Zellic

Prepared for
dexCEO
DexFi

Prepared by
Bryce Casaje
Michael Guarino
Zellic

May 19, 2025

DexFi

OpSec 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 DexFi from April 28th to May 2nd, 2025. During this engagement, Zellic reviewed DexFi's operations for security vulnerabilities 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:

- Does DexFi perform their standard operations in as secure of a way as possible?
- What would be the impact if a single employee account was breached?
- Do DexFi employees who are most likely to be targeted by an attacker practice good personal operational security?
- Do DexFi developers follow all other general best operational security practices?

1.3. Non-goals and Limitations

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

- Any application-level security components

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

Although access to the code was given, the scope of the assessment did not include review of business logic, design, or dependencies (beyond deployment configuration and CI/CD)

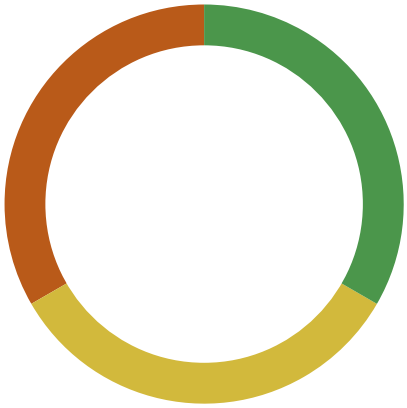
1.4. Results

During our assessment on the scoped DexFi files, we discovered three findings. No critical issues were found. One finding was of high impact, one was of medium impact, and one was of low impact.

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

Breakdown of Finding Impacts

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



2. Introduction

2.1. About DexFi

DexFi contributed the following description of DexFi:

> DexFi offers an ecosystem of financial products designed to empower users and simplify the DeFi experience.

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 files 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:

DexFi Files

Type	N/A
Platform	Web
Target	infrastructure
Repository	https://github.com/dexfinance-com/infrastructure ↗
Version	`e838cf54961ab8107e873a710b78e5d063fea409`
Programs	*
Target	GitHub organization and associated repositories
Version	`Configuration at the time of assessment`

2.4. Project Overview

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

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:

Bryce Casaje
Engineer
bryce@zellic.io

Michael Guarino
Engineer
michael@zellic.io

2.5. Project Timeline

The key dates of the engagement are detailed below.

April 28, 2025	Kick-off call
April 28, 2025	Start of primary review period
May 2, 2025	End of primary review period

3. Detailed Findings

3.1. Excessive GitHub organization permissions

Target	GitHub Organization		
Category	Coding Mistakes	**Severity**	High
Likelihood	Medium	**Impact**	High

Description

Several members of DexFi's GitHub organization have the "owner" role. Of 10 overall members, 5 are granted ownership. Organization owners have full administrative access to the organization. This level of access is overly broad and excessive, as most organization members only require a smaller breadth and depth of permissions to function. Granting members ownership should be reserved for true owners who require full access, or for ensuring ownership continuity by adding an additional owner if there is only one as recommended by GitHub.

Impact

Members with ownership over the organization can control any administrative settings, which diminishes the value of any properly configured security setting. In addition to an employee with ownership using ownership permissions to non-maliciously bypass operational security practices, the breach of an employee's account or an insider threat incident could have severe consequences, including modifying/deleting code or accessing secrets which would allow further damage across additional services.

Recommendations

The principle of least privilege should be followed when configuring organization access. If a user does not require certain permissions to complete their designated tasks, then they should not be given those permissions. If user's do need organization ownership to complete their tasks, then other authorization recalibrations should be made so that such broad permissions are no longer required. All users should be assessed to determine which permissions are necessary and have their roles updated to only include those that are necessary.

Remediation

DexFi has acknowledged this issue and stated the following:

All members of the DexFi GitHub organization have had the "owner" role removed, except for the DexFi admin account. Furthermore, all GitHub users in the organization are now required to have 2FA (two-factor authentication) enabled, in accordance with the findings from the security audit.

