HashEye

// SMART CONTRACT SECURITY AUDIT

EthStaker Deposit CLI

Security assessment by HashEye · prepared for EthStaker

HASHEYE AUDITED

PROJECT	EthStaker Deposit CLI
CLIENT	EthStaker
CATEGORY	Blockchain
PUBLISHED	March 1, 2026
REPORT ID	research-ethstaker-deposit-cli-2026-03-01-11vc37

This report was produced under HashEye's layered review process — **automated detection**, **pattern correlation**, and **senior manual verification** — with every finding signed off by a human reviewer. Full findings detail and on-chain attestation are available on the report page at hasheye.io/audits/research-ethstaker-deposit-cli-2026-03-01-11vc37.

EthStaker Deposit CLI Security Assessment March 4, 2026

Prepared for: Rémy Roy EthStaker

Prepared by: Bo Henderson

HashEye

PUBLIC

Table of Contents Table of Contents 1 Project Summary 2 Executive Summary 3 Project Goals 5 Project Targets 6 Project Coverage 7 Automated Testing 8 Summary of Findings 10 Detailed Findings 11 1. Fork version collision is possible, allowing devnet deposits to be replayed on mainnet 11 2. Sensitive files on Windows can be overwritten 12 A. Vulnerability Categories 13 B. Code Quality Recommendations 15 C. Fix Review Results 16 Detailed Fix Review Results 16 About HashEye 17 Notices and Remarks 18

HashEye 1 EthStaker Deposit CLI

PUBLIC Security Assessment

Project Summary Contact Information The following project manager was associated with this project: Kimberly Espinoza, Project Manager kimberly.espinoza@hasheye.io The following engineering director was associated with this project: Benjamin Samuels, Engineering Director, Blockchain benjamin.samuels@hasheye.io The following consultant was associated with this project: Bo Henderson, Consultant bo.henderson@hasheye.io Project Timeline The significant events and milestones of the project are listed below. Date Event January 7, 2026 Pre-project kickoff call January 20, 2026 Delivery of report draft January 20, 2026 Report readout meeting January 22, 2026 Delivery of final comprehensive report February 18, 2026 Completion of fix review March 4, 2026 Delivery of updated report with fix review appendix

HashEye 2 EthStaker Deposit CLI

PUBLIC Security Assessment

Executive Summary Engagement Overview EthStaker engaged HashEye to review the security of its deposit command-line tool (deposit-cli). This tool creates and manages files required to stake as a validator on the Ethereum network. It is a fork of the staking-deposit-cli tool developed by the Ethereum Foundation with additional functionality and improved performance. One consultant conducted the review from January 12 to January 16, 2026, for a total of one engineer-week of effort. Our testing efforts focused on the safety and security of mnemonic and private key data as well as the overall security of the project's development and release practices. With full access to the source code and documentation, we performed static and dynamic testing of the target codebase using both automated and manual processes. Observations and Impact Our review of the EthStaker deposit-cli tool uncovered only minor issues that would be difficult to exploit and have minimal impact. Most notably, defense-in-depth measures while writing sensitive files on Windows machines are lacking, increasing the risk that such files could be overwritten and lost (TOB-ETHSTAKER2-2). This codebase adheres to best practices, and no issues were identified in the changes made to support features introduced by the Pectra hard fork. Recommendations Based on the findings identified during the security review, HashEye recommends that EthStaker take the following steps: • Remediate the findings disclosed in this report. These findings should be addressed through direct fixes or broader refactoring efforts. • Reconsider the tool's symlink support. The deposit-cli tool silently follows all symlinks. This may cause files to be saved to unexpected locations or give attackers the opportunity to take malicious actions, such as redirecting sensitive files to locations where they have greater privileges. However, symlinks have legitimate use cases as well. Consider using the os.path.islink method to detect whether an output directory is a symlink and print a warning, such as the one below.

WARNING: Output directory is a symbolic link pointing to: /other/location Only proceed if you intentionally created this symlink. Continue? [y/N]:

HashEye 3 EthStaker Deposit CLI

PUBLIC Security Assessment

Finding Severities and Categories

The following tables provide the number of findings by severity and category. EXPOSURE ANALYSIS Severity Count High 0 Medium 0 Low 1 Informational 1 Undetermined 0

CATEGORY BREAKDOWN Category Count Data Validation 2

HashEye 4 EthStaker Deposit CLI

PUBLIC Security Assessment

Project Goals The engagement was scoped to provide a security assessment of the EthStaker deposit-cli tool. Specifically, we sought to answer the following non-exhaustive list of questions: • Are there any incorrect error-prone steps that could cause an honest user to leak or lose their mnemonic, password, or other sensitive data? • Is the CI and overall development process adequately protected against supply-chain risks? • Could an attacker subvert filesystem path searches in a way that could lead to a crash or the use of an incorrect translation file? • Is the right cryptography used, and is it used correctly?

