Tabr

Tabi

Audit Report

Fri Jan 24 2025

contact@bitslab.xyz https://twitter.com/scalebit_

ScaleBit

Tabi Audit Report

1 Executive Summary

1.1 Project Information

Description	Tabichain is a scalable, high-throughput Proof-of-Stake blockchain that is fully compatible and interoperable with Ethereum. It's built using the Cosmos SDK which runs on top of the Tendermint Core consensus engine
Type	L1
Auditors	ScaleBit
Timeline	Mon Dec 09 2024 - Fri Jan 24 2025
Languages	Go
Platform	Others
Methods	Dependency Check, Fuzzing, Static Analysis, Manual Review
Source Code	https://github.com/tabilabs/tabi
Commits	79dc8c4cd20af7e20b76c69265f7614233a70f84 32c6b5dc644731cad0e73bef7fba7243f0f398de

1.2 Files in Scope

The following are the directories of the original reviewed files.

Directory
https://github.com/tabilabs/tabi/app/ante/tabi
https://github.com/tabilabs/tabi/app/app.go
https://github.com/tabilabs/tabi/x/claims
https://github.com/tabilabs/tabi/x/limiter
https://github.com/tabilabs/tabi/x/captains
https://github.com/tabilabs/tabi/x/token-convert

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	8	6	2
Informational	2	2	0
Minor	1	1	0
Medium	3	1	2
Major	1	1	0
Critical	1	1	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):

- Integer overflow/underflow
- Infinite Loop
- Infinite Recursion
- Race Condition
- Traditional Web Vulnerabilities
- Memory Exhaustion Attack
- Disk Space Exhaustion Attack
- Side-channel Attack
- Denial of Service
- Replay Attacks
- Double-spending Attack
- Eclipse Attack
- Sybil Attack
- Eavesdropping Attack
- Business Logic Issues
- Contract Virtual Machine Vulnerabilities
- Coding Style Issues

1.5 Methodology

Our security team adopted **"Dependency Check"**, **"Automated Static Code Analysis"**, **"Fuzz Testing"**, and **"Manual Review"** to conduct a comprehensive security test on the code in a manner closest to real attacks. The main entry points and scope of the security testing are specified in the **"Files in Scope"**, which can be expanded beyond the scope according to actual testing needs. The main types of this security audit include:

(1) Dependency Check

A comprehensive check of the software's dependency libraries was conducted to ensure all external libraries and frameworks are up-to-date and free of known security vulnerabilities.

(2) Automated Static Code Analysis

Static code analysis tools were used to find common programming errors, potential security vulnerabilities, and code patterns that do not conform to best practices.

(3) Fuzz Testing

A large amount of randomly generated data was inputted into the software to try and trigger potential errors and exceptional paths.

(4) Manual Review

The scope of the code is explained in section 1.2.

(5) Audit Process

- Clarify the scope, objectives, and key requirements of the audit.
- Collect related materials such as software documentation, architecture diagrams, and lists of dependency libraries to provide background information for the audit.
- Use automated tools to generate a list of the software's dependency libraries and employ professional tools to scan these libraries for security vulnerabilities, identifying outdated or known vulnerable dependencies.
- Select and configure automated static analysis tools suitable for the project, perform automated scans to identify security vulnerabilities, non-standard coding, and potential risk points in the code. Evaluate the scanning results to determine which findings require further manual review.

- Design a series of fuzz testing cases aimed at testing the software's ability to handle exceptional data inputs. Analyze the issues found during the testing to determine the defects that need to be fixed.
- Based on the results of the preliminary automated analysis, develop a detailed code review plan, identifying the focus of the review. Experienced auditors perform line-by-line reviews of key components and sensitive functionalities in the code.
- If any issues arise during the audit process, communicate with the code owner in a timely manner. The code owners should actively cooperate (this may include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- Necessary information during the audit process will be well documented in a timely manner for both the audit team and the code owner.

2 Summary

This report has been commissioned by Tabi with the objective of identifying any potential issues and vulnerabilities within the source code of the Tabi repository, as well as in the repository dependencies that are not part of an officially recognized library. In this audit, we have employed the following techniques to identify potential vulnerabilities and security issues:

(1) Dependency Check

A comprehensive analysis of the software's dependency libraries was conducted using the dependency check tool.

