OpenText™ Static Application Security Testing

ユーザガイド

Version : 25.4

PDF Generated on : May 18, 2026

Table of Contents

1. ユーザガイド

このセクションでは、OpenText™ Static Application Security Testing (OpenText SAST) を使用して、ほとんどの主要なプログラミングプラットフォームでコードをスキャンする方法について説明します。このセクションは、セキュリティ監査とセキュアコーディングを担当するユーザを対象にしています。

1.1. サポートとドキュメント

カスタマサポートにお問い合わせの際は、次の製品情報をご提供ください。
ソフトウェアバージョン: 25.4.0
ソフトウェアリリース日:25.4.0

カスタマサポートへのお問い合わせ

カスタマサポートWebサイトにアクセスして、次の作業を実行できます。

- ライセンスとエンタイトルメントの管理
- 技術サポートリクエストの作成と管理
- ドキュメントやナレッジ記事の閲覧
- ソフトウェアのダウンロード
- コミュニティの探索

詳細情報

OpenText Application Security Testing製品の詳細については、「**Application Security**」を参照してください。

製品の機能紹介ビデオ

Fortify Unplugged YouTube™ チャンネルで、OpenText Application Security Softwareの製品と機能をハイライトするビデオをご覧いただけます。

1.2. 変更ログ

次の表に、このヘルプ/ドキュメントに加えられた変更を示します。このヘルプ/ドキュメントの改訂版は、変更が製品の機能に影響を与える場合にのみ、ソフトウェアリリース間で発行されます。

ソフトウェアリリース/ ドキュメントのバージョン	変更点
25.4.0	追加: • 新しいXcodeビルドとMSBuildのバージョンを追加しました(「サポートされるビルドツール」を参照) • 新しい.NET (Core)、C#、Java、Go、Kotlin、およびSftのバージョンを追加しました(「言語の互換性」を参照) • 新しいコンパイラバージョンのOpenJDK javacおよびSwiftcを追加しました(「サポートされるコンパイラ」を参照) • 新しい `com.fortify.sca.rules.lslibrary` プロパティと `com.fortify.sca.rules.enablePQCRules` プロパティを追加しました(fortify-rules.properties) • ライブラリコードの分析に関するページを追加しました(ライブラリコードの分析) • 新しい `com.fortify.sca.EnableSubtraceFiltering` プロパティを追加しました(変換および分析フェーズのプロパティ) • 複合フィルタに関するセクションを「フィルタファイルによる問題の除外」に追加しました 更新: • *<languages>の変換*についてのすべての言及を*<languages>の分析*に変更しました • すべての言語セクションを最上レベルに作成し、識別しやすくしました。 • 分析プロセスの概要を簡素化しました

ソフトウェアリリース/ ドキュメントのバージョン	変更点
	• ビルド統合セクションは、それぞれの言語セクションに移動されました。 • Java、Kotlin、およびAndroidセクションがマージされました。(「Java、Kotlin、およびJSPプロジェクトの分析」を参照) • iOSセクションを再編成しました。(「iOSおよびXcodeプロジェクトの分析」を参照) • スキャンポリシーセクションを分析の概要から移動し、フィルタと結果を向上させる他の方法を結合して新しいセクションに入れました。(「結果の最適化」を参照) • 正規表現分析に関するセクションを、シークレットスキャンの最上部セクションに移動しました 削除: • ScanCentral SASTクライアントを、OpenText SASTインストーラから削除しました。 • Gradleバージョン6.5以前のバージョンを削除しました(「サポートされるビルドツール」を参照)

ソフトウェアリリース/ ドキュメントのバージョン	変更点
25.3.0	追加: • Xcodeビルドバージョンが更新されました(「システム要件」を参照) • FortifyRemoveコメント機能を制御する新しいルールプロパティを追加しました(fortify-rules.properties) • 「FortifyRemoveを使用するコメントのフィルタリング」を追加しました • MacOS ARMインストーラ(「ソフトウェアの取得」を参照) 更新: • *Fortify Sofware Security Content*のすべての言及をOpenText Application Security Contentに変更しました 削除: • xcodeビルドバージョン15、15.0.1、15.1、15.2を削除しました(「サポートされるビルドツール」を参照)。

ソフトウェアリリース/ ドキュメントのバージョン	変更点
25.2.0	追加: • システム要件 • カスタムスキャンポリシーの作成方法についての手順(スキャンポリシーを分析に適用する) 更新: • 組み込まれた製品名の変更(「製品名の変更」を参照) • 製品名の変更に沿ったインストーラファイル名の変更(「OpenText SASTのインストール」に関するトピックを参照) • Visual Studioプロジェクトの変換では、テストプロジェクトは既定で除外されます(「Visual Studioプロジェクトのコマンドライン構文」を参照)。 • Jupyterノートブックのサポートを追加しました(「Pythonコードの変換」を参照) • DjangoやFlaskフレームワークを使って作成されたコードを変換するのにリミタープロパティの設定は不要になりました 削除: • プロパティ `com.fortify.sca.SuppressLowSeverity` および `com.fortify.sca.LowSeverityCutoff` は、Rulepacksで非推奨になったメタ

ソフトウェアリリース/ ドキュメントのバージョン	変更点
	データを参照していたため削除されました。 • この com.fortify.sca.hoa.Enable プロパティはこのヘルプドキュメントから削除されており、今後のリリースで製品から削除される予定です。

ソフトウェアリリース/ ドキュメントのバージョン	変更点
24.4.0	更新: • ARM上のLinux用のインストーラを追加しました(「OpenText SASTのインストール」を参照) • スキャンポリシーは、テイントフラグに基づいてデータフローの問題を除外できます(「分析へのスキャンポリシーの適用」を参照) • デフォルトでは、NPMの依存関係は分析フェーズから除外されます(「NPM依存関係の変換の管理」を参照) • FlaskおよびJinja2のサポートを追加しました(「Pythonコードの変換」を参照) • GoプロジェクトのOpenText SAST変換にカスタムビルドタグを含めるための `-gotags` オプションを追加しました(「カスタムGoビルドタグを含める」および「Goのプロパティ」を参照) • PL/SQLを分析するためのコマンドラインオプションの変更(「変換SQLの分析」を参照) • ビルドツール名の解決を無効にし、ビルドスクリプトファイルをソースファイルとして変換するオプションを追加しました(「変換オプション」および「変換および分析フェーズのプロパティ」を参照) • `-exclude` オプションは、Ant、Bazel、Gradle、およびMavenのビルド統合でサポートされています(Antと

ソフトウェアリリース/ ドキュメントのバージョン	変更点
	の統合、Bazelとの統合、Gradle統合の使用、Fortify Mavenプラグインの使用、および変換オプションを参照) 削除: • モジュール分析は、このヘルプ/ドキュメントから削除されました。この機能は非推奨であり、次回のリリースで製品から削除される予定です。

1.3. はじめに

このセクションでは、次のトピックについて説明します。

1.3.1. 製品名の変更

OpenTextでは、次の製品名を変更中です。

前の名前	新しい名前
Fortify Static Code Analyzer	OpenText™ Static Application Security Testing (OpenText SAST)
Fortify Software Security Center	OpenText™ Application Security
Fortify WebInspect	OpenText™ Dynamic Application Security Testing (OpenText DAST)
Fortify on Demand	OpenText™ Core Application Security
Debricked	OpenText™ Core Software Composition Analysis (OpenText Core SCA)
Fortifyアプリケーションとツール	OpenText™ Application Security Tools

これらの製品名は、製品のスプラッシュページ、マストヘッド、ログインページ、および製品が識別されるその他の場所で変更されました。名前の変更は、製品の機能を明確にし、Fortify Software製品とOpenTextとの整合性を高めることを目的としています。ドキュメントのタイトルページなど、場合によっては、古い名前が一時的に括弧に含まれる場合があります。今後の製品リリースで、さらに多くの変更を予定しています。

1.3.2. OpenText SAST

OpenText SAST (Fortify Static Code Analyzer)は、さまざまな言語でセキュリティ固有のコーディングルールおよびガイドラインの違反を検索する一連のソフトウェアセキュリティアナライザです。OpenText SASTでは、よりセキュアなソフトウェアを提供し、セキュリティコードレビューの効率性、一貫性、および完全性を向上させるのに役立つ分析情報が生成されます。お客様固有のセキュリティルールを組み込むことのできる設計になっています。

サポートされている言語、ライブラリ、コンパイラ、およびビルドツールのリストについては、「システム要件」を参照してください。

OpenText SASTを使用してアプリケーションを分析するには、次の方法があります。

- 次のSecure Code Pluginsのいずれかを使用して、IDEから直接分析を実行します: Visual Studio用のFortify Extension、Eclipse用のFortify Plugin、およびIntelliJ IDEAおよびAndroid Studio用のFortify Analysis Plugin)。この分析は、Fortify Audit Workbenchを使用して実行することもできます。

  セキュリティ脆弱性分析の結果をIDEおよびFortify Audit Workbenchで表示したり、結果をApplication Securityにアップロードすることもできます。ツールの説明については、「OpenText Application Security Tools」を参照してください。

- 分析をビルドシステムに統合するか、またはコマンドラインから分析を実行します。

  このガイドでは、分析のこの実行方法を主に説明します。

1.3.2.1. アナライザについて

OpenText SASTは、8つの脆弱性アナライザ(Buffer、Configuration、Content、Control Flow、Dataflow、Null Pointer、Semantic、およびStructural)で構成されています。各アナライザでは、実行された分析に対応するタイプに必要な情報を提供するために特別にカスタマイズされた別のタイプのルールが受諾されます。ルールは、ソースコード内のセキュリティの脆弱性や安全でない状態が発生する可能性がある要素を特定する定義です。次の表では、各アナライザについて説明します。

アナライザ	説明(Description)
Dataflow	Dataflow Analyzerでは、テイントデータ(ユーザ制御入力またはプライベートデータ)の潜在的に危険な使用が発生する潜在的な脆弱性が検出されます。Dataflow Analyzerは、プロシージャ間テイント伝播分析を使用して、ユーザ入力(またはプライベートデータ)のサイトから、アプリケーションを通って、危険な関数呼び出しや操作に至るデータの流れを検出します。たとえば、Dataflow Analyzerは、ユーザが制御する入力文字列がHTML (クロスサイトスクリプティング)を動的に生成するかどうかを検出し、ユーザが制御する文字列がSQLクエリを構成するかどうか(SQLインジェクション)を検出します。
Control Flow	Control Flow Analyzerでは、一連の危険な操作が検出されます。プログラムで制御フローパスを分析すると、Control Flow Analyzerで一連の操作が特定の順序で実行されているかどうかが判断されます。たとえば、Control Flow Analyzerは、チェックの時刻/使用時刻の問題や競合状態を検出し、XMLリーダーなどのユーティリティが使用前に適切に設定されているかどうかをチェックします。

アナライザ	説明(Description)
Buffer	Buffer Analyzerでは、バッファに保持できる以上のデータの書き込みまたは読み取りが発生するバッファオーバーフローの脆弱性が検出されます。バッファはスタックで割り当てられるか、ヒープで割り当てられます。Buffer Analyzerでは、制限付きのプロシージャ間分析を使用して、バッファオーバーフローが発生する原因になる状態であるかどうかが判断されます。バッファへの実行パスが原因でバッファオーバーフローが発生する場合は、OpenText SASTでバッファオーバーフローの脆弱性としてレポートされ、オーバーフローの原因になる可能性がある変数が指摘されます。バッファオーバーフローの発生原因となる変数の値が汚染(ユーザ制御)されている場合は、その値もOpenText SASTでレポートされ、どのように変数が汚染されているのかを示すデータフロートレースが表示されます。Buffer Analyzerは、バッファアンダーリード状態およびバッファアンダーフロー状態も検出します。
Structural	Structural Analyzerでは、プログラムの構造または定義の潜在的に危険な欠陥が検出されます。Structural Analyzerでは、変数および関数の宣言と使用の両方を含む幅広い範囲が網羅されるため、プログラムの構造を理解することで、検査による検出が難しい場合が多いセキュアなプログラミング手法や技術の違反も特定されます。たとえば、Structural Analyzerは、ハードコードされたシークレット、コード内でのCookieの誤設定、および暗号化の弱点を検出します。

アナライザ	説明(Description)
Configuration	Configuration Analyzerでは、アプリケーション展開環境設定ファイルの間違い、弱点、およびポリシー違反が検索されます。たとえば、Configuration Analyzerでは、Webアプリケーションでユーザセッションに適切なタイムアウトがチェックされます。Configuration Analyzerでは、正規表現分析も実行されます(「正規表現分析」を参照)。
Semantic	Semantic Analyzerでは、プロシージャ内レベルで関数とAPIの潜在的に危険な使用が検出されます。
Content	Content Analyzerでは、HTMLコンテンツのセキュリティ問題とポリシー違反が検索されます。Content Analyzerでは、静的なHTMLページに加えて、動的なHTMLを含むファイル(PHP、JSP、従来のASPファイルなど)でも、これらのチェックが実行されます。
Null Pointer	Null Pointer Analyzerでは、null値が割り当てられたポインタ変数の逆参照が検出されます。Null Pointer Analyzerの検出は、プロシージャ内レベルで実行されます。問題は、null割り当て、逆参照、およびこれらの間のすべてのパスが単一の関数内で発生した場合にのみ検出されます。

1.3.3. ライセンス

OpenText SASTでは、セキュリティ分析の変換フェーズと分析(スキャン)フェーズの両方を実行するためにライセンスが必要です(これらのフェーズの詳細については、「分析プロセス」を参照)。

ご使用の製品のFortifyライセンスファイルをソフトウェアライセンスおよびダウンロード(SLD)ポータルからダウンロードする必要があります。カスタマサポートがアクセス用に提供した資格情報を使用します。

OpenText SASTをインストールするには、Fortifyライセンスファイル(`fortify.license`)が必要です。オプションで、Fortify License and Infrastructure Managerを使用してOpenText SASTの同時ライセンスを管理できます。LIMで管理される同時ライセンスを使用すると、複数のOpenText SASTのインストールで単一のライセンスを共有できます。OpenText SASTのライセンスでLIMを設定する方法については、『*OpenText™ Fortify License and Infrastructure Managerインストールおよび使用ガイド*』を参照してください。OpenText SASTからLIMライセンスを管理する方法については、「LIMライセンスディレクティブ」を参照してください。

Note

OpenText SASTの同時ライセンスを管理するためにOpenText™ Fortify License and Infrastructure Manager (LIM)を使用するには、LIMバージョン21.2.0以降が必要です。

1.3.4. 有効期限が切れたライセンスの更新

OpenText SASTのライセンスは1年ごとに有効期限が切れます。

有効期限が切れたライセンスを更新するには:

- 更新されたFortifyライセンスファイルを、OpenText SASTがインストールされているルートディレクトリに入れます。

有効期限が切れたLIMで管理される同時ライセンスを更新する方法については、『*OpenText™ Fortify License and Infrastructure Managerインストールおよび使用ガイド*』を参照してください。

1.3.5. OpenText Application Security Content

OpenText SASTでは、ルールのナレッジベースを使用して、静的分析用のコードベースにセキュアなコーディング標準が強制的に適用されます。OpenText Application Security Contentは、変換と分析の両方に必要です。OpenText SASTのインストール時に、セキュリティコンテンツをダウンロードしてインストールできます(「OpenText SASTのインストール」を参照)。また、インストール後のタスクとしてfortifyupdateコマンドラインツールを使用して、OpenText Application Security Contentをダウンロードしたり、あらかじめダウンロードしておいたものをインポートしたりすることもできます(「OpenText Application Security Contentの手動インストール」を参照)。

OpenText Application Security Contentは、Fortify Secure Coding Rulepacksおよび外部メタデータで構成されます。

- Fortify Secure Coding Rulepacksには、一般的な言語およびパブリックAPIに対する一般的なセキュアコーディングのイディオムが記述されています。
- 外部メタデータには、Fortifyカテゴリから代替カテゴリ(CWE、OWASP Top 10、PCIなど)へのマッピングが含まれています。

OpenTextは、OpenText SASTとFortify Secure Coding Rulepacksの機能に追加する、カスタムルールを記述する機能を提供します。たとえば、場合によっては、専有セキュリティガイドラインを適用したり、すでにFortify Secure Coding Rulepacksの対象ではないサードパーティのライブラリや事前コンパイルされたその他のバイナリを使用するプロジェクトを分析したりする必要があります。また、外部メタデータをカスタマイズして、さまざまな分類体系(内部アプリケーションのセキュリティ基準や追加のコンプライアンス義務など)にFortifyの問題をマップすることもできます。独自のカスタムルールまたはカスタムの外部メタデータを作成する方法については、『*OpenText™ Static Application Security Testingカスタムルールガイド*』を参照してください。

セキュリティコンテンツを定期的に更新することが推奨されています。`fortifyupdate`を使用すると、最新のセキュリティコンテンツを取得できます。詳細については、「セキュリティコンテンツの更新」を参照してください。

1.3.6. Fortify ScanCentral SAST

OpenText™ ScanCentral SASTを使用すると、OpenText SASTの分析フェーズをビルドコンピュータから、この目的のためにプロビジョニングされたコンピュータのコレクションにオフロードすることでリソースを管理できます。ほとんどの言語では、ScanCentral SASTで変換フェーズと分析(スキャン)フェーズの両方を実行できます。Application Securityのユーザは、FPRファイルをサーバに直接出力するようにScanCentral SASTに指示できます。OpenText SASTのインストール時にScanCentral SASTクライアントをインストールすることもできます。

次の2つの方法のいずれかでコードを分析できます。

- ScanCentral SASTの変換でサポートされている言語でアプリケーションが記述されている場合は、分析の変換フェーズと分析(スキャン)フェーズをScanCentral SASTにオフロードできます。
- ローカルビルドコンピュータ上で変換フェーズを実行し、モバイルビルドセッション(MBS)を生成します。MBSファイルを使用して、ScanCentral SASTでスキャンを開始します。このプロセスでは、ビルドコンピュータを解放することに加えて、必要に応じてリソースを追加することで、ビルドプロセスを中断することなく、システムを拡張することができます。MBSの詳細については、「モバイルビルドセッション」を参照してください。

翻訳でサポートされている特定の言語と、ScanCentral SASTの設定および使用方法については、『*OpenText™ ScanCentral SASTインストール、設定、および使用ガイド*』を参照してください。

1.3.7. OpenText Application Security Tools

OpenTextでは、OpenText SAST、ScanCentral SAST、およびApplication Securityと統合するアプリケーションとツール(Secure Code Pluginsを含む)が用意されています。次の表では、OpenText Application Security Toolsインストーラを使用してインストールできるアプリケーションについて説明します。OpenText Application Security Toolsのインストール方法については、『*OpenText™Application Securityツールガイド*』を参照してください。

アプリケーション	説明(Description)
OpenText™ Fortify Audit Workbench	開発者がセキュリティ上の欠陥を迅速に修正できるように分析結果を整理、調査、および優先順位付けするのに役立つグラフィカルユーザインタフェースを提供するアプリケーション。
OpenText™ Fortify Plugin for Eclipse	プロジェクトのコードベース全体をスキャンして分析し、Eclipse IDEからJavaコードの脆弱性を特定するソフトウェアセキュリティルールを適用する機能を追加します。結果とともに、個々のセキュリティ問題の説明とそれらの解消に関する提案が表示されます。
OpenText™ Fortify Analysis Plugin for IntelliJ IDEAおよびAndroid Studio	プロジェクトのコードベース全体でスキャンを実行し、IntelliJ IDEAおよびAndroid Studioからコード内の脆弱性を識別するソフトウェアセキュリティルールを適用する機能を追加します。
OpenText™ Fortify Extension for Visual Studio	ソリューションおよびプロジェクトのセキュリティ脆弱性をスキャンして見つけ、Visual Studioにスキャン結果を表示する機能を追加します。結果には、見つかった問題のリスト、各問題が表す脆弱性のタイプの説明、それらの修正方法に関する提案が含まれます。この拡張機能には、Application Securityサーバに保存された監査結果を使用する修正機能も含まれています。
OpenText™ Fortifyカスタムルールエディタ	カスタムルールを作成および編集するためのアプリケーション。

アプリケーション	説明(Description)
Fortify Scan Wizard	(ローカルまたはScanCentral SASTを使用してリモートで)コードをスキャンするスクリプトを準備し、必要に応じて結果をApplication Securityにアップロードできるグラフィカルユーザインタフェースを提供します。
BIRTReportGenerator ReportGenerator	FPRファイルから問題レポート(BIRT)およびレガシーレポートを生成するコマンドラインツール。

1.3.8. サンプルプロジェクト

OpenTextでは、個別にダウンロードできるサンプルプロジェクトが `OpenText_SAST_Fortify_Samples_<version>.zip` パッケージで提供されています。

このZIPファイルには、 `basic` ディレクトリと `advanced` ディレクトリの2つが含まれています。各コードサンプルには、OpenText SASTでコードをスキャンしてその結果をFortify Audit Workbenchに表示する方法について説明する `README.txt` ファイルが含まれています。

`basic` ディレクトリには、簡単な言語固有のコードサンプルのセットが含まれています。`advanced` ディレクトリには、より高度なサンプルが含まれています。

1.3.9. 関連ドキュメント

このトピックでは、OpenText Application Security Software製品に関する情報を提供しているドキュメントについて説明します。

すべての製品

以下のドキュメントには、すべての製品に関する一般情報が記載されています。特に明記されている場合を除き、これらのドキュメントは各製品の製品マニュアルのWebサイトで利用できます。

ドキュメント/ファイル名	説明(Description)
OpenText Application Security Softwareについて appsec-docs-n-<*version*>.pdf	このドキュメントでは、OpenText Application Security Softwareのドキュメントにアクセスする方法について説明しています。 **Note** このドキュメントは、製品のダウンロードにのみ含まれています。
OpenText Application Security Softwareリリースノート appsec-rn-<*version*>.pdf	このドキュメントでは、OpenText Application Security Softwareのこのリリースで行われた変更の概要と、他の製品ドキュメントには記載されていない重要な情報について説明します。

OpenText SAST

以下のドキュメントには、OpenText SAST (Fortify Static Code Analyzer)に関する情報が記載されています。特に明記されている場合を除き、これらのドキュメントは、製品マニュアルのWebサイト(www.microfocus.com/documentation/fortify-static-code-analyzer-and-tools)で入手できます。

ドキュメント/ファイル名	説明(Description)
OpenText™ Static Application Security Testingユーザガイド sast-ugd-<*version*>.pdf	このドキュメントでは、多くの主要なプログラミングプラットフォームにOpenText SASTをインストールし、コードをスキャンするために使用する方法について説明します。これは、セキュリティ監査とセキュアコーディングを担当するユーザを対象にしています。
OpenText™ Static Application Security Testingカスタムルールガイド sast-cr-ugd-<*version*>.zip	このドキュメントでは、OpenText SASTのカスタムルールを作成するために必要な情報について説明します。このガイドには、ルール作成の概念を実際のセキュリティ問題に適用する例が含まれています。 **Note** このドキュメントは、製品のダウンロードにのみ含まれています。
OpenText™ Fortify License and Infrastructure Managerインストールおよび使用ガイド lim-ugd-<*version*>.pdf	このドキュメントでは、Fortify License and Infrastructure Manager (LIM)をインストール、設定、使用する方法について説明します。LIMは、ローカルWindowsサーバにインストールして、Dockerプラットフォーム上のコンテナイメージとして使用できます。

OpenText Application Security Tools

次のドキュメントには、OpenText Application Security Toolsに関する情報が記載されています。これらのドキュメントは、製品ドキュメントのWebサイト (https://www.microfocus.com/documentation/fortify-static-code-analyzer-and-tools) で入手できます。

ドキュメント/ファイル名	説明(Description)
OpenText™ Application Securityツールガイド sast-tgd-<*version*>.pdf	このドキュメントでは、Application Security Toolsのインストール方法について説明します。OpenText SASTを使用してコードのスキャン、分析結果の確認、分析結果ファイルの操作などを行うことのできるアプリケーションとコマンドラインツールの概要を提供します。
OpenText™ Fortify Audit Workbenchユーザガイド awb-ugd-<*version*>.pdf	このドキュメントでは、Fortify Audit Workbenchを使用して、ソフトウェアプロジェクトをスキャンして分析結果を監査する方法について説明します。このガイドには、バグトラッカとの統合方法、レポートの作成方法、共同監査の実行方法も記載されています。
OpenText™ Fortify Plugin for Eclipseユーザガイド ep-udg-<*version*>.pdf	このドキュメントでは、Fortify Plugin for Eclipseをインストールして、コードを分析および監査するために使用する方法について説明しています。
OpenText™ Fortify Analysis Plugin for IntelliJ IDEA and Android Studioユーザガイド iap-udg-<*version*>.pdf	このドキュメントでは、Fortify Analysis Plugin for IntelliJ IDEA and Android Studioをインストールして、コードを分析し、必要に応じて分析結果をApplication Securityにアップロードするために使用する方法について説明しています。
OpenText™ Fortify Extension for Visual Studioユーザガイド vse-ugd-<*version*>.pdf	このドキュメントでは、Fortify Extension for Visual Studioをインストールおよび使用して、コードを分析、監査、修復し、ソリューションとプロジェクトのセキュリティに関する問題を解決する方法について説明します。

1.4. システム要件

このコンテンツ章では、システム要件、サポートされている言語、ビルドツール、コンパイラ、およびOpenText SASTソフトウェアパッケージの取得方法について説明します。

このセクションでは、次のトピックについて説明します。

1.4.1. ハードウェア要件

CPU、メモリ、ストレージなどのシステムリソースは、プロジェクトの全体的な分析時間に大きな影響を与える可能性があります。全体的なコードサイズ、構成、言語、コードの複雑さなど、ターゲットプロジェクトのコードベースに関連する多くの要因に依存します。次のガイドでは、多数の異なる実際のアプリケーションをスキャンした経験に基づいて、一般的な開始点について説明します。

アプリケーションのサイズと複雑さ	CPUコア数	RAM (GB)	説明(Description)
小さくシンプル	4	16	サーバまたはデスクトップ上で実行される小規模なスタンドアロンシステム(バッチジョブやコマンドラインツールなど)。以下を含みます: • 10,000未満の関数
小さくシンプル(動的言語)	8	32	複雑なコンピュータモデルで動作するスタンドアロンシステム(税額計算システムやスケジューリングシステムなど)。以下を含みます: • 10,000未満の関数 • 主にJavaScript、TypeScript、Python、PHP、Rubyなどの動的言語

アプリケーションのサイズと複雑さ	CPUコア数	RAM (GB)	説明(Description)
Medium	16	64–128	トランザクションデータ処理を備えた3階層ビジネスシステム(財務システムや商用Webサイトなど)。以下を含みます: • 100,000未満の関数 • 100万行以上のコード
大規模で複雑	32	256	コンテンツを配信するシステム(アプリケーションサーバ、データベースサーバ、コンテンツ管理システムなど)。以下を含みます: • 100万以上の関数 • 数百万行のコード

OpenText SASTは、システムで利用可能なすべてのCPUコアを利用して、大規模なプロジェクトのスキャン時間を短縮します。OpenText SASTを実行すると、スキャンに使用できるハードウェアの全リソースが使用されると想定されるため、OpenText SASTの実行時にCPUを大量に消費する他のプロセスは実行しないようにしてください。

システムリソースの調整に関するその他の考慮事項:

- **仮想システム**—仮想化では、ワークロード内の未使用のリソースを特定し、他のワークロードに再割り当てすることで、ハードウェアリソースをスケールできます。OpenText SAST分析は一般的に長期にわたるリソース集約型プロセスであるため(特に、大規模かつ複雑なプロジェクトで)、リソースのスワッピングを減らすために、仮想化層での専用リソースを推奨しています。
- **CPU**—全体的な処理能力は、分析に必要な合計時間に大きな影響を与える可能性があります。高速クロック速度(コアあたりのGHz)を備える高性能プロセッサを推奨しています。システムで使用可能なコア数と必要なメモリ容量との間には相関関係があることに注意が必要です。
- **メモリ**—最適なパフォーマンスを得るために必要なメモリの量を決定する方法の詳細については、「メモリチューニング」を参照してください。JavaScript、TypeScript、Python、PHP、Rubyなどの動的言語の分析では、スキャンフェーズで他の言語よりも多くのメモリが必要です。
- **ディスクI/O**—プロジェクトの変換とスキャンは、大量のデータをシリアル化し、より高速なストレージから利益を得るI/O集約型のアクティビティです。可能な限り高速なSSDストレージで分析を実行することが推奨されています。
- **関数の数**— `-debug` オプションを使ってスキャンを実行し、サポートログファイルで `NameTable.funs: ###` の値が最後に出現した箇所を探すことで、分析中にモデル化された関数の数を確認できます。

参照情報

サンプルスキャン

1.4.1.1. サンプルスキャン

以下のサンプルスキャンは、専用の仮想マシン上でOpenText SASTバージョン25.4.0を使用して実行されました。これらのスキャンは、OpenText Application Security Content 25.4 Updateで実行されました。次の表は、いくつかの一般的なオープンソースプロジェクトで期待できるスキャン時間を示しています。

言語	プロジェクト名	変換時間 (mm:ss)	分析(スキャン)時間 (mm:ss)	問題の合計数	LOC	システム設定
.NET (C#)	SharpZipLib	01:27	14:05	606	31,863	4個のCPUと32GBのRAMを搭載したWindows Server 2022
ABAP	abap2UI5	00:13	00:52	11	59,111	4個のCPUと32GBのRAMを搭載したLinux (AlmaLinux 9)
C/C++	nasm 0.98.38	00:36	04:49	738	35,997	8個のCPUと32GBのRAMを搭載したLinux (Centos 7)
Java	WebGoat 8	00:17	00:59	252	23,662	4個のCPUと32GBのRAMを搭載したLinux (AlmaLinux 9)

言語	プロジェクト名	変換時間 (mm:ss)	分析(スキャン)時間 (mm:ss)	問題の合計数	LOC	システム設定
Java	WordPress for Android	00:10	01:48	534	35,276	4個のCPUと32GBのRAMを搭載したLinux (AlmaLinux 9)
JavaScript	three.js	06:14	14:43	277	639,230	8個のCPUと32GBのRAM、Java 17を搭載したLinux (AlmaLinux 9)
PHP	CakePHP	00:22	00:03	4,182	136,594	4個のCPUと32GBのRAMを搭載したLinux (AlmaLinux 9)

言語	プロジェクト名	変換時間 (mm:ss)	分析(スキャン)時間 (mm:ss)	問題の合計数	LOC	システム設定
PHP	phpBB 3	00:34	02:05	1,305	206,873	4個のCPUと32GBのRAMを搭載したLinux (AlmaLinux 9)
Python 3	numpy-1.13.3	02:24	09:28	217	563,457	4個のCPUと32GBのRAMを搭載したLinux (AlmaLinux 9)
Swift	MediaBrowser	00:16	01:26	9	17,699	4個のCPUと16 GBのRAMを搭載したmacOS®
TypeScript	rxjs-7.8.1	02:19	07:24	59	204,006	8個のCPUと32GBのRAM、Java 17を搭載したLinux (AlmaLinux 9)

1.4.2. サポートされているプラットフォームとアーキテクチャ

OpenText SASTは、次の表に示すプラットフォームとアーキテクチャをサポートしています。

オペレーティングシステム	プラットフォーム	配布パッケージとバージョン	メモ
Microsoft Windows®	x64	Windows 10, 11 Windows Server 2019, 2022	
Linux®	x64 ARM	CentOS Linux 7.x (7.6以降) Red Hat® Enterprise Linux® 7.x (7.2以降)、8.x (8.2以降)、9.x SUSE® Linux® Enterprise Server 15 Ubuntu® 20.04.1 LTS, 22.04.1 LTS	
macOS®	x64 M series ARM	14, 15	

オペレーティングシステム	プラットフォーム	配布パッケージとバージョン	メモ
IBM® AIX®	Power ISA	7.1, 7.2, 7.3	**Important** IBM XL C/C++ for AIX 16.1ランタイム環境パッケージがインストールされている必要があります。

1.4.3. ソフトウェア要件

OpenText SASTのインストールには、ソフトウェアが必要とする組み込みOpenJDK/JREバージョン17が含まれています。Java 17をインストールする必要はありません。

Note
組み込みOpenJDK/JREを新しいバージョンにアップグレードすることは推奨されません。

OpenText SASTを使用するには、OpenText SASTインストールディレクトリに対する読み込みおよび書き込み権限が必要です。

次の表は、特定のプロジェクトタイプの分析に必要なソフトウェア要件を示しています。

言語	ソフトウェア	オペレーティングシステム
Visual Studio、MSBuild、または.NETプロジェクト	.NET Framework 4.8以降 (MSBuildのみ)	Windows
	.NET SDK 8.0	Windows, Linux
ABAP®/BSP	Fortify ABAP Extractorは、ABAP Platform 2023 / ABAPバージョン7.58で動作するシステム上でサポートされています。	All
Bicep	.NET SDK 8.0	Windows, Linux
COBOL	Microsoft Visual C++ 2017 Redistributable (x86) **Note** これは、レガシーCOBOL分析の要件ではありません。	Windows
Scala	Akkaコンパイラプラグインは、Maven Central Repositoryで利用できます。	All

1.4.4. 言語の互換性

OpenText SASTは、次に示す言語バージョンとの互換性を検証します。これらのバージョンはテスト済みですが、OpenText SASTは柔軟性を念頭に置いて設計されており、明示的に検証されていない他のバージョンを正常にスキャンできます。

ユーザには、OpenText SASTの最新バージョンにアップグレードし、互換性を判断するためにスキャンを実行するようお勧めします。新しい未確認のバージョンのスキャンで問題が発生した場合や、現在サポートされていない言語をスキャンする場合には、OpenTextサポートに問い合わせてください。

言語/フレームワーク	検証済みの互換性
.NET (Core)	2.0-10.x
.NET Framework	2.0–4.8
ABAP/BSP	6.x, 7.x
ActionScript	3.0
Apex	55–61
Bicep	0.12.x–0.15.31
C#	5–14
<C>	C11、C17、C23 (「コンパイラ」を参照)
C++	C++11、C++14、C++17、C++20 (「コンパイラ」を参照)
クラシックASP (VBScriptを使用)	2.0, 3.0
COBOL	IBM Enterprise COBOL for z/OS 6.1~6.3 (CICS、IMS、DB2、およびIBM MQ) Visual COBOL 6.0–8.0
ColdFusion	8–10
Dart™	2.12-3.8
Docker® (Dockerfiles)	任意

言語/フレームワーク	検証済みの互換性
Go™プログラミング言語	1.12–1.25
HCL	2.0 **Note** HCLの言語サポートは、Terraformおよびサポートされるクラウドプロバイダのコードとしてのインフラストラクチャ(IaC)構成に固有のものです。
HTML	5以前
Java (Androidを含む)	7–25
JavaScript	ECMAScript® 2015–2024
JSON	ECMA-404
JSP	1.2–2.1
Kotlin	1.3–2.1
MXML (Flex®)	4
Objective-C/C++	2.0 (「コンパイラ」を参照)
PHP	7.3–8.4

言語/フレームワーク	検証済みの互換性
PL/SQL	8–23
Python®	2.6–3.13
Ruby	1.x
Scala	2.11–2.13, 3.3–3.6
Solidity	0.4.12–0.8.21
Swift®	5.10, 6.0 - 6.2. (サポートされているswiftcバージョンについては、「コンパイラ」を参照)
T-SQL	SQL Server 2005, 2008, 2012
TypeScript	3.6–5.4
VBScript	2.0, 5.0
Visual Basic (VB.NET)	15.0–16.9
Visual Basic	6.0
XML	1.0, 1.1
YAML	1.2

1.4.4.1. ライブラリ、フレームワーク、およびテクノロジ

OpenText SASTは、このセクションに記載されているライブラリ、フレームワーク、および技術をサポートしており、専用のFortify Secure Coding Rulepacksと、主要なサポート対象言語を超えた脆弱性カバレッジを提供しています。

Java

Adobe Flex Blaze DS	Apache Slide	iBatis	Mozilla Rhino	Spring AI
Ajanta	Apache Spring Security (Acegi)	IBM MQ	MyBatis	Spring MVC
Amazon Web Services (AWS) SDK	Apache Struts	IBM WebSphere	MyBatis-Plus	Spring Boot
Android	Apache Tapestry	Jackson	Netscape LDAP API	Spring Data Commons
Android Jetpack	Apache Tomcat	Jakarta Activation	OkHttp	Spring Data JPA
Apache Axiom	Apache Torque	Jakarta EE (Java EE)	OpenCSV	Spring Data MongoDB
Apache Axis	Apache Util	Jasypt	Oracle Application Development Framework (ADF)	Spring Data Redis
Apache Beam	Apache Velocity	Java Annotations	Oracle BC4J	Spring for GraphQL
Apache Beehive NetUI	Apache Wicket	Java Excel API	Oracle JDBC	Spring HATEOAS
Apache Catalina	Apache Xalan	JavaMail	Oracle OA Framework	Spring JMS
Apache Cocoon	Apache Xerces	JAX-RS	Oracle tcDataSet	Spring JMX
Apache Commons	ATG Dynamo	JAXB	Oracle XML Developer Kit (XDK)	Spring Messaging
Apache ECS	Azure SDK	Jaxen	OWASP Enterprise Security API (ESAPI)	Spring Security
Apache Hadoop	Castor	JBoss	OWASP HTML Sanitizer	Spring Webflow
Apache HttpComponents	表示タグ	JDesktop	OWASP Java Encoder	Spring WebSockets
Apache Jasper	Dom4j	JDOM		Spring WS
	GDS AntiXSS	Jetty		Stripes
	Google Cloud	JGroups		Sun JavaServer Faces (JSF)
	Google Dataflow	json-simple		
		JTidy Servlet		
		JXTA		
		JYaml		
		Liferay Portal		

Apache Log4j Apache Lucene Apache MyFaces Apache OGNL Apache ORO Apache POI Apache SLF4J	Google Guava Google Web Toolkit gRPC Gson Hibernate	MongoDB	Plexus Archiver Realm Restlet SAP Web Dynpro Saxon SnakeYAML Spring	Tungsten Weblogic WebSocket XStream YamlBeans ZeroTurnaround ZIP Zip4J

Kotlin

Kotlinのサポートには、Java用にカバーされるすべてのライブラリと、次のKotlinライブラリが含まれています。

Kotlin標準ライブラリ	Android KTX	OkHttp	

Scala

Scalaのサポートには、Java用にカバーされるすべてのライブラリと、次のScalaライブラリが含まれています。

Akka HTTP Scala Play	Scala Slick	

.NET

.NET Framework、.NET Core、および.NET Standard .NET WebSockets ADO.NET Entity Framework ADODB Amazon Web Services (AWS) SDK ASP.NET MVC ASP.NET SignalR ASP.NET Web API	Azure SDK Castle ActiveRecord CsvHelper Dapper DB2 .NET Provider DotNetZip Entity Framework Entity Framework Core fastJSON gRPC	Hot Chocolate IBM Informix .NET Provider Json.NET Log4Net Microsoft ApplicationBlocks Microsoft My Framework Microsoft Practices Enterprise Library Microsoft Web Protection Library	MongoDB MySQL Connector/NET NHibernate NLog Npgsql Open XML SDK Oracle Data Provider for .NET OWASP AntiSamy Saxon	SharePoint Services SharpCompress SharpZipLib SQLite .NET Provider SubSonic Sybase ASE ADO.NET Data Provider Xamarin Xamarin Forms YamlDotNet

C

ActiveDirectory LDAP Apple System Logging (ASL)	CURL Library GLib JNI	MySQL Netscape LDAP ODBC	OpenSSL POSIX Threads SQLite	Sun RPC WinAPI

C++

Boost Smart Pointers MFC	STL WMI	

SQL

Oracle ModPLSQL

PHP

ADOdb Advanced PHP Debugging CakePHP PHP Debug	PHP DOM PHP Extension PHP Hash PHP JSON PHP Mcrypt	PHP Mhash PHP Mysql PHP OCI8 PHP OpenSSL PHP PostgreSQL	PHP Reflection PHP Simdjson PHP SimpleXML PHP Smarty PHP Sodium	PHP WordPress PHP XML PHP XMLReader PHP Zend PHP Zip

JavaScript/TypeScript/HTML5

Angular Anthropic Claude Apollo Server Bluebird child-process-promise Express	Gemini API GraphQL.js Handlebars Helmet iOS JavaScript Bridge jQuery	JS-YAML LangChain Mustache Node.js Azure Storage Node.js Core OpenAI	React React Native React Native Async Storage React Router SAPUI5/OpenUI5	Sequelize Underscore.js Vertex AI Vue

Python

aiopg Amazon Web Services (AWS) Lambda Amazon SageMaker Anthropic Claude Azure Functions boto3 Django Flask	Google Cloud Graphene gRPC httplib2 Jinja2 LangChain libxml2 lxml	memcache-client _mysql MySQL Connector/Python MySQLdb OpenAI oslo.config pandas Paramiko	psycopg2 pycrypto PyCryptodome pycurl pylibmc PyMongo PySpark PyYAML	requests simplejson six TensorFlow Twisted Mail urllib3 Vertex AI WebKit

Ruby

MySQL pg	Rack SQLite	Thor	

Objective-C

AFNetworking Apple AddressBook Apple AppKit Apple CFNetwork Apple ClockKit Apple CommonCrypto Apple CoreData	Apple CoreFoundation Apple CoreLocation Apple CoreServices Apple CoreTelephony Apple Foundation Apple HealthKit	Apple LocalAuthentication Apple MessageUI Apple Security Apple Social Apple UIKit	Apple WatchConnectivity Apple WatchKit Apple WebKit Hpple Objective-Zip Realm	SBJson SFHFKeychainUtils SSZipArchive ZipArchive ZipUtilities ZipZap

Swift

Alamofire Apple AddressBook Apple CFNetwork Apple ClockKit Apple CommonCrypto Apple CoreData	Apple CoreFoundation Apple CoreLocation Apple Foundation Apple HealthKit Apple LocalAuthentication	Apple MessageUI Apple Security Apple Social Apple SwiftUI Apple UIKit Apple WatchConnectivity	Apple WatchKit Apple WebKit Hpple Realm SQLite SSZipArchive	Zip ZipArchive ZIPFoundation ZipUtilities ZipZap

COBOL

Auditor CICS DLI	Micro Focus COBOL Run-time System MQ	POSIX SQL	

Go

GORM logrus gRPC	

Dart

Flutter	

Configuration

.NET Configuration Adobe Flex (ActionScript) Configuration Ajax Frameworks Amazon Web Service (AWS) Ansible AWS CloudFormation Azure Resource Manager (ARM) Build Management	Docker Configuration (Dockerfiles) GitHub Actions Google Android Configuration iOS Property List J2EE Configuration Java Apache Axis Java Apache Log4j Configuration Java Apache Spring Security (Acegi)	Java Apache Struts Java Apache Tomcat Configuration Java Blaze DS Java Hibernate Configuration Java iBatis Configuration Java IBM WebSphere Java MyBatis Configuration	Java OWASP AntiSamy Java Spring and Spring MVC Java Spring Boot Java Spring Mail Java Spring Security Java Spring WebSockets Java Weblogic Kubernetes Mule	OpenAPI Specification Oracle Application Development Framework (ADF) PHP Configuration PHP WordPress Silverlight Configuration Terraform (AWS, Azure, GCP) WS-SecurityPolicy XML Schema

コードとしてのインフラストラクチャ: Amazon Web Services

API Gateway App Mesh AppSync Athena Aurora Backup Batch Certificate Manager CloudFormation CloudFront CloudTrail CloudWatch CodeBuild CodeCommit CodeStar Cognito	Config Configuration Recorder Database Migration Service (DMS) DataSync DocumentDB DynamoDB EC2 Elastic Block Store (EBS) Elastic Container Registry (ECR) Elastic Container Service (ECS) Elastic File System (EFS) Elastic Kubernetes Service (EKS)	Elastic Load Balancing (ELB) ElastiCache EMR FinSpace FSx Global Accelerator Glue GuardDuty HealthLake Identity and Access Management (IAM) Image Builder Key Management Service (KMS) Kinesis Kinesis Video Streams	Lightsail Location Service Lookout for Equipment Mainframe Modernization Managed Streaming for Apache Kafka (MSK) MemoryDB for Redis MQ Neptune OpenSearch Service Quantum Ledger Database (QLDB) RDS Redshift Rekognition Route 53	SageMaker Secrets Manager Simple Notification Service (SNS) Simple Queue Service (SQS) Simple Storage Service (S3) Step Functions Systems Manager Timestream Transfer Family VPC VPC Lattice WorkSpaces Family

コードとしてのインフラストラクチャ: Microsoft Azure

App Service Application Gateway Automation Microsoft Entra Domain Services Azure Health Data Services Azure Kubernetes Service (AKS) Batch Blob Storage	Cache for Redis Cognitive Search Container Registry Cosmos DB Database for MariaDB Database for MySQL Database for PostgreSQL Databricks Data Box	Data Factory Defender for Cloud Event Hubs Front Door Grafana Hostname Binding IoT Central IoT Hub Key Vault Logic Apps	Machine Learning MariaDB Media Services Monitor NetApp Files Private Cloud ポリシー Portal SignalR Service	Site Recovery Spring Apps SQL Storage Accounts Virtual Machine Scale Sets Virtual Machines Web PubSub

コードとしてのインフラストラクチャ: Google Cloud

Access Context Manager AlloyDB Apigee API Management App Engine Artifact Registry	Backup for GKE BigQuery Cloud Bigtable Cloud DNS Cloud Functions Cloud Key Management	Cloud Load Balancing Cloud Logging Cloud Spanner Cloud SQL Cloud Storage Compute Engine	Filestore Google Cloud Platform Google Kubernetes Engine (GKE) Identity and Access Management (IAM)	Media CDN Memorystore Pub/Sub Secret Manager Workflows

Secrets

.netrc	Defined	HashiCorp (Terraform, Vault)	New Relic	Sendbird
1Password	DES	Heroku	npm	SendGrid
Actually Good Encryption (AGE)	DigitalOcean	HexChat	NuGet	Sentry
Adafruit	Docker	HubSpot	Okta	SHA1
Adobe	Doppler	Intercom	OpenVPN	SHA256
Airtable	Droneci	Java	Password in comment	SHA512
Algolia	Dropbox	JFrog (Artifactory)	Password in connection string	Shippo
Alibaba (Aliyun)	Duffel	JSON Web Token	Password in PowerShell script	Shopify
Amazon (AWS, MWS)	Dynatrace	KDE Wallet (Kwallet)	Password in URI	Sidekiq
Apple (macOS)	EasyPost	KeePass	Password Safe	Slack
Apache HTTP	Encryption key	Kraken	PayPal (Braintree)	SonarQube
Asana	Etsy	Kucoin	Pidgin	Square
Atlassian	Facebook	LaunchDarkly	Plaid	Squarespace
Authress	Fastly	Linear	Planetscale	StackHawk
基本アクセス認証	Finicity	LinkedIn	PostgreSQL	Stripe
bcrypt	Finnhub	Lob	Postman	Sumologic
Beamer	Flickr	Mailchimp	Prefect	Telegram
Bearer token	Flutterwave	Mailgun	Pulumi	Travis
Bitbucket	Frame.io	Mapbox	PuTTY	Trello
Bittrex	Freshbooks	Mattermost	PyPI	Twilio
Brevo (Sendinblue)	Git	MD5	RapidAPI	Twitch
	GitHub	MessageBird		Twitter
	GitLab			Typeform
	Gitter			Yandex
	GNOME			Zendesk

Clojars Code Climate Codecov Coinbase Confluent Contentful Databricks Datadog	GNU (Bash) GoCardless Google (API, Google Cloud, OAuth) Grafana	Microsoft (Azure App Storage、Cosmos DB、Functionsと Bitlocker、PowerShell、RDP、VBScript) Microsoft (Outlook) Mutt MySQL Netlify	Readme RSA Security Ruby (Ruby on Rails, RubyGems) Sauce Labs Secret key Secure Shell Protocol (SSH)	

1.4.5. サポートされるビルドツール

OpenText SASTは、次の表に示すビルドツールをサポートしています。

ビルドツール	バージョン	メモ
Apache Ant™	1.10.x	
Bazel	6.x–7.x	Bazel統合はJavaとPythonをサポートしています。
dotnet	6.0–10.x	
Gradle (ビルド統合)	6.6–8.10	OpenText SAST Gradle統合は、Java、Kotlin、およびC/C++をサポートしています。
Gradle (Gradleプラグイン)	6.6–8.5	OpenText SAST Gradle Pluginは、JavaおよびKotlinをサポートしています。
Apache Maven™ Software	3.6.x, 3.8.x, 3.9.x	
MSBuild	14.x–17.14	OpenText SAST MSBuild 17.4統合は、.NET 7.0以降および.NET Framework 4.7.2以降と互換性があります。
xcodebuild	15.3-15.4, 16-16.4, 26	

1.4.6. サポートされているコンパイラ

OpenText SASTは、次の表に示すコンパイラをサポートしています。

コンパイラ	バージョン	オペレーティングシステム
gcc	GNU gcc 6.x– 13	Windows, Linux, macOS
	GNU gcc 4.9–5.x	Windows, Linux, macOS, AIX
g++	GNU g++ 6.x– 13	Windows, Linux, macOS
	GNU g++ 4.9–5.x	Windows, Linux, macOS, AIX
OpenJDK javac	9, 10, 11, 12, 13, 14, 17, 21, 24, 25	Windows, Linux, macOS, AIX
Oracle javac	7, 8, 9	Windows, Linux, macOS
cl (MSVC)	2015, 2017, 2019, 2022	Windows
Clang	15.0.0, 16.0.0, 17.0.0	macOS
Swiftc	5.10, 6.0, 6.0.2, 6.0.3[1], 6.1.0, 6.1.2, 6.2	macOS

[1]OpenText SASTは、次のXcodeバージョンで構築されたアプリケーションをサポートします: 15.3~15.4、16~16.4、26。

1.4.7. OpenText Application Security Content

Fortify Secure Coding Rulepacksは、サポートされているすべてのバージョンのOpenText SASTと後方互換性があります。これにより、Rulepackを更新しても、稼働しているOpenText SASTのインストールが壊れることはありません。

1.4.8. 仮想マシンのサポート

OpenText Application Security Software製品は、仮想マシン環境で承認済みのオペレーティングシステムで実行できます。最小限のハードウェア要件を満たす専用のCPUおよびメモリリソースを提供する必要があります。推奨される処理リソース、メモリリソース、およびディスクリソースで、ネイティブ環境では再現できない問題を見つけた場合は、仮想環境のプロバイダと一緒に作業して問題を解決する必要があります。

Note

VM環境でOpenText Application Security Software製品を実行する場合は、パフォーマンスの低下を避けるため、CPUおよびメモリリソースをVMに完全にコミットしておくことを強く推奨します。

1.4.9. ソフトウェアの取得

OpenText SAST(Static Code Analyzer)は、電子的なダウンロードとして利用できます。ソフトウェアライセンスおよびダウンロード(SLD)ポータルからソフトウェアをダウンロードする方法については、［**お問い合わせ先/セルフヘルプ(Contact Us / Self Help)**］をクリックしてビデオと『*クイックスタートガイド*』を確認してください。

次の表に、使用可能なパッケージを一覧表示してそれぞれの内容を説明します。

File Name	説明(Description)
OpenText_SAST_Fortify_Windows_<*version*>.zip	Windows用のOpenText SASTパッケージ このパッケージには次の内容が含まれています。 • 次のコンポーネントを含む、OpenText SASTインストーラ • Fortify License and Infrastructure Managerインストーラ • OpenText SASTカスタムルールガイドバンドル • OpenText Application Security Softwareについて **Note** OpenText Application Security Content (ルールパックおよび外部メタデータ)は、インストール時にダウンロードできます。
OpenText_SAST_Fortify_Windows_<*version*>.zip.sig	OpenText SAST Windowsパッケージの署名ファイル

File Name	説明(Description)
OpenText_SAST_Fortify_Linux-ARM_<*version*>.tar.gz	ARM上のLinux用のOpenText SASTパッケージ このパッケージには次の内容が含まれています。 • 次のコンポーネントを含む、OpenText SASTインストーラ • OpenText SASTカスタムルールガイドバンドル • OpenText Application Security Softwareについて **Note** OpenText Application Security Content (ルールパックおよび外部メタデータ)は、インストール時にダウンロードできます。
OpenText_SAST_Fortify_Linux-ARM_<*version*>.tar.gz.sig	ARMパッケージ上のOpenText SAST Linuxの署名ファイル

File Name	説明(Description)
OpenText_SAST_Fortify_Linux_<*version*>.tar.gz	Linux用のOpenText SASTパッケージ このパッケージには次の内容が含まれています。 • 次のコンポーネントを含む、OpenText SASTインストーラ • OpenText SASTカスタムルールガイドバンドル • OpenText Application Security Softwareについて **Note** OpenText Application Security Content (ルールパックおよび外部メタデータ)は、インストール時にダウンロードできます。
OpenText_SAST_Fortify_Linux_<*version*>.tar.gz.sig	OpenText SAST Linuxパッケージの署名ファイル

