Bulk Issue Creator (BIC)

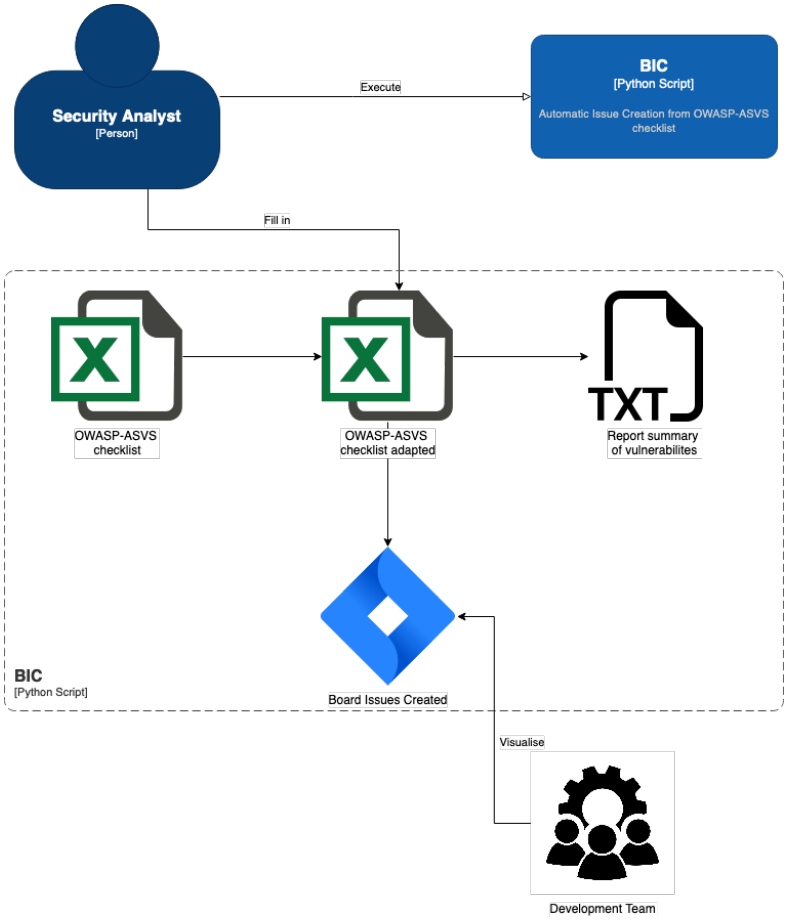


Fig 1 - BIC architecture breakdown

Overview:

- **Goal**: This tool is designed to simplify and automate the process of reporting security flaws detected, using the OWASP-ASVS checklist, to JIRA. It allows the security experts to automatically generate issues from the checklist to perform audits and tests of security controls in a web application.
- **Stakeholders:** Security Analysts, Security Experts, Security Teams, Security Operations Center (SOC), Project Managers and Product Owners.

1. Introduction

One of the many tasks of a Security Analyst is to analyze the code and detect poor implementation of security controls such as those represented in the OWASP Top Ten.

There are several ways to analyze the correct implementation of these security controls. There is an automatic approach in which applications are used that provide final reports

on the overall security posture of an application. These reports point to security flaws and present possible ways to exploit the application, as well as the solution that should be implemented to stop them.

There is also a manual one, as in this case. In this case, the security analyst follows a checklist that can be found in different file formats (such as Excel or OpenDocument). This checklist contains good practices/standards that an application should follow and the Security Analyst should check (manually) those that are validated and those that are not.

In the case of manual validation, after obtaining the results it is necessary to go to JIRA and enter each issue manually. A process that is time-consuming, repetitive and a potential source of errors (human factor).

Jira has a feature that allows Security Analysts to import multiple issues at the same time. Using a CSV file (.csv) with a certain structure (with the fields “Summary”, “Description” and “Labels”) it is possible to report in Bulk. Despite this feature, importing the .csv file still has to be done manually.

It was in this context that the BIC tool was developed. Now Security Analysts can report these issues to Jira iin a simpler and more automatic way.

