NGI REVIEW FACILITY.EU

Name: NGI Emergency Tech Review Facility

Contract nr: LC-01499045

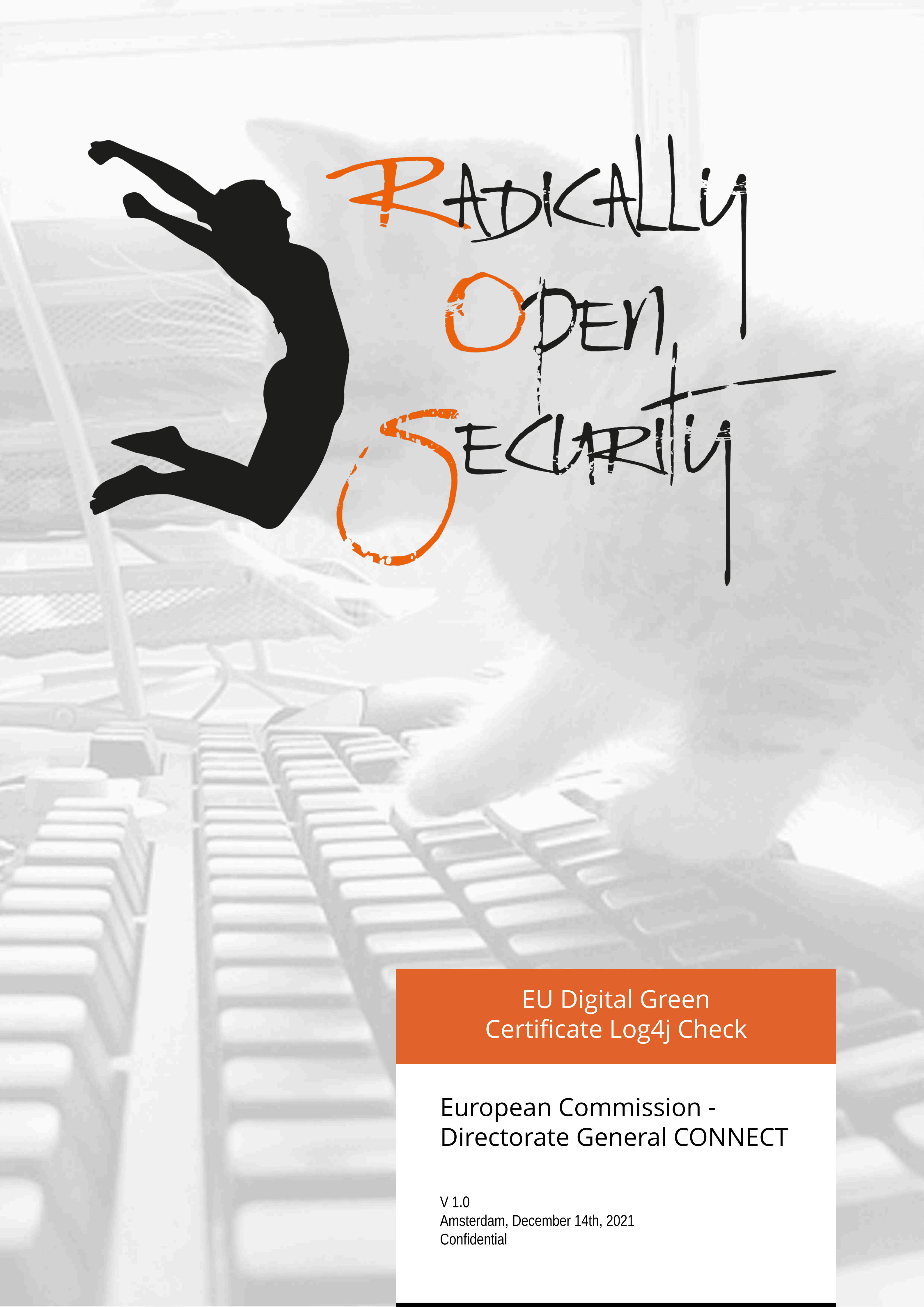
Date of signature: 30-04-2020

Name of the deliverable: Deliverable 3.8 –
High-level assessment EU Digital Green Certificate log4j check

Authors: Tim Hummel, Marcus Bointon (Radically Open Security)

Level of distribution: Confidential, only for members of the facility
(including the Commission Services)

Confidential: Yes



Radically Open Security

EU Digital Green Certificate Log4j Check

European Commission -
Directorate General CONNECT

V 1.0
Amsterdam, December 14th, 2021
Confidential

Document Properties

Client	European Commission - Directorate General CONNECT
Title	EU Digital Green Certificate log4j Check
Targets	The public repositories: https://github.com/eu-digital-green-certificates https://github.com/ehn-dcc-development
Version	1.0
Pentester	Tim Hummel
Authors	Tim Hummel, Marcus Bointon
Reviewed by	Marcus Bointon
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	December 13th, 2021	Tim Hummel	Initial draft
1.0	December 14th, 2021	Marcus Bointon	Review

Contact

For more information about this document and its contents please contact Radically Open Security B.V.

Name	Melanie Rieback
Address	Science Park 608 1098 XH Amsterdam The Netherlands
Phone	+31 (0)20 2621 255
Email	info@radicallyopensecurity.com

Radically Open Security B.V. is registered at the trade register of the Dutch chamber of commerce under number 60628081.