File Name	説明(Description)
OpenText_SAST_Fortify_Mac_<*version*>.tar.gz	macOS用のOpenText SASTパッケージ このパッケージには次の内容が含まれています。 • 次のコンポーネントを含む、OpenText SASTインストーラ • OpenText SASTカスタムルールガイドバンドル • OpenText Application Security Softwareについて **Note** OpenText Application Security Content (ルールパックおよび外部メタデータ)は、インストール時にダウンロードできます。
OpenText_SAST_Fortify_Mac_<*version*>.tar.gz.sig	OpenText SAST macOSパッケージの署名ファイル

File Name	説明(Description)
OpenText_SAST_Fortify_Mac-ARM_<*version*>.tar.gz	macOS-ARM用のOpenText SASTパッケージ このパッケージには次の内容が含まれています。 • 次のコンポーネントを含む、OpenText SASTインストーラ • OpenText SASTカスタムルールガイドバンドル • OpenText Application Security Softwareについて **Note** OpenText Application Security Content (ルールパックおよび外部メタデータ)は、インストール時にダウンロードできます。
OpenText_SAST_Fortify_Mac-ARM_<*version*>.tar.gz.sig	OpenText SAST macOS-ARMパッケージの署名ファイル
OpenText_SAST_Fortify_AIX_<*version*>.tar.gz	AIX用のOpenText SASTパッケージ このパッケージには次の内容が含まれています。 • OpenText SASTインストーラ • OpenText SASTカスタムルールガイドバンドル • OpenText Application Security Softwareについて
OpenText_SAST_Fortify_AIX_<*version*>.tar.gz.sig	OpenText SAST AIXパッケージの署名ファイル

File Name	説明(Description)
OpenText_SAST_Fortify_Samples_<version>.zip	OpenText SASTの使用法を学ぶのに役立つコードサンプル
OpenText_SAST_Fortify_Samples_<version>.zip.sig	OpenText SASTコードサンプルの署名ファイル

1.4.10. ソフトウェアのダウンロードの確認

このトピックでは、カスタマサポートWebサイトからダウンロードした署名ファイルのデジタル署名を検証する方法について説明します。検証により、ダウンロードしたパッケージが署名されてサイトにポストされた後に改ざんされていないことが保証されます。検証に進む前に、OpenText Application Security Software製品ファイルおよび関連する署名(`*.sig`)ファイルをダウンロードします。ソフトウェアを使用するためにパッケージを検証する必要はありません。ただし、セキュリティ上の理由から、組織で必要になる場合があります。

デジタル署名検証のためのシステムの準備

> **Note**
> 次の手順では、サードパーティ製品について説明していますが、使用している特定のサポート対象バージョンとは一致しない場合があります。ご使用のバージョンの手順については、製品ドキュメントを参照してください。

電子メディア検証用にシステムを準備するには、次の手順を実行します。

1. GnuPGのWebサイトに移動します。
2. GnuPG Privacy Guardをダウンロードしてインストールします。
3. 次のように、秘密鍵を生成します。
    1. 次のコマンドを実行します(Windowsシステムでは、 `$` プロンプトなしでコマンドを実行します)。

        ```
        $ gpg --gen-key
        ```

    2. キータイプのプロンプトが表示されたら、［ `DSA and Elgamal` ］を選択します。
    3. キーサイズのプロンプトが表示されたら、［ `2048` ］を選択します。
    4. キーが有効である必要がある時間の長さのプロンプトが表示されたら、［ `key does not expire` ］を選択します。
    5. ユーザ識別の質問に回答し、秘密鍵を保護するパスフレーズを入力します。
4. OpenText GPG公開鍵(圧縮されたtarファイル)を https://mysupport.microfocus.com/documents/10180/0/MF_public_keys.tar.gzからダウンロードします。
5. 公開鍵を抽出します。
6. 次のコマンドを使用して、ダウンロードした各キーをGnuPGでインポートします。

```
gpg --import <path_to_key>|<key_file>
```

1.5. OpenText SASTのインストール

このセクションでは、OpenText SAST (Fortify Static Code Analyzer)をインストールおよびアンインストールする方法について説明します。このセクションでは、インストール後の基本的なタスクについても説明します。システムがハードウェアおよびソフトウェアの最小要件を満たしていることを確認するために、「システム要件」を参照してください。

このセクションでは、次のトピックについて説明します。

1.5.1. OpenText SASTのインストールについて

このセクションでは、OpenText SASTをインストールする方法について説明します。いくつかのコマンドラインツールがOpenText SASTと一緒に自動的にインストールされます(「コマンドラインツール」を参照)。オプションで、ScanCentral SASTクライアントとApplication Security fortifyclientユーティリティをOpenText SASTのインストールに含めることができます。ScanCentral SASTの詳細については、『*OpenText™ ScanCentral SASTインストール、設定、および使用ガイド*』を参照してください。

インストール用のFortifyライセンスファイルと、オプションでLIMライセンスプール資格情報を指定する必要があります。次の表に、OpenText SASTをインストールする方法を示します。

インストール方法	手順
標準インストールウィザードを使用してインストールを実行する	OpenText SASTおよびアプリケーションのインストール
サイレント(無人)でインストールを実行する	OpenText SASTのサイレントインストール
Windows以外のシステムでテキストベースのインストールを実行する	Windows以外のプラットフォームでテキストベースモードでのOpenText SASTとアプリケーションのインストール
Dockerを使用してインストールを実行する	Dockerを使用したOpenText SASTのインストールと実行

パフォーマンスを最適にするには、スキャンするコードが存在するローカルファイルシステムにOpenText SASTをインストールします。

Note

Windows以外のシステムでは、書き込み権限があるホームディレクトリを持っているユーザとしてOpenText SASTをインストールする必要があります。ホームディレクトリを持ってないルート以外のユーザとしてOpenText SASTをインストールしないでください。

インストールが完了したら、「インストール後のタスクについて」を参照して、システムのセットアップを完了するために実行できる追加のステップを確認します。また、インス

トールされた環境設定ファイルを更新して、ランタイム分析、出力、およびOpenText SASTのパフォーマンスを設定することもできます。OpenText SASTの環境設定オプションについては、「環境設定オプション」を参照してください。

1.5.1.1. OpenText SASTのインストール

OpenText SASTをインストールするには:

1. オペレーティングシステムのインストーラファイルを実行して、OpenText SASTセットアップウィザードを起動します。
   - Windows: `OpenText_SAST_Fortify_windows-x64_<version>.exe`
   - Linux: `OpenText_SAST_Fortify_linux-x64_<version>.run` または `OpenText_SAST_Fortify_linux-arm64_<version>.run`
   - macOS: `OpenText_SAST_Fortify_osx-x64_<version>.app.zip` または `OpenText_SAST_Fortify _osx-arm64.app.zip`

     APPインストーラファイルを実行する前に、ZIPファイルを展開します。
   - AIX: `OpenText_SAST_Fortify_aix-ppc64_<version>.run`

   ここで、*<version>*はソフトウェアリリースのバージョンです。次に、［次へ(Next)］をクリックします。
2. ライセンス契約を確認して受諾し、［**次へ(Next)**］をクリックします。
3. (オプション)インストールするコンポーネントを選択して、［次へ(Next)］をクリックします。
4. 一部のタイプのプロジェクト分析に必要な最小限のソフトウェアがシステムに含まれていないことがインストーラによって検出された場合、不足している要件と、その要件を必要とするプロジェクトが「**システム要件**」ページに表示されます。［次へ(Next)］をクリックします。

   すべてのソフトウェア要件については、「ソフトウェア要件」を参照してください。
5. OpenText SASTのインストール先を選択し、［**次へ(Next)**］をクリックします。

   手順3で、ScanCentral SASTクライアントをインストールに含めることを選択した場合は、パスにスペースを含まない場所を指定する必要があります。

   > **Important**
   >
   > OpenText™ Application Security Toolsのインストール先と同じディレクトリにOpenText SASTをインストールしないでください。
6. `fortify.license` ファイルへのパスを指定し、［**次へ(Next)**］をクリックします。

7. (オプション)［**LIMライセンス**］ページで［**はい**］を選択して、Fortify LIM (License and Infrastructure Manager)の同時ライセンスを管理し、［**次へ**］をクリックします。

   Note

   OpenText SASTでライセンスが必要なタスクが実行されたら、アプリケーションでライセンスプールからのLIMリースの取得が試みられます。LIMサーバとの通信上の問題があるためにOpenText SASTでライセンスの取得に失敗した場合は、Fortifyのライセンスファイルが使用されます。この動作を変更するには、`com.fortify.sca.lim.WaitForInitialLicense`ファイル内で`fortify-sca.properties`を使用します(「LIMライセンスのプロパティ」を参照)。

   1. LIM APIのURL、ライセンスプール名、およびライセンスプールパスワードを入力します。
   2. ［次へ(Next)］をクリックします。

      ［**LIM Proxy Settings**］ページが開きます。
   3. LIMサーバへの接続にプロキシサーバが必要な場合は、プロキシホスト(プロキシサーバのホスト名またはIPアドレス)とポート番号(省略可能)を入力します。
   4. ［次へ(Next)］をクリックします。

8. インストールのセキュリティコンテンツを更新するには:

   Note

   インストール時にインターネットにアクセスできない展開環境の場合、fortifyupdateコマンドラインツールを使用してセキュリティコンテンツを更新できます。「OpenText Application Security Contentの手動インストール」を参照してください。

   1. 更新サーバのWebアドレスを入力します。

      セキュリティコンテンツの更新にFortify Rulepack更新サーバを使用するには、Webアドレスを`https://update.fortify.com`のままにします。Application Securityをアップデートサーバとして使用することもできます。
   2. (オプション)更新サーバへの接続にプロキシサーバが必要な場合は、プロキシホストとポート番号を入力します。

3. セキュリティコンテンツを手動で更新する場合は、［**インストール後にセキュリティコンテンツを更新する(Update security content after installation)**］チェックボックスをオフにします。

4. ［次へ(Next)］をクリックします。

9. システム上の以前のインストールから移行するかどうかを指定します。

   以前のインストールから移行すると、OpenText SASTアーティファクトファイルが保持されます。詳細については、「OpenText SASTのアップグレードについて」を参照してください。

   > **Note**
   >
   > `scapostinstall` コマンドラインツールを使用してアーティファクトを移行することもできます。インストール後のツールを使用して以前のインストールから移行する方法については、「プロパティファイルの移行」を参照してください。

   以前のインストールからアーティファクトを移行するには、次の手順に従います。

   1. ［**OpenText SAST（Fortify) の移行(Migration)**］ページで［**はい(Yes)**］を選択して、［**次へ(Next)**］をクリックします。
   2. システム上の既存のインストール場所を指定し、［**次へ(Next)**］をクリックします。

   以前のリリースからのアーティファクトのマイグレーションをスキップするには、移行の選択を［**いいえ(No)**］のままにして、［**次へ(Next)**］をクリックします。

10. ［**インストール準備(Ready to Install)**］ページで［**次へ(Next)**］をクリックして、OpenText SAST、選択したコンポーネント、およびOpenText Application Security Contentをインストールします。

    セキュリティコンテンツの更新を選択した場合は、［セキュリティコンテンツ更新の結果(Security Content Update Result)］ウィンドウにセキュリティコンテンツ更新の結果が表示されます。

11. ［**完了(Finish)**］をクリックして、セットアップウィザードを閉じます。

1.5.1.2. OpenText SASTのサイレントインストール

サイレントインストールを使用すると、ユーザプロンプトを表示せずにインストールを完了できます。サイレントでインストールするには、インストーラに必要な情報を提供するオプションファイルを作成する必要があります。サイレントインストールを使用して、複数のコンピュータにインストールパラメータを複製できます。

Important

OpenText™ Application Security Toolsのインストール先と同じディレクトリにOpenText SASTをインストールしないでください。

OpenText SASTをサイレントでインストールすると、デフォルトでは、インストーラによってApplication Securityがダウンロードされません。オプションファイルでOpenText Application Security Contentのダウンロードを有効にするか、OpenText Application Security Contentを手動でインストールすることができます(OpenText Application Security Contentの手動インストールを参照)。

OpenText SASTをサイレントでインストールするには:

1. オプションファイルを作成します。
    1. 次の行を含むテキストファイルを作成します。

        ```
        fortify_license_path=<license_file_location>
        ```

        ここで、*<license_file_location>*は`fortify.license`ファイルへのフルパスです。

    2. LIMライセンスサーバを使用するには、LIMライセンスプール資格情報を含む次の行をオプションファイルに追加します。

        ```
        lim_url=<lim_url>lim_pool_name=
        <license_pool_name>lim_pool_password=<license_pool_pwd>
        ```

    3. OpenText Application Security Contentの更新で、デフォルトの`https://update.fortify.com`とは異なる場所を使用するには、次の行を追加します。

```
update_server=<update_server_url>
```

4. OpenText Application Security Contentのダウンロードにプロキシサーバが必要な場合は、次の行を追加します。

   ```
   update_proxy_server=<proxy_server>update_proxy_port=
   <port_number>
   ```

5. OpenText Application Security Contentのダウンロードを有効にするには、次の行を追加します。

   update_security_content=1

6. 必要に応じて、オプションファイルにインストール指示を追加します。

   オプションファイルに追加できるインストールオプションのリストを取得するには、コマンドプロンプトを開き、インストーラファイル名と `--help` オプションを入力します。このコマンドでは、使用可能な各コマンドラインオプションの前に二重ダッシュが付けられ、使用可能なパラメータが角括弧で囲まれます。たとえば、インストールの進行状況をコマンドラインに表示する場合は、オプションファイルに `unattendedmodeui=minimal` を追加します。

   メモ:

   - コマンドラインオプションでは大文字と小文字が区別されます。
   - インストールオプションは、サポートされているすべてオペレーティングシステムで同じではあるとは限りません。 `--help` を使用してインストーラを実行して、オペレーティングシステムで使用可能なオプションを確認します。

   次のWindowsオプションファイルの例では、ライセンスファイルの場所、Application Securityサーバの場所とOpenText Application Security Contentを取得するためのプロキシ情報、以前のリリースからの移行要求、およびOpenText SASTのインストールディレクトリの場所を指定しています。

   ```
   fortify_license_path=C:\Users\admin\Desktop\fortify.lice
   nse update_server=https://my_ssc_host:8080/ssc
   update_proxy_server=webproxy.abc.company.com
   update_proxy_port=8080 migrate_sca=1
   install_dir=C:\Fortify
   ```

次のオプションファイルの例は、LinuxおよびmacOS®用です。

```
fortify_license_path=/opt/Fortify/fortify.license
update_server=https://my_ssc_host:8080/ssc
update_proxy_server=webproxy.abc.company.com
update_proxy_port=8080 migrate_sca=1
install_dir=/opt/Fortify
```

2. オプションファイルを保存します。
3. オペレーティングシステムのサイレントインストールコマンドを実行します。

> **Note**
>
> 場合によっては、インストーラを実行する前に、管理者としてコマンドプロンプトを実行する必要があります。

Windows	`OpenText_SAST_Fortify_windows-x64_<version>.exe --mode unattended --optionfile <full_path_to_options_file>`

Linux	`./OpenText_SAST_Fortify_linux-x64_<version>.run --mode unattended --optionfile <full_path_to_options_file>`
	または
	`./OpenText_SAST_Fortify_linux-arm64_<version>.run --mode unattended --optionfile <full_path_to_options_file>`

macOS®	コマンドを実行する前に、ZIPファイルを展開する必要があります。
	`OpenText_SAST_Fortify_osx-x64_<version>.app/Contents/MacOS/installbuilder.sh --mode unattended --optionfile <full_path_to_options_file>`
	または
	`OpenText_SAST_Fortify_osx-arm64_<version>.app/Contents/MacOS/installbuilder.sh --mode unattended --optionfile <full_path_to_options_file>`

AIX	`./OpenText_SAST_Fortify_aix-ppc64_<version>.run --mode unattended --optionfile <full_path_to_options_file>`

インストールが完了すると、インストーラのログファイルが作成されます。このログファイルは、オペレーティングシステムに応じて次の場所に保存されます。

Windows	`C:\Users\<username>\AppData\Local\Temp\OpenTextSASTFortify-<version>-install.log`
Windows以外	`/tmp/OpenTextSASTFortify-<version>-install.log`

1.5.1.3. Windows以外のプラットフォームでテキストベースモードでのOpenText SASTのインストール

コマンドラインでテキストベースのインストールを実行します。インストール時に、インストールを完了するために必要な情報の入力を求めるプロンプトが表示されます。Windowsシステムではテキストベースのインストールがサポートされていません。

Important

OpenText™ Application Security Toolsのインストール先と同じディレクトリにOpenText SASTをインストールしないでください。

OpenText SASTのテキストベースのインストールを実行するには、次の表に示すように、オペレーティングシステム用のテキストベースのインストールコマンドを実行します。

Linux	`./OpenText_SAST_Fortify_linux-x64_<version>.run --mode text`
	または
	`./OpenText_SAST_Fortify_linux-arm64_<version>.run --mode text`

MacOS	コマンドを実行する前に、提供されたZIPファイルを展開する必要があります。
	`OpenText_SAST_Fortify_osx-x64_<version>.app/Contents/MacOS/installbuilder.sh --mode text`
	または
	`OpenText_SAST_Fortify_osx-arm64_<version>.app/Contents/MacOS/installbuilder.sh --mode text`

AIX	`OpenText_SAST_Fortify_aix-ppc64_<version>.run --mode text`

1.5.1.4. OpenText Application Security Contentの手動インストール

OpenText Application Security Content (Fortify Secure Coding Rulepacksとメタデータ)は、インストール時に自動的にインストールできます。ただし、Fortify Rulepack更新サーバからOpenText Application Security Contentをダウンロードしてから、fortifyupdateコマンドラインツールを使用してインストールすることもできます。このオプションは、インストール時にインターネットにアクセスできない展開環境のために提供されています。

リモートサーバまたはローカルにダウンロードされたファイルからOpenText Application Security Contentをインストールするには、fortifyupdateを使用します。

セキュリティコンテンツをインストールするには、次の手順に従います。

1. コマンドウィンドウを開き、 *<sast_install_dir>*/bin/ に移動します。
2. コマンドプロンプトで「 fortifyupdate 」と入力します。

   以前にFortify Rulepack更新サーバからOpenText Application Security Contentをダウンロードした場合は、 fortifyupdate オプションと、ZIPファイルをダウンロードしたディレクトリへのパスを使用して -import を実行します。

この同じツールを使用して、OpenText Application Security Contentを更新することもできます。fortifyupdateコマンドラインツールの詳細については、「セキュリティコンテンツの更新」を参照してください。

1.5.2. Dockerを使用したOpenText SASTのインストールと実行

DockerイメージにOpenText SASTをインストールしてから、OpenText SASTをDockerコンテナとして実行できます。

Note

OpenText SASTは、Dockerでサポートされ ているLinuxプラットフォームでのみ実行できます。

このセクションでは、次のトピックについて説明します。

1.5.2.1. OpenText SASTをインストールするDockerfileの作成

このトピックでは、DockerイメージでOpenText SASTをインストールするDockerfileを作成する方法について説明します。

Dockerfileには、次の命令が含まれている必要があります。

1. ベースイメージに使用されるLinuxシステムを設定します。

   サポートされているプラットフォームとアーキテクチャの詳細については、「サポートされるプラットフォームとアーキテクチャ」を参照してください。

   > **Note**
   >
   > OpenText SASTの実行時にビルドツールを使用する場合は、イメージに必要なビルドツールがインストールされていることを確認します。サポートされているビルドツールの使用については、「サポートされるビルドツール」を参照してください。

2. COPY命令を使用して、OpenText SASTインストーラ、Fortifyライセンスファイル、およびインストールオプションファイルをDockerイメージにコピーします。

   インストールオプションファイルの作成方法については、「OpenText SASTのサイレントインストール」を参照してください。

3. RUN命令を使用してOpenText SASTインストーラを実行します。

   インストーラは無人モードで実行する必要があります。詳細については、「OpenText SASTのサイレントインストール」を参照してください。

4. RUN命令を使用して、runifyupdateを実行してOpenText Application Security Contentをインストールします。

> **Important**
>
> OpenText SASTでプロジェクトの分析を実行するには、OpenText Application Security Contentのインストールが必要です。次の例では、イメージの構築中にダウンロードしたローカルファイルから、OpenText Application Security Contentをインストールします。updateifyupdateツールを使用したOpenText Application Security Contentのダウンロードとインストールの詳細については、「OpenText Application Security Contentの手動インストール」を参照してください。

5. OpenText SASTを実行できるようにイメージを設定するには、ENTRYPOINT命令を使用して、インストールされたsourceanalyzer実行可能ファイルの場所にエントリポイントを設定します。

   sourceanalyzerのデフォルトのインストールパスは `/opt/Fortify/OpenText_SAST_Fortify_<version>/bin/sourceanalyzer` です。

OpenText SASTをインストールするDockerfileの例は、次のとおりです。

```
FROM ubuntu:18.04 WORKDIR /app ENV APP_HOME="/app" ENV
RULEPACK="MyRulepack.zip" COPY fortify.license ${APP_HOME} COPY
OpenText_SAST_Fortify_linux-x64_25.4.0.run ${APP_HOME} COPY
optionFile ${APP_HOME} COPY ${RULEPACK} ${APP_HOME} RUN
./OpenText_SAST_Fortify_linux-x64_25.4.0.run --mode unattended \
--optionfile "${APP_HOME}/optionFile" && \
  /opt/Fortify/OpenText_SAST_Fortify_25.4.0/bin/fortifyupdate -
import ${RULEPACK} && \   rm OpenText_SAST_Fortify_linux-
x64_25.4.0.run optionFile   ENTRYPOINT
["/opt/Fortify/OpenText_SAST_Fortify_25.4.0/bin/sourceanalyzer"]
```

現在のディレクトリからDockerfileを使用してdockerイメージを作成するには、docker buildコマンドを使用する必要があります。例:

```
docker buildx build -f <docker_file> -t <image_name> "."
```

1.5.2.2. コンテナの実行

このトピックでは、OpenText SASTイメージをコンテナとして実行する方法について説明し、変換およびスキャン用のDocker実行コマンドの例を示します。

Note

コンテナでOpenText SASTを実行する際に、特にランタイムコンテナ保護も利用する場合は、OpenText SASTにビルドコマンド(javacなど)を実行するための適切な許可があることを確認します。

OpenText SASTイメージをコンテナとして実行するには、ホストファイルシステムからコンテナに次の2つのディレクトリをマウントする必要があります。

- 分析するソースファイルが含まれるディレクトリ。
- 変換フェーズとスキャンフェーズの間にOpenText SASTビルドセッションを保存し、出力ファイル(ログとFPRファイル)をホストと共有するための一時ディレクトリ。

  このディレクトリは、OpenText SASTの変換コマンドとスキャンコマンドの両方で `–project-root` コマンドラインオプションを使用して指定します。

次のコマンドの例では、 `/sources` に入力ディレクトリ `/src` をマウントし、 `/scratch_docker` に一時ディレクトリをマウントします。この例のイメージ名は `fortify-sast` です。

変換およびスキャン用のDocker実行コマンドの例

次の例では、一時ディレクトリとソースディレクトリをマウントし、変換フェーズ用にコンテナからOpenText SASTを実行します。

```
docker run -v /scratch_local/:/scratch_docker -v /sources/:/src
-it fortify-sast –b MyProject -project-root /scratch_docker
[<sca_options>] /src
```

次の例では、一時ディレクトリをマウントし、分析フェーズ用にコンテナからOpenText SASTを実行します。

```
docker run -v /scratch_local/:/scratch_docker -it fortify-sast
–b MyProject -project-root /scratch_docker –scan [<sca_options>]
–f /scratch_docker/MyResults.fpr
```

`MyResults.fpr` 出力ファイルがホストの `/scratch_local` ディレクトリに作成されます。

1.5.3. OpenText SASTのアップグレード

OpenText SASTをアップグレードするには、現在のバージョンがインストールされている場所とは異なる場所に新しいバージョンをインストールし、以前のインストールから設定を移行します。この移行では、 *<sast_install_dir>*/Core/config ディレクトリにあるアーティファクトファイルが保持および更新されます。

以前のリリースから設定を移行しない場合は、次のデータが変更されていれば、バックアップを保存することが推奨されています。

- *<sast_install_dir>*/Core/config/rules folder
- *<sast_install_dir>*/Core/config/customrules folder
- *<sast_install_dir>*/Core/config/ExternalMetadata folder
- *<sast_install_dir>*/Core/config/CustomExternalMetadata folder
- *<sast_install_dir>*/Core/config/server.properties file
- *<sast_install_dir>*/Core/config/scales folder

新しいバージョンをインストールしたら、以前のバージョンをアンインストールできます。詳細については、「OpenText SASTのアンインストールについて」を参照してください。

Note

以前のバージョンがインストールされたままにしておくこともできます。同じシステムに複数のバージョンがインストールされている場合は、コマンドラインからコマンドを実行すると、最近インストールされたバージョンが使用されます。

1.5.4. OpenText SASTのアンインストールについて

このセクションでは、OpenText SASTをアンインストールする方法について説明します。標準インストールウィザードを使用することも、OpenText SASTをサイレントでインストールすることもできます。Windows以外のシステムでは、テキストベースのアンインストールを実行することもできます。

このセクションでは、次のトピックについて説明します。

1.5.4.1. OpenText SASTのアンインストール

OpenText SASTをアンインストールするには:

1. インストールディレクトリに移動します。
2. 次の表で説明するように、ご使用のオペレーティングシステム用のアンインストールコマンドを実行します。

OS	アンインストールコマンド
Windows	`Uninstall_OpenTextSASTFortify.exe` または、Windowsインタフェースからアプリケーションをアンインストールすることもできます。手順については、Microsoft Windowsのドキュメントを参照してください。
Linux AIX	`./Uninstall_OpenTextSASTFortify`
macOS®	`Uninstall_OpenTextSASTFortify.app`

3. アプリケーション全体を削除するか、個々のコンポーネントを削除するかを指定するよう求めるメッセージが表示されます。どちらかを選択し、［**次へ(Next)**］をクリックします。

   特定のコンポーネントをアンインストールする場合は、［**アンインストールするコンポーネントの選択(Select Components to Uninstall)**］ページで削除するコンポーネントを選択し、［**次へ(Next)**］をクリックします。

4. すべてのアプリケーション設定を削除するかどうかを指定するよう求めるメッセージが表示されます。次のいずれかを実行します。
   - アンインストールするOpenText SASTのバージョンと一緒にインストールされたコンポーネントのアプリケーション設定を削除するには、［**はい(Yes)**］をクリックします。

OpenText SAST (`sca<version>`)アプリケーション設定フォルダは削除されません。

- ［**No**］をクリックして、アプリケーション設定をシステムに保持します。

1.5.4.2. OpenText SASTのサイレントアンインストール

OpenText SASTをサイレントでアンインストールするには、次の手順に従います。

1. インストールディレクトリに移動します。
2. 次の表で説明するように、ご使用のオペレーティングシステム用のアンインストールコマンドを実行します。

OS	アンインストールコマンド
Windows	`Uninstall_OpenTextSASTFortify.exe --mode unattended`
Linux AIX	`./Uninstall_OpenTextSASTFortify --mode unattended`
macOS®	`Uninstall_OpenTextSASTFortify.app/Contents/MacOS/installbuilder.sh --mode unattended`

Note

Windows、Linux、およびmacOS®の場合、アンインストールするOpenText SASTのバージョンと一緒にインストールされたコンポーネントのアプリケーション設定はアンインストーラによって削除されます。

1.5.4.3. Windows以外のプラットフォームでテキストベースモードでのOpenText SASTのアンインストール

テキストベースモードでOpenText SASTをアンインストールするには、

1. インストールディレクトリに移動します。
2. 次の表で説明するように、ご使用のオペレーティングシステム用のアンインストールコマンドを実行します。

OS	アンインストールコマンド
Linux AIX	`./Uninstall_OpenTextSASTFortify --mode text`
macOS®	`Uninstall_OpenTextSASTFortify.app/Contents/MacOS/installbuilder.sh --mode text`

1.5.5. インストール後のタスク

インストール後のタスクでは、OpenText SASTの使用を開始する準備をします。

このセクションでは、次のトピックについて説明します。

1.5.5.1. インストール後処理ツールの実行

インストール後処理コマンドラインツールを使用すると、OpenText SASTの以前のバージョンのプロパティファイルの移行、OpenText Application Security Contentの更新設定、およびApplication Securityへの接続設定を行うことができます。

インストール後処理ツールを実行するには:

1. *<sast_install_dir>*/bin/ に移動します。
2. コマンドプロンプトで「 scapostinstall 」と入力します。
3. 次のいずれかを入力します。
    - 設定を表示するには、「 s 」を入力します。
    - 前のプロンプトに戻るには、「 r 」を入力します。
    - ツールを終了するには、「 q 」を入力します。

1.5.5.2. プロパティファイルの移行

以前のバージョンのOpenText SASTから、システムにインストールされている現在のバージョンにプロパティファイルを移行するには、次の手順に従います。

1. *<sast_install_dir>*/bin/ に移動します。
2. コマンドプロンプトで「 scapostinstall 」と入力します。
3. 「 1 」と入力して、 Migration を選択します。
4. 「 1 」と入力して、 Static Code Analyzer Migration を選択します。
5. 「 1 」と入力して、 Migrate from an existing Fortify installation を選択します。
6. 「 1 」と入力して、 Set previous Fortify installation directory を選択します。
7. 以前のインストールディレクトリを入力します。
8. 「 s 」と入力して、設定を確認します。
9. 「 2 」と入力して、移行を実行します。
10. 「 y 」と入力して、確認します。

1.5.5.3. ロケールの指定

OpenText SASTインストールのデフォルトロケールは英語です。

OpenText SASTインストールのロケールを変更するには、次の手順を実行します。

1. *<sast_install_dir>*/bin/ に移動します。
2. コマンドプロンプトで「 scapostinstall 」と入力します。
3. 「 2 」と入力して、 Settings を選択します。
4. 「 1 」と入力して、 General を選択します。
5. 「 1 」と入力して、 Locale を選択します。
6. 次のいずれかのロケールコードを入力します。
    - en (英語)
    - es (スペイン語)
    - ja (日本語)
    - ko (韓国語)
    - pt_BR (ポルトガル語(ブラジル))
    - zh_CN (簡体字中国語)
    - zh_TW (繁体字中国語)

1.5.5.4. Fortify Security Contentの更新の設定

OpenText Application Security Contentを取得する方法を指定します。サーバに接続する必要がある場合は、プロキシ情報を指定する必要もあります。

OpenText Application Security Content更新の設定を指定するには、次の手順に従います。

1. *<sast_install_dir>*/bin/ に移動します。
2. コマンドプロンプトで「 scapostinstall 」と入力します。
3. 「 2 」と入力して、 Settings を選択します。
4. 「 2 」と入力して、 Fortify Update を選択します。
5. Fortify Rulepack更新サーバのURLを変更するには、「 1 」と入力してから、URLを入力します。

   Fortify Rulepack更新サーバのデフォルトの URLは https://update.fortify.com です。
6. Application Security Content更新用のプロキシを指定するには、次の手順に従います。
   1. 「 2 」と入力して Proxy Server を選択してから、プロキシサーバの名前を入力します。

      プロトコルとポート番号を除外します(例: some.secureproxy.com)。
   2. 「 3 」と入力して Proxy Server Port を選択してから、プロキシサーバのポート番号を入力します。
   3. (オプション)プロキシサーバのユーザ名(オプション 4)とパスワード(オプション 5)を指定することもできます。

1.5.5.5. Application Securityへの接続の設定

Application Securityに接続する方法を指定します。ネットワークでApplication Securityへのアクセスにプロキシサーバが使用されている場合は、プロキシ情報を指定する必要があります。

Application Securityへの接続の設定を指定するには、次の手順に従います。

1. *<sast_install_dir>*/bin/ に移動します。
2. コマンドプロンプトで「 scapostinstall 」と入力します。
3. 「 2 」と入力して、 Settings を選択します。
4. 「 3 」と入力して、 Software Security Center Settings を選択します。
5. 「 1 」と入力して Server URL を選択してから、Application SecurityサーバのURLを入力します。
6. 接続のプロキシ設定を指定するには、次の手順に従います。
    1. 「 2 」と入力して Proxy Server を選択してから、プロキシサーバの名前を入力します。

       プロトコルとポート番号を除外します(例: some.secureproxy.com)。
    2. 「 3 」と入力して Proxy Server Port を選択してから、プロキシサーバのポート番号を入力します。
    3. プロキシサーバのユーザ名とパスワードを指定するには、ユーザ名のオプション 4 とパスワードのオプション 5 を使用します。
7. (オプション)次を指定することもできます。
    - Application SecurityサーバからOpenText Application Security Contentを更新するかどうか(オプション 6)
    - Application Securityユーザ名(オプション 7)

1.5.5.6. プロキシサーバ設定の削除

以前にFortify Rulepack更新サーバまたはApplication Securityのプロキシサーバ設定を指定し、不要になった場合は、それらの設定を削除できます。

OpenText Application Security Contentの更新を取得したり、Application Securityに接続したりするためのプロキシ設定を削除するには:

1. *<sast_install_dir>*/bin/ に移動します。
2. コマンドプロンプトで「 scapostinstall 」と入力します。
3. 「 2 」と入力して、 Settings を選択します。
4. 「 2 」を入力して Fortify Update を選択するか、「 3 」を入力して Software Security Center Settings を選択します。
5. 削除するプロキシ設定に対応する番号を入力し、マイナス記号(-)を入力して設定を削除します。
6. 削除するプロキシ設定ごとに手順5を繰り返します。

1.5.5.7. 信頼された証明書の追加

OpenText SASTから他のOpenText Application Securityソフトウェア製品および外部システムに接続するために、HTTPS経由の通信が必要になる場合があります。次に例をいくつか示します。

- OpenText SASTのデフォルトでは、ライセンス管理にLIMサーバと通信するために、HTTPS接続が必要です。

  `com.fortify.sca.lim.RequireTrustedSSLCert` プロパティでは、LIMサーバとの接続に信頼されたSSL証明書が必要であるかどうかが決定されます。このプロパティの詳細については、「LIMのプロパティ」を参照してください。

- fortifyupdateコマンドラインツールでは、Windowsシステムのインストール時に自動的に、または手動で(「OpenText Application Security Contentの手動インストール」を参照)、HTTPS接続を使用してOpenText Application Security Contentが更新されます。

- ScanCentral SASTセンサとして設定されたOpenText SASTは、HTTPS接続を使用してコントローラと通信します。

HTTPSを使用している場合、OpenText SASTとそのアプリケーションは提示されたSSLサーバ証明書に標準チェックをデフォルトで適用します。これには、証明書が信頼されているかどうかを確認するチェックが含まれます。組織が独自の認証局(CA)を運営し、そのCAから発行された証明書がサーバで提示される接続がOpenText SASTで信頼される必要がある場合は、そのCAを信頼するようにOpenText SASTを設定する必要があります。さもないと、HTTPS接続の使用に失敗する可能性があります。

CAの信頼された証明書をOpenText SASTキーストアに追加する必要があります。OpenText SASTキーストアは、`<sast_install_dir>/jre/lib/security/cacerts` ファイルにあります。keytoolコマンドを使用すると、信頼された証明書をキーストアに追加できます。

信頼された証明書をOpenText SASTキーストアに追加するには、次の手順に従います。

1. コマンドプロンプトを開き、次のコマンドを実行します。

   ```
   <sast_install_dir>/jre/bin/keytool -importcert -alias
   <alias_name> -cacerts -file <cert_file>
   ```

   ここで:

   - `<alias_name>` は、追加する証明書に固有の名前です。

- `<cert_file>` は、PEM形式またはDER形式の信頼されたルート証明書を含むファイルの名前です。

2. キーストアパスワードを入力します。

> **Note**
>
> デフォルトのパスワードは `changeit` です。

3. この証明書を信頼するように求めるプロンプトが表示されたら、［**yes**］を選択します。

1.6. 分析プロセスの概要

このセクションでは、次のトピックについて説明します。

1.6.1. スキャンの基本

コードを変換および分析するためのコマンドの基本的なシーケンスを以下に示します。

1. 指定したビルドIDの既存のOpenText SAST一時ファイルをすべて削除します。

   ```
   sourceanalyzer -b MyProject -clean
   ```

   以前に使用したビルドIDを持つプロジェクトを分析するには、常に、このステップで分析を開始してください。

2. プロジェクトコードを変換します。可能な場合には、ビルド統合を使用して、ソースファイルの選択と変換設定の正しい設定を自動化することをお勧めします。
   通常、ビルド統合には次の形式が使用されます。

   ```
   sourceanalyzer -b MyProject ... <build_command>
   ```

   または、手動で操作を行います。

   ```
   sourceanalyzer -b MyProject <files_to_analyze>
   <options_specific_to_language>
   ```

   変換の詳細については、分析しようとしているプログラミング言語のセクションを参照してください。

3. プロジェクトコードを分析し、結果をFortify Project Results(FPR)ファイルに保存します。

   ```
   sourceanalyzer -b MyProject -scan -f MyResults.fpr
   ```

   詳細については、「分析フェーズ」を参照してください。

   また、OpenText™ ScanCentral SASTを介して簡素化したり、リモートで実行したりもできます。詳細については、『*OpenText™ ScanCentral SASTインストール、設定、および使用ガイド*』を参照してください。

1.6.2. 変換フェーズ

通常どおりにコンパイルされているプロジェクトを正常に変換するには、プロジェクトをビルドするために必要な依存関係があることを確認します。特定の要件がある言語については、特定のソースコードタイプのセクションを参照してください。

分析プロセスの最初のステップ(ファイルの変換)を実行するための基本的なコマンドライン構文は、次のとおりです。

```
sourceanalyzer -b <build_id> ... <files>
```

または

```
sourceanalyzer -b <build_id> ... <compiler_command>
```

変換フェーズは、`sourceanalyzer`コマンドを使用した1回以上のOpenText SASTの呼び出しで構成されます。OpenText SASTでは、ビルドID(`-b`オプション)を使用して呼び出しが相互に関連付けられます。後続の`sourceanalyzer`の呼び出しでは、ビルドIDに関連付けられたファイルリストに新しく指定されたソースファイルまたは環境設定ファイルが追加されます。

変換後に`-show-build-warnings`ディレクティブを使用して、変換フェーズで発生した警告やエラーのリストを表示できます。

```
sourceanalyzer -b <build_id> -show-build-warnings
```

ビルドIDに関連付けられたファイルを表示するには、`-show-files`ディレクティブを使用します。

```
sourceanalyzer -b <build_id> -show-files
```

変換フェーズに関する特別な考慮事項

プロジェクトで変換フェーズを実行する前に、次の特別な考慮事項を検討してください。

- 動的言語(JavaScript/TypeScript、PHP、Python、Ruby)を変換する場合は、1回の呼び出しですべてのソースファイルを同時に指定する必要があります。OpenText SASTでは、後続の呼び出し時にビルドIDで関連付けられたファイルリストに新しいファイルを追加する機能はサポートされていません。

- 生成されるコードは、スクリプト、または解析ツールなどのツールによって、自動的に生成されます。このコードは最適化され、最小化されることもあれば、大きく複雑になることもあります。したがって、このコードでOpenText SASTから報告される脆弱性をすべて修正するのは困難なことがあるため、変換から除外することをお勧めします。 `-exclude` コマンドラインオプションを使用して、このタイプのコードを変換から除外してください。
- ビルドマシン上でプロジェクトを変換し、パフォーマンスの高いシステムでスキャンを実行するには、「モバイルビルドセッションの使用」を参照してください。

1.6.3. 分析フェーズ

分析フェーズでは、変換時に作成された中間ファイルがスキャンされ、脆弱性結果ファイル(FPR)が作成されます。

このフェーズは、`sourceanalyzer`の1回の呼び出しで構成されます。ビルドIDを指定し、`-scan`ディレクティブを他の必要な分析オプションまたは出力オプションとともに含めます(「分析オプション」および「出力オプション」を参照)。

次の例は、分析フェーズを実行し、結果をFPRファイルに保存するコマンドライン構文を示しています。

```
sourceanalyzer -b MyProject -scan -f MyResults.fpr
```

Note

デフォルトでは、OpenText SASTのFPRファイルのソースコードが含まれています。

複数のビルドを単一のスキャンコマンドと組み合わせるには、コマンドラインに追加のビルドを追加します。

```
sourceanalyzer -b MyProject1 -b MyProject2 -b MyProject3 -scan -
f MyResults.fpr
```

1.6.4. 変換フェーズと分析フェーズの検証

Fortify Audit Workbenchの証明書には、スキャンによるコード分析が完了し、有効であるどうかが示されます。Fortify Audit Workbenchのプロジェクト概要には、OpenText SASTでスキャンされたコードに関する次の特定の情報が表示されます。

- スキャンされたファイルのリスト(ファイルサイズとタイムスタンプ付き)
- 変換に使用されるJavaクラスパス(該当する場合)
- 分析に使用されるRulepack
- OpenText SASTのランタイム設定とコマンドラインオプション
- 変換時または分析時に発生したエラーまたは警告
- コンピュータおよびプラットフォームの情報

Note

結果証明書を取得するには、分析フェーズの出力形式にFPRを指定する必要があります。

結果証明書情報を表示するには、Fortify Audit WorkbenchでFPRファイルを開き、**［Tools］>［Project Summary］>［Certification］**の順に選択します。詳細については、*OpenText™ Fortify Audit Workbenchユーザガイド*を参照してください。

1.7. Java、Kotlin、およびJSPプロジェクトの分析

このセクションでは、Java、Kotlin、JSPプロジェクト、およびこれらの言語を組み合わせて使用するプロジェクトを変換する方法について説明します。

OpenText SASTでは、Jakarta EE (Java EE)アプリケーション(JSPファイル、環境設定ファイル、展開デスクリプタを含む)、Javaバイトコード、およびLombokアノテーション付きJavaコードの分析をサポートしています。

このセクションでは、次のトピックについて説明します。

1.7.1. Gradleとの統合

OpenText SASTは、Gradleでビルドされたプロジェクトとの変換統合を提供します。ビルドスクリプトを変更せずに統合するか、タスクを使用してOpenText SASTを起動するOpenText SAST Gradleプラグインを使用できます。

このセクションでは、次のトピックについて説明します。

1.7.1.1. Gradle統合の使用

`build.gradle`ファイルを変更せずに、Gradleを使用してビルドされたプロジェクトを変換できます。ビルドが実行されると、OpenText SASTでソースファイルがコンパイル時に変換されます。代わりに、OpenText SAST Gradle Pluginを使用してGradleビルドスクリプト内から分析を実行できます(「OpenText SAST Gradleプラグインの使用」を参照)。

Gradle統合で特定的にサポートされているプラットフォームおよび言語については、「ビルドツール」を参照してください。プロジェクト内のファイルは、Gradleの統合でサポートされていない言語では変換されません(エラーもレポートされません)。したがって、これらのファイルは分析されず、既存の潜在的な脆弱性は検出されない可能性があります。

OpenText SASTをGradleビルドに統合するには、`sourceanalyzer`実行可能ファイルがPATH環境変数に含まれていることを確認します。プロジェクトをビルドするすべてのGradleコマンドに対して、システムパスの`sourceanalyzer`実行可能ファイルを常に使用します。

Note

OpenText SASTが複数インストールされている場合、Gradleプロジェクトに使用するバージョンは、PATH環境変数内の他のすべてのバージョンのOpenText SASTより前に定義されている必要があります。

次のように、Gradleコマンドラインの前に`sourceanalyzer`コマンドを追加します。

```
sourceanalyzer -b <build_id><sca_options> gradle
[<gradle_options>] <gradle_tasks>
```

Gradle統合の例

```
sourceanalyzer -b MyProject gradle clean build sourceanalyzer -b
MyProject gradle --info assemble
```

ビルドファイル名が`build.gradle`と異なる場合は、次の例に示すように、ビルドファイル名に`--build-file`オプションを付けます。

```
sourceanalyzer -b MyProject gradle --build-file sample.gradle
clean assemble
```

次の例に示すように、Gradle Wrapper (`gradlew`)を使用することもできます。

```
sourceanalyzer -b MyProject gradlew [<gradle_options>]
```

プロジェクトを変換し、変換からファイルを除外する場合:

```
sourceanalyzer -b MyProject -exclude "src\test\**\*" gradlew
build
```

アプリケーションでXMLまたはプロパティ環境設定ファイルが使用されている場合は、これらのファイルを個別の `sourceanalyzer` コマンドを使って変換します。プロジェクトファイルで使用したときと同じビルドIDを使用します。次に例を示します。

```
sourceanalyzer -b MyProject <path_to_xml_files>sourceanalyzer -b
MyProject <path_to_properties_files>
```

OpenText SASTがgradleまたはgradlewを使ってプロジェクトを変換した後、次の例に示すように分析フェーズを実行して、結果をFPRファイルに保存できます。

```
sourceanalyzer -b MyProject -scan -f MyResults.fpr
```

参照情報

OpenText SAST Gradleプラグインの使用

1.7.1.2. Gradle統合のトラブルシューティング

OpenText SASTGradle統合を使用したGradleビルドで設定キャッシング(`--configuration-cache` オプション)を使用した場合、ビルドは次のメッセージをレポートします。

```
Configuration cache problems found in this build.
```

次のようなメッセージが表示される場合もあります。

```
FAILURE: Build failed with an exception...
```

このメッセージは、OpenText SAST変換に関しては無視して構いません。プロジェクトが変換されているからです。プロジェクトが変換されていることは、 `-show-files` オプションを使用して確認できます。例:

```
sourceanalyzer -b mybuild -show-files
```

1.7.1.3. Gradleプラグインの使用

OpenText SASTのインストールには、`<sast_install_dir>/plugins/gradle`にGradleプラグインが含まれています。OpenText SAST Gradleプラグインを使用するには、まずJavaまたはKotlinプロジェクト用にプラグインを設定してから、そのプラグインを使用してプロジェクトを分析する必要があります。Gradleプラグインは、分析用に3つのOpenText SASTタスク(sca.clean、sca.translate、およびsca.scan)を提供しています。特にOpenText SAST Gradleプラグイン用にサポートされているプラットフォームと言語については、「ビルドツール」を参照してください。

Note

OpenText SASTが複数インストールされている場合、Gradleプロジェクトに使用するバージョンは、PATH環境変数内の他のすべてのバージョンのOpenText SASTより前に定義されている必要があります。

OpenText SAST Gradleプラグインを設定するには:

1. Gradle設定ファイルを編集して、プラグインへのパスを指定します。

   Groovy DSL (`settings.gradle`):

   ```
   pluginManagement { repositories { gradlePluginPortal()
   maven { url = uri("file://<sast_plugin_path>") } } }
   ```

   Kotlin DSL (`settings.gradle.kts`):

   ```
   pluginManagement { repositories { maven(url =
   uri("file://<sast_plugin_path>")) gradlePluginPortal() } }
   ```

2. 以下の例に示すように、ビルドスクリプトにエントリを追加します。

   Groovy DSL (`build.gradle`):

   ```
   id 'com.fortify.sca.plugins.gradlebuild' version '25.4'
   ```

   および

```
SCAPluginExtension { buildId = "MyProject" options = ["-
encoding", "utf-8", "-logfile", "MyProject.log", "-debug-
verbose"] }
```

または次のエントリの例では、変換からファイルが除外されます。

```
SCAPluginExtension { buildId = "MyProject" options = ["-
encoding", "utf-8", "-logfile", "MyProject.log", "-debug-
verbose", "-exclude", "src/test/**/*"] }
```

Kotlin DSL (`build.gradle.kts`):

```
plugins { id ("com.fortify.sca.plugins.gradlebuild") version
"25.4" ... }
```

および

```
SCAPluginExtension { buildId = "MyProject" options =
listOf("-encoding", "utf-8", "-logfile", "MyProject.log", "-
debug-verbose") }
```

または次のエントリの例では、変換からファイルが除外されます。

```
SCAPluginExtension { buildId = "MyProject" options =
listOf("-encoding", "utf-8", "-logfile", "MyProject.log", "-
debug-verbose", "-exclude", "src/test/**/*") }
```

3. Gradle設定ファイルとGradleビルドファイルを保存して閉じます。

次のコマンドシーケンスを使用して、JavaまたはKotlinプロジェクトを分析します。

- 既存のJavaまたはKotlinプロジェクトビルドの既存のOpenText SAST一時ファイルをすべて削除するには、次のコマンドを実行します。

  ```
  gradlew sca.clean
  ```

- 設定済みのJavaまたはKotlinプロジェクトの変換フェーズを実行するには、次のコマンドを実行します。

```
gradlew sca.translate
```

- 設定済みのJavaまたはKotlinプロジェクトを分析するには、次のコマンドを実行します。

  ```
  gradlew sca.scan
  ```

  このタスクは、OpenText SASTによってプロジェクトがOpenText SAST Gradleプラグインを使用してすでに変換されている場合に、正常に実行されます。

サブプロジェクトを持つJavaまたはKotlinプロジェクトの操作

JavaまたはKotlinマルチプロジェクトビルド(サブプロジェクトを含む)がある場合は、OpenText SAST Gradleプラグインを `allprojects` ブロックを使用して設定する必要があります。以下に例を示します。

Groovy DSL (`build.gradle`)

```
allprojects { apply plugin:
"com.fortify.sca.plugins.gradlebuild" SCAPluginExtension {
buildId = "MyProject" options = ["-encoding", "utf-8", "-
logfile", "MyProject.log", "-debug-verbose"] ... } }
```

Kotlin DSL (`build.gradle.kts`):

```
allprojects { apply(plugin =
"com.fortify.sca.plugins.gradlebuild") SCAPluginExtension {
buildId = "MyProject" options = listOf("-encoding", "utf-8", "-
logfile", "MyProject.log", "-debug-verbose") ... } }
```

参照情報

Gradle統合の使用

1.7.2. Mavenとの統合

OpenText SASTには、Mavenプロジェクトのビルドに次の機能を追加する方法を提供するMavenプラグインが含まれています。

- OpenText SASTで消去、変換、スキャンする
- 変換されたプロジェクトのモバイルビルドセッション(MBS)をOpenText SASTからエクスポートする
- 変換されたコードをScanCentral SASTに送信する
- 結果をApplication Securityにアップロードする

プラグインを直接使用することも、プラグインの機能をビルドプロセスに統合することもできます。

このセクションでは、次のトピックについて説明します。

1.7.2.1. Fortify Mavenプラグインのインストールと更新

Fortify Mavenプラグインは、`<sast_install_dir>/plugins/maven`に配置されています。このディレクトリには、バイナリバージョンとソースバージョンのプラグインがzipアーカイブとtarballアーカイブの両方で含まれています。プラグインをインストールするには、使用するバージョン(バイナリまたはソース)を抽出し、付属の`README.TXT`ファイルの指示に従います。アーカイブを展開したディレクトリでインストールを実行します。

Mavenのサポートされているバージョンについては、「ビルドツール」を参照してください。

以前のバージョンのFortify Mavenプラグインがインストールされている場合は、最新バージョンをインストールします。

Fortify Mavenプラグインのアンインストール

Fortify Mavenプラグインをアンインストールするには、`<maven_local_repo>/repository/com/fortify/ps/maven/plugin`ディレクトリからすべてのファイルを手動で削除します。

1.7.2.2. Fortify Mavenプラグインのインストールのテスト

Fortify Mavenプラグインをインストールしたら、付属のサンプルファイルのいずれかを使用して正しくインストールされていることを確認します。

Eightballサンプルファイルを使用してFortify Mavenプラグインをテストするには、次の手順に従います。

1. `sourceanalyzer` 実行可能ファイルを含むディレクトリをパス環境変数に追加します。

   例:

   ```
   export set PATH=$PATH:/<sast_install_dir>/bin
   ```

   または

   ```
   set PATH=%PATH%;<sast_install_dir>/bin
   ```

2. 「`sourceanalyzer -version`」と入力してパスの設定をテストします。

   パスの設定が正しい場合は、OpenText SASTにバージョン情報が表示されます。

3. サンプルのEightballディレクトリ(`<root_dir>/samples/EightBall`)に移動します。

4. 次のコマンドを入力します。

   ```
   mvn com.fortify.sca.plugins.maven:sca-maven-plugin:
   <ver>:clean
   ```

   ここで、`<ver>`は使用しているFortify Mavenプラグインのバージョンです。バージョンが指定されていない場合は、ローカルリポジトリにインストールされている最新バージョンのFortify Mavenプラグインが Mavenで使用されます。

   Note

   Fortify Mavenプラグインのバージョンを表示するには、テキストエディタで`pom.xml`に抽出した`pom.xml`ファイルを開きます。Fortify Mavenプラグインのバージョンは、`<version>`要素で指定されます。

5. ステップ4のコマンドが正常に完了した場合は、Fortify Mavenプラグインが正しくインストールされています。次のメッセージが表示された場合は、Fortify Mavenプラグインが正しくインストールされていません。

   ```
   [ERROR] Error resolving version for plugin
   'com.fortify.sca.plugins.maven:sca-maven-plugin' from the
   repositories
   ```