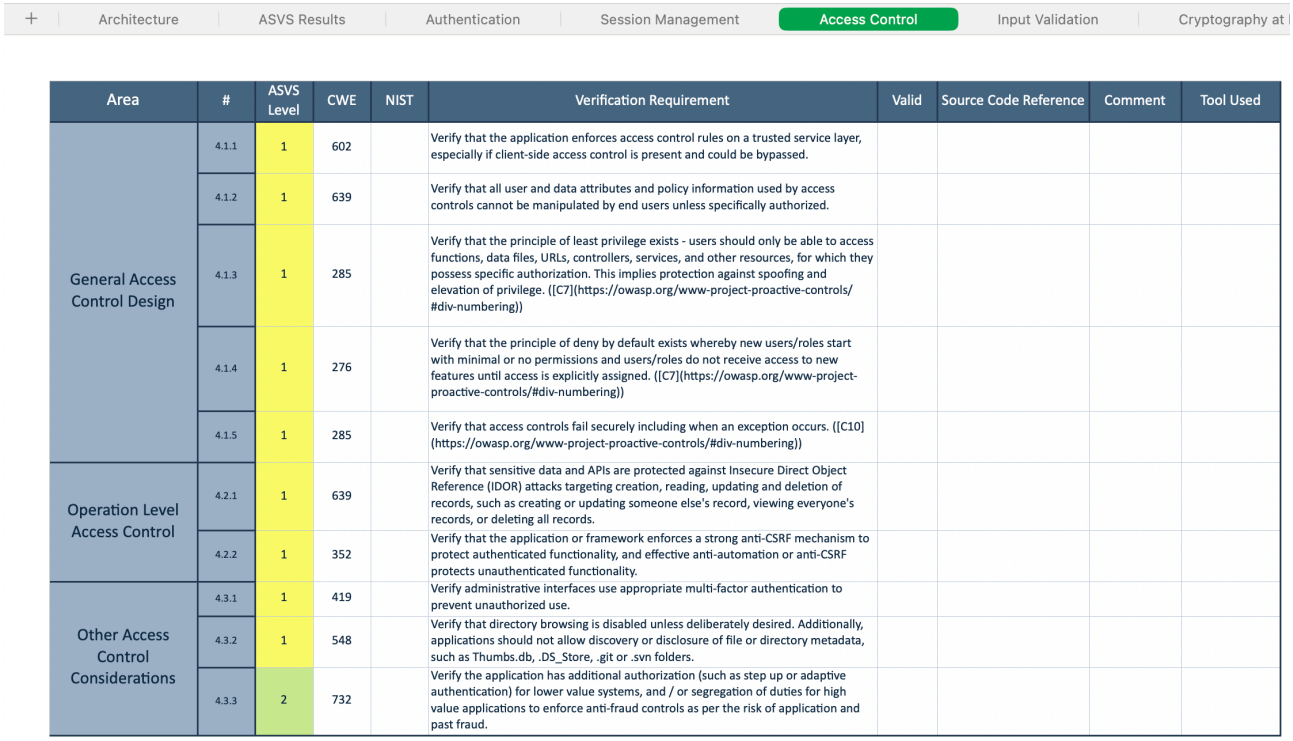
2. Excel Layout

The Excel file is divided into several sheets. Each sheet represents a topic. To download the Excel file that is being discussed, see [1].

These sheets all have the same structure. They are organised by 10 columns, in which:

- Area: Sections of the respective chapter (selected sheet)
- #: Identification number of each requirement
- ASVS Level: Requirement ASVS Level [1, 3]
- CWE (Common Weakness Enumeration) - a community-developed list of software and hardware weakness types. The number represented in this column is the ID of each requirement in that list.
- NIST: Starting with version 4.0 of owasp-asvs, the “Authentication” and “Session Management” chapters (sheets) conform to NIST Special Publication 800-63 [2]. Where each value corresponds to the standard listed in this publication.
- Verification Requirement: Requirement description
- Valid: Column intended to assist whether or not the security control of the Web application is well implemented. Values can be {Valid, Non-Valid, Not Applicable}
- Source Code Reference: Where the user can place a note where in the code each security control is or is not implemented
- Comment: Column that can be used to place any auxiliary comments

- Tool Used: Software/Procedure that was used to verify the implementation of security control (e.g. Burp Suite, Developer Tools — in the browser, code analysis, etc…)