(2) Automated Static Code Analysis

The code quality was examined using a code scanner.

(3) Manual Code Review

The primary focus of the manual code review was:

- tabilabs/tabi

During the audit, we identified 8 issues of varying severity, listed below.

ID	Title	Severity	Status
GO-1	Dependency Issue	Medium	Acknowledged
TX-1	A `.proto` File Is Missing The `cosmos.msg.v1.signer` Metadata	Medium	Acknowledged
CAP-1	`Antehandler` Can Be Bypassed	Critical	Fixed
KEY-1	Include Length for Keys with Address Type Content	Medium	Fixed
LTX-1	Hardcoded ChainIDs Could Lead to Replay Attacks	Major	Fixed

MEM-1	Improvement Loops in Member Management	Informational	Fixed
MSG-1	Inadequate Validation for ReportType Range in MsgCommitReport	Minor	Fixed
PAR-1	Improvement Validation Logic in `UpdateSaleLevel` Function	Informational	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the Tabi repository :

Admin

- `UpdateParams` : Allows the admin to update module parameters.
- `AddAuthorizedMembers` : Adds members to the list of authorized participants.
- `RemoveAuthorizedMembers` : Removes members from the list of authorized participants.
- `UpdateSaleLevel` : Updates the sale level of the system.
- `CreateCaptainNode` : Creates a new captain node under a specified division.
- `CommitReport` : Commits a validated report to the chain.
- `CommitComputingPower` : Records computing power rewards for specific users.
- `ClaimComputingPower` : Claims computing power for a specified node.

User

- `Claims` : Withdraws accumulated rewards, allowing the sender to specify a receiver address.
- `ConvertTabi` : Converts Tabi tokens to VeTabi tokens.
- `ConvertVetabi` : Converts VeTabi tokens to Tabi tokens and issues a voucher for future redemption.
- `WithdrawTabi` : Withdraws Tabi tokens or returns VeTabi tokens using a voucher.
- `CancelConvert` : Cancels an ongoing conversion and unlocks the associated VeTabi tokens.

4 Findings

GO-1 Dependency Issue

Severity: Medium

Discovery Methods: Dependency Check

Status: Acknowledged

Code Location:

go.mod#1

Descriptions:

