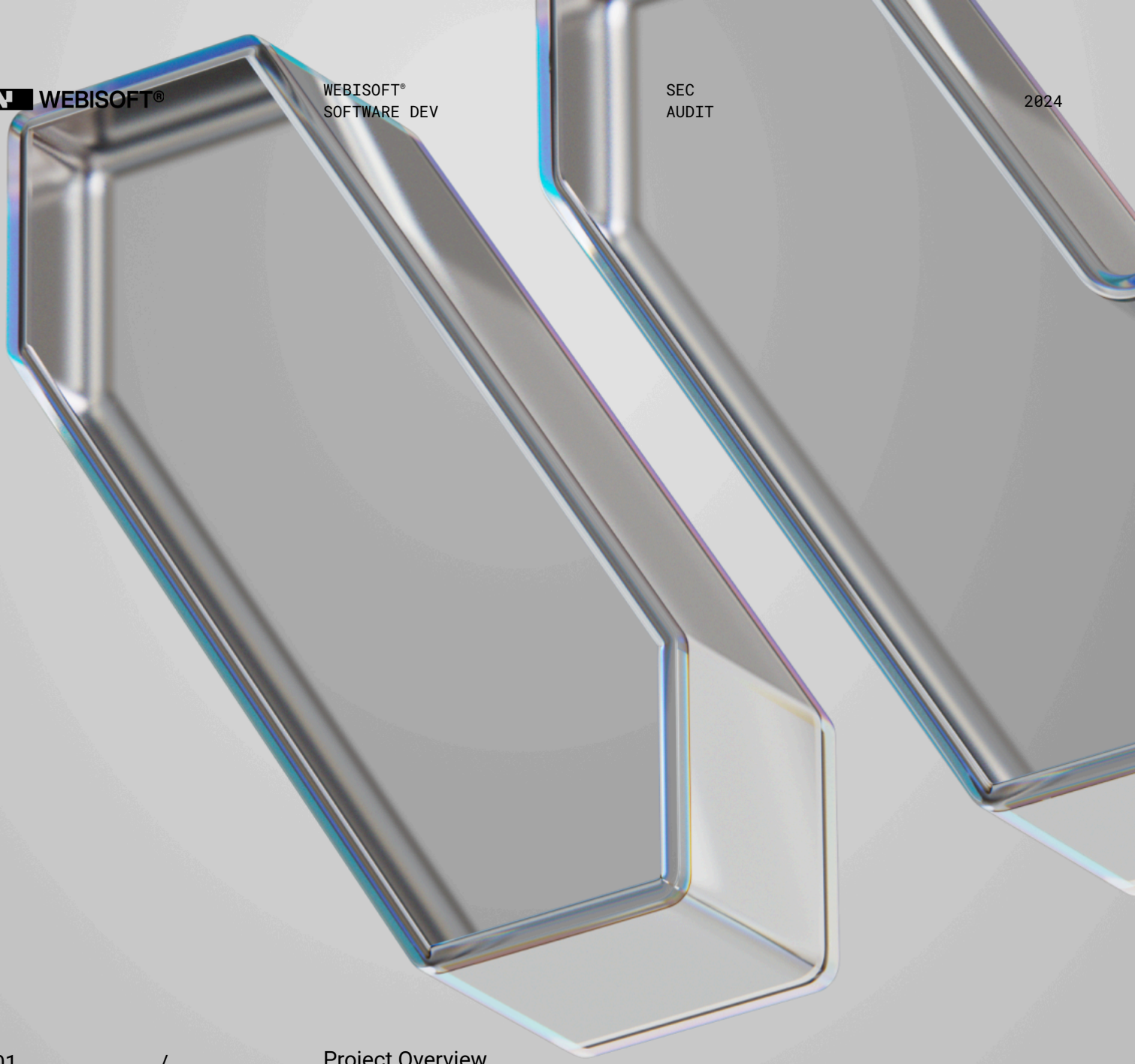
WEBISOFT X
Kii – Chain Audit



001 / Project Overview

This is the beginning of something great.

Disclaimer

This audit report (“Report”) has been prepared by Webisoft for Kii Chain and is provided solely as a reference to assist in understanding the reviewed code, systems, or operations at the time of the assessment. It is made available on an “as is” basis, without any express or implied warranties, including, but not limited to, any warranties of accuracy, completeness, reliability, fitness for a particular purpose, or non-infringement. Webisoft, its auditors, employees, agents, representatives, and affiliates (collectively, the “Providers”) make no claim that the findings, opinions, or conclusions herein are definitive or free from error. Under no circumstances will the Providers be liable for any direct, indirect, incidental, consequential, special, punitive, or exemplary damages—whether foreseeable or not—arising from your use of, reliance on, or reference to this Report, regardless of the theory of liability and even if the Providers have been advised of the possibility of such damages. This Report does not constitute legal, financial, investment, tax, or any other form of professional advice. You bear sole responsibility for conducting your own independent evaluations, due diligence, and for any actions or decisions you make based on this Report. By accessing, reviewing, or otherwise making use of this Report, you acknowledge and agree to the terms set forth in this Disclaimer. If you do not accept these terms, you should not use, rely upon, or refer to this Report.

Project Overview /

Webisoft was engaged to conduct a security audit on Kii chain from November 18th to November 25th, 2024. The security assessment was scoped to possible vulnerabilities, best practices and any potential bugs.

Scope

Target: https://github.com/KiiChain/kiichain3
Technologies: Cosmos SDK.
Audit Objectives:

- Identify potential security vulnerabilities.
- Ensure adherence to best practices.
- Validate logical correctness and resilience.

