BLOCKSEC

# Security Audit Report for Chain Upgrades

**Date:** October 16, 2026 **Version:** 1.0

**Contact:** contact@blocksec.com

# Contents

**Chapter 1 Introduction** **1**
1.1 About the Audit Target . . . 1
1.2 Disclaimer . . . 1
1.3 Procedure of Auditing . . . 2
1.3.1 Security Issues . . . 2
1.3.2 Additional Recommendation . . . 3
1.4 Security Model . . . 3

**Chapter 2 Findings** **4**
2.1 Security Issue . . . 4
2.1.1 Ineffective KV store deletion in the upgrade v0.50.15 . . . 4
2.2 Recommendation . . . 5
2.2.1 Remove the stale EVM store-key regression test . . . 5

## Report Manifest

| Item | Description |
| --- | --- |
| Client | Oraichain |
| Target | Chain Upgrades |

## Version History

| Version | Date | Description |
| --- | --- | --- |
| 1.0 | October 16, 2026 | First release |

## Signature



Digitally signed by: BlockSec (https://blocksec.com)

SHA1 digest of public key: BAB106D3708BC6DE0963CE05E9478C24B326AE1C

Date: 2026-10-26 03:15:19+0000

**About BlockSec** BlockSec focuses on the security of the blockchain ecosystem and collaborates with leading DeFi projects to secure their products. BlockSec is founded by top-notch security researchers and experienced experts from both academia and industry. They have published multiple blockchain security papers in prestigious conferences, reported several zero-day attacks of DeFi applications, and successfully protected digital assets that are worth more than 14 million dollars by blocking multiple attacks. They can be reached at Email, Twitter and Medium.

# Chapter 1 Introduction

## 1.1 About the Audit Target

| Information | Description |
|---|---|
| Type | Chain |
| Language | Go |
| Approach | Semi-automatic and manual verification |

The audit target (hereinafter referred to as the Target) of this audit is the code repository [1] of Chain Upgrades of Oraichain.

Chain Upgrades of Oraichain is an upgrade for a Cosmos SDK-based chain. After a security incident in August 2026, the EVM modules were removed from the chain in an emergency hard fork, while their underlying stores were kept in place. This v0.50.15 update completes that removal by pruning the retired EVM stores through a coordinated chain upgrade, preserving the ability to query historical governance proposals that contain EVM messages, and improving the default build configuration.

Note this audit only focuses on the codes located in the following directories/files.

- app/evmlegacy/codec.go
- app/upgrades/v05015/upgrades.go
- app/app.go
- app/upgrades.go

Other files are not within the scope of the audit. Additionally, all dependencies of the Target are considered reliable in terms of both functionality and security, and are therefore not included in the audit scope.

The auditing process is iterative. Specifically, we would audit the commits that fix the discovered issues. If there are new issues, we will continue this process. The commit SHA values during the audit are shown in the following table. Our audit report is responsible for the code in the initial version (`Version 1`), as well as new code (in the following versions) to fix issues in the audit report. Code prior to and including the baseline version (`Version 0`), where applicable, is outside the scope of this audit and assumes to be reliable and secure.

| Project | Version | Commit Hash |
|---|---|---|
| wasmd | `Version 0` | `05576404cfe461b51b920ec930cf7dcec6e6db91` |
| | `Version 1` | `12a800c382ade2dbd4687a5ed40dd65496ee39a6` |
| | `Version 2` | `d9ab421a501746760b254332d6a94f29802c724c` |

## 1.2 Disclaimer

This audit report does not constitute investment advice or a personal recommendation. It does not consider, and should not be interpreted as considering or having any bearing on,

[1] https://github.com/oraichain/wasmd

the potential economics of a token, token sale or any other product, service or other asset. Any entity should not rely on this report in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset.

This audit report is not an endorsement of any particular project or team, and the report does not guarantee the security of any particular project. This audit does not give any warranties on discovering all security issues of the Target, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues. As one audit cannot be considered comprehensive, we always recommend proceeding with independent audits and a public bug bounty program to ensure the security of the Target.

