opentext™ | Cybersecurity

Fortify SAST 23.1

Technical Awareness Webinar

10 May 2023 | Frans van Buul, Sr Prod Manager Fortify SAST

Agenda

General notes

- The official language count
- Relevant releases
- Installer split
- Looking ahead: Fortify-on-Solaris end-of-life.

Version upgrades

- Languages
- Build tools
  - Side note: working with unsupported Gradle versions

New features / additional coverage

- .NET on Linux
- Dart/Flutter
- Python improvements
- Scan Policy
- Rule Properties
- 2023.R1 rule pack highlights.

opentext™ | Cybersecurity

General notes

Fortify SAST supports

31+

languages as of version 23.1

Language count

1	C#
2	VB.NET
3	TypeScript
4	JavaScript
5	Java
6	Kotlin
7	Scala
8	COBOL
9	Swift
10	Objective-C
11	Objective-C++
12	C
13	C++
14	Python
15	PHP

16	Go
17	Salesforce Apex
18	Ruby
19	ABAP
20	PL/SQL
21	T-SQL
22	ColdFusion
23	ActionScript
24	Visual Basic
25	VBScript
26	HTML
27	XML
28	JSON
29	YAML
30	HCL (new as of 22.1)

31	**Dart/Flutter (new as of 23.1)**

Roadmap/expectations beyond 23.1 (the standard disclaimers apply):

32	Bicep (23.2)
33	Solidity (23.2)
34	PL/pgSQL (2024)
35	MySQL Stored Proc Language (2024)
36	Powershell (2024)
37	Bash (2024/2025)
38	Rust
39	Groovy

The “+” refers to template-like-languages like JSP, Razor, Handlebars, etc.

Relevant releases

Fortify SCA 23.1.0 (18 May 2023)

Note: this will include Xcode 14.3. We have no patch 23.1.1 currently in sight.

Security Content 2023 Update 1 (31 March 2023)

Security Content 2023 Update 2 (expected 30 June 2023)

Installer split

As of 23.1, the former installer “SCA and Apps” has been split into

- **SCA 23.1**: sourceanalyzer and a handful of tools useful in non-end-user environments, including fortifyupdate, packagescanner, pwdtool, scancentral, SCAState
- **Apps and Tools 23.1**: auditworkbench, (BIRT)ReportGenerator, CustomRulesEditor, ScanWizard, fortifyclient
- FPRUtility and iidmigrator are present in both

Rationale:

- Smaller footprint in non-interactive environments like Docker containers acting as ScanCentral sensors; also, smaller footprint for auditors who use tools but don’t scan themselves.
- Easier, and in the future, decoupled, releases.

Covered during 9 May SSC & Tools TAW by Young Park; also, Anna Karyakina has demoed this feature on 9 May demo day (recorded).

Heads-up: Solaris support likely to be removed 23.2

Facts:

1. SCA is a Java-based application. We need to run on an LTS version of Java, currently, Java 11. The next LTS available is Java 17.
2. Customers increasingly run composition analysis on Fortify itself, and there's big pressure to update vulnerable components.
3. For one important component (Spring), the newer versions require Java 17. The end of security patches for Java 11-compatible versions is approaching. If we're still on Java 11 at that point, we'll get stuck with non-fixable vulnerable components.
4. Oracle has stopped creating Java versions for Solaris as of Java 15 (https://openjdk.org/jeps/381).

➡ We are forced to drop Solaris support soon. If you have customer for whom that is an issue, please contact me directly to discuss strategy.

opentext™ | Cybersecurity

Version upgrades

Language version upgrades

Language	22.2	23.1	Notes
.NET	5–6	5–7	
C#	5–10	5–11	
Apex	55	55–57	By design, newer (not-yet-released) versions like 58/Summer ‘23 will work as well.
Go	1.12-1.17	1.12–1.19	Go 1.20 (released 1 Feb) practically works as well due to limited changes. Go generics (new in 1.18) are not fully analyzed.
JavaScript	2015–2021	2015–2022	
Kotlin	1.3–1.6	1.3–1.7	Kotlin 1.8 (released 28 Dec) practically works as well due to limited changes.
PHP	7.3–7.4, 8.0–8.1	7.3–7.4, 8.0–8.2	
Python	2.6–2.7, 3.0–3.9	2.6–2.7, 3.0–3.11	Not just a version bump; also a new and improved translator.
TypeScript	2.8, 3.x, 4.0–4.5	2.8, 3.x, 4.x	This means 4 new minor versions (4.6–4.9). Since the next one will be 5.0, we’re now listing as 4.x.
Angular	2–13	2–15	This is in-between a language (for which we mention versions) and a framework (for which we don’t mention versions).

