



ETHEREUM FOUNDATION

**Eth-docker**  
**Security Assessment Report**

*Version: 2.0*

**May, 2023**

# Contents

<b>Introduction</b>	<b>2</b>
Disclaimer . . . . .	2
Document Structure . . . . .	2
Overview . . . . .	2
<b>Security Assessment Summary</b>	<b>3</b>
Findings Summary . . . . .	3
<b>Detailed Findings</b>	<b>4</b>
<b>Summary of Findings</b>	<b>5</b>
Use of Weak Entropy Source for Secret Tokens . . . . .	6
Check of Creating Permissions When Executing as root . . . . .	8
Check of Compose Files Before Running Specific Services . . . . .	9
Missing Build Section in Lodestar Validator Client Compose File . . . . .	10
Miscellaneous General Comments . . . . .	11
<b>A Vulnerability Severity Classification</b>	<b>12</b>

## Introduction

Sigma Prime was commercially engaged to perform a time-boxed security review of the eth-docker wrapper. The review focused solely on the security aspects of the implementation, though general recommendations and informational comments are also provided.

## Disclaimer

Sigma Prime makes all effort but holds no responsibility for the findings of this security review. Sigma Prime does not provide any guarantees relating to the function of the implementation. Sigma Prime makes no judgements on, or provides any security review, regarding the underlying business model or the individuals involved in the project.

## Document Structure

The first section provides an overview of the functionality of the eth-docker wrapper contained within the scope of the security review. A summary followed by a detailed review of the discovered vulnerabilities is then given which assigns each vulnerability a severity rating (see [Vulnerability Severity Classification](#)), an *open/closed/resolved* status and a recommendation. Additionally, findings which do not have direct security implications (but are potentially of interest) are marked as *informational*.

The appendix provides additional documentation, including the severity matrix used to classify vulnerabilities within eth-docker

## Overview

eth-docker is an automation wrapper to run an Ethereum node. It uses the client's Docker images or source code and adds an entrypoint script to handle optional components such as MEV, checkpoint sync and additional parameters.

eth-docker's key management commands allow to import validator keys to the clients using the keymanager API.

## Security Assessment Summary

This review was conducted on the files hosted on the [eth-docker repository](#) and were assessed at commit [1af5c11](#). Retesting activities targeted commit [860acd1](#).

*Note: external libraries and dependencies were excluded from the scope of this assessment.*

The manual code review section of the report is focused on identifying any and all issues/vulnerabilities associated with the business logic implementation of the Bash scripts and Docker files. This includes their internal interactions, intended functionality and correct implementation of key management functionality, including its cryptographic functions used for key generation, key storage and network communication. Additionally, the manual review process focused on the hardening settings of the Docker configuration.

To support this review, the testing team used the following automated testing tools:

- ShellCheck: <https://github.com/koalaman/shellcheck>
- Snyk CLI: <https://github.com/snyk/cli>

Output for these automated tools is available upon request.

## Findings Summary

The testing team identified a total of 5 issues during this assessment. Categorised by their severity:

- Medium: 1 issue.
- Informational: 4 issues.

## Detailed Findings

This section provides a detailed description of the vulnerabilities identified within the eth-docker wrapper. Each vulnerability has a severity classification which is determined from the likelihood and impact of each issue by the matrix given in the Appendix: [Vulnerability Severity Classification](#).

A number of additional recommendations are also described in this section and are labelled as “informational”.

Each vulnerability is also assigned a **status**:

- **Open:** the issue has not been addressed by the project team.
- **Resolved:** the issue was acknowledged by the project team and updates have been made to mitigate the related risk.
- **Closed:** the issue was acknowledged by the project team but no further actions have been taken.

# Summary of Findings

ID	Description	Severity	Status
ED-01	Use of Weak Entropy Source for Secret Tokens	Medium	Resolved
ED-02	Check of Creating Permissions When Executing as root	Informational	Resolved
ED-03	Check of Compose Files Before Running Specific Services	Informational	Resolved
ED-04	Missing Build Section in Lodestar Validator Client Compose File	Informational	Resolved
ED-05	Miscellaneous General Comments	Informational	Resolved

<b>ED-01</b>	Use of Weak Entropy Source for Secret Tokens		
Asset	*		
Status	<b>Resolved:</b> See <a href="#">Resolution</a>		
Rating	Severity: Medium	Impact: High	Likelihood: Low

### Description

The `$RANDOM` variable is used in generating the following:

- Nimbus API token
- Nimbus JWT secret
- Reth JWT secret
- Besu JWT secret
- Lodestar API token
- Geth JWT secret
- Erigon JWT secret
- Prysm wallet password
- Nethermind JWT secret
- Teku keymanager API password

As specified in the [Bash reference manual](#), each time the `$RANDOM` parameter is referenced, it returns a random 16-bit integer within a value range of 0-32767.