   Mavenのローカルリポジトリをチェックし、Fortify Mavenプラグインの再インストールを試みます。

1.7.2.3. Fortify Mavenプラグインの使用

Mavenプロジェクトでは、次の2つの方法で分析を実行します。

- OpenText SASTビルド統合時

  この方法では、プロジェクトをビルドするために使用されるMavenコマンドの前に `sourceanalyzer` コマンドとOpenText SASTオプションを追加します。OpenText SASTのビルド統合の一環としてファイルを分析するには、次の手順に従います。

  1. 以前のビルドを消去します。

     ```
     sourceanalyzer -b MyProject -clean
     ```

  2. コードを変換します。

     ```
     sourceanalyzer -b MyProject [<sca_options>]
     [<mvn_command_with_options>]
     ```

     例:

     ```
     sourceanalyzer -b MyProject mvn package
     ```

     ```
     sourceanalyzer -b MyProject -exclude "**/Test/*.java"
     mvn clean install
     ```

     使用可能なOpenText SASTオプションについては、「コマンドラインインタフェース」を参照してください。

  3. 次の例に示すように、スキャンを実行して結果をFPRファイルに保存します。

     ```
     sourceanalyzer -b MyProject [<sca_scan_options>] -scan -
     f MyResults.fpr
     ```

- Mavenプラグインとして

  この方法では、 `mvn` コマンドを使用して分析タスクを目標として実行します。たとえば、次のコマンドを使用してソースコードを変換します。

```
mvn com.fortify.sca.plugins.maven:sca-maven-
plugin:25.4.0:translate
```

たとえば、次のコマンドを使用してソースコードを変換し、テストファイルを除外します。

```
mvn -Dfortify.sca.exclude="**/Test/*.java"
com.fortify.sca.plugins.maven:sca-maven-
plugin:25.4.0:translate
```

この方法でコードを分析するには、Fortify Mavenプラグインに付属のドキュメントを参照してください 。次の表では、Fortify Mavenプラグインをインストールした後にドキュメントが置かれる場所について説明します。

パッケージタイプ	ドキュメントの場所
バイナリ	*<root_dir>*/docs/index.html
ソース	*<root_dir>*/sca-maven-plugin/target/site/index.html

1.7.3. Bazelとの統合

Bazelのビルドと統合するために、OpenText SASTはソースファイルをコンパイル時に変換します。したがって、Bazelのビルドの前提条件は、Bazelのビルドが正常に実行されることです。サポートされているBazelのバージョンについては、「ビルドツール」を参照してください。

Bazelと統合するには、Bazelのワークスペースディレクトリに移動し、構築するBazelのターゲットでs sourceanalyzerを実行します。次のように、変換のための他のsourceanalyzerオプションを指定できます。

```
sourceanalyzer -b <build_id> <sca_options> bazel build <target>
```

プロジェクトを変換し、変換からファイルを除外する場合:

```
sourceanalyzer -b MyProjectC -exclude C:\test\MY-JAVA-
APP\src\proj\content.py bazel build //projc:my-python-prj
```

1.7.3.1. Java Bazelの統合例

特定のターゲットのプロジェクトを変換する場合:

```
sourceanalyzer -b MyProjectA bazel build //proja:my-prj
```

パッケージ `proja/abc` 内のターゲット `abc` を変換する場合:

```
sourceanalyzer -b MyProjectA bazel build //proja/abc
```

または

```
sourceanalyzer -b MyProjectA bazel build //proja/abc:abc
```

パッケージ `proja/abc` 内のすべてのターゲットを変換する場合:

```
sourceanalyzer -b MyProjectA bazel build //proja/abc:all
```

`projb/` ディレクトリ内のすべてのターゲットを変換する場合:

```
sourceanalyzer -b MyProjectB bazel build //projb/...
```

変換に特定のJDKバージョンを指定する場合:

```
sourceanalyzer -b MyProjectC -jdk 17 bazel build //projc:my-
java-prj
```

プロジェクトを変換し、変換からファイルを除外する場合:

```
sourceanalyzer -b MyProjectC -exclude C:\test\MY-JAVA-
APP\src\main\java\com\example\HelpContent.java bazel build
//projc:my-java-prj
```

OpenText SASTとBazelの統合では、複数のターゲットおよび関連アクション(ターゲットの除外など)はサポートされていません。

1.7.4. Antとの統合

Antビルドファイルを使用するプロジェクト用に、Javaソースファイルを変換できます。この統合は、Antの `build.xml` ファイルを変更せずにコマンドラインに適用できます。ビルドが実行されると、OpenText SASTで `javac` タスクの呼び出しがすべて傍受され、Javaソースファイルがコンパイル時に変換されます。プロパティはすべて、ANT_OPTS環境変数に追加して、Antに渡すようにします。sourceanalyzerコマンドには含めないでください。

Note

アプリケーションの一部であるJSPファイル、環境設定ファイル、またはJava以外のソースファイルを個別のステップで変換する必要があります。

Antの統合を使用するには、 `sourceanalyzer` 実行可能ファイルがPATH環境変数に入っていることを確認します。

次のように、Antコマンドラインの前に `sourceanalyzer` コマンドを追加します。

```
sourceanalyzer -b <build_id> [<sca_options>] ant [<ant_options>]
```

たとえば、Javaプロジェクトを変換し、変換からファイルを除外するには、次のようにします。

```
sourceanalyzer -b MyProjectA -logfile MyProjectA.log -exclude
src/module-info.java ant
```

1.7.5. JavaおよびKotlinの手動変換構文

JavaまたはKotlinコードを手動で変換するには、すべてのソースファイルをコマンドラインに含め、.jarファイル、.classファイル、またはソースファイルを介してすべての依存関係を提供します。依存関係を提供しない場合、スキャン結果が最適でない可能性があります。

KotlinからJavaへの相互運用性では、`–sourcepath`オプションで提供されるKotlinファイルをサポートしません。`–sourcepath`オプションの詳細については、「Javaコマンドラインオプション」を参照してください。

JavaまたはKotlinコードを変換するための基本的なコマンドライン構文を次の例に示します。

```
sourceanalyzer -b <build_id> -cp <classpath>
[<translation_options>] <files> | <file_specifiers>
```

ここで:

- `<translation_options>` コンパイラに渡されるオプションです。
- `-cp <classpath>` JavaおよびKotlinシンボルの解決に使用するクラスパスを指定します。

  プロジェクトをビルドするために通常使用されるJAR依存関係を含めます。複数のパスはセミコロン(Windows)またはコロン(Windows以外)で区切ります。

  javacと同様に、OpenText SASTではクラスをクラスパスに出現する順序でロードします。リスト内に同じ名前のクラスが複数ある場合、OpenText SASTでは最初にロードされたクラスを使用します。次の例では、`A.jar`と`B.jar`の両方に`MyData.class`という名前のクラスが含まれる場合、OpenText SASTでは`MyData.class`からの`A.jar`を使用します。

  ```
  sourceanalyzer -cp A.jar:B.jar myfile.java
  ```

  `-cp`オプションで重複するクラスを使用しないように強く推奨します。

  OpenText SASTではJARファイルを次の順序でロードします。

  1. `-cp`オプションから
  2. `jre/lib`から
  3. から `<sast_install_dir>`/Core/default_jars

これにより、`-cp` オプションで指定したJARに同様の名前のクラスを含めることで、ライブラリクラスを上書きできます。

使用可能なすべてのJava固有のコマンドラインオプションの詳細については、「Java/J2EEコマンドラインオプション」を参照してください。

Javaコードを使用すると、OpenText SASTはさらにコンパイラをエミュレートして、カスタムビルドスクリプトに容易に統合できます。

OpenText SASTでコンパイラをエミュレートするには、次のコマンドを入力します。

```
sourceanalyzer -b <build_id> javac [<translation_options>]
```

1.7.5.1. Java、Kotlin、およびJSPのコマンドラインオプション

次の表では、Javaコマンドラインオプション(Java SEおよびJakarta EE用)について説明します。

Java、Kotlin、またはJakarta EEのオプション	説明(Description)
`-appserver weblogic | websphere`	JSPファイルを処理するアプリケーションサーバを指定します。 **同等のプロパティ名:** `com.fortify.sca.AppServer`
`-appserver-home <dir>`	アプリケーションサーバのホームを指定します。 • Oracle® WebLogic®の場合、これは`server/lib`ディレクトリを含むディレクトリへのパスです。 • IBM® WebSphere®の場合、これは`JspBatchCompiler`スクリプトを含むディレクトリへのパスです。 **同等のプロパティ名:** `com.fortify.sca.AppServerHome`
`-appserver-version <version>`	アプリケーションサーバのバージョンを指定します。 **同等のプロパティ名:** `com.fortify.sca.AppServerVersion`

Java、Kotlin、またはJakarta EEのオプション	説明(Description)
`-cp <paths>` | `-classpath <paths>`	JavaまたはKotlin依存関係の解決に使用するクラスパスを指定します。形式はjavacと同じです。ディレクトリのセミコロン区切りまたはコロン区切りリストです。次の例に示すように、OpenText SASTファイル指定子を使用することもできます。 `-cp "build/classes:lib/*.jar"` ファイル指定子の詳細については、ファイルとディレクトリの指定を参照してください。 **同等のプロパティ名:** `com.fortify.sca.JavaClasspath`
`-extdirs <dirs>`	javac `extdirs` オプションと同様に、ディレクトリのセミコロン区切りまたはコロン区切りリストを使用できます。これらのディレクトリにあるJARファイルは、クラスパスに暗黙的に含まれます。 **同等のプロパティ名:** `com.fortify.sca.JavaExtdirs`
`-java-build-dir <dirs>`	コンパイル済みJavaソースを含む1つ以上のディレクトリを指定します。

Java、Kotlin、またはJakarta EEのオプション	説明(Description)
`-source <version>` | `-jdk <version>`	JavaまたはKotlinコードを記述するJava™ Development Kit (JDK)バージョンを示します。サポートされているバージョンについては、「サポートされる言語」を参照してください。デフォルトはバージョン11です。 **同等のプロパティ名:** `com.fortify.sca.JdkVersion`
`-custom-jdk-dir`	JDKを含むディレクトリを指定します。OpenText SASTインストール(`<sast_install_dir>/Core/bootcp/`)に含まれていないバージョンを指定するには、このオプションを使用します。サポートされているバージョンについては、「サポートされる言語」を参照してください。 **同等のプロパティ名:** `com.fortify.sca.CustomJdkDir`

Java、Kotlin、またはJakarta EEのオプション	説明(Description)
`-show-unresolved-symbols`	変換されたJavaソースファイルで参照されている未解決のタイプ、フィールド、および関数が、変換の最後に表示されます。リストされるのは、レシーバータイプが解決済みJavaタイプであるフィールドおよび関数参照のみです。各クラス、フィールド、および関数が、コード内で最初に変換された出現箇所のソース情報とともに表示されます。この情報はログファイルにも書き込まれます。 **同等のプロパティ名:** `com.fortify.sca.ShowUnresolvedSymbols`
`-sourcepath <dirs>`	スキャンに含まれていないが名前解決に使用されるソースコードを含むディレクトリのリストをセミコロン区切りまたはコロン区切りで指定します。ソースパスはクラスパスに似ていますが、解決にはクラスファイルではなくソースファイルが使用されます。ターゲットファイルリストによって参照されるソースファイルのみが変換されます。 **同等のプロパティ名:** `com.fortify.sca.JavaSourcePath`

Java、Kotlin、またはJakarta EEのオプション	説明(Description)
`-jvm-default <mode>`	本体がKotlinインターフェイスに入っているメソッドの `DefaultImpls` クラスの生成を指定します。*<mode>*の有効な値は: • `disable` —本体を持つメソッドが含まれている各インターフェイスに対して、 `DefaultImpls` クラスを生成するように指定します。 • `all` —インターフェイスに `DefaultImpls` の注釈が付く場合に、 `@JvmDefaultWithCompatibility` クラスを生成するように指定します。 • `all-compatibility` —インターフェイスに `DefaultImpls` の注釈が付かない限り、 `@JvmDefaultWithoutCompatibility` クラスを生成するように指定します。 **同等のプロパティ名:** `com.fortify.sca.KotlinJvmDefault`

JavaおよびKotlinのプロパティ

1.7.5.2. Javaのコマンドライン例

クラスパスとして `MyServlet.java` が指定された1つのファイル `javaee.jar` を変換するには、次のコマンドを入力します。

```
sourceanalyzer -b MyServlet -cp lib/javaee.jar MyServlet.java
```

`src` ディレクトリ内のすべての `.java` ファイルをクラスパスとして使用して `lib` ディレクトリ内のすべてのJARファイルを変換するには、次のコマンドを入力します。

```
sourceanalyzer -b MyProject -cp "lib/*.jar" "src/**/*.java"
```

javacコンパイラを使用して `MyCode.java` ファイルを変換およびコンパイルするには、次のコマンドを入力します。

```
sourceanalyzer -b MyProject javac -classpath libs.jar
MyCode.java
```

1.7.5.3. Kotlinコマンドラインの例

クラスパスとして `MyKotlin.kt` が指定された1つのファイル `A.jar` を変換するには、次のコマンドを入力します。

```
sourceanalyzer -b MyProject -cp lib/A.jar MyKotlin.kt
```

`src` ディレクトリ内のすべての `.kt` ファイルをクラスパスとして使用して `lib` ディレクトリ内のすべてのJARファイルを変換するには、次のコマンドを入力します。

```
sourceanalyzer -b MyProject -cp "lib/**/*.jar" "src/**/*.kt"
```

gradlewを使用してgradleプロジェクトを変換するには、次のコマンドを入力します。

```
sourceanalyzer -b MyProject gradlew clean assemble
```

`src` ディレクトリおよびサブディレクトリ内の `src/java` およびすべてのJARファイルのJava依存関係をクラスパスとして使用して `lib` ディレクトリ内のすべてのファイルを変換するには、次のコマンドを入力します。

```
sourceanalyzer –b MyProject –cp "lib/**/*.jar" -sourcepath
"src/java" "src"
```

1.7.6. Kotlinスクリプトの分析

OpenText SASTでは、試験的なスクリプトのカスタマイズを除くKotlinスクリプトの変換をサポートしています。スクリプトのカスタマイズには、外部プロパティの追加、静的または動的な依存関係の提供などがあります。スクリプト定義(テンプレート)はカスタムスクリプトの作成に使用され、テンプレートは `*.kts` 拡張子に基づいてスクリプトに適用されます。OpenText SASTでは `*.kts` ファイルを変換しますが、これらのテンプレートは適用されません。

1.7.7. KotlinとJavaの変換相互運用性

プロジェクトにJavaコードを参照するKotlinコードが含まれている場合、別のKotlinファイルを参照するKotlinファイルの場合と同じ方法で、Javaファイルを変換ツールに渡すことができます。変換されたプロジェクトソースの一部として、または `–sourcepath` パラメータとして渡すことができます。

プロジェクトにKotlinコードを参照するJavaコードが含まれている場合、JavaコードとKotlinコードが同じOpenText SASTインスタンスに変換され、Kotlin要素へのJava参照が正しく解決されるようにしてください。KotlinからJavaへの相互運用性では、`–sourcepath` オプションで提供されるKotlinファイルをサポートしません。`–sourcepath` オプションの詳細については、「Java、Kotlin、およびJSPコマンドラインオプション」を参照してください。

1.7.8. Javaの警告の処理

変換中に生成されたすべての警告を表示するには、スキャンフェーズを開始する前に次のコマンドを入力します。

```
sourceanalyzer -b <build_id> -show-build-warnings
```

Java変換の警告

Javaコード変換に次の警告が表示される場合があります。

警告	解決策
`Unable to resolve type... Unable to resolve function... Unable to resolve field... Unable to locate import... Unable to resolve symbol...`	これらの警告は通常、リソースが不足している場合に発生します。たとえば、アプリケーションのビルドに必要な一部の `.jar` および `.class` ファイルが指定されていない可能性があります。 これらの警告を解決するには、アプリケーションが使用する必要なファイルをすべて含める必要があります。
`Multiple definitions found for class...`	この警告は通常、Javaファイル内でクラスが重複している場合に発生します。 これらの警告を解決するには、警告に表示されているソースファイルは同一のファイルが重複して変換対象のソースファイルに複数回含められているものではないことを(たとえば、同じプロジェクトの2つのバージョンが含まれていることを)確認します。重複が存在する場合は、変換するファイルからその1つを削除します。そうすると、OpenText SASTで使用するクラスのバージョンを決定できるようになります。 この警告は、クラスが欠落している可能性も示しています。これを解決するには、必要なすべてのJARファイルをクラスパスに追加してください。

1.7.9. Jakarta EE (Java EE)アプリケーションの分析

Jakarta EEアプリケーションを変換するには、OpenText SASTでJavaソースファイルおよびJakarta EEコンポーネント(JSPファイル、展開デスクリプタ、および環境設定ファイルなど)を処理します。Jakarta EEアプリケーション内のすべての関連ファイルを1ステップで処理することができますが、プロジェクトでは、ビルドプロセスに統合したり、組織内のさまざまな利害関係者のニーズを満たしたりするために、手順をコンポーネントに分割する必要が生じることがあります。

このセクションでは、次のトピックについて説明します。

1.7.9.1. Javaファイルの変換

Jakarta EEアプリケーションを変換するには、Javaファイルの変換に使用したものと同じ手順を使用します。たとえば、「Javaのコマンドライン例」を参照してください。

1.7.9.2. JSPプロジェクト、環境設定ファイル、および展開ディスクリプタの変換

Jakarta EE (Java EE)アプリケーションのJavaファイルを変換するほかに、JSPファイル、環境設定ファイル、および展開デスクリプタの変換が必要になることがあります。JSPファイルは、Webアプリケーションアーカイブ(WAR)の一部である必要があります。ソースディレクトリがすでにWARファイル形式で構成されている場合は、ソースディレクトリからJSPファイルを直接変換できます。そうでない場合は、アプリケーションを展開し、展開ディレクトリからJSPファイルを変換する必要があります。

例:

```
sourceanalyzer -b MyJavaApp "/**/*.jsp" "/**/*.xml"
```

ここで、`/**/*.jsp`はJSPプロジェクトファイルの場所を参照し、`/**/*.xml`は環境設定ファイルおよび展開ディスクリプタファイルの場所を参照します。

1.7.9.3. Jakarta EE (Java EE)変換の警告

Jakarta EEアプリケーションの変換で次の警告が表示される場合があります。

```
Could not locate the root (WEB-INF) of the web application.
Please build your web application and try again. Failed to parse
the following jsp files: <list_of_jsp_files>
```

この警告は、Webアプリケーションが標準のWARディレクトリ形式で展開されていないか、必要なライブラリの完全なセットが含まれていないことを示します。この警告を解決するには、Webアプリケーションが開かれたWARディレクトリ形式であり、アプリケーションに必要なすべての `WEB-INF/lib` ファイルおよび `WEB-INF/classes` ファイルが正しい `.jar` および `.class` ディレクトリに格納されていることを確認してください。また、すべてのタグのすべての `TLD` ファイル、およびそれらのタグの実装に対応するJARファイルが存在することも確認してください。

1.7.10. Javaバイトコードの分析

JavaバイトコードとJSP/Javaコードを同じ `sourceanalyzer` 呼び出しで変換しないことをお勧めします。同じビルドIDを持つ複数の `sourceanalyzer` 呼び出しを使用して、バイトコードとJSP/Javaコードの両方を含むプロジェクトを変換します。

バイトコードを変換するには:

1. `fortify-sca.properties` ファイルに次のプロパティを追加します(または、 `-D` オプションを使用してコマンドラインにこれらのプロパティを含めます)。

   `com.fortify.sca.fileextensions.class=BYTECODE`
   `com.fortify.sca.fileextensions.jar=ARCHIVE`

   これは、OpenText SASTで `.class` および `.jar` ファイルを処理する方法を指定します。

2. 次のいずれかを実行します。

   - OpenText SASTでバイトコードクラスを通常のJavaファイルに逆コンパイルして変換に含めることを要求します。

     `fortify-sca.properties` ファイルに次のプロパティを追加します。

     ```
     com.fortify.sca.DecompileBytecode=true
     ```

     または、 `-D` オプションを使用して、変換フェーズのコマンドラインにこのプロパティを含めます。

     ```
     sourceanalyzer -b MyProject -
     Dcom.fortify.sca.DecompileBytecode=true -cp "lib/*.jar"
     "src/**/*.class"
     ```

   - OpenText SASTで逆コンパイルせずにバイトコードを変換することを要求します。

     最善の結果を得るため、完全なデバッグ情報(`javac -g`)を使用してバイトコードをコンパイルすることを推奨しています。

     変換するJavaバイトコードファイルを指定することで、変換フェーズにバイトコードを含めます。最善のパフォーマンスを得るため、スキャンが必要な `.jar` または `.class` ファイルだけを指定します。次の例では、 `.class` ファイルが変換されています。

```
sourceanalyzer -b MyProject -cp "lib/*.jar"
"src/**/*.class"
```

1.7.11. JSP変換および分析に関する問題のトラブルシューティング

次のセクションでは、JSP分析に関するトラブルシューティング情報について説明します。

一部のJSPを変換できない

OpenText SASTでは、組み込みコンパイラまたは特定のアプリケーションサーバJSPコンパイラを使用して、分析のためにJSPファイルをJavaファイルに変換します。OpenText SASTでJSPファイルをJavaファイルに変換する際にJSPパーサで問題が発生すると、次のようなメッセージが表示されます。

```
Failed to translate the following jsps into analysis model.
Please see the log file for any errors from the jsp parser and
the user manual for hints on fixing those
<list_of_jsp_files>
```

これは通常、次の1つ以上の理由で発生します。

- Webアプリケーションが適切な展開可能WARディレクトリ形式でレイアウトされていない
- アプリケーションに必要なJARファイルまたはクラスの一部が見つからない
- アプリケーションのタグライブラリまたはそれらの定義(TLD)の一部が見つからない

問題の詳細を取得するには、次の手順を実行します。

1. エディタでOpenText SASTログファイルを開きます。
2. 次の文字列を検索します。
    - `Jsp parser stdout:`
    - `Jsp parser stderr:`

JSPパーサではこれらのエラーを生成します。エラーを解決してOpenText SASTを再実行します。

Jakarta EEアプリケーションの分析方法の詳細については、「Jakarta EE (Java EE)アプリケーションの変換」を参照してください。

JSP関連カテゴリで問題の数が増加した

分析結果に含まれるクロスサイトスクリプティングなどのJSP関連カテゴリの脆弱性数が以前のOpenText SASTバージョンと比較して大幅に増加している場合は、分析フェーズで `-legacy-jsp-dataflow` オプションを指定できます(`-scan` オプションも指定します)。このオプションを使用すると、JSP関連のデータフローに対する追加のフィルタリングが有効になり、検出される偽陽性が減少します。

`fortify-sca.properties` ファイルで指定できるこのオプションと同等のプロパティは `com.fortify.sca.jsp.LegacyDataflow` です。

1.8. Androidプロジェクトの分析

このセクションでは、AndroidアプリケーションのJavaソースコードを変換する方法について説明します。OpenText SASTを使用して、次のいずれからでもGradleでコードをスキャンできます。

- オペレーティングシステムのコマンドライン
- Android Studioで実行中のターミナルウィンドウ

Gradleの使い方はどちらの方法でも同じです。

Note

Fortify Analysis Plugin for IntelliJ IDEA and Android Studioを使用すると、Android StudioからAndroidコードを直接スキャンすることもできます。詳細については、*OpenText™ Fortify Analysis Plugin for IntelliJ IDEA and Android Studioユーザガイド*を参照してください。

このセクションでは、次のトピックについて説明します。

1.8.1. Androidプロジェクト変換の前提条件

Androidプロジェクトを変換するための前提条件は次のとおりです。

- Android Studioと関連するAndroid SDKは、スキャンを実行するシステムにインストールされます。
- Androidプロジェクトでは、ビルドにGradleを使用します。

  Gradleを使用しない古いプロジェクトがある場合は、関連付けられているAndroid StudioプロジェクトにGradleサポートを追加する必要があります。

  Androidプロジェクトの作成に使用するバージョンのAndroid Studioに付属するGradleと同じバージョンを使用します。
- アプリケーションのプロジェクトのAndroidコードをビルドするために必要なすべての依存関係が用意されていることを確認します。
- Android Studio内に表示されないコマンドウィンドウからAndroidコードを変換するには、Gradle Wrapper (`gradlew`)がシステムパスで定義されている必要があります。

1.8.2. Androidコード分析のコマンドライン構文

Androidプロジェクトをスキャンするには、gradlewを使用します。これは、Gradle Wrapperを使用する点以外はGradleを使用することに似ています。Gradle Wrapperを使用してAndroidプロジェクトを変換する方法については、Gradleの統合を参照してください。

1.8.3. Androidレイアウトファイルで検出されたフィルタリングの問題

Androidプロジェクトにレイアウトファイル(ユーザインタフェースの設計に使用)が含まれている場合、プロジェクトファイルにはAndroid Studioによって自動的に生成される `R.java` ソースファイルが含まれている可能性があります。プロジェクトをスキャンすると、OpenText SASTでこれらのレイアウトファイルに関連する問題を検出できます。

レイアウトファイルでレポートされた問題を標準的な監査に含めて、これらの問題が誤検出かどうかを慎重に判断できるようにすることをお勧めします。関心がないレイアウトファイルで問題を特定した後は、「結果の最適化」の説明に従って、問題をフィルタで除外できます。インスタンスIDに基づいて問題をフィルタできます。

1.9. Visual Studioプロジェクトの分析

OpenText SASTでは、次のタイプのVisual Studioプロジェクトの変換をサポートするためのビルド統合を提供しています。

- C/C++プロジェクト
- .NET Frameworkと.NET Coreをターゲットとする C#プロジェクト
- ASP.NETフレームワークとASP.NET Coreをターゲットとする ASP.NETアプリケーション
- Android™およびiOSプラットフォームをターゲットとするXamarinアプリケーション

関連するプログラミング言語およびフレームワークと、Visual StudioおよびMSBuildのサポートされているバージョンの一覧については、「サポートされる言語」および「サポートされるビルドツール 」を参照してください。

このセクションでは、次のトピックについて説明します。

1.9.1. Visual Studioプロジェクト変換の前提条件

変換する各プロジェクトを完成させ、エラーなしでビルドできる環境で変換を実行することをお勧めします。ソフトウェア環境要件のリストについては、「ソフトウェア要件」を参照してください。完全なプロジェクトには、次の要素が含まれます。

- 必要なすべてのソースコードファイル(C/C++、C#、またはVB.NET)。
- 必要なすべての参照ライブラリ。

  これには、関連するフレームワークの参照ライブラリ、NuGetパッケージ、およびサードパーティ製ライブラリが含まれます。
- C/C++プロジェクトの場合は、Visual StudioまたはMSBuildインストールに属していない必要なすべてのヘッダファイルを含めます。
- ASP.NETおよびASP.NET Coreプロジェクトの場合は、必要なすべてのASP.NETページファイルを含めます。

  サポートされているASP.NETページのタイプは、ASAX、ASCX、ASHX、ASMX、ASPX、AXML、BAML、CSHTML、Master、RAZOR、VBHTML、およびXAMLです。

1.9.2. Visual Studioプロジェクトのコマンドライン構文

Visual Studioソリューションまたはプロジェクトを変換する基本的な構文では、OpenText SAST変換コマンドの一部として、プロジェクトの対応するビルドオプションを指定します。そのようにすると、ソリューションおよびプロジェクトファイルを分析し、適切な変換ステップを自動的に実行するビルド統合が開始されます。

Important

適切なプロジェクト依存関係とリソースのすべてをビルド統合が間違いなく取得できるようにするため、OpenText SASTコマンドはビルドの環境設定にアクセスできるコマンドプロンプトから実行する必要があります。変換に最適な環境を確保するために、Visual Studioの開発者コマンドプロンプトからこのコマンドを実行するよう強く推奨します。

次の例では、Visual Studioソリューション `Sample.sln` に含まれるすべてのプロジェクトがOpenText SASTによって変換されます。セミコロンで区切ったプロジェクトのリストを指定して、1つ以上の特定のプロジェクトを変換することもできます。

デフォルトでは、テストプロジェクトは変換から除外されます。NUnit、xunit、またはMSTestを参照するソリューション内のプロジェクトは、テストプロジェクトと見なされます。テストプロジェクトを変換に含めるには、MSBuildオプション `/p:ScaForceTranslateTestProjects=True` をsourceanalyzerコマンドに追加します。

- WindowsまたはLinux上の.NET 6.0以降のソリューションの場合は、次のコマンドを使用してソリューションを変換します。

    1. 必要に応じて次のコマンドを実行し、以前のプロジェクトビルドから中間ファイルを削除します。

        ```
        dotnet clean Sample.sln
        ```

    2. 必要に応じて次のコマンドを実行し、必要なすべての参照ライブラリがダウンロードされ、プロジェクトにインストールされるようにします。次のコマンドをプロジェクトの最上位フォルダから実行します。

        ```
        dotnet restore Sample.sln
        ```

3. プロジェクトのビルドの実装方法に応じて、次のOpenText SASTコマンドのいずれかを実行します。このコマンドには、追加のビルドパラメータを含めることができます。

   ```
   sourceanalyzer –b MyProject dotnet msbuild Sample.sln
   ```

   または

   ```
   sourceanalyzer –b MyProject dotnet build Sample.sln
   ```

- Windows上のC、C++、および.NET Frameworkソリューション(4.8.x以前)の場合は、次のコマンドを使用してソリューションを変換します。

  ```
  sourceanalyzer –b MyProject msbuild /t:rebuild
  [<msbuild_options>] Sample.sln
  ```

  Note

  Visual Studioの開発者コマンドプロンプトではなくWindowsのコマンドプロンプトからOpenText SASTを実行する場合は、環境を設定し、プロジェクトのビルドに必要なMSBuild実行可能ファイルへのパスをPATH環境変数に含める必要があります。

変換が完了したら、次の例に示すように、分析フェーズを実行して結果をFPRファイルに保存します。

```
sourceanalyzer –b MyProject -scan -f MyResults.fpr
```

1.9.3. Visual Studioプロジェクトの変換に関する特殊なケースの処理

このセクションでは、次のトピックについて説明します。

1.9.3.1. スクリプトからの変換の実行

スクリプトなどの非対話型モードで変換を実行するには、OpenText SAST変換を実行する前に次のコマンドを実行して、変換に最適な環境を確立してください。

```
cmd.exe /k <vs_install_dir>/Common7/Tools/VSDevCmd.bat
```

ここで、*<vs_install_dir>*はVisual Studioをインストールしたディレクトリです。

1.9.3.2. プレーンな.NETおよびASP.NETプロジェクトの変換

プレーンな.NETおよびASP.NETプロジェクトは、Windowsコマンドプロンプトおよび Visual Studio環境から変換できます。Windowsコマンドプロンプトから変換する場合は、プロジェクトのビルドに必要なMSBuild実行可能ファイルへのパスがPATH環境変数に含まれていることを確認してください。

1.9.3.3. C/C++およびXamarinプロジェクトの変換

C/C++およびXamarinプロジェクトは、Visual Studioの開発者コマンドプロンプトまたはFortify Extension for Visual Studioから変換する必要があります。

Note

Xamarinプロジェクトでは、対応するネイティブAndroid APIのルールがFortify Secure Coding Rulepacksに存在する場合、Xamarin.Android APIのカスタムルールを使用する必要はありません。使用すると、重複した問題が報告される可能性があります。

1.9.3.4. 空白を含む設定を持つプロジェクトの変換

空白を含む環境設定またはその他の設定ファイルを使用してプロジェクトがビルドされている場合は、設定値を引用符で囲む必要があります。たとえば、環境設定 `Sample.sln` を使用してビルドされたVisual Studioソリューション `My Configuration` を変換するには、次のコマンドを使用します。

```
sourceanalyzer –b MySampleProj msbuild /t:rebuild
/p:Configuration="My Configuration" Sample.sln
```

1.9.3.5. Visual Studioソリューションの単一プロジェクトの変換

Visual Studioソリューションに複数のプロジェクトが含まれている場合は、ソリューション全体ではなく1つのプロジェクトを変換することもできます。プロジェクトファイルには、`proj`で終わるファイル名拡張子(`.vcxproj`や`.csproj`など)が付いています。1つのプロジェクトを変換するには、MSBuildコマンドのパラメータとしてソリューションの代わりにプロジェクトファイルを指定します。

次の例では、`Sample.vcxproj`プロジェクトファイルを変換します。

```
sourceanalyzer –b MySampleProj msbuild /t:rebuild Sample.vcxproj
```

1.9.3.6. 複数の実行可能ファイルをビルドするプロジェクトの分析

Visual StudioまたはMSBuildプロジェクトで複数の実行可能ファイル(ファイル名拡張子 `*.exe` を持つファイルなど)をビルドする場合は、分析結果の誤検知の問題を避けるため、実行ファイルごとに個別に分析フェーズを実行することを強くお勧めします。これを行うには、分析フェーズの実行時に `–binary-name` オプションを使用して、実行可能ファイル名または.NETアセンブリ名をパラメータとして指定します。

次の例は、Sample1 (.NETアセンブリ名が関連付けられていないC++プロジェクト)とSample2 (.NETアセンブリ名 `Sample.sln` を持つ.NETプロジェクト)という2つのプロジェクトで構成されるVisual Studioソリューション `Sample2` を変換して分析する方法を示しています。各プロジェクトは、それぞれ別個の実行可能ファイル(`Sample1.exe` と `Sample2.exe`)をビルドします。分析結果は `Sample1.fpr` ファイルと `Sample2.fpr` ファイルに保存されます。

```
sourceanalyzer -b MySampleProj msbuild /t:rebuild Sample.sln
sourceanalyzer -b MySampleProj -scan -binary-name Sample1.exe -f
Sample1.fpr sourceanalyzer -b MySampleProj -scan -binary-name
Sample2.exe -f Sample2.fpr
```

`-binary-name` オプションについて詳しくは、分析オプションを参照してください。

1.9.4. Visual Studioプロジェクトを変換する別の方法

このセクションでは、Visual Studioプロジェクトを変換する別の方法について説明します。

このセクションでは、次のトピックについて説明します。

1.9.4.1. Visual Studioソリューション用の別の変換オプション

Visual Studioソリューションでのみ使用できる変換方法には、次の2種類があります。

- Fortify Extension for Visual Studioを使用する

  Fortify Extension for Visual Studioは、変換および分析(スキャン)フェーズを1ステップで実行します。

- OpenText SASTコマンドにdevenvコマンドを追加する

  次のコマンドは、Visual Studioソリューション `Sample.sln` を変換します。

  ```
  sourceanalyzer –b MySampleProj devenv Sample.sln /rebuild
  ```

  OpenText SASTはdevenvの呼び出しを同等のMSBuildの呼び出しに変換するため、この場合、このコマンドを使用するソリューションはdevenvツールではなくMSBuildによってビルドされます。

1.9.4.2. OpenText SASTを明示的に実行せずに変換する

OpenText SASTを直接起動せずにVisual Studioプロジェクトを変換するオプションがあります。これには、 `Fortify.targets` および `<sast_install_dir>\Core\private-bin\sca\MSBuildPlugin` ディレクトリの<code><sca_install_dir>\Core\private-bin\sca\MSBuildPlugin</code>にある `Framework` ファイルが必要です。プロジェクトをビルドするビルドコマンドラインで絶対パスまたは相対パスを使用してファイルを指定できます。使用しているビルドコマンド(`Dotnet` または `Framework`)に応じて、それぞれ `dotnet.exe` ディレクトリまたは `MSBuild.exe` ディレクトリを含むパスを使用します。例:

```
dotnet.exe msbuild /t:rebuild
/p:CustomAfterMicrosoftCommonTargets=
<sast_install_dir>\Core\private-
bin\sca\MSBuildPlugin\Dotnet\Fortify.targets Sample.sln
```

または

```
msbuild.exe /t:rebuild /p:CustomAfterMicrosoftCommonTargets=
<sast_install_dir>\Core\private-
bin\sca\MSBuildPlugin\Framework\Fortify.targets Sample.sln
```

プロジェクトの変換を設定するために設定できる環境変数は複数あります。これらの変数の多くはデフォルト値を持ち、OpenText SASTでは変数が設定されていない場合に使用されます。これらの変数を次の表に示します。

環境変数	説明(Description)	デフォルト値
FORTIFY_MSBUILD_BUILDID	変換のOpenText SASTビルドIDを指定します。この値は必ず設定してください。 これは、OpenText SAST `-b` オプションと同じです。	なし
FORTIFY_MSBUILD_DEBUG	デバッグモードを有効にします。これは、OpenText SAST `–debug` オプションと同じです。	False
FORTIFY_MSBUILD_DEBUG_VERBOSE	冗長デバッグモードを有効にします。これは、OpenText SAST `–debug-verbose` オプションと同じです。両方がtrueに設定されている場合は、FORTIFY_MSBUILD_DEBUGよりも優先されます。	False
FORTIFY_MSBUILD_MEM	変換のメモリ要件をJVM `-Xmx` オプションの形式で指定します。たとえば、 `-Xmx2G` になります。	システムで使用可能な物理メモリに基づいて自動割り当て

環境変数	説明(Description)	デフォルト値
FORTIFY_MSBUILD_SCALOG	OpenText SASTログファイルの場所(絶対パス)を指定します。 これは、OpenText SAST `-logfile` オプションと同じです。	`%LOCALAPPDATA%/Fortify/sca/log/sca.log`

1.10. JavaScriptおよびTypeScriptコードの分析

JavaScript、TypeScript、JSX、およびTSXソースファイルが含まれるJavaScriptプロジェクト、およびHTMLファイルに埋め込まれているJavaScriptを分析できます。

一部のJavaScriptフレームワークは、平文のJavaScriptにトランスパイルされ(ソースからソースへのコンパイル)、これが生成されるコードになります。このタイプのコードを除外するには、`-exclude`コマンドラインオプションを使用してください。

JavaScriptコードおよびTypeScriptコードを変換する場合は、すべてのソースファイルを1回の呼び出しで一緒に指定してください。OpenText SASTでは、後続の呼び出し時にビルドIDで関連付けられたファイルリストに新しいファイルを追加する機能はサポートされていません。

OpenText SASTでは、圧縮されたJavaScript (*.min.js)を変換しません。

Note

変換から除外する圧縮されたJavaScriptファイルには、OpenText SASTで自動的に検出できないタイプがあります。これらのファイルは、`-exclude`コマンドラインオプションを使用して直接除外してください。

このセクションでは、次のトピックについて説明します。

1.10.1. ピュアJavaScriptプロジェクトの変換

JavaScriptを変換する基本的なコマンドライン構文は次のとおりです。

```
sourceanalyzer –b <build_id> <js_file_or_dir>
```

ここで、 *<js_file_or_dir>* は変換するJavaScriptファイルの名前、または複数のJavaScriptファイルを含むディレクトリのいずれかです。 *.js に *.js を指定して、複数のファイルを変換することもできます。

1.10.2. 依存関係の除外

特定の依存関係の変換を回避するには、`fortify-sca.properties`ファイル内の適切なプロパティ設定に追加します。次のプロパティで指定されたファイルは変換されません。

- `com.fortify.sca.skip.libraries.ES6`
- `com.fortify.sca.skip.libraries.jQuery`
- `com.fortify.sca.skip.libraries.javascript`
- `com.fortify.sca.skip.libraries.typescript`

各プロパティには、ファイル名(パス情報なし)のカンマまたはコロン区切りリストを指定します。

これらのプロパティで指定されたファイルは、ローカルファイルとインターネット上のファイルの両方に適用されます。たとえば、JavaScriptコードに次のローカルファイル参照が含まれるとします。

```
<script src="js/jquery-ui.js" type="text/javascript"
charset="utf-8"></script>
```

デフォルトでは、`com.fortify.sca.skip.libraries.jQuery`ファイル内の`fortify-sca.properties`プロパティに`jquery-us.js`が含まれるため、OpenText SASTでは前の例に示したファイルを変換しません。

ファイル名には正規表現を使用できます。OpenText SASTでは`'(-?\d+\.\d+\.\d+)?'`プロパティ値に含まれる各ファイル名の`.min.js`または`.js`の前に正規表現`com.fortify.sca.skip.libraries.jQuery`を自動的に挿入します。

Note

`-exclude`コマンドラインオプションを使用して、ローカルファイルまたはディレクトリ全体を除外することもできます。このオプションの詳細については、変換オプションを参照してください。

詳細な分析を行うために、依存関係の言語が`-disable-language`オプションで無効になっている場合でも、依存関係ファイルが変換に含まれます。言語を無効にするオプションの詳細については、「変換オプション」を参照してください。

1.10.3. NPM依存関係の除外

デフォルトで、OpenText SASTではコードにインポートされたNPM依存関係のみを変換します。この動作は、次の2つのプロパティで変更できます。

- `com.fortify.sca.follow.imports` プロパティは、インポートされたファイルを解決して変換に含めることをOpenText SASTに指示します。

  このプロパティは、デフォルトで有効になっています。このプロパティの値をfalseに設定すると、コマンドラインに明示的に含まれていないNPM依存関係は変換に含まれません。

- `com.fortify.sca.exclude.unimported.node.modules` プロパティは、`com.fortify.sca.follow.imports` プロパティによって特定的にインポートされたファイルを除き、node_modulesディレクトリ内のすべてのファイルを変換から除外することをOpenText SASTに指示します。

  このプロパティはデフォルトで有効になっています。これは、ビルドシステムにのみ必要な依存関係など、最終プロジェクトには必要ない依存関係の変換を避けるためです。

1.10.4. NPMの依存関係

デフォルトでは、OpenText SASTはNPM依存関係(`node_modules` ディレクトリ内のファイル)の問題を報告しません。この設定は、 `com.fortify.sca.exclude.node.modules` プロパティで設定されます。このプロパティは、デフォルトで `true` に設定されています。

Note

`com.fortify.sca.exclude.node.modules` が `true` に設定されている場合には、-excludeオプションを使用してノードモジュールを除外することをお勧めしません。結果の品質が変わる可能性があるためです。

参照情報

node_modules依存関係の除外の例

1.10.4.1. NPM依存関係の除外の例

次の例は、NPM依存関係を除外する3つの異なるシナリオを示しています。これらの例では、次のディレクトリ構造を使用します。

```
./ RootProjectDir innerSrcDir node_modules
innerProjectReferencedModule index.ts
moduleNotReferencedByProject index.ts innerProject.ts (contains
import from innerProjectReferencedModule) node_modules
projectReferencedModule index.ts moduleNotReferencedByProject
index.ts projectMain.ts (contains import from
projectReferencedModule)
```

例1

この例は、`com.fortify.sca.exclude.unimported.node.modules`が`false`に設定された状態で、ファイルが変換される場合を示しています。この場合、`com.fortify.sca.follow.imports`と`com.fortify.sca.exclude.unimported.node.modules`が両方とも`true`に設定されています。

```
sourceanalyzer RootProjectDir/ -
Dcom.fortify.sca.exclude.node.modules=false
```

例1の変換には、次のファイルが含まれます。

```
./RootProjectDir/innerSrcDir/innerProject.ts
./RootProjectDir/innerSrcDir/node_modules/innerProjectReferenced
Module/index.ts ./RootProjectDir/projectMain.ts
./RootProjectDir/node_modules/projectReferencedModule/index.ts
```

例2

この例は、プロジェクトによって参照されているモジュールに加えて、解決中に見つかったがプロジェクトによって参照されていないモジュールも変換に含める場合を示しています。

```
sourceanalyzer RootProjectDir/ -
Dcom.fortify.sca.exclude.unimported.node.modules=false
```

例2の変換には、次のファイルが含まれます。

```
./RootProjectDir/innerSrcDir/innerProject.ts
./RootProjectDir/innerSrcDir/node_modules/innerProjectReferenced
Module/index.ts
./RootProjectDir/innerSrcDir/node_modules/moduleNotReferencedByP
roject/index.ts ./RootProjectDir/projectMain.ts
./RootProjectDir/node_modules/projectReferencedModule/index.ts
./RootProjectDir/node_modules/moduleNotReferencedByProject/index
.ts
```

例3

この例は、-excludeオプションを使用して、任意の `node_modules` ディレクトリ内のすべてのファイルを除外する場合を示しています。 `-exclude` オプションは、`com.fortify.sca.follow.imports` および `com.fortify.sca.exclude.unimported.node.modules` プロパティの設定に基づくモジュールの解決策を上書きします。

```
sourceanalyzer RootProjectDir/ -exclude "**/node_modules/*.*"
```

例3の変換には、次のファイルが含まれます。

```
./RootProjectDir/innerSrcDir/innerProject.ts
./RootProjectDir/projectMain.ts
```

1.10.5. HTMLファイルを使用したJavaScriptプロジェクトの変換

プロジェクトにJavaScriptファイルに加えてHTMLファイルが含まれている場合は、次の例に示すように、 `com.fortify.sca.EnableDOMModeling` ファイルまたはコマンドラインで `fortify-sca.properties` プロパティをtrueに設定します。

```
sourceanalyzer –b MyProject <js_file_or_dir> -
Dcom.fortify.sca.EnableDOMModeling=true
```

`com.fortify.sca.EnableDOMModeling` プロパティをtrueに設定すると、DOM関連のクロスサイトスクリプティングの問題など、DOM関連の攻撃に関する偽陰性レポートを減らすことができます。

Note

このオプションを有効にすると、OpenText SASTではHTMLファイル内のDOMツリー構造をモデル化するJavaScriptコードが生成されます。分析フェーズの時間が長くなる場合があります(分析する変換済みコードが多くなるため)。

`com.fortify.sca.EnableDOMModeling` プロパティを `true` に設定した場合は、OpenText SASTがDOMモデリングに含める追加のHTMLタグを `com.fortify.sca.DOMModeling.tags` プロパティで指定することもできます。デフォルトで、OpenText SASTはHTMLタグ `body` 、 `button` 、 `div` 、 `form` 、 `iframe` 、 `input` 、 `head` 、 `html` 、および `p` を含めます。

たとえば、HTMLタグ `ul` および `li` をDOMモデルに追加で含めるには、次のコマンドを使用します。

```
sourceanalyzer –b MyProject <js_file_or_dir> -
Dcom.fortify.sca.DOMModeling.tags=ul,li
```

1.10.6. 外部JavaScriptまたはHTMLを変換に含める

`src` 属性で指定された外部JavaScriptまたはHTMLファイルを含めるために、OpenText SASTでダウンロードして変換フェーズに含められるドメインを指定できます。これを行うには、 `com.fortify.sca.JavaScript.src.domain.whitelist` プロパティで1つ以上のドメインを指定します。

Note

このプロパティは、 `fortify-sca.properties` ファイル内でグローバルに設定できます。

たとえば、HTMLファイルに次のステートメントが含まれているとします。

```
<script src='http://xyzdomain.com/foo/bar.js' language='text/javascript'/>
</script>
```

`xyzdomain.com` ドメインがファイルをダウンロードするのに安全な場所であると確信している場合は、次のプロパティ指定をコマンドラインに追加して、変換フェーズに含めることができます。

```
-
Dcom.fortify.sca.JavaScript.src.domain.whitelist="xyzdomain.com/
foo"
```

Note

プロパティ値では、ドメインから `www.` プレフィックスを省略できます。たとえば、元のHTMLファイルのsrcタグで `www.google.com` からファイルをダウンロードするように指定している場合は、 `google.com` ドメインを指定するだけです。

複数のドメインを信頼するには、次の例に示すように、縦棒文字(`|`)区切りで各ドメインを含める必要があります。

```
-Dcom.fortify.sca.JavaScript.src.domain.whitelist=
"xyzdomain.com/foo|abcdomain.com|123.456domain.com”
```

プロキシサーバを使用している場合は、次の例に示すように、プロキシサーバ情報をコマンドラインに含める必要があります。

```
-Dhttp.proxyHost=example.proxy.com -Dhttp.proxyPort=8080
```

プロキシサーバオプションの完全なリストについては、ネットワーキングプロパティに関するJavaのドキュメントを参照してください。

1.11. PythonおよびJupyter Notebookの分析

OpenText SASTはPythonアプリケーションを変換し、`.py`拡張子を持つファイルをPythonソースコードとして処理します。拡張子が`.ipynb`のファイルはJupyter Notebookとして認識されます。OpenText SASTは、Jupyter NotebookとDjuangoおよびFlaskフレームワークの変換をサポートしています。

このセクションでは、次のトピックについて説明します。

1.11.1. Bazelとの統合

Bazelのビルドと統合するために、OpenText SASTはソースファイルをコンパイル時に変換します。したがって、Bazelのビルドの前提条件は、Bazelのビルドが正常に実行されることです。サポートされているBazelのバージョンについては、「ビルドツール」を参照してください。

Bazelと統合するには、Bazelのワークスペースディレクトリに移動し、構築するBazelのターゲットでs sourceanalyzerを実行します。次のように、変換のための他のsourceanalyzerオプションを指定できます。

```
sourceanalyzer -b <build_id> <sca_options> bazel build <target>
```

プロジェクトを変換し、変換からファイルを除外する場合:

```
sourceanalyzer -b MyProjectC -exclude C:\test\MY-JAVA-
APP\src\proj\content.py bazel build //projc:my-python-prj
```

1.11.1.1. Python Bazelの統合例

特定のターゲットのプロジェクトを変換する場合:

```
sourceanalyzer -b MyProjectA bazel build //proja:my-prj
```

パッケージ `proja/abc` 内のターゲット `abc` を変換する場合:

```
sourceanalyzer -b MyProjectA bazel build //proja/abc
```

または

```
sourceanalyzer -b MyProjectA bazel build //proja/abc:abc
```

パッケージ `proja/abc` 内のすべてのターゲットを変換する場合:

```
sourceanalyzer -b MyProjectA bazel build //proja/abc:all
```

`projb/` ディレクトリ内のすべてのターゲットを変換する場合:

```
sourceanalyzer -b MyProjectB bazel build //projb/...
```

変換にPythonプロジェクトの依存関係を指定する場合:

```
sourceanalyzer -b MyProjectD -python-path
/usr/local/lib/python3.6/ bazel build //projd:my-python-app
```

OpenText SASTとBazelの統合では、複数のターゲットおよび関連アクション(ターゲットの除外など)はサポートされていません。

1.11.2. Python変換コマンドライン構文

Pythonコードを変換する基本的なコマンドライン構文は次のとおりです。

```
sourceanalyzer -b <build_id> -python-version <python_version> -
python-path <dirs> <files>
```

Note

Pythonコードを変換する場合は、すべてのソースファイルを1回の呼び出しで一緒に指定してください。OpenText SASTでは、後続の呼び出し時にビルドIDで関連付けられたファイルリストに新しいファイルを追加する機能はサポートされていません。

1.11.2.1. Pythonコマンドラインオプション

次の表では、Pythonオプションについて説明します。

Pythonオプション	説明(Description)
`-python-version <version>`	スキャンするPythonソースコードのバージョンを指定します。`<version>`の有効な値は、`2`および`3`です。デフォルト値は`3`です。 **同等のプロパティ名:** `com.fortify.sca.PythonVersion`
`-python-no-auto-root-calculation`	モジュールおよびパッケージのインポートに使用する、すべてのプロジェクトソースファイルの共通ルートディレクトリの自動計算を無効にします。 **同等のプロパティ名:** `com.fortify.sca.PythonNoAutoRootCalculation`
`-python-path <dirs>`	追加のインポートディレクトリのセミコロン区切り(Windows)またはコロン区切り(Windows以外)リストを指定します。`-python-path`オプションを使用して、パッケージまたはモジュールのインポートに使用するパスを指定できます。このオプションを使用してネームスペースパッケージディレクトリのすべてのパスを含めてください。OpenText SASTでは、インポートされるファイルごとに指定されたパスを順番に検索し、最初に検出されたファイルを使用します。 **同等のプロパティ名:** `com.fortify.sca.PythonPath`

Pythonオプション	説明(Description)
`-django-template-dirs <dirs>`	Djangoテンプレートを含むディレクトリのセミコロン区切り(Windows)またはコロン区切り(Windows以外)リストを指定します。OpenText SASTでは、Djangoテンプレートファイルごとに指定されたパスを順番に検索し、最初に検出されたテンプレートファイルを使用します。 **同等のプロパティ名:** `com.fortify.sca.DjangoTemplateDirs`
`-django-disable-autodiscover`	OpenText SASTでDjangoテンプレートを自動検出しないことを指定します。 **同等のプロパティ名:** `com.fortify.sca.DjangoDisableAutodiscover`
`-jinja-template-dirs <dirs>`	Jinja2テンプレートを含むディレクトリのセミコロン区切り(Windows)またはコロン区切り(Windows以外)リストを指定します。OpenText SASTでは、Jinja2テンプレートファイルごとに指定されたパスを順番に検索し、最初に検出されたテンプレートファイルを使用します。 **同等のプロパティ名:** `com.fortify.sca.JinjaTemplateDirs`

Pythonオプション	説明(Description)
`-disable-template-autodiscover`	OpenText SASTでDjangoまたはJinja2テンプレートを自動検出しないことを指定します。 **同等のプロパティ名:** `com.fortify.sca.DisableTemplateAutodiscover`

Pythonのプロパティ

1.11.2.2. Pythonのコマンドライン例

WindowsでPython 3コードを変換する場合:

```
sourceanalyzer -b Python3Proj -python-path
"C:\Python312\Lib;C:\Python312\Lib\site-packages" src/*.py
```