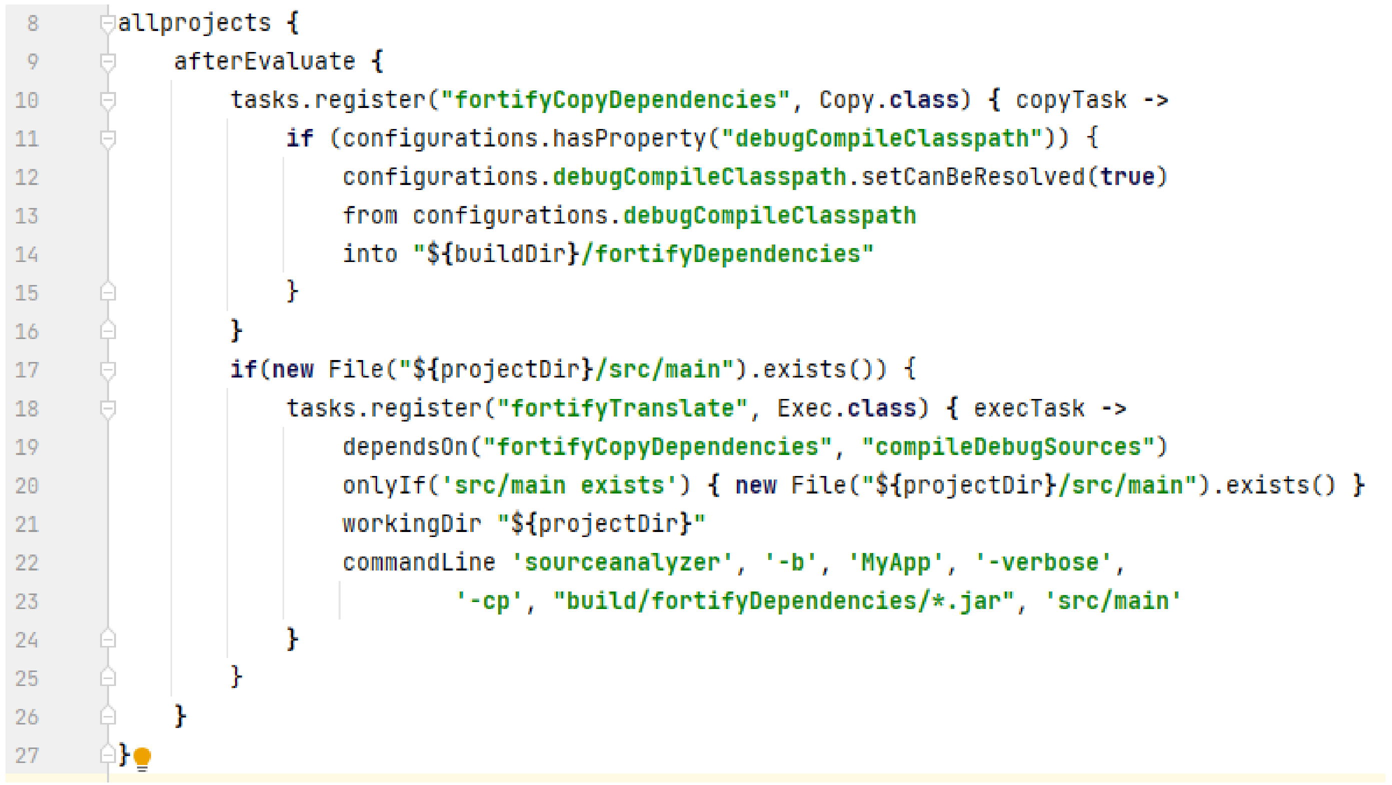
Post-23.1 release, we’ll redesign the way we communicate about supported languages, so facts like these find their way into the official docs.

