WP2

DIGIT B1 - EP Pilot Project 645

Deliverable 10: **List of Tools and Methods for Communicating the Results of Code Reviews**

Specific contract n°226 under Framework Contract n° DI/07172 – ABCIII

May 2016

DIGIT Fossa WP2 – Governance and Quality of Software Code – Auditing of Free and Open Source Software.

Deliverable 10: List of methods for communicating the results of code reviews

Author:

Disclaimer

Contents

List of Tables

List of Figures

ACRONYMS AND ABBREVIATIONS

API	Application Programming Interface
CERN	Conseil Européen pour la Recherche Nucléaire (renamed as Organisation Européen pour la Recherche Nucléaire in 1954 but acronym was retained)
CVE	Common Vulnerabilities and Exposures
CVRF	Common Vulnerability Reporting Framework
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
DG	Directorate General
EP	European Parliament
ESAPI	Enterprise Security Application Programming Interface
EUI	European Institutions
FIRST	Forum of Incident Response and Security Teams
FOSSA	Free and Open Source Software Auditing
FOSS	Free and Open Source Software
GASSP	Generally Accepted System Security Principles
ICASI	Industry Consortium for Advancement of Security on the Internet
I^2SF	International Information Security Foundation
ISA	Interoperability Solutions for European Public Administrations
NIST	National Institute of Standards and Technology
OS	Operating System
OSS	Open Source Software
OSSWATCH	Open Source Software Watch
OWASP	Open Web Application Security Project
SAST	Static Application Security Testing
SDLC	Software Development Life Cycle
SEO	Search Engine Optimization
WP	Work Package

1 INTRODUCTION

1.1. Objective of this Document and Intended Audience

This document represents the deliverable 10 included within TASK-07: Analysis of methods for communicating the results of code reviews, targeting their automated communication.

The objective is to analyse the methods for communicating the results of code reviews to be used in the European Institutions.

This document targets the DIGIT areas interested and/or responsible for the code reviews, related practices and tools.

1.2. Scope

To entirely understand the scope of the document, it is necessary to understand the aim of the Work Package (WP) 2. The WP2 has four tasks:

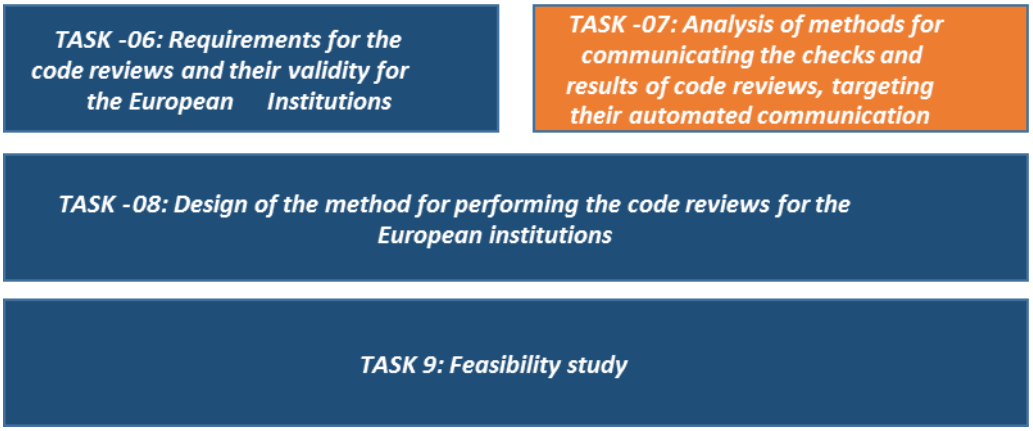
- Task 6: Requirements for the code reviews that aim to define the list of requirements for proper code reviews and their validity for the European Institutions, as well as to prepare an analysis of how they fit into the working methods of the European Commission and the European Parliament.
- Task 7: Analysis of the methods for communicating the results of the code reviews, targeting their automated communication.

  Tasks 6 and 7 will provide the requirements that the methodology, defined in task 8, needs to fulfil. For this reason, deliverables 9 and 10 (output of tasks 6 and 7) are complementary.
- Task 8: Design of the code review process to be used in the European Institutions, taking into account the requirements defined in tasks 6 and 7.
- Task 9: Feasibility study of the method defined to perform code reviews, to be used in the European Institutions.

This is deliverable 10, the result of task 7, and therefore covers the requirements for the communication methods of the results of the code review.

A graphical representation of WP2 tasks and their relationship is depicted in Figure 1.

Figure 1. WP2 Tasks



1.3. Document Structure

This document consists of the following sections:

- Section 1: **Introduction**, which describes the objectives of this deliverable, intended audience and Scope.
- Section 2: **Methodological Approach to Building the Analysis**, which describes the steps that we followed to identify the requirements for the communication methods, in line with the scope.
- Section 3: **Requirements for the Communication Methods Inside the European Institutions,** where the desirable requirements for the communication methods are identified.
- Section 4: **Study of the Current Communication Methods and Tools**, that includes a list of in-use and optional tools and methods for communicating the results.
- Section 5: **Proposed Tools and Methods for Communicating the Results of Code Reviews**, which details the results of the research conducted and the proposed tools and methods.
- Section 6: **References**.
- Section 7: **Annexes**.

1.4. Key Success Factors

All steps described in Section 2 - Methodological approach to building the analysis, will ensure the fulfilment of key success factors related to this deliverable:

- The method for doing code review includes necessary tasks to obtain a complete set of correct results (use of automatic tools, code reviews by peers, screening by an external agent).