Area	#	ASVS Level	CWE	NIST	Verification Requirement	Valid	Source Code Reference	Comment	Tool Used
General Access Control Design	4.1.1	1	602		Verify that the application enforces access control rules on a trusted service layer, especially if client-side access control is present and could be bypassed.				
	4.1.2	1	639		Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.				
	4.1.3	1	285		Verify that the principle of least privilege exists - users should only be able to access functions, data files, URLs, controllers, services, and other resources, for which they possess specific authorization. This implies protection against spoofing and elevation of privilege. ([C7](https://owasp.org/www-project-proactive-controls/#div-numbering))				
	4.1.4	1	276		Verify that the principle of deny by default exists whereby new users/roles start with minimal or no permissions and users/roles do not receive access to new features until access is explicitly assigned. ([C7](https://owasp.org/www-project-proactive-controls/#div-numbering))				
	4.1.5	1	285		Verify that access controls fail securely including when an exception occurs. ([C10](https://owasp.org/www-project-proactive-controls/#div-numbering))				
Operation Level Access Control	4.2.1	1	639		Verify that sensitive data and APIs are protected against Insecure Direct Object Reference (IDOR) attacks targeting creation, reading, updating and deletion of records, such as creating or updating someone else's record, viewing everyone's records, or deleting all records.				
	4.2.2	1	352		Verify that the application or framework enforces a strong anti-CSRF mechanism to protect authenticated functionality, and effective anti-automation or anti-CSRF protects unauthenticated functionality.				
Other Access Control Considerations	4.3.1	1	419		Verify administrative interfaces use appropriate multi-factor authentication to prevent unauthorized use.				
	4.3.2	1	548		Verify that directory browsing is disabled unless deliberately desired. Additionally, applications should not allow discovery or disclosure of file or directory metadata, such as Thumbs.db, .DS_Store, .git or .svn folders.				
	4.3.3	2	732		Verify the application has additional authorization (such as step up or adaptive authentication) for lower value systems, and / or segregation of duties for high value applications to enforce anti-fraud controls as per the risk of application and past fraud.				

Fig 2 - Original excel structure

It was necessary to adapt the Excel structure, with the aim of collecting new properties of the issues to be reported.

Firstly, distinguish between the security controls that are not implemented, those that need to be reported. As well as the issue type, to be able to establish their priority.

2.1 New Layout

The new Excel structure is very similar to the one mentioned above as it shows the next Figure 3.

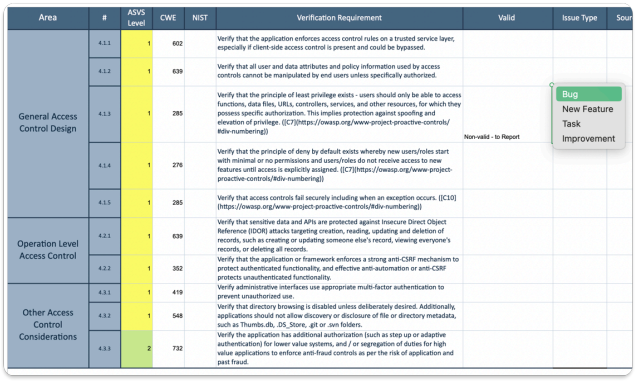


Fig 3 - Excel adapted Structure

The only changes are in the possible values of the "Valid" column. Now the values can be {Valid, Not Applicable, Non-valid - to Report, Non-valid - not for Reporting}.

This change was made to be able to distinguish between security controls that are not validated, which ones should or should not be reported. The issues that will be reported are only those with the value "Non-valid - to Report" in the "Valid" column. Those with any other option (including empty) are not reported.

The other change was the creation of an "Issue Type" column, where the user can define the type of issue that will be reported to JIRA. The values that are defined by default are: {Bug, New Feature, Task, Improvement}.

If an issue is reported without its type being defined in this column, by default it will be of the "Task" type.

3. Jira Project Configuration

When a project is created in Jira, it is necessary to make some changes to its settings for the application to work properly.

