Zellic

GotSui

Web Application Security Assessment

October 16, 2023

Prepared for:

Mysten Labs Ltd.

Prepared by:

SeungHyeon Kim and Yuhang Wu

Zellic Inc.

Contents

About Zellic

Zellic was founded in 2020 by a team of blockchain specialists with more than a decade of combined industry experience. We are leading experts in smart contracts and Web3 development, cryptography, web security, and reverse engineering. Before Zellic, we founded perfect blue, the top competitive hacking team in the world. Since then, our team has won countless cybersecurity contests and blockchain security events.

Zellic aims to treat clients on a case-by-case basis and to consider their individual, unique concerns and business needs. Our goal is to see the long-term success of our partners rather than simply provide a list of present security issues. Similarly, we strive to adapt to our partners' timelines and to be as available as possible. To keep up with our latest endeavors and research, check out our website zellic.io or follow @zellic_io on Twitter. If you are interested in partnering with Zellic, please contact us at hello@zellic.io.

1 Executive Summary

Zellic conducted a security assessment for Mysten Labs Ltd. from October 11th, 2023 to October 18th, 2023. During this engagement, Zellic reviewed GotSui's code for security vulnerabilities, design issues, and general weaknesses in security posture.

1.1 Goals of the Assessment

In a security assessment, goals are framed in terms of questions that we wish to answer. These questions are agreed upon through close communication between Zellic and the client. In this assessment, we sought to answer the following questions:

- Could there be any problems that might result in the loss of user funds?
- Are there any potential issues that could lead to the loss of user data?
- Are there any issues that might lead to a leakage of user privacy?
- Are there any vulnerabilities that could result in a DOS attack on the project?

1.2 Non-goals and Limitations

We did not assess the following areas that were outside the scope of this engagement:

- Deployment issues
- Backend components
- Infrastructure relating to the project
- Key custody

Due to the time-boxed nature of security assessments in general, there are limitations in the coverage an assessment can provide.

During this assessment, the absence of a specific project-execution environment prevented us from evaluating potential vulnerabilities related to third-party code hosting, domain names, emails, DNS, and other related environmental factors.

1.3 Results

During our assessment on the scoped GotSui modules, we discovered one finding, which was of low impact.

Additionally, Zellic recorded its notes and observations from the assessment for

Mysten Labs Ltd.'s benefit in the Discussion section (4) at the end of the document.

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	0
Low	1
Informational	0

2 Introduction

2.1 About GotSui

GotSui is a browser-based application that makes sending SUI as easy as sending a link.

2.2 Methodology

During a security assessment, Zellic works through standard phases of security auditing including both automated testing and manual review. These processes can vary significantly per engagement, but the majority of the time is spent on a thorough manual review of the entire scope.

Alongside a variety of tools and analyzers used on an as-needed basis, Zellic focuses primarily on the following classes of security and reliability issues:

Basic coding mistakes. Many critical vulnerabilities in the past have been caused by simple, surface-level mistakes that could have easily been caught ahead of time by code review. Depending on the engagement, we may also employ sophisticated analyzers such as model checkers, theorem provers, fuzzers, and so on as necessary. We also perform a cursory review of the code to familiarize ourselves with the modules.

Business logic errors. Business logic is the heart of any smart contract application. We examine the specifications and designs for inconsistencies, flaws, and weaknesses that create opportunities for abuse. For example, these include problems like unrealistic tokenomics or dangerous arbitrage opportunities. To the best of our abilities, time permitting, we also review the contract logic to ensure that the code implements the expected functionality as specified in the platform's design documents.

Integration risks. Several well-known exploits have not been the result of any bug within the contract itself; rather, they are an unintended consequence of the contract's interaction with the broader DeFi ecosystem. Time permitting, we review external interactions and summarize the associated risks: for example, flash loan attacks, oracle price manipulation, MEV/sandwich attacks, and so on.

Code maturity. We look for potential improvements in the codebase in general. We look for violations of industry best practices and guidelines and code quality standards. We also provide suggestions for possible optimizations, such as gas

optimization, upgradeability weaknesses, centralization risks, and so on.

For each finding, Zellic assigns it an impact rating based on its severity and likelihood. There is no hard-and-fast formula for calculating a finding's impact. Instead, we assign it on a case-by-case basis based on our judgment and experience. Both the severity and likelihood of an issue affect its impact. For instance, a highly severe issue's impact may be attenuated by a low likelihood. We assign the following impact ratings (ordered by importance): Critical, High, Medium, Low, and Informational.

Zellic organizes its reports such that the most important findings come first in the document, rather than being strictly ordered on impact alone. Thus, we may sometimes emphasize an "Informational" finding higher than a "Low" finding. The key distinction is that although certain findings may have the same impact rating, their *importance* may differ. This varies based on various soft factors, like our clients' threat models, their business needs, and so on. We aim to provide useful and actionable advice to our partners considering their long-term goals, rather than a simple list of security issues at present.

Finally, Zellic provides a list of miscellaneous observations that do not have security impact or are not directly related to the scoped modules itself. These observations — found in the Discussion (4) section of the document — may include suggestions for improving the codebase, or general recommendations, but do not necessarily convey that we suggest a code change.

2.3 Scope

The engagement involved a review of the following targets:

GotSui Modules

Repository	https://github.com/MystenLabs/gotsui
Version	gotsui: `c97a52417b6e4ff055321642bd44ac0c2f0c5e57`
Program	*.ts, *.tsx, *.js
Type	Web application
Platform	Web, mobile

2.4 Project Overview

Zellic was contracted to perform a security assessment with two consultants for a total of one person-week. The assessment was conducted over the course of one calendar week.