1.5. Deliverables

1 Deliverable 11: Design of the Method for Performing the Code Reviews for the European Institutions.

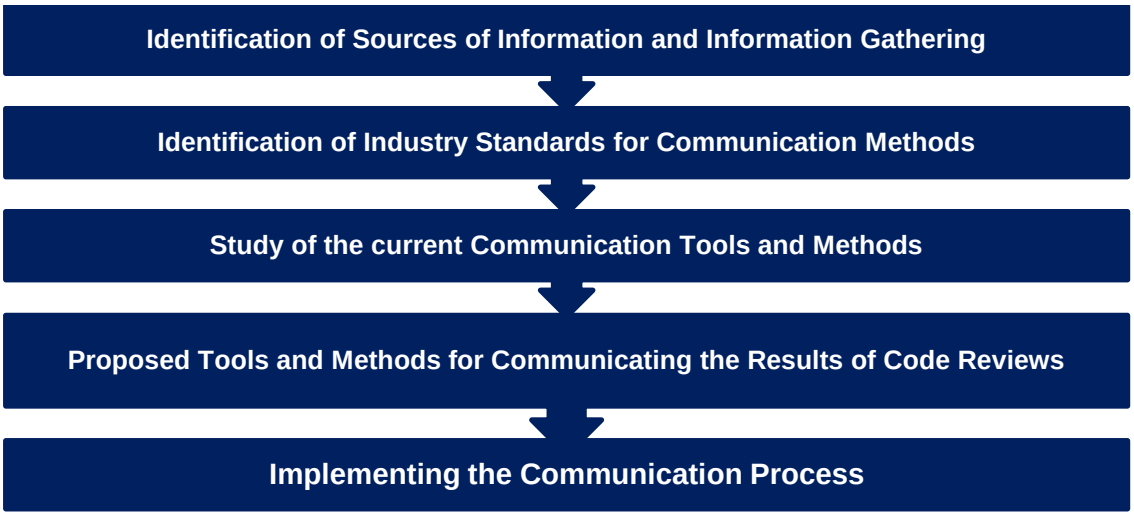
2 METHODOLOGICAL APPROACH TO BUILDING THE ANALYSIS

The goals of this task are: 1) to identify a set of requirements for the communication methods, that need to be followed by the European Institutions when communicating the results of the code reviews, and 2) to analyse and propose tools and methods for communicating such results.

The process that we will define in TASK-08: Design of the Method for Performing the Code Reviews for European Institutions, will fulfil all the requirements regarding the communication methods defined in Section 3.

The approach that has been used to execute this task has four steps:

Figure 2. Methodological approach to building the analysis



The following sections contain detailed explanations of each one of these steps and the activities that will be carried out in them.

2.1. Identification of Sources of Information and Information Gathering

This is the first step to be carried out, in which we aim to determine a set of desirable requirements that are directly related to the different methods of communication available to distribute the results of audits and, in this particular case, code reviews.

- There are many sources available from which it is possible to obtain information for these cases, including websites, organisations, books, etc.

- **Websites**: there are multiple sites on the Internet that provide a lot of helpful information, requirements, guidelines and general good practices regarding reporting in general that must be applied to code review audits. The most relevant ones used within this study are:
    - http://www.bridging-the-gap.com/ [1] This website has been used to identify the requirements of a good reporting. Despite the fact that it is a website about general reporting, it was useful to collect the desirable characteristics of good reporting.
    - http://betterevaluation.org/ [2] [3]. There are two references to this website. As the website mentioned above, this site was useful to identify the requirements of a good report.
    - http://oss-watch.ac.uk/ [4] This website, focused on everything related to Free Software, Open Source Software and Open Source Hardware provides in its reference a guidance about tools and best practices about reporting.

      Even though the main focus of the sites is not reporting, they all provide information on the subject, and as such are a good sources of information and references.
- **Research groups and organisations**: there are also a lot of organisations and entities that focus on carrying out research, publishing articles and generating guides for secure code reviews, and that are an invaluable source of information for this study. Some of the most important ones are:
    - **The MITRE Corporation:** It is an American not-for-profit company that operates multiple Federally Funded Research and Development Centres(FFRDCs), among them the National Institute of Standards and Technology (NIST). They provide innovative, practical solutions for some of the most critical challenges in defence and intelligence, aviation, civil systems, homeland security, the judiciary, healthcare, and cybersecurity. [5]
    - **OWASP:** The Open Web Application Security Project (OWASP) is a worldwide not-for-profit charitable organization focused on improving the security of software. The online community creates freely-available articles, methodologies, documentation, tools, and technologies in the field of web application security.
    - **OSSWATCH**: OSS Watch is the United Kingdom's advisory service for issues relating to free software and open source software based at the University of Oxford. OSS Watch provides unbiased advice and guidance on the use, development, and licensing of free software, open source software, and open source hardware. Despite the fact that they seem to be inactive, their information is valuable.
    - **Mozilla Open Source Support:** Mozilla Open Source Support (MOSS) is an awards program specifically focused on supporting the Open Source and Free Software movement. Mozilla is part of the Open Source and Free Software movement – MOSS, a systematic way to provide a new level of support to this community. [6]

- **Core Infrastructure Initiative:** It is a project of the Linux Foundation that was announced on 24 April 2014 in the wake of the Heartbleed bug. It aims to encourage a collaborative, pre-emptive approach for strengthening cyber security by funding and supporting free and open-source software projects that are critical to the functioning of the Internet and other major information systems.

- **Compliant Documents:** Regarding the Generally Accepted System Security Principles (GASSP), two main articles were consulted:
  - [6] **"**NIST Special Publication 800-14' from **NIST**, available at http://csrc.nist.gov/publications/nistpubs/800-14/800-14.pdf.
  - [7] "Generally Accepted System Security Principles" from **I²SF** (International Information Security Foundation), available at http://www.infosectoday.com/Articles/gassp.pdf

Once we researched the sources of information, we gathered the relevant information to define the requirements for the communication methods, explained in Section 3.

2.2. Identification of Industry Standards for Communication Methods

Once we identified the information sources available on the Internet and gathered the relevant information, the next step focuses on researching industry standards that are designed for, or can be applied to, code review audits.

In this aspect, there are a lot of options, many of them being relevant not only in enterprise environments but also in general usage as well. In some cases, they have been adopted by open-source communities and development groups. The most relevant ones, that are used not only for code reviews but for audits in general, include:

- CVE: *Common Vulnerabilities and Exposures*
- CWE: *Common Weakness Enumeration*
- CVSS: *Common Vulnerability Scoring System*
- CVRF: *Common Vulnerability Reporting Framework*
- There are also several organisations that focus on the development of these industry standards in order to maintain and improve them to properly cover the ever-evolving security paradigm. They have not been mentioned before, including ICASI (*Industry Consortium for Advancement of Security on the Internet*) and FIRST (*Forum of Incident Response and Security Teams*) as notable references.