This project uses some vulnerable dependency libraries, many of which are related to the Cosmos SDK

```
Vulnerability #1: GO-2024-3339
    Transaction decoding may result in a stack overflow or resource exhaustion
    in github.com/cosmos/cosmos-sdk
  More info: https://pkg.go.dev/vuln/GO-2024-3339
  Module: github.com/cosmos/cosmos-sdk
    Found in: github.com/cosmos/cosmos-sdk@v0.46.15
    Fixed in: github.com/cosmos/cosmos-sdk@v0.47.15
    Example traces found:

Vulnerability #2: GO-2024-3279
    Mismatched bit-length validation in can lead to panic in cosmossdk.io/math
  More info: https://pkg.go.dev/vuln/GO-2024-3279
  Module: cosmossdk.io/math
    Found in: cosmossdk.io/math@v1.0.0-rc.0
    Fixed in: cosmossdk.io/math@v1.4.0
    Example traces found:

Vulnerability #3: GO-2024-2948
    Code Execution on Git update in github.com/hashicorp/go-getter
  More info: https://pkg.go.dev/vuln/GO-2024-2948
  Module: github.com/hashicorp/go-getter
    Found in: github.com/hashicorp/go-getter@v1.7.0
    Fixed in: github.com/hashicorp/go-getter@v1.7.5
    Example traces found:
```

```
Vulnerability #4: GO-2024-2874
    Inter-Blockchain Communication (IBC) protocol "Huckleberry" vulnerability in
    github.com/cosmos/ibc-go
  More info: https://pkg.go.dev/vuln/GO-2024-2874
  Module: github.com/cosmos/ibc-go/v6


Vulnerability #5: GO-2024-2800
    Argument injection when fetching remote default Git branches in
    github.com/hashicorp/go-getter
  More info: https://pkg.go.dev/vuln/GO-2024-2800
  Module: github.com/hashicorp/go-getter
    Found in: github.com/hashicorp/go-getter@v1.7.0
    Fixed in: github.com/hashicorp/go-getter@v1.7.4
    Example traces found:

Vulnerability #6: GO-2024-2694
    Potential Reentrancy using Timeout Callbacks in ibc-hooks in
    github.com/cosmos/ibc-go
  More info: https://pkg.go.dev/vuln/GO-2024-2694
  Module: github.com/cosmos/ibc-go/v6
    Found in: github.com/cosmos/ibc-go/v6@v6.1.1
    Fixed in: github.com/cosmos/ibc-go/v6@v6.3.0
    Example traces found:

Vulnerability #7: GO-2024-2687
    HTTP/2 CONTINUATION flood in net/http
  More info: https://pkg.go.dev/vuln/GO-2024-2687
  Module: golang.org/x/net
    Found in: golang.org/x/net@v0.9.0
    Fixed in: golang.org/x/net@v0.23.0

Vulnerability #8: GO-2024-2611
    Infinite loop in JSON unmarshaling in google.golang.org/protobuf
  More info: https://pkg.go.dev/vuln/GO-2024-2611
  Module: google.golang.org/protobuf
    Found in: google.golang.org/protobuf@v1.30.0
    Fixed in: google.golang.org/protobuf@v1.33.0
    Example traces found:
      #1: server/json_rpc.go:88:26: server.StartJSONRPC calls http.Server.Serve, which
eventually calls json.Decoder.Peek

Vulnerability #9: GO-2024-2584
```

```
    Slashing evasion in github.com/cosmos/cosmos-sdk
  More info: https://pkg.go.dev/vuln/GO-2024-2584
  Module: github.com/cosmos/cosmos-sdk
    Found in: github.com/cosmos/cosmos-sdk@v0.46.15
    Fixed in: github.com/cosmos/cosmos-sdk@v0.47.10
    Example traces found:

Vulnerability #10: GO-2024-2572
    Missing BlockedAddressed Validation in Vesting Module in
    github.com/cosmos/cosmos-sdk
  More info: https://pkg.go.dev/vuln/GO-2024-2572
  Module: github.com/cosmos/cosmos-sdk
    Found in: github.com/cosmos/cosmos-sdk@v0.46.15
    Fixed in: github.com/cosmos/cosmos-sdk@v0.47.9
    Example traces found:

Vulnerability #11: GO-2024-2571
    Invalid block proposal in github.com/cosmos/cosmos-sdk
  More info: https://pkg.go.dev/vuln/GO-2024-2571
  Module: github.com/cosmos/cosmos-sdk
    Found in: github.com/cosmos/cosmos-sdk@v0.46.15
    Fixed in: github.com/cosmos/cosmos-sdk@v0.47.9
    Example traces found:
      #1: app/app.go:224:28: app.NewTabi calls baseapp.NewBaseApp

Vulnerability #12: GO-2023-2409
    Denial of service when decrypting attacker controlled input in
    github.com/dvsekhvalnov/jose2go
  More info: https://pkg.go.dev/vuln/GO-2023-2409
  Module: github.com/dvsekhvalnov/jose2go
    Found in: github.com/dvsekhvalnov/jose2go@v1.5.0
    Fixed in: github.com/dvsekhvalnov/jose2go@v1.5.1-0.20231206184617-48ba0b76bc88
    Example traces found:

Vulnerability #13: GO-2023-2153
    Denial of service from HTTP/2 Rapid Reset in google.golang.org/grpc
  More info: https://pkg.go.dev/vuln/GO-2023-2153
  Module: google.golang.org/grpc
    Found in: google.golang.org/grpc@v1.54.0
    Fixed in: google.golang.org/grpc@v1.56.3
    Example traces found:

Vulnerability #14: GO-2023-1881
```

```
    The x/crisis package does not charge ConstantFee in
    github.com/cosmos/cosmos-sdk
  More info: https://pkg.go.dev/vuln/GO-2023-1881
  Module: github.com/cosmos/cosmos-sdk
    Found in: github.com/cosmos/cosmos-sdk@v0.46.15
    Fixed in: N/A

Vulnerability #15: GO-2023-1821
    The x/crisis package does not cause chain halt in
    github.com/cosmos/cosmos-sdk
  More info: https://pkg.go.dev/vuln/GO-2023-1821
  Module: github.com/cosmos/cosmos-sdk
    Found in: github.com/cosmos/cosmos-sdk@v0.46.15
    Fixed in: N/A

Your code is affected by 15 vulnerabilities from 8 modules.
This scan also found 2 vulnerabilities in packages you import and 9
vulnerabilities in modules you require, but your code doesn't appear to call
these vulnerabilities.
Use '-show verbose' for more details.
```

