RELAYED PACKETS

// INFORMATIONAL

Description

In the provided configuration for the IBC relayer running Hermes, the parameter `tx_confirmation` is set to `false`:

`relayer-ibc.toml`:

```
[mode.packets]
enabled = true
clear_interval = 100
clear_on_start = true
tx_confirmation = false
auto_register_counterparty_payee = false
```

When `tx_confirmation` is set to `false`, the relayer will not periodically verify whether the submitted IBC transactions have been successfully delivered. Specifically, the relayer will not check acknowledgment, reception, or timeout confirmations for packets relayed between chains. This setting disables critical telemetry metrics such as `acknowledgment_packets_confirmed`, `receive_packets_confirmed`, and `timeout_packets_confirmed`, which monitor the successful delivery of packets.
Disabling transaction confirmation introduces the risk of undetected packet delivery failures. If a transaction fails to be acknowledged or delivered, the relayer will not automatically retry or notify that an issue has occurred.

Score

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:N/Y:N (0.0)

Recommendation

This is a feature to be considered to activate in case a double-check to validate the packets have been relayed correctly is desired.

Remediation

ACKNOWLEDGED: The **Kima team** acknowledged the risk of this issue.

7.7 GAS USAGE CAN BE OPTIMIZED IN RELAYED PACKETS

// INFORMATIONAL

Description

In the provided Hermes IBC relayer configuration, the gas parameters for interacting with Osmosis are set as follows:

`relayer-ibc.toml`:

```
default_gas = 200000
max_gas = 4000000
gas_price = { price = 1, denom = 'uosmo' }
gas_multiplier = 1.5
```

While the current configuration is functional, it is suboptimal for interacting with the Osmosis chain. The high gas price of `1 uosmo` and the gas multiplier of `1.5` result in excessive gas usage and fees, which could be significantly optimized. According to the recommended settings from Osmosis' shared templates, lower values can be used for both the `gas_price` and `gas_multiplier` to reduce the cost of gas without compromising transaction efficiency.

Score

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:N/Y:N (0.0)

Recommendation

To align with Osmosis' recommended configurations and reduce gas expenditure, the following changes are suggested:

1. **Lower the `gas_price`**: Reduce the gas price from `1 uosmo` to `0.1 uosmo` or even as low as `0.0026 uosmo`, which is the minimum viable price in some cases. This will significantly reduce the transaction fees.**Updated gas price:**

```
gas_price = { price = 0.1, denom = 'uosmo' }
```

2. **Adjust the** `gas_multiplier`: Lower the gas multiplier from `1.5` to `1.1` as recommended by Osmosis. This will help in adjusting the gas limit calculations more efficiently without over-allocating gas.**Updated gas multiplier:**

```
gas_multiplier = 1.1 # Optimized multiplier for gas calculation
```

3. **Optional - Tweak** `default_gas` and `max_gas` if needed: While `default_gas = 200000` and `max_gas = 4000000` are reasonable, further tuning can be applied depending on the relayer's interaction patterns with the chain. However, the key optimizations are the gas price and multiplier.

Remediation

ACKNOWLEDGED: The **Kima team** acknowledged the risk of this issue.

7.8 DEFINE GLOBAL LABELS FOR TELEMETRY ANALYTICS

// INFORMATIONAL

Description

In the current app.toml configuration file, the `global-labels` field under the `[telemetry]` section is empty. Defining global labels can significantly enhance the quality and usefulness of telemetry data for analytics purposes.

```
[telemetry]
# ... other configurations ...
global-labels = [
]
```

Score

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:N/Y:N (0.0)

Recommendation

Define relevant global labels to provide context for all telemetry metrics. Here's an example of how to update the configuration:

```
[telemetry]
# ... other configurations ...
global-labels = [
  ["chain_id", "kima-1"],
  ["environment", "mainnet"],
  ["validator_name", "kima-validator-1"],
  ["region", "us-west-2"],
  ["version", "v1.0.0"]
]
```

Remediation

ACKNOWLEDGED: The **Kima team** acknowledged the risk of this issue.

8. AUTOMATED TESTING

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped component. Among the tools used were `staticcheck`, `gosec`, `semgrep`, `codeQL` and `Nancy`. After Halborn verified all the files and scoped structures in the repository and was able to compile them correctly, these tools were leveraged on scoped structures. With these tools, Halborn can statically verify security related issues across the entire codebase.
The result of those automatic tools were analyzed and none of the highlighted findings pose any risk to the application.

Halborn strongly recommends conducting a follow-up assessment of the project either within six months or immediately following any material changes to the codebase, whichever comes first. This approach is crucial for maintaining the project's integrity and addressing potential vulnerabilities introduced by code modifications.