Zellic

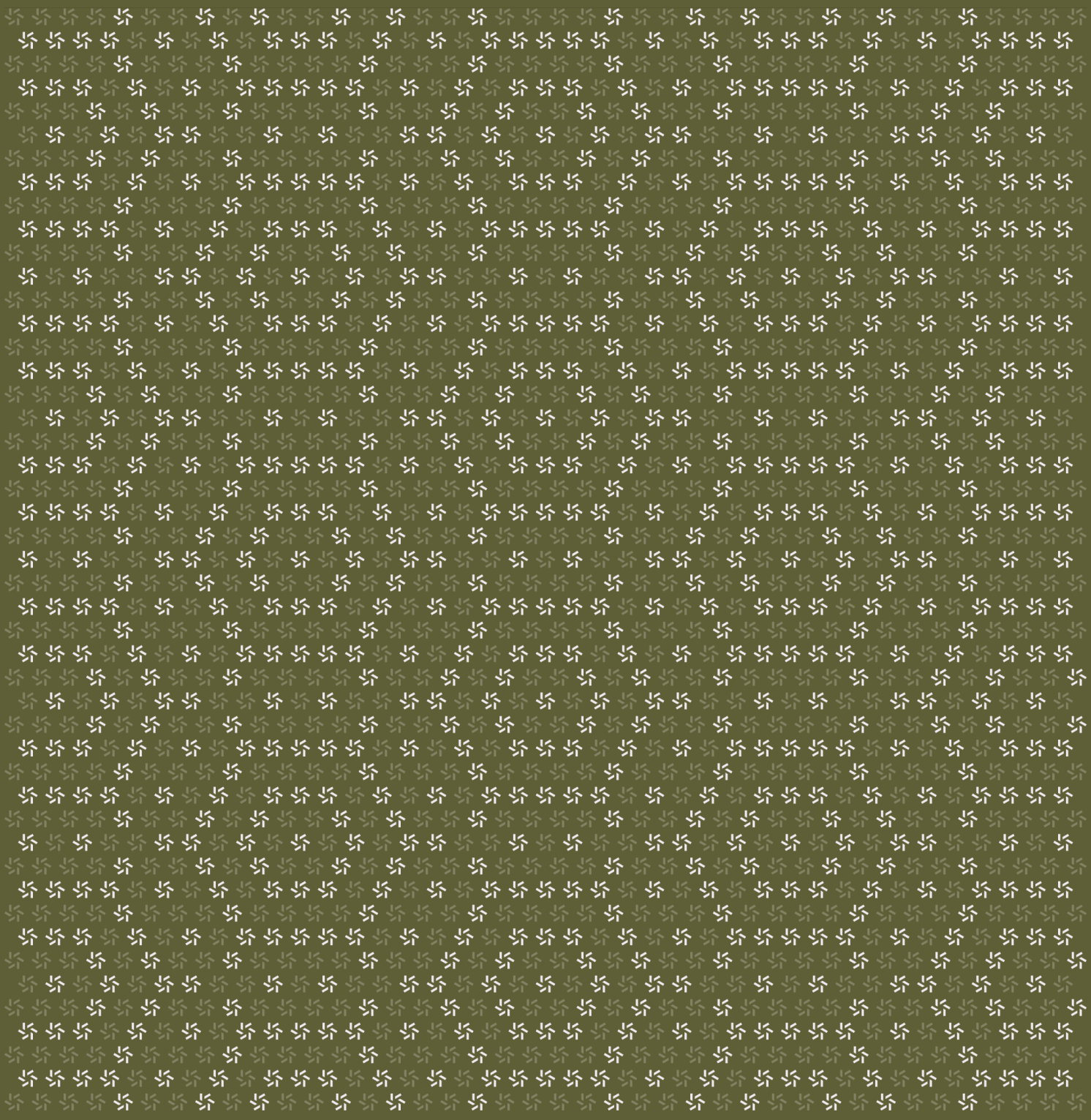
Prepared for
Edward Nygma
Sotatek

Prepared by
Seungjun Kim
Maik Robert
Zellic

February 6, 2025

Mina Token Bridge

Web Application Security Assessment



Contents

About Zellic

Zellic is a vulnerability research firm with deep expertise in blockchain security. We specialize in EVM, Move (Aptos and Sui), and Solana as well as Cairo, NEAR, and Cosmos. We review L1s and L2s, cross-chain protocols, wallets and applied cryptography, zero-knowledge circuits, web applications, and more.