WindowsでPython 2コードを変換する場合:

```
sourceanalyzer -b MyPython2 -python-version 2 -python-path
"C:\Python27\Lib;C:\Python27\Lib\site-packages" src/*.py
```

Windows以外でPython 3コードを変換する場合:

```
sourceanalyzer -b Python3Proj -python-path
/usr/lib/python3.12:/usr/local/lib/python3.12/site-packages
src/*.py
```

Windows以外でPython 2コードを変換する場合:

```
sourceanalyzer -b MyPython2 -python-version 2 -python-path
/usr/lib/python2.7:/usr/local/lib/python2.7/site-packages
src/*.py
```

1.11.3. 仮想環境でのPythonの変換

このセクションでは、仮想環境でPythonプロジェクトを変換する方法について説明します。プロジェクトのすべての依存関係が仮想環境にインストールされていることを確認してください。仮想環境でPythonプロジェクトを変換するには、プロジェクトの依存関係を指定する `-python-path` オプションを含めます。

Python仮想環境の例

仮想環境名が `myenv` で、プロジェクトの依存関係が `myenv/lib/python<version>/site-packages` ディレクトリにインストールされているPythonプロジェクトを変換するには、次のように入力します。

```
sourceanalyzer –b mybuild -python-path
"myenv/lib/python<version>/site-packages/" myproject/
```

conda環境の例

conda環境名が `myenv` で、プロジェクトの依存関係が `<conda_install_dir>/envs/myenv/lib/python<version>/site-packages` ディレクトリにインストールされているPythonプロジェクトを変換するには、次のように入力します。

```
sourceanalyzer –b mybuild -python-path "
<conda_install_dir>/envs/myenv/lib/python<version>/site-
packages/" myproject/
```

1.11.4. インポート済みのモジュールとパッケージを含める

Pythonアプリケーションを変換してスキャンに備えるため、OpenText SASTではアプリケーションが使用するインポート済みのモジュールおよびパッケージを検索します。OpenText SASTでは、Pythonランタイムシステムがインポート済みのモジュールとパッケージを検索するために使用する `PYTHONPATH` 環境変数を考慮しません。

OpenText SASTでは、次の順序でディレクトリのリストを使用して、インポート済みのモジュールとパッケージを検索します。

1. すべてのプロジェクトソースファイルの共通ルートディレクトリ。OpenText SASTが自動的に計算します。たとえば、2つのプロジェクトディレクトリ `PrimaryDir/project1/*` および `PrimaryDir/project2/*` が存在する場合、共通ルートディレクトリは `PrimaryDir` です。

   インポート済みモジュールおよびパッケージの検索ターゲットである共通ルートディレクトリを削除するには、変換コマンドに `-python-no-auto-root-calculation` オプションを含める必要があります。

2. `-python-path` オプションで指定されたディレクトリ。

   OpenText SASTでは、標準Pythonライブラリのモジュールのサブセットを変換に含めます(「builtins」モジュール、もともとCで記述されたすべてのモジュールなど)。OpenText SASTでは最初にOpenText SASTに含まれるセットで標準Pythonライブラリモジュールを検索し、次に `-python-path` オプションで指定されたパスで検索します。OpenText SASTで見つからないモジュールをPythonコードでインポートすると、警告が表示されます。標準Pythonライブラリのすべてのモジュールが見つかるようにするには、 `-python-path` リストで標準Pythonライブラリへのパスを追加します。

3. 変換中のファイルを含むカレントディレクトリ。たとえば、OpenText SASTで `PrimaryDir/project1/a.py` を変換すると、インポート済みモジュールおよびパッケージを検索する最後のディレクトリとしてディレクトリ `PrimaryDir/project1` が追加されます。

1.11.5. ネームスペースパッケージを含める

ネームスペースパッケージを変換するには、`-python-path`オプションを使用してネームスペースパッケージディレクトリへのすべてのパスを含める必要があります。たとえば、複数のフォルダにネームスペースパッケージ`package_name`の2つのサブパッケージが含まれている場合は:

```
/path_1/package_name/subpackageA
/path_2/package_name/subpackageB
```

Sourceanalyzerコマンドラインに、`/path_1;/path_2`オプションとともに`-python-path`を含めます。

1.11.6. DjangoとFlaskの変換

デフォルトでは、OpenText SASTはプロジェクトのルートディレクトリ内のDjangoおよびJinja2テンプレートの検出を試みます。検出されたDangoおよびJinja2テンプレートはすべて自動的に変換に追加されます。Sourceanalyzerコマンドに `-django-template-dirs` または `-jinja-template-dirs` オプションを追加して、DjangoまたはJinja2テンプレートファイルの追加の場所を指定できます。

OpenText SASTでDjangoおよびJinja2テンプレートを自動的に検出しないようにする場合には、 `-disable-template-autodiscover` オプションを使用します。プロジェクトにDjangoまたはJinja2テンプレートが必要なのに、テンプレートが予期しない場所にあるようにプロジェクトが設定されている場合には、次のWindows以外の例に示されているように、 `-disable-template-autodiscover` オプションに加えて `-django-template-dirs` または `-jinja-template-dirs` オプションを使用して、テンプレートを含むディレクトリを指定します。

```
sourceanalyzer -b djangoProj -python-path
/usr/lib/python3.12:/usr/local/lib/python3.12/site-packages
djangoProj -django-template-dirs
djangoProj/templatedir1:/djangoProj/dir2 -disable-template-
autodiscover
```

```
sourceanalyzer -b flaskProj -python-path
/usr/lib/python3.12:/usr/local/lib/python3.12/site-packages
flaskProj -jinja-template-dirs
flaskProj/templatedir1:/flaskProj/dir2 -disable-template-
autodiscover
```

次の例では、Windows上でDangoテンプレートとJinja2テンプレートを組み合わせて使用するPythonプロジェクトを変換します。

```
sourceanalyzer -b pythonProj -python-path
"C:\Python312\Lib;C:\Python312\Lib\site-packages" flaskProj -
django-template-dirs
"C:\djangoProj\templatedir1;C:\djangoProj\dir2" -jinja-template-
dirs "C:\flaskProj\templatedir1;C:\flaskProj\dir2" -disable-
template-autodiscover
```

1.12. CおよびC++コードの分析

このセクションでは、CおよびC++コードを変換する方法について説明します。OpenText SASTは標準のANSI CおよびC++をサポートしており、すべての非標準のC++構成をサポートしているとは限りません。

Important

このセクションでは、Visual StudioまたはMSBuildプロジェクトの一部*ではない*CおよびC++コードを変換する方法について説明します。Visual StudioまたはMSBuildプロジェクトの変換方法については、「Visual StudioおよびMSBuildプロジェクトの変換」を参照してください。

このセクションでは、次のトピックについて説明します。

1.12.1. CおよびC++コード変換の前提条件

プロジェクトをビルドするために必要な依存関係(サードパーティ製ライブラリのヘッダを含む)があることを確認してください。OpenText SAST変換では、オブジェクトファイルと静的/動的ライブラリのファイルは必要ありません。

1.12.2. Makeとの統合

OpenText SASTをmakeと統合するには、ビルドプロセスのために`sourceanalyzer`とMakeを組み合わせて実行します。次のビルドコマンドを使用してプロジェクトを構築する場合の例:

```
make clean make make install
```

次のサンプルコマンドを使用して、プロジェクト全体を同時に変換およびコンパイルできます。

```
make clean sourceanalyzer -b MyProject make make install
```

ビルド統合の代替方法として、ビルドスクリプトを変更して、各コンパイラ、リンカ、およびアーカイバ操作の前に`sourceanalyzer`コマンドを追加することもできます。たとえば、makefileでは、これらのツールの名前に変数が定義されることが多いです。

```
CC=gcc
CXX=g++
LD=ld AR=ar
```

makefile内のツール参照の前に、`sourceanalyzer`コマンドと適切なオプションを付加できます。

```
CC=sourceanalyzer -b MyProject gcc CXX=sourceanalyzer -b
MyProject g++ LD=sourceanalyzer -b MyProject ld
AR=sourceanalyzer -b MyProject ar
```

各操作に同じビルドIDを使用する場合は、OpenText SASTで自動的に、個別に変換された各ファイルが変換された単一のプロジェクトに結合されます。

1.12.3. CMakeとの統合

Windows以外のシステムでは、OpenText SASTコマンドにJSONコンパイルデータベースを組み込むことによって、CMakeを使用してビルドされたプロジェクトを変換できます。これは、MakefileジェネレータとNinjaジェネレータでのみサポートされています(詳細については、『CMake Reference Documentation』を参照してください)。

OpenText SASTをCMakeビルドと統合するには:

1. CMakeプロジェクト用の `compile_commands.json` ファイルを生成します。

   `-DCMAKE_EXPORT_COMPILE_COMMANDS=yes` 設定コマンドに `cmake` を追加します。例:

   ```
   cmake -G Ninja -DCMAKE_EXPORT_COMPILE_COMMANDS=yes
   ```

2. 次のようにして、 `sourceanalyzer` コマンドにJSONコンパイルデータベースを含めます。

   ```
   sourceanalyzer -b <build_id> compile_commands.json
   ```

1.12.4. Gradleとの統合

Gradle統合には、C++アプリケーションプラグインに関する前提条件があります。Gradleファイルに次のいずれかの形式で追加してください。

```
apply plugin: 'cpp'
```

```
plugins { id 'cpp-application' }
```

Gradleの統合は、Gradleまたはgradlewのコマンドラインにsourceanalyzerコマンドを前に付加するだけで、次のように簡単にできます。

```
sourceanalyzer -b <build_id><sca_options> gradle
[<gradle_options>] <gradle_tasks>
```

詳細なガイドについては、「JavaとKotlinの統合: Gradle統合の使用」を参照してください。

1.12.5. CおよびC++の手動変換構文

コンパイラに渡されるコマンドラインオプションは、プリプロセッサの実行に影響を与え、言語の機能や拡張機能を有効または無効にすることができます。OpenText SASTでコンパイラと同じようにソースコードを解釈するには、C/C++ソースコードの変換フェーズで完全なコンパイラコマンドラインが必要です。元のコンパイラコマンドの前に `sourceanalyzer` コマンドとオプションを指定します。

1つのファイルを変換する基本的なコマンドライン構文は次のとおりです。

```
sourceanalyzer -b <build_id> [<sca_options>] <compiler>
[<compiler_options>] <file>.c
```

ここで:

- *<sca_options>* OpenText SASTに渡されるオプションです。
- *<compiler>* 使用するC/C++コンパイラの名前(`gcc` 、 `g++` 、 `cl` など)です。サポートされているC/C++コンパイラのリストについては、「サポートされる言語」を参照してください。
- *<compiler_options>* C/C++コンパイラに渡されるオプションです。
- *<file>*.c ASCIIまたはUTF-8エンコーディング形式である必要があります。

Note

OpenText SASTのすべてのオプションをコンパイラオプションの前に指定する必要があります。

コンパイラコマンドは、単独で実行したときに正常に完了する必要があります。コンパイラコマンドが失敗する場合は、コンパイラコマンドの前に指定したOpenText SASTコマンドも失敗します。

たとえば、次のコマンドを使用してファイルをコンパイルするとします。

```
gcc -I. -o hello.o -c helloworld.c
```

この場合、次のコマンドを使用してこのファイルを変換できます。

```
sourceanalyzer -b MyProject gcc -I. -o hello.o -c helloworld.c
```

OpenText SASTは、変換フェーズの一部として元のコンパイラコマンドを実行します。前のコマンド例では、スキャンに適した変換済みのソースと、`hello.o`を実行して得られるオブジェクトファイル`gcc`の両方が生成されます。OpenText SAST `-nc`オプションを使用して、コンパイラの実行を無効にすることができます。

1.12.6. 前処理されたCおよびC++コードのスキャン

コンパイルの前にC/C++ビルドでOpenText SASTがサポートしていないサードパーティ製のCプリプロセッサを実行する場合は、中間ファイルでOpenText SAST変換を開始する必要があります。OpenText SASTのタッチレスビルド統合では、ビルドでサポートされていないプリプロセッサとサポートされているコンパイラをパイプチェーンではなく一時ファイルで接続された2つのコマンドとして実行した場合に、中間ファイルが自動的に変換されます。

1.12.7. C/C++のプリコンパイル済みヘッダファイル

一部のC/C++コンパイラは、コンパイルのパフォーマンスを向上させるプリコンパイル済みヘッダファイルをサポートしています。コンパイラによっては、この機能の実装によって微妙な副作用が発生します。この機能を有効にすると、コンパイラが警告やエラーなしで誤ったソースコードを受け入れる可能性があります。その結果、OpenText SASTで変換エラーが報告されてもコンパイラでは報告されないという矛盾が発生する可能性があります。

コンパイラのプリコンパイル済みヘッダ機能を使用する場合は、プリコンパイル済みヘッダを無効にしてから、完全ビルドを実行してソースコードが正しくコンパイルされるのを確認してください。

1.13. iOSおよびXcodeプロジェクトの分析

このセクションでは、iOSアプリケーションのSwift、Objective-C、およびObjective-C++ソースコードを変換する方法について説明します。プロジェクトのソースファイルを識別するために、OpenText SASTは自動的にXcodeコマンドラインツールのXcodebuildに統合されます。

このセクションでは、次のトピックについて説明します。

1.13.1. iOSプロジェクト変換の前提条件

iOSプロジェクトを変換するための前提条件は次のとおりです。

- Objective-C++プロジェクトは、非脆弱なObjective-Cランタイム(ABIバージョン2または3)を使用する必要があります。
- Appleの `xcode-select` コマンドラインツールを使用して、Xcodeパスを設定します。OpenText SASTでは、システムグローバルなXcode設定を使用して、Xcodeツールチェーンとヘッダを検索します。
- 正常なXcodeビルドに必要なすべてのソースファイルが提供されていることを確認します。

  `-exclude` オプションを使用して、分析からファイルを除外できます(iOSコード分析のコマンドライン構文を参照)。
- プロジェクトをビルドするために必要な依存関係が用意されている必要があります。
- Swiftコードを変換するには、CocoaPodsを含むすべてのサードパーティモジュールを使用できることを確認します。ブリッジヘッダも使用可能である必要があります。ただしXcodeでは、通常、ビルド中にこれらのファイルを自動的に生成します。
- プロジェクトにバイナリ形式のプロパティリストファイルが含まれる場合は、先にファイルをXML形式に変換する必要があります。これを行うには、Xcodeの `putil` コマンドを使用できます。
- Objective-Cプロジェクトを変換するには、サードパーティライブラリのヘッダが使用可能な必要があります。
- Watchkit®アプリケーションを変換するには、iPhoneアプリケーションターゲットとWatchKit Extensionターゲットの両方を変換してください。

1.13.2. iOSコード分析のコマンドライン構文

次のコマンドライン構文を使用して、iOSコードを変換できます。

```
sourceanalyzer -b <build_id> xcodebuild [<compiler_options>]
```

ここで、 *<compiler_options>* はXcodeコンパイラに渡される、サポートされているオプションです。何らかの build を指定した <compiler_options> オプションを含める必要があります。OpenText SAST Xcodebuildの統合では、 xcodebuild archive などの代替ビルドコマンドの出力形式はサポートされていません。

Note

このコマンドを実行すると、xcodebuildによってソースコードがコンパイルされます。

分析からファイルを除外するには、 -exclude オプションを使用します(変換オプションを参照)。ファイルがXcodeのビルドに含まれている場合でも、除外指定に一致するすべてのソースファイルが変換されません。次に例を示します。

```
sourceanalyzer -b MyProject -exclude "**/TestFile.swift"
xcodebuild clean build
```

アプリケーションがプロパティリストファイル(たとえば *<file>*.plist)を使用している場合は、これらのファイルを別の sourceanalyzer コマンドで変換します。プロジェクトファイルの変換に使用したものと同じビルドIDを使用します。次に例を示します。

```
sourceanalyzer -b MyProject <path_to_plist_files>
```

プロジェクトでCocoaPodsを使用している場合は、 -workspace を含めてプロジェクトをビルドします。例:

```
sourceanalyzer -b DemoAppSwift xcodebuild clean build -workspace
DemoAppSwift.xcworkspace -scheme DemoAppSwift -sdk
iphonesimulator
```

変換が完了したら、次の例に示すように、分析フェーズを実行して結果をFPRファイルに保存できます。

```
sourceanalyzer -b DemoAppSwift -scan -f MyResults.fpr
```

1.14. PHPコードの分析

`MyPHP.php` という名前の単一のPHPファイルを変換する構文を次の例に示します。

```
sourceanalyzer -b <build_id> MyPHP.php
```

ソースまたは `php.ini` ファイルエントリに相対パス名が含まれる(`./` または `../` で始まる)ファイルを変換するには、次の例に示すようにPHPソースルートを設定します。

```
sourceanalyzer -php-source-root <path> -b <build_id> MyPHP.php
```

`-php-source-root` オプションの詳細については、「PHPコマンドラインオプション」の説明を参照してください。

PHPコードを変換する場合は、すべてのソースファイルを1回の呼び出しで一緒に指定してください。OpenText SASTでは、後続の呼び出し時にビルドIDで関連付けられたファイルリストに新しいファイルを追加する機能はサポートされていません。

このセクションでは、次のトピックについて説明します。

1.14.1. PHPコマンドラインオプション

次の表では、PHP固有のコマンドラインオプションについて説明します。

PHPオプション	説明(Description)
`-php-source-root <path>`	プロジェクトルートディレクトリへの絶対パスを指定します。相対パス名は最初にカレントディレクトリから展開されます。ファイルが見つからない場合は、指定したPHPソースルートディレクトリからパスが展開されます。 **同等のプロパティ名:** `com.fortify.sca.PHPSourceRoot`
`-php-version <version>`	PHPのバージョンを指定します。デフォルトのバージョンは8.2です。有効なバージョンのリストについては、「サポートされる言語」を参照してください。 **同等のプロパティ名:** `com.fortify.sca.PHPVersion`

PHPのプロパティ

1.15. Goコードの分析

このセクションでは、Goコードを変換する方法について説明します。OpenText SASTでは、Windows、Linux、およびmacOS®上のGoコードの分析をサポートしています。

このセクションでは、次のトピックについて説明します。

1.15.1. Goコマンドライン構文

最善の結果を得るには、プロジェクトがコンパイル可能で、必要なすべての依存関係を利用できる必要があります。

次のエンティティは、変換(およびスキャン)から除外されます。

- ベンダフォルダ
- サブフォルダ内の任意の `go.mod` ファイルによって定義されたプロジェクト(%PROJECT_ROOT%の下にある `go.mod` ファイルによって定義されたプロジェクトを除く)
- `_test.go` サフィックスが付くすべてのファイル(ユニットテスト)

Goコードを変換する基本的なコマンドライン構文は次のとおりです。

```
sourceanalyzer -b <build_id> [-gopath <dir>] [-goroot <dir>]
<files>
```

最善の結果を得るために、すべてのGoプロジェクトにGoモジュールを使用し、Goコードを1つずつモジュールに変換するようお勧めします。sourceanalyzerコマンドの*<files>*パラメータの値が、 `go.mod` ファイルを含むディレクトリ内に存在するようにしてください。これは、プロジェクトをビルドするために `go build` コマンドを実行するディレクトリと同じです。プロジェクトが複数のモジュールで構成されている場合には、sourceanalyzerコマンドを同じ*<build_id>*値で複数回実行して、すべてのモジュールの変換結果をまとめることができます。

ビルド用のGOPATH開発モードの使用は引き続きサポートされますが、これは、2つのスキャンをFireify Audit WorkbenchやApplication Securityなどのツールで比較しようとする場合に問題を引き起こす可能性があります。モジュールの固定識別子パスを定義する `go.mod` ファイルがない場合、Go言語システムは各モジュールをローカルファイルシステム上の絶対パスで識別します。したがって、異なるサブディレクトリや異なるマシンから同じモジュールを2回スキャンすると、異なるモジュール識別子が生成されるため、マッチング問題が2つのスキャン間で正しく関連付けられません。GOPATH開発モードは、Goコンパイラおよびsdkに対して非推奨となり、今後のGo 1.xxリリースで削除される予定です。

1.15.2. Goコマンドラインオプション

次の表では、Goコード変換専用のコマンドラインオプションについて説明します。

Goオプション	説明(Description)
`-gotags <go_build_tags>`	Goプロジェクトのカスタムビルドタグのカンマ区切りリストを指定します。これは、`go`コマンドの`-tags`オプションに相当します。詳細については、「カスタムGoビルドタグを含める」を参照してください。 **同等のプロパティ名:** `com.fortify.sca.gotags`

Goオプション	説明(Description)

`-gopath <dir>`	Goプロジェクトの変換に使用するGOPATH環境変数の値を指定します。このオプションを指定しない場合、OpenText SASTではGOPATHシステム環境変数の既存の値が使用されます。
	
	gopathディレクトリは絶対パスとして指定する必要があります。次の例は、`<dir>`に有効な値です。
	
	`/home/projects/go_workspace/my_proj`
	`C:\projects\go_workspace\my_proj`
	
	次の例は、`<dir>`に無効な値です。
	
	`go_workspace/my_proj`
	
	このオプションとGOPATHシステム環境変数が設定されていない場合、gopathのデフォルトは、ユーザのホームディレクトリにある`go`という名前のサブディレクトリです(Linuxの場合は`$HOME/go`、Windowsの場合は`%USERPROFILE%\go`)。ただし、そのディレクトリにGoディストリビューションが含まれている場合を除きます。
	
	モジュールを使用する場合、GOPATH環境変数でパッケージのインポートを解決する必要はありません。ただし、不足しているモジュールの依存関係をダウンロードするときに使用する出力ディレクトリは、引き続きGOPATHによって決定されます。

Goオプション	説明(Description)
	Note OpenText SASTでは、パッケージのインポートを解決するためにGOPATH環境変数のみに依存する古いGoプロジェクトを完全にはサポートしていません。 **同等のプロパティ名:** `com.fortify.sca.GOPATH`
`-goroot <dir>`	Goインストールの場所を指定します。このオプションを指定しない場合は、GOROOTシステム環境変数が使用されます。 このオプションが指定されず、GOROOTシステム環境変数が設定されていない場合、OpenText SASTではOpenText SASTインストールに含まれるGoコンパイラを使用します。 **同等のプロパティ名:** `com.fortify.sca.GOROOT`

Goオプション	説明(Description)
`-goproxy <url>`	1つ以上のプロキシURLをカンマ区切りで指定します。また、`direct`または`off`(ネットワークの使用を無効化)を指定することもできます。 このオプションが指定されず、GOPROXYシステム環境変数が設定されていない場合、OpenText SASTは`https://proxy.golang.org,direct`を使用します。 **同等のプロパティ名:** `com.fortify.sca.GOPROXY`

Goのプロパティ

1.15.3. カスタムGoビルドタグを含める

Goプロジェクトにカスタムビルドタグを必要とするファイルが含まれる場合には、`-gotags` オプションを使用して、これらのビルドタグをOpenText SAST変換に含めることができます。例:

```
sourceanalyzer -b MyProject -gotags release "src/**/*.go"
```

OpenText SAST `-gotags` オプションでは、オペレーティングシステム、アーキテクチャ、またはGoバージョン(`//go:build linux` 、 `//go:build arm` 、 `//go:build go1.21` など)の自動ビルドタグの上書きを許可しません。Goプロジェクトを別のオペレーティングシステムまたはアーキテクチャ向けに変換するには、GOOSおよびGOARCH環境変数で適切なクロスコンパイルターゲットを設定します。特定のGoバージョンを設定するには、GOROOT環境変数または `-goroot` オプションで、Go SDKバージョンのパスを指定します。

1.15.4. 依存関係の解決

OpenText SASTでは、Goに組み込む2つの依存関係管理システムをサポートしています。

- モジュール

  モジュールを使用するGoプロジェクトを変換するには、必要な依存関係を指定する `go.mod` ファイルと、ダウンロードした依存関係を検証するための対応する `go.sum` ファイルを、プロジェクトに含める必要があります。sourceanalyzerコマンドで、 `go.mod` ファイルを含むディレクトリをプロジェクトルートとして指定します。

  OpenText SASTでは、ネイティブのGoツールチェーンを使用して必要なすべての依存関係をダウンロードします。OpenText SASTを実行するコンピュータでインターネットへのアクセスが制限されている場合は、次のいずれかを実行します。

  - Artifactoryなどのアーティファクト管理システムを使用している場合には、GOPROXY環境変数を設定するか、「Goコマンドラインオプション」で説明されている `-goproxy` オプションを使用します。
  - モジュールとベンダを使用して、必要なすべての依存関係をダウンロードします。

    手動ベンダを使用する場合には、変換を開始する前に、GOFLAGS環境変数を `-mod=vendor` に設定します。

- GOPATHの依存関係の解決

  depなどのサードパーティの依存関係管理システムを使用している場合は、変換を開始する前に、すべての依存関係をダウンロードする必要があります。

  GOPATH開発モードでは、ローカルファイルシステム上の絶対パスを使用して依存関係を識別します。この場合、異なるサブディレクトリや異なるマシンからのスキャンを相互に関連付けるときに問題が発生する可能性があります。

参照情報

Goコマンドライン構文

1.16. DartおよびFlutterコードの分析

このセクションでは、DartおよびFlutterコードを変換する方法について説明します。OpenText SASTでは、WindowsおよびLinux上のDartおよびFlutterコードの分析をサポートしています。

このセクションでは、次のトピックについて説明します。

1.16.1. DartおよびFlutter変換の前提条件

DartプロジェクトとFlutterプロジェクトを変換するための前提条件は次のとおりです。

- サポートされているDart SDK (Dart専用プロジェクトの場合)およびFlutter SDK (Flutterプロジェクトの場合)がシステムにインストールされている必要があります。サポートされているDartおよびFlutter SDKのバージョンについては、「サポートされている言語」を参照してください。
- 次のいずれかのコマンドを実行して、プロジェクトの依存関係をダウンロードします。
  - Flutterプロジェクトの場合は、`flutter pub get`を使用します。
  - Dart専用プロジェクトの場合は、`dart pub get`を使用します。

  たとえば、プロジェクトルートが`myproject`のFlutterプロジェクトの依存関係をダウンロードするには、次のコマンドを実行します。

```
cd myproject flutter pub get
```

> **Important**
>
> プロジェクトに、`pubspec.yaml`ファイルが異なるネストされたパッケージが含まれている場合は、パッケージルートごとに`dart pub get`または`flutter pub get`を実行する必要があります。

> **Important**
>
> プロジェクトディレクトリに次のものが含まれている必要があります。
>
> - `pubspec.yaml`ファイル。このファイルは依存関係を指定します
> - `.dart_tool`ディレクトリ。このディレクトリには、`pub`ツールによって自動的に生成される`package_config.json`ファイルが含まれます

1.16.2. DartおよびFlutterのコマンドライン構文

DartおよびFlutterのコードを変換する基本的なコマンドライン構文は次のとおりです。

```
sourceanalyzer –b <build_id> <translation_options> <dirs>
sourceanalyzer –b <build_id> <translation_options> <files>
```

1.16.3. DartおよびFlutterのコマンドライン例

`my_app` プロジェクトルートディレクトリを使用してDartプロジェクトまたはFlutterプロジェクトを変換するには:

```
sourceanalyzer -b MyProject my_app/
```

`a_widget.dart` プロジェクトルートディレクトリ内の `my_app` ファイルを変換するには、次のコマンドを使用します。

```
sourceanalyzer -b MyProject my_app/a_widget.dart
```

`my_dart_proj` ディレクトリ内のすべてのdartソースファイルを変換するには:

```
sourceanalyzer -b MyProject "my_dart_proj/**/*.dart"
```

1.17. Salesforce ApexおよびVisualforceコードの分析

このセクションでは、次のトピックについて説明します。

1.17.1. ApexおよびVisualforce変換の前提条件

ApexプロジェクトやVisualforceプロジェクトを変換するには、スキャンするすべてのソースコードが、OpenText SASTをインストールしたのと同じマシン上で使用可能である必要があります。

カスタムSalesforce®アプリをスキャンするには、そのアプリを開発して展開したSalesforce組織(org)からローカルコンピュータにダウンロードします。ダウンロードしたアプリは次の要素で構成されています。

- `.cls` 拡張子を持つファイル内のApexクラス
- `.page` 拡張子を持つファイル内のVisualforce Webページ
- `.trigger` 拡張子を持つファイルに含まれている、データベースの「trigger」関数と呼ばれるApexコードファイル
- `.component` 拡張子を持つファイル内のVisualforceコンポーネントファイル
- `.object` 拡張子を持つファイル内のオブジェクト

Salesforce Webサイトで入手できるAnt移行ツールを使用して、Salesforceクラウド内の組織からローカルコンピュータにアプリをダウンロードします。プロジェクトマニフェストファイルが、`build.xml` ファイル内の指定したターゲットに対して正しく設定されていることを確認します。たとえば、次の `package.xml` マニフェストファイルは、OpenText SASTに、すべてのクラス、カスタムオブジェクト、ページ、およびコンポーネントを提供します。

```
 <?xml version="1.0" encoding="UTF-8"?> <Package
xmlns=http://soap.sforce.com/2006/04/metadata> <types>
<members>*</members> <name>ApexClass</name> </types> <types>
<members>*</members> <name>ApexTrigger</name> </types> <types>
<members>*</members> <name>ApexPage</name> </types> <types>
<members>*</members> <name>ApexComponent</name> </types> <types>
<members>*</members> <name>CustomObject</name> </types>
<version>55.0</version> </Package>
```

Ant移行ツールのドキュメントを使用して、取得ターゲットを設定します。AppExchangeのアプリを組織で使用している場合、それらのアプリはパッケージ化されたターゲットとしてダウンロードされる必要があります。

1.17.2. ApexおよびVisualforceのコマンドライン構文

ApexおよびVisualforceのコードを変換する基本的なコマンドライン構文は次のとおりです。

```
sourceanalyzer -b <build_id> <files>
```

ここで、 *<files>* はApexまたはVisualforceのファイルまたはソースファイルへのパスです。

Important

ソースファイルでサポートされているファイル拡張子は、 .cls 、 .component 、 .trigger 、 .object 、および .page です。

1.18. ABAPコードの分析

ABAPコード変換では、SAP®データベースからコードを抽出してスキャン用に準備するための追加の準備手順が必要です。詳しくは、移送依頼のインポートを参照してください。このセクションは、SAPとABAPの基本を理解していることを前提としています。

このセクションでは、次のトピックについて説明します。

1.18.1. ソースファイルのダウンロードについて

ABAPコードを変換するために、Fortify ABAP Extractorプログラムはソースファイルをプレゼンテーションサーバにダウンロードし、必要に応じてOpenText SASTを開始します。 ファイルをローカルシステムにダウンロードしてオペレーティングシステムのコマンドを実行するには、アクセス許可を持つアカウントを使用する必要があります。

Extractorプログラムはオンラインで実行されるため、抽出するために選択されたソースファイルのボリュームが許容されるプロセス実行時間を超えた場合は、`max dialog work process time reached`メッセージを受信することがあります。これを回避するには、大規模なプロジェクトを連続する小さいExtractorタスクとしてダウンロードします。 たとえば、プロジェクトが4つの異なるパッケージで構成されている場合、各パッケージを同じプロジェクトディレクトリに個別にダウンロードします。 例外が頻繁に発生する場合は、SAP Basis管理者と協力して最大時間制限(`rdisp/max_wprun_time`)を大きくします。

ABAPからパッケージが抽出されたら、Fortify ABAP Extractorは`TDEVC`からパッケージ名と一致する`parentcl`フィールドを持つすべてを抽出します。次に、`TDEVC`から抽出したものと同じ`parentcl`フィールドを持つ残りのすべてを`TDEVC`から再帰的に抽出します`TDEVC`から抽出されるフィールドは<code>devclass</code>です。`TDEVC`から抽出されるフィールドは`devclass`です。

`devclass`の値はプログラム名のセットとして扱われ、指定できるプログラム名と同様に処理されます。

`TRDIR`からプログラムを抽出するには、名前フィールドを次のいずれかと比較します。

- 選択画面で指定したプログラム名
- `TDEVC`から抽出された値のリスト(パッケージが提供された場合)

`TRDIR`の行は名前フィールドに指定したプログラム名を含む行であり、式`LIKE programname`を使用して行が抽出されます。

この最終的な名前のリストを`READ REPORT`で使用して、SAPシステムからコードを取得します。このメソッドは、レコードに対して単に`REPORTS`だけでなく、クラスとメソッドも読み込みます。

`READ REPORT`を呼び出すたびに、ローカルシステムの一時フォルダにファイルが作成されます。OpenText SASTは、この一連のファイルを変換およびスキャンしてFPRファイルを作成します。FPRファイルは、Fortify Audit Workbenchを使用して開きます。

ABAPのプロパティ

1.18.1.1. INCLUDEの処理

ソースコードがダウンロードされると、Fortify ABAP Extractorはソース内の `INCLUDE` ステートメントを検出します。検出すると、分析のためにincludeのターゲットをローカルコンピュータにダウンロードします。

1.18.2. 移送依頼のインポート

ABAPコードをスキャンするには、Fortify ABAP Extractorの移送依頼をSAPサーバにインポートする必要があります。移送依頼は `<sast_install_dir>/Tools/SAP_Extractor.zip` にあります。

Fortify ABAP Extractorのパッケージ(`SAP_Extractor.zip`)には次のファイルが含まれています。

- `K900<release_number>.<system_id>`
- `R900<release_number>.<system_id>`

これらのファイルが、ローカル移送ドメインの外部からSAPシステムにインポートする必要があるSAP移送依頼の構成要素です。SAP管理者または移送依頼をシステムにインストールする権限を持つ個人に、移送依頼をインポートしてもらってください。これらのファイルには、プログラム、トランザクション(YSCA)、およびプログラムユーザインタフェースが含まれています。これらをシステムにインポートした後、SAPデータベースからコードを抽出してOpenText SASTスキャン用のコードを作成できます。

インストールに関する注意事項

移送依頼のインポートエラー(`Install release does not match the current version`)が発生した場合には、移送依頼のインストールが失敗しました。サポートされているABAPバージョンについては、「ソフトウェア要件」を参照してください。

この問題を解決するには、次の手順を実行します。

1. 移送依頼のインポートを再実行します。

   ［Import Transport Request］ダイアログボックスが開きます。

2. ［**Options**］タブを選択します。
3. ［**Ignore Invalid Component Version**］チェックボックスをオンにます。
4. インポート手順を完了します。

それでも問題が解決しない場合や、異なるテーブル構造を持つSAPバージョンがシステムで実行されている場合には、OpenText SASTがABAPコードをスキャンできるように、独自の技術を使用してABAPファイル構造をエクスポートすることをお勧めします。

1.18.3. OpenText SASTのお気に入りリストへの追加

OpenText SASTのお気に入りリストへの追加はオプションですが、追加すると、OpenText SASTスキャンにすばやくアクセスしてスキャンを開始できます。次の手順では、日常業務でユーザメニューを使用すると仮定しています。別のメニューから作業を行う場合は、使用するメニューにお気に入りリンクを追加します。OpenText SASTのエントリを作成する前に、SAPサーバが実行中であり、WebベースのクライアントのSAP Easy Accessエリアにアクセスしていることを確認してください。

OpenText SASTをお気に入りリストに追加するには、次の手順を実行します。

1. ［**SAP Easy Access**］メニューから、トランザクションボックスに「`S000`」と入力します。

   ［**SAP Menu**］が開きます。

2. ［**Favorites**］フォルダを右クリックし、［**Insert transaction**］を選択します。

   ［**Manual entry of a transaction**］ダイアログボックスが開きます。

3. ［**トランザクションコード(Transaction Code)**］ボックスに「`YSCA`」と入力します。

4. 緑色のチェックマークアイコンをクリックします。

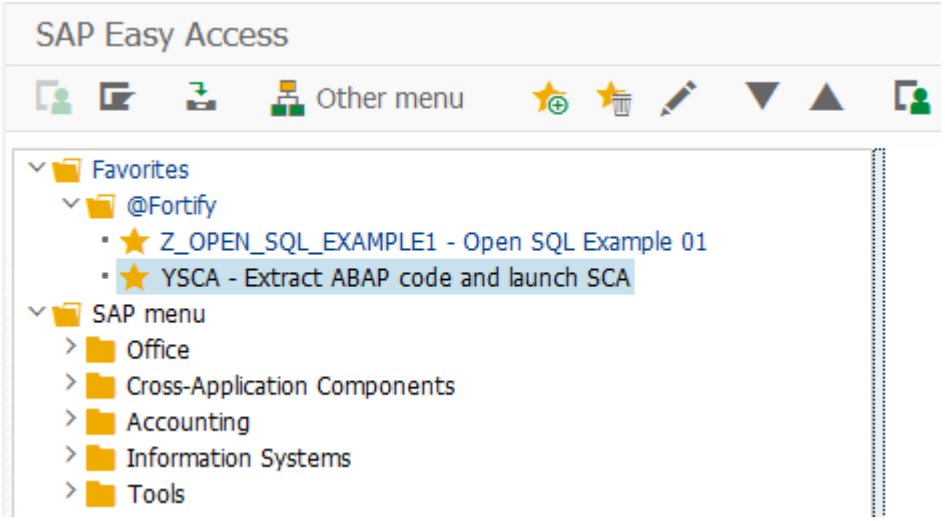
   ［お気に入り(Favorites)］リストに［ABAPコードを抽出してSCAを起動する(Extract ABAP code and launch SCA)］という項目が表示されます。

5. ［ABAPコードを抽出してSCAを起動する(Extract ABAP code and launch SCA)］リンクをクリックしてFortify ABAP Extractorを開始します。

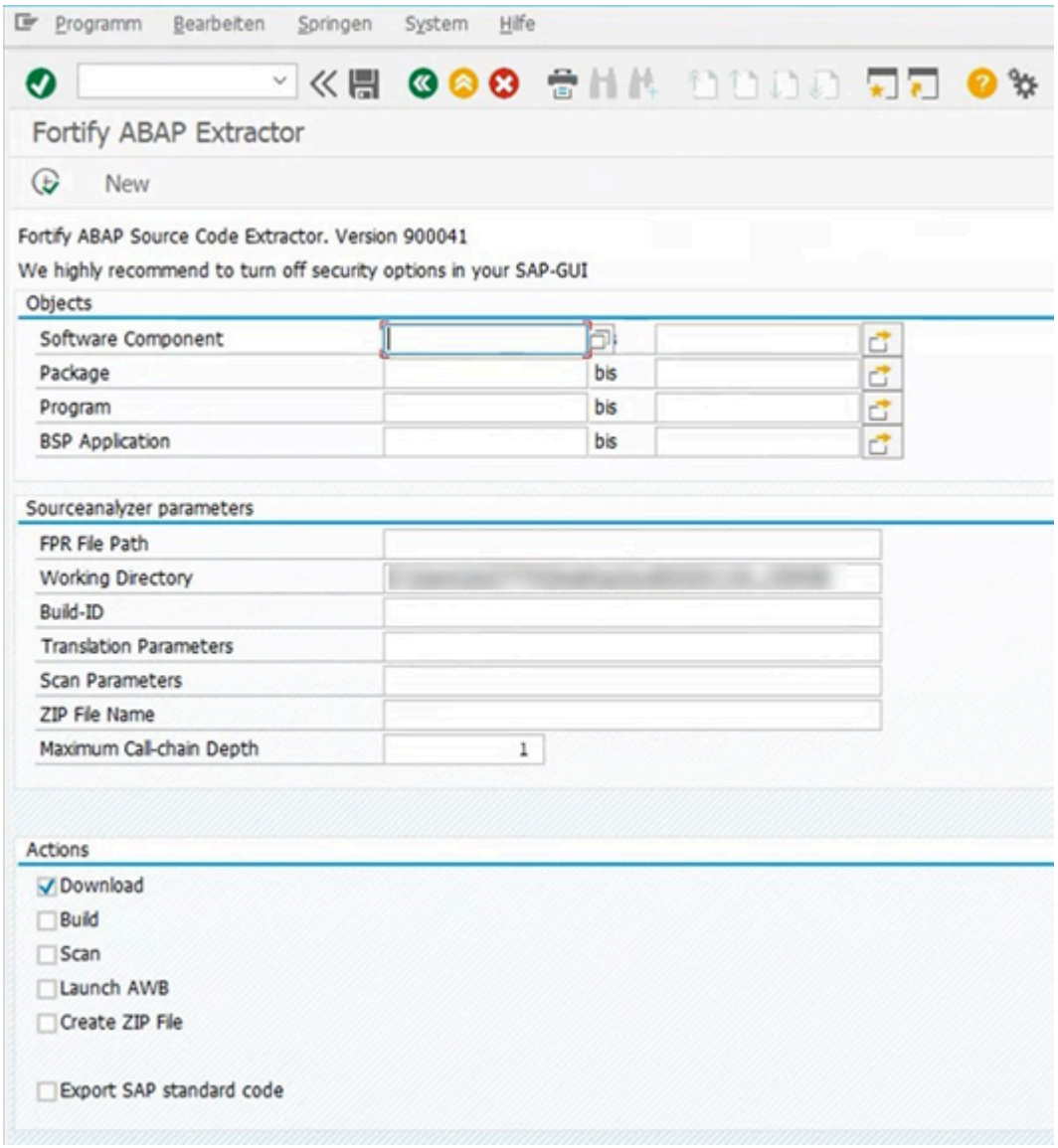
1.18.4. Fortify ABAP Extractorの実行

Fortify ABAP Extractorを実行するには、次の手順を実行します。

1. ［**お気に入り(Favorites)**］リンクまたはトランザクションコードからFortify ABAP Extractorを起動するか、手動でExtractorオブジェクトを起動します。

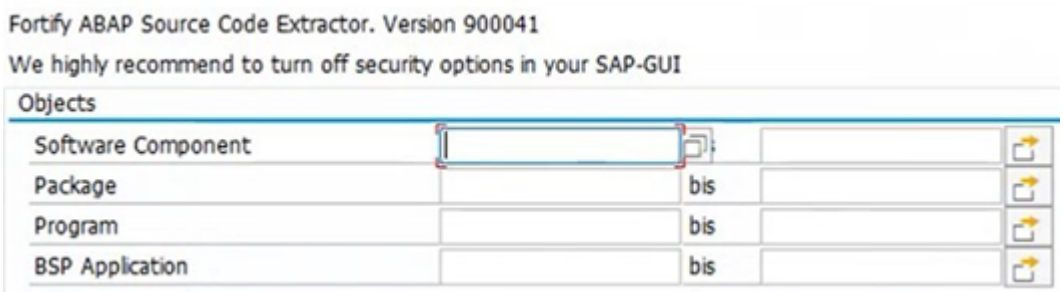


これにより、Fortify ABAP Extractorが開きます。

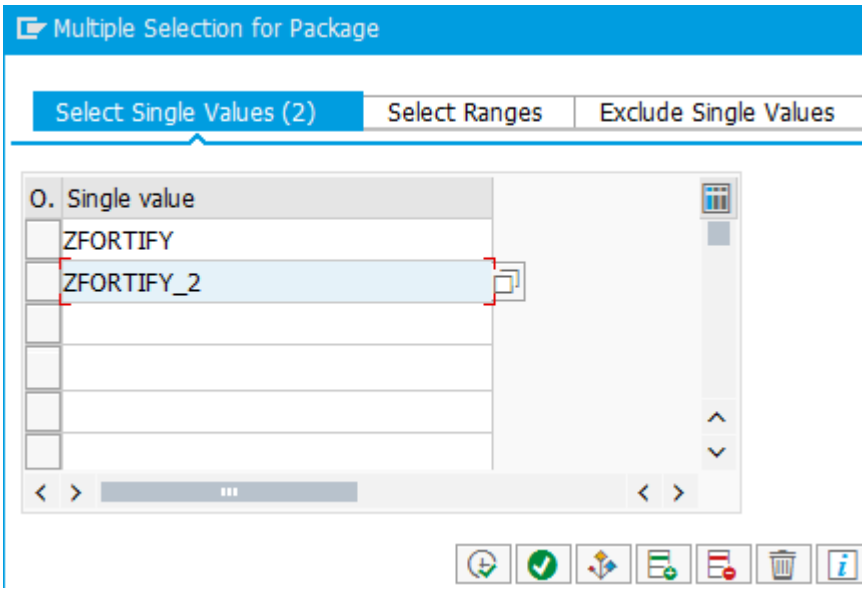


2. ダウンロードするコードを選択します。

   スキャンするソフトウェアコンポーネント、パッケージ、プログラム、またはBSPアプリケーションの範囲を開始名と終了名で指定します。



> **Note**
>
> 複数のオブジェクトまたは範囲を指定できます。



3. 次の表に示すOpenText SAST固有のパラメータを指定します。

フィールド	説明(Description)
FPR File Path	(オプション)スキャン結果ファイル(FPR)を保存するディレクトリを入力または選択します。FPRファイルの名前をパス名に含めてください。抽出プロセスを実行している同じコンピュータにダウンロードしたコードを自動的にスキャンするには、FPRファイルパスを指定する必要があります。
作業ディレクトリ(Working Directory)	抽出したソースコードを保存するディレクトリを入力または選択します。
Build-ID	(オプション)スキャンのビルドIDを入力します。OpenText SASTでは変換されたソースコードを識別するためにビルドIDを使用します。これはコードをスキャンするために必要です。抽出プロセスを実行している同じコンピュータにダウンロードしたコードを自動的に変換するには、ビルドIDを指定する必要があります。
変換パラメータ(Translation Parameters)	(オプション)追加のOpenText SASTコマンドライン変換オプションを入力します。抽出プロセスを実行している同じコンピュータにダウンロードしたコードを自動的に変換したり、変換オプションをカスタマイズしたりするには、変換オプションを指定する必要があります。

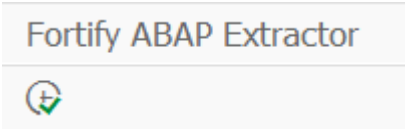
フィールド	説明(Description)
スキャンパラメータ(Scan Parameters)	(オプション)OpenText SASTコマンドラインスキャンオプションを入力します。抽出プロセスを実行している同じコンピュータにダウンロードしたコードを自動的にスキャンしたり、スキャンオプションをカスタマイズしたりするには、スキャンオプションを指定する必要があります。
ZIPファイル名(ZIP File Name)	(オプション)圧縮パッケージで出力する場合は、ZIPファイル名を入力します。
Maximum Call-chain Depth	グローバルSAP関数Fは、Fが明示的に選択されていない限り、または明示的に選択されたコードで始まる関数呼び出しのチェーンを介してFに到達可能で、そのチェーンの長さがこの数以下である場合を除き、ダウンロードされません。カスタマサポートの指示がない限り、2より大きい値を指定しないことを推奨します。

4. 次の表に示すアクション情報を指定します。

フィールド	説明(Description)
Download	SAPデータベースから抽出したソースコードをOpenText SASTでダウンロードするには、［ダウンロード(Download)］チェックボックスをオンにします。
ビルド(Build)	OpenText SASTでダウンロード済みのすべてのABAPコードを変換し、指定したビルドIDを使用してそのコードを保存するには、［ビルド(Build)］チェックボックスをオンにします。このアクションを実行するには、Fortify ABAP Extractorを実行しているコンピュータにOpenText SASTのインストール済みバージョンが必要です。多くの場合、OpenText SASTがインストールされているシステムにダウンロードしたソースコードを移動する方が簡単です。
スキャン(Scan)	指定したビルドIDのスキャンをOpenText SASTで実行するには、［スキャン(Scan)］チェックボックスをオンにします。このアクションでは、変換(ビルド)アクションが以前に実行されている必要があります。このアクションを実行するには、Fortify ABAP Extractorを実行しているコンピュータにOpenText SASTのインストール済みバージョンが必要です。多くの場合、ダウンロードしたソースコードを事前定義されたOpenText SASTコンピュータに移動する方が簡単です。
Launch AWB	Fortify Audit Workbenchを起動して指定したFPRファイルを開くには、［AWBの起動(Launch AWB)］チェックボックスをオンにします。

フィールド	説明(Description)
ZIPファイルの作成(Create ZIP File)	出力を圧縮するには、［ZIPファイルの作成(Create ZIP File)］チェックボックスをオンにします。また、ソースコードをSAPデータベースから抽出した後で、出力を手動で圧縮することもできます。
Export SAP standard code	カスタムコードに加えてSAP標準コードもエクスポートするには、［SAP標準コードのエクスポート(Export SAP standard code)］チェックボックスをオンにします。

5. ［実行(Execute)］をクリックします。



1.18.5. Fortify ABAP Extractorのアンインストール

ABAP Extractorをアンインストールするには、次の手順を実行します。

1. ABAP Workbenchで、Object Navigatorを開きます。
2. Y_FORTIFY_ABAPパッケージを選択します。
3. ［**Programs**］タブを展開します。
4. 次の要素を右クリックして、［**Delete**］を選択します。
    - プログラム: Y_FORTIFY_SCA

1.19. COBOLコードの分析

COBOL変換はWindowsシステムでのみ実行され、最新のCOBOL方言をサポートします。または、レガシーCOBOL変換を使用できます(「レガシーCOBOL変換の使用」を参照)。

COBOLコードの変換でサポートされている技術のリストについては、「サポートされる言語」を参照してください。OpenText SASTは、現時点ではCOBOLアプリケーションのカスタムルールをサポートしていません。

Note

OpenText SASTでCOBOLをスキャンするには、COBOLスキャン機能を明確に含むOpenText SASTライセンスファイルが必要です。必要なライセンスファイルの入手方法の詳細については、カスタマサポートにお問い合わせください。

このセクションでは、次のトピックについて説明します。

1.19.1. COBOLソースファイルとコピーブックファイルの変換準備

COBOLプログラムを分析する前に、OpenText SASTを実行するWindowsシステムに次のプログラムコンポーネントをコピーする必要があります。

- COBOLソースコード

  COBOLソースコードファイルには、拡張子 `.CBL` 、 `.cbl` 、 `.COB` 、または `.cob` を使用することを強く推奨しています。ソースコードファイルに拡張子がないか、拡張子が標準以外のものである場合は、「ファイル拡張子を持たないCOBOLソースファイルの変換」および「任意のファイル拡張子を持つCOBOLソースファイルの変換」の手順に従う必要があります。

- COBOLソースコードで使用しているすべてのコピーブックファイル

  これには、COBOLソースコードで参照するすべてのSQL INCLUDEファイル(SQL INCLUDEファイルは技術的にはコピーブックファイル)が含まれます。

  > **Important**
  >
  > コピーブックファイルには、拡張子 `.CPY` または `.cpy` を使用する必要があります。

  COBOLソースコードに次が含まれている場合:

  ```
  COPY FOO
  ```

  または

  ```
  EXEC SQL INCLUDE FOO END-EXEC
  ```

  `FOO` はCOBOLコピーブックの名前で、対応するコピーブックファイルは名前が `FOO.CPY` または `FOO.cpy` になります。

COBOLソースコードファイルを `sources` という名前のディレクトリに、コピーブックファイルを `copybooks` という名前のディレクトリに配置することを推奨しています。これらのディレクトリは同じレベルで作成してください。

1.19.2. COBOLのコマンドライン構文

1つのCOBOLソースコードファイルを変換するために使用される基本的な構文は次のとおりです。

```
sourceanalyzer -b <build_id><path>
```

変換されたCOBOLプログラムをスキャンして、分析結果をFPRファイルに保存するために使用される基本的な構文は次のとおりです。

```
sourceanalyzer -b <build_id> -scan -f <results>.fpr
```

参照情報

ファイルとディレクトリの指定

1.19.2.1. ファイル拡張子を持たないCOBOLソースファイルの変換

メインフレームからCOBOLソースファイル(コピーブックファイルではなく)を取得し、これに `.COB` や `.CBL` のファイル拡張子がない場合(COBOLファイル名としては一般的です)、変換コマンドラインに次の項目を含める必要があります。

```
-noextension-type COBOL
```

次のコマンドの例では、ファイル拡張子を持たないCOBOLソースコードを変換します。

```
sourceanalyzer -b MyProject -noextension-type COBOL -copydirs
copybooks sources
```

1.19.2.2. 任意のファイル拡張子を持つCOBOLソースファイルの変換

任意の拡張子(`.xyz`)を持つCOBOLソースファイルがある場合は、変換コマンドラインに次の項目を含める必要があります。

```
-Dcom.fortify.sca.fileextensions.xyz=COBOL
```

ファイルまたはディレクトリ指定子を使用する場合は、式 `*.xyz` も含める必要があります(ファイルとディレクトリの指定を参照)。

1.19.2.3. COBOLコマンドラインオプション

次の表では、COBOLコマンドラインオプションについて説明します。レガシーCOBOL変換を使用するには、「レガシーCOBOL変換コマンドラインオプション」を参照してください。

COBOLオプション	説明(Description)
`-copydirs <dirs>`	OpenText SASTでコピーブックファイルを検索する、1つ以上のディレクトリをセミコロン区切りで指定します。 **Note** このオプションでは、ワイルドカードは使用できません。 **同等のプロパティ名:** `com.fortify.sca.CobolCopyDirs`
`-dialect <dialect>`	COBOLの方言を指定します。*<dialect>*の有効な値は、`COBOL390`および`MICROFOCUS`です。dialect値では、大文字と小文字を区別しません。デフォルト値は`COBOL390`です。 **同等のプロパティ名:** com.fortify.sca.CobolDialect
`-checker-directives <directives>`	1つ以上のCOBOLチェッカディレクティブをセミコロン区切りで指定します。 **Note** このオプションは、OpenText™ Server Expressの上級ユーザ向けです。 **同等のプロパティ名:** `com.fortify.sca.CobolCheckerDirectives`