2.3. Study of the Current Communication Tools and Methods

This step focuses on determining the different methods and tools for communicating the results of code reviews that are available on the European Institutions and within DIGIT, and of any processes or existing guidelines that may be in use.

For this purpose, it was necessary to have the support of DIGIT staff in order to gather this information and to obtain the necessary details from the different solutions in use or available to use. Also at this point, information gathered beforehand is also included to complement this chapter.

Moreover, a search for alternative solutions will also be carried out in order to identify those that could be potentially used to communicate the code review results. From this solutions, an initial review will be done in order to determine their suitability and applicability for the purposes described on the methods in this document.

2.4. Proposed Tools and Methods for Communicating the Results of Code Reviews

The final step of the methodology covers the selection of a tool, or set of tools, for communicating the results of the code reviews, and the design of the methods and processes to follow.

The first part will include a list of the tools identified, both within and outside the European Institutions, including a brief description. A justification is provided for those tools that are discarded.

The second part will include a structure of the methods and processes that must be followed in order to establish a proper communication activity of the results of the code reviews that will be carried out, or requested by DIGIT.

2.5. Implementing the Communication Process Using Industry Standards

In this step, an explanation will be provided regarding the implementation of the communication process using the industry standards identified during the research.

3 REQUIREMENTS FOR THE COMMUNICATION METHODS INSIDE THE EUROPEAN INSTITUTIONS

As a result of the research conducted in section 2.1, we identified three main areas that are required for a successful communication method:

1. **Content of the communication**: This area will cover the information that needs to be provided with each check and/or communication made according to the method established.
2. **Format of the communication:** This covers the definition of the format that must be followed in order to communicate the final results report.
3. **Communication channel:** This covers the selection of the communication channels that will be available to distribute the results. These channels can vary if requested by the stakeholders during the planning phase of a code review.

In the following sections, each area and its related requirements is described.

3.1. Content of the communication

The main result obtained from a Code Review is a well written, complete and easy-to-understand report that highlights the security issues and their severity, detected during the code review.

The report has to be informative and useful to the executive management, the stakeholders responsible for the code, and the more technical staff, including the developers themselves.

Therefore, we propose building a Full Report containing all the relevant information from the code review, covering the different audiences indicated on the previous paragraph. This way, the structure of the report will provide each audience with the information needed, and if properly defined, will generate specific reports for them.

The sections a full report should include are the following:

1. **Executive Summary**

   The executive summary consolidates and summarizes the results of the code reviews, presenting a detailed enough overview of the situation and provides easy-to-manage indicators to give an overall evaluation of the code review results.

It should be written in a clear, concise way, easily understood by people, avoiding technicalities and presenting the information in an objective way. It should contain graphs or similar visual aids to emphasize relevant points. For this purpose, an overview of the action plan is also recommended to include, if applicable. Results should be presented in high-level, with their respective score and any relevant recommendations.

This section should be included at the beginning of the document, after the indexes and introduction but before any other content section (methodology, results…) in order to allow easy access and enable fast reading.

2. Common content

All reports should include several "common" sections that provide insight on the report and allow any reader to understand the detail of what has been done. This includes the following points:

- **Introduction**: explaining the scope, objectives and the context of the code review, allowing the reader a basic understanding on what is the motive of the audit.
- **Methodology**: details of the process and methods that have been followed in order to carry out the code review and how the results are obtained and scores calculated. This should contain enough detail to allow the code review to be repeatable if needed.
- **Audit details**: containing all the information gathered during the code review and the execution of the tests.
- **Recommendations and conclusions**: recommendations generated in order to fix or mitigate the findings, alongside with any conclusions that may be deemed necessary.
- **Annexes**: a section to allow the code reviewer to include any additional information, not covered on the previous sections that may be of relevance for the audit.

3. Technical Report

The audit details section there should be a specific section with a detailed description of the test cases and the results obtained.

This section is targeted to software architects and developers, allowing them to easily understand the weaknesses or vulnerabilities found, how they were found, and their technical details.

The following information is included for each finding:

- Description of the finding.
- Details of the finding (including the root cause, areas affected, code details, etc.).
- Steps taken in order to detect and identify the finding.
- Scoring (severity rating) of the finding/issue based on different factors such as integrity, confidentiality, availability, ease of discovery…

This section should contain all the findings, regardless of their severity. However, it is possible to sort them based on their severity so that the developers and software architects can focus on the most critical ones first.

Also, if a vulnerability found has a direct or indirect impact on other modules or sections of the code, this will be clearly stated on the results to ensure that it is taken into consideration when evaluating possible solutions.

Finally, the severity rating contributes to the calculation of risk rating and helps prioritise the remediation effort. Typically, the assignment of a risk rating to the vulnerability involves a specific risk analysis based upon variables such as threat, vulnerability and impact. This severity should be presented both for each individual variable and as a global score of the issue itself.

Going deeper into the severity rating, regarding on the different standards identified and mentioned on subsection 2.2, such as CVE, CVSS, etc., the severity of the issue will be determined by calculating the risk value of the Threat, Vulnerability and Impact factors.

- **Threat**: focuses on those factors that are directly related with the attack vector, and more specifically the probability that the attacker (either intentionally or unintentionally) manages to successfully take advantage of the issue. To measure the threat, the following should be considered:
  - *Skills required:* The needed abilities of the attacker to perform the attack..
  - *Opportunity:* Access and resources needed to carry on the attack.
  - *Dimension:* Profile and permissions of the attacker.
- **Vulnerability**: those details specifically related to the vulnerability itself, focusing on the chance of its discovery and/or exploitation. To measure the easiness of discovery, exploitation and awareness on the issue.
  - Ease of discovery: How easy is to detect the issue.
  - Ease of exploitation: How easy is to take advantage of the issue.
  - Awareness: How much the issue is known.
- **Impact**: it is centred on the common security concepts (confidentiality, integrity and availability) and how the issue identified affects them. To measure the consequences of the exploitation of the issue on the target
  - **Confidentiality**: Privacy or the ability to control or restrict access so that only authorized individuals can view sensitive information.
  - **Integrity:** Information is accurate and reliable and has not been subtly changed or tampered with by an unauthorized party..
  - **Availability:** Information and other critical assets are accessible to customers and the business when needed..