3.2. Secrets in source code

Target	GitHub Repositories		
Category	Coding Mistakes	**Severity**	Medium
Likelihood	Medium	**Impact**	Medium

Description

Several repos across DexFi's GitHub organization were found to hard code secrets within source code. Secrets range from API keys with minimal impact, to Slack webhooks, to GitHub access tokens with complete organization permissions. Active secrets are located in current code as well as within commit history. In addition to the issue of being hardcoded within source code, care should be taken to ensure that any authentication tokens with configurable scope are only given the minimal permissions required for their function.

The `.npmrc` file in the `vaults-v3-backend` repo history contains active GitHub tokens, one of which has excessive permissions within it's commit history.

API keys and Slack webhooks are also found within the same repo, in the `config/*.json` files.

Impact

Any disclosure of source code to an attacker could result in losses wherever visible secrets are used. Excessively scoped tokens could result in far greater loss than if tokens were otherwise minimally scoped. For example, an attacker that manages read-only access to the affected repository code through any method could then locate an overly scoped GitHub token and then make use of their elevated privileges for further attacks. Similar situations could apply for other hardcoded secrets. For example, allowing an attacker Slack webhook access or access to other third-party APIs that are in use.

Recommendations

All sensitive secrets should be stored securely rather than hardcoded within the source, even if the code is not intended to be public. Created tokens should only include the minimally required permissions for function.

Any currently disclosed secrets should be rotated so that they can not be abused if an attacker finds them in the future. Measures should be implemented to ensure that any secrets are not stored in source code in the future, which can be accomplished by implementing GitHub's automated code scanning or by implementing tooling within automated pipelines. All repos across the organization should be examined to determine where secrets are improperly stored and have

any occurrences removed.

Remediation

DexFi has acknowledged this issue and stated the following:

> All repos across the organization have been examined to determine where secrets were improperly stored to have those occurrences removed. The development team has been notified to migrate the semi-secret token inside the [vaults-v3-backend] repo into Github secrets. Furthermore, we are coordinating with our colleagues to rewrite the repository history to remove all sensitive data. Of which all the remaining tasks will be completed before any private repos are marked public.

3.3. Pull requests merged without code reviews

Target	GitHub Repositories		
Category	Coding Mistakes	**Severity**	Low
Likelihood	Low	**Impact**	Low

Description

Important repositories within DexFi's organization do not require code reviews to merge pull requests. For example, the `vaults-v3-backend` repo contains many merged pull requests without any reviews at all. Code reviews for pull requests will help stop any problematic code from reaching the main branch.

Impact

The lack of code reviews prior to merging pull requests can result in vulnerable code being pushed to the main branch of the repository. Additionally, lack of code review could result in an attacker using a breached account to immediately push their own code without any barriers.

Recommendations

Mandatory code review should be enabled for all important code repositories with potential for negative impact as a result of undesired code being pushed. In addition to this process, it would be beneficial for old pull requests which were never reviewed to be retroactively reviewed to ensure that no undesirable code was merged into the main branches due to the lack of prior review.

Remediation

DexFi has acknowledged this issue and stated the following:

> Mandatory code reviews have been enabled for the important repositories. Individual contributer's pull requests now require unique branches, that each must pass successful checks, before requiring the approval of the repository's manager before being merged into production.

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 security automation

The GitHub organization appears to be lacking in security-specific automation tasks across code repositories. While not an immediate threat, DexFi may want to consider implementing static code analysis tools at a minimum for any code that is pushed. Due to the current nature of hardcoded secrets, implementing this form of automation will likely have a strong benefit in preventing further occurrences in the future. In addition to stopping hardcoded secrets, security automated scanning can also help identify any code-level vulnerabilities which may have otherwise not been caught by code review.

4.2. Enforce single sign on

DexFi could benefit from enforcing single sign on for users across all applications. This measure will help enforce stronger password practices and greatly reduce the attack surface for potential account theft. Solutions like Google Workspace, Okta, or a self-hosted setup can work well.

4.3. Enable audit logging

It's important to have audit logging of all security-relevant actions across the entire organization. This can be used to build alerting tools or to aid in investigation in the event of a breach. Enabling and utilizing logging services within your cloud provider will help catch incidents early or understand what went wrong and what was done if not immediately caught.

5. OpSec Overview

This section summarizes Zellic's operational security assessment of DexFi. We observed and inquired about security practices relating to DexFi's development workflow, web3 security practices and personal security hygiene.