The scope of this audit is limited to the code mentioned in Section 1.1. Unless explicitly specified, the security of the language itself (e.g., the solidity language), the underlying compiling toolchain and the computing infrastructure are out of the scope.

## 1.3 Procedure of Auditing

We perform the audit according to the following procedure.

- **Vulnerability Detection** We first scan the Target with automatic code analyzers, and then manually verify (reject or confirm) the issues reported by them.
- **Semantic Analysis** We study the business logic of the Target and conduct further investigation on the possible vulnerabilities using an automatic fuzzing tool (developed by our research team). We also manually analyze possible attack scenarios with independent auditors to cross-check the result.
- **Recommendation** We provide some useful advice to developers from the perspective of good programming practice, including gas optimization, code style, and etc.

We show the main concrete checkpoints in the following.

### 1.3.1 Security Issues

* Access control
* Permission management
* Whitelist and blacklist mechanisms
* Initialization consistency
* Improper use of the proxy system
* Reentrancy
* Denial of Service (DoS)
* Untrusted external call and control flow
* Exception handling
* Data handling and flow
* Events operation
* Error-prone randomness
* Oracle security
* Business logic correctness
* Semantic and functional consistency

- Emergency mechanism
- Economic and incentive impact

### 1.3.2 Additional Recommendation

- Gas optimization
- Code quality and style

**Note** *The previous checkpoints are the main ones. We may use more checkpoints during the auditing process according to the functionality of the project.*

## 1.4 Security Model

To evaluate the risk, we follow the standards or suggestions that are widely adopted by both industry and academy, including OWASP Risk Rating Methodology [2] and Common Weakness Enumeration [3]. The overall *severity* of the risk is determined by *likelihood* and *impact*. Specifically, likelihood is used to estimate how likely a particular vulnerability can be uncovered and exploited by an attacker, while impact is used to measure the consequences of a successful exploit.

In this report, both likelihood and impact are categorized into two ratings, i.e., *high* and *low* respectively, and their combinations are shown in Table 1.1.

**Table 1.1:** Vulnerability Severity Classification

| Impact \ Likelihood | *High* | *Low* |
|---|---|---|
| *High* | High | Medium |
| *Low* | Medium | Low |

Accordingly, the severity measured in this report are classified into three categories: **High**, **Medium**, **Low**. For the sake of completeness, **Undetermined** is also used to cover circumstances when the risk cannot be well determined.

Furthermore, the status of a discovered item will fall into one of the following five categories:

- **Undetermined** No response yet.
- **Acknowledged** The item has been received by the client, but not confirmed yet.
- **Confirmed** The item has been recognized by the client, but not fixed yet.
- **Partially Fixed** The item has been confirmed and partially fixed by the client.
- **Fixed** The item has been confirmed and fixed by the client.

[2] https://owasp.org/www-community/OWASP_Risk_Rating_Methodology

[3] https://cwe.mitre.org/

# Chapter 2 Findings

In total, we found **one** potential security issue. Besides, we have **one** recommendation.

- Medium Risk: 1
- Recommendation: 1

| ID | Severity | Description | Category | Status |
|---|---|---|---|---|
| 1 | Medium | Ineffective KV store deletion in the upgrade v0.50.15 | Security Issue | Fixed |
| 2 | - | Remove the stale EVM store-key regression test | Recommendation | Acknowledged |

The details are provided in the following sections.