All of these factors will help rank the Threat, Vulnerability and Impact of each issue with **Low**, **Medium** or **High** values. Those values will help determine the severity of the issue. This will be

detailed further in Deliverable 11: Design of the Method for Performing the Code Reviews for the European Institutions.

3.2. Format of the Communication

It is fundamental to have the format of the code review results properly defined to ensure that it is understandable and intuitive. The lack of proper and common formatting would only serve to confuse the recipients and even cause the omission of critical information.

Regarding the formatting, there are two factors to consider: content sections structure and document formatting.

- **Content sections structure**: applies mainly to the final report, which contains the contents specified in section 3.1. The detailed description of the contents and a structure of these sections and any relevant considerations will be provided in "*Deliverable 11 – Design of the method for performing code reviews for the European Institutions*".
- **Document formatting:** there are two scenarios to consider: the final report and the communication checks:
    - ***Final report***: it will be formatted following a template generated with the structure that will be provided in Deliverable 11 (as indicated on the previous paragraph) and will be delivered in PDF or similar format to allow recipients to access all the information but preventing them from modifying the contents without proper authorization. Optionally, the use of signed documents could be considered.
    - ***Results communication***: the information to distribute covers the issues and findings of the code reviews only, omitting methodology, recommendations, executive report and other sections not directly related to the vulnerabilities themselves. The format to follow is the CVRF v1.1, which is included in Annex 9.1.

Alternatively, and as a result of the research, we have identified different, but closely related, methods to score and catalogue weaknesses and vulnerabilities including CVE (Common Vulnerability Enumeration), CWE (Common Weaknesses Enumeration) from The MITRE CORPORATION and CVSS (Common Vulnerability Scoring System) from FIRST. These standards can also be used for the enumeration and scoring of the findings if there is a need to publish them on the Internet.

3.3. Communication Channel

There are multiple options for distributing the results of the code reviews, ranging from the use of Internet to cloud-based solutions. However, our focus will be online solutions.

Based on the requirements and needs of the European Institutions and DIGIT, we recommend using an online tool to distribute the code review results as it is the quickest and easiest one to use. Also this communication channel supports the implementation of user access management systems based on roles and profiles to easily control user access to the information, which could also be integrated with existing user management solutions (i.e. Active Directory or LDAP). This tool (or tools) will be used to distribute the full report generated during the code review.

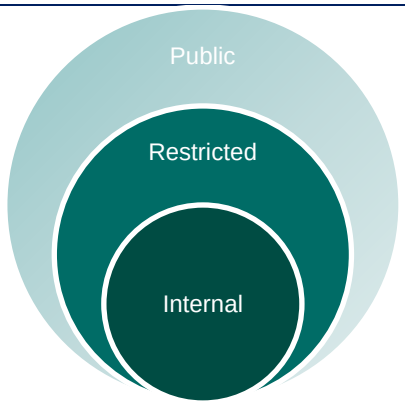
3.4. Identification of Staff that Will Use the Methods

Several roles participate in the distribution of the report of the code review results, each one having different responsibilities and tasks to be carried out. The participants considered, either directly or indirectly are:

- **Code review team**: The code review team is in charge of generating the report of the code review results, preparing the content and ensuring that it is ready for distribution. However, it will only be distributed once it has been validated by the respective stakeholder.
  If needed, the code review team will be in charge of making any updates, changes or fixes to the document in order to ensure that it complies with applicable regulations, directives or distribution formatting.
- **DIGIT and EUI stakeholders**: the final report is not distributed until it is validated by the stakeholders responsible for the code review. Furthermore, these stakeholders, as well as stakeholders from other institutions, are also recipients of the report.
- **Developers and FOSS communities**: they are also possible recipients of the reports and/or the results (depending on the scope set for the project).
- **General public**: if specified by the stakeholders responsible for the code reviews, it is also possible to distribute the final reports to the general public. Alternatively, it is also possible to publish only the new critical vulnerabilities found using CVE, CWE, CVSS or similar systems.

Now, as there is a wide range of participants, and there are privacy and security concerns to consider, the following groups of recipients are considered, as depicted in Figure 3.

Figure 3: Recipients of Report of Code Review Results



- **Public**: Available to the general public, either the full report, the results, or the public vulnerabilities
- **Restricted**: includes authorised staff from DIGIT, EU Institutions and FOSS Communities.
- **Internal**: only authorised staff from DIGIT and the EU Institutions, and the code review team.

An important decision that needs to be agreed upon the planning phase of each code review, is that if a new critical vulnerability is found and has not been published previously, during the assessment phase (before the final report is generated), to decide whether the information about that vulnerability will be distributed and to whom. This will be done only for very specific cases, and several factors must come together:

- It has been agreed during the planning of the code review to allow mid-assessment publications.
- It is a critical vulnerability or weaknesses.
- It has not been discovered/reported before (there is no CVE, CWE, CVSS or similar published).
- The DIGIT stakeholder in charge approves the dissemination of the information.

If these factors are matched, then it is acceptable to publish the corresponding CVE, CWE, CVSS or equivalent, following their respective publication process, in order to inform the FOSS communities and/or relevant developers of the issue, its criticality and all the relevant data.

Once the findings have been assessed, the corresponding developers will be notified and a solution plan will be developed, establishing well-defined deadlines for the findings to be fixed, depending on their severity (ex. 30/90 days). If the developers do not fix the issues, or refuse to work on them, then the responsibility will fall on the stakeholders in charge, that will have to decide who will perform this task.

4 INDUSTRY STANDARDS FOR COMMUNICATION METHODS

There are several standards defined that contain a format to report the vulnerabilities, weaknesses and/or issues, as well as the methods to follow in order to report them. The most common standards where briefly mentioned in subsection 2.2, and are explained here in more detail:

- **CVE**: (Common Vulnerabilities and Exposures): it is a dictionary of publicly known information security vulnerabilities and exposures.
- **CWE** (Common Weakness Enumeration): it is similar to CVE, but focused on providing common means to measure and distribute information about software weaknesses both in the applications' architecture and design.
- **CVSS** (Common Vulnerability Scoring System): it is an industry standard, available for free and open use, designed to evaluate and distribute known computer system security vulnerabilities. It provides a standardized scoring system covering multiple factors and a variety of security areas.
- **CVRF** (Common Vulnerability Reporting Framework): it is an industry standard developed to manage the reporting of vulnerabilities. It has been developed by the non-profit organization ICASI. It can be linked to other common security standards including CVE, CWE and CVSS. It provides a machine-readable format that can be automatically processed to allow better automation and integration of different audit and reporting solutions. A Mind Map schema of this standard is attached in Section 9 – Annex 1.

