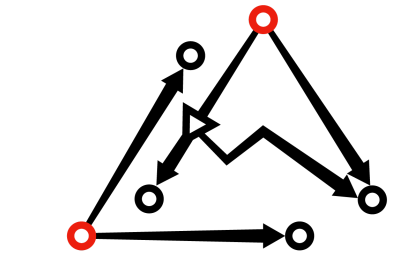

Least Authority

PRIVACY MATTERS

Starport Network

Security Audit Report

All in Bits

Final Audit Report: 22 July 2022

Table of Contents

Overview

Background

All in Bits has requested that Least Authority perform a security audit of the Starport Network, a blockchain that facilitates the launch of new blockchains using the Cosmos-SDK framework.

Project Dates

- **April 18 - June 21:** Initial Code Review *(Completed)*
- **June 24:** Delivery of Initial Audit Report *(Completed)*
- **July 19 - 20 :** Verification Review *(Completed)*
- **July 22:** Delivery of Final Audit Report *(Completed)*

The dates for verification and delivery of the Final Audit Report will be determined upon notification from the All in Bits team that the code is ready for verification.

Review Team

- Shareef Maher Dweikat, Security Research and Engineer
- Alejandro Flores, Security Researcher and Engineer
- Denis Kolegov, Security Researcher and Engineer
- ElHassan Wanas, Security Researcher and Engineer

Coverage

Target Code and Revision

For this audit, we performed research, investigation, and review of the Starport Network followed by issue reporting, along with mitigation and remediation instructions as outlined in this report.

The following code repository is considered in-scope for the review:

- Starport Network:
  https://github.com/tendermint/spn/releases/tag/v0.2.0

Specifically, we examined the Git revision for our initial review:

`f5d9d7013e9a26a745a01b3d0771e856f1c11d37`

For the verification, we examined the Git revision:

09f44be1e0272c1873b2448f839342d5187eae67

For the review, this repository was cloned for use during the audit and for reference in this report:

Starport Network: https://github.com/LeastAuthority/AllinBits_Starport_Network

All file references in this document use Unix-style paths relative to the project's root directory.

In addition, any dependency and third-party code, unless specifically mentioned as in-scope, were considered out of scope for this review.

Supporting Documentation

The following documentation was available to the review team:

- starport-network-audit.zip *(shared with Least Authority via Slack on 25 January 2022)*
- participation_module.pdf *(shared with Least Authority via Slack on 11 February 2022)*

Areas of Concern

Our investigation focused on the following areas:

- Correctness of the implementation;
- Common and case-specific implementation errors;
- Adversarial actions and other attacks on the Starport Network;
- Potential misuse and gaming of the Starport Network;
- Attacks that impacts funds, such as the draining or manipulation of funds;
- Mismanagement of funds via transactions;
- Denial of Service (DoS) and other security exploits that would impact the Starport Network's intended use or disrupt the execution;
- Vulnerabilities in the Starport Network's code;
- Vulnerabilities in the Starport Network's interactions with other existing Cosmos-SDK modules and components;
- Protection against malicious attacks and other ways to exploit the Starport Network;
- Performance of the components under load, leading to potential security vulnerabilities;
- Exposure of any critical information during interaction with any external libraries;
- Use and management of dependencies and the process for updating dependent libraries;
- Data privacy, data leaking, and information integrity;
- Inappropriate permissions and excess authority; and
- Anything else as identified during the initial analysis phase.

Findings

General Comments

Starport Network is a blockchain that aims to simplify the launching of new blockchains within the Cosmos Ecosystem by utilizing a command-line interface (CLI) tool. Any user launching a blockchain using the Starport Network becomes the coordinator of that blockchain and can define the set of validators at the launch and set various parameters, such as staking amounts and gas prices. All the information needed to create a blockchain is then stored in the Starport Network and can be used by the set of validators at the launch to generate the genesis of the blockchain by utilizing a CLI tool.

The Starport Network relies on Cosmos-SDK modules for the signing of transactions and the security and verification of inputs, thereby making the Cosmos-SDK a single point of failure in case of a vulnerability. We found that the system's high dependency on the Cosmos-SDK might increase the complexity of the system in case there is a need for refactoring the code in the future.

Several Cosmos modules that compose the Starport Network are adapted without further modification. Our team checked those modules for implementation errors and bugs and did not identify any. Our team focused on those modules for which logic has been implemented by the All in Bits team and found the logic implemented to be straightforward, with no implementation errors discovered. These modules rely on the Cosmos-SDK functionality for the most part, with modifications made to the functionality of the Keeper of each module.