1.19.3. レガシーCOBOL変換の使用

次のいずれかの条件に当てはまる場合は、レガシーCOBOL変換を使用してください。

- Windows以外のオペレーティングシステムでOpenText SASTを実行している。

  サポートされているWindows以外のプラットフォームおよびアーキテクチャについては、「プラットフォームとアーキテクチャ」を参照してください。

- 使用しているCOBOL方言が、デフォルトのCOBOL変換でサポートされているものとは異なる(「COBOLコマンドラインオプション」の `-dialect` オプションを参照)。

「COBOLソースファイルとコピーブックファイルの変換準備」の説明に従ってCOBOLソースコードファイルとコピーブックファイルを準備し、「COBOLのコマンドライン構文」で説明されているコマンドライン構文を使用します。レガシーCOBOL変換では、ファイル拡張子の有無に関係なく、コピーブックファイルを受け入れることにご注意ください。コピーブックファイルにファイル拡張子がある場合は、`-copy-extensions` コマンドラインオプションを使用します(「レガシーCOBOL変換コマンドラインオプション」を参照)。

1.19.3.1. レガシーCOBOL変換コマンドラインオプション

次の表では、レガシーCOBOL変換のコマンドラインオプションについて説明します。

レガシーCOBOLオプション	説明(Description)
`-cobol-legacy`	レガシーCOBOL変換を使用したCOBOLコードの変換を指定します。このオプションは、レガシーCOBOL変換を有効にする場合に必要です。 **同等のプロパティ名:** `com.fortify.sca.CobolLegacy`
`-copydirs <dirs>`	OpenText SASTでコピーブックファイルを検索する、1つ以上のディレクトリをセミコロン区切りまたはコロン区切りで指定します。 **同等のプロパティ名:** `com.fortify.sca.CobolCopyDirs`
`-copy-extensions <ext>`	1つ以上のコピーブックファイル拡張子をセミコロン区切りまたはコロン区切りで指定します。 **同等のプロパティ名:** `com.fortify.sca.CobolCopyExtensions`

レガシーCOBOLオプション	説明(Description)
-fixed-format	すべてのコード行のカラム8～72の間のソースコードのみを検索するようOpenText SASTに指示する、固定形式のCOBOLを指定します。デフォルトはフリーフォーマットです。 IBM® Enterprise COBOLコードは通常、固定形式です。以下の場合は、-fixed-formatオプションが必要になる可能性があります。 • COBOL変換が無期限にハングする可能性がある • OpenText SASTによりCOBOL変換で多数の解析エラーが報告される **同等のプロパティ名:** com.fortify.sca.CobolFixedFormat

1.20. Rubyコードの分析

このセクションでは、次のトピックについて説明します。

1.20.1. Rubyコマンドライン構文

Rubyコードを変換する基本的なコマンドライン構文は次のとおりです。

```
sourceanalyzer –b <build_id> <file>
```

ここで、 *<file>* はスキャンするRubyファイルの名前です。複数のRubyファイルを含めるには、次の例に示すように、ファイルをスペースで区切ります。

```
sourceanalyzer –b <build_id> file1.rb file2.rb file3.rb
```

個々のRubyファイルをリストするだけでなく、アスタリスク(*)のワイルドカードを使用して、指定したディレクトリ内のすべてのRubyファイルを選択できます。たとえば、 src ディレクトリ内のすべてのRubyファイルを検索するには、次の sourceanalyzer コマンドを使用します。

```
sourceanalyzer –b <build_id> src/*.rb
```

Note

Rubyコードを変換する場合は、すべてのソースファイルを1回の呼び出しで一緒に指定してください。OpenText SASTでは、後続の呼び出し時にビルドIDで関連付けられたファイルリストに新しいファイルを追加する機能はサポートされていません。

1.20.1.1. Rubyコマンドラインオプション

次の表は、Rubyの変換オプションについて説明しています。

Rubyオプション	説明(Description)
`-ruby-path <dirs>`	Rubyライブラリを含むディレクトリへのパスを1つ以上指定します(ライブラリの追加を参照)。 **同等のプロパティ名:** `com.fortify.sca.RubyLibraryPaths`
`-rubygem-path <dirs>`	RubyGemsの場所へのパスを指定します(Gemパスの追加を参照)。 **同等のプロパティ名:** `com.fortify.sca.RubyGemPaths`

Rubyのプロパティ

1.20.2. ライブラリの追加

Rubyソースコードに特定のライブラリが必要な場合は、Rubyライブラリを `sourceanalyzer` コマンドに追加します。RubyのgemとともにインストールされるすべてのRubyライブラリを含めます。たとえば、 `utils.rb` ディレクトリ内に `/usr/share/ruby/myPersonalLibrary` ファイルがある場合は、次のオプションを `sourceanalyzer` コマンドに追加します。

```
-ruby-path /usr/share/ruby/myPersonalLibrary
```

複数のライブラリはセミコロン(Windows)またはコロン(Windows以外)で区切ります。次に、Windows以外のシステムでのオプションの例を示します。

```
-ruby-path /path/one:/path/two:/path/three
```

1.20.3. Gemパスの追加

Rubyのすべてのgemとその依存関係パスを追加するには、Rubyのすべてのgemをインポートします。Rubyのgemパスを取得するには、`gem env`コマンドを実行します。**GEM PATHS**の下で次のようなディレクトリを探します。

```
/home/myUser/gems/ruby-version
```

このディレクトリには、システムにインストールされているすべてのgemファイルのディレクトリが含まれる`gems`という名前の別のディレクトリがあります。この例では、コマンドラインで次のコマンドを使用します。

```
-rubygem-path /home/myUser/gems/ruby-version/gems
```

複数の`gems`ディレクトリがある場合は、次のようにセミコロン(Windows)またはコロン(Windows以外)で区切ります。

```
-rubygem-path /path/to/gems:/another/path/to/more/gems
```

Note

Windowsシステムの場合は、`gems`ディレクトリをセミコロンで区切ります。

1.21. その他の言語および設定の分析

このセクションでは、次のトピックについて説明します。

1.21.1. Solidityコードの分析

Solidityコードを変換およびスキャンするための基本的なコマンドライン構文は次のとおりです。

```
sourceanalyzer -b <build_id> <files> sourceanalyzer -b
<build_id> -scan -f <results>.fpr
```

依存関係のインポート

OpenText SAST変換では、相対パスと絶対パスで指定したファイルのインポートステートメントのみがサポートされます。ライブラリのインポートステートメントはサポートされません。

コンパイラのバージョンの管理

OpenText SASTは、pragmaステートメントを使用してコード内で参照されているコンパイラを、Solidityコンパイラリポジトリからダウンロードします。デフォルトでは、OpenText SASTはSolidityコンパイラを `${flight.workdir}/solidity` にダウンロードします。

ファイルにpragmaステートメントが含まれていない場合は、 `^0.8.0` の既定値が使用されます。分析で使用する別のデフォルトコンパイラバージョンを指定するには、コマンドラインに `flight.solidity.defaultCompilerVersion` プロパティを含めることができます。指定するバージョンは、Solidityコンパイラリポジトリに存在する必要があります。例:

```
sourceanalyzer -b MyProject ./ sourceanalyzer -b MyProject -scan
-Dflight.solidity.defaultCompilerVersion=0.8.16 -f MyResults.fpr
```

Solidityコンパイラをダウンロードするための接続にプロキシが必要な場合は、 `-Dhttps.proxyHost` および `-Dhttps.proxyPort` を使用してプロキシ情報を含めてください。例:

```
sourceanalyzer -b MyProject ./ sourceanalyzer -b MyProject -scan
-Dhttps.proxyHost=MyProxyHost -Dhttps.proxyPort=1234 -f
MyResults.fpr
```

`flight.solidity.defaultCompilerVersion` は、 `fortify-sca.properties` ファイルに追加できます。

参照情報

OpenText SASTのプロパティファイル

1.21.2. FlexおよびActionScriptの分析

ActionScriptを変換するための基本的なコマンドライン構文は次のとおりです。

```
sourceanalyzer -b <build_id> -flex-libraries <libs> <files>
```

ここで:

<libs> は、セミコロンで区切られた(Windows)またはコロンで区切られた(Windows以外)「リンク」するライブラリ名のリストで、 *<files>* は変換するファイルです。

1.21.2.1. FlexおよびActionScriptコマンドラインオプション

Flexファイルを変換するには、次のコマンドラインオプションを使用します。この情報は、各説明に記載されているproperties環境設定ファイル(`fortify-sca.properties`)でも指定できます。

FlexおよびActionScriptオプション	説明(Description)
-flex-sdk-root *<dir>*	有効なFlex SDKのルートの場所を指定します。このディレクトリには、flex-config.xmlファイルを含むフレームワークフォルダが含まれている必要があります。MXMLC実行可能ファイルを含むbinフォルダも含まれている必要があります。 **同等のプロパティ名:** com.fortify.sca.FlexSdkRoot
-flex-libraries *<libs>*	リンクするライブラリ名をセミコロン区切り(Windows)またはコロン区切り(Windows以外)のリストで指定します。ほとんどの場合、このリストにはflex.swc、framework.swc、playerglobal.swc (通常、Flex SDKルートのframeworks/libs/にある)が含まれています。 **Note** SWCファイルまたはSWCファイルをFlexライブラリとして指定できます(SWZは現在サポートされていません)。 **同等のプロパティ名:** com.fortify.sca.FlexLibraries

FlexおよびActionScriptオプション	説明(Description)
`-flex-source-roots <dirs>`	MXMLソースが保存されているルートディレクトリをセミコロン区切り(Windows)またはコロン区切り(Windows以外)のリストで指定します。通常、これらには `com` という名前のサブフォルダが含まれます。 たとえば、指定されたFlexソースルートが `foo/bar/src` である場合、`foo/bar/src/com/fortify/manager/util/Foo.mxml` は `com.fortify.manager.util.Foo` という名前のオブジェクト(パッケージ `Foo` 内の `com.fortify.manager.util` という名前のオブジェクト)に変換されます。 **同等のプロパティ名:** `com.fortify.sca.FlexSourceRoots`

Note

`-flex-sdk-root` および `–flex-source-roots` オプションは主にMXML変換用であり、純粋なActionScriptをスキャンする場合は省略可能です。`–flex-libraries` はすべてのActionScriptリンク済みライブラリを解決するために使用します。

OpenText SASTではMXMLファイルをActionScriptに変換し、ActionScriptパーサを介して実行します。生成されたActionScriptは簡単に分析でき、Flexランタイムモデルのように厳密な正解を示すものではありません。その結果、MXMLファイルで解析エラーが発生する可能性があります。たとえば、XML解析が失敗し、ActionScriptへの変換が失敗し、結果のActionScriptの解析も失敗する可能性があります。元のソースコードに対して明確な接続がないというエラーが発生した場合は、カスタマサポートまでお知らせください。

FlexおよびActionScriptのプロパティ

1.21.2.2. ActionScriptのコマンドライン例

次の例では、ActionScriptを変換するためのコマンドライン構文を示します。

例1

次の例は、1つのMXMLファイルと1つのMXMLライブラリ(`MyLib.swf`)のみを含む単純なアプリケーションの例です。

```
sourceanalyzer -b MyFlexApp -flex-libraries lib/MyLib.swf -flex-
sdk-root /home/myself/flex-sdk/ -flex-source-roots .
my/app/FlexApp.mxml
```

これにより、含めるライブラリの場所、Flex SDK、およびFlexソースルートの場所が識別されます。 `/my/app/FlexApp.mxml` にある1つのMXMLファイルは、 `my.app` という1つのActionScriptクラスとしてMXMLアプリケーションの変換になり、 `FlexApp` パッケージに入れられます。

例2

次の例は、ソースファイルが `src` ディレクトリに対して相対的であるアプリケーションの例です。1つのSWFライブラリ `MyLib.swf` と、Flex SDKのFlexライブラリとフレームワークライブラリを使用します。

`sourceanalyzer -b MyFlexProject -flex-sdk-root /home/myself/flex-sdk/ -flex-source-roots src/ -flex-libraries lib/MyLib.swf "src/**/*.mxml" "src/**/*.as"`

この例では、Flex SDKを見つけ、ファイル指定子を使用して、 `.as` ファイルと `.mxml` ファイルを `src` フォルダに含めます。この例では、 `.SWC` にある `–flex-sdk-root` ファイルを明示的に指定する必要はありません。ただし、この例では例示のために明示的に指定しています。OpenText SASTにより、指定したFlex SDKルート内にあるすべての `.SWC` ファイルが自動的に検索され、ActionScriptまたはMXMLファイルの変換に使用されるライブラリであると見なされます。

例3

この例では、Flex SDKルートとFlexライブラリがプロパティファイルで指定されています。これは、sourceanalyzerを実行するたびに情報を入力するのは時間がかかり、データが頻繁には変わらないためです。アプリケーションを2つのセクションに分割し、mainセクションフォルダとmodulesフォルダに保存します。各フォルダには、パスの起点になる `src` フォルダが含まれています。ファイル指定子には、両方の `.mxml` フォルダ内にあるすべての `.as` および `src` ファイルを選択するワイルドカードが含まれています。 `main/src/com/foo/util/Foo.mxml` 内の `MXML` ファイルは、たとえば次のようにソースル

ートを指定して、 `com.foo.util` パッケージ内の `Foo` という名前のActionScriptクラスとして変換されます。

```
sourceanalyzer -b MyFlexProject -flex-source-roots
main/src:modules/src "./main/src/**/*.mxml" "./main/src/**/*.as"
"./modules/src/**/*.mxml" "./modules/src/**/*.as"
```

1.21.2.3. 解決の警告の処理

変換中に生成されたすべての警告を表示するには、スキャンフェーズを開始する前に次のコマンドを入力します。

```
sourceanalyzer -b <build_id> -show-build-warnings
```

ActionScriptの警告

次のようなメッセージが表示される場合があります。

```
The ActionScript front end was unable to resolve the following
imports: a.b at y.as:2. foo.bar at somewhere.as:5. a.b at
foo.mxml:8.
```

このエラーは、OpenText SASTで必要なすべてのライブラリが見つからないときに発生します。OpenText SASTで分析を完了するために、(`-flex-libraries` オプションまたは `com.fortify.sca.FlexLibraries` プロパティを使用して)追加のSWCライブラリまたはSWC Flexライブラリを指定する必要がある場合があります。

1.21.3. ColdFusionコードの分析

CFMLページ内の未定義の変数を汚染されているものとして扱う場合は、`<sast_install_dir>/Core/config/fortify-sca.properties` の次の行をコメント解除します。

```
#com.fortify.sca.CfmlUndefinedVariablesAreTainted=true
```

これは、register-globalsスタイルの脆弱性に注意するようにDataflow Analyzerに指示します。ただし、このプロパティを有効にすると、インクルードページ内の変数がそれ以前に発生したインクルードページの汚染された値に初期化されることがDataflow Analyzerで判明した場合に支障があります。

1.21.3.1. ColdFusionコマンドライン構文

ColdFusionソースコードを変換するための基本的なコマンドライン構文は次のとおりです。

```
sourceanalyzer -b <build_id> -source-base-dir <dir> <files> |
<file_specifiers>
```

ここで:

- *<build_id>* プロジェクトのビルドIDを指定します
- *<dir>* Webアプリケーションのルートディレクトリを指定します
- *<files>* | *<file_specifiers>* はCFMLソースコードファイルを指定します

  <file_specifiers> の使用方法については、「ファイルの指定」を参照してください。

Note

OpenText SASTは、 -source-base-dir ディレクトリを起点として各CFMLソースファイルの相対パスを計算します。OpenText SASTでは、インスタンスIDを生成するときに、これらの相対パスを使用します。アプリケーションソースツリー全体を別のディレクトリに移動する場合、 -source-base-dir オプションに適切なパラメータを指定すると、OpenText SASTで生成されるインスタンスIDは同じままです。

1.21.3.2. ColdFusion (CFML)コマンドラインオプション

次の表では、CFMLオプションについて説明します。

ColdFusionのオプション	説明(Description)
`-source-base-dir <web_app_root_dir> <files> | <file_specifiers>`	Webアプリケーションのルートディレクトリ。 **同等のプロパティ名:** `com.fortify.sca.SourceBaseDir`

ColdFusion (CFML)のプロパティ

1.21.4. SQLの分析

Windows (および.NETプロジェクトの場合に限り、Linuxも)のOpenText SASTでは、 .sql 拡張子を持つファイルはPL/SQLではなくT-SQLであることを前提とします。Windowsで .sql 拡張子を持つPL/SQLファイルを使用する場合は、PL/SQLとして扱われるようにOpenText SASTを設定する必要があります。

PL/SQLを変換およびスキャンするための基本的な構文は次のとおりです。

```
sourceanalyzer -b <build_id> -sql-language PL/SQL <files>
sourceanalyzer -b <build_id> -sql-language PL/SQL -scan -f
<results>.fpr
```

または、 .sql 拡張子を持つファイルのデフォルトの動作を変更できます。 fortify-sca.properties ファイルで、 com.fortify.sca.fileextensions.sql プロパティを PLSQL に設定します。

T-SQLを変換およびスキャンするための基本的な構文は次のとおりです。

```
sourceanalyzer -b <build_id> -sql-language TSQL <files>
sourceanalyzer -b <build_id> -scan -f <results>.fpr
```

SQLのプロパティ

1.21.4.1. PL/SQLのコマンドライン例

次のコマンド例では、2つのPL/SQLファイルを変換してスキャンします。

```
sourceanalyzer -b MyProject -sql-language PL/SQL x.pks y.pks
sourceanalyzer -b MyProject -sql-language PL/SQL -scan -f
MyResults.fpr
```

次のコマンド例では、`sources`ディレクトリ内のすべてのPL/SQLファイルを変換してスキャンします。

```
sourceanalyzer -b MyProject -sql-language PL/SQL
"sources/**/*.pks" sourceanalyzer -b MyProject -sql-language
PL/SQL -scan -f MyResults.fpr
```

1.21.4.2. T-SQLのコマンドライン例

次の例では、2つのT-SQLファイルを変換します。

```
sourceanalyzer -b MyProject x.sql y.sql
```

次の例では、`sources`ディレクトリ内のすべてのT-SQLファイルを変換します。

```
sourceanalyzer -b MyProject "sources\**\*.sql"
```

Note

この例は、`fortify-sca.properties`内の`com.fortifv.sca.fileextensions.sql`プロパティが、プロパティのデフォルト値である`TSQL`に設定されていることを想定しています。

1.21.5. Scalaコードの分析

Scalaコードの変換に必要なものは次のとおりです。

- Akkaコンパイラプラグイン

  このプラグインは、Maven Central Repositoryからダウンロードできます。

- Akka(以前のLightbend)ライセンスファイル

  このライセンスファイルは、OpenText SASTのインストールに含まれ、`<sast_install_dir>/plugins/lightbend`ディレクトリにあります。

ライセンスの設定方法およびScalaコードの変換方法については、Akkaドキュメント『Fortify SCA for Scala』を参照してください。

Important

プロジェクトに、Scala以外のソースコードが含まれている場合は、分析フェーズを実行する前にScala Fortifyコンパイラプラグインを使用してScalaコードを変換してから、同じビルドIDでsourceanalyzerを使用してその他のソースコードを変換する必要があります。

1.21.6. コードとしてのインフラストラクチャ(IaC)の分析

OpenText SASTは、Azure Resource Manager (ARM)、Bicep、AWS CloudFormation、およびHCLテンプレートを理解します。

Note

HCLの分析サポートは、Terraformおよびサポートされるクラウドプロバイダのコードとしてのインフラストラクチャ(IaC)構成に固有のものです。

最適な結果を得るには、テンプレートファイルの展開が有効であることを確認してください。テンプレートには次が含まれていてはなりません。

- 静的であり、ローカルで検出可能な検証エラー(タイプエラーや未定義の変数または関数への参照など)。
- テンプレートの解釈中、ただしリソースが展開または変更される前に発生する、展開前エラー(無効な配列のインデックス付け操作など)。
- クラウドで発生する展開エラー(存在しないリソースの動的参照など)。

AWS CloudFormationのファイル名拡張子を `.json` 、 `.yaml` 、 `.template` 、または `.txt` にすることが推奨されています。OpenText SASTでは、他の言語やファイルタイプでそれらの拡張子が一般的に使用されていない場合にのみ、他の拡張子をサポートしています(`.java` や `.html` など)。

デフォルトで、OpenText SASTでは、HCL拡張子 `.hcl` および `.tf` を持つファイルを変換します。

ARM変換コマンドラインの例

ARMテンプレートを変換する場合:

```
sourceanalyzer -b MyProject ArmTemplate.json
```

ディレクトリ内のすべてのARMテンプレートを変換する場合:

```
sourceanalyzer -b MyProject "src/**/*.json"
```

Bicep変換コマンドラインの例

1つのBicepテンプレートを変換する場合:

```
sourceanalyzer -b MyProject BicepTemplate.bicep
```

ディレクトリ内のすべてのBicepテンプレートを変換する場合:

```
sourceanalyzer -b MyProject "src/**/*.bicep"
```

AWS CloudFormation変換コマンドラインの例

異なる拡張子を持つAWS CloudFormationテンプレートを変換する場合:

```
sourceanalyzer -b MyProject CFTemplateA.template
CFTemplateB.yaml CFTemplateC.json CFTemplateD.customext
```

`.template` 拡張子を持つディレクトリ内のすべてのAWS CloudFormationテンプレートを変換する場合:

```
sourceanalyzer -b MyProject "src/**/*.template"
```

`.json` または `.yaml` の拡張子を持つディレクトリ内のすべてのAWS CloudFormationテンプレートを変換する場合:

```
sourceanalyzer -b MyProject "src/**/*.json" "src/**/*.yaml"
```

HCL変換コマンドラインの例

異なる拡張子を持つ2つのHCLテンプレートを変換する場合:

```
sourceanalyzer -b MyProject HCLTemplateA.hcl HCLTemplateB.tf
```

ディレクトリ内のすべてのHCLテンプレートを変換する場合:

```
sourceanalyzer -b MyProject "src/**/*.tf" "src/**/*.hcl"
```

JSONの変換

YAMLの変換

1.21.7. JSONの分析

デフォルトで、OpenText SASTでは、JSON拡張子 `.json` を持つファイルをJSONとして変換します。次の例では、1つのJSONファイルを変換します。

```
sourceanalyzer -b MyProject x.json
```

次の例では、 `sources` ディレクトリ内のすべてのJSONファイルを変換します。

```
sourceanalyzer -b MyProject "sources/**/*.json"
```

1.21.8. YAMLの分析

デフォルトで、OpenText SASTでは、YAML拡張子 `.yaml` および `.yml` を持つファイルを変換します。次の例では、異なるファイル拡張子を持つ2つのYAMLファイルを変換します。

```
sourceanalyzer -b MyProject x.yaml y.yml
```

次の例では、 `sources` ディレクトリ内のすべてのYAMLファイルを変換します。

```
sourceanalyzer -b MyProject "sources/**/*.yaml"
"sources/**/*.yml"
```

1.21.9. Dockerfileの分析

デフォルトでは、OpenText SASTは、ファイル名が`Dockerfile*`、`dockerfile*`、`*.Dockerfile`、および`*.dockerfile`のいずれかの形式のファイルをDockerfileとして認識します。

Note

`com.fortify.sca.fileextensions`プロパティを使用して、Dockerfileを検出するために使用されるファイル名拡張子を変更できます。「変換と分析フェーズのプロパティ」を参照してください。

OpenText SASTでは、Dockerfile内のエスケープ文字としてバックスラッシュ(`\`)とバッククォート(`` ` ``)を受け付けます。Dockerfileでエスケープ文字が設定されていない場合、OpenText SASTではバックスラッシュがエスケープ文字と見なされます。

Dockerfileを含むディレクトリを変換する構文を次の例に示します。

```
sourceanalyzer -b <build_id> <dir>
```

Dockerfileの形式が正しくない場合、OpenText SASTはログファイルにエラーを書き込んでファイルを解析できないことを示し、そのDockerfileの分析をスキップします。ログに書き込まれるエラーの例を次に示します。

```
Unable to parse dockerfile ProjA.Dockerfile, error on Line 1:20:
mismatched input '\n' expecting {LINE_EXTEND, WHITESPACE} Unable
to parse config file
C:/Users/jsmith/MyProj/docker/dockerfile/ProjA.Dockerfile
```

1.21.10. ASP/VBScript仮想ルートの分析

OpenText SASTでは、ASP仮想ルートを処理できます。物理ディレクトリにマップするエイリアスとして仮想ディレクトリを使用するWebサーバの場合、OpenText SASTを使用するとエイリアスを使用できます。

たとえば、`Include`および`Library`という名前の仮想ディレクトリがあり、それぞれ物理ディレクトリ`C:\WebServer\CustomerOne\inc`および`C:\WebServer\CustomerTwo\Stuff`を参照しているとします。

次の例では、*virtual*インクルードを使用するアプリケーションのASP/VBScriptコードを示しています。

```
<!--#include virtual="Include/Task1/foo.inc"-->
```

この例で、前述のASPコードは次の物理的な場所にあるファイルを参照します。

```
C:\Webserver\CustomerOne\inc\Task1\foo.inc
```

この例では、実際のディレクトリが仮想ディレクトリ名`Include`に置き換わります。

仮想ルートへの対応

各仮想ディレクトリのマッピングをOpenText SASTに提供するには、次の例に示すように、OpenText SASTコマンドライン呼び出しで`com.fortify.sca.ASPVirtualRoots.name_of_virtual_directory`プロパティを設定する必要があります。

```
sourceanalyzer -Dcom.fortify.sca.ASPVirtualRoots.
<virtual_directory>=
<full_path_to_corresponding_physical_directory>
```

Note

Windowsでは、物理パスにスペースが含まれている場合は、プロパティ設定を引用符で囲む必要があります。
`sourceanalyzer "-Dcom.fortify.sca.ASPVirtualRoots.<virtual_directory>=<full_path_to_corresponding_physical_directory>"`

前のセクションの例で展開するには、次のプロパティ値をOpenText SASTに渡します。

```
-
Dcom.fortify.sca.ASPVirtualRoots.Include="C:\WebServer\CustomerO
ne\inc" -
Dcom.fortify.sca.ASPVirtualRoots.Library="C:\WebServer\CustomerT
wo\Stuff"
```

これにより、 Include は C:\WebServer\CustomerOne\inc に、 Library は C:\WebServer\CustomerTwo\Stuff にマップされます。

OpenText SASTで #include ディレクティブが出現する場合:

```
<!-- #include virtual="Include/Task1/foo.inc" -->
```

OpenText SASTでは、プロジェクトに Include という名前の物理ディレクトリが含まれているかどうかを特定します。このような物理ディレクトリがない場合、OpenText SASTはそのランタイムプロパティを調べて、 -Dcom.fortify.sca.ASPVirtualRoots.Include= "C:\WebServer\CustomerOne\inc" 設定を見つけます。その後、OpenText SASTでは、 C:\WebServer\CustomerOne\inc\Task1\foo.inc ファイルを探します。

または、 fortify-sca.properties にある fortify-sca.properties ファイルでこのプロパティを設定できます。次の例に示すように、物理ディレクトリのパス内のバックスラッシュ文字(\)をエスケープする必要があります。

```
com.fortify.sca.ASPVirtualRoots.Library=C:\\WebServer\\CustomerT
wo\\Stuff
com.fortify.sca.ASPVirtualRoots.Include=C:\\WebServer\\CustomerO
ne\\inc
```

Note

以前のバージョンのASPVirtualRootプロパティは引き続き有効です。
OpenText SASTコマンドラインで次のように使用できます。

-Dcom.fortifv.sca.ASPVirtualRoots=C:\WebServer\CustomerTwo\Stuff; C:\WebServer\CustomerOne\inc

これはOpenText SASTに対し、virtualインクルードディレクティブの解決時に、指定した順序でリストされているディレクトリを検索するように促します。

仮想ルートの使用例

次のファイルがあります。

```
C:\files\foo\bar.asp
```

このファイルを指定するには、次のインクルードを使用します。

```
<!-- #include virtual="/foo/bar.asp">
```

次に、`sourceanalyzer`コマンドで仮想ルートを次のように設定します。

```
-Dcom.fortify.sca.ASPVirtualRoots=C:\files\foo
```

これにより、仮想ルートの前から`/foo`がストライプされます。`foo`を`com.fortify.sca.ASPVirtualRoots`プロパティで指定しない場合、OpenText SASTでは`C:\files\bar.asp`を探して失敗します。

仮想ルートを指定するシーケンスは次のとおりです。

1. ソースのパスの最初の部分を削除します。
2. パスの最初の部分を、コマンドラインで指定した仮想ルートに置き換えます。

1.21.11. Classic ASPのコマンドライン例

VBScriptで記述された、`MyASP.asp`という名前の単一ファイルのClassic ASPを変換するには、次のように入力します。

```
sourceanalyzer -b mybuild "MyASP.asp"
```

1.21.12. VBScriptのコマンドライン例

`myApp.vb` という名前のVBScriptファイルを変換するには、次のコマンドを入力します。

```
sourceanalyzer -b mybuild "myApp.vb"
```

1.22. ライブラリコードの分析

ライブラリコードは、他のアプリケーションに統合するように設計された再利用可能なソフトウェアコンポーネントまたはモジュールを指します。特定のプログラムのビジネスロジックとエントリポイントを含むアプリケーションコードとは異なり、通常、ライブラリコードは次の通りになります。

- 複数のプロジェクトにわたって汎用的で再利用可能
- メインエントリポイント(main()メソッドなど)がない
- 他のアプリケーションが使用する機能(ユーティリティクラス、フレームワーク、SDKなど)を提供します。

ライブラリコードは他のアプリケーションコードから呼び出されることを想定しているため、通常は、ユーザが制御できるデータ自体のインタフェースは提供されず、SAST技術が通常見つけ出せる結果は最小限に抑えられます。

ライブラリコードとアプリケーションコードの比較

機能	アプリケーションコード	ライブラリコード
エントリポイント	通常は「あり」	なし
目的	ビジネスロジックの実装	再利用可能な機能の提供
使用	スタンドアロンまたは展開	他のアプリに組み込まれている
分析の焦点	プログラムの完全な動作	APIの開示と使用パターン

ライブラリコードの効果的な分析

ライブラリコードを効果的にスキャンするには、コードをライブラリとして扱うOpenText SASTを設定する必要があります。

言語に合わせた通常の方法でコードを変換します。適切な言語の分析の詳細については、このユーザガイドの該当するセクションを参照してください。

スキャンの準備ができたら、スキャンステップ中に次のプロパティを設定します。

```
com.fortify.sca.rules.IsLibrary=true
```

このプロパティを有効にすると、分析エンジンは外部アプリケーションがライブラリコードを呼び出す呼び出しを模倣して、より詳細な分析を提供します。

その他の使用事例

ライブラリに加えて、アプリケーションコードをライブラリコードに似たように見せる宣言型エンドポイントフレームワークも多く存在します。

ご使用のWeb APIが現在カバーされていないフレームワークを使用している場合(「サポートされている技術」を参照)、このプロパティを有効にすると、フレームワークのカバレッジが模倣されることもありますが、誤ったフローがさらに生じることもあります。

Note
この機能は現在Javaコードでのみサポートされています。

Caution
このプロパティを有効にすると、外部のコードがアプリケーションに呼び込むのを模倣し、攻撃対象面が大幅に増加して、問題が目立って増え、より多くのリソースを使用する可能性があります。指定されたユースケースや指示がない限り、通常はこれをアプリケーションコードで有効化すべきではありません。また、Spring BootやJAX-RSなど、すでにカバーされている多数の宣言エンドポイントフレームワークをサポートするために、このプロパティを有効にする必要はありません。

1.23. シークレットのスキャン

OpenText SASTスキャンは一連のアナライザで構成されており、そのうちの1つはあらゆるファイル形式にわたる情報を一般的に検出できます。これにより、OpenText SASTは、シークレットなどのプレーンビューに隠された情報や、不可視制御文字を含む攻撃の使用など、プログラミング言語にかかわらず脆弱な弱点を見つけることができます。

この設定方法の詳細については、「正規表現の分析」を参照してください。

1.23.1. 正規表現分析

正規表現(regex)分析では、正規表現ルールを使用して、ファイルコンテンツとファイル名の両方の脆弱性を検出できます。この分析では、プロジェクトファイル内の脆弱なシークレット(パスワード、キー、資格情報など)を検出できます。

Important

Regex分析は言語に依存しないため、OpenText SASTが公式にはサポートしていないファイルタイプの脆弱性を検出する可能性があります。

正規表現分析では、変換フェーズに含まれるすべてのファイルパスとパスパターンが再帰的に検査されます。見つかったすべてのファイルは、特に除外されていない限り分析されます。正規表現分析に含まれているファイルを管理するには、次のオプションを使用します。

- 変換フェーズで `-exclude` オプションを使用してファイルまたはディレクトリを除外します。

  このオプションの詳細については、変換オプションを参照してください。

- デフォルトでは、正規表現分析から検出可能なバイナリファイルがすべて除外されます。分析にバイナリファイルを含めるには、次のプロパティを `fortify-sca.properties` ファイルに追加します(または、 `-D` オプションを使用して、このプロパティをコマンドラインに含めます)。

  ```
  com.fortify.sca.regex.ExcludeBinaries = false
  ```

- デフォルトでは、スキャン時間が許容可能になるように、10 MBを超えるファイルが正規表現分析から除外されます。次のプロパティを使用して、最大ファイルサイズ(メガバイト単位)を変更できます。

  ```
  com.fortify.sca.regex.MaxSize = <max_file_size_mb>
  ```

正規表現分析はデフォルトで有効になっています。正規表現分析を無効にするには、次のプロパティを `fortify-sca.properties` ファイルに追加するか、コマンドラインに含める必要があります。

```
com.fortify.sca.regex.Enable = false
```

正規表現分析のプロパティ

1.24. 結果の最適化

このセクションでは、OpenText SASTで多様なコードベースを分析する際に結果を最適化するためのガイドラインとヒントについて説明します。

1.24.1. 分析へのスキャンポリシーの適用

分析(スキャン)フェーズでは、最も深刻な脆弱性を特定してコードを迅速に修復できるようにするために、スキャンポリシーを指定できます。次の表で、提供されている3つのスキャンポリシーについて説明します。

ポリシー名	説明(Description)
security	これはデフォルトのスキャンポリシーであり、コード品質に関連する問題、一般的に信頼されるソースからのデータフロー、および通常はノイズと見なされる問題を分析結果から除外します。セキュリティ問題のコード修正に重点を置く場合はこのポリシーを使用します。
devops	このスキャンポリシーは、ノイズと見なされる可能性が高い追加の問題を除外し、優先度の低い問題を減らすことで、セキュリティポリシーを拡張します。このスキャンポリシーは、スキャン速度を優先して、開発者が結果を(中間監査なしで)直接レビューする場合に使用します。このスキャンポリシーを適用した後に残る問題は、修正が必要な重大なセキュリティ問題である可能性があります。 **Note** ローカルのsecurityスキャンポリシーに加えたカスタマイズが、このdevopsスキャンポリシーに自動的に組み込まれることはありません。
classic	このスキャンポリシーでは、問題は除外されません。このスキャンポリシーを使ってすべての問題を確認できますし、隠れた問題を見つけやすくするためにプロジェクトテンプレートで問題をフィルター処理する方法もおすすめです。

分析用のスキャンポリシーを指定するには、次の例に示すように、分析フェーズに `-scan-policy` (または `-sc`)オプションを含める必要があります。

```
sourceanalyzer -b MyProject -scan -scan-policy devops -f
MyResults.fpr
```

または、 com.fortify.sca.ScanPolicy ファイルの fortify-sca.properties プロパティを使用してスキャンポリシーを指定することもできます。例:

```
com.fortify.sca.ScanPolicy=devops
```

Note

分析用のスキャンポリシー設定に加えて、フィルタファイルを適用できます(「フィルタファイルによる問題の除外」を参照)。この場合、OpenText SASTによってスキャンポリシーとフィルタファイルの両方が分析に適用されます。

カスタムスキャンポリシーの作成

スキャンポリシーファイルは、 *<sast_install_dir>*/Core/config/scales ディレクトリにあります。スキャンポリシーごとに1つのファイルがあります。これらのポリシーファイルの設定を変更してスキャンポリシーをカスタマイズしたり、独自のスキャンポリシーファイルを作成することができます。スキャンポリシーファイルに使用される構文については、「フィルタファイルによる問題の除外」を参照してください。

カスタムスキャンポリシーファイルを作成するには:

1. *<sast_install_dir>*/Core/config/scales/ に移動します。
2. テキストエディタを開き、 scan-policy-*<name>*.txt という名前のファイルを作成します。ここで、 *<name>*はカスタムスキャンポリシーの名前です。
3. フィルタを scan-policy-*<name>*.txt ファイルに追加して保存します。
4. 分析にカスタムスキャンポリシーを使用するには、次の例に示すようにコマンドを入力します。この例では、スキャンポリシーファイル名は scan-policy-myscanpolicy.txt です。

   ```
   sourceanalyzer -b MyProject -scan -scan-policy myscanpolicy
   -f MyResults.fpr
   ```

   または、 fortify-sca.properties ファイルにカスタムスキャンポリシーを指定できます。

参照情報

変換と分析フェーズのプロパティ

1.24.2. フィルタファイルによる問題の除外

`sourceanalyzer` コマンドを実行すると、特定の脆弱性インスタンス、ルール、および脆弱性カテゴリをフィルタ処理するファイルを作成できます。 `-filter` 分析オプションを使用してファイルを指定します。

フィルタファイルは、任意のテキストエディタで作成できるテキストファイルです。このファイルで必要ないフィルタ項目のみを指定します。

Note

このセクションで説明するフィルタタイプは、フィルタファイルとスキャンポリシーファイルの両方に適用されます(「分析へのスキャンポリシーの適用」を参照)。

次の表に、使用可能なフィルタタイプとそれぞれの例を示します。

フィルタタイプ	メモ	例

Category	カテゴリのみを指定すると、すべてのサブカテゴリがフィルタで除外されます
	
	Note
	OpenText SASTでは、分析が実行される前の初期化フェーズでカテゴリフィルタが適用されます。
		`Poor Error Handling`
		
		`J2EE Bad Practices: Leftover Debug Code`

インスタンスID	特定の問題のインスタンスID **Note** OpenText SASTでは、分析フェーズの後にインスタンスIDフィルタが適用されます。	`6291C6A33303ED270C269917AA8A1005`

フィルタタイプ	メモ	例
ルールID	特定の問題のレポートを生成するルールID **Note** OpenText SASTでは、分析が実行される前の初期化フェーズでルールIDフィルタが適用されます。	`823FE039-A7FE-4AAD-B976-9EC53FFE4A59`

優先度[1]	優先度の値は、昇順で、`low`、`medium`、`high`、`critical`です。	`priority <= low`
		
		`priority < medium`

テイントフラグ	テイントフラグ式を括弧で囲みます。式を指定するには、`&&`、`||`、および`!`論理演算子を使用します。テイントフラグの一覧については、『*OpenText™ Static Application Security Testingカスタムルールガイド*』を参照してください。	`(SYSTEMINFO || EXCEPTIONINFO)`
		
		`(WEB || (DATABASE && PRIVATE))`
		
		`(NETWORK && !XSS)`

影響[1]		`impact < 0.5`
見込み[1]		`likelihood <= 1.5`

フィルタタイプ	メモ	例
信頼度[1]		`confidence < 1.8`
可能性[1]		`probability <= 1.2`
精度[1]		`accuracy <= 1.0`

[1]優先度フィルタとメタデータフィルタには、「未満」(`<`)または「以下」(`<=`)を使用します。

複合フィルタ

異なる行にフィルタを指定すると、OpenText SASTは各フィルタ行を一度に1行ずつ適用します。また、これらの演算子を1行に結合し、ブール論理演算子(`&&` 、 `||` 、 `!`)と中カッコ `{` `}` を使用して式をグループ化すると、より高度なフィルタを作成できます。

たとえば、 `Cross-Site Scripting` の問題をフィルタで除外したい場合に、問題に `4.0` 未満の `confidence` があり、テイントフラグに `DATABASE` または `LDAP` が含まれているとします。

次のフィルタを使用できます:

```
{ Cross-Site Scripting && confidence < 4.0 } && ( DATABASE ||
LDAP )
```

複合フィルタの一部が、事後スキャンのみを実行できるフィルタタイプである場合、通常は事前スキャンをフィルタする項目がある場合でも、事後スキャンが実行されます。

Note
中括弧に関係なく、テイントフラグフィルタは丸括弧で囲む必要があります。

参照情報

フィルタファイルの例

1.24.2.1. フィルタファイルの例

たとえば、次の出力は EightBall.java サンプルのスキャンによる出力です。このサンプルプロジェクトは、 OpenText_SAST_Fortify_Samples_*<version>*.zip ディレクトリ内の <code>Fortify_SCA_Samples_<version>.zip</code>アーカイブに含まれています。

次のコマンドは、分析結果を生成するために実行されます。

```
sourceanalyzer -b eightball EightBall.java sourceanalyzer -b
eightball -scan
```

次の結果は、検出された5つの問題を示しています。

```
[F7A138CDE5235351F6A4405BA4AD7C53 : low : Unchecked Return Value
: semantic ] EightBall.java(12) : Reader.read()
[6291C6A33303ED270C269917AA8A1005 : high : Path Manipulation :
dataflow ] EightBall.java(12) : ->new FileReader(0)
EightBall.java(8) : <=> (filename) EightBall.java(8) : <-
>Integer.parseInt(0->return) EightBall.java(6) : <=> (filename)
EightBall.java(4) : ->EightBall.main(0)
[176CC0B182267DD538992E87EF41815F : critical : Path Manipulation
: dataflow ] EightBall.java(12) : ->new FileReader(0)
EightBall.java(6) : <=> (filename) EightBall.java(4) : -
>EightBall.main(0) [E4B3ACF92911ED6D98AAC15876739EC7 : high :
Unreleased Resource : Streams : controlflow ] EightBall.java(12)
: start -> loaded : new FileReader(...) EightBall.java(14) :
loaded -> end_of_scope : end scope : Resource leaked
EightBall.java(12) : start -> loaded : new FileReader(...)
EightBall.java(12) : java.io.IOException thrown
EightBall.java(12) : loaded -> loaded : throw EightBall.java(12)
: loaded -> end_of_scope : end scope : Resource leaked :
java.io.IOException thrown [BB9F74FFA0FF75C9921D0093A0665BEB :
low : J2EE Bad Practices : Leftover Debug Code : structural ]
EightBall.java(4)
```

次の処理を実行するフィルタファイルの例を次に示します。

- J2EE Bad Practiceカテゴリに関連する結果をすべて削除する
- インスタンスIDに基づいてパス操作を削除する

- 特定のルールIDから生成されたデータフローの問題を削除する

```
#This is a category to filter from scan output
J2EE Bad Practices #This is an instance ID of a specific issue
to be filtered #from scan output
6291C6A33303ED270C269917AA8A1005 #This is a specific Rule ID
that leads to the reporting of a #specific issue in the scan
output: in this case the #dataflow sink for a Path Manipulation
issue.
823FE039-A7FE-4AAD-B976-9EC53FFE4A59
```

フィルタ処理された出力をテストするには、このテキストをコピーし、`test_filter.txt`という名前でファイルに貼り付けます。

`test_filter.txt`ファイルにフィルタ処理を適用するには、次のコマンドを実行します。

```
sourceanalyzer -b eightball -scan -filter test_filter.txt
```

フィルタ処理された分析では、次の結果が生成されます。

```
[176CC0B182267DD538992E87EF41815F : critical : Path Manipulation
: dataflow ] EightBall.java(12) : ->new FileReader(0)
EightBall.java(6) : <=> (filename) EightBall.java(4) : -
>EightBall.main(0) [E4B3ACF92911ED6D98AAC15876739EC7 : high :
Unreleased Resource : Streams : controlflow ] EightBall.java(12)
: start -> loaded : new FileReader(...) EightBall.java(14) :
loaded -> end_of_scope : end scope : Resource leaked
EightBall.java(12) : start -> loaded : new FileReader(...)
EightBall.java(12) : java.io.IOException thrown
EightBall.java(12) : loaded -> loaded : throw EightBall.java(12)
: loaded -> end_of_scope : end scope : Resource leaked :
java.io.IOException thrown
```

1.24.3. フィルタセットを使用した問題の除外

Fortify Audit Workbenchで作成した問題テンプレートでフィルタセットを使用して、分析結果から問題をフィルタできます。分析フェーズ中に問題を非表示にするフィルタセットを適用すると、OpenText SASTはその非表示の問題をFPRに書き込みません。これを行うには、Fortify Audit Workbenchを使用してフィルタセットを作成し、そのフィルタセットと、そのフィルタセットが含まれている問題テンプレートを使用してOpenText SASTスキャンを実行します。Fortify Audit Workbenchでフィルタとフィルタセットを作成する方法について詳しくは、『*OpenText™ Fortify Audit Workbenchユーザガイド*』を参照してください。

次の例では、FPRから問題を削除するために、問題テンプレート内でフィルタを作成して使用する方法の基本的な手順について説明します。

1. OWASP Top 10 2021を使用し、この標準のカテゴリに分類された問題のみを表示するとします。Fortify Audit Workbenchで、`OWASP_Filter`という名前の新しいフィルタセットを作成します。
2. Fortify Audit Workbenchで、`OWASP_Filter`フィルタセット内に表示フィルタを作成します。

   ```
   If [OWASP Top 10 2021] does not contain A Then hide issue
   ```

   このフィルタは、問題を検索して、名前に「A」を含むOWASP Top 10 2021のカテゴリに問題がマップされない場合は、それを非表示にします。OWASP Top 10 2021のカテゴリはすべて「A」(A01、A02、...、A10)で始まるため、文字「A」を含まないカテゴリはOWASP Top 10 2021に存在しません。このフィルタによって、問題はFortify Audit Workbenchのビューに表示されなくなりますが、FPRには残ったままになります。
3. Fortify Audit Workbenchで、問題テンプレートを`IssueTemplate.xml`という名前のファイルにエクスポートします。
4. OpenText SASTを使用し、次のコマンドを指定して、分析フェーズでフィルタセットを指定します。

   ```
   sourceanalyzer -b MyProject -scan -project-template
   IssueTemplate.xml -Dcom.fortify.sca.FilterSet=OWASP_Filter -
   f MyFilteredResults.fpr
   ```

フィルタセットで問題をフィルタすると、FPRのサイズは削減されますが、通常、スキャン時間は短縮されません。OpenText SASTは、問題を計算してFPRファイルに書き込むかどうかを判断した後で、フィルタセットを調べます。フィルタセット内のフィルタによって、OpenText SASTがロードするルールタイプが決まります。

1.24.4. FortifyRemoveコメントを使用したフィルタ

リンター、コンパイラ、静的解析ツールがIDEに直接組み込まれているように、開発者はこれらのツールの結果をコードから直接制御することに慣れています。同様に、OpenText SASTによって引き起こされる問題を管理するために、インラインコメントを必要に応じて使うこともできます。開発者は、問題を引き起こすルールID、または検索結果のカテゴリを `FortifyRemove()` に指定して、問題が報告されるのを防ぐことができます。

コメント付きで問題が削除されると、OpenText SASTは削除された問題をその場所やカテゴリを含めてログに記録します。

Note

この機能は、JavaおよびC#コードでデフォルトで利用可能かつ有効にされています。この機能は、 `com.fortifv.sca.rules.EnableRuleComments=false` を設定することで `fortify-rules.properties` で無効にできます。詳細については、「fortify-rules.properties」を参照してください。

基本的なコメント

たとえば、次のような `Java Hello World` アプリケーションがあるとします。

```
public class MyClass { public static void main(String[] args) {
System.out.println("Hello World"); } }
```

*J2EE Bad Practices: Leftover Debug Code*として*メイン関数*でトリガされる、`625EEE1F-464F-42DC-85D6-269A637EF747` のIDを持つルールがある場合を考えます。

開発者が同意せず、この問題を今後表示したくない場合は、次のいずれかの設定により、問題が表示されなくなります。

```
public class MyClass { // FortifyRemove(ID="625EEE1F-464F-42DC-
85D6-269A637EF747") public static void main(String[] args) {
System.out.println("Hello World"); } }
```

または

```
public class MyClass { // FortifyRemove(Category="J2EE Bad
Practices: Leftover Debug Code") public static void
main(String[] args) { System.out.println("Hello World"); } }
```

注: 文字列引数には「"」または「'」のいずれかを使用できます。

ワイルドカード

ワイルドカード(`*`)を使用して、カテゴリを展開し、複数のサブカテゴリまたは複数の一致するカテゴリをカバーできます。
例:

```
// FortifyRemove(Category="Cross-Site Scripting: *")
```

クロスサイトスクリプティングに関する問題のバリアントはすべて削除されます。
一方:

```
// FortifyRemove(Category="Cross-Site *")
```

クロスサイトスクリプティングに関する問題のすべてのバリアントと、「クロスサイト」で始まるカテゴリ(たとえば「クロスサイトリクエスト偽造」)も削除します。

複数の条件

ワイルドカードを使用する以外に、文字列の配列を取る `Categories` プロパティまたは `IDs` プロパティを使用して、それぞれ、複数のカテゴリまたは複数のルールIDを指定できます。
例

```
// FortifyRemove(Categories=["Cross-Site Scripting: Reflected",
"Cross-Site Scripting: Persistent"])
```

`Cross-Site Scripting: Reflected` または `Cross-Site Scripting: Persistent` の問題が次の行に表示されるのを防ぎます。

```
// FortifyRemove(IDs=["A", "B", "C", "D"]
```

`A` 、 `B` 、 `C` 、または `D` のルールが次の行でトリガされるのを防ぎます。
また、複数の条件をセミコロン(`;`)で区切って指定することもできます。
例:

```
// FortifyRemove(Category="SQL Injection"; ID="ABCD-1234")
```

これにより、`SQL Injection`の問題が次の行で発生しないようにし、ルールID`ABCD-1234`からトリガされる問題も防げます。

正当な理由の追加

問題は、`FortifyRemove`のコメント付きで、削除済みとして記録されます。削除情報と共にログに記録される文字列を受け入れる`justification`プロパティを指定することで、なぜ問題が削除されるのかを詳しく説明できます。
例:

```
// FortifyRemove(Category="Cross-Site Scripting: *";
Justification="We remove XSS here because we're using custom
framework XYZ that automatically protects against the attack")
```

1.24.5. Fortify Java注釈

OpenTextでは、2つのバージョンのFortify Java注釈ライブラリが提供されています。

- 保持ポリシーがCLASS (`FortifyAnnotations-CLASS.jar`)に設定された注釈。

  このバージョンのライブラリでは、コンパイル時にFortify Java注釈がバイトコードに反映されます。

- 保持ポリシーがSOURCE (`FortifyAnnotations-SOURCE.jar`)に設定された注釈。

  このバージョンのライブラリでは、Fortify Java注釈が使用されるコードがコンパイルされた後にバイトコードに反映されません。

OpenText Application Security Software製品を使ってアプリケーションのバイトコードを分析する場合(たとえばOpenText™ Core Application Security評価を使う場合)、注釈保持ポリシーがCLASSに設定されたバージョンを使用してください。OpenText Application Security Software製品を使ってアプリケーションのソースコードを分析する場合は、いずれかのバージョンのライブラリを使用できます。ただし、保持ポリシーがSOURCEに設定されているライブラリを使用することが強く推奨されています。

> **Important**
>
> コード内の潜在的なセキュリティ上の問題に関する情報が漏洩する可能性があるため、Fortify Java注釈を運用コードに残すのはセキュリティ上のリスクになります。保持ポリシーがCLASSに設定されている注釈は内部の分析にのみ使用し、アプリケーションの運用ビルドでは決して使用しないことが推奨されています。

このセクションでは、使用可能な注釈の概要を示します。サンプルアプリケーションは、`OpenText_SAST_Fortify_Samples_<version>.zip` ディレクトリ内の <code>Fortify_SCA_Samples_<version>.zip</code>アーカイブに含まれています。ディレクトリに含まれている `README.txt` ファイルには、サンプルアプリケーション、アプリケーションから発生する可能性がある問題、およびこれらの問題をFortify Java注釈を使用して解決する方法が記述されています。

Fortify Javaの注釈には、次の2つの制限があります。

- 各注釈は1つの入力または1つの出力のみ(あるいはその両方)を指定できます。
- 同じターゲットには各タイプの注釈を1つのみ適用できます。

OpenTextでは、次の主要な3つのタイプの注釈が提供されています。

- データフローの注釈

- フィールドと変数の注釈
- その他の注釈

独自のカスタム注釈がサポートされるルールを作成することもできます。詳細については、カスタマサポートにお問い合わせください。

1.24.5.1. データフローの注釈

データフローの注釈には、データフロールールと同様に、ソース、シンク、パススルー、検証の4種類があります。すべてがメソッドに適用され、パラメータ名または文字列 `this` および `return` によって入力、出力、またはその両方を指定します。また、データフローのソースおよびシンク注釈を関数の引数に適用することもできます。