As best practices for communicating code review findings, the most useful ones to follow are:

- **Appropriate tittle of the vulnerability**: The tittle of the finding must capture the issue clearly, succinctly, and convincingly for the intended audience. In general, It is recommended to phrase the title in a positive way such as "Encode output to prevent Cross-site Scripting (XSS)".
- **Location of the vulnerability**: The finding must be specific about the location in both the code and as a URL. If the finding represents a persistent problem, then the location should provide many examples of actual instances of the problem.
- **Detail the vulnerability:** Provided enough detail about the issue so the vulnerability and the possible attack scenarios are clearly understood, and the keys factors driving likelihood and impact of the exploitation of the issue are easily known.
- **Risk ranking:** Assign a qualitative value to each finding based on their severity. Some possible risk ratings are Critical, High, Medium, Low or Info. The boundaries of these values will be specified in

Deliverable 11: Design of the Method for Performing the Code Reviews for the European Institutions. Justifying the assigned risk ratings will allow stakeholders (especially non-technical ones) to gain more understanding of the issue at hand.

Two key points to identify are:

- *Probability* (Likelihood => ease of discovery and execution): Avg. Threat & Vulnerability
- *Impact*: Business and/or Technical

- **Provide solutions:** In order to resolve or mitigate the effects of the findings some remediation actions must be provided. These can be fixes, workarounds, best practices, etc.
- I**nclude references:** If existing, references to the issue must be included (known CVE, CWE, Security Bulletins, etc.).

5 STUDY OF THE CURRENT COMMUNICATION TOOLS AND METHODS

This section covers the identification of the tools available for communicating the results of the code reviews, as well as the methods for communicating the results using these tools, regardless of the recipients of the information:

- **Communication tools**: all the tools that could be used to distribute results, including those already in use by DIGIT and the European Institutions.
- **Methods for communicating the results**: this will include a process structure defining the main points that need to be followed, independent of the tools used.

5.1. Identification of the Tools to Communicate the Results of the Code Reviews

This section covers the tools that can be used to communicate the results of the code reviews and that are in use within DIGIT/EUIs, and also the optional tools.

Regarding the tools in use within the EUI, we identified the following ones:

- **Confluence (Atlassian, Inc.)**: it is a collaborative platform designed to allow to easily share and contribute knowledge and information. It is widely used in development processes to assemble and share ideas, product requirements and documentation. It has been used as an enterprise wiki / content collaboration tool for a long time.
- **Jira**: It is a bug and issue tracking environment that includes multiple project management functionalities. It supports integration with many other open source tools, including Confluence.
- **JoinUp**: It is a collaborative platform created by the European Commission and funded by the European Union via the Interoperability Solutions for European Public Administrations (ISA) Programme. It offers several services that aim to help e-Government professionals in sharing their experience with each other. [8]
- **E-mail**: the most common tool for electronic communication between people and organisations; different clients and servers are in use in the institutions, all compatible with common e-mail standards.

According to the information provided by DIGIT, all of these tools are quite valuable communication tools to distribute the Code Review results, and can potentially be used as a communication platform within the EUI.

We want to highlight the CERT-EU as a valid option to publish the results of Code Reviews, as it has systems in place designed to easily and reliably distribute information among the European Institutions regarding security incidents, vulnerabilities and any other security-related data.

It is important to consider that these tools are not mutually exclusive; therefore, by using one tool, other options are not immediately discarded and could be used one alongside the others.

On the other hand, there are also other options available, both commercial and open-source, that can potentially be used to manage these communications and distribute the results of the code reviews.

The following are the most relevant ones reviewed during the information gathering process of sections 2.1 and 2.2:

- **ThreadFix:** it is a software vulnerability aggregation and management system designed to reduce the time needed to fix software vulnerabilities. ThreadFix imports the results from dynamic, static and manual testing to provide a centralized view of software security flaws across multiple development teams and applications. There used to be a free “Community Edition” available for non-commercial use, but this version has been discontinued in favour of fostering the commercial versions already available.
- **jsreport:** It is an open source reporting software. Reports in jsreport are defined using JavaScript templating engines and being rendered at the jsreport server into PDF or HTML. It is written in nodejs and mongodb by Jan Blaha and it's currently in the first public preview version. jsreport uses PhantomJS and Apache FOP to transform XML generated by JavaScript templating engines into PDF.
- **Crystal Reports:** It is a business intelligence application, currently marketed to small businesses by SAP SE. It is used to design and generate reports from a wide range of data sources. It supports data sources like data bases (PostgreSQL, MS Access, MS SQL Server, MySQL, Oracle, etc.), spread sheets like MS Excel, text files, XML files, etc. It became integrated with Visual Studio versions prior to 2010
- **Jasper Reports:** It is an open source Java reporting tool that can write to a variety of targets, such as: screen, a printer, into PDF, HTML, Microsoft Excel, RTF, ODT, Comma-separated values or XML files. It can be used in Java-enabled applications, including Java EE or web applications, to generate dynamic content. It reads its instructions from an XML or jasper file. Its reports are defined in an XML file format, called JRXML, which can be hand-coded, generated, or designed using a tool. The file format is defined by a Document Type Definition (DTD) or XML schema for newer versions, providing limited interoperability. The main difference between using XML and a .jasper file is that the XML file should be compiled at runtime using the JasperCompileManager class. It can be integrated with IDEs such as Eclipse, NetBeans or Websphere.

Even though most of these tools are not considered as “Vulnerability Management Tools” (with the exception of ThreadFix), they are considered appropriate for the task of communicating the results of the Code Reviews to the intended recipients because of their performance, which includes document management, users and groups management capabilities, group emailing and issue tracking, features that make them valuable for this purpose.

5.2. Identification of Methods to Communicate the Results of the Code Reviews

Alongside the tools used for communications, there is also the need to have methods defined in order to use them to distribute the reports and checks of the results from the code reviews.

