DG DIGIT
Unit B3

# Lessons Learned Report

## EU-FOSSA Pilot Project

Date: 25/04/2018
Doc. Version: Final
Template Version: 2.5

PM² PROJECT MANAGEMENT METHODOLOGY

*This template is based on PM² v2.5*

*For the latest version of this template please visit the PM² Wiki*

## TABLE OF CONTENTS

# 1. INTRODUCTION

EU-FOSSA (Free and Open Source Software Auditing) was a pioneering pilot project for the Commission, establishing and security auditing the use of free and open source software within the EU institutions, especially interacting with open source software communities and the general public on this matter. Proving the usefulness of this approach was a key objective of the project, and the project team was not disappointed.

This document outlines the key lessons learned to take forward to the EU-FOSSA 2 preparatory action project as well as to provide an executive summary[1].

# 2. LESSONS LEARNED

Lessons have been grouped into areas that include execution and reflection at the end of the project. Each line is a lesson learned and can be seen in the detailed document mentioned in the footnote.

| **EU Entity Inclusion** |
|---|
| • The pilot primarily included directorates from the European Commission, in particular DIGIT, and some parts of the European Parliament. |
| **Information collection and completeness** |
| • A method for creating inventory of open source software has been defined which includes automated search scripts and manual intervention.<br>• It was noted that information for the European Parliament was not available to allow a full assessment.<br>• Involvement of open source software communities was also limited although this was not considered critical to the objectives of the project.<br>• Open source software information collection methods can be improved to allow a wider and more consistent capture.<br>• Collected inventories can now form a baseline for future exercises and perhaps even scenarios analysis. |
| **Open source software information analysis → Critical software list** |
| • The project created formal metrics to analyse the captured list of open source software, which allows an informed selection of software for code reviews.<br>• Metrics include measures for *software criticality*, *impact on security* and *project sustainability.*<br>• This has led to the creation of a shortlist of candidates for code reviews. |
| **Code review methodology** |
| • A code review methodology has been defined and will act as a base for future code reviews.<br>• It is recognised that this methodology may need further work to improve its applicability to a wider range of software (technologies, programming languages).<br>• The code review methodologies will evolve, just as software development evolves.<br>• A software application can be developed to capture attributes during the code review for later assessment using graphs and indicators. |
| **Code review experience** |
| • The code review process is complicated and laborious, and work had to be sliced into components for faster parallel processing.<br>• Even if the end result was satisfactory, there are doubts on whether the companies asked to conduct code reviews were ***specialist code review*** companies.<br>• It is recommended therefore that one of the two software selected, Apache HTTP Server and its associated libraries, be subject to further testing, as it is on such critical impact to the EU and the public. |

[1] For reference, a highly detailed list of lessons learned is available in the Project End Report – https://webgate.ec.europa.eu/CITnet/confluence/pages/viewpage.action?pageId=497945013 .

| |
|---|
| • It is recommended for Code Reviews being considered a part of the development practice. |
| **Usefulness of code reviews** |
| • In the final analysis, it was felt that code reviews, being a highly detailed manual process (despite some automation), were not very effective in finding security bugs.<br>• Rather than invalidate them altogether, it would be best that they are considered one of many tools in the security audits e.g. penetration testing, bug bounties.<br>• In fact, the project recommends that each project is subject to some degree of *code review viability assessment* prior to engaging. |
| **Public engagement** |
| • The public engagement exercise was deemed to be an all-round success.<br>• Public vote was used to select the two software packages for further review.<br>• General public supported the project and expressed views of it being a good way for European funds to be used. |
| **Developer engagement** |
| • Cooperation and communication with the FOSS developer communities though initially challenging, was a success.<br>• Developer communication ought to be simplified with greater emphasis on the results.<br>• Their feedback showed agreement of the software selected by EU-FOSSA for further audit.<br>• Whilst Joinup is a useful forum for communication, it is suggested that for developer engagement, other direct forums, methods be considered, including direct channels for submitting security issues. |
| **Best practices** |
| • Existing practices used by developers and EU institutions for open source software development have been identified and a set of best practice guidelines created for both.<br>• Inventory assessment and Code review procedures have also been defined and act as baselines for future. |
| **EU open source usage (more than just software)** |
| • We now for the first time know the precise extent of usage of open source software within the participating EU institutions, and find that it is extensive. This extends to development software, development frameworks, code management tools, code libraries amongst others. NB: former inventories of open source software were based on surveys. |
| **EU-FOSSA pilot project achievements** |
| • The project has contributed significantly to the assessment of the current security of OSS components in use at the European institutions.<br>• The project has proven the EU investing in open source software as useful, both for the measurable results as well as general public support. |
| **Other suggestions** |
| • A role to manage the open source software inventories of the EU institutions ought to be considered.<br>• Software selected for further review could take into consideration the relative size of their developer communities. The smaller communities are likely to be underinvested in security. |

## 3. CONCLUSION

The EU-FOSSA 2 Preparatory Action should build upon the success of the Pilot Project, improving the approach in particular by running Bug Bounties as a more cost effective way of addressing the software vulnerability assessment and by having stronger emphasis on communication, in order to stronger engage with the interested external stakeholders.