Although the system design and implementation generally adhere to best practices, we identified issues and suggestions to improve the overall security and quality of the implementation.

System Design

Security has been taken into consideration in the design of the Starport Network as demonstrated by a similar design for each of the modules, appropriate whitelisting and validation of expected inputs, and logical processes. We were unable to identify exploits, successfully take advantage of race conditions, or find any access control issues. However, we did identify issues in the design of the Starport Network, including a function that checks that an account has not reached the total number of shares, which could potentially result in a spread between the expected total number of shares and the actual total number of shares. We recommend that the function logic be updated (Issue D).

The connection to the server to download the Starport package accepts connections in TLS 1.0, which has been upgraded and could make the transmissions vulnerable to Man-in-the-Middle (MITM) attacks. We recommend only allowing the server to establish TLS 1.2 connections or greater (Issue C).

Our team briefly reviewed the design and implementation of components of the Starport Network that were out for scope for this review in order to build a better understanding of the intended functionality of the in-scope components. We found there is a potential opportunity to perform a Server-Side Request Forgery (SSRF) attack as a result of insufficient input validation implemented for the function `getHashFromUrl`. This could lead to information about the network being leaked to an attacker. We recommend that the input URL be validated appropriately (Issue B).

Code Quality

We found that the Starport Network implementation codebase is well organized and generally adheres to best practices. However, we found some instances of duplicate functionality that reduces the readability of the code and increases the risk of errors in the maintenance of the codebase. We recommend that duplicate functionality be removed and that function names describe the purpose of the function accurately (Suggestion 3). We also suggest an improvement in the error handling of the function `RegisterProviderClientIDFromChannelID` to prevent system panics or unexpected behavior (Suggestion 2).

Tests

There are a wide number of tests that cover the expected scope for each function and module of the code. However, we identified an issue in the network tests performed by the system whereby race conditions occur as a result of two goroutines accessing the same variable concurrently. This could lead to the tests behaving unexpectedly. We recommend that the tests be reimplemented to prevent race conditions from occurring (Issue A).

Documentation

The project documentation, which included helpful diagrams, was accurate and provided a sufficient overview of the system, each of the components, and their interactions.

Code Comments

The codebase is well commented in accordance with Go code style recommendations.

Scope

The scope included the security-critical components of the Starport Network implementation. However, low-level CLI functionality for interaction with the blockchain was out of scope for this review.

Dependencies

We found that several dependencies of the implementation are outdated. The use of unmaintained dependencies could lead to the introduction of vulnerabilities and bugs into the Starport codebase. We recommend using up-to-date, well-maintained libraries (Suggestion 1).

Specific Issues & Suggestions

We list the issues and suggestions found during the review, in the order we reported them. In most cases, remediation of an issue is preferable, but mitigation is suggested as another option for cases where a trade-off could be required.

ISSUE / SUGGESTION	STATUS
Issue A: Data Races in the Networks Tests	Unresolved
Issue B: Potential SSRF in getHashFromURL (Out of Scope)	Resolved
Issue C: Insecure TLS Versions Are Supported	Unresolved
Issue D: The Check for Maximum Shares Is Incorrect	Resolved
Suggestion 1: Update Outdated Dependencies in Go	Unresolved
Suggestion 2: Fix Null clientID in RegisterProviderClientIDFromChannelID	Resolved
Suggestion 3: Change Identical Function Definitions for Specific Use Cases	Resolved.

Issue A: Data Races in the Networks Tests

Synopsis

The tests contain the code that creates two goroutines accessing the same variable concurrently, and at least one of the accesses is a write.

To reproduce the issue, run the tests with the `race` flag as follows:

```
go test -race ./…
```