Prior to Zellic, we founded the #1 CTF (competitive hacking) team ↗ worldwide in 2020, 2021, and 2023. Our engineers bring a rich set of skills and backgrounds, including cryptography, web security, mobile security, low-level exploitation, and finance. Our background in traditional information security and competitive hacking has enabled us to consistently discover hidden vulnerabilities and develop novel security research, earning us the reputation as the go-to security firm for teams whose rate of innovation outpaces the existing security landscape.

For more on Zellic's ongoing security research initiatives, check out our website zellic.io ↗ and follow @zellic_io ↗ on Twitter. If you are interested in partnering with Zellic, contact us at hello@zellic.io ↗.

1. Overview

1.1. Executive Summary

Zellic conducted a security assessment for Mina Foundation from January 17th to January 31st, 2025. During this engagement, Zellic reviewed Sotatek's development of Mina Token Bridge's code for security vulnerabilities, design issues, and general weaknesses in security posture.

1.2. 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:

- Are there any client-side security issues like cross-site scripting?
- Is the authentication for the admin backend working as intended?
- Is user input handled safely?

1.3. Non-goals and Limitations

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

- Outside infrastructure relating to the project
- On-chain components

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

During this assessment, setting up a fully functional test environment on the agreed-upon commit proved challenging. The setup documentation lacked crucial steps, which only became apparent after debugging issues where certain functions were either not called at all or executed with incorrect values. While the front-end components were relatively easy to initialize, the backend consists of multiple components, including a main application, a database, a BullMQ instance using Redis, and eight separate crawlers—each of which had to be manually started.

Due to these challenges and time constraints, only parts of the application were tested dynamically. Additionally, limited availability from the Sotatek team during most of the audit period meant a significant amount of time was spent troubleshooting issues that could have been avoided with more comprehensive documentation.

As a result, full end-to-end testing was not possible. This impacted the verification of Finding 3.1. ↗, which could only be analyzed statically, as it could not be reproduced in a test environment.

1.4. Results

During our assessment on the scoped Mina Token Bridge modules, we discovered one finding, which was of high impact.

Additionally, Zellic recorded its notes and observations from the assessment for the benefit of Sotatek in the Discussion section (4. ↗).

Breakdown of Finding Impacts

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

2. Introduction

2.1. About Mina Token Bridge

Sotatek contributed the following description of Mina Token Bridge:

> The Mina Token Bridge enables seamless, secure, and efficient asset transfers between EVM blockchain and the Mina Chain.

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:

Unsafe code patterns. Many vulnerabilities in web-based applications arise from oversights during development. Missing an authentication check on a single API endpoint can critically impact the security of the overall application. Failure to properly sanitize and encode user input can lead to vulnerabilities like SQL injection, server-side request forgery, and more. We use both automated tools and extensive manual review to identify unsafe code patterns.

Business logic errors. Business logic is the heart of all web applications. We carefully review logic to ensure that the code securely and correctly implements the specified functionality. We also thoroughly examine all specifications for inconsistencies, flaws, and vulnerabilities, including for risks of fraud and abuse.

Integration risks. Web projects frequently have many third-party dependencies and interact with services and libraries that are not under the developer's control. We review the project's external interactions and identify potentially dangerous behavior resulting from compatibility issues and data inconsistencies.

Code maturity. We look for possible improvements across the entire codebase, reviewing violations of industry best practices and code quality standards. We suggest improvements to code clarity, documentation, and testing practices.

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:

Mina Token Bridge Modules

Type	TypeScript
Platform	Web
Target	Multichain Bridge Backend
Repository	https://github.com/sotatek-dev/multichain-bridge-backend ↗
Version	`919bb3bc214df45bf0a225395765886484647d0f`
Programs	`*.ts` `*.tsx`
Target	Mina Bridge FE User
Repository	https://github.com/sotatek-dev/mina-bridge-fe-user ↗
Version	`5a136ffe89975ead3782f4e6399592a962f027df`
Programs	`*.ts` `*.tsx`

Target	Mina Bridge FE Admin
Repository	https://github.com/sotatek-dev/mina-bridge-fe-admin ↗
Version	`7e23b9872fa4c30ca680adc4d8fbd6ae16f57b2c`
Programs	`*.ts` `*.tsx`

2.4. Project Overview

Zellic was contracted to perform a security assessment for a total of three person-weeks. The assessment was conducted by two consultants over the course of two calendar weeks.

Contact Information

The following project managers were associated with the engagement:

Jacob Goreski
Engagement Manager
jacob@zellic.io ↗

Chad McDonald
Engagement Manager
chad@zellic.io ↗

The following consultants were engaged to conduct the assessment:

Seungjun Kim
Engineer
seungjun@zellic.io ↗

Maik Robert
Engineer
maik@zellic.io ↗

2.5. Project Timeline

The key dates of the engagement are detailed below.

