[2019-04-26] Change Log

New version of Kiuwan

A ***new version of Kiuwan*** has been released.

Major changes are:

- ***Support for Scala programming language***
- ***Support for SAML Single Sign-On (SSO)***
- ***Vulnerabilities and License Management in Kiuwan Insights***
- ***Insights support for Ruby***
- ***Kiuwan for Developers plugin for VisualStudio Code***

CQM is the default Model (i.e. a concrete set of active and pre-configured rules):

- If you are using ***CQM***,
  - ***new rules will automatically become active*** and will be applied to new analyses
- If you are using your own **custom model, your model remains unchaged,** but *you can modify it and activate the new rules* (in case you want to be applied to your code).

You can find new rules by comparing this release of CQM against previous version. A detailed description of the behavior of these new rules is available in rule's description.

A ***new version of Kiuwan Engine*** has been released that incorporates ***bug fixes, performance and reliability improvements in rules and parsers***.

Kiuwan Engine is the binary code executed when an analysis is run.

- ***If the engine is not blocked*** in your Kiuwan account, ***the engine will upgrade automatically*** to the last version of Kiuwan Engine once a new analysis is run
- ***If the engine is blocked***, your kiuwan ***engine will not be modified***.

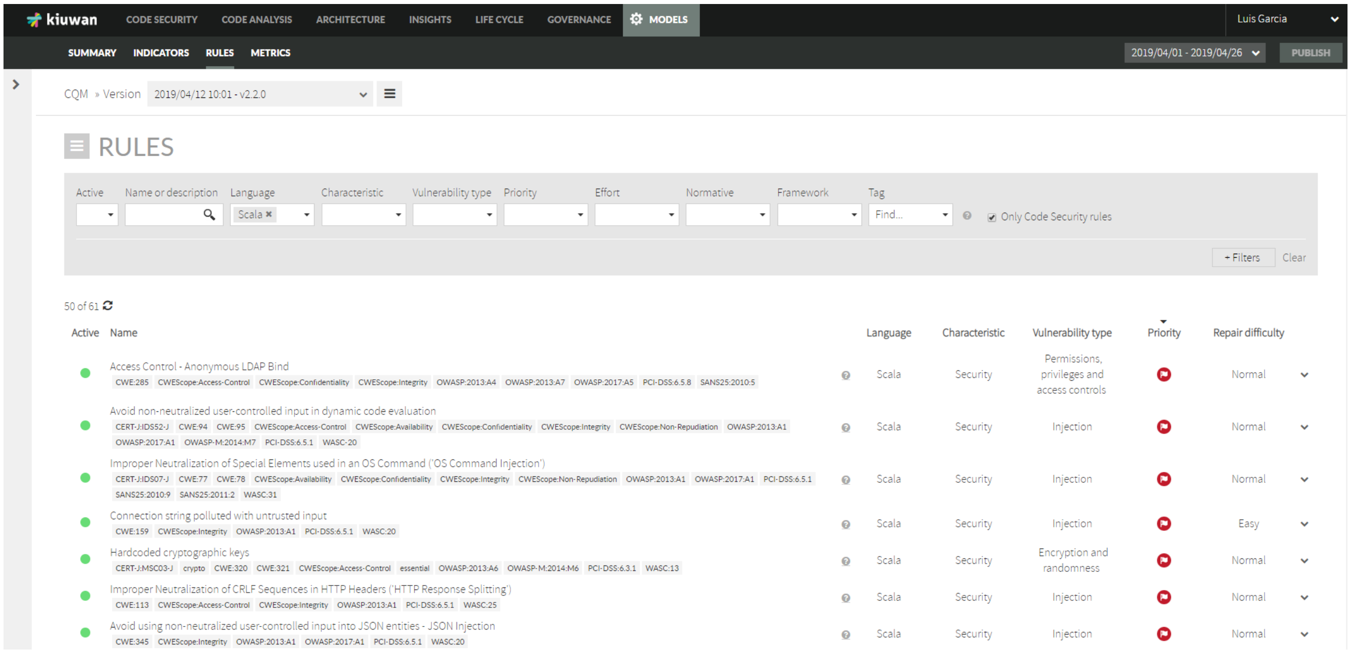
Support for Scala programming language

Scala is a a general-purpose programming language providing support for functional programming and a strong static type system.

As Scala is becoming a widely adopted programming language, ***Kiuwan*** now incorporates ***support to analyze Scala source files***, thus searching for code and desing conditions that are indicative of ***security vulnerabilities***.

Kiuwan provides ***61 Security Rules*** specifically suited to ***Scala*** programming language.

You can find these rules going to ***Models Management***, select ***CQM*** and search for ***Rules*** applying to ***Scala language***



Version: 2019/04/12 10:01 - v2.2.0

Active · Name or description · Language: Scala · Characteristic · Vulnerability type · Priority · Effort · Normative · Framework · Tag: Find...