Suggestion:

It is recommended to update the dependency libraries to the latest version. The "govulncheck" tool can be used to check for vulnerabilities

Resolution:

The project team will fix this issue in a future version.

TX-1 A `.proto` File Is Missing The `cosmos.msg.v1.signer` Metadata

Severity: Medium

Discovery Methods: Manual Review

Status: Acknowledged

Code Location:

proto/tabi/token-convert/v1/tx.proto#28;

proto/tabi/token-convert/v1/tx.proto#39

Descriptions:

In the file `(proto/tabi/token-convert/v1/tx.proto)`, none of the messages have the "cosmos.msg.v1.signer" metadata specified.

```
// MsgConvertTabi represents a message to convert Tabi to Vetabi.
message MsgConvertTabi {
 // coin
 cosmos.base.v1beta1.Coin coin = 1 [(gogoproto.nullable) = false];
 // sender
 string sender = 2 [(cosmos_proto.scalar) = "cosmos.AddressString"];
}

// MsgConvertTabiResponse defines the Msg/ConvertTabi response type.
message MsgConvertTabiResponse {}

// MsgConvertVetabi represents a message to convert Vetabi to Tabi.
message MsgConvertVetabi {
 // coin
 cosmos.base.v1beta1.Coin coin = 1 [(gogoproto.nullable) = false];
 // strategy
 string strategy = 2;
 // sender
 string sender = 3 [(cosmos_proto.scalar) = "cosmos.AddressString"];
}
```

Suggestion:

Add "cosmos.msg.v1.signer" metadata to every message in the `.proto` file.

Resolution:

This issue does not have any security impact.The project team will fix this issue in a future version.

CAP-1 `Antehandler` Can Be Bypassed

Severity: Critical

Discovery Methods: Manual Review

Status: Fixed

Code Location:

app/ante/tabi/captains.go#49

Descriptions:

In the file (app/ante/tabi/captains.go), the `restrict` function filters 5 types of `sdk.Msg`, but it can be bypassed by the `MsgExec` message from the `x/authz` module. For detailed information, refer to this link: `https://jumpcrypto.com/writing/bypassing-ethermint-ante-handlers/`

Suggestion:

In the `checkDisabledMsgs` function in the file `app/ante/cosmos/authz.go`, `evmos` defines a blacklist Msg mechanism, which can be referred to for implementation.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

KEY-1 Include Length for Keys with Address Type Content

Severity: Medium

Discovery Methods: Manual Review

Status: Fixed

Code Location:

x/captains/types/keys.go#213;

x/captains/types/keys.go#169

Descriptions:

In the file `(x/captains/types/keys.go)`, the `NodeByOwnerPrefixStoreKey` function does not account for the variable length of the `sdk.AccAddress` type when constructing the key, which may lead to potential key collisions.

Key construction algorithm of `NodeByOwnerPrefixStoreKey`: `<prefix_key><owner><delimiter>`

Key collision scenario:

`<prefix_key><owner><delimiter>`

01 8800 00

01 88 00

Suggestion:

All `sdk.AccAddress` instances must be processed through the following function:

`address.MustLengthPrefix(address)`

Resolution:

This issue has been fixed. The client has adopted our suggestions.