ソース注釈

注釈パラメータで使用できる値は、 `this` 、 `return` 、または関数パラメータ名です。たとえば、ターゲットメソッドの出力に汚染を割り当てることができます。

```
@FortifyDatabaseSource("return") String []
loadUserProfile(String userID) { ... }
```

たとえば、ターゲットメソッドの引数に汚染を割り当てることができます。

```
void retrieveAuthCode(@FortifyPrivateSource String authCode) {
... }
```

特定のソース注釈に加えて、OpenTextには `FortifySource` という名前の、汎用の*信頼できない*汚染ソースが用意されています。

ソース注釈の完全なリストを次に示します。

- `FortifySource`
- `FortifyDatabaseSource`
- `FortifyFileSystemSource`
- `FortifyNetworkSource`
- `FortifyPCISource`
- `FortifyPrivateSource`
- `FortifyWebSource`

パススルー注釈

パススルー注釈では、ターゲットメソッドの入力から出力にすべての汚染が転送されます。 `FortifyNumberPassthrough` および `FortifyNotNumberPassthrough` の場合は、出力から汚染を割り当てたり、削除したりすることもできます。 `in` 注釈パラメータで使用できる値は、 `this` または関数パラメータ名です。 `out` 注釈パラメータで使用できる値は、 `this` 、 `return` 、または関数パラメータ名です。

```
@FortifyPassthrough(in="a",out="return") String
toLowerCase(String a) { ... }
```

データが純粋に数値であることを示すには、`FortifyNumberPassthrough`を使用します。数値データは、ソースに関係なく、クロスサイトスクリプティングなどの特定のタイプの問題を引き起こす可能性があります。`FortifyNumberPassthrough`を使用すると、このタイプの誤検出が減少します。プログラムで文字データを数値型(intやint[]など)に分解する場合は、`FortifyNumberPassthrough`を使用できます。プログラムで数値データを文字または文字列データに連結する場合は、`FortifyNotNumberPassthrough`を使用します。

パススルー注釈の完全なリストを次に示します。

- `FortifyPassthrough`
- `FortifyNumberPassthrough`
- `FortifyNotNumberPassthrough`

シンク注釈

シンク注釈では、適切なタイプの汚染がターゲットメソッドの入力に到達したときに問題が報告されます。注釈パラメータで使用できる値は、`this`または関数パラメータ名です。

```
@FortifyXSSSink("a") void printToWebpage(int a) { ... }
```

関数の引数または戻りパラメータに注釈を適用することもできます。次の例では、汚染が引数`a`に到達したときに問題が報告されます。

```
void printToWebpage(int b, @FortifyXSSSink String a) { ... }
```

シンク注釈の完全なリストを次に示します。

- `FortifySink`
- `FortifyCommandInjectionSink`
- `FortifyPCISink`
- `FortifyPrivacySink`
- `FortifySQLSink`
- `FortifySystemInfoSink`
- `FortifyXSSSink`

検証注釈

検証注釈では、ターゲットメソッドの出力から汚染が削除されます。注釈パラメータで使用できる値は、`this`、`return`、または関数パラメータ名です。

```
@FortifyXSSValidate("return") String xssCleanse(String a) { ...
}
```

検証シンク注釈の完全なリストを次に示します。

- `FortifyValidate`
- `FortifyCommandInjectionValidate`
- `FortifyPCIValidate`
- `FortifyPrivacyValidate`
- `FortifySQLValidate`
- `FortifySystemInfoValidate`
- `FortifyXSSValidate`

1.24.5.2. フィールドと変数の注釈

これらの注釈は、フィールドと変数(ほとんどの場合)に適用できます。

パスワードとプライベートの注釈

パスワードとプライベートの注釈を使用して、ターゲットのフィールドまたは変数がパスワードであるのか、プライベートデータであるのかを示します。

```
@FortifyPassword String x; @FortifyNotPassword String pass;
@FortifyPrivate String y; @FortifyNotPrivate String cc;
```

この例では、「x」という文字列がパスワードとして識別され、プライバシー違反およびハードコーディングされたパスワードがチェックされます。「pass」という文字列は、パスワードとして識別されません。注釈がない場合は、誤検知が発生する可能性があります。`FortifyPrivate` および `FortifyNotPrivate` の注釈も同様に機能しますが、プライバシー違反の問題は発生しません。

負でない注釈と0以外の注釈

これらの注釈を使用して、ターゲットのフィールドまたは変数で許可されていない値を指定します。

```
@FortifyNonNegative int index; @FortifyNonZero double divisor;
```

この例では、`index` に負の値が割り当てられているか、`divisor` にゼロが割り当てられている場合に問題がレポートされます。

1.24.5.3. その他の注釈

戻り値チェックの注釈

`FortifyCheckReturnValue` 注釈を使用して、戻り値をチェックする必要がある関数のリストにターゲットメソッドを追加します。

```
@FortifyCheckReturnValue int openFile(String filename) { ... }
```

危険の注釈

`FortifyDangerous` 注釈を使用すると、ターゲット関数、フィールド、変数、またはクラスの使用がレポートされます。注釈パラメータに使用できる値は、`CRITICAL`、`HIGH`、`MEDIUM`、または `LOW` です。これらの値は、Fortify Priority Orderの値に基づいて問題を分類する方法を示しています。

```
@FortifyDangerous{"CRITICAL"} public class DangerousClass {
@FortifyDangerous{"HIGH"} String dangerousField;
@FortifyDangerous{"LOW"} int dangerousMethod() { ... } }
```

1.25. パフォーマンスの最適化

このセクションでは、OpenText SASTでさまざまな種類のコードベースを分析する際に、メモリ使用量とパフォーマンスを最適化するためのガイドラインとヒントについて説明します。

このセクションでは、次のトピックについて説明します。

1.25.1. ウィルス対策ソフトウェア

ウィルス対策ソフトウェアを使用すると、OpenText SASTのパフォーマンスが悪影響を受ける可能性があります。スキャン時間が長い場合に、ウィルス対策ソフトウェアスキャンから内部のOpenText SASTファイルを一時的に除外することが推奨されています。ソースコードが存在するディレクトリでも同じ操作を実行できますが、分析時のパフォーマンスの影響は内部ディレクトリの場合よりも小さいです。

デフォルトでは、OpenText SASTの内部ファイルが次の場所に作成されます。

- Windows: `c:\Users\<username>\AppData\Local\Fortify\sca<version>`
- Windows以外: `<userhome>/.fortify/sca<version>`

ここで、*<version>*は使用しているOpenText SASTのバージョンです。

1.25.2. ハードウェアに関する考慮事項

さまざまなソースコードがあるため、メモリ使用量やスキャン時間を正確に予測することはできません。次のさまざまな要因によってメモリ使用量やパフォーマンスが影響を受けます。

- コードのタイプ
- コードベースのサイズと複雑さ
- 使用される補助言語(JSP、JavaScript、HTMLなど)
- 脆弱性の数
- 脆弱性のタイプ(使用されるアナライザ)

OpenTextでは、実際のアプリケーションスキャンの結果に基づいて、次の「最適な推測」というハードウェアの推奨事項が開発されました。次の表には、アプリケーションの複雑性に基づいた推奨事項のリストを示します。一般に、使用可能なコアの数を増やすと、スキャン時間が短縮される可能性があります。

アプリケーションの複雑さ	CPUコア数	RAM (GB)	平均スキャン時間	説明 (Description)
単純	4	16	1時間	サーバまたはデスクトップ上で実行されるスタンドアロンシステム(バッチジョブやコマンドラインツールなど)。
Medium	8	32	5時間	複雑なコンピュータモデルで動作するスタンドアロンシステム(税額計算システムやスケジューリングシステムなど)。
複雑	16	128	4日間	トランザクションデータ処理を備えた3階層ビジネスシステム(財務システムや商用Webサイトなど)。

アプリケーションの複雑さ	CPUコア数	RAM (GB)	平均スキャン時間	説明 (Description)
非常に複雑	32	256	7日間以上	コンテンツを配信するシステム(アプリケーションサーバ、データベースサーバ、コンテンツ管理システムなど)。

Note

TypeScriptスキャンおよびJavaScriptスキャンでは、分析時間が大幅に長くなります。アプリケーション内のコード行の合計がTypeScriptまたはJavaScriptの20%を超える場合は、2番目に高い推奨事項を使用します。

システム要件については、「ハードウェア要件」を参照してください。ただし、大規模で複雑なアプリケーションの場合は、OpenText SASTでより高い能力を持つハードウェアが必要です。その内容は次のとおりです。

- **ディスクI/O** - OpenText SASTはI/O集中型であるため、ハードドライブが高速になるほど、多くのI/Oトランザクションが節約されます。7,200 RPMドライブが推奨されていますが、10,000 RPMドライブ(WD Raptorなど)またはSSDドライブの方が適切です。
- **メモリ** - 最適なパフォーマンスを得るために必要なメモリの量を決定する方法の詳細については、「メモリチューニング」を参照してください。
- **CPU**—2.1 GHz以上のプロセッサが推奨されています。

1.25.3. チューニングオプション

OpenText SASTでは、複雑なプロジェクトの処理に長い時間がかかる場合があります。次のようにさまざまなフェーズで時間が費やされます。

- 変換
- 分析

OpenText SASTでは、大きな分析結果ファイル(FPR)が生成される可能性があります。その結果、監査とApplication Securityへのアップロードに長い時間がかかる場合があります。このフェーズは、次のように呼ばれます。

- 監査/アップロード

次の表では、時間のかかるさまざまなフェーズでパフォーマンスを向上させる方法に関するヒントを示します。

フェーズ	オプション	説明(Description)	詳細情報
変換	`-export-build-session` `-import-build-session`	さまざまなコンピュータでの変換とスキャン	モバイルビルドセッション
分析	`-quick`	クイックスキャンの実行	クイックスキャン
分析	`-scan-precision`	スキャン精度の設定	短縮ダイヤルを使用したスキャン速度の設定
分析	`-bin`	バイナリに関連するファイルのスキャン	コードベースの分割
分析	`-Xmx<size>M | G`	最大ヒープサイズの設定	メモリのチューニング
分析	`-Xss<size>M | G`	各スレッドのスタックサイズの設定	メモリのチューニング
分析 監査/アップロード	`-filter <file>`	フィルタファイルを使用したフィルタの適用	フィルタファイルの使用
分析 監査/アップロード	`-disable-source-bundling`	FPRファイルからのソースファイルの除外	FPRからのソースコードの除外

1.25.4. クイックスキャン

クイックスキャンモードを使用すると、プロジェクトを高速にスキャンして、重大な問題や優先度の高い問題を特定できます。OpenText SASTは、分析の深さを減らすことでスキャンを高速に実行します。さらに、Quick Viewフィルタセットも適用されます。クイックスキャン設定は設定可能です。クイックスキャンモードの設定について詳しくは、fortify-sca-quickscan.propertiesを参照してください。

クイックスキャンは、評価を通じて多くのアプリケーションを取得し、問題をすばやく見つけて修復を開始できるようにするための最適な方法です。パフォーマンスがどの程度向上するかは、アプリケーションの複雑さとサイズによって異なります。このスキャンはフルスキャンよりも高速ですが、結果セットの堅牢性は劣ります。可能な限りフルスキャンを実行することをお勧めします。

リミッタ

OpenText SASTの分析の深さは、利用可能なリソースに依存する場合があります。OpenText SASTは、複雑性のメトリックを使用して、これらのリソースと検出可能な脆弱性の数のバランスを取ります。このため、OpenText SASTが十分なリソースを使用できないと見なされる場合は、特定の機能を実行しないことがあります。

OpenText SASTでは、ユーザがOpenText SASTのリミッタプロパティを使用して「カットオフ」ポイントを制御できます。アナライザごとに異なるリミッタが用意されています。クイックスキャンを使用して、これらのリミッタの事前定義されたセットを実行できます。リミッタについて詳しくは、fortify-sca-quickscan.propertiesを参照してください。

クイックスキャンモードを有効にするには、`-quick`オプションとともに`-scan`オプションを使用します。クイックスキャンモードを有効にすると、OpenText SASTは標準の`<sast_install_dir>/Core/config/fortify-sca-quickscan.properties`ファイルに加えて`<sast_install_dir>/Core/config/fortify-sca.properties`ファイルのプロパティを適用します。`fortify-sca-quickscan.properties`ファイルを編集して、OpenText SASTが使用するリミッタを調整できます。`fortify-sca.properties`を変更すると、クイックスキャンの動作にも影響します。クイックスキャンモードでパフォーマンスを調整し、正確なスキャンを実行するためにフルスキャンをデフォルト設定のままにすることを推奨します。クイックスキャンモードのプロパティの詳細については、「OpenText SASTのプロパティファイル」を参照してください。

クイックスキャンとフルスキャンの使用

- **フルスキャンを定期的に実行する** - フルスキャンは、クイックスキャンモードでは検出されない問題を検出できる可能性があるため、定期的に実行することが重要です。

ソフトウェアのイテレーションごとに、少なくとも1回はフルスキャンを実行します。可能であれば、週末など、開発ワークフローが中断されないタイミングで定期的にフルスキャンを実行します。

- **クイックスキャンとフルスキャンを比較する** - クイックスキャンの正確な影響を評価するには、同じコードベースでクイックスキャンとフルスキャンを実行します。Fortify Audit Workbenchでクイックスキャンの結果を開き、それをフルスキャンにマージします。問題を**新しい問題**でグループ化して、フルスキャンで検出されたがクイックスキャンでは検出されなかった問題のリストを生成します。
- **クイックスキャンとApplication Security** — フルスキャンの結果が上書きされるのを防ぐため、デフォルトではApplication Securityは、クイックスキャンモードでスキャンされてアップロードされたFPRファイルを無視します。ただし、Application Securityのアプリケーション版は、クイックスキャンでスキャンされたFPRファイルを処理するように設定できます。詳しくは、*OpenText™ Application Securityユーザガイド*の分析結果の処理ルールを参照してください。

1.25.5. 短縮ダイヤルを使用したスキャン速度の設定

分析フェーズの精度レベルを指定して、スキャンの速度と深さを設定できます。これらの精度レベルを使用して、たとえば、パイプラインに適合するようにスキャン時間を調整すると、開発者がコードの作業を続けながら一連の脆弱性をすばやく見つけ出すことができます。短縮ダイヤル設定を使用したスキャンは、フルスキャンよりも高速ですが、結果セットの堅牢性は劣ります。可能な限りフルスキャンを実行することをお勧めします。

精度レベルでは、設定プロパティを各レベルに関連付けることで、スキャンの深さと精度を制御します。各レベルの設定プロパティファイルは、`<sast_install_dir>/Core/config/scales` ディレクトリにあります。レベルごとに1つのファイル(`level-<precision_level>.properties`)があります。これらのファイルの設定を変更して、独自の精度レベルを作成できます。

メモ:

- Application Securityでは、4未満の精度レベルで作成され、アップロードされた分析結果がデフォルトでブロックされます。ただし、これらの精度レベルでスキャンされ、アップロードされた監査プロジェクトが処理されるように、Application Securityのアプリケーションバージョンを設定できます。
- 短縮ダイヤルスキャンをフルスキャンとマージすると、アプリケーションに残っている以前のスキャンの問題が削除される場合があります(フルスキャンを実行すると再び検出されます)。

スキャンの短縮ダイヤル設定を指定するには、次の例に示すように、スキャンフェーズに `-scan-precision` (または `-p`)オプションを含める必要があります。

```
sourceanalyzer -b MyProject -scan -scan-precision <level> -f
MyResults.fpr
```

Note

短縮ダイヤル設定と `-quick` オプションを同じスキャンコマンドで使用することはできません。

次の表では、4つの精度レベルについて説明します。

精度レベル	説明(Description)
1	これは最も高速なスキャンであり、数個のファイルをスキャンするのにお勧めします。この精度レベルのスキャンでは、Buffer Analyzer、Control Flow Analyzer、Dataflow Analyzer、およびNull Pointer Analyzerがデフォルトで無効になります。
2	この精度レベルのスキャンでは、すべてのアナライザがデフォルトで有効になります。リミッタを減らして実行すると、スキャンの実行速度が向上します。その結果、検出される問題の数が少なくなります。
3	この精度レベルでは、中間開発スキャンの速度が最大50%向上します(報告される問題も減少します)。具体的には、このレベルではJavaやC/C++などの型付け言語のスキャン時間が向上します。
4	これはフルスキャンと同じです。

`com.fortify.sca.PrecisionLevel` ファイルの `fortify-sca.properties` プロパティを使用してスキャン精度レベルを指定することもできます。例:

```
com.fortify.sca.PrecisionLevel=1
```

1.25.6. コードベースの分割

大規模なプロジェクトを独立したモジュールに分割すると、効率が向上します。たとえば、ポータルアプリケーションを構成する複数のモジュールが互いに独立しているか、モジュール間のやり取りが少ない場合は、モジュールを個別に変換およびスキャンできます。ただし、何らかのやり取りがある場合は、データフローの問題を検出できない可能性があることに注意する必要があります。

C/C++の場合は、`–bin`オプションとともに`–scan`オプションを使用すると、スキャン時間が短縮される可能性があります。バイナリファイルをパラメータとして渡す必要があります(`-bin <filename>.exe -scan`または`-bin <filename>.dll -scan`など)。OpenText SASTでは、バイナリに関連するファイルが検索され、スキャンされます。これは、makefileに複数のバイナリがある場合に便利です。

次の表は、コードベースを分割する際に役立つOpenText SASTコマンドラインオプションのリストです。

オプション	説明(Description)
`-bin <binary>`	スキャンするソースファイルのサブセットを指定します。ビルド時に名前付きバイナリにリンクされたソースファイルのみがスキャンに含められます。このオプションを複数回使用すると、スキャンに複数のバイナリが含まれるように指定できます。
`-show-binaries`	他のバイナリの生成時に作成されたが、使用されていないオブジェクトが表示されます。ビルドに完全に統合されている場合は、作成されたバイナリがすべて表示されます。
`-show-build-tree`	`-bin` オプションとともに使用すると、バイナリを作成するために使用されるファイルと、それらのファイルをツリーレイアウトで作成するために使用されるファイルがすべて表示されます。`-bin` オプションが存在しない場合、OpenText SAST ではバイナリごとにツリーが表示されます。

1.25.7. アナライザと言語の制限

場合によっては、1つのアナライザの実行や特定の言語の分析に膨大な時間が費やされることがあります。このアナライザや言語は、セキュリティ要件にとって重要ではない可能性があります。実行するアナライザとOpenText SASTで変換する言語を制限できますが、その結果は最適とは言えない場合があります。

このセクションでは、次のトピックについて説明します。

1.25.7.1. アナライザの無効化

特定のアナライザを無効にするには、スキャン時にOpenText SASTに `-analyzers` オプションを追加して、有効にするアナライザのカンマ区切りまたはコロン区切りリストを指定する必要があります。 `-analyzers` オプションの有効なパラメータ値は、`buffer` `content` `configuration` 、 `controlflow` 、 `dataflow` 、 `nullptr` 、 `semantic` 、および `structural` です。

たとえば、Dataflow、Control Flow、およびBufferアナライザのみを含むスキャンを実行するには、次のスキャンコマンドを使用します。

```
sourceanalyzer -b MyProject -analyzers
dataflow:controlflow:buffer -scan -f MyResults.fpr
```

OpenText SASTのプロパティファイル(`com.fortify.sca.DefaultAnalyzers`)に `com.fortify.sca.DefaultAnalyzers` を設定して、同じ操作を実行することもできます。たとえば、前のスキャンコマンドと同等の結果を得る場合は、プロパティファイルに次の値を設定します。

```
com.fortify.sca.DefaultAnalyzers=dataflow:controlflow:buffer
```

1.25.7.2. 言語の無効化

特定の言語を無効にするには、除外する言語のリストを指定する `-disable-language` オプションを変換フェーズに含めます。有効な言語の値は次のとおりです。

`abap`、`actionscript`、`apex`、`cfml`、`cobol`、`configuration`、`cpp`、`dart`、`dotnet`、`golang`、`objc`、`php`、`python`、`ruby`、`swift`、および `vb`。

たとえば、設定およびPHPファイルを除外する変換を実行するには、次のコマンドを使用します。

```
sourceanalyzer -b MyProject <src_files> -disable-language
configuration:php
```

OpenText SASTのプロパティファイル(`com.fortify.sca.DISabledLanguages`)で `com.fortify.sca.DISabledLanguages` プロパティを設定して、言語を無効にすることもできます。たとえば、前の変換コマンドと同等の結果を得る場合は、プロパティファイルに次の値を設定します。

```
com.fortify.sca.DISabledLanguages=configuration:php
```

`-disable-language` で使用できない言語については、`-exclude` オプションを使用します。詳細については、「変換オプション」を参照してください。

1.25.8. FPRファイルの最適化

このセクションでは、監査結果(FPR)ファイルに関連するパフォーマンスの問題を処理する方法について説明します。これらのトピックでは、スキャン時間の短縮方法、FPRファイルサイズの削減方法、および大きなFPRファイルを開くためのヒントについて説明します。

このセクションでは、次のトピックについて説明します。

1.25.8.1. フィルタファイルの使用

ファイルを使用して、特定の脆弱性インスタンス、ルール、および脆弱性カテゴリを分析結果から除外できます。特定の問題カテゴリまたはルールが特定のスキャンに関連していないと判断した場合は、それらをFPRに追加しないようにOpenText SASTを設定できます。フィルタファイルを使用すると、スキャン時間と分析結果ファイルのサイズの両方を削減できます。

たとえば、指定したファイルを読み取るだけの単純なプログラムをスキャンする場合、パス操作の問題は機能の一部として計画されていない可能性が高いため、あまり見たくないかもしれません。パス操作の問題をフィルタで除外するには、次の1行を含むファイルを作成します。

`Path Manipulation`

このファイルを `filter.txt` という名前で保存します。次の例に示すように、分析フェーズで `-filter` オプションを使用します。

```
sourceanalyzer -b MyProject -scan -filter filter.txt -f
MyResults.fpr
```

`MyResults.fpr` の分析出力には、パス操作カテゴリの問題は含まれません。フィルタファイルの詳細と例については、「フィルタファイルによる問題の除外」を参照してください。

1.25.8.2. フィルタセットの使用

問題テンプレートのフィルタによって、OpenText SASTの結果の表示方法が決まります。フィルタに加えてフィルタセットを使用すると、同時に使用するフィルタを選択できます。各FPRには、関連付けられた問題テンプレートがあります。フィルタセットを使用すると、問題テンプレートのフィルタで指定した条件に基づいて問題の数を減らすことができます。これにより、FPRのサイズを大幅に削減できます。

これを行うには、Fortify Audit Workbenchを使用してフィルタセット内にフィルタを作成してから、そのフィルタセットと、そのフィルタセットが含まれている問題テンプレートを使用してOpenText SASTスキャンを実行します。フィルタセットの作成方法の詳細と基本的な例については、「フィルタセットによる問題の除外」を参照してください。

Note

フィルタセットで問題をフィルタリングすると、FPRのサイズは削減されますが、通常、スキャン時間は短縮されません。OpenText SASTは、問題を計算してFPRファイルに書き込むかどうかを判断した後で、フィルタセットを調べます。フィルタセット内のフィルタによって、OpenText SASTがロードするルールタイプが決まります。

1.25.8.3. FPRからのソースコードの除外

FPRからソースコード情報を除外することで、FPRファイルのサイズを削減できます。これは、大きなソースファイルやコードベースの場合に特に便利です。通常、この方法を使用しても、小さなソースファイルのスキャン時間は短縮されません。

OpenText SASTでFPRにソースコードが含まれないようにするために使用できるプロパティがあります。 `<sast_install_dir>/Core/config/fortify-sca.properties` いずれのプロパティもファイル内で設定するか、コマンドラインでオプションを指定することができます。次の表では、これらの設定について説明します。

プロパティ名	説明(Description)

`com.fortify.sca.FPRDisableSourceBundling=true`
コマンドラインオプション:
`-disable-source-bundling`
	FPRからソースコードを除外します。

`com.fortify.sca.FVDLDisableSnippets=true`
コマンドラインオプション:
`–fvdl-no-snippets`
	FPRからコードスニペットを除外します。

次のコマンドライン例では、両方のオプションを使用して、FPRからソースコードとコードスニペットの両方を除外しています。

```
sourceanalyzer -b MyProject -disable-source-bundling -fvdl-no-
snippets -scan -f MySourcelessResults.fpr
```

1.25.8.4. FPRファイルサイズの削減

FPRファイルのサイズを減らすには、いくつかの方法があります。結果に影響を与えずにこれを実現する最も簡単な方法は、「FPRからのソースコードの除外」で説明されているように、FPRからソースコードを除外することです。また、マージされたFPRのサイズをFPRUtilityで削減することもできます(『*OpenText™ Application Securityツールガイド*』を参照)。

FPRから除外する項目を選択するために使用できるプロパティが他にもいくつかあります。これらのプロパティを `<sast_install_dir>/Core/config/fortify-sca.properties` ファイルで設定するか、分析(スキャン)フェーズのコマンドラインでオプションを指定することができます。

プロパティ名	説明(Description)
com.fortify.sca.FPRDisableMetatable=true **コマンドラインオプション:** -disable-metatable	FPRからメタテーブルを除外します。Fortify Audit Workbenchは、メタテーブルを使用して関数ビューに情報をマップします。
com.fortify.sca.FVDLDisableDescriptions=true **コマンドラインオプション:** -fvdl-no-descriptions	FPRからルールの説明を除外します。カスタムの説明を使用しない場合は、Fortify Taxonomy (https://vulncat.fortify.com)の説明が使用されます。
com.fortify.sca.FVDLDisableEngineData=true **コマンドラインオプション:** -fvdl-no-enginedata	FPRからエンジンデータを除外します。これは、Fortify Audit Workbenchでファイルを開くときにFPRに多くの警告が含まれている場合に便利です。 **Note** FPRからエンジンデータを除外する場合は、FPRをApplication Securityにアップロードする前に、ローカルでFPRを現在の監査プロジェクトとマージする必要があります。FPRにはOpenText SASTのバージョンが含まれていないため、Application Securityはサーバ上でFPRをマージできません。

プロパティ名	説明(Description)

`com.fortify.sca.FVDLDisableProgramData=true`
 	
コマンドラインオプション:
`-fvdl-no-progdata`
	FPRからプログラムデータを除外します。これにより、Fortify Audit Workbenchの関数ビューから汚染されたソースの情報が削除されます。通常、このプロパティがFPRファイルの全体的なサイズに及ぼす影響は最小限です。

1.25.8.5. 大きなFPRファイルを開く

大きなFPRファイルをFortify Audit Workbenchで開くのに必要な時間を短縮するために、`<sast_install_dir>/Core/config/fortify.properties` ファイルにいくつかのプロパティを設定できます。これらのプロパティについて詳しくは、『*OpenText™ Application Security ツールガイド*』を参照してください。次の表は、大きなFPRファイルを開く時間を短縮するために使用できるプロパティについて説明しています。

プロパティ名	説明(Description)
`com.fortify.model.DisableProgramInfo=true`	Fortify Audit Workbenchのコードナビゲーション機能の使用を無効にします。
`com.fortify.model.IssueCutoffStartIndex=<num>` (包括的) `com.fortify.model.IssueCutoffEndIndex=<num>` (排他的)	問題カットオフの開始インデックスと終了インデックスを設定します。`IssueCutoffStartIndex` プロパティ(包括的)と `IssueCutoffEndIndex` (排他的)で、表示する問題のサブセットを指定できます。たとえば、最初の100個の問題を表示するには、次のように指定します。 `com.fortify.model.IssueCutoffStartIndex=0` `com.fortify.model.IssueCutoffEndIndex=101` `IssueCutoffStartIndex` はデフォルトで `0` なので、このプロパティを指定する必要はありません。
`com.fortify.model.IssueCutoffByCategoryStartIndex=<num>` (包括的) `com.fortify.model.IssueCutoffByCategoryEndIndex=<num>` (排他的)	問題カットオフの開始インデックスをカテゴリ別に設定します。これら2つのプロパティは、前のカットオフプロパティと似ていますが、カテゴリごとに指定する点が異なります。たとえば、各カテゴリの最初の5つの問題を表示するには、次のように指定します。 `com.fortify.model.IssueCutoffByCategoryEndIndex=6`
`com.fortify.model.MinimalLoad=true`	FPRからロードされるデータを最小化します。関数ビューの使用も制限され、Fortify Audit WorkbenchがFPRからソースをロードできなくなる場合があります。

プロパティ名	説明(Description)
`com.fortify.model.MaxEngineErrorCount=<num>`	FPRからロードする、OpenText SASTによって報告される警告の数を指定します。スキャンの警告が多いプロジェクトでは、この数をデフォルトの3000から減らすと、大きなFPRファイルのロード時間を短縮できます。
`com.fortify.model.ExecMemorySetting`	Fortify Audit Workbenchでiidmigratorやfortifyupdateなどの外部コマンドラインツールを開始するためのJVMヒープメモリサイズを指定します。

1.25.9. 長時間実行されているスキャンの監視

OpenText SASTの実行中に大規模で複雑なスキャンを実行すると、完了までに長い時間がかかることがよくあります。スキャン中に、状況を常に把握できるとは限りません。デバッグログをカスタマサポートチームに提供することが推奨されていますが、OpenText SASTで実行されている操作とそのパフォーマンスをリアルタイムで確認するには、いくつかの方法があります。

このセクションでは、次のトピックについて説明します。

1.25.9.1. SCAStateツールの使用

SCAStateコマンドラインツールを使用すると、分析フェーズ中に最新の状態分析情報を確認できます。SCAStateツールは `<sast_install_dir>/bin` ディレクトリにあります。分析のライブビューに加えて、OpenText SASTが分析フェーズ中にどこで時間を費やしているかを示すタイマとカウンタのセットも用意されています。SCAStateの使用法の詳細については、「*OpenText SASTスキャンステータスの確認*」を参照してください。

1.25.9.2. JMXツールの使用

ツールを使用すると、JMXテクノロジでOpenText SASTを監視できます。これらのツールでは、長期間にわたってOpenText SASTのパフォーマンスを追跡する方法が提供されます。これらのツールの詳細については、Oracle®のドキュメントを参照してください。

Note

これらはサードパーティのツールであり、OpenTextでは提供もサポートもされていません。

このセクションでは、次のトピックについて説明します。

1.25.9.2.1. JConsoleの使用

JConsoleは、JMX仕様に準拠した対話型のモニタリングツールです。JConsoleの欠点は、出力を保存できないことです。

JConsoleを使用するには、最初に追加のJVMパラメータをいくつか設定する必要があります。次の環境変数を設定します。

```
export SCA_VM_OPTS="-Dcom.sun.management.jmxremote -
Dcom.sun.management.jmxremote.port=9090 -
Dcom.sun.management.jmxremote.ssl=false -
Dcom.sun.management.jmxremote.authenticate=false"
```

JMXパラメータを設定したら、スキャンを開始します。スキャン時に次のコマンドを使用して、JConsoleを起動してOpenText SASTをローカルまたはリモートで監視します。

```
jconsole <host_name>:9090
```

1.25.9.2.2. Java VisualVMの使用

Java VisualVMでは、JConsoleと同じ機能が提供されています。また、JVMに関する詳細情報も提供され、監視情報をアプリケーションスナップショットファイルに保存できます。これらのファイルは保存して、後でJava VisualVMで開くことができます。

JConsoleと同様に、Java VisualVMを使用する前に、「JConsoleの使用」の説明と同じJVMパラメータを設定する必要があります。

JVMパラメータを設定したら、スキャンを開始します。次のコマンドを使用すると、Java VisualVMを起動してスキャンをローカルまたはリモートで監視できます。

```
jvisualvm <host_name>:9090
```

1.26. モバイルビルドセッションの使用

OpenText SASTモバイルビルドセッション(MBS)を使用すると、あるコンピュータでプロジェクトを変換し、別のコンピュータでスキャンできます。モバイルビルドセッション(MBSファイル)には、ソースコードを含めて、分析フェーズに必要なファイルがすべて含まれています。スキャン時間を改善するには、ビルドコンピュータ上で変換を実行し、次のいずれかを実行して、より装備の良いコンピュータを使用してスキャンを行います。

- ScanCentral SASTクライアントを使用して、分析のためにMBSをセンサに移動します(ScanCentral SASTを参照)。
- ビルドセッション(MBSファイル)を、OpenText SASTがインストールされている別のコンピュータに移動し、MBSをインポートして(「モバイルビルドセッションのインポート」を参照)、それから分析を実行します。

Note

OpenText Core Application Security (Fortify on Demand)ユーザは、MBSファイルを生成して、一部の言語をアップロードする変換されたコードをパッケージ化できます。

変換を実行するシステムと分析を実行するシステムの両方に同じバージョンのOpenText Application Security Content (Rulepack)がインストールされている必要があります。

このセクションでは、次のトピックについて説明します。

1.26.1. モバイルビルドセッションバージョンの互換性

変換コンピュータ上のOpenText SASTバージョンと分析コンピュータ上のOpenText SASTバージョンに互換性がある必要があります。バージョン番号の形式は、*<major>.<minor>.<patch>.<build_number>* (25.4.0.0140など)です。変換コンピュータと分析コンピュータの両方でOpenText SASTの*<major>*および*<minor>*部分が一致している必要があります。たとえば、25.4.0と25.4.xには互換性があります。OpenText SASTのバージョン番号を確認するには、コマンドラインで「`sourceanalyzer -v`」と入力します。

次のコマンドを使用してMBSファイルからビルドIDとOpenText SASTバージョンを取得できます。

```
sourceanalyzer -import-build-session <file>.mbs -
Dcom.fortify.sca.ExtractMobileInfo=true
```

1.26.2. モバイルビルドセッションの作成

変換を実行したコンピュータで次のコマンドを発行して、モバイルビルドセッションを生成します。

```
sourceanalyzer -b <build_id> -export-build-session <file>.mbs
```

ここで、 *<file>*.mbs はOpenText SASTのモバイルビルドセッションに指定するファイル名です。

1.26.3. モバイルビルドセッションのインポート

スキャンを実行するコンピュータに `<file>.mbs` ファイルを移動したら、モバイルビルドセッションをOpenText SASTのプロジェクトルートディレクトリにインポートできます。

モバイルビルドセッションをインポートするには、次のコマンドを入力します。

```
sourceanalyzer -import-build-session <file>.mbs
```

OpenText SASTのモバイルビルドセッションをインポートしたら、分析フェーズに進むことができます。変換に使用されたときと同じビルドIDでスキャンを実行します。

複数のモバイルビルドセッションを単一のMBSファイルにマージすることはできません。エクスポートされたビルドセッションごとに、一意のビルドIDが必要です。ただし、同じOpenText SASTのインストール時にすべてのビルドDがインポートされたら、 `-b` オプションを使用して1回のスキャンで複数のビルドDをスキャンできます(「分析フェーズ」を参照)。

1.27. トラブルシューティング

このセクションでは、次のトピックについて説明します。

1.27.1. 終了コード

次の表に、取り得るOpenText SAST終了コードを示します。

終了コード	説明(Description)
0	成功
1	一般的な障害
2	入力ファイルが不正です (これは、OpenText SASTでサポートされていない拡張子を持つファイルの変換が試みられたことを示す場合があります)
3	プロセスがタイムアウトしました
4	分析が完了し、番号付きの警告メッセージがコンソールまたはログファイルに書き込まれました
5	分析が完了し、番号付きのエラーメッセージがコンソールまたはログファイルに書き込まれました
6	スキャンフェーズで問題の結果を生成できませんでした
7	有効なライセンスを検出できないか、または実行時にLIMライセンスが期限切れになりました

デフォルトで、OpenText SASTでは終了コード0、1、2、3、または7だけが返されます。

`com.fortify.sca.ExitCodeLevel` ファイル内の `com.fortify.sca.ExitCodeLevel` プロパティを設定することで、デフォルトの終了コードオプションを拡張できます。

有効な値は次のとおりです。

- `nothing` —デフォルトの終了コード(0、1、2、3、または7)を返します。

- `warnings` —デフォルトの終了コードに加えて、終了コード4と5を返します。
- `errors` —デフォルトの終了コードに加えて、終了コード5を返します。
- `no_output_file` —デフォルトの終了コードに加えて、終了コード6を返します。

1.27.2. メモリのチューニング

スキャンに必要な物理RAMの量は、コードの複雑さによって異なります。デフォルトでは、システムで使用可能な物理メモリに基づいて、OpenText SASTにより使用するメモリが自動的に割り当てされます。通常はこれで十分です。出力オプションで説明するように、-Xmxコマンドラインオプションを使用してJavaヒープサイズを調整できます。

このセクションでは、分析中にOutOfMemoryエラーが発生した場合の解決方法について説明します。

Note

このセクションで説明するメモリ割り当てオプションを設定し、`SCA_VM_OPTS` 環境変数を設定してすべてのスキャンに対して実行することができます。

このセクションでは、次のトピックについて説明します。

1.27.2.1. Javaヒープの枯渇

Javaヒープの枯渇は、OpenText SASTスキャン時に発生する可能性のある最も一般的なメモリの問題です。これは、OpenText SASTでコードのスキャンに使用するJava仮想マシンに割り当てるヒープスペースが少なすぎるために発生します。次の症状からJavaヒープの枯渇を識別できます。

症状

これらのメッセージの1つ以上がOpenText SASTログファイルとコマンドライン出力に出現します。

```
There is not enough memory available to complete analysis. For
details on making more memory available, please consult the user
manual. java.lang.OutOfMemoryError: Java heap space
java.lang.OutOfMemoryError: GC overhead limit exceeded
```

解決策

Javaヒープの枯渇の問題を解決するには、スキャンの開始時にOpenText SAST Java仮想マシンに多くのヒープ領域を割り当てます。ヒープサイズを大きくするには、OpenText SASTスキャンの実行時に `-Xmx` コマンドラインオプションを使用します。たとえば、`-Xmx1G` は1GBを使用できるようにします。このパラメータを使用する前に、Javaヒープ領域で許容される最大値を決定してください。最大値は、使用可能な物理メモリによって異なります。

32 GB～48 GBのヒープサイズは、内部JVMの実装上推奨されていません。この範囲のヒープサイズでは、32 GBよりもパフォーマンスが低下します。32GB未満のヒープサイズはJVMによって最適化されます。スキャンで32GBを超えるサイズが必要な場合は、64GB以上が必要になります。ガイドラインとして、メモリを大量に消費するその他のプロセスが実行されていないと仮定すると、使用可能なメモリの2/3を超えて割り当てないでください。

システムがOpenText SAST実行専用の場合は、変更する必要があります。ただし、メモリを大量に消費するその他のプロセスとシステムリソースが共有されている場合は、それらの他のプロセスの許容値を差し引いてください。

Note

常駐していても(OpenText SASTの実行中に)アクティブではないプロセスは、オペレーティングシステムがディスクにスワップする可能性があっても考慮する必要はありません。環境内で使用可能なメモリよりも多くの物理メモリをOpenText SASTに割り当てると、「スラッシュ」が発生する可能性があります。この場合、通常はシステム上の他の機能とともにスキャンの速度が低下します。

1.27.2.2. ネイティブヒープの枯渇

ネイティブヒープの枯渇はまれなシナリオで、Java仮想マシンが起動時にJavaメモリ領域を割り当てできるものの、ネイティブ操作(ガーベジコレクションなど)のために残されたリソースがあまりに少なくなります。最終的には、プロセスが即座に終了する致命的なメモリ割り当てエラーが発生します。

症状

ネイティブヒープの枯渇は、OpenText SASTプロセスの異常終了およびコマンドラインでの次の出力により識別できます。

```
# A fatal error has been detected by the Java Runtime
Environment: # # java.lang.OutOfMemoryError: requested ... bytes
for GrET ...
```

これは致命的なJava仮想マシンエラーであるため、通常は、作業ディレクトリにファイル名 `hs_err_pidNNN.log` で作成されたエラーログが伴います。

解決策

この問題はプロセス内で過大な負荷が発生していることが原因であるため、Javaメモリ領域(Javaヒープ)に使用されるメモリの量を減らすことが解決策です。この値を小さくすると、負荷の問題が減り、スキャンを正常に完了させることができます。

1.27.2.3. スタックオーバーフロー

Javaアプリケーションのスレッドごとに独自のスタックがあります。スタックには、戻りアドレス、関数/メソッド呼び出しの引数などが保持されます。スレッドが再帰的アルゴリズムを使用して大規模な構造体を処理する傾向がある場合、すべての戻りアドレスのために大きなスタックが必要になる場合があります。JVMでは、`-Xss`オプションを使用してそのサイズを設定できます。

症状

通常、このメッセージはOpenText SASTログファイルに出現しますが、コマンドライン出力にも表示される場合があります。

```
java.lang.StackOverflowError
```

解決策

デフォルトのスタックサイズは16MBです。スタックサイズを大きくするには、`-Xss`コマンドに`sourceanalyzer`オプションを渡します。たとえば、`-Xss32M`を使用するとスタックが32MBに増えます。

1.27.3. 複雑な関数のスキャン

スキャン中に、Dataflow Analyzerで分析を完了できない関数が出現し、次のメッセージをレポートする場合があります。

```
Function <name> is too complex for <analyzer> analysis and will
be skipped (<identifier>)
```

ここで:

- *<name>* は、ソースコード関数の名前です。
- *<analyzer>* は、アナライザの名前です。
- *<identifier>* は、複雑性のタイプであり、次のいずれかになります。
  - l : 個別の場所が多すぎる
  - m : メモリ不足
  - s : スタックサイズが小さすぎる
  - t : 時間がかかりすぎる分析
  - v : 関数アクセス数が制限を超えた

OpenText SASTによる分析の深さは、利用可能なリソースに依存する場合があります。OpenText SASTは、複雑性のメトリックを使用して、これらのリソースと検出可能な脆弱性の数のバランスを取ります。このため、OpenText SASTが十分なリソースを使用できないと見なされる場合は、特定の関数の分析を中止することがあります。これは通常、「Function too complex」メッセージが表示される場合です。

このメッセージが表示されても、OpenText SASTでプログラム内の機能が完全に無視されたことを意味するわけではありません。たとえば、Dataflow Analyzerでは通常、分析を完了するまでに何度も関数にアクセスするため、それまでのアクセスではこの複雑性の限界に達していない可能性があります。この場合、結果には、前回の訪問で学習した情報がすべて含まれます。

リミッタと呼ばれるOpenText SASTプロパティを使用して、「中止」ポイントを制御できます。アナライザごとに異なるリミッタが用意されています。

次のセクションでは、この問題の解決方法について説明します。

このセクションでは、次のトピックについて説明します。

1.27.3.1. Dataflow Analyzerのリミッタ

Dataflow Analyzerには、次の3種類の複雑性識別子があります。

- `l`: 個別の場所が多すぎる
- `m`: メモリ不足
- `s`: スタックサイズが小さすぎる
- `v`: 関数アクセス数が制限を超えた

`s`で識別される問題を解決するには、`-Xss`を16MBを超える値に設定してスタックサイズを増やします。

複雑性識別子`m`を解決するには、OpenText SASTの物理メモリを増やします。

複雑性識別子`l`を解決するには、OpenText SASTプロパティファイル`<sast_install_dir>/Core/config/fortify-sca.properties`またはコマンドラインで次のリミッタを調整できます。

プロパティ名	デフォルト値
`com.fortify.sca.limiters.MaxTaintDefForVar`	1000
`com.fortify.sca.limiters.MaxTaintDefForVarAbort`	4000
`com.fortify.sca.limiters.MaxFieldDepth`	4

`MaxTaintDefForVar`リミッタは関数の複雑性を表すディメンションレス値ですが、`MaxTaintDefForVarAbort`は上限を示します。`MaxFieldDepth`リミッタを使用して、Dataflow Analyzerが特定のオブジェクトを分析する際の精度を測定します。OpenText SASTは常に可能な限り最高の精度でオブジェクトの分析を試みます。

指定された関数が指定された精度で`MaxTaintDefForVar`制限を超える場合、Dataflow Analyzerはその関数を(`MaxFieldDepth`リミッタを減らすことによって)低い精度で分析します。精度を低下させると、分析の複雑性が軽減されます。精度をさらに下げることができない場合、OpenText SASTは、分析が終了するか複雑性が

`MaxTaintDefForVarAbort` リミッタを超えるまで、最低の精度で分析を続行します。言い換えれば、OpenText SASTでは関数から少なくとも何らかの結果を得るために、最低の精度で最善を尽くそうとします。OpenText SASTが `MaxTaintDefForVarAbort` リミッタに達すると、関数の分析が完全に中止され、「Function too complex」(関数が複雑すぎる)という警告が表示されます。

複雑性識別子 `v` を解決するには、 `com.fortify.sca.limiters.MaxFunctionVisits` プロパティを調整できます。このプロパティは、テイント伝播アナライザから関数にアクセスする最大回数を設定します。デフォルトは `50` です。

1.27.3.2. 制御フローおよびNullポインタアナライザのリミッタ

制御フローアナライザとNullポインタアナライザには、次の2種類の複雑性識別子があります。

- `m`: メモリ不足
- `t`: 時間がかかりすぎる分析

Dataflow Analyzerが関数の複雑性を処理する方法により、無限に時間がかかるという問題はありません。ただし、Control Flow AnalyzerとNull Pointer Analyzerは、複雑な関数を分析する際に非常に長い時間がかかる場合があります。そのため、OpenText SASTではこのような場合に分析を中止する方法が用意されていて、複雑性識別子`t`で「Function too complex」というメッセージが表示されます。

これらのアナライザで関数分析に費やす最大時間を変更するには、OpenText SASTプロパティファイル`<sast_install_dir>/Core/config/fortify-sca.properties`またはコマンドラインで次のプロパティ値を調整できます。

プロパティ名	説明	デフォルト値
`com.fortify.sca.CtrlflowMaxFunctionTime`	単一の関数に制御フロー分析の時間制限(ミリ秒)を設定します。	600000 (10分)
`com.fortify.sca.NullPtrMaxFunctionTime`	単一の関数にNullポインタ分析の時間制限(ミリ秒)を設定します。	300000 (5分)

複雑性識別子`m`を解決するには、OpenText SASTの物理メモリを増やします。

Note

これらのリミッタや時間設定を増やすと、複雑な関数の分析にかかる時間が長くなります。リミッタ/時間の特定の値における正確なパフォーマンスへの影響を具体的に示すのは、当該関数に依存するため困難です。「Function too complex」という警告を表示しないようにする場合は、リミッタ/時間を極めて高い値に設定できます。ただし、スキャン時間が許容できないものになる可能性があります。

1.27.4. 問題の非決定性

並行分析モードで実行すると、問題の非決定性が発生する可能性があります。問題が発生した場合は、カスタマサポートに問い合わせ、並行分析モードを無効にしてください。並行分析モードを無効にすると逐次分析になり、処理速度は大幅に低下しますが、複数回のスキャンで決定論的な結果が得られます。

並行分析モードを無効にするには、次の手順を実行します。

1. `fortify-sca.properties` ディレクトリにある `fortify-sca.properties` ファイルをテキストエディタで開きます。
2. `com.fortify.sca.MultithreadedAnalysis` プロパティの値を `false` に変更します。

```
com.fortify.sca.MultithreadedAnalysis=false
```

1.27.5. ログファイルの場所の確認

今後、システム要件のドキュメントとリリースノートで `-debug` 、 `-verbose` 、および `-debug-verbose` オプションが非推奨となる予定です。GUIツールチームは、これらのオプションをツールで使用します。したがって、GUIツールチームにも知らせる必要があります。>>

デフォルトでは、OpenText SASTによって次の場所にログファイルが作成されます。

- Windows: `C:\Users\<username>\AppData\Local\Fortify\sca<version>\log`
- Windows以外: `<userhome>/.fortify/sca<version>/log`

ここで、 `<version>` は使用しているOpenText SASTバージョンです。

次の表で、OpenText SASTのデフォルトのログファイルについて説明します。

ファイル名	説明(Description)
`sca.log` `scaX.log`	標準ログには、sourceanalyzerの実行時に発生した情報メッセージ、警告、およびエラーのログが記録されます。
`sca_FortifySupport.log` `scaX_FortifySupport.log`	OpenText SASTサポートログには、次の機能があります。 • 標準ログファイルと同じログメッセージに、詳細を追加 • 標準ログファイルに含まれていない追加の詳細メッセージ このログファイルは、カスタマサポートまたは開発チームが問題をトラブルシュートする場合に役立ちます。

コマンドラインでログファイルを指定するには、「その他のオプション」を参照してください。

解決できない警告またはエラーが発生した場合は、OpenText SAST Supportログファイルをカスタマサポートにお送りください。

1.27.6. ログファイルの設定

OpenText SASTでログファイルに書き込む情報を設定するには、ログプロパティを設定(「ログプロパティ」を参照)し、 `<sast_install_dir>/Core/config/log4j2.xml` ファイルを更新します。次のログファイル設定を指定できます。

- ログファイルの場所と名前

  プロパティ: `com.fortify.sca.LogFile`

- ログレベル(ログレベルについてを参照)

  プロパティ: `com.fortify.sca.LogLevel`

- sourceanalyzerを実行するごとにログファイルを上書きするかどうか

  プロパティ: `com.fortify.sca.ClobberLogFile`

  コマンドラインオプション: `-clobber-log`

`log4j2.xml` ファイルに変更を加える方法については、https://logging.apache.org/log4j/2.x/manual/index.htmlを参照してください。

ログレベルについて

選択したログレベルでは、そのレベルと同等以上のすべてのログメッセージが出力されます。次の表は、ログレベルを最低から最高の順に示しています。たとえば、デフォルトのログレベルであるINFOには、INFO、WARN、ERROR、およびFATALレベルのログメッセージが含まれます。 `com.fortify.sca.LogLevel` ファイル内の `com.fortify.sca.LogLevel` プロパティ、またはコマンドラインで `-D` オプションを使用して、ログレベルを設定できます。

ログレベル	説明(Description)
DEBUG	カスタマサポートまたは開発チームが問題のトラブルシューティングに使用できる情報を含む
INFO	変換またはスキャンプロセスに関する基本的な情報
WARN	変換またはスキャンが停止しなかったが、正確な結果を得るために注意が必要になる可能性がある問題に関する情報
ERROR	注意が必要になる可能性がある問題に関する情報
FATAL	変換またはスキャンが中止したエラーに関する情報

1.27.7. 問題の報告と機能拡張の要求

フィードバックは、この製品の成功に不可欠です。機能拡張やパッチを要求したり、問題を報告したりするには、カスタマサポート(https://www.microfocus.com/support)にアクセスしてください。

カスタマサポートに連絡する場合は、次の情報を含める必要があります。

- 製品: OpenText SAST
- OpenText SASTのバージョン番号および任意の独立したOpenText SASTモジュール: バージョン番号を特定するには、次のコマンドを実行します。

  ```
  sourceanalyzer -version
  ```

- プラットフォーム: (たとえば、Red Hat Enterprise Linux *<version>*)
- オペレーティングシステム: (Linuxなど)

機能拡張を要求する場合は、機能拡張の説明を含めてください。

問題を報告する場合は、サポートが問題を再現できるよう十分な詳細を入力してください。説明が詳しくなるほど、サポートが問題を分析して解決するまでの時間を短縮できます。問題が発生したときのログファイル、またはログファイルで関連する部分も含めてください。

1.28. コマンドライン参照

このセクションでは、一般的なOpenText SASTのコマンドラインオプションと分析対象のソースファイルを指定する方法について説明します。特定の言語に固有のコマンドラインオプションについては、その言語のセクションで説明します。

このセクションでは、次のトピックについて説明します。

1.28.1. ファイルとディレクトリの指定

ファイル指定子は、ワイルドカード文字を使用して長いファイルリストまたはディレクトリをOpenText SASTに渡すことができる式です。OpenText SASTでは、2種類のワイルドカード文字が認識されます。単一のアスタリスク文字(*)はファイル名の一部に一致し、二重のアスタリスク文字(**)はディレクトリに再帰的に一致します。1つ以上のファイル、1つ以上のファイル指定子、またはファイルとファイル指定子の組み合わせを指定できます。複数のファイル指定子はセミコロン(Windows)またはコロン(Windows以外)で区切ります。

```
<files> | <file_dir_specifiers>
```

Windowsおよび多くのLinuxのシェルでは、アスタリスク文字(*)を含むパラメータが自動的に展開されます。そのため、ファイル指定子の式を引用符で囲む必要があります。また、Windowsでは、スラッシュ(/)の代わりにバックスラッシュ(\)をディレクトリ区切り文字として使用することもできます。