The following subsections detail specific aspects of operational security, what actions or measures are taken to remain secure, and how effectively risk is mitigated.

5.1. Development OpSec

Credentials in source code

Git repositories were reviewed to determine if any sensitive credentials were stored in source code, and if so, what the impact of any leaked credentials could be.

It was found during assessment that several sensitive secrets were stored in source code within Git repositories.

Cloud secret management

Credentials can be stored in secure cloud secret management which will prevent much of the issues that are caused by storing credentials in source code, as well as provide additional features like centralized and fine-grained access control, extensive audit logging, automatic secret rotation, and hardware-backed secure storage.

It was found during assessment that secure cloud secret management was in use.

Continuous Deployment

Continuous deployment can be used to enforce checks before anything hits production. Systems like GitHub actions can be used to run automated unit tests to ensure that all tests still pass. Security-based automation can also be implemented, for example to implement static code analysis scanning to help catch any security vulnerabilities that may have been implemented in the code.

It was found during assessment that continuous deployment is implemented across important repositories, but additional effort could be made towards security automation as well.

Code Review

Before any code is deployed, it must be reviewed by at least one other person. Doing so will help prevent any undesirable code from reaching production. Systems in GitHub or GitLab work well for code review, while third-party code review assessments are also invaluable to securing code.

It was found during assessment that code review is required within many code repositories, but not all.

Single sign on

Enforcing single sign on for organization users can aid in password security. The attack surface shrinks from several applications and services to a single password manager, making it much easier to manage.

At the time of our review, SSO was not enabled as mentioned in Discussion Point 4.2. ↗.

5.2. Personal OpSec

To aid in personal operational security assessment, key DexFi contacts were asked about individual habits to ensure they are following best practices. Of the following areas, DexFi contacts confirmed that all were appropriately followed.

Personal Security Hygiene

DexFi contacts were asked about habits including keeping software up to date and not clicking on random links. Performing either of these actions can result in breaches.

Password Managers and Multi-Factor Authentication

DexFi contacts were asked about behavior regarding password manager use, including which types and regarding configuration. Using secure password managers with hardware-based authentication devices or TOTP are considered best practices for keeping secure.

Email

DexFi contacts were asked to confirm they were using secure, reputable email providers and also to ensure that their current mailboxes were not unknowingly being forwarded to unknown addresses. Additionally, enablement of multi-factor authentication on email accounts was also undergone by DexFi contacts.

Social Media Accounts

DexFi contacts were asked about configuration and habits around social media accounts or platforms such as Twitter, Telegram, or Discord. Care was taken to ensure that accounts met recommended security configurations, including secure multi-factor authentication, separation of work and personal accounts, and other platform-specific practices.

5.3. Web3 OpSec

Crypto

In addition to development and personal operational security, some attention was also paid to DexFi's Web3 operational security. Particularly, DexFi contacts were asked about their multisig setup, key backups, third-party custody, and wallets (hardware, mobile, or web). Ensuring that these aspects are following best practices helps greatly reduce risk.

6. Assessment Results

During our assessment on the scoped DexFi files, we discovered three findings. No critical issues were found. One finding was of high impact, one was of medium impact, and one was of low impact.

At the time of our assessment, the reviewed operational security practices were in a moderate state with room for some key improvements. Practices around GitHub and pushing code to repositories stood out as the main area's for improvement.

The current combination of issues surrounding GitHub organization and repository security result in risk of increased impact in the event of a breach. Adjusting permissions and implementing some best practice defense-in-depth measures will help to greatly reduce impact in the event of a breached organization member account. Other changes like enforcing code review or adding automated security scanning will result in less chances for risky code to exist in production, which would either result in direct application-level vulnerabilities, or manifest as leverage points for attackers to pivot to different services or accounts via leaked credentials.

Outside of strict GitHub repository and organization security, some adjustments could be made to the cloud or organization-wide configurations such as implementing SSO or audit logging.

Overall, once the identified issues are resolved, DexFi should feel comfortable with their current operational security of what was assessed. Issues identified via this assessment should be kept in mind in the future to ensure that none are reintroduced.

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.