LTX-1 Hardcoded ChainIDs Could Lead to Replay Attacks

Severity: Major

Discovery Methods: Manual Review

Status: Fixed

Code Location:

x/evm/types/legacy_tx.go#202

Descriptions:

The current implementation includes hardcoded chain IDs 9789 and 9788, representing the mainnet and testnet, respectively, as indicated in the following snippet:

```
// MainnetChainID defines the Tabi EIP155 chain ID for mainnet
MainnetChainID = "tabi_9788"
// TestnetChainID defines the Tabi EIP155 chain ID for testnet
TestnetChainID = "tabi_9789"
```

```
if !(chainID.Cmp(big.NewInt(9789)) == 0 || chainID.Cmp(big.NewInt(9788)) == 0) {
    return errorsmod.Wrapf(
        errortypes.ErrInvalidChainID,
        "chain ID must be 9789 or 9788 on Tabi, got %s", chainID,
    )
}
```

However, Tabi has launched two testnets, `testnet.tabiscan.com` and `testnetv2.tabiscan.com`, which also utilize chain IDs 9789 and 9788. Reusing these chain IDs in production could result in cross-chain replay attacks.

Suggestion:

Assign and configure a new, unique chain ID for the Tabi mainnet that is not currently in use by any testnet. Update the source code to replace the hardcoded values.

Resolution:

This issue has been fixed. commit: 44e732be7ae81630ffc9b27faa51351cebd624c7

MEM-1 Improvement Loops in Member Management

Severity: Informational

Discovery Methods: Manual Review

Status: Fixed

Code Location:

x/captains/keeper/members.go#82,18

Descriptions:

In both the `SetAuthorizedMembers` and `HasAuthorizedMember` functions, the loops continue iterating through all elements even after the necessary condition is met (`allowAdd = false` or `allowAuthz = true`). While this does not affect functionality, it results in slight inefficiency.

Suggestion:

Introduce break statements to exit the loops early once the condition is satisfied. This minor adjustment enhances code efficiency without altering behavior.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

MSG-1 Inadequate Validation for ReportType Range in MsgCommitReport

Severity: Minor

Discovery Methods: Manual Review

Status: Fixed

Code Location:

x/captains/types/msg.go#155

Descriptions:

```
  if msg.ReportType > ReportType_REPORT_TYPE_END || msg.ReportType ==
ReportType_REPORT_TYPE_UNSPECIFIED {
    return errorsmod.Wrap(sdkerrors.ErrInvalidRequest, "invalid report type")
  }
```

In the `ValidateBasic` method of the `MsgCommitReport`, the validation for `msg.ReportType` checks whether the value is greater than `ReportType_REPORT_TYPE_END` or equal to `ReportType_REPORT_TYPE_UNSPECIFIED`. However, since `msg.ReportType` is of type int32, it can potentially hold negative values, which are not addressed by the current condition. This could lead to invalid ReportType values passing the validation.

Suggestion:

Modify the validation logic to ensure that msg.ReportType is within the expected range of 1-4 (inclusive).

Resolution:

This issue has been fixed. The client has adopted our suggestions.

PAR-1 Improvement Validation Logic in `UpdateSaleLevel` Function

Severity: Informational

Discovery Methods: Manual Review

Status: Fixed

Code Location:

x/captains/keeper/params.go#41

Descriptions:

```
func (k Keeper) UpdateSaleLevel(ctx sdk.Context, newSaleLevel uint64) (uint64, error) {
    params := k.GetParams(ctx)
    oldSaleLevel := params.CurrentSaleLevel
    if oldSaleLevel > newSaleLevel {
        return oldSaleLevel, types.ErrInvalidSaleLevel
    }

    params.CurrentSaleLevel = newSaleLevel
    if err := k.SetParams(ctx, params); err != nil {
        return oldSaleLevel, err
    }

    return oldSaleLevel, nil
}
```

In the `UpdateSaleLevel` function, the condition `if oldSaleLevel > newSaleLevel` does not account for cases where the new sale level is equal to the current sale level (`oldSaleLevel == newSaleLevel`). Allowing such updates results in unnecessary operations without any effect on the state.

Suggestion:

Revise the condition to `if oldSaleLevel >= newSaleLevel` to restrict updates to only when the new sale level is strictly greater than the current one. This will eliminate redundant updates and improve the function's logic.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

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 or assets 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 or assets 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.