By default, when a Blank Project is created, the only issue type that exists is “Task”. It is then necessary to create the missing issue types:

- Bug
- Improvement
- New Feature

The following Figures 4 to 8 demonstrate an example of the “New Feature” creation process.

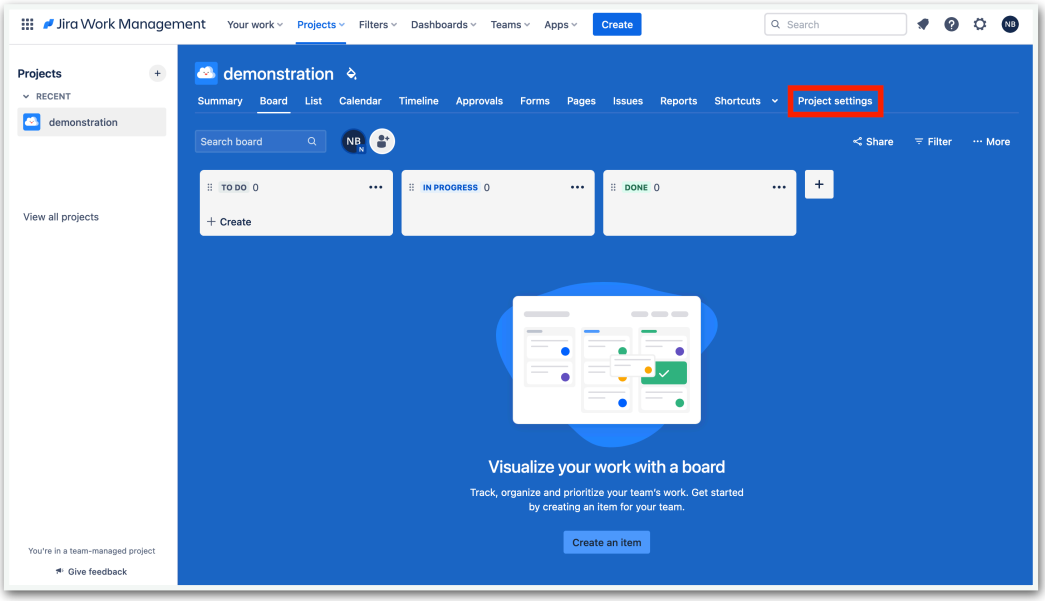


Fig 4 - Project Settings

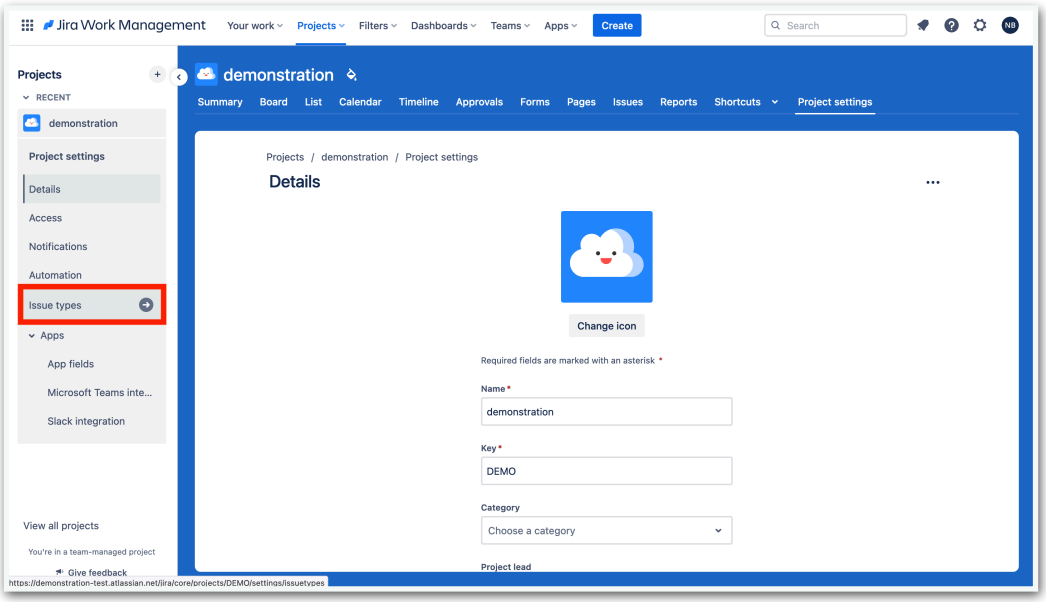


Fig 5 - Issue Types

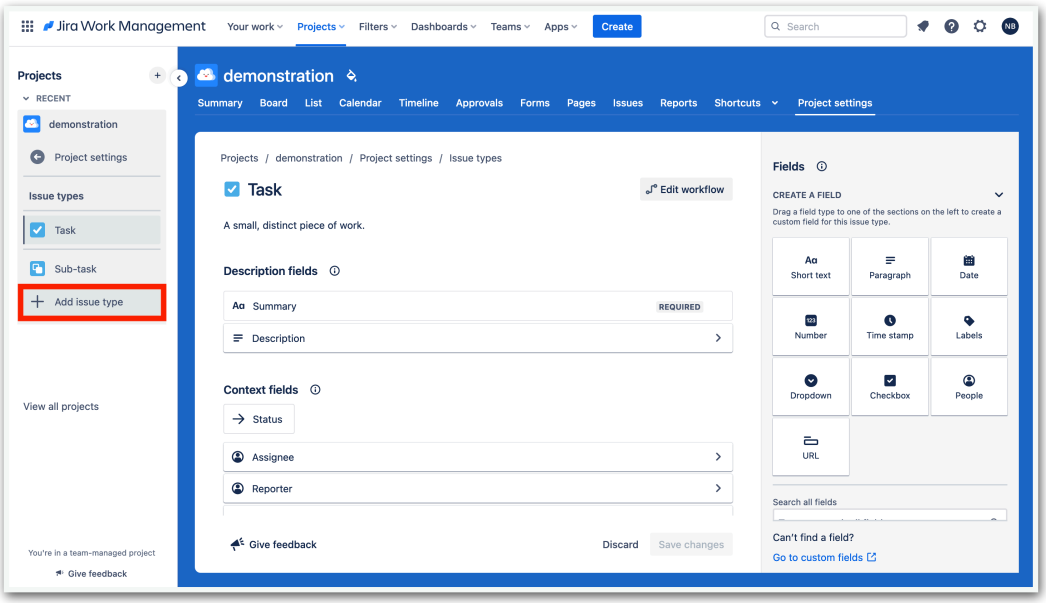


Fig 6 - Add Issue Type

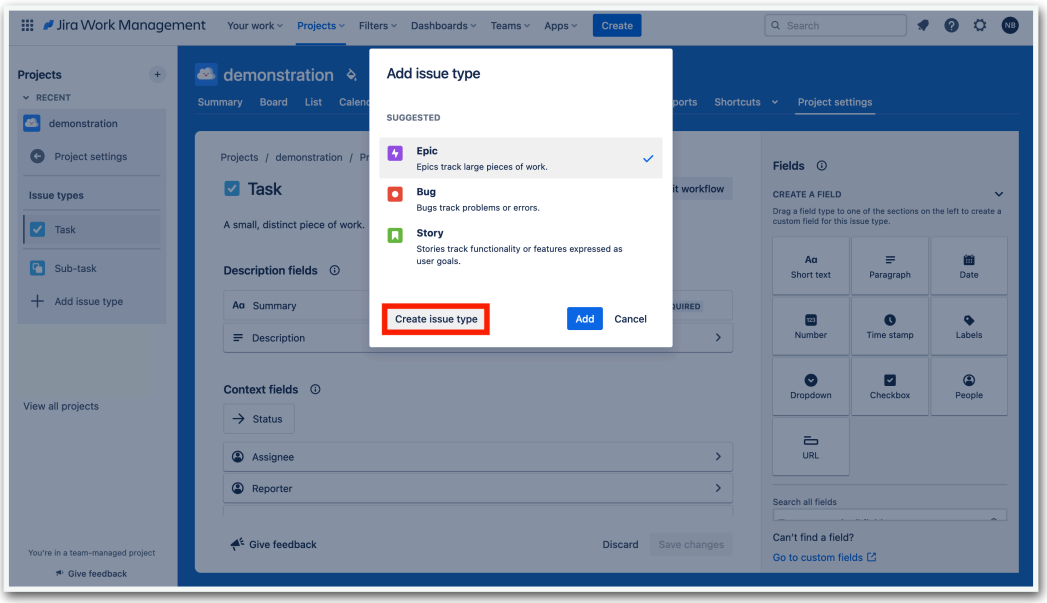


Fig 7 - Create Issue Type