## 2.1 Security Issue

### 2.1.1 Ineffective KV store deletion in the upgrade v0.50.15

**Severity** Medium

**Status** Fixed in `Version 2`

**Introduced by** `Version 1`

**Description** The file `v05015/upgrades.go` defines the variable Upgrade to prune the KV stores (i.e., `evm`, `feemarket`, `erc20`, `precisebank`) of the soft-removed EVM modules. However, the file `app.go` removes the registration of all keys, which are expected to be removed in the upgrade. Since the key registration is removed, the expected store keys cannot be found during the upgrade. As a result, the intended kvstore deletion for the expected KV stores becomes ineffective.

```
18var Upgrade = upgrades.Upgrade{
19  UpgradeName:        UpgradeName,
20  CreateUpgradeHandler: CreateUpgradeHandler,
21  StoreUpgrades: storetypes.StoreUpgrades{
22    Added: []string{},
23    // Prune the KV stores of the EVM modules that were soft-removed in the
24    // v0.50.14 fork (evm, feemarket, erc20, precisebank). The v0.50.14 fork
25    // ran via BeginBlocker so it could not trigger the upgrade store loader;
26    // this is the first x/upgrade migration able to drop them.
27    Deleted: []string{"evm", "feemarket", "erc20", "precisebank"},
28  },
29}
```

**Listing 2.1:** app/upgrades/v05015/upgrades.go

```
398  keys := storetypes.NewKVStoreKeys(
399    authtypes.StoreKey, banktypes.StoreKey, stakingtypes.StoreKey, crisistypes.StoreKey,
400    minttypes.StoreKey, distrtypes.StoreKey, slashingtypes.StoreKey,
401    govtypes.StoreKey, paramstypes.StoreKey, consensusparamtypes.StoreKey, upgradetypes.StoreKey,
           feegrant.StoreKey, authzkeeper.StoreKey,
```

```
402    evidencetypes.StoreKey, circuittypes.StoreKey,
403    // nftkeeper.StoreKey,
404    group.StoreKey,
405    // non sdk store keys
406    capabilitytypes.StoreKey, ibcexported.StoreKey, ibctransfertypes.StoreKey, ibcfeetypes.
           StoreKey,
407    wasmtypes.StoreKey, icahosttypes.StoreKey,
408    icacontrollertypes.StoreKey, clocktypes.StoreKey, globalfeetypes.StoreKey, ibchookstypes.
           StoreKey, packetforwardtypes.StoreKey, tokenfactorytypes.StoreKey,
409    // EVM modules (evm, feemarket, erc20, precisebank) were soft-removed in the
410    // v0.50.14 fork and their KV stores are pruned by the v0.50.15 upgrade
411    // (app/upgrades/v05015 StoreUpgrades.Deleted). They must NOT be mounted here
412    // from v0.50.15 on: a mounted store that is absent from CommitInfo makes
413    // rootmulti.loadVersion panic on every restart ("version of store evm
414    // mismatch root store's version").
415    txfeestypes.StoreKey,
416  )
```

**Listing 2.2:** app/app.go

**Impact** The KV store deletion becomes ineffective due to the registration removal of expected keys.

**Suggestion** Revise the code accordingly.

**Clarification from BlockSec** Although the fix allows the upgrade to prune the retired states, it may introduce potential failures in restarting nodes due to the inconsistent construction of the commit hash. This potential risk is acknowledged and accepted by the project.

## 2.2 Recommendation

### 2.2.1 Remove the stale EVM store-key regression test

**Status** Acknowledged

**Introduced by** `Version 1`

**Description** In the file `app/evm_removed_test.go`, the test `TestEVMStoreKeysStillMounted()` still requires the four retired EVM store keys to remain mounted. The constructor in `app/app.go` no longer registers these keys, then the function `GetKey()` returns `nil` and the test fails on the first assertion. It is recommended to remove this regression test.

```
59// TestEVMStoreKeysStillMounted fails if someone "cleans up" the soft-removed EVM stores.
60func TestEVMStoreKeysStillMounted(t *testing.T) {
61  wasmApp := app.Setup(t)
62
63  for _, name := range mountedEVMStoreKeys {
64    require.NotNil(t, wasmApp.GetKey(name),
65      "store key %q must stay mounted for pre-fork state compatibility", name)
66  }
67}
```

**Listing 2.3:** app/evm_removed_test.go

**Suggestion** Remove the stale EVM store-key regression test.


BOOST WEB3 THROUGH NEXT-GENERATION SECURITY & USABILITY INNOVATIONS