January 17, 2025	Start of primary review period
January 31, 2025	End of primary review period

3. Detailed Findings

3.1. Bypassing daily quota may lead to stuck funds

Target	src/modules/crawler/job-unlock.provider.ts		
Category	Coding Mistakes	**Severity**	High
Likelihood	Medium	**Impact**	High

Description

The bridge has an adjustable daily quota of ETH that can be bridged. On the front end, this check is done on the client side to prevent users from entering a number that would be over the remaining daily quota. Once a bridge-deposit transaction has made it through the various crawlers, a check is done on the backend if the current bridging transaction would bring the total sum of all transactions by that user above the daily threshold.

```
private async isPassDailyQuota(address: string, networkReceived:
    ENetworkName): Promise<boolean> {
const fromDecimal = this.configService.get(
    networkReceived === ENetworkName.MINA ? EEnvKey.DECIMAL_TOKEN_EVM :
    EEnvKey.DECIMAL_TOKEN_MINA,
);
const [dailyQuota, todayData] = await Promise.all([
    await this.commonConfigRepository.getCommonConfig(),
    await this.eventLogRepository.sumAmountBridgeOfUserInDay(address),
]);
assert(!!dailyQuota, 'daily quota undefined');
this.logger.info(\`totals: ${todayData.totalamount}
    and quota ${dailyQuota.dailyQuota}\`);
if (
    todayData?.totalamount &&
    BigNumber(todayData.totalamount).isGreaterThan(addDecimal(dailyQuota.
    dailyQuota, fromDecimal))
) {
    return true;
}
return false;
}
```

See that `isPassDailyQuota()` is called in the `handleSendTxJob()` function, which is responsible for the sum-of-transactions check.

```
private async handleSendTxJobs(data: IJobUnlockPayload) {
// check if there is enough threshhold -> then create an unlock job.
if (await this.isPassDailyQuota(data.senderAddress, data.network)) {
    this.logger.warn('this tx exceed daily quota, skip until next day',
    data.eventLogId);
    await this.eventLogRepository.update(data.eventLogId, { nextSendTxJobTime:
    getNextDayInUnix().toString() });
    return;
}
await this.queueService.addJobToQueue<IUnlockToken>(
    this.getSenderQueueName(data.network),
    {
    eventLogId: data.eventLogId,
    },
    {
    attempts: 5,
    removeOnComplete: {
        age: this.jobRemoveDueDate,
    },
    backoff: this.sendTxJobBackOff,
    },
);
}
```

If the transactions pass the daily quota, they are moved to be processed the next day. A user may decide to bypass the client-side checks and send multiple transactions or a single large transaction that breaches the daily quota. Each would then be moved to be processed the following day, where the same check would be done, again passing the daily quota and being moved to the next day yet again. As this constraint can never be satisfied, since the transaction will always be bigger than the daily quota, the funds are stuck until admin intervention. We were unable to verify this finding due to the lack of a fully working test environment.

Impact

A user who decides to bypass the limits enforced by the UI may have their funds stuck in the bridging contract if they send an amount that is above the daily quota.

Recommendations

Ensure that user funds cannot become stuck if they manage to send transactions that are above the daily quota. Either refund the deposit, send the entire amount on the following day, or bridge tokens up to the daily quota and send the rest of the amount the following day.

Remediation

The team has acknowledged the finding and responded with the following comment:

> For MINA, currently we don't apply any daily quota checking mechanism since MINA has limited data structures to work with. For ETH, we skip daily quota checks to reduce gas usage, because ETH price is expensive. For the reason why we move over daily quoted transactions to the next day. Admin should set the max value of each bridge transaction to be smaller than the daily quota, then there will be no transactions pending forever due to its value exceeding the daily quota. In the next version, due to the limitations of variables and storage of Mina network, we will need to move this logic to BE. This issue will not happen even though the admin can configure it differently than we expected above.

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. Lack of documentation

Well-maintained documentation is invaluable for both auditors and new developers when getting familiar with a codebase. A lack of documentation makes it harder to read, understand, and extend the code, while outdated documentation can be just as problematic—creating confusion when it no longer reflects how the code actually works. We found that this project lacked adequate documentation, particularly regarding deployment and setup.

