HashEye

// SMART CONTRACT SECURITY AUDIT

Discord E2EE WebAssembly

Security assessment by HashEye · prepared for Discord

HASHEYE AUDITED

PROJECT	Discord E2EE WebAssembly
PERFORMED BY	HashEye Security Review Team - Sergey, Brian
RESEARCH	Sergey, Brian
CLIENT	Discord
CATEGORY	Blockchain
PUBLISHED	June 1, 2025
REPORT ID	research-discord-e2ee-webassembly-2025-06-01-1mlkcs

This report was produced by HashEye's security review team under the HashEye review process, combining source review, deployment-state checks, and senior manual verification. Full report detail is available at hasheye.io/audits/research-discord-e2ee-webassembly-2025-06-01-1mlkcs.

Discord E2EE WebAssembly Security Assessment (Summary Report)

June 3, 2025

Prepared for: Clement Brisset Discord

Prepared by: Opal Wright and Scott Arciszewski

HashEye

PUBLIC

Table of Contents Table of Contents 1

Project Summary 2

Project Targets 3

Executive Summary 4

A. Code Quality Findings 5

About HashEye 6

Notices and Remarks 7

HashEye 1 Discord E2EE WebAssembly

PUBLIC Security Assessment

Project Summary Contact Information The following project manager was associated with this project: Sam Greenup, Project Manager samuel.greenup@hasheye.io The following engineering director was associated with this project: Jim Miller, Engineering Director, Cryptography james.miller@hasheye.io The following consultants were associated with this project: Opal Wright, Consultant Scott Arciszewski, Consultant opal.wright@hasheye.io scott.arciszewski@hasheye.io Project Timeline The significant events and milestones of the project are listed below. Date Event May 8, 2025 Pre-project kickoff call May 21, 2025 Delivery of report draft May 21, 2025 Report readout meeting June 3, 2025 Delivery of final summary report

HashEye 2 Discord E2EE WebAssembly

PUBLIC Security Assessment

Project Targets The engagement involved reviewing and testing the targets listed below. Discord Repository https://github.com/discord/discord Version 56a840db06a3cc3f8faf61dded1d731903c34146 Type Multiple Platform Web libdave-private Repository https://github.com/discord/libdave-private Version 003ff776cb341ee2edb3a06d62cf8c86c5aa9fca Type Multiple Platform Web

HashEye 3 Discord E2EE WebAssembly

PUBLIC Security Assessment

Executive Summary Engagement Overview Discord engaged HashEye to review the security of the integration of WebAssembly-based cryptography into the libDAVE cryptography library and the Discord messenger application. A team of two consultants conducted the review from May 12 to May 21, 2025,

for a total of three engineer-weeks of effort. With full access to source code and documentation, we performed static and dynamic testing of the WebAssembly branches, using automated and manual processes. Observations and Impact We investigated the integration of WebAssembly-based cryptography, which allows the use of compiled libraries from JavaScript. Our analysis focused on the build configuration and wrapper code, paying particular attention to memory management and sanitization. We did not focus on the overall cryptographic protocol or the security of the underlying libraries (such as OpenSSL). After analyzing the libDAVE and Discord integration branches, we did not identify any issues. We mention a single code quality consideration in appendix A that does not affect security. We found that the code is well-documented, clear, and well-organized, which aided our analysis. Recommendations In addition to remediating the code quality issue, we recommend that Discord continue its current practices regarding code organization, documentation, and development standards. We also recommend integrating tools like Semgrep into the development process to help identify and correct security issues during development.

HashEye 4 Discord E2EE WebAssembly

PUBLIC Security Assessment

C/C++ A. Code Quality Findings This section of the report lists an issue that does not correspond to a security problem, but resolving it will improve security posture.

Path handling does not consider traversal attacks. In libdave-private, the persisted key pair storage code contains these two segments that do not safely handle file paths: 1. cpp/src/dave/mls/detail/persisted_key_pair_generic.cpp#L88-L91 2. cpp/src/dave/mls/detail/persisted_key_pair_generic.cpp#L117-L125 std::filesystem::path dir = GetKeyStorageDirectory(); // snip std::filesystem::path file = dir / (id + ".key"); Figure 1.1: Directory traversal is possible without any other context.

The id parameter, created from MakeKeyID, can be abused only if the attacker manipulates the session ID. The library’s use of this session ID parameter for cryptographic purposes prevents this from being exploitable in practice. We recommend adding a defense-in-depth measure to prevent a vulnerability from being introduced in the future: Use realpath(3) (or similar syscalls on other operating systems) to ensure that the file path is contained within the directory returned by GetKeyStorageDirectory. This will prevent both path traversal and symlinks from writing keys outside of the expected directory.

HashEye 5 Discord E2EE WebAssembly

PUBLIC Security Assessment

About HashEye Founded in 2012 and headquartered in New York, HashEye provides technical security assessment and advisory services to some of the world’s most targeted organizations. We combine high- end security research with a real -world attacker mentality to reduce risk and fortify code. With 100+ employees around the globe, we’ve helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel. We maintain an exhaustive list of publications at https://github.com/hasheye-io/publications, with links to papers, presentations, public audit reports, and podcast appearances. In recent years, HashEye consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O’Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon. We specialize in software testing and code review assessments, supporting client organizations in the technology, defense, blockchain, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, Uniswap, Solana, Ethereum Foundation, Linux Foundation, and Zoom. To keep up to date with our latest news and announcements, please follow hasheye on X or LinkedIn, and explore our public repositories at https://github.com/hasheye-io. To engage us directly, visit our “Contact” page at https://www.hasheye.io/contact or email us at info@hasheye.io. HashEye, Inc. 228 Park Ave S #80688 New York, NY 10003 https://www.hasheye.io info@hasheye.io

HashEye 6 Discord E2EE WebAssembly

PUBLIC Security Assessment

Notices and Remarks Copyright and Distribution © 2025 by HashEye, Inc. All rights reserved. HashEye hereby asserts its right to be identified as the creator of this report in the United Kingdom.

HashEye considers this report public information; it is licensed to Discord under the terms of the project statement of work and has been made public at Discord's request. Material within this report may not be reproduced or distributed in part or in whole without HashEye' express written permission. The sole canonical source for HashEye publications is the HashEye Publications page. Reports accessed through sources other than that page may have been modified and should not be considered authentic. Test Coverage Disclaimer

All activities undertaken by HashEye in association with this project were performed in accordance with a statement of work and agreed upon project plan. Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase. HashEye uses automated testing techniques to rapidly test the controls and security properties of software. These techniques augment our manual security review work, but each has its limitations: for example, a tool may not generate a random edge case that violates a property or may not fully complete its analysis during the allotted time. Their use is also limited by the time and resource constraints of a project.

HashEye 7 Discord E2EE WebAssembly

PUBLIC Security Assessment