☑ Only Code Security rules

50 of 61

Active	Name	Language	Characteristic	Vulnerability type	Priority	Repair difficulty
●	Access Control - Anonymous LDAP Bind CWE:285 CWEScope:Access-Control CWEScope:Confidentiality CWEScope:Integrity OWASP:2013:A4 OWASP:2013:A7 OWASP:2017:A5 PCI-DSS:6.5.8 SANS25:2010:5	Scala	Security	Permissions, privileges and access controls	●	Normal
●	Avoid non-neutralized user-controlled input in dynamic code evaluation CERT-J:IDS52-J CWE:94 CWE:95 CWEScope:Access-Control CWEScope:Availability CWEScope:Confidentiality CWEScope:Integrity CWEScope:Non-Repudiation OWASP:2013:A1 OWASP:2017:A1 OWASP-M:2014:M7 PCI-DSS:6.5.1 WASC-20	Scala	Security	Injection	●	Normal
●	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') CERT-J:IDS07-J CWE:77 CWE:78 CWEScope:Availability CWEScope:Confidentiality CWEScope:Integrity CWEScope:Non-Repudiation OWASP:2013:A1 OWASP:2017:A1 PCI-DSS:6.5.1 SANS25:2010:9 SANS25:2011:2 WASC:31	Scala	Security	Injection	●	Normal
●	Connection string polluted with untrusted input CWE:159 CWEScope:Integrity OWASP:2013:A1 PCI-DSS:6.5.1 WASC:20	Scala	Security	Injection	●	Easy
●	Hardcoded cryptographic keys CERT-J:MSC03-J crypto CWE:320 CWE:321 CWEScope:Access-Control essential OWASP:2013:A6 OWASP-M:2014:M6 PCI-DSS:6.3.1 WASC:13	Scala	Security	Encryption and randomness	●	Normal
●	Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting') CWE:113 CWEScope:Access-Control CWEScope:Integrity OWASP:2013:A1 PCI-DSS:6.5.1 WASC:25	Scala	Security	Injection	●	Normal
●	Avoid using non-neutralized user-controlled input into JSON entities - JSON Injection CWE:345 CWEScope:Integrity OWASP:2013:A1 OWASP:2017:A1 PCI-DSS:6.5.1 WASC:20	Scala	Security	Injection	●	Normal

Support for SAML Single Sign-On (SSO)

Since April 2019 release, ***Kiuwan allows you to login in a Single Sign-One (SSO) environment***.

By implementing SSO, a user is able to log in to different independent systems through the use of a single set of credentials, centrally managed in a repository.

SSO can be implemented through different protocols, being ***SAML*** the most widely used.

Kiuwan provides support for SAML SSO, thus allowing you to integrate Kiuwan with most corporate users' credentials repositories (*Active Directory FS*, *Azure AD*, *CA Single Sing-On* ,etc).

> ***Kiuwan provides support for SAML SSO***, thus allowing you to integrate Kiuwan with most corporate users' credentials repositories (*Active Directory FS*, *Azure AD*, *CA Single Sing-On* ,etc).
>
> You can read more at How to integrate Kiuwan with SAML SSO

Vulnerabilities and Licenses Policy Management in Kiuwan Insights

When searching for ***Vulnerabilities on open source code*** used by your application, you can now define your custom policy. Thus, you can decide now if raising alerts on specific vulnerabilities and conditions.

Same way, when Kiuwan Insights searches for ***License risks***, you can now fully customize your License policy and adapt Insights' finding to your organization,

You can find further info at Vulnerabilities Management and Licenses Policies Management