The repository did include flow diagrams, in the form of PUML files, which were helpful in understanding how a bridging transaction should work. However, the documentation for deploying and running the codebase appeared to be significantly outdated. The automated pm2 start script was missing critical steps required for the application to function properly, and these steps were also absent from the repository's "Getting Started" section. While `docker-compose.dev.yaml` included some of the missing steps, it was never referenced in the documentation. As a result, troubleshooting the deployment took considerably more time than necessary—time that could have been saved with up-to-date documentation outlining all the necessary steps to get the application running.

Improving and maintaining clear, up-to-date documentation would greatly benefit both auditors and developers working with the codebase.

4.2. Insufficient Test Coverage

While some test cases were present—such as verifying that an admin user could log in—overall test coverage was lacking. There were no negative test cases to confirm that non-admin users would be correctly rejected, and many critical parts of the application had no tests at all.

Robust test coverage is essential for catching common bugs, ensuring that a function or flow behaves as expected, and preventing regressions during active development. Expanding test coverage, particularly for core components, would improve the reliability and maintainability of the codebase.

5. System Design

This provides a description of the high-level components of the system and how they interact, including details like a function's externally controllable inputs and how an attacker could leverage each input to cause harm or which invariants or constraints of the system are critical and must always be upheld.

Not all components in the audit scope may have been modeled. The absence of a component in this section does not necessarily suggest that it is safe.

5.1. Component: Admin front end

Description

The admin front end is responsible for changing various configuration options for the bridge. An admin is able to log in with their wallet, which lets them edit options like the daily quota limit as well as view the bridging history of all users.

Invariants

- An unauthorized user should not be able to change options on the bridge via the admin API endpoints.
- A non-admin user should not be able to log in and obtain an admin JSON Web Token (JWT).

Test coverage

Cases covered

- Log in with an admin Mina wallet.
- Log in with an admin Ethereum wallet.

Cases not covered

- Log in with an invalid admin wallet.
- Call admin API endpoints with an admin JWT, no JWT, or forged JWT.

Attack surface

The attack surface itself is limited in the log-in and config-update API endpoints. Admin wallets are in the environment file, and users trying to log in are compared to the values from the environment file; if they are not a match, the user is unable to log in. Admin API endpoints check that a valid admin JWT is present, making sure that no regular users are able to interact with them.

5.2. Component: User front end

Description

The user front end is responsible for handling the bridging operations. A user connects to the service with their MetaMask or Auro Wallet, after which they are able to deposit ETH or WETH to be bridged to the opposite chain.

Invariants

- A user should not be able to bypass the daily quota.

Test coverage

Cases covered

- No test coverage.

Cases not covered

- No test coverage.

Attack surface

The attack surface itself is extremely limited, with the application only having two inputs and a button. As far as the UI goes, there are no real issues that could be exploited strictly on the client side. One issue was found regarding the client-side daily quota check; this can be trivially bypassed to make it possible to enter ETH values that are above the daily limit. A user is then able to deposit an amount that exceeds the daily quota (see Finding 3.1. ↗), which leads to that transaction being stuck as the backend does not handle this edge case sufficiently.

5.3. Component: Multichain bridge backend

Description

The multichain bridge backend is the application that handles all operations for the admin and user front ends. Aside from handling the input coming in from the two front ends, it is also responsible for the bridging logic. Various crawlers are set up to retrieve the on-chain state, database entries, and job queues to bridge the user's ETH to WETH or WETH to ETH.

Invariants

- A user should not be able to bypass the daily limit of tokens to bridge.
- User-controlled data should be handled securely.
- The admin API must enforce strict authentication to ensure that only authorized users can access its functionalities.
- Crawlers should function properly.

Test coverage

While some limited test coverage is present, not everything is covered by a test case.

Attack surface

The attack surface is mainly code that directly handles user-controlled input — for example, the admin API, which is secured by requiring a valid JWT, which is only obtained by logging in with a valid admin wallet. The user front-end output is extremely limited, and most values are not directly controlled by the user.

Nonetheless, some of the controllable values are inserted into a database, which may lead to SQL injection if the input is handled improperly. No cases of insecure data handling relating to SQL operations were found.

The check dealing with the daily quota happens after a user has already made the deposit. It is coded in such a way that funds may get stuck, if a user makes a deposit that exceeds the currently configured daily quota, which has been detailed in Finding 3.1. ↗.

The crawler setup was mainly investigated via static analysis and some limited dynamic testing. Due to time constraints, it was not possible to get a fully working setup using the commit agreed upon for the audit.

6. Assessment Results

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

During our assessment on the scoped Mina Token Bridge modules, we discovered one finding, which was of high impact.

6.1. 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.