Note

ファイル指定子は、コンパイラやビルド統合を必要とする言語には適用されません。

次の表には、ファイルおよびディレクトリ指定子の例を示します。

ファイルまたはディレクトリ指定子	説明(Description)
<dir> "*<dir>*/**/*"	名前付きディレクトリと任意のサブディレクトリ内またはディレクトリパラメータで使用時の名前付きディレクトリ内のすべてのファイルに一致します。
"*<dir>*/**/Example.java"	名前付きディレクトリまたは任意のサブディレクトリ内で見つかった Example.java という名前の任意のファイルに一致します。
"*<dir>*/*.java" "*<dir>*/*.jar"	名前付きディレクトリ内で見つかった指定した拡張子付きの任意のファイルに一致します。
"*<dir>*/**/*.kt" "*<dir>*/**/*.jar"	名前付きディレクトリまたは任意のサブディレクトリ内で見つかった指定した拡張子付きの任意のファイルに一致します。
"*<dir>*/**/beta/**"	パスに beta が含まれている(ファイル名として beta を含む)名前付きディレクトリ内で見つかったすべてのディレクトリおよびファイルに一致します。
"*<dir>*/**/classes/"	名前付きディレクトリおよび任意のサブディレクトリ内で見つかった classes という名前のすべてのディレクトリとファイルに一致します。

ファイルまたはディレクトリ指定子	説明(Description)
`"**/test/**"`	パスに `test` 要素が含まれている(ファイル名として `test` を含む)現在のディレクトリツリー内のすべてのファイルに一致します。

`"**/test/**/*;**/build/**/*"`
または
`"**/test/**/*:**/build/**/*"`
	パスに `test` または `build` 要素が含まれている(ファイル名として `test` または `build` を含む)現在のディレクトリツリー内のすべてのファイルに一致します。

`"**/webgoat/*"`	現在のディレクトリツリーにある任意の `webgoat` ディレクトリ内のすべてのファイルに一致します。 一致する: • `/src/main/java/org/owasp/webgoat` • `/test/java/org/owasp/webgoat` 一致しない(`assignments` ディレクトリが一致しない) • `/test/java/org/owasp/webgoat/assignments`

1.28.2. ディレクティブ

一度に1つのディレクティブのみを使用します。ディレクティブを変換コマンドや分析コマンドと組み合わせて使用しないでください。次の表で説明するディレクティブを使用して、以前の変換コマンドに関する情報のリストを表示します。

ディレクティブ	説明(Description)
`-clean`	すべてのOpenText SAST中間ファイルとビルドレコードを削除します。ビルドIDを指定すると、そのビルドIDに関連するファイルとビルドレコードのみが削除されます。
`-show-binaries`	他のバイナリの生成時に使用されていない、作成済みのすべてのオブジェクトが表示されます。ビルドに完全に統合されている場合は、作成されたバイナリがすべて表示されます。
`-show-build-ids`	既知のすべてのビルドIDのリストが表示されます。
`-show-build-tree`	`-bin` オプションを使用してスキャンすると、バイナリの作成に使用されたファイルと、それらのファイルの作成に使用されたファイルをすべてツリーレイアウトで表示します。 `-bin` オプションが存在しない場合は、バイナリごとにツリーが表示されます。 **Note** このオプションを使用すると、大量の情報が生成される可能性があります。

ディレクティブ	説明(Description)
`-show-build-warnings`	`-b` オプションを指定して使用すると、変換フェーズで発生したエラーおよび警告がコンソールに表示されます。 **Note** Fortify Audit Workbenchでは、これらのエラーおよび警告が結果証明書タブにも表示されます。
`-show-files`	指定したビルドIDに含まれているファイルが表示されます。`-bin` オプションが存在する場合は、バイナリに渡されたソースファイルのみが表示されます。
`-show-loc`	`-b` オプションを指定して使用すると、変換されたコードの行数が表示されます。

1.28.2.1. LIMライセンスディレクティブ

OpenText SASTでは、LIMライセンスの使用状況を管理するためのディレクティブが提供されています。LIMのライセンスプール資格情報を保存またはクリアできます。指定したライセンスプールでデタッチされたリースが許可されている場合は、オフライン分析用にデタッチされたリースを要求(およびリリース)することもできます。

Note

デフォルトでOpenText SASTはLIMサーバにHTTPS接続する必要があり、信頼された証明書が必要です。詳細については、「信頼された証明書の追加」を参照してください。

次の表で説明するディレクティブは、LIMで管理されるライセンスに使用します。

LIMディレクティブ	説明(Description)

`-store-license-pool-credentials "<lim_url>|<lim_pool_name>|<lim_pool_pwd>|<proxy_url>|<proxy_user>|<proxy_pwd>"`	OpenText SASTでライセンスにLIMが使用されるように、LIMのライセンスプール資格情報を保存します。プロキシ情報はオプションです。OpenText SASTでは、このディレクティブで指定されたプールパスワードとプロキシ資格情報が `fortify-sca.properties` ファイルに暗号化されたデータとして保存されます。OpenText SASTのインストール後にライセンスプール資格情報が変更された場合は、このディレクティブを再度実行して新しい資格情報を保存できます。
	
	例:
	
	`sourceanalyzer -store-license-pool-credentials " https://<ip_address>:<port> |TeamA|mypassword"`
	
	関連付けられたプロパティ名:
	`com.fortify.sca.lim.Url`
	`com.fortify.sca.lim.PoolName`
	`com.fortify.sca.lim.PoolPassword`
	`com.fortify.sca.lim.ProxyUrl`
	`com.fortify.sca.lim.ProxyUsername`
	`com.fortify.sca.lim.ProxyPassword`

`-clear-license-pool-credentials`	LIMのライセンスプール資格情報を `fortify-sca.properties` ファイルから削除します。ライセンスプール資格情報が変更された場合は、このディレクティブを使用して削除してから、`-store-license-pool-credentials` ディレクティブを使用して新しい資格情報を保存できます。

LIMディレクティブ	説明(Description)
-request-detached-lease *<duration>*	このシステムで指定した期間(分単位)に排他的に使用されるように、LIMのライセンスプールからデタッチされたリースを要求します。これにより、企業のイントラネットから切断されている場合でもOpenText SASTを実行できます。 **Note** このディレクティブを使用するには、デタッチされたリースが許可されるようにライセンスプールが設定されている必要があります。
-release-detached-lease	デタッチされたリースをライセンスプールに戻します。

1.28.3. 変換オプション

次の表は、ほとんどの変換コマンドで使用できる一般的な変換オプションについて説明しています。

変換オプション	説明(Description)
-b *<build_id>*	ビルドIDを指定します。OpenText SASTでは、ビルドIDを使用して、ビルドの一環としてコンパイルおよび結合されたファイルが追跡され、後でそれらのファイルがスキャンされます。 **同等のプロパティ名:** com.fortify.sca.BuildID
-disable-language *<languages>*	変換フェーズから除外する言語のコロン区切りリストを指定します。有効な言語の値は次のとおりです。 abap、actionscript、apex、cfml、cobol、configuration、cpp、dart、dotnet、golang、objc、php、python、ruby、swift、および vb。 **同等のプロパティ名:** com.fortify.sca.DISabledLanguages
-enable-language *<languages>*	変換する言語のコロン区切りリストを指定します。有効な言語の値は次のとおりです。 abap、actionscript、apex、cfml、cobol、configuration、cpp、dart、dotnet、golang、objc、php、python、ruby、swift、および vb。 **同等のプロパティ名:** com.fortify.sca.EnabledLanguages

変換オプション	説明(Description)
`-exclude <file_specifiers>`	変換から除外するファイルを指定します。変換から除外されたファイルはスキャンもされません。複数のファイルパスはセミコロン(Windows)またはコロン(Windows以外)で区切ります。次の例では、任意の `Test` サブディレクトリ内のすべてのJavaファイルを除外します。 `sourceanalyzer -b MyProject –cp "**/*.jar" "**/*" -exclude "**/Test/*.java"` ファイル指定子の使い方について詳しくは、ファイルとディレクトリの指定を参照してください。 **同等のプロパティ名:** `com.fortify.sca.exclude`

変換オプション	説明(Description)
`-encoding <encoding_name>`	ソースファイルのエンコーディングタイプを指定します。OpenText SASTでは、エンコードの異なるソースファイルを含むプロジェクトをスキャンできます。マルチエンコーディングされたプロジェクトを使用するには、OpenText SASTで最初にソースコードファイルが読み込まれる際に、変換フェーズで `-encoding` オプションを指定する必要があります。OpenText SASTでは、このエンコーディングがビルドセッションで記憶され、FVDLファイルに反映されます。 有効なエンコーディング名は `java.nio.charset.Charset` です。 通常、エンコーディングタイプを指定しないと、OpenText SASTではエンコーディングパラメータなしで `file.encoding` コンストラクタから `java.io.InputStreamReader` が使用されます。一部のケース(ActionScriptパーサの場合など)では、OpenText SASTのデフォルトは `UTF-8` エンコーディングです。 **同等のプロパティ名:** `com.fortify.sca.InputFileEncoding`
`-nc`	コンパイラのコマンドラインの前に指定すると、OpenText SASTでソースファイルが変換されますが、コンパイラは実行されません。

変換オプション	説明(Description)
`-noextension-type <file_type>`	拡張子がないソースファイルにファイルタイプを指定します。有効なファイルタイプの値は、`ABAP`、`ACTIONSCRIPT`、`APEX`、`APEX_OBJECT`、`APEX_TRIGGER`、`ARCHIVE`、`ASPNET`、`ASP`、`ASPX`、`BITCODE`、`BSP`、`BYTECODE`、`CFML`、`COBOL`、`CSHARP`、`DART`、`DOCKERFILE`、`FLIGHT`、`GENERIC`、`GO`、`HCL`、`HOCON`、`HTML`、`INI`、`JAVA`、`JAVA_PROPERTIES`、`JAVASCRIPT`、`JINJA`、`JSON`、`JSP`、`JSPX`、`JUPYTER`、`KOTLIN`、`MSIL`、`MXML`、`OBJECT`、`PHP`、`PLSQL`、`PYTHON`、`RUBY`、`RUBY_ERB`、`SCALA`、`SWIFT`、`SWC`、`SWF`、`TLD`、`SQL`、`TSQL`、`TYPESCRIPT`、`VB`、`VB6`、`VBSCRIPT`、`VISUAL_FORCE`、`VUE`、`XML`、および`YAML`です。
`-disable-compiler-resolution`	ソースファイルとして変換中に、ビルドツールと同じ名前(gradlewなど)を持つビルドスクリプトファイルを含める場合に指定します。 **同等のプロパティ名:** `com.fortify.sca.DisableCompilerName`

変換オプション	説明(Description)
`-project-root`	変換および分析フェーズで生成された中間ファイルを保存するディレクトリを指定します。OpenText SASTでは、このプロジェクトのルートディレクトリにある中間ファイルを広く活用します。場合によっては、このディレクトリをネットワークドライブではなくローカルストレージに配置すると、分析のパフォーマンスが向上します。 **同等のプロパティ名:** `com.fortify.sca.ProjectRoot`

1.28.4. 分析オプション

次の表では、一般的な分析オプション(通常は `-scan` を使用)について説明します。

分析オプション	説明(Description)
-b *<build_id>*	前の変換コマンドで使用されるビルドIDを指定します。 **同等のプロパティ名:** com.fortify.sca.BuildID
-scan	OpenText SASTで、指定したビルドIDのセキュリティ分析が実行されます。
-scan-policy <policy_name> | -sc <policy_name>	分析のためのスキャンポリシーを指定します。有効なポリシー名は、classic、security、およびdevopsです。詳細については、「分析へのスキャンポリシーの適用」を参照してください。 **同等のプロパティ名:** com.fortify.sca.ScanPolicy
-analyzers *<analyzer_list>*	アナライザのコロン区切りリストまたはカンマ区切りリストで有効にするアナライザを指定します。有効なアナライザ名は、buffer、content、configuration、controlflow、dataflow、nullptr、semantic、およびstructuralです。このオプションを使用して、セキュリティ要件に必要ないアナライザを無効にすることができます。 **同等のプロパティ名:** com.fortify.sca.DefaultAnalyzers

分析オプション	説明(Description)
`-p <level>` | `-scan-precision <level>`	短縮ダイヤルを使用して、スキャン精度レベルを指定してプロジェクトをスキャンします。スキャン精度レベルが低いほど、スキャンのパフォーマンスは高速になります。有効な値は、`1`、`2`、`3`、および`4`です。詳細については、「スキャン速度の設定」を参照してください。 **同等のプロパティ名:** `com.fortify.sca.PrecisionLevel`
`-project-root`	変換および分析フェーズで生成された中間ファイルを保存するディレクトリを指定します。OpenText SASTでは、このプロジェクトのルートディレクトリにある中間ファイルを広く活用します。場合によっては、このディレクトリをネットワークドライブではなくローカルストレージに配置すると、分析のパフォーマンスが向上します。 **同等のプロパティ名:** `com.fortify.sca.ProjectRoot`
`-project-template <file>`	スキャンに使用する問題テンプレートファイルを指定します。これにより、ローカルコンピュータのスキャンのみが影響を受けます。FPRをApplication Securityにアップロードする場合は、アプリケーションバージョンに割り当てられた問題テンプレートが使用されます。 **同等のプロパティ名:** `com.fortify.sca.ProjectTemplate`

分析オプション	説明(Description)
`-quick`	`fortify-sca-quickscan.properties` ファイルを使用してプロジェクトのクイックスキャンを実行し、重大な問題や優先度の高い問題がないか調べます。この方法では、分析の詳細さは低下します。デフォルトでは、クイックスキャンの場合、Buffer AnalyzerとControl Flow Analyzerが無効になります。さらに、Quick Viewフィルタセットも適用されます。詳細については、「クイックスキャン」を参照してください。 **同等のプロパティ名:** `com.fortify.sca.QuickScanMode`
`-filter <file>`	結果フィルタファイルを指定します。詳細については、「結果の最適化」を参照してください。 **同等のプロパティ名:** `com.fortify.sca.FilterFile`
`-bin <binary>` | `-binary-name <binary>`	スキャンするソースファイルのサブセットを指定します。ビルド時に名前付きバイナリにリンクされたソースファイルのみがスキャンに含められます。このオプションを複数回使用すると、スキャンに複数のバイナリが含まれるように指定できます。 **同等のプロパティ名:** `com.fortify.sca.BinaryName`

分析オプション	説明(Description)
`-disable-default-rule-type <type>`	カスタムルールをテストするために使用します。デフォルトのRulepackで指定されたタイプのルールをすべて無効にします。このオプションを複数回使用すると、複数のルールタイプを指定できます。 `<type>` パラメータは、XMLタグからサフィックス `Rule` を除いた値です。たとえば、DataflowSourceRule要素の場合は `DataflowSource` を使用します。また、特性化ルールの特定のセクション(`Characterization:Control flow` 、 `Characterization:Issue` 、 `Characterization:Generic` など)を指定することもできます。 `<type>` パラメータでは、大文字と小文字が区別されません。

分析オプション	説明(Description)
`-no-default-issue-rules`	カスタムルールをテストするために使用します。問題が直接発生するデフォルトのRulepackでルールを無効にします。OpenText SASTでは、関数の動作を特徴付けるルールは引き続きロードされます。 **Note** これは、DataflowSink、Semantic、Controlflow、Structural、Configuration、Content、Statistical、Internal、Characterization:Issueルールタイプを無効にした場合と同等です。 **同等のプロパティ名:** `com.fortify.sca.NoDefaultIssueRules`
`-no-default-rules`	カスタムルールをテストするために使用します。デフォルトのRulepackからルールがロードされないようにします。OpenText SASTでは、説明要素と言語ライブラリのRulepackが処理されますが、ルールは処理されません。 **同等のプロパティ名:** `com.fortify.sca.NoDefaultRules`

分析オプション	説明(Description)
`-no-default-source-rules`	カスタムルールをテストするために使用します。デフォルトのRulepackでソースルールを無効にします。 **Note** 特性化ソースルールは無効になりません。 **同等のプロパティ名:** `com.fortify.sca.NoDefaultSourceRules`
`-no-default-sink-rules`	カスタムルールをテストするために使用します。デフォルトのRulepackでシンクルールを無効にします。 **Note** 特性化シンクルールは無効になりません。 **同等のプロパティ名:** `com.fortify.sca.NoDefaultSinkRules`
`-rules` *<file>* | *<dir>*	カスタムのルールパックまたはディレクトリを指定します。このオプションを複数回使用すると、複数のRulepackファイルを指定できます。ディレクトリを指定すると、OpenText SASTでは、そのディレクトリ内の `.bin` および `.xml` の拡張子を持つすべてのファイルが含められます。 **同等のプロパティ名:** `com.fortify.sca.RulesFile`

1.28.5. 出力オプション

次の表では、出力オプションについて説明します。これらのすべてのオプションを分析フェーズで適用します(`-scan` オプションも指定します)。 `build-label` 、 `build-project` 、および `build-version` オプションは、変換フェーズで指定できます。分析フェーズで再度指定すると、それらのオプションは上書きされます。

出力オプション	説明(Description)
`-f <file>` | `-output-file <file>`	分析結果が書き込まれるファイルを指定します。出力ファイルを指定しない場合は、OpenText SASTから端末に出力が書き込まれます。 **同等のプロパティ名:** `com.fortify.sca.ResultsFile`

出力オプション	説明(Description)
-format *<format>*	出力形式を制御します。有効なオプションは、fpr、fvdl、fvdl.zip、text、およびautoです。デフォルトはautoであり、-fオプションで指定されたファイルのファイル名拡張子に基づいて出力形式が選択されます。 FVDLは、OpenText SASTの詳細な分析結果が含まれるXMLファイルです。これには、脆弱性の詳細、ルールの説明、コードスニペット、スキャン時に使用されるコマンドラインオプション、スキャンのエラーまたは警告が含まれています。 FPRは、FVDLファイル、およびスキャン時に使用されるソースコードのコピー、外部メタデータ、カスタムルール(該当する場合)などの追加情報が含まれる分析結果のパッケージです。Fortify Audit Workbenchは、自動的に.fpr拡張子に関連付けられます。 **Note** 結果証明書を使用する場合は、fpr形式を指定する必要があります。結果証明書については、『*OpenText™ Fortify Audit Workbenchユーザガイド*』を参照してください。 一部の情報がFPRファイルまたはNFDLファイルに含まれないようにすると、スキャン時間や出力ファイルサイズを改善できます。この表のその他のオプション、および「FPRファイルの最適化」を参照してください。 **同等のプロパティ名:** com.fortify.sca.Renderer

出力オプション	説明(Description)
-append	-f オプションで指定されたファイルに結果を追加します。結果のFPRファイルには、以前のスキャンによる問題と現在のスキャンによる問題が含まれています。ビルド情報およびプログラムデータ(ソースとシンクのリスト)セクションもマージされます。このオプションを使用するには、出力ファイルの形式を fpr または fvdl にする必要があります。 -format 出力オプションについては、この表の説明を参照してください。 (分析中のプログラムに関する情報とは異なり) OpenText Application Security Contentの情報、コマンドラインオプション、システムプロパティ、警告、エラー、OpenText SASTの実行に関するその他の情報を含むエンジンデータはマージされません。エンジンデータは -append オプションを使用してマージされないため、OpenTextでは、 -append で生成された結果が証明されません。 このオプションを指定しない場合は、OpenText SASTで新しい結果がFPRファイルに追加され、 previous の結果として古い結果にラベルが付けられます。 一般に、同時にアプリケーション全体を分析できない場合にのみ、 -append オプションを使用します。 **同等のプロパティ名:** com.fortify.sca.OutputAppend

出力オプション	説明(Description)
`-build-label <label>`	分析結果に含めるプロジェクトのラベルを指定します。このオプションは、変換フェーズや分析フェーズ中に含めることができます。OpenText SASTでは、このラベルはコード分析に使用されません。変換と分析の両方にこのオプションを指定した場合、最後に指定したラベルだけが分析結果に渡されます。 **同等のプロパティ名:** `com.fortify.sca.BuildLabel`
`-build-project <project_name>`	分析結果に含めるプロジェクトの名前を指定します。このオプションは、変換フェーズや分析フェーズ中に含めることができます。OpenText SASTでは、この名前はコード分析に使用されません。 **同等のプロパティ名:** `com.fortify.sca.BuildProject`
`-build-version <version>`	分析結果に含めるプロジェクトのバージョンを指定します。このオプションは、変換フェーズや分析フェーズ中に含めることができます。OpenText SASTでは、このバージョンはコード分析に使用されません。 **同等のプロパティ名:** `com.fortify.sca.BuildVersion`

出力オプション	説明(Description)
`-disable-source-bundling`	ソースファイルを分析結果ファイルから除外します。分析結果にはスニペットが含まれます。 **同等のプロパティ名:** `com.fortify.sca.FPRDisableSourceBundling`
`-fvdl-no-descriptions`	OpenText Application Security Contentの説明を分析結果ファイルから除外します。 **同等のプロパティ名:** `com.fortify.sca.FVDLDisableDescriptions`
`-fvdl-no-enginedata`	エンジンデータを分析結果ファイルから除外します。エンジンデータには、OpenText Application Security Contentの情報、コマンドラインオプション、システムプロパティ、警告、エラー、およびOpenText SASTの実行に関するその他の情報が含まれています。 **同等のプロパティ名:** `com.fortify.sca.FVDLDisableEngineData`

出力オプション	説明(Description)
`-fvdl-no-progdata`	プログラムデータを分析結果ファイルから除外します。これにより、Fortify Audit Workbenchの関数ビューからテイントソース情報が削除されます。 **同等のプロパティ名:** `com.fortify.sca.FVDLDisableProgramData`
`-fvdl-no-snippets`	コードスニペットを分析結果ファイルから除外します。 **同等のプロパティ名:** `com.fortify.sca.FVDLDisableSnippets`

1.28.6. その他のオプション

次の表では、その他のオプションについて説明します。

その他のオプション	説明(Description)
`@<file>`	指定したファイルからコマンドラインオプションを読み込みます。プレーンテキストの*<file>*にはオプションとパラメータが含まれ、それぞれが各行に分離されています。 たとえば、`sourceanalyzer -b my_build_id -source 17 -cp lib.jar Test.java` コマンドを実行する代わりに、`sourceanalyzer @optfile.txt` を実行できます。ここで、`optfile.txt` は以下を含むファイルです: `"-b" "my_build_id" "-source" "17" "-cp" "lib.jar" "Test.java"`
`-h` | `-?` | `-help`	コマンドラインオプションの概要を出力します。
`-debug`	OpenText SAST Supportログファイルにデバッグ情報を含めます。この情報は、カスタマサポートによるトラブルシューティングにのみ役立ちます。 **同等のプロパティ名:** `com.fortify.sca.Debug`
`-debug-verbose`	これは `-debug` オプションと同じですが、特に解析エラーに関する詳細が含まれています。 **同等のプロパティ名:** `com.fortify.sca.DebugVerbose`

その他のオプション	説明(Description)
`-debug-mem`	パフォーマンス情報をOpenText SAST Supportログの中に含めます。 **同等のプロパティ名:** `com.fortify.sca.DebugTrackMem`
`-verbose`	詳細ステータスメッセージをコンソールとOpenText SAST Supportログファイルに送信します。 **同等のプロパティ名:** `com.fortify.sca.Verbose`
`-logfile <file>`	OpenText SASTで作成されるログファイルを指定します。デフォルトのログファイルの場所については、「ログファイルの場所」を参照してください。 **同等のプロパティ名:** `com.fortify.sca.LogFile`
`-clobber-log`	sourceanalyzerが実行されるたびにログファイルを上書きするようにOpenText SASTに指示します。このオプションを指定しない場合は、OpenText SASTでログファイルに情報が追加されます。 **同等のプロパティ名:** `com.fortify.sca.ClobberLogFile`
`-quiet`	コマンドラインの進行状況情報を無効にします。 **同等のプロパティ名:** `com.fortify.sca.Quiet`

その他のオプション	説明(Description)
`-version` | `-v`	OpenText SASTのバージョンと、OpenText SASTと一緒に含まれているさまざまな独立したモジュールのバージョンが表示されます(その他の機能はすべて、OpenText SASTに含まれています)。
`-autoheap`	システムで使用可能な物理メモリに基づいて、メモリの自動割り当てを有効にします。これはデフォルトのメモリ割り当て設定です。

その他のオプション	説明(Description)
`-Xmx<size>M | G`	OpenText SASTで使用されるメモリ使用量の最大値を手動で指定します。 **Note** このオプションで最大メモリを手動で指定する代わりに、`-autoheap`で定義されたデフォルトのメモリ割り当て設定を使用するようにお勧めします。 32 GB～48 GBのヒープサイズは、内部JVMの実装上推奨されていません。この範囲のヒープサイズでは、32 GBよりもパフォーマンスが低下します。JVMでは、32 GB未満のヒープサイズが最適化されます。スキャンで32GBを超えるサイズが必要な場合は、64GB以上が必要になります。ガイドラインとして、メモリを大量に消費するその他のプロセスが実行されていないと仮定すると、使用可能なメモリの2/3を超えて割り当てないでください。 このオプションを指定する場合は、パフォーマンスが低下するため、物理的に使用可能なメモリよりも多く割り当てないでください。ガイドラインとして、メモリを大量に消費するその他のプロセスが実行されていないと仮定すると、使用可能なメモリの2/3を超えて割り当てないでください。

1.29. 環境設定オプション

OpenText SASTインストーラでは、一連のプロパティファイルがシステムに保存されます。プロパティファイルには、OpenText SASTのランタイム分析、出力、およびパフォーマンスに環境設定可能な設定が含まれています。

このセクションでは、次のトピックについて説明します。

1.29.1. プロパティファイル

プロパティファイルは、 *<sast_install_dir>*/Core/config ディレクトリに配置されています。インストールされたプロパティファイルには、デフォルト値が含まれています。プロパティファイルのプロパティを変更する前に、プロジェクトリーダーに相談することが推奨されています。環境設定ファイル内のプロパティはいずれも、任意のテキストエディタで変更できます。 -D オプションを使用して、コマンドラインでプロパティを指定することもできます。

次の表に、OpenText SASTプロパティファイルを示します。OpenText SASTアプリケーションおよびツールのプロパティファイルについては、『*OpenText™ Application Securityツールガイド*』を参照してください。

プロパティファイル名	説明(Description)	詳細情報
fortify-sca.properties	OpenText SAST環境設定プロパティを定義します。	fortify-sca.properties
fortify-sca-quickscan.properties	OpenText SASTのクイックスキャンに適用される環境設定プロパティを定義します。	fortify-sca-quickscan.properties
fortify-rules.properties	ルールの動作を決定する環境設定プロパティを定義します。	fortify-rules.properties

1.29.1.1. プロパティファイルの形式

プロパティファイルの各プロパティは、文字列のペアで構成されます。1番目の文字列はプロパティ名、2番目の文字列はプロパティ値です。

```
com.fortify.sca.fileextensions.htm=HTML
```

上記のように、プロパティでは `.htm` ファイルで使用される変換が設定されます。プロパティ名は `com.fortify.sca.fileextensions.htm` であり、値は `HTML` に設定されています。

Note

Windowsシステムのパスをプロパティ値として指定する場合は、バックスラッシュ文字(\)をバックスラッシュでエスケープする必要があります(例: `com.fortify.sca.ASPVirtualRoots.Library=C:\\WebServer\\CustomerA\\inc`)。

無効なプロパティは、プロパティファイルからコメントアウトされています。これらのプロパティを有効にするには、コメント記号(#)を削除してから、プロパティファイルを保存します。次の例では、プロパティファイルで `com.fortify.sca.LogFile` プロパティが無効であり、環境設定の一部ではありません。

```
# default location for the log file
#com.fortify.sca.LogFile=${com.fortify.sca.ProjectRoot}/sca/log/
sca.log
```

1.29.1.2. 設定の上書き

OpenText SASTでは、プロパティ設定が特定の順序で使用されます。以前に設定したプロパティは、指定した値で上書きできます。プロパティファイルを変更する際は、この順序に注意してください。

次の表には、OpenText SASTプロパティの優先順序を示します。

順序	プロパティの指定	説明(Description)
1	`-D` オプション付きのコマンドライン	コマンドラインで指定されたプロパティの優先度は最高であり、任意のスキャンで指定できます。
2	OpenText SASTのクイックスキャン環境設定ファイル	**Note** クイックスキャンとスキャン精度レベルのいずれかを指定できます。したがって、これらのプロパティ設定の優先度はどちらも2番目です。 クイックスキャン環境設定ファイル(`fortify-sca-quickscan.properties`)で指定されたプロパティの優先度は2番目ですが、 `-quick` オプションを付けてクイックスキャンモードが有効になっている場合に限られます。
	OpenText SASTのスキャン精度プロパティファイル	スキャン精度プロパティファイルで指定されたプロパティの優先度は2番目ですが、 `-scan-precision` オプションを付けてスキャン精度が有効になっている場合に限られます。

順序	プロパティの指定	説明(Description)
3	OpenText SAST環境設定ファイル	OpenText SAST環境設定ファイル(`fortify-sca.properties`)で指定されたプロパティの優先度は最低です。すべてのスキャンでより永続的にプロパティ値が変更されるように、このファイルを編集します。

また、OpenText SASTは内部でデフォルト値が定義されている一部のプロパティに依存します。

1.29.2. fortify-sca.properties

次のセクションでは、 `fortify-sca.properties` ファイルで使用できるプロパティについて説明します。このプロパティファイルで使用できるその他のプロパティについては、fortify-sca-quickscan.propertiesを参照してください。各プロパティの説明には、値の型、デフォルト値、同等のコマンドラインオプション(該当する場合)、および例が含まれます。

このセクションでは、次のトピックについて説明します。

1.29.2.1. 変換と分析フェーズのプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、変換フェーズや分析(スキャン)フェーズに適用される一般的なプロパティです。

プロパティ名	説明(Description)
変換とスキャン	
`com.fortify.sca.BuildID`	ビルドのビルドIDを指定します。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** `-b`
`com.fortify.sca.CmdlineOptionsFileEncoding`	`@<filename>` で指定されたコマンドラインオプションファイルのエンコードを指定します(「その他のオプション」を参照)。たとえば、このプロパティを使用して、オプションファイルでUnicodeのファイルパスを指定できます。有効なエンコード名は `java.nio.charset.Charset` で定義されています。 このプロパティは `fortify-sca.properties` ファイルでのみ有効であり、 `fortify-sca-quickscan.properites` ファイルや `-D` オプションでは機能しません。 **値の型:** 文字列 **デフォルト:** JVMシステムのデフォルトエンコーディング **例:** `com.fortify.sca.CmdlineOptionsFileEncoding=UTF-8`

プロパティ名	説明(Description)
com.fortify.sca.DISabledLanguages	変換フェーズから除外する言語のコロン区切りリストを指定します。有効な言語の値は次のとおりです。 abap、actionscript、apex、cfml、cobol、configuration、cpp、dart、dotnet、golang、objc、php、python、ruby、swift、および vb。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** -disable-language
com.fortify.sca.EnabledLanguages	変換する言語のコロン区切りリストを指定します。有効な言語の値は次のとおりです。 abap、actionscript、apex、cfml、cobol、configuration、cpp、dart、dotnet、golang、objc、php、python、ruby、swift、および vb。 **値の型:** 文字列 **デフォルト:** com.fortify.sca.DISabledLanguages プロパティで明示的に除外しない限り、指定したソース内のすべての言語が変換されます。 **コマンドラインオプション:** -enable-language

プロパティ名	説明(Description)
com.fortify.sca.DisableCompilerName	trueに設定すると、OpenText SASTには、ソースファイルとして変換中に、ビルドツールと同じ名前(gradlewなど)を持つビルドスクリプトファイルが含められます。 **値の型:** ブール値 **デフォルト:** false **コマンドラインオプション:** -disable-compiler-resolution

プロパティ名	説明(Description)
com.fortify.sca.ProjectRoot	変換および分析フェーズで生成された中間ファイルを保存するディレクトリを指定します。OpenText SASTでは、このプロジェクトのルートディレクトリにある中間ファイルを広く活用します。場合によっては、このディレクトリをネットワークドライブではなくローカルストレージに配置すると、分析のパフォーマンスが向上します。 **値の型:** 文字列(パス) **デフォルト(Windows):** ${win32.LocalAppdata}/Fortify ${win32.LocalAppdata} は、Windowsのローカルアプリケーションデータシェルフォルダをポイントする変数です。 **デフォルト(Windows以外):** $home/.fortify **コマンドラインオプション:** -project-root **例:** com.fortify.sca.ProjectRoot=C:\Users*<username>*\AppData\Local\
変換	

プロパティ名	説明(Description)
com.fortify.sca.fileextensions.java com.fortify.sca.fileextensions.cs com.fortify.sca.fileextensions.js com.fortify.sca.fileextensions.py com.fortify.sca.fileextensions.rb com.fortify.sca.fileextensions.aspx com.fortify.sca.fileextensions.php これは部分的なリストです。完全なリストについては、プロパティファイルを参照してください。	ビルド統合を必要としない言語の特定のファイル名拡張子を変換する方法を指定します。有効な拡張子の種類は、ABAP、ACTIONSCRIPT、APEX、APEX_OBJECT、APEX_TRIGGER、ARCHIVE、ASPNET、ASP、ASPX、BITCODE、BSP、BYTECODE、CFML、COBOL、CSHARP、DART、DOCKERFILE、FLIGHT、GENERIC、GO、HCL、HOCON、HTML、INI、JAVA、JAVA_PROPERTIES、JAVASCRIPT、JINJA、JSON、JSP、JSPX、JUPYTER、KOTLIN、MSIL、MXML、OBJECT、PHP、PLSQL、PYTHON、RUBY、RUBY_ERB、SCALA、SWIFT、SWC、SWF、TLD、SQL、TSQL、TYPESCRIPT、VB、VB6、VBSCRIPT、VISUAL_FORCE、VUE、XML、および YAML です。 **値の型:** 文字列(有効な言語タイプ) **既定:** 完全なリストについては、fortify-sca.properties ファイルを参照してください。 **例:** com.fortify.sca.fileextensions.java=JAVA com.fortify.sca.fileextensions.cs=CSHARP com.fortify.sca.fileextensions.js=TYPESCRIPT com.fortify.sca.fileextensions.py=PYTHON

プロパティ名	説明(Description)

	`com.fortify.sca.fileextensions.swift=SWIFT`
	`com.fortify.sca.fileextensions.razor=ASPNET`
	`com.fortify.sca.fileextensions.php=PHP`
	`com.fortify.sca.fileextensions.tf=HCL`
	
	`oracle:<path_to_script>` の値を指定して、言語タイプをプログラムで指定することもできます。指定した拡張子と一致するファイル名のコマンドラインパラメータを1つ受け入れるスクリプトを指定します。このスクリプトは、有効なOpenText SASTファイルの種類(前のリストを参照)をstdoutに書き込み、戻り値0で終了する必要があります。ゼロ以外の戻りコードが返された場合、またはスクリプトが存在しない場合、このファイルは変換されず、ログファイルにOpenText SASTの警告が書き込まれます。
	
	例:
	`com.fortify.sca.fileextensions.jsp=oracle:<path_to_script>`

プロパティ名	説明(Description)

`com.fortify.sca.compilers.javac=com.fortify.sca.util.compilers.JavacCompiler`
 	
`com.fortify.sca.compilers.c++=com.fortify.sca.util.compilers.GppCompiler`
 	
`com.fortify.sca.compilers.make=com.fortify.sca.util.compilers.TouchlessCompiler`
 	
`com.fortify.sca.compilers.mvn=com.fortify.sca.util.compilers.MavenAdapter`
 	
これは部分的なリストです。完全なリストについては、プロパティファイルを参照してください。
	カスタム名のコンパイラを指定します。
	
	値の型: 文字列(コンパイラ)
	
	デフォルト: 完全なリストについては、`fortify-sca.properties`ファイルの「Compilers」セクションを参照してください。
	
	例:
	
	「my-gcc」がgccコンパイラであることをOpenText SASTに知らせるには、次のようにします。
	
	`com.fortify.sca.compilers.my-gcc=com.fortify.sca.util.compilers.GccCompiler`
	
	メモ:
	
	• コンパイラ名の先頭または末尾をアスタリスク(*)にして、0個以上の文字と一致させることができます。
	• clang/clang++の実行は、gcc/g++コマンド名ではサポートされていません。次のように指定できます。`com.fortify.sca.compilers.g++=com.fortify.sca.util.compilers.GppCompiler`

プロパティ名	説明(Description)
`com.fortify.sca.UseAntListener`	trueに設定すると、OpenText SASTによってコンパイラオプションに `com.fortify.dev.ant.SCAListener` が追加されます。 **値の型:** ブール値 **デフォルト:** `false`
`com.fortify.sca.exclude`	変換から除外するファイルを1つ以上指定します。複数のファイルはセミコロン(Windows)またはコロン(Windows以外)で区切ります。ファイル指定子の使い方について詳しくは、ファイルとディレクトリの指定を参照してください。 **値の型:** 文字列 **デフォルト:** 無効 **コマンドラインオプション:** `-exclude` **例:** `com.fortify.sca.exclude=file1.x;file2.x`

プロパティ名	説明(Description)
com.fortify.sca.InputFileEncoding	ソースファイルのエンコーディングタイプを指定します。OpenText SASTでは、異なる形式でエンコードされたソースファイルを含むプロジェクトをスキャンできます。マルチエンコーディングされたプロジェクトを使用するには、OpenText SASTで最初にソースコードファイルが読み込まれる際に、変換フェーズで `-encoding` オプションを指定する必要があります。OpenText SASTでは、このエンコーディングがビルドセッションで記憶され、FVDLファイルに反映されます。 通常、エンコーディングタイプを指定しないと、OpenText SASTではエンコーディングパラメータなしで `file.encoding` コンストラクタから `java.io.InputStreamReader` が使用されます。一部のケース(ActionScriptパーサの場合など)では、OpenText SASTのデフォルトは `UTF-8` です。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** `-encoding` **例:** `com.fortify.sca.InputFileEncoding=UTF-16`

プロパティ名	説明(Description)
com.fortify.sca.RegExecutable	Windowsプラットフォームでは、reg.exe システムユーティリティへのパスを指定します。Cygwin内からOpenText SASTを実行する場合でも、Cygwin構文ではなくWindows構文でパスを指定します。バックスラッシュをエスケープする場合は、バックスラッシュを追加します。 **値の型:** 文字列(パス) **デフォルト:** reg **例:** com.fortify.sca.RegExecutable=C:\\Windows\\System32\\reg.exe

プロパティ名	説明(Description)
`com.fortify.sca.xcode.TranslateAfterError`	xcodebuildサブプロセスがゼロ以外の終了コードで終了した場合にxcodebuildタッチレスアダプタが変換を続行するかどうかを指定します。falseに設定すると、xcodebuildでゼロ以外の終了コードが発生した後で変換が中止され、同じ終了コードでOpenText SASTタッチレスビルドが停止します。trueにすると、OpenText SASTタッチレスビルドによってxcodebuildが終了する前に識別されたビルドファイルの変換が実行され、(他のエラーが発生しない限り)終了コード0でOpenText SASTが終了します。 この設定に関係なく、ゼロ以外のコードでxcodebuildが終了した場合は、xcodebuildの終了コード、stdout、およびstderrがログファイルに書き込まれます。 **値の型:** ブール値 **デフォルト:** `false`
スキャン	
`com.fortify.sca.AddImpliedMethods`	trueに設定すると、OpenText SASTで、継承による実装が発生した場合に暗黙的なメソッドが生成されます。 **値の型:** ブール値 **デフォルト:** `true`

プロパティ名	説明(Description)
`com.fortify.sca.alias.Enable`	trueに設定すると、エイリアス分析が有効になります。 **値の型:** ブール値 **デフォルト:** `true`
`com.fortify.sca.analyzer.controlflow.EnableTimeOut`	Control Flow Analyzerのタイムアウトを有効にするかどうかを指定します。 **値の型:** ブール値 **デフォルト:** `true`
`com.fortify.sca.BinaryName`	スキャンするソースファイルのサブセットを指定します。ビルド時に名前付きバイナリにリンクされたソースファイルのみがスキャンに含められます。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** `-bin` または `-binary-name`

プロパティ名	説明(Description)
com.fortify.sca.DefaultAnalyzers	実行する分析タイプのカンマまたはコロン区切りリストを指定します。このプロパティの有効な値は、buffer、content、configuration、controlflow、dataflow、nullptr、semantic、および structural です。 **値の型:** 文字列 **デフォルト:** このプロパティはコメントアウトされ、すべての分析タイプがスキャンで使用されます。 **コマンドラインオプション:** -analyzers
com.fortify.sca.DisableFunctionPointers	trueに設定すると、スキャン時に関数ポインタが無効になります。 **値の型:** ブール値 **デフォルト:** false
com.fortify.sca.EnableAnalyzer	デフォルトのアナライザに加えて、スキャンに使用するアナライザのカンマまたはコロン区切りのリストを指定します。このプロパティの有効な値は、buffer、content、configuration、controlflow、dataflow、nullptr、semantic、および structural です。 **値の型:** 文字列 **デフォルト:** (なし)

プロパティ名	説明(Description)

`com.fortify.sca.EnableSubtraceFilterin g`	trueに設定した場合、問題が特定の問題のサブトレースである部分重複をフィルタで除外します。
	たとえば、エンジンがトレースに関して2つの同様の問題を見つけた場合は、次のようになります。
	`A → B → C → D`
	`B → C → D`
	2番目の問題は、1番目のサブトレースの重複として削除され、より長い問題だけが残されます。これは全体的に、より正確な問題です。
	
	値の型: ブール値
	
	デフォルト: `true`

`com.fortify.sca.ExitCodeLevel`	デフォルトの終了コードオプションを拡張します。終了コードとこのプロパティの有効な値については、終了コードを参照してください。
`com.fortify.sca.FilterFile`	スキャンのフィルタファイルのパスを指定します。詳しくは、フィルタファイルについてを参照してください。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** `-filter`

プロパティ名	説明(Description)
`com.fortify.sca.FilteredInstanceIDs`	フィルタファイルを使用して除外するIIDのカンマ区切りリストを指定します。 **値の型:** 文字列 **デフォルト:** (なし) **例:** `com.fortify.sca.FilteredInstanceIDs=CA4E1623A2424919B98EC19FCA279FFA,4418B3DC072647158B3758E6183C14CD`
`com.fortify.sca.FilteredRuleLanguages`	ルールを削除する言語のカンマ区切りリストまたはコロン区切りリストを指定します。有効な言語の値は次のとおりです。 `abap`、`actionscript`、`apex`、`cfml`、`cobol`、`configuration`、`cpp`、`dart`、`dotnet`、`golang`、`objc`、`php`、`python`、`ruby`、`swift`、および`vb`。 **値の型:** 文字列 **デフォルト:** (なし) **例:** `com.fortify.sca.FileredRuleLanguages=apex:php`
`com.fortify.sca.MaxPassthroughChainDepth`	関数呼び出しの入力パラメータと出力パラメータ間のテイントパスの長さを指定します。 **値の型:** 整数 **デフォルト:** `4`

プロパティ名	説明(Description)
com.fortify.sca.MultithreadedAnalysis	OpenText SASTを並行分析モードで実行するかどうかを指定します。 **値の型:** ブール値 **デフォルト:** true
com.fortify.sca.Phase0HigherOrder.Languages	高次分析を実行する言語のカンマ区切りリストを指定します。高次解析は、現代の動的言語で一般的に用いられる高次コードを通じて、データフローを追跡する能力を向上させます。有効な値は、python、swift、ruby、javascript、および typescript です。 **値の型:** 文字列 **デフォルト:** python,ruby,swift,javascript,typescript
com.fortify.sca.Phase0HigherOrder.Timeout.Hard	高次分析の合計時間(秒)を指定します。ハードタイムアウト制限に達すると、アナライザはすぐに終了します。 何らかの問題で分析時間が長くなりすぎる場合に備えて、このタイムアウト制限が推奨されています。固定ポイントのパスリミッタまたはソフトタイムアウトのいずれかが先に発生するように、ハードタイムアウトをソフトタイムアウトより約50%長く設定することをお勧めします。 **値の型:** 数値 **デフォルト:** 2700

プロパティ名	説明(Description)
`com.fortify.sca.PrecisionLevel`	スキャンの精度を指定します。スキャンの精度を下げると、実行速度が向上します。有効な値は、`1`、`2`、`3`、および`4`です。 **値の型:** 数値 **デフォルト:** (なし) **コマンドラインオプション:** `-scan-precision` | `-p`
`com.fortify.sca.ProjectTemplate`	スキャンに使用する問題テンプレートファイルを指定します。これにより、ローカルコンピュータのスキャンのみが影響を受けます。FPRをApplication Securityにアップロードする場合は、アプリケーションバージョンに割り当てられた問題テンプレートが使用されます。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** `-project-template` **例:** `com.fortify.sca.ProjectTemplate=test_issuetemplate.xml`

プロパティ名	説明(Description)
com.fortify.sca.QuickScanMode	trueに設定すると、OpenText SASTでクイックスキャンが実行されます。OpenText SASTでは、fortify-sca-quickscan.properties設定ファイルの代わりにfortify-sca.propertiesの設定が使用されます。 **値の型:** ブール値 **デフォルト:** (無効) **コマンドラインオプション:** -quick
com.fortify.sca.ScanPolicy	レポートされた脆弱性の優先順位を決めるためのスキャンポリシーを指定します(「分析へのスキャンポリシーの適用」を参照)。有効なスキャンポリシー値は、classic、security、およびdevopsです。 **値の型:** 文字列 **デフォルト:** security **コマンドラインオプション:** -scまたは-scan-policy

プロパティ名	説明(Description)
`com.fortify.sca.ThreadCount`	並行分析モードのスレッド数を指定します。リソース制約のために使用するスレッド数を減らす必要がある場合にのみ、このプロパティを追加します。スキャン時間の増加やスキャンに関する問題が発生した場合は、使用するスレッド数を減らして問題を解決できます。 **値の型:** 整数 **デフォルト:** (使用可能なプロセッサコア数)
`com.fortify.sca.TypeInferenceFunctionTimeout`	1つの関数の分析で型推論に使用できる時間(秒)。0に設定した場合、または指定しなかった場合は、無制限になります。 **値の型:** 倍精度 **デフォルト:** `60`
`com.fortify.sca.TypeInferenceLanguages`	型推論を使用する言語のカンマまたはコロン区切りのリスト。この設定により、動的に型付けされた言語の分析精度が向上します。 **値の型:** 文字列 **デフォルト:** `javascript,python,ruby,typescript`

プロパティ名	説明(Description)
`com.fortify.sca.TypeInferencePhase0Timeout`	フェーズ0 (プロシージャ間分析)で型推論に使用できる合計時間(秒)を指定します。0に設定した場合、または指定しなかった場合は、無制限になります。 **値の型:** 倍精度 **デフォルト:** `300`
`com.fortify.sca.UniversalBlacklist`	すべてのアナライザから隠す関数のコロン区切りリストを指定します。 **値の型:** 文字列 **デフォルト:** `.*yyparse.*`

1.29.2.2. 正規表現分析のプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、正規表現分析に適用されます。

プロパティ名	説明(Description)
`com.fortify.sca.regex.Enable`	trueに設定すると、正規表現分析が有効になります。 **値の型:** ブール値 **デフォルト:** `true`
`com.fortify.sca.regex.ExcludeBinaries`	trueに設定すると、正規表現分析からバイナリファイルが除外されます。 **値の型:** ブール値 **デフォルト:** `true`
`com.fortify.sca.regex.MaxSize`	正規表現分析でスキャンされるファイルの最大サイズ(メガバイト単位)を指定します。この最大ファイルサイズを超えるファイルは正規表現分析から除外されます。 **値の型:** 数値 **デフォルト:** `10`

参照情報

正規表現分析

1.29.2.3. LIMライセンスのプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、LIMでのライセンスに適用されます。

プロパティ名	説明(Description)
com.fortify.sca.lim.Url	LIMサーバのAPI URLを指定します。この値をテキストエディタで直接編集しないでください。この値を変更するには、コマンドラインオプションを使用します。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** -store-license-pool-credentials **例:** https://*<ip_address>*:*<port>*
com.fortify.sca.lim.PoolName	LIMライセンスプールの名前を指定します。この値をテキストエディタで直接編集しないでください。この値を変更するには、コマンドラインオプションを使用します。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** -store-license-pool-credentials

プロパティ名	説明(Description)
com.fortify.sca.lim.PoolPassword	LIMライセンスプールのパスワード(暗号化)を指定します。この値をテキストエディタで直接編集しないでください。この値を変更するには、コマンドラインオプションを使用します。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** -store-license-pool-credentials
com.fortify.sca.lim.ProxyUrl	LIMサーバへの接続に使用するプロキシサーバを指定します。 **値の型:** 文字列 **デフォルト:** (なし) **例:** http://proxy.example.com:8080 https://proxy.example.com **コマンドラインオプション:** -store-license-pool-credentials
com.fortify.sca.lim.ProxyUsername	LIMサーバに接続するためのプロキシ認証用の暗号化されたユーザ名を指定します。この値をテキストエディタで直接編集しないでください。この値を変更するには、コマンドラインオプションを使用します。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** -store-license-pool-credentials

プロパティ名	説明(Description)
com.fortify.sca.lim.ProxyPassword	LIMサーバに接続するためのプロキシ認証用の暗号化パスワードを指定します。この値をテキストエディタで直接編集しないでください。この値を変更するには、コマンドラインオプションを使用します。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** -store-license-pool-credentials
com.fortify.sca.lim.RequireTrustedSSLCert	trueに設定すると、信頼された証明書がない状態でLIMサーバに接続しようとしたときに失敗します。このプロパティをfalseに設定すると、信頼された証明書がない状態でLIMサーバに接続しようとしたときにメッセージが表示されます。 **値の型:** ブール値 **デフォルト:** true
com.fortify.sca.lim.WaitForInitialLicense	trueに設定し、LIMライセンスプールの資格情報が保存されている場合、OpenText SASTではLIMライセンスが使用可能になるのを待機してから変換またはスキャンが開始されます。このプロパティをfalseに設定すると、LIMライセンスを取得できない場合にOpenText SASTが中止します。 **値の型:** ブール値 **デフォルト:** true

LIMライセンスディレクティブ

1.29.2.4. ルールのプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、ルール(およびカスタムルール)とRulepackに適用されます。

プロパティ名	説明(Description)
com.fortify.sca.DefaultRulesDir	OpenTextによって提供される暗号化ルールファイルの検索時に使用するディレクトリを設定します。 **値の型:** 文字列(パス) **デフォルト:** ${com.fortify.Core}/config/rules
com.fortify.sca.RulesFile	カスタムのルールパックまたはディレクトリを指定します。ディレクトリを指定すると、そのディレクトリ内の .bin および .xml 拡張子を持つすべてのファイルが含められます。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** -rules
com.fortify.sca.CustomRulesDir	カスタムルールの検索時に使用するディレクトリを設定します。 **値の型:** 文字列(パス) **デフォルト:** ${com.fortify.Core}/config/customrules

プロパティ名	説明(Description)
com.fortify.sca.RulesFileExtensions	ルールファイルのファイル拡張子のリストを指定します。このリストに拡張子が含まれている *<sast_install_dir>*/Core/config/rules (または -rules オプションで指定されたディレクトリ)内のファイルが含められます。 .bin 拡張子は、このプロパティの値に関係なく常に含められます。このプロパティの区切り記号は、システムパスのセパレータです。 **値の型:** 文字列 **デフォルト:** .xml
com.fortify.sca.NoDefaultRules	trueに設定すると、デフォルトのルールパックのルールがロードされません。OpenText SASTでは、説明要素と言語ライブラリのルールパックが処理されますが、ルールは処理されません。 **値の型:** ブール値 **デフォルト:** (なし) **コマンドラインオプション:** -no-default-rules

プロパティ名	説明(Description)
com.fortify.sca.NoDefaultIssueRules	trueに設定すると、問題の直接の原因となるデフォルトのルールパックのルールが無効になります。OpenText SASTでは、関数の動作を特徴付けるルールは引き続きロードされます。これは、カスタムの問題ルールを作成するときに役立ちます。 **値の型:** ブール値 **デフォルト:** (なし) **コマンドラインオプション:** -no-default-issue-rules
com.fortify.sca.NoDefaultSourceRules	trueに設定すると、デフォルトのルールパックのソースルールが無効になります。これは、カスタムのソースルールを作成するときに役立ちます。 **Note** 特性化ソースルールは無効になりません。 **値の型:** ブール値 **デフォルト:** (なし) **コマンドラインオプション:** -no-default-source-rules

プロパティ名	説明(Description)
com.fortity.sca.NoDefaultSinkRules	trueに設定すると、デフォルトのルールパックのシンクルールが無効になります。これは、カスタムのシンクルールを作成するときに役立ちます。 **Note** 特性化シンクルールは無効になりません。 **値の型:** ブール値 **デフォルト:** (なし) **コマンドラインオプション:** -no-default-sink-rules

1.29.2.5. JavaおよびKotlinのプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、JavaおよびKotlinコードの変換に適用されます。

プロパティ名	説明(Description)
`com.fortify.sca.JavaClasspath`	JavaまたはKotlinソースコードの分析時に使用するクラスパスを指定します。複数のパスはセミコロン(Windows)またはコロン(Windows以外)で区切ります。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** `-cp` または `-classpath`
`com.fortify.sca.JdkVersion`	JavaまたはKotlin変換用のJavaソースコードのバージョンを指定します。 **値の型:** 文字列 **デフォルト:** `11` **コマンドラインオプション:** `-jdk` または `-source`
`com.fortify.sca.CustomJdkDir`	OpenText SASTインストール(`<sast_install_dir>/Core/bootcp/`)に含まれていないJDKバージョンを格納しているディレクトリを指定します。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** `-custom-jdk-dir`