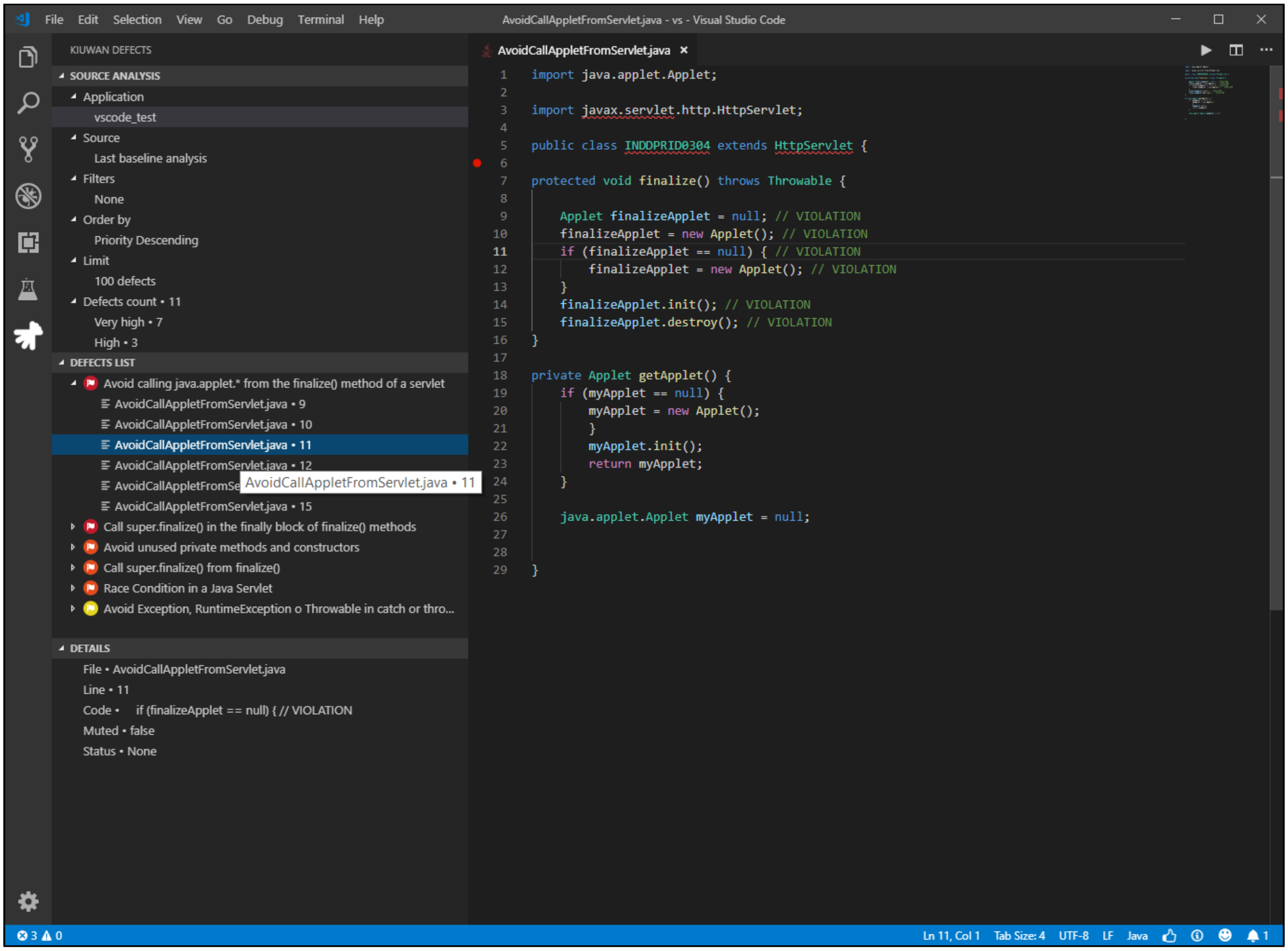
Insights support for Ruby

Kiuwan Insights now support discovering of ***Ruby*** open-source ***dependencies*** used by your app,as well as informing of ***Ruby vulnerabilities*** reported to NVD

Kiuwan for Developers plugin for VisualStudio Code

Kiuwan for Developers (K4D) for Microsoft Visual Studio Code has just been released.

VS Code plugin will facilitate and automate compliance with security normative, quality standards and best practices for several languages.


File
Edit
Selection
View
Go
Debug
Terminal
Help
AvoidCallAppletFromServlet.java - vs - Visual Studio Code
AvoidCallAppletFromServlet.java ×
Ln 11, Col 1
Tab Size: 4
UTF-8
LF
Java
3
0
1

KIUWAN DEFECTS

SOURCE ANALYSIS

- Application
  - vscode_test
- Source
  - Last baseline analysis
- Filters
  - None
- Order by
  - Priority Descending
- Limit
  - 100 defects
- Defects count • 11
  - Very high • 7
  - High • 3

DEFECTS LIST

- Avoid calling java.applet.* from the finalize() method of a servlet
  - AvoidCallAppletFromServlet.java • 9
  - AvoidCallAppletFromServlet.java • 10
  - AvoidCallAppletFromServlet.java • 11
  - AvoidCallAppletFromServlet.java • 12
  - AvoidCallAppletFromSe AvoidCallAppletFromServlet.java • 11
  - AvoidCallAppletFromServlet.java • 15
- Call super.finalize() in the finally block of finalize() methods
- Avoid unused private methods and constructors
- Call super.finalize() from finalize()
- Race Condition in a Java Servlet
- Avoid Exception, RuntimeException o Throwable in catch or thro...

DETAILS

File • AvoidCallAppletFromServlet.java
Line • 11
Code • if (finalizeApplet == null) { // VIOLATION
Muted • false
Status • None

```java
import java.applet.Applet;

import javax.servlet.http.HttpServlet;

public class INDDPRID0304 extends HttpServlet {

protected void finalize() throws Throwable {

    Applet finalizeApplet = null; // VIOLATION
    finalizeApplet = new Applet(); // VIOLATION
    if (finalizeApplet == null) { // VIOLATION
        finalizeApplet = new Applet(); // VIOLATION
    }
    finalizeApplet.init(); // VIOLATION
    finalizeApplet.destroy(); // VIOLATION
}

private Applet getApplet() {
    if (myApplet == null) {
        myApplet = new Applet();
        }
        myApplet.init();
        return myApplet;
    }

    java.applet.Applet myApplet = null;


}
```

You can find further info at Kiuwan for Developers for Microsoft Visual Studio Code