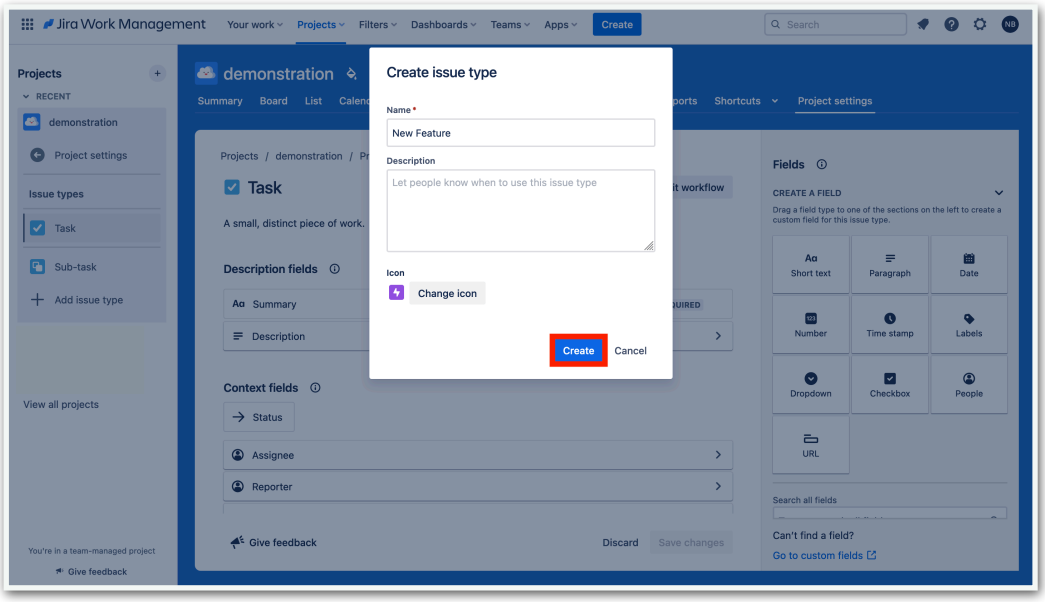


Fig 8 - Create new Issue Type

Once all the issue types mentioned above have been created, it is also necessary to add the “Priority” field to each one. The following Figures 9 to 11 illustrate this drag-and-drop process for the “New Feature” issue type.