Using this parameter as a source of entropy for the MD5 hashing algorithm restricts the key space to only 32768 values. The generation of a bruteforce list may be slower when a JWT token is created by concatenating two results into a 64-bit value because of the number of possible combinations, but is definitely a problem when creating a 32-bit value.

In principle, these tokens can be enumerated, such that if an attacker had access to an exposed API of a validator client from an eth-docker setup they could access privileged functions of the validator (e.g. perform a withdrawal) and also hijack communication with the Execution Layer via the known JWT secret.

*Note: After reviewing the development team's [response](#), the likelihood of exploitation has been updated to "Low", downgrading the overall severity from "High" to "Medium".*

### Recommendations

Possible ways to mitigate this issue are:

- use of `$SRANDOM` variable to expand this value to a 32-bit pseudo-random number
- use of `openssl` command
- seed `$RANDOM` with the least significant bits of the nanosecond precision timer `RANDOM=$(date +%N | cut -b4-9)`

## Resolution

The development team has resolved this issue as per commit [09155a2](#).



<b>ED-02</b>	Check of Creating Permissions When Executing as root
Asset	ethd
Status	<b>Resolved:</b> See <a href="#">Resolution</a>
Rating	Informational

### Description

On lines [565], [848], [1163], [1164], [1168] and [1803] of ethd , there are no checks to ensure that created folders and files will have \$OWNER permissions when executing as root.

Created folders and files with root permissions will not be writable by any other user.

### Recommendations

Consider using the \$EUID variable to check if the program is being run as root in which case relevant commands should be run as the \$OWNER .

### Resolution

The recommendation has been implemented in commit [6a2cbe5](#).

<b>ED-03</b>	Check of Compose Files Before Running Specific Services
Asset	ethd
Status	<b>Resolved:</b> See <a href="#">Resolution</a>
Rating	Informational

### Description

Commands `import` and `create-prysm-wallet` on lines [1189] and [1197] of `ethd` do not check the selected compose files before trying to run the corresponding services.

Use of these key management commands when the relevant compose files are not selected will result in errors.

### Recommendations

Consider checking the `COMPOSE_FILE` variable of `.env` before running the corresponding services.

### Resolution

The recommendation has been implemented in commit [cf71918](#).

<b>ED-04</b>	Missing Build Section in Lodestar Validator Client Compose File
Asset	lodestar-vc-only.yml
Status	<b>Resolved:</b> See <a href="#">Resolution</a>
Rating	Informational

## Description

Lodestar validator client compose file is missing the `build` section specifying the Dockerfile.

Without the `build` section, it is not possible to build the Docker image.

## Recommendations

Ensure that the `build` section is defined properly.

## Resolution

The issue has been addressed in commit [284af28e](#).

<b>ED-05</b>	Miscellaneous General Comments
Asset	*
Status	<b>Resolved:</b> See <a href="#">Resolution</a>
Rating	Informational

## Description

This section details miscellaneous findings discovered by the testing team:

1. **Unnecessary sudo request** - The `__maybe_sudo` variable on line [39] can get set, requiring sudo access if the docker daemon is not started. The user could start the daemon externally without the script asking for sudo access. The script could prompt the user to start the daemon, by checking if the service is running.
2. **Use of clipboard for sensitive strings.** Consider giving the user the option of using the clipboard for sensitive strings instead of storing the data in a file.

## Recommendations

Ensure that the comments are understood and acknowledged, and consider implementing the suggestions above.

## Resolution

The development team have acknowledged these findings, addressing them where appropriate as follows:

1. **Unnecessary sudo request**  
The issue has been fixed in commit [ac69601](#).
2. **Use of clipboard for sensitive strings**  
The development team have acknowledged the issue and decided to make no code changes at this time, with the following comment:  
*JWT secret and API token need to be present in files for the clients to function. Prysm wallet password is stored in a file so the Prysm client can open it. Keymanager API token and Prysm wallet password are printed to stdout if the user requests them.*

## Appendix A Vulnerability Severity Classification

This security review classifies vulnerabilities based on their potential impact and likelihood of occurrence. The total severity of a vulnerability is derived from these two metrics based on the following matrix.

Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
		Likelihood		

Table 1: Severity Matrix - How the severity of a vulnerability is given based on the *impact* and the *likelihood* of a vulnerability.

σ'