The output will contain a data race report detected by the Go race detector:

```
WARNING: DATA RACE

Read at 0x00c000d928b0 by goroutine 60:

github.com/cosmos/cosmos-sdk/server/api.(*Server).Close()

github.com/cosmos/cosmos-sdk@v0.45.2/server/api/server.go:123 +0x250

github.com/cosmos/cosmos-sdk/testutil/network.(*Network).Cleanup()

github.com/cosmos/cosmos-sdk@v0.45.2/testutil/network/network.go:481 +0x222

github.com/tendermint/spn/testutil/network.New()
```

```
/spn/testutil/network/network.go:33 +0x84

github.com/tendermint/spn/x/campaign/client/cli_test.networkWithCampaignObject
s()

/spn/x/campaign/client/cli/query_campaign_test.go:33 +0x4a4

github.com/tendermint/spn/x/campaign/client/cli_test.TestListCampaign()

      /spn/x/campaign/client/cli/query_campaign_test.go:88 +0x4c

...

Previous write at 0x00c000d928b0 by goroutine 40:

github.com/cosmos/cosmos-sdk/server/api.(*Server).Start()

github.com/cosmos/cosmos-sdk@v0.45.2/server/api/server.go:109 +0x2e4

github.com/cosmos/cosmos-sdk/testutil/network.startInProcess.func1()

github.com/cosmos/cosmos-sdk@v0.45.2/testutil/network/util.go:86 +0x104

…

Goroutine 60 (running) created at:

testing.(*T).Run()

/usr/local/opt/go/libexec/src/testing/testing.go:1486 +0x724

…

Goroutine 40 (running) created at:

github.com/cosmos/cosmos-sdk/testutil/network.startInProcess()

github.com/cosmos/cosmos-sdk@v0.45.2/testutil/network/util.go:85 +0xe0b

github.com/tendermint/spn/testutil/network.New()

/spn/testutil/network/network.go:33 +0x84

github.com/tendermint/spn/x/campaign/client/cli_test.networkWithCampaignObject
s()

/spn/x/campaign/client/cli/query_campaign_test.go:33 +0x4a4

github.com/tendermint/spn/x/campaign/client/cli_test.TestListCampaign()
```

Impact

Data races would most likely impact the logic of the system or cause crashes or memory corruptions, which could lead to a Denial of Service attack. These data races impact network tests only.

Mitigation

We recommend that tests be reimplemented to guarantee that only one network test is running at any given time, as is required by the network test utilities.

Status

The Starport team has responded that CLI tests are not critical for the security of the blockchain. As such, they do not intend to address this issue.

Verification

Unresolved.

Issue B: Potential SSRF in getHashFromURL (Out of Scope)

Location

`x/launch/client/cli/tx_create_chain.go#L90`

Synopsis

The function `getHashFromUrl` provides an oracle that can be used to determine whether a local port is open or closed. This creates an opportunity for an SSRF attack. For example, if the port 22 on server.com were open, then the response would be as follows:

```
net/http: HTTP/1.x transport connection broken: malformed HTTP response
"SSH-2.0-OpenSSH_5.3"
```

If the port is closed, then the response would be as follows:

```
dial tcp 92.63.64.162:23: connect: connection refused
```

Impact

This vulnerability can be used to implement network reconnaissance attacks, which reveal information about the network that can be used in a larger attack.

Remediation

We recommend that input URLs be checked to only contain target ports (e.g., 443).

Status

The Starport team implemented the validation of the target port number as suggested.

Verification

Resolved.

Issue C: Insecure TLS Versions Are Supported

Synopsis

To install Starport or Ignite CLI, one of the following commands must be run:

```
curl https://get.starport.network/tendermint/spn@latest! | sudo bash
```

or

```
curl https://get.ignite.com/cli! | bash
```

These URLs share the same network host with the same TLS protocol configuration.

An SSL Labs scanner run against the host shows that TLS 1.0 and TLS 1.1 protocols are both supported.

Impact

TLS 1.0 and TLS 1.1 do not support strong cipher suites and security mechanisms. As a result, attackers could mount MITM attacks, allowing the modification of the downloading software packages.

Mitigation

Use TLS 1.2 and TLS 1.3 protocols only.

Status

The Starport team has responded that the installation links will be removed in the future. As a result, the suggestion remains unresolved at the time of this verification.

Verification

Unresolved.

Issue D: The Check for Maximum Shares Is Incorrect

Location

`x/campaign/types/shares.go`

Synopsis

The function `IsTotalSharesReached` is intended to check that share allocations do not exceed the `maximumTotalShareNumber`. However, share allocations could exceed the expected `maximumTotalShareNumber` by providing the same `coin` denomination multiple times with all amounts below the `maximumTotalShareNumber`. The implementation of `IsTotalSharesReached` compares every Coin in the shares' amount to ensure it is not more than the `maximumTotalShareNumber`. A valid request could provide multiple Coin values in the `AddShares` message's shares parameter, which would allow an arbitrary number of allocations of the same denomination so long as all the amounts are less than the `maximumTotalShareNumber`.

This current implementation of the function would not prevent these allocations from occurring.

Impact

Incorrect tracking of the allocation of shares could result in an attacker draining assets illegitimately.

Remediation

We recommend that the check verify the sum of the shares' allocation amounts against the `maximumTotalShareNumber`.