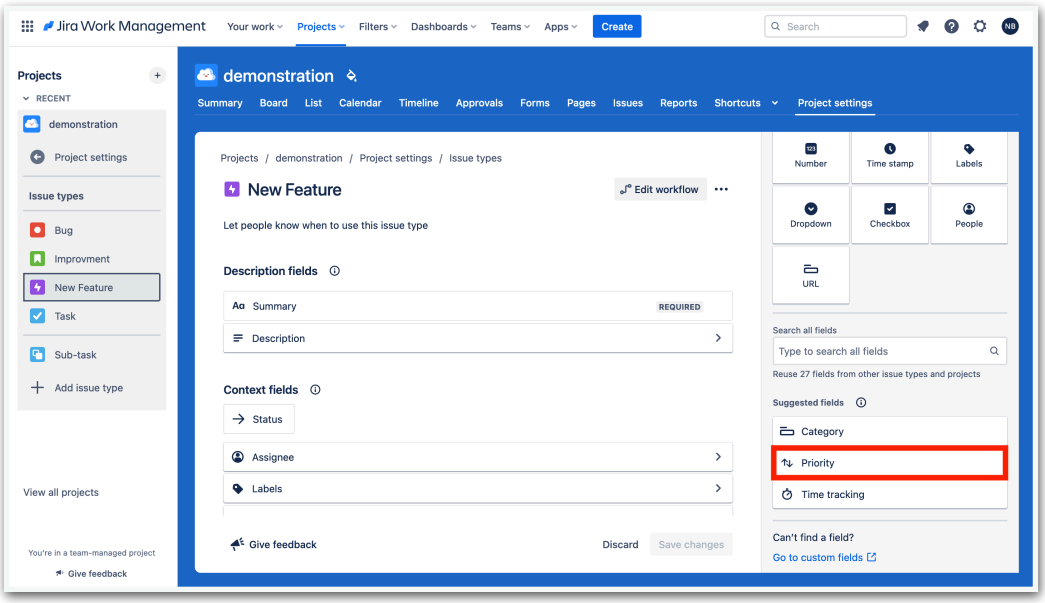


Fig 9 - Initial state of Issue Type structure

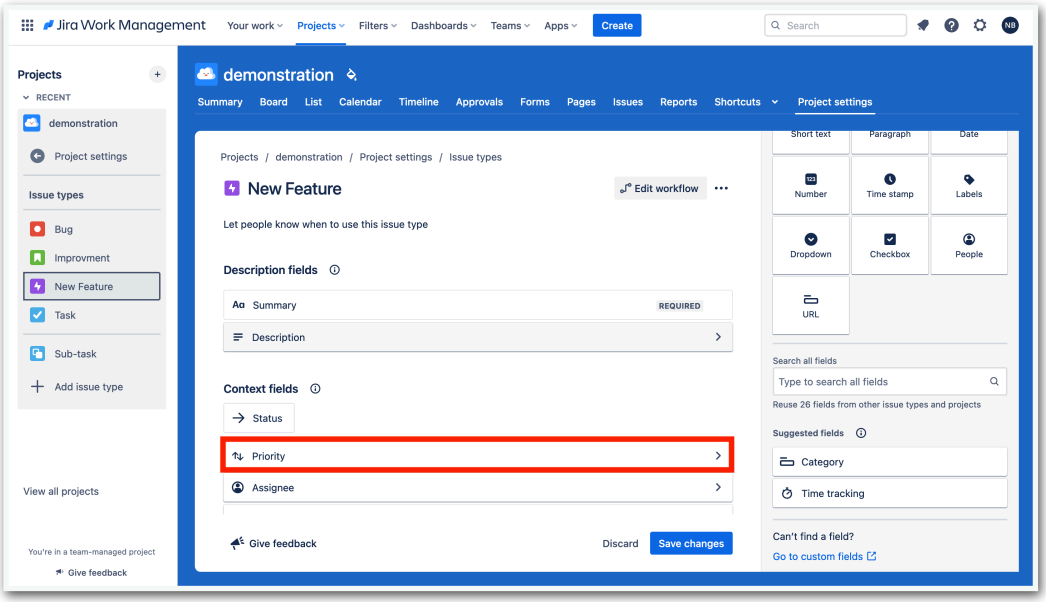


Fig 10 - Add “Priority” field to the structure

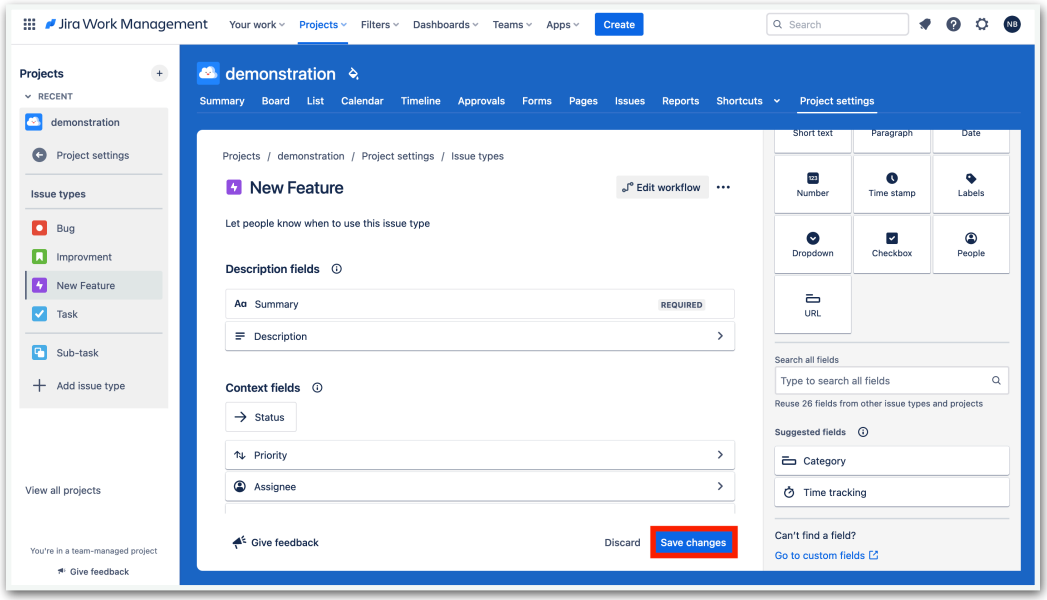


Fig 11 - Save changes

After carrying out this process for all Issue Types (“Task” comes by default), the configuration of the project in Jira is complete.

4. BIC Usage

4.1. Installation and Dependencies

To install the BIC tool you can clone the online GitHub repository, by executing the next command:

```
git clone URL__XPTO
```

In the repository there is also a .txt file (“requirements.txt”). This file is for the external libraries installation purposes.

4.2. Dependencies

When developing this tool, it was necessary to install external libraries. In this section, they and their versions will be listed.

First of all, is necessary to have “python”. It can be done using “brew” (MacOS). Run the following command:

```
brew install python3
```

Then, the pip3 is installed automatically.

The external libraries installed are listed in the next table (Table 1):

LIBRARY	VERSIONS
jira	3.5.2
openpyxl	3.1.2
requests	2.31.0

Table 1 - External libraries

To install them, use:

```
pip3 install -r requirements.txt
```

4.3. Project Based Configuration

