Jira-Security-Issue-Classifier Plugin (JiraSecPlugin) Installation Guide V1.1

I. Introduction

JiraSecPlugin is a simple-to-use plugin for classifying Jira issue as security related or not.

JiraSecPlugin now comes with an integrated machine learner model. Further, we have validated the plugin model's algorithm using rigorous empirical analysis and comparing the performances of using different machine learning techniques and simple string pattern matches.
A classifier can either be a machine learning model or a string pattern match function.
Our empirical results show that machine learners trained with project's vocabularies in addition to generic security keywords produce powerful and impressive classification models when compared to using only generic security keywords.

Our approach makes the plugin quite attractive and provides a basis for many use cases beyond the Jira platform. The plugin reuses a core classifier library that we have built and can be integrated into different development platforms with similar needs.

We created 4 categories of terms to form the features/attributes that can be used to train a classifier. Some security vocabularies may vary across projects and organizations. These are mostly reflected in the "Asset", "Control", and "Indirect" terms. Security terms are thus divided into these four categories:

1. Assets or Personally Identifiable Information (PII),
2. Direct (terms related to attacks and vulnerabilities),
3. Control (terms related to implemented security controls), and
4. Indirect (terms that are indirectly related to security and not in the above 3 categories).

It is possible to use these categories to determine the ranking (Importance) of the classifier result. This is done by assigning different weights to each category.

We have extracted a set of terms from different sources such as the CWE, OWASP, CVE, RFC 4949, and industrial issue tracking databases. However, to improve your own classification model, you would need to augment these terms with your project specific vocabularies (assets, control, and indirect) terms and train a new classification model. Check the "jira-security-plugin-machine-learning-guide-v1.1" to see how to train a classification model for your model.

II. Pre-Installation

1. Create Custom Fields

The plugin uses 3 custom fields (*Security Issue, Security Importance, and Security Description*). These fields must be created before the plugin is installed. Please note the space between the words (e.g. Security Issue and not SecurityIssue). The fields must be correctly defined otherwise you will not see any result. You can reconfigure these fields as you like in the "security-issue-classifier.properties" file. See **Configuration Files** section.

A. Security Issue

Variables

Name	Description	Type	Options
Security Issue	Custom field that	Select List (Single	YES

	shows whether a recorded or updated issue is security related or not	Choice)	NO DON'T KNOW

B. Security Importance

Variables and Description

Name	Description	Type	Options
Security Importance	Custom field that displays computed importance of an issue classified as security related	Select List (Single Choice)	HIGH MEDIUM LOW NONE

C. Security Description

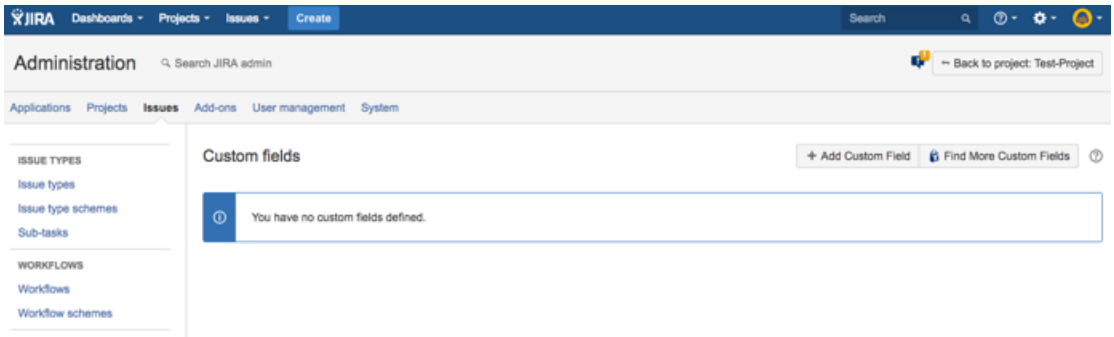
Variable and Description

Name	Description	Type	Options
Security Description	Custom field that displays customized message as feedback about the classification and terms used for the decision	Text Field (multi-line)	-