Based on our research, there are no established communication methods defined for the European Institutions to share the results of code reviews (there are methods for other concepts and audits). In most cases, multi-purpose methods are used to distribute the results, such as the direct distribution of the reports using e-mail systems or internal document repositories, where access is given to those parties that need it.

Alternatively, there are multiple options available outside of the European Institutions, many of them supported by open and industry standards.

1. First, there are the traditional distribution methods, such as sending e-mails, file-sharing and online document repositories to share the code review results with the intended audience.
2. Second, it is much more common to only share the technical details of the code reviews themselves, using different tools. For example, most communities make use of their own bug-tracking systems to manage the vulnerabilities, weaknesses and issues found by other users, collaborators and even the general public. While these systems are designed for general vulnerability/weakness/issue reporting, they are perfectly useful for distributing the findings of a code review.
3. Finally, and oriented to the IT sector and commercially-focused areas, there are several standards defined that contain a format in which to report the vulnerabilities, weaknesses and/or issues, as well as methods to follow in order to report them. This is quite important, as the vulnerabilities published following these standards must contain a set of required information to ensure that it can be verified, evaluated and eventually fixed (or at least mitigated). The most common standards where briefly mentioned in subsection 2.2 and in chapter 4.

5.3. Comparison between the results of the study and the requirements defined

Once a list is defined that contains the tools and methods that are available, either within the European Institutions or optional ones, the next logical step is to compare these solutions with the requirements that have been defined in Section 3.

Due to the nature of the requirements, some of them apply exclusively to either the method or the tool. The compliance of the tools is depicted in Table 1.

Table 1: Comparison between tools and requirements

Tool	Content	Format	Channel
Confluence	Allows document uploading.	Supports any kind of documents.	Sharing documents. E-mail links to documents.
Jira	Allows document uploading.	Supports any kind of documents.	Sends email notifications to specific groups or recipients.
JoinUp	Allows document uploading. Provides document repository.	Supports any kind of documents.	Sends email notifications to intended recipients. Document sharing Forums and wiki.
E-mail	Allows sending documents.	Supports any kind of document. No formatting supported (CVE, CWE, CVSS…).	Sends email to intended recipients.
Threadfix	Uses custom internal reports, generates PDF and CSV. Integrates results from other tools.	Own internal reports, vulnerability analysis and prioritization.	Issue tracker, integration with other common tools.
JSReports	Does not allow document uploading. It creates the documents from an uploaded JSON.	Creates HTML, PDF and TXT. Development in progress. Custom formats supported, but has to be mapped by the user.	Allows document sharing. Sends links to documents. Allows to send emails. User management and roles.

Tool	Content	Format	Channel
Crystal reports	Allows document uploading.	Supports MS Word, MS Excel, HTML, ODBC, PDF, PT, RTF, TXT, CSV, REC, and XML.	Links to documents. File sharing. Does not send e-mails.
Jasper Reports	Does not allow document uploading. It allows uploading the data sources but not the document itself. Documents are stored locally.	Supports various formats like XML, CSV, JSON, etc…	Does not allow file sharing. Does not allow links to documents. Does not send emails.

On the other hand, there are also requirements that apply for the methods for communicating the results. As the methods identified are very specific, they can be sorted differently, as shown on Table 2:

Table 2: Comparison between methods and requirements

Requirement	Applicable methods
Content	Final Report
Format	CVE, CWE, CVSS
Channel	N/A

In this particular case, the main point missing is a process, or methodology, defining with certainty the steps to take in order to communicate the results of a code review. This includes from the moment the results are finalized and up to the point where they are distributed among the intended recipients.

6 PROPOSED TOOLS AND METHODS FOR COMMUNICATING THE RESULTS OF CODE REVIEWS

At this point there is a list of available tools (either already in use on the European Institutions or as feasible alternatives), as well as a list of existing methods and standards for communicating the results of code reviews. These options have been compared against the requirements set on this deliverable, in order to determine their possibilities.

At this point, and in order to allow the inclusion of these concepts on the code review methodology, the following solutions have been selected to be used to distribute the results and checks:

Table 3: Tools/solutions selected

Tool	Description
JIRA	This tool meets most of the requirements set for the final report sharing process, as it allows uploading, storing and sharing documents easily and even automatically once configured. It provides a robust user management system that allows the creation of restricted groups and facilitates the process of sending updates and new information to specific collectives. Further details can be found in Section 3.4.
JoinUp	It is a viable alternative to JIRA, meeting also most of the requirements and allowing the sharing of the final report documents. It excels at these tasks, however it does not include a group management system by default. It also provides additional functionalities that could potentially be automated to distribute checks to the intended parties.
jsreport	In order to distribute the checks, this open-source tool provides a robust and highly configurable alternative capable of automating the process of parsing raw information provided in JSON format, generating distributable reports and on a web front-end. It also allows easy sharing to different groups of users.

Regarding the communication methods to be used to distribute the results of the code review, the following were selected:

- **Final report document** containing the structure indicated on the content and format requirements.
- Use of the **Common Vulnerability Reporting Framework (CVRF)** to manage the check generation and distribution.

Now, as both the tools and methods for communicating the results have already been selected, the final step is to define the existing scenarios where they will be used and their process. There are two main scenarios defined: critical vulnerability reporting (during the assessment phase) and standard reporting (at the end of the code review).

6.1. Reporting of a Critical Vulnerability Found During the Assessment Phase

The first possible scenario considered is that during the assessment a critical, not previously known vulnerability is found or identified. As in this phase its validity and score have been checked, it is important to determine if there is a need to communicate it as early as possible, even if the final report is still on the works and will not be ready for a while.

There is one main exception: the following procedure will not be carried out if this scenario is not approved on the planning phase of the code review, meaning the involved stakeholders wish to receive all the results at the same time during the reporting phase and are not concerned on receiving the critical vulnerabilities separately.

If this is not the case, the following steps will be followed:

- Validate and cross-check the severity and the scoring of the vulnerability.
- Determine if it has already been reported, is known, or a newer hotfix or patch is available to solve or mitigate it.
- Contact the DIGIT stakeholders in charge and provide all the related information of the vulnerability, asking them for the approval to distribute this information with priority to the audience defined during the scope of the project. **If approval is not obtained, the following steps are discarded**.
- Prepare the information and format it according to the template in use (i.e. CVRF).
- Distribute, or send it to the DIGIT stakeholder for distribution, among the parties that have to be involved. As this is a critical vulnerability scenario, **no information would be sent to the general public**, even if they are set as a potential audience in the scope of the project.