HashEye 5 EthStaker Deposit CLI

PUBLIC Security Assessment

Project Targets The engagement involved reviewing and testing the following target. ethstaker-deposit-cli Repository https://github.com/ethstaker/ethstaker-deposit-cli Version 611aaeaed9e0c555ab26d0c085b6fb6b995f2063 Type Python Platform Windows, macOS, Linux, Docker

HashEye 6 EthStaker Deposit CLI

PUBLIC Security Assessment

Project Coverage This section provides an overview of the analysis coverage of the review, as determined by our high-level engagement goals. Our approaches included the following: • A comprehensive automated and manual review of the Python source code located in the ethstaker_deposit directory. • Manual testing of commands implemented by the deposit-cli tool. • A review of the build and release processes, with a focus on supply chain risks.

HashEye 7 EthStaker Deposit CLI

PUBLIC Security Assessment

Automated Testing HashEye uses automated techniques to extensively test the security properties of software. We use both open-source static analysis and fuzzing utilities, along with tools developed in-house, to perform automated testing of source code and compiled software. Test Harness Configuration We used the following tools in the automated testing phase of this project: Tool Description Policy Actionlint A static checker for GitHub Actions workflow files N/A Bandit A static analysis tool designed to find common security issues in Python code N/A Semgrep An open-source static analysis tool for finding bugs and enforcing code standards when editing or committing code and during build time N/A CodeQL A code analysis engine developed by GitHub to automate security checks N/A HashEye Claude Code skills Claude Code skills for security research, vulnerability detection, and audit workflows N/A Methodology We utilized Claude Code skills developed in-house to run tools and perform static analysis: • The audit-context-building skill was used to help Claude Code research the codebase so that it could more effectively analyze the code and answer auditor follow-up questions. • The differential-review skill was used in two passes to assess changes between the previously audited commit (66054f57) and the current target commit. ○ The first pass was general. Major changes were highlighted, areas of concern were identified, and

the resulting report was manually reviewed. ○ The second pass was focused exclusively on changes made to support new features introduced by the Pectra hard fork.

HashEye 8 EthStaker Deposit CLI

PUBLIC Security Assessment

• The static-analysis skill guided Claude in using Semgrep and CodeQL to perform a comprehensive tool-backed review of the codebase.

HashEye 9 EthStaker Deposit CLI

PUBLIC Security Assessment

Summary of Findings The table below summarizes the findings of the review, including details on type and severity. ID Title Type Severity 1 Fork version collision is possible, allowing devnet deposits to be replayed on mainnet Data Validation Informational 2 Sensitive files on Windows can be overwritten Data Validation Low

HashEye 10 EthStaker Deposit CLI

PUBLIC Security Assessment

Detailed Findings 1. Fork version collision is possible, allowing devnet deposits to be replayed on mainnet Severity: Informational Difficulty: High Type: Data Validation Finding ID: TOB-ETHSTAKER2-1 Target: ethstaker_deposit/utils/validation.py

Description Users can specify custom devnet configurations via the devnet_chain_setting parameter, which accepts JSON that specifies arbitrary fork versions. The validate_devnet_chain_setting function validates the structure of this JSON input, but does not verify that the provided fork version is unique or check it against known production chain fork versions. While the genesis_validators_root is used to provide cross-chain replay protection for most validator operations, a zero-bytes value is always used for deposits. Ethereum specifications explicitly note that domain separation is relaxed while depositing to allow deposits to be valid across hard-fork upgrades. Furthermore, specifications recommend using a different GENESIS_FORK_VERSION value on testnets. Exploit Scenario In preparation for a mainnet deposit, Bob configures a devnet with the mainnet fork version 0x00000000 and a network name of “mydevnet”. He generates deposit files for testing, but accidentally submits them to mainnet. They are accepted and processed, causing Bob to prematurely submit his mainnet deposit. Recommendations Short term, maintain a list of known production chain fork versions and prohibit devnets from using them.

HashEye 11 EthStaker Deposit CLI

PUBLIC Security Assessment

2. Sensitive files on Windows can be overwritten Severity: Low Difficulty: High Type: Data Validation Finding ID: TOB-ETHSTAKER2-2 Target: ethstaker_deposit/utils/file_handling.py

Description The sensitive_opener function does not provide the os.O_EXCL flag while opening sensitive files on non-POSIX systems. This flag is available on Windows machines and should be enabled to provide equal overwrite protection across both operating systems. def sensitive_opener(path: Union[str, bytes, 'os.PathLike[Any]'], flags: int) → int: """ Opener to be used with the open built-in function to correctly assign permissions to sensitive files when created and written to for the first time. """ if os.name == 'posix': return os.open(path, flags | os.O_EXCL, 0o400) else: return os.open(path, flags) Figure 2.1: Utility for opening sensitive files (ethstaker-deposit-cli/ethstaker_deposit/utils/file_handling.py#L19-L27) Furthermore, although Windows ignores most of the Unix-style permission bits, the owner bit (S_IWUSR/0o200) is interpreted. If a 0o400 mode is passed to os.open on Windows, it will set the read-only attribute after the file handle is closed, providing another layer of defense against the overwriting of sensitive files. Recommendations Short term, remove the conditional statement that triggers different behavior on non-POSIX machines to use the os.O_EXCL flag and 0o400 argument on all machines.