Status

The Starport team has added a call to Cosmos SDK's `Coins` validation function, which checks for unsorted or duplicated denominations. Additionally, a check has been added to ensure that all coins provided have the `share` prefix in their denomination.

Verification

Resolved.

Suggestions

Suggestion 1: Update Outdated Dependencies in Go

Synopsis

The `ibc-go`, `cosmos-sdk`, `cobra`, `tendermint`, `tendermint/starport`, `tendermint/tm-db`, and `protobuf` dependencies are reported as outdated by the `go-mod-outdated` utility and should be updated. Outdated dependencies may have software bugs and vulnerabilities, which may impact the security of the entire system.

Mitigation

We recommend updating the aforementioned dependencies. We also recommend regularly running the `go-mod-outdated` tool to capture any outdated dependencies.

Status

The Starport team has responded that updating packages requires breaking changes in the chain logic and the audited code. As a result, outdated packages have not been updated as of the time of this verification.

Verification

Unresolved.

Suggestion 2: Fix Null clientID in RegisterProviderClientIDFromChannelID

Location

`x/monitoringc/keeper/handshake.go#L43`

Synopsis

The `RegisterProviderClientIDFromChannelID` function registers the provider `clientID`. However, if an error is returned, then the `clientID` will be a null value. In some cases, this could result in panics or undefined behavior.

Mitigation

We recommend removing the empty `clientID` from the returned error.

Status

The Starport team has removed the empty `clientID` from the returned error as suggested.

Verification

Resolved.

Suggestion 3: Change Identical Function Definitions for Specific Use Cases

Location

/x/launch/types/keys.go#L67

/x/launch/types/keys.go#L81

Use the following function definition:

/x/launch/types/keys.go#L53

/x/campaign/types/keys.go#L64

Uses the following function definition:

/x/campaign/types/keys.go#L50

Synopsis

In these two different files, the definition of one function within the code is duplicated in the other functions of the code. This is an unnecessary definition of extra functions if one is already set. If the usage of these functions varies amongst them, this should also be made explicit in the code and by avoiding using the exact code.

Mitigation

We recommend that a general function be defined instead of having two or three identical functions. In cases where the functions have different purposes, we recommend the accurate naming of functions to better represent specific use cases.

Status

The Starport team modified function definitions as suggested.

Verification

Resolved.

About Least Authority

We believe that people have a fundamental right to privacy and that the use of secure solutions enables people to more freely use the Internet and other connected technologies. We provide security consulting services to help others make their solutions more resistant to unauthorized access to data and unintended manipulation of the system. We support teams from the design phase through the production launch and after.

The Least Authority team has skills for reviewing code in multiple Languages, such as C, C++, Python, Haskell, Rust, Node.js, Solidity, Go, JavaScript, ZoKrates, and circom, for common security vulnerabilities and specific attack vectors. The team has reviewed implementations of cryptographic protocols and distributed system architecture in cryptocurrency, blockchains, payments, smart contracts, and zero-knowledge protocols. Additionally, the team can utilize various tools to scan code and networks and build custom tools as necessary.

Least Authority was formed in 2011 to create and further empower freedom-compatible technologies. We moved the company to Berlin in 2016 and continue to expand our efforts. We are an international team that believes we can have a significant impact on the world by being transparent and open about the work we do.

For more information about our security consulting, please visit https://leastauthority.com/security-consulting/.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis

Our audit techniques include manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's website to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. As we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation. We hypothesize what vulnerabilities may be present and possibly resulting in Issue entries, then for each, we follow the following Issue Investigation and Remediation process.

Documenting Results

We follow a conservative and transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even before having verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this, we analyze the feasibility of an attack in a live system.

Suggested Solutions

We search for immediate and comprehensive mitigations that live deployments can take, and finally, we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our Initial Audit Report, and before we perform a verification review.

Before our report, including any details about our findings and the solutions are shared, we like to work with your team to find reasonable outcomes that can be addressed as soon as possible without an overly negative impact on pre-existing plans. Although the handling of issues must be done on a case-by-case basis, we always like to agree on a timeline for a resolution that balances the impact on the users and the needs of your project team.

Resolutions & Publishing

Once the findings are comprehensively addressed, we complete a verification review to assess that the issues and suggestions are sufficiently addressed. When this analysis is completed, we update the report and provide a Final Audit Report that can be published in whole. If there are critical unaddressed issues, we suggest the report not be published and the users and other stakeholders be alerted of the impact. We encourage that all findings be dealt with and the Final Audit Report be shared publicly for the transparency of efforts and the advancement of security learnings within the industry.