Due to the particularities and specific characteristics of this scenario, in order to contact the stakeholders and key personnel and provide all the necessary information regarding the critical finding, it is highly recommended to make use of e-mail, or similar, method of communication as it can provide the needed exchange in an efficient and timely manner.

6.2. Standard Reporting

The second scenario is the standard one, achieved by following the methodology defined on *Deliverable 11: Design of the Method for Performing the Code Reviews for the European Institutions*. At this point, the results of the code review audit are assessed and consolidated into a single report.

- Prepare the information and format it according to the template in use (i.e. CVRF).
- Contact the DIGIT stakeholders in charge and provide the final report and the formatted checks for their validation.
- If the report is not validated, the code review team will be responsible of fixing and updating the document to fit the needs specified by DIGIT.
- Once approved, the documents will be distributed to the involved parties and the target audience using the communication methods selected during the planning stage of the code review.
- The dissemination of the final report and the results checks can be done either by the code review team or the DIGIT stakeholders, again depending on how it was defined during the planning phase.

For this scenario the proposed method is the use of a content manager or document repository to manage the documents generated from the reports and checks of the Code Review. This solution must provide user access management to meet the security and privacy requirements for the groups defined in point 3.4 and depicted in figure 3. It will make the documents accessible for all the intended recipients according to their specific permissions. The notification of new information and reports can be carried out via email to the corresponding recipients. These recipients will access the information stored in the repository using their assigned credentials.

According to ‘Deliverable 11: Section 3.1.1 Planning’, FOSS developers will be contacted and engaged during the planning phase of the code review, and are also included in the distribution list to receive the report with the results of the code review.

The dissemination of the results will be carried out as explained in “Deliverable 11: Section 3.1.4.2”. During the planning phase some decisions are taken, such as the identification of the FOSS communities and project owner points of contact, engagement of these contacts starts, and the access control to the documentation is defined.

The selection of these tools and methods aims to improve the result sharing process, minimising the interactions that the user has to perform in order to send the results of the code review to the intended recipients.

7 IMPLEMENTING THE COMMUNICATION PROCESS USING INDUSTRY STANDARDS

The implementation of the communication will be conducted as follows:

1. Once the results of the code review are assessed, their severity will be ranked as explained above in this document at subsection 3.1 following a custom variant of a standardised score methodology, such as CVSS (v3) or CWSS.
2. After the evaluation and scoring of the findings based on their Threat, Vulnerability and Impact and the impact analysis of the findings, the development of the executive summary and the final detailed technical report including all the results, recommendations and conclusions of the code review starts.
3. The report of the checks that contain the information about findings of the code review is also developed following a standardised, and widely used, framework such as CVRF. At least the following information about the vulnerabilities should be provided:
    - Title of the vulnerability
    - Details/explanation of the vulnerability
    - Location(s) of the vulnerability
    - Severity ranking of the vulnerability
    - Suggested remediation
    - References
4. Once the reports and checks are completed, they must be approved by the stakeholders responsible for the reports.
5. Once the approval is obtained, the reports are ready for dissemination to the identified recipients using the tools proposed in Section 6. It is recommended the use of Confluence and/or JIRA to provide access to the Final report and the Detailed Technical Report and the use of jsreport for the communication of the results. Jsreport allows to map an XML that follows the CVRF schema and create a proper report while Confluence/JIRA allow the dissemination of the reports to different groups of recipients taking advantage of the users and groups management feature of these applications. In the case of JSReports this feature can be implemented by scripts, which are oriented to the automation of the communication.
6. The identification of a critical and previously not known issue is a situation that can happen during the assessment prior to the reporting. In this special situation, the stakeholder in charge can be informed to take further actions, as explained in Section 6.

8 REFERENCES

[1] Bridging the Gap, “How Do You Collect Requirements for a Reporting System?,” Bridging the Gap, [Online]. Available: http://www.bridging-the-gap.com/help-a-ba-how-do-you-collect-requirements-for-a-reporting-system/. [Accessed May 2016].

[2] betterevaluation.org, “Reporting Requirements,” http://betterevaluation.org/, [Online]. Available: http://betterevaluation.org/plan/reportandsupportuse/identify_reporting_requirements. [Accessed May 2016].

[3] betterevaluation.org, “Reports And Supports,” http://betterevaluation.org/, [Online]. Available: http://betterevaluation.org/plan/reportandsupportuse/report. [Accessed May 2016].

[4] OSS WATCH, “Essential tools for running a community-led project,” OSS WATCH, 22 February 2011. [Online]. Available: http://oss-watch.ac.uk/resources/communitytools. [Accessed May 2016].

[5] D. Buttner, “Sample Secure Code Review Report,” The MITRE Corporation, Junuary 2014. [Online]. Available: https://www.mitre.org/publications/all/sample-secure-code-review-report. [Accessed May 2016].

[6] Mozilla community, “MOSS - MozillaWiki,” Mozilla community, 22 June 2016. [Online]. Available: https://wiki.mozilla.org/MOSS. [Accessed July 2016].

[7] M. &. G. B. Swanson, “NIST Special Publication 800-14 : Generally Accepted Principles and Practices for Securing Information Technology Systems,” September 1996. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-14/800-14.pdf. [Accessed May 2016].

[8] R. S. Poore, “Generally Accepted System Security Principles,” International Information Security Foundation, June 1999. [Online]. Available: http://www.infosectoday.com/Articles/gassp.pdf. [Accessed May 2016].

[9] European Commission, “About Joinup,” European Commission, [Online]. Available: https://joinup.ec.europa.eu/page/about_us. [Accessed May 2016].

[10] ICASI, “Cvrf 1.1 mindmap – ICASI,” ICASI, [Online]. Available: http://www.icasi.org/cvrf/. [Accessed May 2016].

[11] European Commission, “joinup,” [Online]. Available: https://joinup.ec.europa.eu/.

[12] CERT-EU, “About Us,” CERT-EU, [Online]. Available: http://cert.europa.eu/cert/plainedition/en/cert_about.html. [Accessed May 2016].