Methodology

As part of the auditing process, the code was manually inspected using two approaches:

1. **Analytical Review:**
   a. A detailed examination of the source code to identify potential security vulnerabilities, design inconsistencies, and logical flaws.
   b. This included evaluating the functions, data structures, and interactions to detect possible exploits, unintended behaviors, and inefficiencies.
2. **Checklist Validation:**
   a. The checklist covered areas such as adherence to Cosmos programming conventions, secure account management, error handling, access control, resource allocation, and transaction execution.

The goal of this audit is to ensure the structural integrity of the source code, validate adherence to Cosmo's security standards, and confirm that the code performs its intended functions without vulnerabilities or inefficiencies. Recommendations for improvement have been provided where applicable to address identified issues and enhance overall code quality.

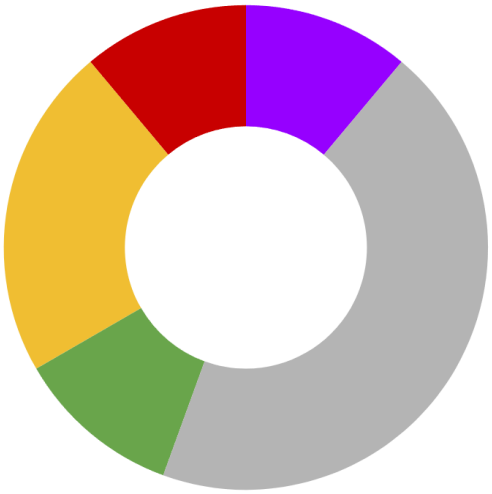
Summary of Findings /

Overall, we found 10 potential vulnerabilities. These vulnerabilities were categorized by severity according to the following table:

Overall, we found 9 potential vulnerabilities. These vulnerabilities were categorized by severity according to the following table:

	Low Impact	Medium Impact	High Impact
Very Likely	Medium	High	Critical
Somewhat Likely	Low	Medium	High
Unlikely	Informational	Low	Medium

The chart below displays the findings by their severity.



Severity	Count
Critical	0
High	5
Medium	3
Low	2
Informational	0

Detailed Findings / Below are the detailed findings of our audit

- **WS-KII-01** Unrestricted Hook Execution Time in Epoch Transitions

Impact: High
Likelihood: Medium
Severity: High

Description:
The epoch module allows unlimited execution time for hooks during epoch transitions, which could lead to block production delays or chain halts if hooks perform resource-intensive operations.

Recommendation:
Implement a context-based timeout mechanism that cancels hook execution if it exceeds a predefined duration. This should be configurable through governance parameters and include proper error handling for timeout scenarios.

- **WS-KII-02** Missing Upper Bound Validation for Epoch Duration

Impact: Low
Likelihood: Low
Severity: Informational

Description:
The module only validates that epoch duration is non-zero, allowing arbitrarily long epochs that could impact protocol operations and resource utilization.

Recommendation:
Define a maximum epoch duration as a governance parameter with a reasonable default. Include comprehensive validation in both the epoch creation and modification processes, with proper error handling for out-of-bounds values.

-

WS-KII-03
Unchecked Panic Conditions in GetStartDateTime/GetEndDateTime

Impact: Medium
Likelihood: Low
Severity: Low

Description:
Direct panic calls in time parsing in minter.go functions(GetStartDateTime/GetEndDateTime) could halt chain operation if date format corruption occurs.

Recommendation:
Implement error handling instead of panic, with state recovery mechanisms

WS-KII-04
Unprotected Minter Updates via Governance

Impact: High
Likelihood: Low
Severity: Medium

Description:
HandleUpdateMinterProposal allows complete minter parameter changes with minimal validation.

Recommendation:
Add comprehensive validation suite and rate limiting for minter updates.

WS-KII-05
Lack of Bounded Execution in MultiMintHooks

Impact: Low
Likelihood: Low
Severity: Low

Description:
MultiMintHooks executes all hooks without timeout or resource limits.

Recommendation:
Implement hook execution limits and timeouts.

WS-KII-06

Unbounded Denom retrieval in getDenomsFromCreator

Impact: Medium
Likelihood: Low
Severity: Medium

Description:
getDenomsFromCreator retrieves all denoms without pagination, potentially loading unlimited data into memory.

Recommendation:
Implement pagination with max limit parameters and cursor-based iteration.

WS-KII-07

Inadequate Denom Creation Validation

Impact: High
Likelihood: Medium
Severity: High

Description:
In tokenfactory/createdenom.go, validateCreateDenom lacks comprehensive checks

Recommendation:
Add subdenom character set validation
Implement length restrictions
Add pattern validation
Include reserved name checks

WS-KII-08

Unrestricted Module Account Creation

Impact: High
Likelihood: Medium
Severity: High

Description:
In tokenfactory/keeper.go, CreateModuleAccount lacks permission checks

Recommendation:
Add module account creation restrictions
Implement privilege checks

WS-KII-09

Unchecked fee collector

Impact: High
Likelihood: Low
Severity: High

Description:
GetFeeCollectorAddress ignores the error which could lead to lost fees

Recommendation:
Handle the error and panic or return error if fee collector is invalid

WS-KII-10

Not registered types

Impact: Medium
Likelihood: Medium
Severity: Medium

Description:
MsgAssociate, MsgInternalEVMCall and MsgInternalEVMDelegateCall are not registered in RegisterInterfaces in x/evm/types/codec.go. Hence the module will not route the messages to an appropriate server.

Recommendation:
Register the interface accordingly.

See you soon.

CONTACT US

William Marchand
Phil Therien

william@webisoft.com
phil@webisoft.com

460 Sainte-Catherine W.
Suite 305
Montreal, Quebec

Canada
H3B 1A6