Contact Information

The following project manager was associated with the engagement:

Chad McDonald, Engagement Manager
chad@zellic.io

The following consultants were engaged to conduct the assessment:

SeungHyeon Kim, Engineer
seunghyeon@zellic.io

Yuhang Wu, Engineer
yuhang@zellic.io

2.5 Project Timeline

The key dates of the engagement are detailed below.

October 10, 2023	Kick-off call
October 11, 2023	Start of primary review period
October 16, 2023	End of primary review period

3 Detailed Findings

3.1 Denial of service

- **Target**: app/callback/page.tsx
- **Category**: Coding Mistakes
- **Likelihood**: High
- **Severity**: Low
- **Impact**: Low

Description

The endpoint `/callback` is intended to get session information from `window.location.hash`. The page will try to apply the given session on the local storage when it hits. However, the page contains no validation logic for a given session.

```
const callbackSession = useSessionStore((state) => state.callbackSession);
const { mutate, error } = useMutation(
  async () => {
    const params = new URLSearchParams(window.location.hash.slice(1));
    return callbackSession(params);
  },
  {
    onSuccess(data) {
      router.replace(data || "/account");
    },
  }
);
```

The result is that the pages will produce errors when a user navigates with the wrong session. The errors are related to JWT, and the log of the root cause will be like below.

```
Uncaught JWTInvalid: Invalid JWT
```

For the JWT validation, the project uses the `useDecodedJWT` function defined in `lib/session.ts`.

```
export function useDecodedJWT() {
  const jwt = useSessionStore((state) => state.jwt);
  return useMemo(() => (jwt ? jose.decodeJwt(jwt) : null), [jwt]);
```

```
}
```

As a result, the jose library will fail to decode `jwt`, and it will break page rendering due to error.

Impact

The user will be unable to use any page functions until they clear the browser local storage.

Recommendations

Check the given session before it applies on the local storage. Also, guide the user to sign in again if the page produces errors with JWT validation.

Remediation

This issue has been acknowledged by Mysten Labs Ltd., and a fix was implemented in commit 305c01e0.

4 Discussion

The purpose of this section is to document miscellaneous observations that we made during the assessment. These discussion notes are not necessarily security related and do not convey that we are suggesting a code change.

4.1 Link-based transfer needs enhanced encryption for secure token transmission

The entire process of sending tokens can be broken down into two methods:

1. **Direct transfer.** Tokens are directly sent from the current user to a specific address. This transaction is signed by the current user.
2. **Link-based transfer.** The current user sends tokens to a randomly generated new address and creates a link. This link contains the private key of the new address. The user can then forward this link to the recipient. Upon clicking "claim" on the link, the recipient signs a transaction using the private key provided in the `location.hash`, thereby transferring the tokens from the new address to their own address.

For these methods, every private key corresponds to an address (which is essentially the hash of the address) that has a length of 32 bytes. To claim the link, the recipient only needs to provide the private key; the address is automatically derived from the private key.

However, in some scenarios (for instance, when the sender wants to ensure that the link is not hijacked by attackers to maliciously steal tokens), the private key inside the link needs to be encrypted. Only when the recipient enters the correct passphrase can the tokens be extracted. This security measure can be achieved by employing different encryption techniques like AES, DES, and so forth. By doing so, as long as the sender transmits the decryption key separately to the recipient, it ensures that intermediaries would not be able to hijack the process.

4.2 Enhancing the `get_salt` interface: Implementing measures to prevent replay attacks

The current `get_salt` interface works by having the user input a JWT token provided by Google, which then returns a salt for authentication. However, this interface currently

lacks measures to prevent replay attacks.

This means if an attacker can intercept a user's JWT token and replay it to the `get_salt` interface, they can obtain the salt. With the returned salt from the `get_salt` interface, the attacker can then impersonate the user and perform a series of operations. Therefore, measures to prevent replay attacks need to be added to the `get_salt` interface.

For instance, each JWT token could be used only once or each JWT token could be valid for a limited time. In this way, even if the attacker captures a user's JWT token, they can only use it once or within a limited time frame, thus preventing a replay attack.

5 Assessment Results

At the time of our assessment, the reviewed code was not deployed to the internet.

During our assessment on the scoped GotSui modules, we discovered one finding, which was of low impact. Mysten Labs Ltd. acknowledged the finding and implemented a fix.

5.1 Disclaimer

This assessment does not provide any warranties about finding all possible issues within its scope; in other words, the evaluation results do not guarantee the absence of any subsequent issues. Zellic, of course, also cannot make guarantees about any code added to the project after the version reviewed during our assessment. Furthermore, because a single assessment can never be considered comprehensive, we always recommend multiple independent assessments paired with a bug bounty program.

For each finding, Zellic provides a recommended solution. All code samples in these recommendations are intended to convey how an issue may be resolved (i.e., the idea), but they may not be tested or functional code. These recommendations are not exhaustive, and we encourage our partners to consider them as a starting point for further discussion. We are happy to provide additional guidance and advice as needed.

Finally, the contents of this assessment report are for informational purposes only; do not construe any information in this report as legal, tax, investment, or financial advice. Nothing contained in this report constitutes a solicitation or endorsement of a project by Zellic.