HashEye 12 EthStaker Deposit CLI

PUBLIC Security Assessment

A. Vulnerability Categories The following tables describe the vulnerability categories, severity levels, and difficulty levels used in this document. Vulnerability Categories Category Description Access Controls Insufficient authorization or assessment of rights Auditing and Logging Insufficient auditing of actions or logging of problems Authentication Improper identification of users Configuration Misconfigured servers, devices, or software components Cryptography A breach of system confidentiality or integrity Data Exposure Exposure of sensitive information Data Validation Improper reliance on the structure or values of data Denial of Service A system failure with an availability impact Error Reporting Insecure or insufficient reporting of error conditions Patching Use of an outdated software package or library Session Management Improper identification of authenticated users Testing Insufficient test methodology or test coverage Timing Race conditions or other order-of-operations flaws Undefined Behavior Undefined behavior triggered within the system

HashEye 13 EthStaker Deposit CLI

PUBLIC Security Assessment

Severity Levels Severity Description Informational The issue does not pose an immediate risk but is relevant to security best practices. Undetermined The extent of the risk was not determined during this engagement. Low The risk is small or is not one the client has indicated is important. Medium User information is at risk; exploitation could pose reputational, legal, or moderate financial risks. High The flaw could affect numerous users and have serious reputational, legal, or financial implications.

Difficulty Levels Difficulty Description Undetermined The difficulty of exploitation was not determined during this engagement. Low The flaw is well known; public tools for its exploitation exist or can be scripted. Medium An attacker must write an exploit or will need in-depth knowledge of the system. High An attacker must have privileged access to the system, may need to know complex technical details, or must discover other weaknesses to exploit this issue.

HashEye 14 EthStaker Deposit CLI

PUBLIC Security Assessment

B. Code Quality Recommendations This appendix contains recommendations for findings that do not have immediate or obvious security implications. However, addressing them may enhance the code’s readability and may prevent the introduction of vulnerabilities in the future. • Clarify errors due to misconfigured devnets. Consider having the code exit early with a descriptive error message in the following cases. ○ If the multiplier parameter is set to zero, the deposit-cli tool will crash with a divide-by-zero error during deposit amount validation. ○ If the default None value for the GENESIS_VALIDATORS_ROOT parameter on Ephemery is not updated to the real value, then, while an exit transaction is being generated, the deposit-cli tool will crash with a low-level TypeError during ForkData type checks. This misconfiguration is gracefully handled elsewhere; this pattern should be applied consistently. • Add validation for JSON schemas. Incorrect data types in ingested JSON files can be misinterpreted, leading to confusing errors. For example, if an integer string such as “1234” instead of an integer is provided to the crypto.kdf.params.n property of the keystore file, then the encode_bytes method will incorrectly interpret it as a hex string and throw a low-level TypeError later. Instead, the deposit-cli tool should exit immediately with a descriptive error when ingesting a keystore with types that do not conform to the EIP-2335 schema, using, for example, the jsonschema library.

HashEye 15 EthStaker Deposit CLI

PUBLIC Security Assessment

C. Fix Review Results When undertaking a fix review, HashEye reviews the fixes implemented for issues identified in the original report. This work involves a review of specific areas of the source code and system configuration, not comprehensive analysis of the system. On February 18, 2026, HashEye reviewed the fixes and mitigations implemented by the EthStaker team for the issues identified in this report. We reviewed each fix to determine its effectiveness in resolving the associated issue. In summary, EthStaker has resolved the two issues described in this report. For additional information, please see the Detailed Fix Review Results below. ID Title Severity Status 1 Fork version collision is possible, allowing devnet deposits to be replayed on mainnet

Informational Resolved 2 Sensitive files on Windows can be overwritten Low Resolved Detailed Fix Review Results

TOB-ETHSTAKER2-1: Fork version collision is possible, allowing devnet deposits to be replayed on mainnet Resolved in commit f700b100 . A Dict of all fork versions used by the validation logic was added to check devnets against all known production chains. In the case of a conflict, a clear error message including the conflicting network name is returned. Test fixtures have been updated to use non-colliding fork versions. TOB-ETHSTAKER2-2: Sensitive files on Windows can be overwritten Resolved in commit 843d68f0 . The conditional statement that gave Windows weaker protections was removed, unifying file creation across all platforms. The code now applies os.O_EXCL (fail if the file exists) and 0o400 (read-only) on all systems. Test helpers have been updated to handle read-only file deletion on Windows.

HashEye 16 EthStaker Deposit CLI

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 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 17 EthStaker Deposit CLI

PUBLIC Security Assessment

 Test Coverage Disclaimer

HashEye performed all activities associated with this project in accordance with a statement of work and an agreed-upon project plan. Security assessment projects are time-boxed and often rely on information 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 software controls and security properties. 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. A project’s time and resource constraints also limit their use.

HashEye 18 EthStaker Deposit CLI