2. Steps to create custom fields in JIRA

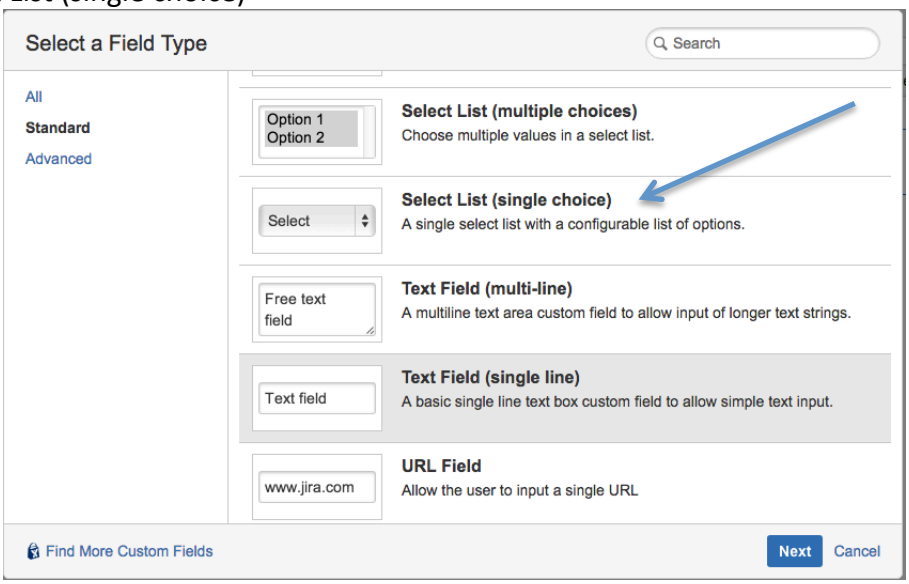
A. Add Custom Field

Click on "Add Custom Field" button



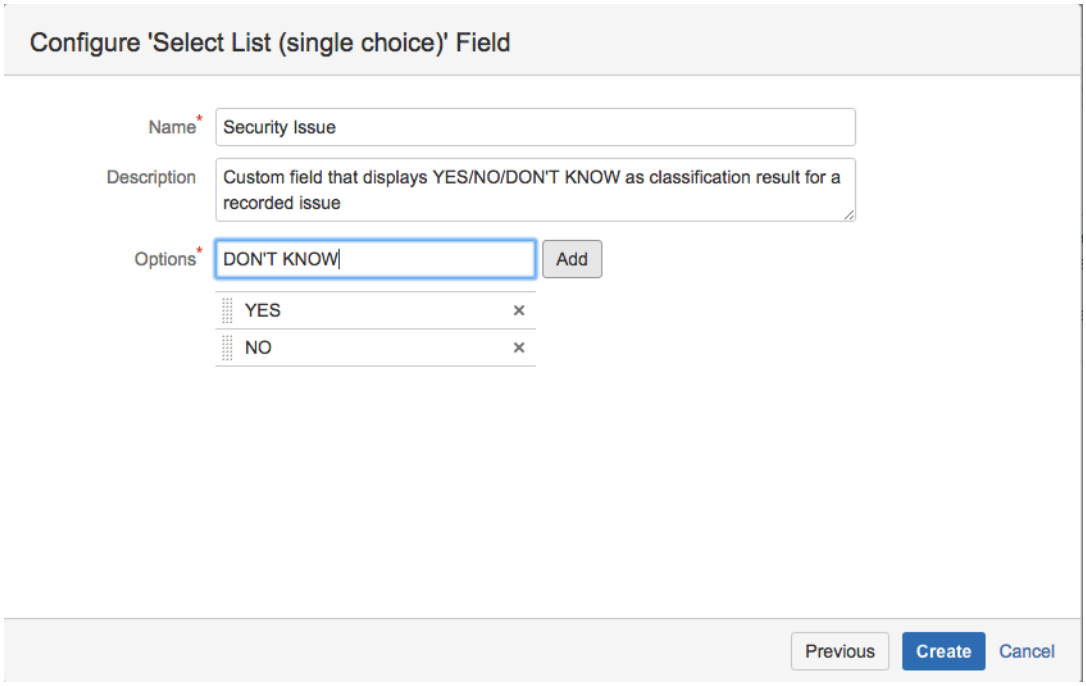
B. Select a Field Type

Select the "Select List (single choice)