To use this tool, there are some environment variables that have to be configured (in the BIC.py file) according to the project being used.

In the downloaded file at the beginning, after the “imports” section, the values that need to be changed are:

- username: e-mail of the Jira account
- base_url: Url of the project in Jira
- api_key: The API token of the project
- project_name: Name of the project in Jira
- project_key: Project key of the project

4.4. Usage

The correct syntax for running the tool is:

```
python3 BIC.py <path_to_excel> <name_of_report>
```

- BIC.py: name of the script
- <path_to_excel>: Path to the owasp-asvs excel file
- <name_of_report>: name of the txt file that summarizes the security controls that are not valid

It is mandatory to execute the script first before starting to fill in the excel. This is an essential step, as it is in this first execution that Excel will adapt its structure to the one required by the tool to work properly.

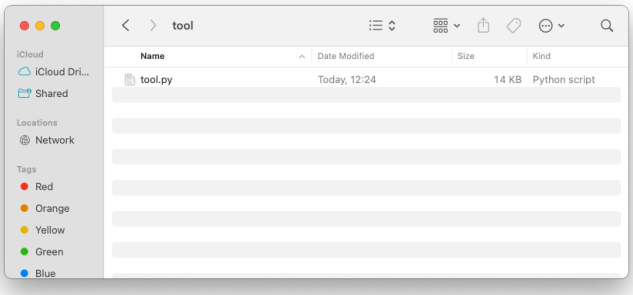


Fig 4 - Before running the tool

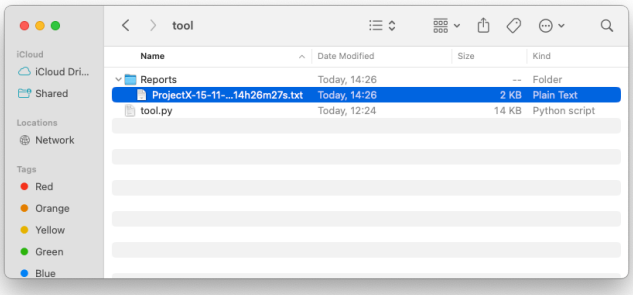


Fig 5 - After running the tool

The report mentioned in the second argument is saved in a directory “Reports”, in the directory where de script is, as it show in the next figures.

5. References

[1] https://github.com/shenril/owasp-asvs-checklist/raw/master/ASVS-checklist-en.xlsx

[2] https://pages.nist.gov/800-63-3/