Build tool/compiler version upgrades

Language	22.2	23.1	Notes
Gradle	5.0–7.4.x	5.0–7.4.x, 8.0.x	7.6 and 8.1 are not yet supported. Our support is for the Groovy DSL, not the Kotlin DSL. Android Studio Giraffe (beta as of 20 April) defaults to the Kotlin DSL for new projects, which is likely to trigger some customer tickets at some point. Roadmap for 23.2 to fix this. Unsupported usually means hard fail for Gradle. You can work around unsupported Gradle versions – more on that next slide.
Maven	3.0.x, 3.5.x, 3.6.x, 3.8.x	3.0.x, 3.5.x, 3.6.x, 3.8.x, 3.9.x	Newer versions usually don't break this integration; e.g., 22.2 works fine with 3.9.x.
MSBuild	14.0, 15.x, 16.x, 17.0–17.2	14.0, 15.x, 16.x, 17.0–17.5	
Xcode	13–14.2	13–14.3	See sysreq guide for the exact list of individual versions. 14.1 support was introduced in patch 22.2.1. This also turned out to work for 14.2; no separate patch was needed.
gcc/g++ (on any OS)	4.9, 5.x	4.9, 5.x	This is about AIX and Solaris.
gcc/g++ (on Win/Lin/Mac)	4.9, 5.x, 6.x–10.4	4.9, 5.x, 6.x–10.4, 11	Originally, the 22.2 docs specifically listed 10.2.1, excluding 6.x–9.x and other 10.x.

Working with unsupported Gradle versions

When the Gradle integration doesn't work, you can manually add some build script code to the top-level build.gradle file, e.g.,



```
allprojects {
    afterEvaluate {
        tasks.register("fortifyCopyDependencies", Copy.class) { copyTask ->
            if (configurations.hasProperty("debugCompileClasspath")) {
                configurations.debugCompileClasspath.setCanBeResolved(true)
                from configurations.debugCompileClasspath
                into "${buildDir}/fortifyDependencies"
            }
        }
        if(new File("${projectDir}/src/main").exists()) {
            tasks.register("fortifyTranslate", Exec.class) { execTask ->
                dependsOn("fortifyCopyDependencies", "compileDebugSources")
                onlyIf('src/main exists') { new File("${projectDir}/src/main").exists() }
                workingDir "${projectDir}"
                commandLine 'sourceanalyzer', '-b', 'MyApp', '-verbose',
                        '-cp', "build/fortifyDependencies/*.jar", 'src/main'
            }
        }
    }
}
```

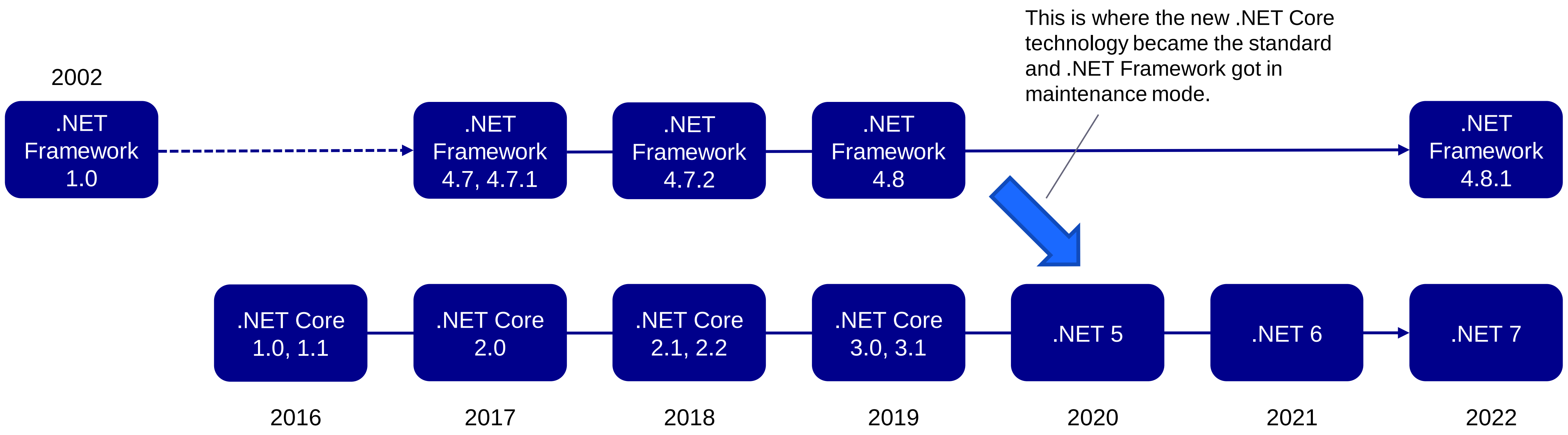
Note: this works, but it's not completely generic; you will need to tweak this to the specific case.

opentext™ | Cybersecurity

New features

.NET on Linux

Let's quickly recap .NET history:



.NET, platform dependence, and Fortify

.NET Framework was platform-independent *in theory* from the beginning.