C. Configure Security Issue custom field

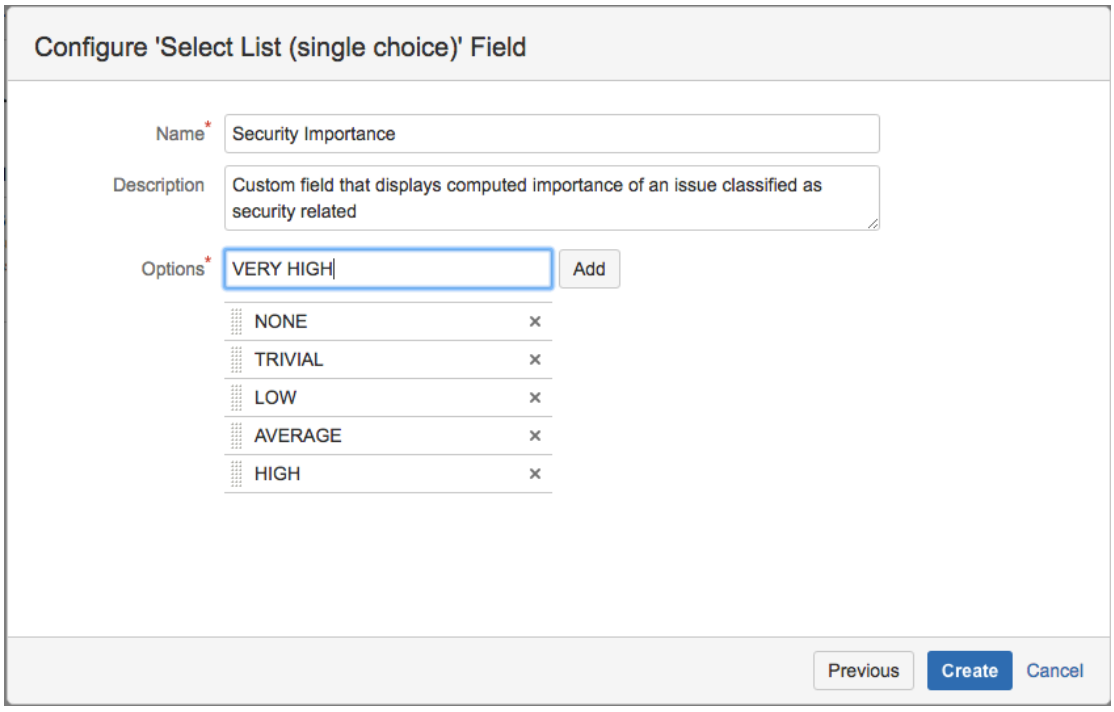
Fill all the fields as shown below.



D. Configure Security Importance custom field

Repeat the step in Sections A and B.