Table of Contents

1 Executive Summary

1.1 Introduction

Between December 13, 2021 and December 14, 2021, Radically Open Security B.V. carried out a quick check to look for traces of the vulnerable log4j library for European Commission - Directorate General CONNECT.

1.2 Scope of Work

The scope of this evaluation was limited to the following two GitHub organisations, each containing multiple repositories:

- https://github.com/eu-digital-green-certificates retrieved 12:55-12:56 on December 13th, 2021
- https://github.com/ehn-dcc-development retrieved 14:12-14:13 on December 13th, 2021

- The scope ONLY includes publicly available repositories. We did not examine any private developer repositories that may exist.
- The scope ONLY includes the content of the repositories and not the servers and infrastructure they run on.

1.3 Project objectives

The objective was to look for traces of the vulnerable log4j library in the target repositories. The reason for this investigation is the high severity issue CVE-2021-44228 in the open-source log4j logging library. More details at https://www.lunasec.io/docs/blog/log4j-zero-day/.

1.4 Timeline

The quick check took place between December 13, 2021 and December 14, 2021.

1.5 Conclusion

We conclude that log4j was not explicitly included in the build.

Our analysis suggests that log4j may be used on the servers, but we could not check that thoroughly by looking on the source code. We recommend scanning all servers that use java to check for the presence of log4j. Ideally use a trusted tool for that and be careful if using the ad-hoc created not vetted scan tools made by the security community.

We recommend running further dependency checks later on. The true extent of affected components and dependencies is likely to expand as investigations progress, so run dependency checks regularly, ideally in an automated way.

During the check a few of the repositories were patched and now explicitly require a patched version of log4j (2.15.0 and up) as a dependency. We get the impression that the developers are proactive and on a good path to address exposure to the vulnerability, however, this needs to be part of an ongoing patching strategy, not just in reaction to high-profile known issues.

We want to emphasize that security is a process – this quick-check is just a one-time snapshot. Security must be continuously evaluated and improved. Please don't hesitate to let us know if you have any further questions, or need further clarification on anything in this report.

2 Analysis

Given the large number of repositories, we could not check everything individually in the time available. We instead performed several activities to find traces of log4j on a larger scale.

Limitation: The vulnerability is quite new, therefore instances of vulnerable dependencies might be unknown as yet, and thus not appear in dependency checkers and lists.

2.1 Activity 1

We searched in all repositories for the string "log4j", "log.", "Logger." For all instances found we tried to identify how the underlying logging is provided.

We found:

- "log4j" is listed as a third-party component in the THIRD-PARTY.md document in the dgc-cli and dgc-gateway repositories. However, that doesn't mean it is actually used, as this seems to be a manually edited text file. The actual dependencies listed in pom.xml and docker files do not reference log4j.
- Java projects use lombok and @Slf4j, and nowhere do we see the default logging provider being overridden. Unless explicitly mentioned, the default logging provider should not be log4j. See http://www.slf4j.org/log4shell.html and http://logback.qos.ch/.
- The string "log4j" also appears in the repository publish-unit-test-result-action, but that seems to be an artifact of testing and not a vulnerable component itself; It seems that the test server had it in its classpath.

This paints a confusing image. The THIRD-PARTY.md and test artifact suggest that log4j is used. However we don't see it explicitly included by the dependency management. We suspect this is due to the use of the spring framework which is included in several repositories. Their documentation describes that depending on the configuration and the files available in the Java classpath, log4j can be used as logging provider. See https://github.com/spring-projects/spring-boot/blob/main/spring-boot-project/spring-boot-docs/src/docs/asciidoc/howto/logging.adoc.

We conclude that either log4j is not explicitly included in the build, or we do not understand how it is included. Evidence suggests that log4j may be installed on servers using the target projects even if it is unused, so we recommend scanning for log4j on the servers the products are deployed on to be certain.

Update 17:00 on December 13th, 2021 it seems the maintainer of the dgc-gateway repository pushed an update that explicitly updates dependencies to specify a non-vulnerable version of log4j: https://github.com/eu-digital-green-certificates/dgc-gateway/commit/421549b3b79eb38baaebe68697de2d1667f5aa2e

Update 23:24 on December 13th, 2021 The same has occurred for the dgc-lib repository: https://github.com/eu-digital-green-certificates/dgc-lib/commit/a2bc8f2b56b00e31f3de1f4b5629e01a58b45bf1

2.2 Activity 2

We used the https://snyk.io/ database to check for vulnerabilities in the available java pom.xml dependency files. The tool does discover some potential vulnerabilities, but nothing related to log4j. We recommend the developers run their own vulnerability scans, which seems to be part of the standard workflow already.

2.3 Activity 3

We used https://github.com/darkarnium/CVE-2021-44228 and https://github.com/1lann/log4shelldetect to scan the release binaries of dgc-cli and dgc-gateway and found no matches. We take this as indication that our assessment from Activity 1 is correct.

2.4 Activity 4

We used the Dutch National Cyber Security Center notification to check for any software that is likely used in the repositories or is listed in the dependencies. No matches were found at the time of the check. Source: https://github.com/NCSC-NL/log4shell/blob/3587d31a8d45c75c8661f3bcb86f3b61dab23997/software/README.md

Front page image by Slava (https://secure.flickr.com/photos/slava/496607907/), "Mango HaX0ring",
Image styling by Patricia Piolon, https://creativecommons.org/licenses/by-sa/2.0/legalcode.