[13] ICASI, “The Common Vulnerability Reporting Framework (CVRF) – ICASI,” ICASI, [Online]. Available: http://www.icasi.org/cvrf/. [Accessed May 2016].

[14] OWASP, “Reporting,” OWASP, 01 January 2014. [Online]. Available: https://www.owasp.org/index.php/Reporting. [Accessed May 2016].

[15] OWASP, “OWASP RFP-Criteria - OWASP,” OWASP, 28 March 2016. [Online]. Available: https://www.owasp.org/index.php/OWASP_RFP-Criteria#Reporting_Interface.. [Accessed May 2016].

[16] OWASP, “Testing Guide Introduction: Reportting Requirements,” OWASP, [Online]. Available: https://www.owasp.org/index.php/Testing_Guide_Introduction#Security_Test_Data_Analysis_and_Reporting. [Accessed May 2016].

[17] Atlassian, “Upload Files - Atlassian Documentation,” [Online]. Available: https://confluence.atlassian.com/doc/upload-files-139513.html. [Accessed May 2016].

[18] The MITRE Corporation, “The Importance of Manual Secure Code Review,” The MITRE Corporation, 16 January 2016. [Online]. Available: https://www.mitre.org/capabilities/cybersecurity/overview/cybersecurity-blog/the-importance-of-manual-secure-code-review. [Accessed April 2016].

[19] OWASP, “How to Write an Application Code Review Finding - OWASP,” OWASP, 09 September 2010. [Online]. Available: https://www.owasp.org/index.php/How_to_Write_an_Application_Code_Review_Finding. [Accessed May 2016].

[20] “Confidentiality, Integrity & Availability,” International Security Risk Management Consortium, 2009. [Online]. Available: http://ishandbook.bsewall.com/risk/Methodology/CIA.html. [Accessed 2016].

[21] Linux Foundation, “Core Infrastructure Initiative,” Linux Foundation, [Online]. Available: https://www.coreinfrastructure.org/.

9 ANNEXES

9.1. ANNEX 1: CVRF v1.1 Mind Map

Figure 4. Mind Map of CVRF v1.1 [9]

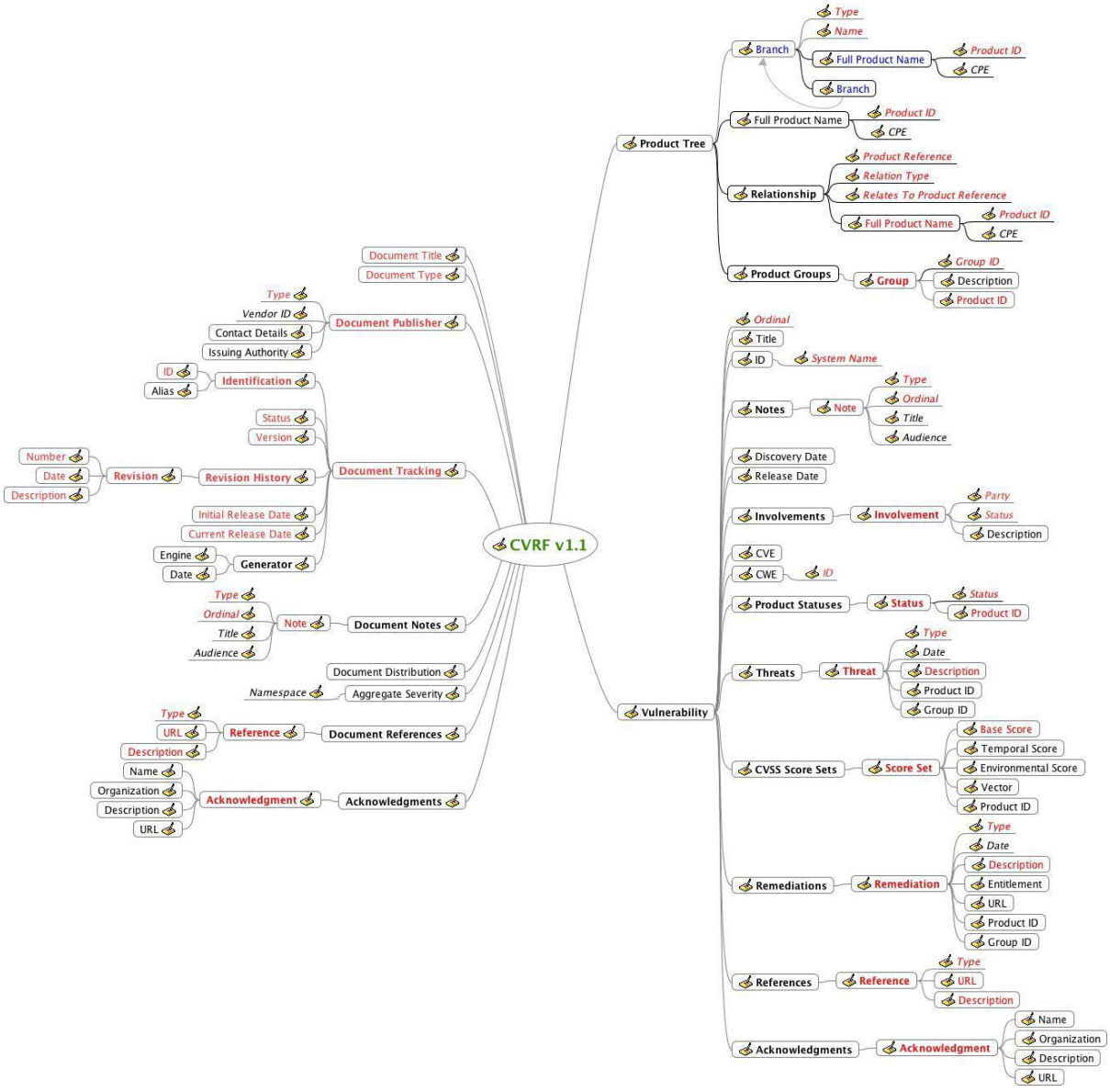


Table 4: Footnotes of Mind Map of CVRF v1.1

Node Type	Meaning
Bubble	Element
Underline	Attribute

Font Colour	Meaning
Red	Required
Black	Normal
Blue	Choice

Font Type	Meaning
Normal	Normal
Bold	Container