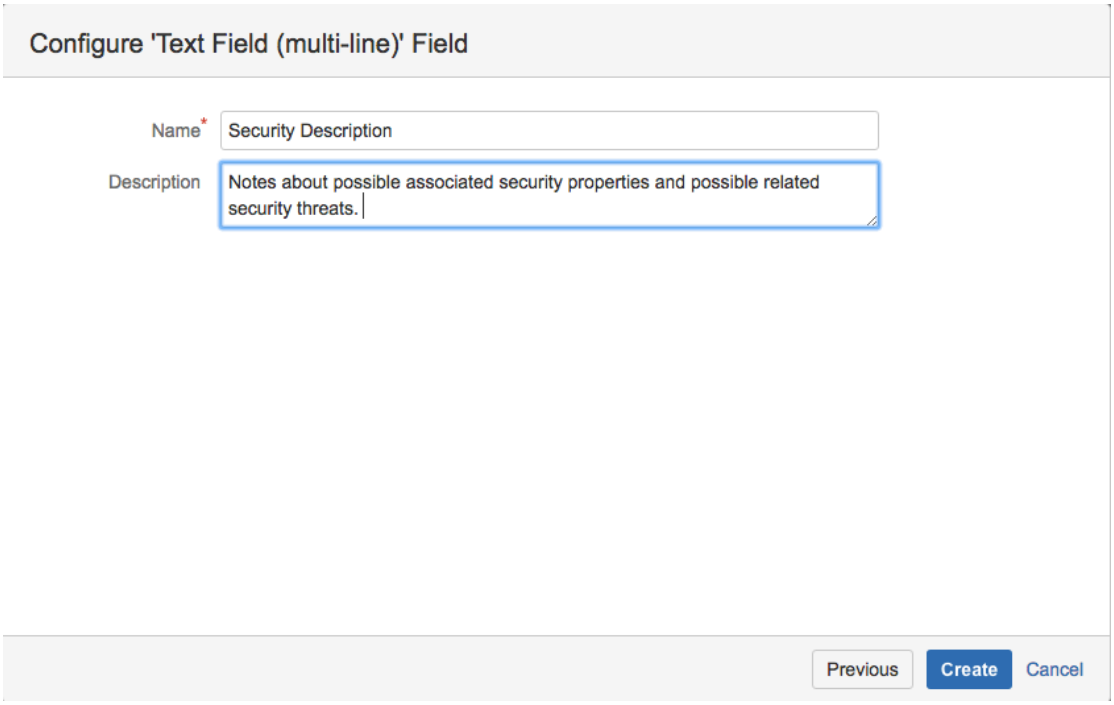
Fill the fields as shown below



E. Configure Security Description custom field

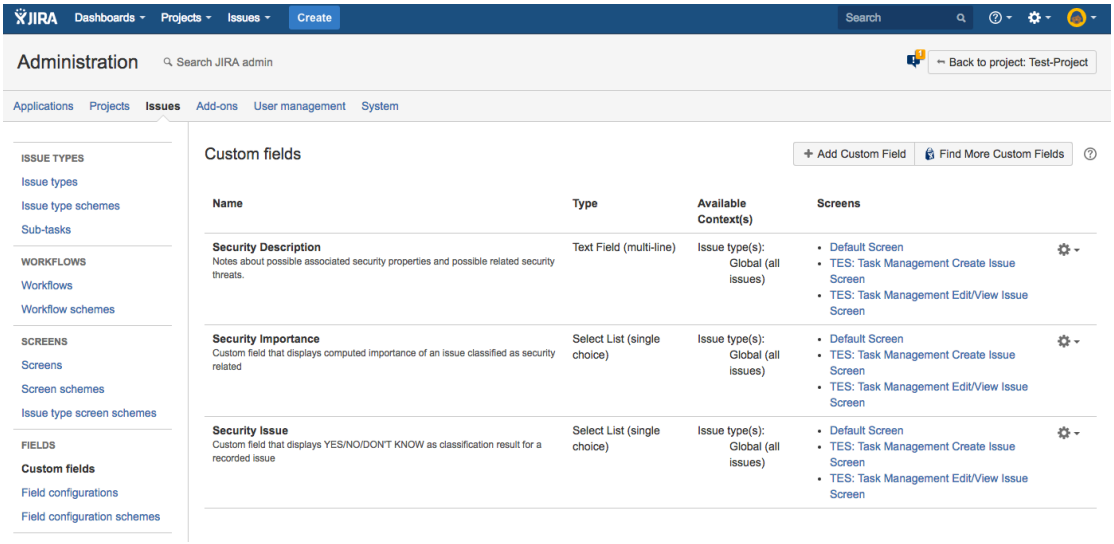
Repeat the step in Figure 2.

Fill the fields as shown in Figure 5.



F. Associate custom fields with Screens

You need to associate the custom fields to all relevant screens.



III. Obtain and Install plugin jar

1. Download plugin jar directly from BitBucket

URL: https://bitbucket.org/ootos/jirasecplugin/downloads/jirasecplugin-1.1.jar

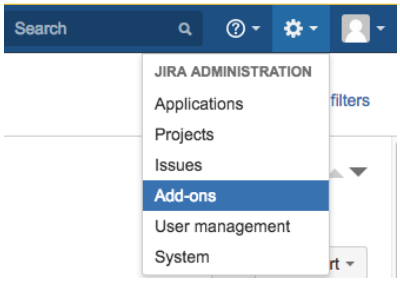
Verify the authenticity and integrity of the jar file using the SHA256 Checksum.