プロパティ名	説明(Description)
com.fortify.sca.JavaSourcepath	スキャンに含まれていないがネームレゾリューションに使用されるJavaまたはKotlinソースファイルディレクトリのセミコロン(Windows)またはコロン(Windows以外)区切りリストを指定します。ソースパスはクラスパスに似ていますが、解決にはクラスファイルではなくソースファイルが使用されます。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** `-sourcepath`
com.fortify.sca.Appserver	JSPファイルを処理するアプリケーションサーバを指定します。有効な値は、`weblogic` または `websphere` です。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** `-appserver`
com.fortify.sca.AppserverHome	アプリケーションサーバのホームディレクトリを指定します。WebLogicの場合、これは `server/lib` を含むディレクトリへのパスです。WebSphereの場合、これは `JspBatchCompiler` スクリプトを含むディレクトリへのパスです。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** `-appserver-home`

プロパティ名	説明(Description)
`com.fortify.sca.AppserverVersion`	WebLogicまたはWebSphereアプリケーションサーバのバージョンを指定します。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** `-appserver-version`
`com.fortify.sca.JavaExtdirs`	WebLogicおよびWebSphereアプリケーションサーバのクラスパスに暗黙的に含めるディレクトリを指定します。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** `-extdirs`
`com.fortify.sca.JavaSourcepathSearch`	trueに設定すると、OpenText SASTで、ターゲットファイルリストによって参照されるJavaソースファイルのみが変換されます。それ以外の場合、OpenText SASTではソースパスに含まれるすべてのファイルが変換されます。 **値の型:** ブール値 **デフォルト:** `true`

プロパティ名	説明(Description)
`com.fortify.sca.DefaultJarsDirs`	一般的に使用されるJARファイルのディレクトリのセミコロンまたはコロン区切りリストを指定します。これらのディレクトリにあるJARファイルは、クラスパスオプション(`-cp`)の最後に追加されます。 **値の型:** 文字列 **デフォルト:** `default_jars`
`com.fortify.sca.DecompileBytecode`	trueに設定すると、Javaバイトコードが変換用に逆コンパイルされます。 **値の型:** ブール値 **デフォルト:** `false`
`com.fortify.sca.jsp.UseSecurityManager`	trueに設定すると、JSPパーサでJSPセキュリティマネージャが使用されます。 **値の型:** ブール値 **デフォルト:** `true`
`com.fortify.sca.jsp.DefaultEncoding`	JSPのエンコードを指定します。 **値の型:** 文字列(エンコーディング) **デフォルト:** `ISO-8859-1`

プロパティ名	説明(Description)
`com.fortify.sca.jsp.LegacyDataflow`	trueに設定すると、JSP関連のデータフローに対する追加のフィルタリングが有効になり、誤検出の量が減少します。 **値の型:** ブール値 **デフォルト:** `false` **コマンドラインオプション:** `-legacy-jsp-dataflow`
`com.fortify.sca.KotlinJvmDefault`	本体がKotlinインターフェイスに入っているメソッドの `DefaultImpls` クラスの生成を指定します。有効な値は次のとおりです。 • `disable` — 本体を持つメソッドが含まれている各インターフェイスに対して、 `DefaultImpls` クラスを生成するように指定します。 • `all` — インターフェイスに `DefaultImpls` の注釈が付く場合に、 `@JvmDefaultWithCompatibility` クラスを生成するように指定します。 • `all-compatibility` — インターフェイスに `DefaultImpls` の注釈が付かない限り、 `@JvmDefaultWithoutCompatibility` クラスを生成するように指定します。 **値の型:** 文字列 **デフォルト:** `disable`

プロパティ名	説明(Description)
com.fortify.sca.ShowUnresolvedSymbols	trueに設定すると、変換されたJavaソースファイルで参照されている未解決のタイプ、フィールド、および関数が、変換の最後に表示されます。 **値の型:** ブール値 **デフォルト:** false **コマンドラインオプション:** -show-unresolved-symbols

Java、Kotlin、およびJSPプロジェクトの分析

1.29.2.6. Visual StudioおよびMSBuildプロジェクトのプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、.NETプロジェクトおよびソリューションの変換に適用されます。

プロパティ名	説明(Description)

`WinForms.TransformDataBindings`
`WinForms.TransformMessageLoops`
`WinForms.TransformChangeNotificationPattern`
`WinForms.CollectionMutationMonitor.Label`
`WinForms.ExtractEventHandlers`
	さまざまな.NETオプションを設定します。
	値の型: ブール値および文字列
	デフォルトと例:
	`WinForms.TransformDataBindings=true`
	`WinForms.TransformMessageLoops=true`
	`WinForms.TransformChangeNotificationPattern=true`
	`WinForms.CollectionMutationMonitor.Label=WinFormsDataSource`
	`WinForms.ExtractEventHandlers=true`

`com.fortify.sca.ASPVirtualRoots.<virtual_path>`	使用する仮想ルートへのフルパスのセミコロン区切りリストを指定します。
	値の型: 文字列
	デフォルト: (なし)
	例:
	`com.fortify.sca.ASPVirtualRoots.Library=c:\\WebServer\\CustomerTwo\\Stuff`
	`com.fortify.sca.ASPVirtualRoots.Include=c:\\WebServer\\CustomerOne\\inc`

プロパティ名	説明(Description)
`com.fortify.sca.DisableASPExternalEntries`	trueに設定すると、スキャンのASP外部エントリが無効になります。 **値の型:** ブール値 **デフォルト:** `false`

Visual StudioおよびMSBuildプロジェクトの変換

1.29.2.7. JavaScriptおよびTypeScriptのプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、JavaScriptおよびTypeScriptコードの変換に適用されます。

プロパティ名	説明(Description)
`com.fortify.sca.EnableDOMModeling`	trueに設定すると、OpenText SASTで、変換フェーズ中にHTMLファイルによって生成されたDOMツリーをモデル化するJavaScriptコードが生成され、DOM関連の問題(クロスサイトスクリプティング(XSS)の問題など)が特定されます。変換するコードに、埋め込みまたは参照先のJavaScriptコードを含むHTMLファイルが含まれている場合は、このプロパティを有効にしてください。 このプロパティを有効にすると、変換時間が増える可能性があります。 **値の型:** ブール値 **デフォルト:** `false`
`com.fortify.sca.DOMModeling.tags`	`com.fortify.sca.EnableDOMModeling` プロパティをtrueに設定した場合は、OpenText SASTのDOMモデリングに含める追加のHTMLタグ名をカンマ区切りで指定できます。 **値の型:** 文字列 **デフォルト:** `body`、`button`、`div`、`form`、`iframe`、`input`、`head`、`html`、および `p`。 **例:** `com.fortify.sca.DOMModeling.tags=ul,li`

プロパティ名	説明(Description)
`com.fortify.sca.JavaScript.src.domain.whitelist`	OpenText SASTで参照先のJavaScriptファイルをスキャン用にダウンロードできる信頼されたドメイン名を指定します。URLの区切り記号として縦棒を使用します。 **値の型:** 文字列 **デフォルト:** (なし) **例:** `com.fortify.sca.JavaScript.src.domain.whitelist=http://www.xyz.com|http://www.123.org`
`com.fortify.sca.DisableJavascriptExtraction`	trueに設定すると、JSP、JSPX、PHP、およびHTMLファイルに埋め込まれたJavaScriptコードは抽出されず、スキャンされません。 **値の型:** ブール値 **デフォルト:** `false`
`com.fortify.sca.EnableTranslationMinifiedJS`	trueに設定すると、圧縮されたJavaScriptファイルの変換が有効になります。 **値の型:** ブール値 **デフォルト:** `false`

プロパティ名	説明(Description)
`com.fortify.sca.skip.libraries.ES6` `com.fortify.sca.skip.libraries.jQuery` `com.fortify.sca.skip.libraries.javascript` `com.fortify.sca.skip.libraries.typescript`	変換されないJavaScriptまたはTypeScript技術ライブラリファイルのカンマまたはコロン区切りリストを指定します。ファイル名には正規表現を使用できます。`'(-\d\.\d\.\d)?'` プロパティ値に含まれる各ファイル名の `.min.js` または `.js` の前に、正規表現 `com.fortify.sca.skip.libraries.jQuery` が自動的に挿入されます。 **値の型:** 文字列 **デフォルト:** • ES6: `es6-shim.min.js,system-polyfills.js,shims_for_IE.js` • jQuery: `jquery.js,jquery.min.js, jquery-migrate.js,jquery-migrate.min.js, jquery-ui.js,jquery-ui.min.js, jquery.mobile.js,jquery.mobile.min.js, jquery.color.js,jquery.color.min.js, jquery.color.svg-names.js, jquery.color.svg-names.min.js, jquery.color.plus-names.js, jquery.color.plus-names.min.js, jquery.tools.min.js`  • javascript: `bootstrap.js,bootstrap.min.js,typescript.js,typescriptServices.js` • typescript: `typescript.d.ts,typescriptServices.d.ts`

プロパティ名	説明(Description)
com.fortify.sca.follow.imports	trueに設定すると、importステートメントで組み込まれたファイルが変換に含められます。 **値の型:** ブール値 **デフォルト:** true
com.fortify.sca.exclude.node.modules	trueに設定すると、node_modulesディレクトリ内のファイルは分析フェーズから除外されます。 **値の型:** ブール値 **デフォルト:** true
com.fortify.sca.exclude.unimported.node.modules	node_modulesディレクトリ内のソースコードを除外するかどうかを指定します。trueに設定すると、インポートされたnode_modulesのみが変換に含められます。 このプロパティは、com.fortify.sca.exclude.node.modulesプロパティの値がfalseに設定されている場合にのみ適用されます。 **値の型:** ブール値 **デフォルト:** true

JavaScriptおよびTypeScriptコードの変換

1.29.2.8. Pythonのプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、Pythonコードの変換に適用されます。

プロパティ名	説明(Description)
com.fortify.sca.PythonPath	追加のインポートディレクトリのセミコロン区切り(Windows)またはコロン区切り(Windows以外)リストを指定します。OpenText SASTでは、Pythonランタイムシステムでインポートファイルの検索に使用されるPYTHONPATH環境変数は考慮されません。このプロパティを使用して、追加のインポートディレクトリを指定します。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** -python-path
com.fortify.sca.PythonVersion	スキャンするPythonソースコードのバージョンを指定します。有効な値は、2 および 3 です。 **値の型:** 数値 **デフォルト:** 3 **コマンドラインオプション:** -python-version

プロパティ名	説明(Description)
`com.fortify.sca.PythonNoAutoRootCalculation`	trueに設定すると、モジュールとパッケージのインポートに使用するすべてのプロジェクトファイルに共通のルートディレクトリの自動計算が無効になります。詳しくは、「インポート済みのモジュールとパッケージを含める」を参照してください。 **値の型:** ブール値 **デフォルト:** `false` **コマンドラインオプション:** `-python-no-auto-root-calculation`
`com.fortify.sca.DjangoTemplateDirs`	Djangoテンプレートのディレクトリのセミコロン区切り(Windows)またはコロン区切り(Windows以外)リストを指定します。OpenText SASTでは、Djangoの `settings.py` ファイルのTEMPLATE_DIRS設定は使用されません。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** `-django-template-dirs`
`com.fortify.sca.DjangoDisableAutodiscover`	OpenText SASTでDjangoテンプレートを自動検出しないことを指定します。 **値の型:** ブール値 **デフォルト:** (なし) **コマンドラインオプション:** `-django-disable-autodiscover`

プロパティ名	説明(Description)
com.fortify.sca.JinjaTemplateDirs	Jinja2テンプレートのディレクトリのセミコロン区切り(Windows)またはコロン区切り(Windows以外)リストを指定します。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** -jinja-template-dirs
com.fortify.sca.DisableTemplateAutodiscover	OpenText SASTでDjangoまたはJinja2テンプレートを自動検出しないことを指定します。 **値の型:** ブール値 **デフォルト:** (なし) **コマンドラインオプション:** -disable-template-autodiscover

Pythonコードの変換

1.29.2.9. Goのプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、Goコードの変換に適用されます。

プロパティ名	説明(Description)
`com.fortify.sca.gotags`	Goプロジェクトのカスタムビルドタグを指定します。これは、`go` コマンドの `-tags` オプションに相当します。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:-** -gotags
`com.fortify.sca.GOPATH`	プロジェクト/ワークスペースのルートディレクトリを指定します。 **値の型:** 文字列 **デフォルト:** (GOPATHシステム環境変数)
`com.fortify.sca.GOROOT`	Goインストールの場所を指定します。 **値の型:** 文字列 **デフォルト:** (GOROOTシステム環境変数)
`com.fortify.sca.GOPROXY`	1つ以上のプロキシURLをカンマ区切りで指定します。`direct` または `off` を指定することもできます。 **値の型:** 文字列 **デフォルト:** (GOPROXYシステム環境変数)

参照情報

Goコードの変換

1.29.2.10. Rubyのプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、Rubyコードの変換に適用されます。

プロパティ名	説明(Description)
`com.fortify.sca.RubyLibraryPaths`	Rubyライブラリを含むディレクトリへのパスを1つ以上指定します。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** `-ruby-path`
`com.fortify.sca.RubyGemPaths`	RubyGemsの場所へのパスを1つ以上指定します。プロジェクトにスキャンするgemが関連付けられている場合は、この値を設定します。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** `-rubygem-path`

Rubyコードの変換

1.29.2.11. COBOLのプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、COBOLコードの変換に適用されます。

プロパティ名	説明(Description)
com.fortify.sca.CobolCopyDirs	OpenText SASTでコピーブックファイルを検索する、1つ以上のディレクトリをセミコロン区切りまたはコロン区切りで指定します。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** -copydirs
com.fortify.sca.CobolDialect	COBOLの方言を指定します。方言の有効な値は、 COBOL390 または MICROFOCUS です。方言の値では、大文字と小文字を区別しません。 **値の型:** 文字列 **デフォルト:** COBOL390 **コマンドラインオプション:** -dialect
com.fortify.sca.CobolCheckerDirectives	1つ以上のCOBOLチェッカディレクティブをセミコロン区切りで指定します。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** -checker-directives

プロパティ名	説明(Description)
com.fortify.sca.CobolLegacy	trueに設定すると、レガシーCOBOL変換が有効になります。 **値の型:** ブール値 **デフォルト:** false **コマンドラインオプション:** -cobol-legacy
com.fortify.sca.CobolFixedFormat	trueに設定すると、固定形式のCOBOLが指定され、すべてのコード行のカラム8～72のソースコードのみを検索するようにOpenText SASTに指示されます(レガシーCOBOL変換のみ)。 **値の型:** ブール値 **デフォルト:** false **コマンドラインオプション:** -fixed-format
com.fortify.sca.CobolCopyExtensions	1つ以上のコピーブックファイル拡張子をセミコロン区切りまたはコロン区切りで指定します(レガシーCOBOL変換のみ)。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** -copy-extensions

COBOLコードの変換

1.29.2.12. PHPのプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、PHPコードの変換に適用されます。

プロパティ名	説明(Description)
`com.fortify.sca.PHPVersion`	PHPのバージョンを指定します。有効なバージョンのリストについては、「サポートされる言語」を参照してください。 **値の型:** 文字列 **デフォルト:** `8.2` **コマンドラインオプション:** `-php-version`
`com.fortify.sca.PHPSourceRoot`	PHPのソースルートを指定します。 **値の型:** ブール値 **デフォルト:** (なし) **コマンドラインオプション:** `-php-source-root`

PHPコードの変換

1.29.2.13. ABAPのプロパティ

次の表に示すプロパティは、ABAPコードの変換に適用されます。

プロパティ名	説明(Description)
com.fortify.sca.AbapDebug	trueに設定すると、OpenText SASTでメッセージをデバッグするためにABAPステートメントが追加されます。 **値の型:** ブール値 **デフォルト:** (なし)
com.fortify.sca.AbapIncludes	OpenText SASTでABAPのINCLUDEディレクティブが検出されたときに、名前付きディレクトリが検索されます。 **値の型:** 文字列(パス) **デフォルト:** (なし)

1.29.2.14. FlexおよびActionScriptのプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、FlexおよびActionScriptコードの変換に適用されます。

プロパティ名	説明(Description)
com.fortify.sca.FlexLibraries	「リンク先」のライブラリをセミコロン(Windows)またはコロン(Windows以外)で区切って指定します。このリストには、flex.swc、framework.swc、およびplayerglobal.swcを含める必要があります(これらは通常、Flex SDKルートのframeworks/libsディレクトリにあります)。このプロパティは、主にActionScriptを解決するために使用します。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** -flex-libraries
com.fortify.sca.FlexSdkRoot	有効なFlex SDKのルートの場所を指定します。このフォルダには、flex-config.xmlファイルを含むフレームワークフォルダが含まれている必要があります。binフォルダも含まれている必要があります。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** -flex-sdk-root

プロパティ名	説明(Description)
com.fortify.sca.FlexSourceRoots	Flexプロジェクトの追加のソースディレクトリを指定します。複数のディレクトリはセミコロン(Windows)またはコロン(Windows以外)で区切ります。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** -flex-source-root

1.29.2.15. ColdFusion (CFML)のプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、CFMLコードの変換に適用されます。

プロパティ名	説明(Description)
com.fortify.sca.CfmlUndefinedVariablesAreTainted	trueに設定すると、OpenText SASTでCFMLページ内の未定義の変数が汚染されたものとして処理されます。これは、register-globals-style脆弱性を監視するDataflow Analyzerに対するヒントとして機能します。ただし、このプロパティを有効にすると、インクルードページ内の変数がそれ以前に発生したインクルードページ内の汚染された値に初期化されるデータフローの検出に支障をきたします。 **値の型:** ブール値 **デフォルト:** false
com.fortify.sca.CaseInsensitiveFiles	trueに設定すると、大文字と小文字を区別しないファイルシステムを使用して開発され、大文字と小文字が区別されるファイルシステム上でスキャンされたアプリケーションでは、CFMLファイルの大文字と小文字が区別されなくなります。 **値の型:** ブール値 **デフォルト:** (無効)
com.fortify.sca.SourceBaseDir	ColdFusionプロジェクトのベースディレクトリを指定します。 **値の型:** 文字列(パス) **デフォルト:** (なし) **コマンドラインオプション:** -source-base-dir

ColdFusionコードの変換

1.29.2.16. SQLのプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、SQLコードの変換に適用されます。

プロパティ名	説明(Description)
`com.fortify.sca.SqlLanguage`	SQL言語のバリアントを指定します。有効なSQL言語タイプの値は、`PLSQL` (Oracle PL/SQLの場合)および `TSQL` (Microsoft T-SQLの場合)です。 **値の型:** 文字列 **デフォルト:** `TSQL` **コマンドラインオプション:** `-sql-language`

SQLの変換

1.29.2.17. 出力のプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、分析出力に適用されます。

プロパティ名	説明(Description)
com.fortify.sca.ResultsFile	結果が書き込まれるファイル。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** -f **例:** com.fortify.sca.ResultsFile=MyResults.fpr
com.fortify.sca.Renderer	出力形式を制御します。有効な値は、fpr、fvdl、text、および auto です。auto のデフォルトでは、-f オプションで指定されたファイルの拡張子に基づいて出力形式が選択されます。 **値の型:** 文字列 **デフォルト:** auto **コマンドラインオプション:** -format
com.fortify.sca.OutputAppend	trueに設定すると、OpenText SASTで既存の結果ファイルに結果が追加されます。 **値の型:** ブール値 **デフォルト:** false **コマンドラインオプション:** -append

プロパティ名	説明(Description)
com.fortify.sca.ResultsAsAvailable	trueに設定すると、OpenText SASTで使用可能になった結果が出力されます。これは、(出力ファイルを指定するための) -f オプションを指定せずに、stdoutに出力する場合に役立ちます。 **値の型:** ブール値 **デフォルト:** false
com.fortify.sca.BuildLabel	スキャンされたプロジェクトのラベルを指定します。OpenText SASTでは、このラベルは使用されませんが、結果に含められます。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** -build-label
com.fortify.sca.BuildProject	スキャンされたプロジェクトの名前を指定します。OpenText SASTでは、この名前は使用されませんが、結果に含められます。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** -build-project

プロパティ名	説明(Description)
com.fortify.sca.BuildVersion	スキャンされたプロジェクトのバージョンを指定します。OpenText SASTでは、このバージョン番号は使用されませんが、結果に含められます。 **値の型:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** -build-version
com.fortify.sca.MachineOutputMode	情報をインタラクティブに出力せずに、スクリプトまたはOpenText SASTツールで使用できる形式で出力します。スキャンの進行状況を1行で表示せずに、コンソールの前の行の下に新しい行を出力して、更新された進行状況を表示します。 **値の型:** ブール値 **デフォルト:** (無効) **コマンドラインオプション:** -machine-output
com.fortify.sca.SnippetContextLines	問題の周囲に表示するコードの行数を設定します。スニペットでは常に、エラーが発生した行の両側にある2行のコードが含められます。デフォルトでは、5行のコードが表示されます。 **値の型:** 数値 **デフォルト:** 2

プロパティ名	説明(Description)
`com.fortify.sca.FVDLDisableDescriptions`	trueに設定すると、分析結果ファイル(FVDL)からOpenText Application Security Contentの説明が除外されます。 **値の型:** ブール値 **デフォルト:** `false` **コマンドラインオプション:** `-fvdl-no-descriptions`
`com.fortify.sca.FVDLDisableEngineData`	trueに設定すると、分析結果ファイル(FVDL)からエンジンデータが除外されます。 **値の型:** ブール値 **デフォルト:** `false` **コマンドラインオプション:** `-fvdl-no-enginedata`
`com.fortify.sca.FVDLDisableLabelEvidence`	trueに設定すると、分析結果ファイル(FVDL)からラベル証拠が除外されます。 **値の型:** ブール値 **デフォルト:** `false`

プロパティ名	説明(Description)
`com.fortify.sca.FVDLDisableProgramData`	trueに設定すると、分析結果ファイル(FVDL)から `ProgramData` セクションが除外されます。 **値の型:** ブール値 **デフォルト:** `false` **コマンドラインオプション:** `-fvdl-no-progdata`
`com.fortify.sca.FVDLDisableSnippets`	trueに設定すると、分析結果ファイル(FVDL)からコードスニペットが除外されます。 **値の型:** ブール値 **デフォルト:** `false` **コマンドラインオプション:** `-fvdl-no-snippets`
`com.fortify.sca.FVDLStylesheet`	分析結果のスタイルシートの場所を指定します。 **値の型:** 文字列(パス) **デフォルト:** `${com.fortify.Core}/resources/sca/fvdl2html.xsl`

1.29.2.18. モバイルビルドセッション(MBS)のプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、MBSファイルに適用されます。

プロパティ名	説明(Description)
`com.fortify.sca.MobileBuildSessions`	falseに設定すると、OpenText SASTはソースファイルをビルドセッションディレクトリにコピーしません。 **値の型:** ブール値 **デフォルト:** `true`
`com.fortify.sca.ExtractMobileInfo`	trueに設定すると、OpenText SASTでモバイルビルドセッションからビルドIDとOpenText SASTのバージョン番号が抽出されます。 **Note** OpenText SASTでは、このプロパティを使用してもモバイルビルドは抽出されません。 **値の型:** ブール値 **デフォルト:** `false`

モバイルビルドセッション

1.29.2.19. プロキシプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、プロキシ設定に適用されます。

プロパティ名	説明(Description)
`com.fortify.sca.https.proxyHost`	プロキシホスト名を指定します。 **値の型:** 文字列 **デフォルト:** (なし)
`com.fortify.sca.https.proxyPort`	プロキシポート番号を指定します。 **値の型:** 数値 **デフォルト:** (なし)

1.29.2.20. ログプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、ログファイルに適用されます。

プロパティ名	説明(Description)
com.fortify.sca.LogFile	デフォルトのログファイルの名前と場所を指定します。 **値の型:** 文字列(パス) **デフォルト:** ${com.fortify.sca.ProjectRoot}/log/sca.log および ${com.fortify.sca.ProjectRoot}/log/sca_FortifySupport.log **コマンドラインオプション:** -logfile
com.fortify.sca.LogLevel	両方のログファイルの最小ログレベルを指定します。有効な値は、DEBUG、INFO、WARN、ERROR、および FATAL です。詳しくは、ログファイルへのアクセスおよびログファイルの設定を参照してください。 **値の型:** 文字列 **デフォルト:** INFO
com.fortify.sca.ClobberLogFile	trueに設定すると、OpenText SASTで sourceanalyzerを実行するたびにログファイルが上書きされます。 **値の型:** ブール値 **デフォルト:** false **コマンドラインオプション:** -clobber-log

プロパティ名	説明(Description)
`com.fortify.sca.PrintPerformanceDataAfterScan`	trueに設定すると、OpenText SASTでスキャンの完了後にパフォーマンス関連のデータがOpenText SAST Supportログファイルに書き込まれます。デバッグモードでは、この値は自動的にtrueに設定されます。 **値の型:** ブール値 **デフォルト:** `false`

ログファイルの設定

1.29.2.21. デバッグのプロパティ

次の表に示す `fortify-sca.properties` ファイルのプロパティは、デバッグ設定に適用されます。

プロパティ名	説明(Description)
`com.fortify.sca.Debug`	OpenText SAST Supportログファイルにデバッグ情報を含めます。この情報は、カスタマサポートによるトラブルシューティングにのみ役立ちます。 **値の型:** ブール値 **デフォルト:** `false` **コマンドラインオプション:** `-debug`
`com.fortify.sca.DebugVerbose`	これは `com.fortify.sca.Debug` プロパティと同じですが、特に解析エラーに関する詳細が含まれています。 **値の型:** ブール値 **デフォルト:** (無効) **コマンドラインオプション:** `-debug-verbose`
`com.fortify.sca.Verbose`	trueに設定すると、OpenText SAST Supportログファイルに詳細メッセージが含められます。 **値の型:** ブール値 **デフォルト:** `false` **コマンドラインオプション:** `-verbose`

プロパティ名	説明(Description)
`com.fortify.sca.DebugTrackMem`	trueに設定すると、追加のパフォーマンス情報がOpenText SAST Supportログに書き込まれます。 **値の型:** ブール値 **デフォルト:** (無効) **コマンドラインオプション:** `-debug-mem`
`com.fortify.sca.CollectPerformanceData`	trueに設定すると、パフォーマンスを追跡する追加のタイマが有効になります。 **値の型:** ブール値 **デフォルト:** (無効)
`com.fortify.sca.Quiet`	trueに設定すると、コマンドラインの進行状況情報が無効になります。 **値の型:** ブール値 **デフォルト:** `false` **コマンドラインオプション:** `-quiet`
`com.fortify.sca.MonitorSca`	trueに設定すると、OpenText SASTのメモリ使用量が監視され、JVMガーベッジコレクションが過剰に発生したときに警告が表示されます。 **値の型:** ブール値 **デフォルト:** `true`

1.29.3. fortify-sca-quickscan.properties

OpenText SASTでは、クイックスキャンと呼ばれるより詳細なスキャンが提供されています。このオプションでは、 `fortify-sca-quickscan.properties` ファイルのプロパティ値を使用して、プロジェクトがクイックスキャンモードでスキャンされます。デフォルトでは、クイックスキャンによって分析の深さが減少し、Quick Viewフィルタセットが適用されます。Quick Viewフィルタセットでは、優先度の高い重大な問題のみが提供されます。

Note

このファイルのプロパティは、スキャンのコマンドラインで `-quick` オプションを指定した場合にのみ使用されます。

次の表には、2つのデフォルト値セット(クイックスキャンのデフォルト値と通常スキャンのデフォルト値)を示します。デフォルト値が1つのみ表示されている場合は、通常スキャンとクイックスキャンの両方の値が同じです。

プロパティ名	説明(Description)
`com.fortify.sca.CtrlflowMaxFunctionTime`	単一の関数に制御フロー分析の時間制限(ミリ秒)を設定します。 **値の型:** 整数 **クイックスキャンのデフォルト:** `30000` **デフォルト:** `600000`
`com.fortify.sca.DisableAnalyzers`	スキャン時に無効にするアナライザのカンマ区切りリストまたはコロン区切りリストを指定します。有効なアナライザ名は、`buffer`、`content`、`configuration`、`controlflow`、`dataflow`、`nullptr`、`semantic`、および`structural`です。 **値の型:** 文字列 **クイックスキャンのデフォルト:** `controlflow:buffer` **デフォルト:** (なし)

プロパティ名	説明(Description)
com.fortify.sca.FilterSet	使用するフィルタセットを指定します。このプロパティを問題テンプレートとともに使用すると、スキャン後ではなくスキャン時にフィルタ処理できます。使用するフィルタセットを含む問題テンプレートを指定するには、「変換と分析フェーズのプロパティ」の「com.fortify.sca.ProjectTemplate」を参照してください。 このプロパティをQuick Viewに設定すると、大きな影響を受ける可能性があり、発生する可能性が高いルールと、大きな影響を受ける可能性があるが、発生する可能性は低いルールが実行されます。フィルタ処理された問題はFPRに書き込まれないため、FPRのサイズを削減できます。フィルタセットの詳細については、『*OpenText™ Fortify Audit Workbenchユーザガイド*』を参照してください。 **値の型:** 文字列 **クイックスキャンのデフォルト:** Quick View **デフォルト:** (なし)

プロパティ名	説明(Description)
`com.fortify.sca.FPRDisableMetatable`	Fortify Audit Workbenchで関数ビューの情報を含むメタテーブルの作成を無効にします。このメタテーブルでは、ソースウィンドウで変数を右クリックして宣言を表示できます。C/C++のスキャンに非常に長い時間がかかる場合は、このプロパティをtrueに設定すれば、スキャン時間が数時間短縮される可能性があります。 **値の型:** ブール値 **クイックスキャンのデフォルト:** `true` **デフォルト:** `false` **コマンドラインオプション:** `-disable-metatable`
`com.fortify.sca.FPRDisableSourceBundling`	FPRファイルにソースコードが含まれることを無効にします。スキャン時にOpenText SASTでマーク付きのソースコードファイルが生成されないようにします。クイックスキャンの結果として生成されるFPRファイルをApplication Securityにアップロードする場合は、このプロパティを `false` に設定する必要があります。 **値の型:** ブール値 **クイックスキャンのデフォルト:** `true` **デフォルト:** `false` **コマンドラインオプション:** `-disable-source-bundling`

プロパティ名	説明(Description)
`com.fortify.sca.NullPtrMaxFunctionTime`	単一の関数にNullポインタ分析の時間制限(ミリ秒)を設定します。標準のデフォルトは5分間です。この値を短い制限に設定すると、スキャン時間全体が短縮されます。 **値の型:** 整数 **クイックスキャンのデフォルト:** `10000` **デフォルト:** `300000`
`com.fortify.sca.TrackPaths`	制御フロー分析のパストラッキングを無効にします。パスパストラッキングでは、問題に関するより詳細なレポートが提供されますが、より長いスキャン時間が必要です。これをJSPでのみ無効にするには、`NoJSP`に設定します。すべての関数を無効にするには、`None`を指定します。 **値の型:** 文字列 **クイックスキャンのデフォルト:** (なし) **デフォルト:** `NoJSP`
`com.fortify.sca.limiters.ConstraintPredicateSize`	Buffer Analyzerで複雑な計算にサイズ制限を指定します。スキャン時間を改善するためにBuffer Analyzerで指定されたサイズ値より大きい計算をスキップします。 **値の型:** 整数 **クイックスキャンのデフォルト:** `10000` **デフォルト:** `500000`

プロパティ名	説明(Description)
com.fortify.sca.limiters.MaxChainDepth	Dataflow Analyzerでデータが追跡される呼び出しの最大深さを制御します。この値を大きくしてデータフロー分析の範囲を拡大すると、スキャン時間が長くなります。 **Note** 呼び出しの深さは、プログラムのエントリポイント(main() など)からの呼び出しの深さではなく、テイントソースとシンク間にあるデータフローパス上の呼び出しの最大深さを示します。 **値の型:** 整数 **クイックスキャンのデフォルト:** 3 **デフォルト:** 5
com.fortify.sca.limiters.MaxFunctionVisits	テイント伝播アナライザから関数にアクセスされる回数を設定します。 **値の型:** 整数 **クイックスキャンのデフォルト:** 5 **デフォルト:** 50

プロパティ名	説明(Description)
com.fortify.sca.limiters.MaxPaths	単一のデータフロー脆弱性についてレポートするパスの最大数を制御します。この値を変更しても、検出される結果は変更されません。個々の結果で表示されるデータフローパスの数のみが表示されます。 **Note** このプロパティを 5 より大きい値に設定することは、スキャン時間が長くなる可能性があるため、推奨されていません。 **値の型:** 整数 **クイックスキャンのデフォルト:** 1 **デフォルト:** 5
com.fortify.sca.limiters.MaxTaintDefForVar	Dataflow Analyzerに複雑さの制限を設定します。データフローでは、この複雑さメトリックを特定の精度レベルで超える関数の分析精度が徐々に減少します。この値は、変数チェーンに対して追跡されるテイントの量を制御します。 **値の型:** 整数 **クイックスキャンのデフォルト:** 250 **デフォルト:** 1000

プロパティ名	説明(Description)
`com.fortify.sca.limiters.MaxTaintDefForVarAbort`	関数の複雑さにハード制限を設定します。関数の複雑性が最小精度レベルでこの制限を超える場合は、アナライザで関数の分析がスキップされます。 **値の型:** 整数 **クイックスキャンのデフォルト:** `500` **デフォルト:** `4000`

1.29.4. fortify-rules.properties

このトピックでは、 `fortify-rules.properties` ファイルで使用できるプロパティについて説明します。

結果の改善

これらのプロパティを使用して、新しいルールセットの有効化、ルールのフィルタリング、またはOpenText DASTによる結果の相関関係の有効化など、スキャン結果の動作を変更します。

プロパティ名	説明(Description)
`com.fortify.sca.rules.EnableRuleComments`	trueに設定すると、`// FortifyRemove()` コメントを使用して、結果に問題が表示されるのを阻止できます。詳細については、「FortifyRemoveコメントを使ったフィルタリング」を参照してください。 **値の型:** ブール値 **デフォルト:** `true`
`com.fortify.sca.rules.IsLibrary`	trueに設定すると、すべてのパブリック関数変数にWEB、XSS、およびPRIVATEテイントを追加する新しいエントリポイントルールがコードで有効になります(特定の除外が適用されます)。(現在のところ、JavaおよびJVM言語のみ適用されます) **値の型:** ブール値 **デフォルト:** `false`
`com.fortify.sca.rules.enablePQCRules`	trueに設定すると、Post-Cryptographicの脅威に関連する問題を特定するルールが有効になります。サポートされている言語やライブラリなど、詳細については、セキュリティコンテンツの更新とドキュメントを参照してください。 **値の型:** ブール値 **デフォルト:** `false`

DASTの相関関係と検証

プロパティ名	説明(Description)
com.fortify.sca.rules.enable_wi_correlation	trueに設定し、サポートされているフレームワークを持つアプリケーションをOpenText SASTでスキャンすると、OpenText™ Dynamic Application Security Testingにインポートして結果を改善できる結果ファイルが生成されます。 **値の型:** ブール値 **デフォルト:** false

Google Cloud Functionの統合

Google Cloud Functionsをスキャンすると、JSONまたはYAMLのクラウドビルド設定ファイルを提供するか、以下の表のプロパティを設定して結果を最適化します。

プロパティ名	説明(Description)
com.fortify.sca.rules.GCPFunctionName	JSON/YAMLのCloud Build設定ファイルが存在しない場合に呼び出されるサーバレス関数の名前。 **値の型:** 文字列 **デフォルト:** (なし)
com.fortify.sca.rules.GCPHttpTrigger	trueに設定した場合、スキャンされるクラウド関数はHTTPトリガです。 **値の型:** ブール値 **デフォルト:** false

正規表現をカスタマイズするプロパティ

コード内の脆弱性を特定するために多くの手法が使われますが、コード内の識別子を見つけるために正規表現に頼るルールもあり、これらは多くの場合、プロパティによって設定

できます。次の表は、ルールで使用される正規表現を変更するために使用できるプロパティのリストを示しています。

正規表現とシェル構文の衝突を防ぐため、コマンドラインに直接設定するのではなく、`fortify-rules.properties` ファイル内でこれらの設定を行うことが勧められています。

プロパティ名	説明(Description)
com.fortify.sca.rules.password_regex.global	言語固有のルールプロパティが設定されている場合を除き、すべての言語のパスワード識別子に一致する正規表現。 **値の型:** 文字列 **デフォルト:** (?i)(s|_)?(user|usr|member|admin|guest|login|default| new|current|old|client|server|proxy|sql server| my|mysql|mongo|mongodb|db|database|ldap|smtp| email|email(_)?smtp)?(_|\.)?(pass(wd|word|phrase)|secret)
com.fortify.sca.rules.password_regex.abap	ABAPコードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)

プロパティ名	説明(Description)
com.fortify.sca.rules.password_regex.actionscript	ActionScriptコードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)
com.fortify.sca.rules.password_regex.apex	Salesforce Alesコードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)
com.fortify.sca.rules.password_regex.cfml	ColdFusion (CFML)コードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (なし)

プロパティ名	説明(Description)
com.fortify.sca.rules.password_regex.cobol	COBOLコードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)
com.fortify.sca.rules.password_regex.config	XMLのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。正規表現修飾子は使用しないでください。この値では、大文字と小文字が区別されません。 **値の型:** 文字列 **デフォルト:** (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)

プロパティ名	説明(Description)
com.fortify.sca.rules.password_regex.cpp	CおよびC++コードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)
com.fortify.sca.rules.password_regex.dart	Dartコードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)
com.fortify.sca.rules.password_regex.dotnet	.NETコードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)

プロパティ名	説明(Description)
com.fortify.sca.rules.password_regex.docker	Dockerfileのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** .*pass(wd|word|phrase).*
com.fortify.sca.rules.password_regex.golang	Goコードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)
com.fortify.sca.rules.password_regex.java	Javaコードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)

プロパティ名	説明(Description)
com.fortify.sca.rules.password_regex.javascript	JavaScriptおよびTypeScriptコードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)
com.fortify.sca.rules.password_regex.json	JSONのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (?i).*pass(wd|word|phrase).*
com.fortify.sca.rules.password_regex.jsp	JSPコードのパスワード識別子に一致するために使用される正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)

プロパティ名	説明(Description)
com.fortify.sca.rules.password_regex.objc	Objective-CおよびObjective-C++コードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (?i)(s|_)?(user|usr|member|admin|guest|login|default| new|current|old|client|server|proxy|sql server| my|mysql|mongo|mongodb|db|database|ldap|smtp| email|email(_)?smtp)?(_|\.)?(token|pin|pass(wd|word|phrase))
com.fortify.sca.rules.password_regex.php	PHPコードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)

プロパティ名	説明(Description)
com.fortify.sca.rules.password_regex.powershell	PowerShellファイル内のパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (?i)([a-z_]*|\{.*)(pass(wd|word|phrase)|pwd)(.*\}|[a-z_]*)
com.fortify.sca.rules.password_regex.properties	プロパティファイルのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)
com.fortify.sca.rules.password_regex.python	Pythonコードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)

プロパティ名	説明(Description)
com.fortify.sca.rules.password_regex.ruby	Rubyコードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)
com.fortify.sca.rules.password_regex.sql	SQLコードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)

プロパティ名	説明(Description)
com.fortify.sca.rules.password_regex.swift	Swiftコードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (?i)(s|_)?(user|usr|member|admin|guest|login|default| new|current|old|client|server|proxy|sql server| my|mysql|mongo|mongodb|db|database|ldap|smtp| email|email(_)?smtp)?(_|\.)?(token|pin|pass(wd|word|phrase))
com.fortify.sca.rules.password_regex.vb	VB6コードのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.password_regex.global の値)

プロパティ名	説明(Description)
com.fortify.sca.rules.password_regex.yaml	YAMLのパスワード識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現パスワードルールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (?i).*pass(wd|word|phrase).*
com.fortify.sca.rules.key_regex.global	言語固有の正規表現キールールプロパティが設定されている場合を除き、すべての言語のキー識別子に一致する正規表現。 **値の型:** 文字列 **デフォルト:** (?i)((enc|dec)(ryption|rypt)?|crypto|secret|private)(_)?key
com.fortify.sca.rules.key_regex.abap	ABAPコードのキー識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現キールールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.key_regex.global の値)

プロパティ名	説明(Description)
com.fortify.sca.rules.key_regex.actionscript	ActionScriptコードのキー識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現キールールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.key_regex.global の値)
com.fortify.sca.rules.key_regex.cfml	CFMLコードのキー識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現キールールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.key_regex.global の値)
com.fortify.sca.rules.key_regex.cpp	CおよびC++コードのキー識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現キールールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.key_regex.global の値)

プロパティ名	説明(Description)
com.fortify.sca.rules.key_regex.golang	Goコードのキー識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現キールールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.key_regex.global の値)
com.fortify.sca.rules.key_regex.java	Javaコードのキー識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現キールールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.key_regex.global の値)
com.fortify.sca.rules.key_regex.javascript	JavaScriptおよびTypeScriptコードのキー識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現キールールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.key_regex.global の値)

プロパティ名	説明(Description)
com.fortify.sca.rules.key_regex.jsp	JSPコードのキー識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現キールールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.key_regex.global の値)
com.fortify.sca.rules.key_regex.objc	Objective-CおよびObjective-C++コードのキー識別子と照合するために使用する正規表現。このプロパティを設定すると、グローバル正規表現キールールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.key_regex.global の値)
com.fortify.sca.rules.key_regex.php	PHPコードのキー識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現キールールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.key_regex.global の値)

プロパティ名	説明(Description)
com.fortify.sca.rules.key_regex.python	Pythonコードのキー識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現キールールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.key_regex.global の値)
com.fortify.sca.rules.key_regex.ruby	Rubyコードのキー識別子と照合するために使用される正規表現。このプロパティを設定すると、グローバル正規表現キールールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.key_regex.global の値)
com.fortify.sca.rules.key_regex.sql	SQLコードのキー識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現キールールプロパティが上書きされます。 **デフォルト:** (com.fortify.sca.rules.key_regex.global の値)

プロパティ名	説明(Description)
com.fortify.sca.rules.key_regex.swift	Swiftコードのキー識別子と照合するために使用される正規表現。このプロパティを設定すると、グローバル正規表現キールールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.key_regex.global の値)
com.fortify.sca.rules.key_regex.vb	Visual Basic 6コードのキー識別子に一致する正規表現。このプロパティを設定すると、グローバル正規表現キールールプロパティが上書きされます。 **値の型:** 文字列 **デフォルト:** (com.fortify.sca.rules.key_regex.global の値)

1.30. コマンドラインツール

OpenText SASTコマンドラインツールを使用すると、OpenText Application Security Contentの管理、インストール後の設定、およびスキャンの監視を実行できます。これらのツールは、 *<sast_install_dir>*/bin にあります。Windows用のツールは、 .bat または .cmd ファイルとして提供されます。次の表では、OpenText SASTと一緒にインストールされるコマンドラインツールについて説明します。

Note

デフォルトでは、OpenText SASTツールのログファイルは次のディレクトリに書き込まれます。

- Windows: C:\Users*<username>*\AppData\Local\Fortify*<tool_name>*-*<version>*\log
- Windows以外: *<userhome>*/.fortify/*<tool_name>*-*<version>*/log

ツール	説明(Description)	詳細情報
fortifyupdate	インストールされているセキュリティコンテンツを現在のバージョンと比較し、必要に応じてアップデートします	OpenText Application Security Contentの更新について
FPRUtility	このツールを使用すると、次の処理を実行できます。 • 監査されたプロジェクトをマージする • FPR署名を検証する • FPRファイルからの情報を表示する • ソースコードファイルと監査プロジェクトをFPRファイルに結合または分割する • FPRを変更する	*OpenText™ Application Securityツールガイド*
scapostinstall	このツールを使用すると、OpenText SASTの以前のバージョンのpropertiesファイルを移行したり、ロケールを指定したり、セキュリティコンテンツの更新やApplication Securityに使用するプロキシサーバを指定したりできます。	インストール後処理ツールの実行

ツール	説明(Description)	詳細情報
SCAState	分析フェーズ中にJVMに関する状態分析情報を提供します。	SCAStateを使用したスキャンステータスの確認

このセクションでは、次のトピックについて説明します。

1.30.1. OpenText Application Security Contentの更新について

fortifyupdateコマンドラインツールを使用して、最新のFortify Secure Coding RulepacksとメタデータをopenTextからダウンロードできます。

fortifyupdateツールは、OpenText SASTインストールに含まれている既存のセキュリティコンテンツに関する情報を収集し、この情報を使用してFortify Rulepack更新サーバに問い合わせます。サーバは新しいセキュリティコンテンツまたは更新されたセキュリティコンテンツを返し、古いセキュリティコンテンツをOpenText SASTインストールから削除します。インストールが最新の場合は、その旨のメッセージが表示されます。

このセクションでは、次のトピックについて説明します。

1.30.1.1. OpenText Application Security Contentの更新

fortifyupdateコマンドラインツールを使用して、セキュリティコンテンツをダウンロードするか、セキュリティコンテンツのローカルコピーをインポートします。このツールは *<sast_install_dir>*/bin ディレクトリにあります。

このツールのデフォルトの読み込みタイムアウトは180秒です。タイムアウト設定を変更するには、 rulepackupdate.SocketReadTimeoutSeconds 環境設定ファイルに server.properties プロパティを追加します。詳細については、『*OpenText™ Application Securityツールガイド*』を参照してください。

fortifyupdateの基本的なコマンドライン構文を次の例に示します。

```
fortifyupdate [<options>]
```

Fortify Rulepack更新サーバからの最新のFortify Secure Coding Rulepacksおよび外部メタデータを使用してOpenText SASTインストールを更新するには、次のコマンドを入力します。

```
fortifyupdate
```

ローカルシステムからセキュリティコンテンツを更新するには、次のコマンドを入力します。

```
fortifyupdate -import <my_local_rules>.zip
```

資格情報を使用してApplication Securityサーバからセキュリティコンテンツを更新するには、次のコマンドを入力します。

```
fortifyupdate -url <ssc_url> -sscUser <username> -sscPassword
<password>
```

1.30.1.2. fortifyupdateコマンドラインオプション

次の表では、fortifyupdateオプションについて説明します。

fortifyupdateオプション	説明(Description)
-acceptKey	公開鍵を受諾する場合に指定します。このオプションを指定すると、公開鍵の入力を求めるプロンプトは表示されません。 -url オプションを使って非標準の場所からOpenText Application Security Contentを更新する場合は、このオプションを使用して公開鍵を受諾します。
-acceptSSLCertificate	サーバによって提供されるSSL証明書を使用する場合に指定します。
-import *<file>*.zip	セキュリティコンテンツを含むZIPファイルをインポートします。デフォルトでは、Rulepackが *<sast_install_dir>*/Core/config/rules ディレクトリにインポートされます。
-coreDir *<dir>*	fortifyupdateで更新が保存されるコアディレクトリを指定します。このオプションが指定されていない場合、fortifyupdateによる更新は *<sast_install_dir>*で実行されます。 **Important** *<sast_install_dir>*/config/keys フォルダのコンテンツをコピーして、このディレクトリ内の config/keys フォルダに貼り付けしてから、 fortifyupdate を実行してください。
-includeMetadata	外部メタデータのみを更新することを指定します。

fortifyupdateオプション	説明(Description)
-includeRules	ルールパックのみを更新することを指定します。
-locale *<locale>*	ロケールを指定します。指定したロケールにセキュリティコンテンツが存在しない場合は、英語がデフォルトです。有効な値は次のとおりです。 • en (英語) • es (スペイン語) • ja (日本語) • ko (韓国語) • pt_BR (ポルトガル語(ブラジル)) • zh_CN (簡体字中国語) • zh_TW (繁体字中国語) **Note** この値では、大文字と小文字を区別しません。 または、 fortify.properties 環境設定ファイルでセキュリティコンテンツ更新のデフォルトロケールを指定できます。詳細については、『*OpenText™ Application Securityツールガイド*』を参照してください。
-proxyhost *<host>*	プロキシサーバのネットワーク名またはIPアドレスを指定します。
-proxyport *<port>*	プロキシサーバのポート番号を指定します。

fortifyupdateオプション	説明(Description)
`-proxyUsername <username>`	プロキシサーバが認証を必要とする場合、ユーザ名を指定します。
`-proxyPassword <password>`	プロキシサーバが認証を必要とする場合、パスワードを指定します。
`-showInstalledRules`	現在インストールされているRulepackを、カスタムルールとカスタムメタデータを含めて表示します。
`-showInstalledExternalMetadata`	現在インストールされている外部メタデータを表示します。
`-url <url>`	セキュリティコンテンツをダウンロードするURLを指定します。デフォルトのURLは、`https://update.fortify.com`、または`server.properties`環境設定ファイル内の`rulepackupdate.server`プロパティに設定された値です。 `server.properties`環境設定ファイルの詳細については、『*OpenText™ Application Securityツールガイド*』を参照してください。 Application Security URLを指定することで、Application Securityサーバからセキュリティコンテンツをダウンロードできます。
`-url`オプションを使ってApplication Securityからセキュリティコンテンツを更新する場合には、次のいずれかのタイプの資格情報を指定します。	

fortifyupdateオプション	説明(Description)
-sscUsername -sscPassword	ユーザ名とパスワードでApplication Securityユーザアカウントを指定します。
-sscAuthToken	タイプがUnifiedLoginToken、CIToken、またはToolsConnectTokenのApplication Security認証トークンを指定します。

1.30.2. SCAStateを使用したスキャンステータスの確認

SCAStateツールを使用して、分析フェーズ中に最新の状態分析情報を確認します。

状態を確認するには:

1. スキャンを開始します。
2. 別のコマンドウィンドウを開きます。
3. コマンドプロンプトで次のコマンドを入力します。

```
SCAState [<options>]
```

参照情報

SCAStateコマンドラインオプション

1.30.2.1. SCAStateコマンドラインオプション

次の表では、SCAStateオプションについて説明します。

SCAStateオプション	説明(Description)
-a | --all	使用可能なすべての情報を表示します。
-debug	SCAStateの振る舞いをデバッグする場合に役立つ情報を表示します。
-ftd | --full-thread-dump	各スレッドのスレッドダンプを出力します。
-h | --help	SCAStateツールのヘルプ情報を表示します。
-hd <filename> | --heap-dump <filename>	ヒープダンプの書き込みファイルを指定します。このファイルはリモートスキャンの作業ディレクトリを基準に解釈されます。これは必ずしもSCAStateを実行しているディレクトリとは限りません。
-liveprogress	実行中のスキャンの現在のステータスを表示します。これはデフォルトです。可能であれば、この情報は別の端末ウィンドウに表示されます。
-nogui	別のウィンドウではなく、現在の端末ウィンドウにOpenText SASTの状態情報を表示します。
-pi | --program-info	スキャン中のソースコードに関する情報(含まれるソースファイルと関数の数など)が表示されます。

SCAStateオプション	説明(Description)
`-pid <process_id>`	現在実行中のOpenText SASTプロセスIDを指定します。複数のOpenText SASTプロセスが同時に実行されている場合は、このオプションを使用します。 Windowsシステム上でプロセスIDを取得するには、次の手順を実行します。 1. コマンドウィンドウを開きます。 2. コマンドプロンプトで「`tasklist`」と入力します。 プロセスのリストが表示されます。 3. リスト内で`java.exe`プロセスを見つけ、PIDをメモします。 Linuxシステム上でプロセスIDを取得するには、次の手順を実行します。 • コマンドプロンプトで「`ps aux | grep sourceanalyzer`」と入力します。
`-progress`	コマンドが発行された時点までのスキャン情報を表示します。これには、経過時間、分析の現在のフェーズ、および取得済みの結果数が含まれます。
`-properties`	環境設定を表示します(パスワードなどの機密情報は含まれません)。
`-scaversion`	現在実行中のsourceanalyzerのOpenText SASTバージョン番号を表示します。

SCAStateオプション	説明(Description)
`-td` | `--thread-dump`	メインスキャンスレッドのスレッドダンプを出力します。
`-timers`	OpenText SASTで計測されるタイマおよびカウンタからの情報を表示します。
`-version`	SCAStateバージョンを表示します。
`-vminfo`	JVM標準MXBeansで提供される、ClassLoadingMXBean、CompilationMXBean、GarbageCollectorMXBeans、MemoryMXBean、OperatingSystemMXBean、RuntimeMXBean、およびThreadMXBeanに関する統計情報を表示します。
<none>	スキャン進行状況情報を表示します(これは `-progress` と同じです)。

Note

OpenText SASTでは、Javaプロセス情報をTMPシステム環境変数の場所に書き込みます。Windowsシステムでは、TMPシステム環境変数の場所は `C:\Users\<username>\AppData\Local\Temp` です。別の場所を指すようにこのTMPシステム環境変数を変更した場合、SCAStateは `sourceanalyzer` Javaプロセスを見つけ出せず、予期された結果を返しません。この問題を解決するには、新しいTMPの場所に一致するようにTMPシステム環境変数を変更します。SCAStateをWindows管理者として実行することを推奨します。

For more info, visit https://docs.microfocus.com