The “Mono” project provided an open-source .NET Framework implementation that runs on Linux. This is not popular among our customers.

.NET Core / .NET has been implemented to be truly platform-independent. We have many customers using this on Linux.

Fortify has historically only supported .NET translation on Windows. As of 23.1, we also support this on Linux. The following conditions apply:

- This requires an installation of .NET 6.
- It (officially) applies to .NET 6 and newer only. It certainly won’t work with .NET Framework.
- While we support Linux, we don’t support Mac.

.NET, build tools, and Fortify

For many languages, Fortify build-tool integration is an option. For .NET, it’s mandatory.

Historically, the build tool for .NET Framework was “msbuild”. This is only officially available on Windows.

For .NET Core / .NET, there’s another build tool called “dotnet”, available everywhere.

Until 23.1, Fortify only support .NET translation through “msbuild”. That would not allow us to implement Linux support. So, 23.1 supports the “dotnet” command as well.

.NET on Linux

Live demo

Dart/Flutter

Dart

Dart is a language; files are by convention named *.dart.

We support versions 2.12–2.18. (2.x pre 2.12 may work. 1.x will not work.)

Most important use case is cross-platform mobile client development. **This is the use case Fortify focuses on.**

Can also be used for web client development, desktop apps, server apps.

Flutter

Flutter is a framework for mobile/web/desktop development.

We support versions 2.0-3.3.

It is built on top of the Dart language and imported as a dependency in a Dart app. There are no “flutter files”.

Dart/Flutter

It is impossible to have a Flutter application that isn’t a Dart application.

It is theoretically possible to have a Dart application that isn’t a Flutter application, but that’s theory only.

Logically, we should be talking about Dart as a language and Flutter as a framework.

Many customers will simply ask for “Flutter” support.

→ For all these reasons, we talk about “Dart/Flutter” in many places and have listed Flutter in the language tables of the sysreq document.

Dart/Flutter - limitations

The Dart/Flutter rule content will go out in R2 (30 June). Until that time, this feature isn’t usable by customers.

There are certain language constructs not yet fully taken into account by the translator, and this will be further developed in 23.2.

- This does not affect the ability to run a translation.
- It will, of course, lead to false negatives in certain situations.
- We are calling out Dart/Flutter in 23.1 as “supported” and *not* as “preview”, “beta” or anything. SAST support in *never* complete.

Fortify support for Dart/Flutter is for Windows and Linux. MacOS is not supported.

Dart/Flutter – practical notes

Dart translation works directly:

- `sourceanalyzer –b <buildid> <other options> <directory or file to translate>`

(Detail: files translated are currently not being sent to STDOUT, even when –verbose)

Dart/Flutter dependencies must be present prior to translation, by invoking

- `flutter pub get (for Flutter apps)`
- `dart pub get (for the Dart-but-not-Flutter app)`

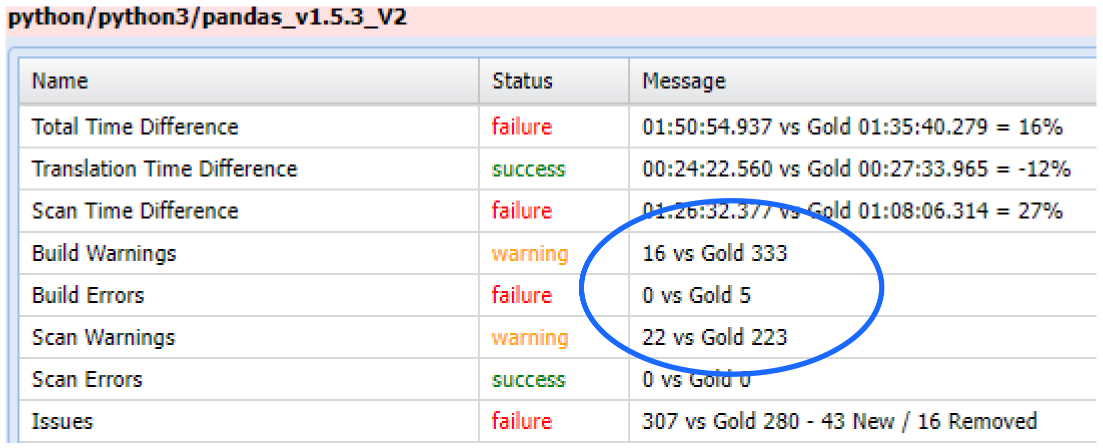
If there are nested packages with their own pubspec.yaml files, this should be done in those directories as well.