A. Use the Universal Plugin Manager

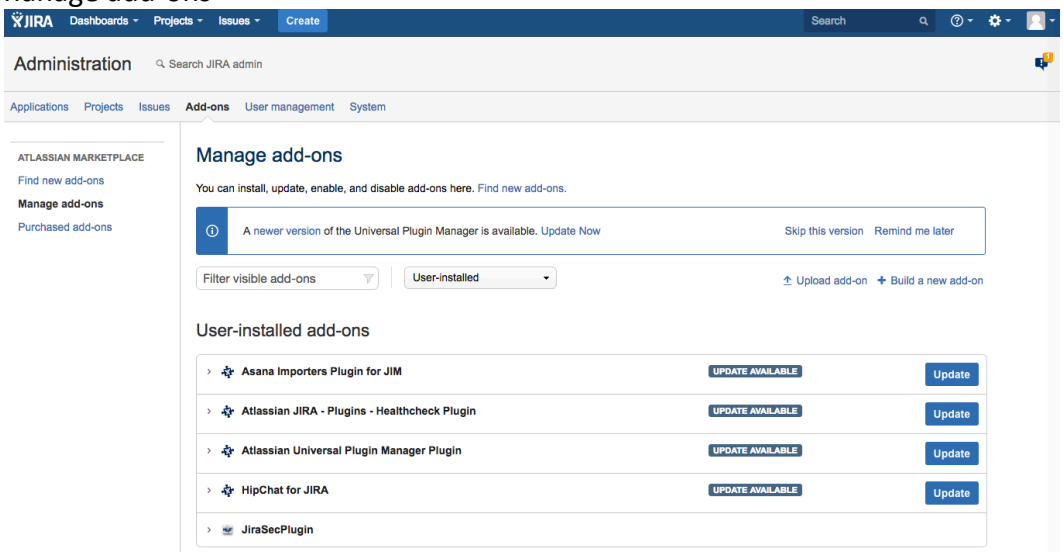
(https://confluence.atlassian.com/display/UPM/About+the+Universal+Plugin+Manager)

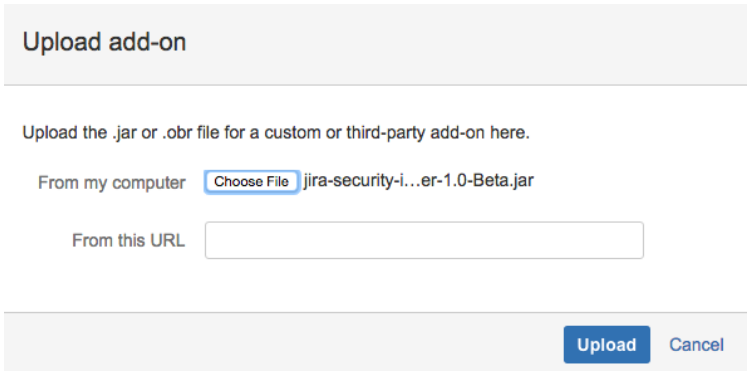
From the top navigation bar, click:


> **Add-ons** to access the UPM.



Select Manage add-ons





B. Manual addition

Navigate to $JIRA_HOME/plugins/installed-plugins folder on your Jira server
Place the JiraSecPlugin in the folder
Restart your server