Dart/Flutter

Live demo

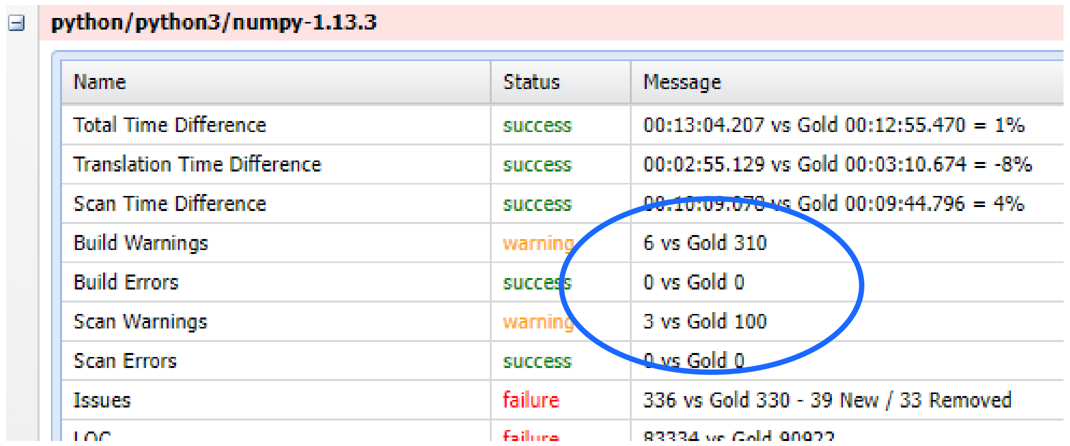
Python Improvements

New translator with drastically lower error/warning counts, examples from test bed:



python/python3/pandas_v1.5.3_V2

Name	Status	Message
Total Time Difference	failure	01:50:54.937 vs Gold 01:35:40.279 = 16%
Translation Time Difference	success	00:24:22.560 vs Gold 00:27:33.965 = -12%
Scan Time Difference	failure	01:26:32.377 vs Gold 01:08:06.314 = 27%
Build Warnings	warning	16 vs Gold 333
Build Errors	failure	0 vs Gold 5
Scan Warnings	warning	22 vs Gold 223
Scan Errors	success	0 vs Gold 0
Issues	failure	307 vs Gold 280 - 43 New / 16 Removed



python/python3/numpy-1.13.3

Name	Status	Message
Total Time Difference	success	00:13:04.207 vs Gold 00:12:55.470 = 1%
Translation Time Difference	success	00:02:55.129 vs Gold 00:03:10.674 = -8%
Scan Time Difference	success	00:10:09.078 vs Gold 00:09:44.796 = 4%
Build Warnings	warning	6 vs Gold 310
Build Errors	success	0 vs Gold 0
Scan Warnings	warning	3 vs Gold 100
Scan Errors	success	0 vs Gold 0
Issues	failure	336 vs Gold 330 - 39 New / 33 Removed
LOC	failure	83334 vs Gold 90922

Constructs are now understood that weren't understood before.

Python Improvements

Live demo

Scan Policy

This is the feature originally slated as “Noise Reduction Level”; changed this to “Scan Policy” for better perception.

It is designed to give users an options to configure Fortify to provide more relevant results. Compared to many other features we have for that,

- this is *very* easy to use;
- it suppresses results in an early stage; they don’t even occur in the FPR.

It is based on an existing (but not well-known) feature called “filter files”. We’ll review this first.

Filter Files pre-23.1

Filter Files are text files, with Unix-style ‘#’ comments allowed.

They can be used to filter out a specific

- vulnerability instance (by instance ID)
- rule (by rule ID; can also be done using a SuppressionRule custom rule)
- category or category/subcategory (by name)

They are specified during scan/analysis time (not during translation).

They are specified by the “-filter” argument. This points to the relative or absolute path of the filter.

Multiple filter files may be provided.