2. Install directly from JIRA Atlassian Market

URL: https://marketplace.atlassian.com/
Plugin Name: JiraSecPlugin

IV. Configuration Files

1. Description of configuration files

	File	Description

1	controlterms.prop	Contains security keywords that are associated with implemented security controls (e.g. ACL, HTTPS, IPSec, Log-server, encryption)
2	directterms.prop	Contains security keywords that are associated with attacks and existing malwares (e.g. xss, intruder, buffer overflow, intrusion, deface, heartbleed, etc.)
3	indirectterms.prop	Consists of keywords that could be indirectly associated with security (e.g. stacktrace, disk space, null pointer, etc.)
4	piiterms.prop	The list consists of personally identifiable information terms (pin, credit card, password, token, phi, etc.)
5	separators.prop	Summary and description often contain special characters that could affect tokenization (e.g. ,/@;:.). Such characters can be included in the list. The program removes listed characters from the text before tokenization.
6	stopwords.prop	Contains stopwords that should be removed before tokenizing the text such as "as, a, to, be, etc". The list can be expanded to include more stopwords
7	message.prop	Custom messages for each term category or combination of categories
8	importance.prop	Defined scale to categorize the classified issue (see section below for details)
9	security-issue-classifier.properties	Property file for making configuration changes (see the section below for details)
10	model.cls	Default Random Forest model trained on the set of terms (pii, direct, control, indirect). To improve your model, you need to train on your own organization's/project's dataset and replace the default model in JiraSecPlugin folder
11	modeleval.cls	Default performance results of model.cls

2. security-issue-classifier.properties

learner_or_term_search=ml	choose whether to use machine learning classifier (ml) or only keyword/term search (ts) for classification
method=3	Choose between method 1, 2 or 3. Method 1 uses LevenshteinDistance algorithm for term with length > term_min_len. This is the number of changes needed to change one String into another, where each change is a single character modification (deletion, insertion or substitution). #Method 2 uses JaroWinklerDistance algorithm for term with length > term_min_len. The Jaro measure is the weighted sum of percentage of matched characters from each file and transposed characters. Winkler increased this measure for matching initial characters. #Method 3 uses full string search for term with length <= term_min_len and substring search for term with length > term_min_len
term_min_len=4	#Assumption-the shorter the length of term the lower the probability of making typo or other errors
levenshtein_threshold=1 jarowinkler_threshold=0.95	#Set Threshold for determining accepted terms with LevenshteinDistance/jarowinkler algorithm: has the benefit of absorbing human typo errors in the commit desc/summary
override_user_changes=false	# reclassify (i.e.use model's result) and overrides user's settings each time description or summary is changed or comments are added or edited. (true or false)
piimax=4 directmax=1 controlmax=5 indirectmax=5	#security terms and maximum count for determining score. It is used to adjust the weight for each category. A full weight is obtained when the number of terms equals or exceed the maximum set. You can use this to downgrade or upgrade scores – SecurityImportance. In addition, it can be used to turn-off/disable a category by setting the value to 0.
wdirect=0.4 wpii=0.4 wcontrol=0.1 windirect=0.1	#weights for direct, pii, control and indirect terms. This provides the relative importance of the terms for each category. The sum of the four weights must always be 1, otherwise default hardcoded values will be used.
option_yes = YES option_no = NO option_dk = DON'T KNOW	#security issue labels – Values defined must match the labels in JIRA
sec_issue=Security Issue sec_importance=Security Importance sec_desc=Security Description	#Custom fields are defined here. Values defined must match the labels in JIRA sec_issue and sec_importance are mandatory and must have values. To disable sec_desc, set the value to empty (e.g. sec_desc=)

3. importance.prop

0=NONE 1=LOW 2=MEDIUM 3=HIGH	#Importance levels - options. This scale can be changed as needed (e.g. 0 – 5 with their own labels). However, the labels defined in the jira custom field (Security Importance) must be changed as well

V. Updating configuration files

Overtime, you might have needs to update the configuration files. For examples, you may wish to add/remove some terms from the keyword list, or you may want to change the algorithm parameters. Locate the "JiraSecPlugin" folder in the Jira home directory directory (Note that this folder is protected

and can only be written to by the owner/administrator). Open the configuration file you would want to change, update the values, save and close it. Disable and Enable the plugin from the “Add-on -> Manage add-ons” to activate your new changes.
To train a new classification model, please refer to the jira-security-plugin-machine-learning-guide-v1.1.

VI. Extracting user modified logs

It is possible for user to modify the plugin-classified issue: For instance, users can change the value for Security Issue if it is deemed to be false positive. Similarly, user can downgrade Security Importance. These changes are stored in a log file “usermodify-jira.log” and can be downloaded to train and improve the classifier in the future. An example is to look for common patterns (terms) and put them in the relevant term categories. A new model can then be trained for future usage.