They operate

- before rendering (so filtered out results do not appear in the FPR),
- but after the analysis itself (so filters do not improve performance except for the rendering phase)

Scan Policy in 23.1

The Filter Files mechanism has been extended to support filtering based on numerical properties of results, e.g.:

```
Impact <= 1.5

Likelihood <= 1.5
```

In addition to the `–filter` flag, there now is a `–scan–policy` (or shorthand: `–sc`) flag.This looks for scan policy/filter files under Core/config/scales.

E.g., `–scan–policy` security will apply filter Core/config/scales/scan-policy-security.txt

Users can put their own files there and they will migrate when upgrading SCA.

Out-of-the-box scan policies 23.1

Classic

This is the default scan policy.

It is a file with just some comments.

23.1 behaves, by default, like 22.2.

Security

Filters out 39 categories that are quality related.

That doesn't mean that these never can lead to a security issue, however

- Usually they don't
- These things tend to be picked up by other tools in the pipeline.

Devops

Filters out everything that "Security" filters out.

Filters out 7 more categories that are often considered noise

Insecure Randomness, Weak XML Schema, Log Forging, Mass Assignment, Access Control: Database, Build Misconfiguration, Often Misused

Filters out issues with Impact <= 1.5 or Likelihood <=1.5

Scan Policy

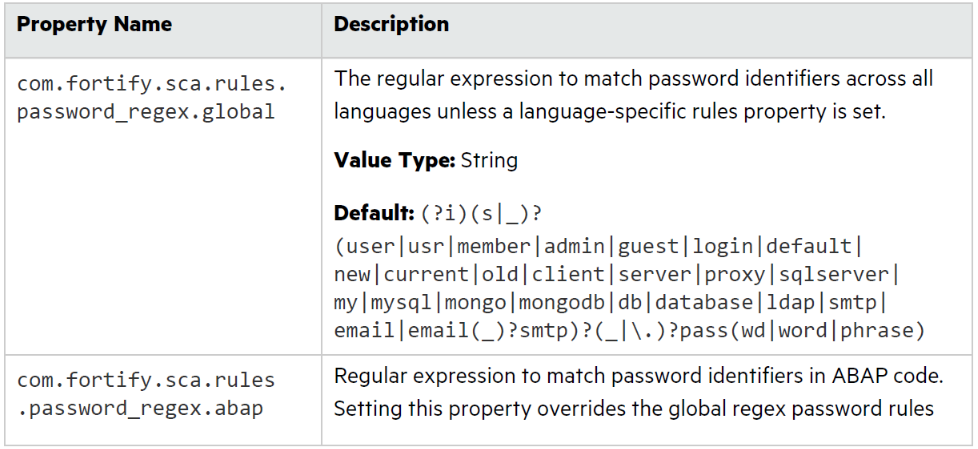
Live demo

2023.R1 rules with SCA 23.1: Rule Properties

Some rules now can be configured through properties.

Standard config file for those is “fortify-rules.properties”. Appendix B of the user guide has the details.

Primary use case is to override default regexes for passwords/keys:



Property Name	Description
`com.fortify.sca.rules.password_regex.global`	The regular expression to match password identifiers across all languages unless a language-specific rules property is set. **Value Type:** String **Default:** `(?i)(s|_)?(user|usr|member|admin|guest|login|default|new|current|old|client|server|proxy|sqlserver|my|mysql|mongo|mongodb|db|database|ldap|smtp|email|email(_)?smtp)?(_|\.)?pass(wd|word|phrase)`
`com.fortify.sca.rules.password_regex.abap`	Regular expression to match password identifiers in ABAP code. Setting this property overrides the global regex password rules

Note: this applies to identification of passwords and keys through structural rules, catching cases such as `const myKey = “xyz”`. It has nothing to do with our regex analyzer.

Note: Peter Blay did an extensive demo of this during 9 May demo day (recorded)

opentext™ | Cybersecurity

2023.R1 rule pack: other highlights

Many, many language updates. To truly support a new version of something, we need SCA and rule support. R1 contained updates for

- Go 1.17, Python 3.10, ECMAScript 2022, iOS SDK 16, Salesforce Apex 57, .NET 7

In addition, there are several new things

- Vue 2
- Google Dataflow / Java Apache Beam

As well as huge category expansions in IaC

- AWS Terraform
- Azure Terraform
- Azure ARM

Be sure to check out the release notes: https://community.microfocus.com/cyberres/fortify/w/fortify-product-announcements/44775/opentext-fortify-software-security-content-2023-update-1

Great Code Demands Great Security