Micro Focus
Fortify Static Code Analyzer

ソフトウェアバージョン: 21.2.0

ユーザガイド

ドキュメントリリース日 : 2021年 11月
ソフトウェアリリース日 : 2021年 11月

法的通知

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK

https://www.microfocus.com

保証

Micro Focusとその関連会社およびライセンサ(以下「Micro Focus」)の製品およびサービスに関する保証は、製品およびサービスに付属する保証規定に明示されている内容に限定されます。本書のいかなる記述も、追加の保証を構成するものではありません。Micro Focusは、本書の技術的内容や編集に関する誤りや欠落に関して責任を負いません。ここに記載する情報は、予告なしに変更されることがあります。

権利の制限

機密性のあるコンピュータソフトウェアです。明確な指示がある場合を除き、保持、使用、またはコピーには、Micro Focusからの有効なライセンスが必要です。FAR 12.211および12.212に従って、商用コンピュータソフトウェア、コンピュータソフトウェアドキュメント、および商用品目の技術データは、米国政府に対して、ベンダーの標準商用ライセンスに基づいてライセンスされます。

著作権表示

商標表示

このドキュメントに記載されているすべての商標、サービスマーク、製品名、およびロゴは、該当する所有者に帰属します。

ドキュメントの更新情報

このドキュメントのタイトルページには、次の識別情報が記載されています。

- ソフトウェアバージョン番号
- ドキュメントリリース日。ドキュメントが更新されるたびに変更されます
- ソフトウェアリリース日。ソフトウェアのこのバージョンのリリース日付を示します

このドキュメントは7月 13, 2022に作成されました。最新の更新を確認する場合や、最新のドキュメントを使用しているかを確認する場合は、次のサイトをご覧ください。

https://www.microfocus.com/support/documentation

目次

序文

Micro Focus Fortifyカスタマサポートへのお問い合わせ

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

https://www.microfocus.com/support

詳細情報

Fortifyソフトウェア製品について詳しくは、次のリンクを参照してください。

https://www.microfocus.com/cyberres/application-security

ドキュメントセットについて

Fortifyソフトウェアのドキュメントセットには、すべてのFortifyソフトウェア製品およびコンポーネントのインストールガイド、ユーザガイド、および展開ガイドが含まれています。また、新機能、既知の問題、および最新の更新情報について説明するテクニカルノートとリリースノートもあります。これらのドキュメントの最新バージョンには、次のMicro Focus製品ドキュメントWebサイトからアクセスできます。

https://www.microfocus.com/support/documentation

リリース間のドキュメント更新のお知らせを受け取るには、Micro FocusコミュニティのFortify製品のお知らせを購読してください。

https://community.microfocus.com/cyberres/fortify/w/fortify-product-announcements

Fortify製 品 の機 能 紹 介 ビデオ

YouTubeのFortify Unpluggedチャネルで、Fortifyの製 品 と機 能 を紹 介 するビデオをご覧 いただけます。

https://www.youtube.com/c/FortifyUnplugged

変更ログ

次の表に、このドキュメントで行われた変更を示します。この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/ドキュメントバージョン	変更点
21.2.0	追加: • "信頼された証明書の追加" ページ44 - Fortify Static Code AnalyzerおよびそのツールとHTTPS経由の通信を必要とする他のFortify製品との接続に関する情報 • "正規表現分析" ページ51 - 正規表現ルールを使用してファイルを分析する新機能 • "ライセンス" ページ20および"LIMライセンスディレクティブ" ページ137 - Fortify License and Infrastructure Manager (LIM)を使用して同時ライセンスを管理する新機能 更新: • "Fortify Static Code Analyzerとアプリケーションのインストール" ページ30 - Fortify License and Infrastructure Manager (LIM)サーバの使用手順を追加し、AIXおよびSolarisオペレーティングシステム用のインストーラを追加しました。 • "インポート済みのモジュールとパッケージを含める" ページ78 - Fortify Static Code Analyzerで、自動的に計算された共通ルートディレクトリからモジュールとパッケージをインポートできるようになりました。 • "Scalaコードの変換" ページ114 - Scalaコードを分析するためにLightbend Enterprise Suiteのライセンスを購入するという要件を削除しました • "変換オプション" ページ126 - MSBuildおよびxcodebuild統合で`-exclude`オプションがサポートされるようになりました • "BIRTレポートの生成" ページ152 - PCI SSFコンプライアンスレポートバージョン1.1のサポートを追加しました • "Fortify Scan Wizardを使用するための準備" ページ187 - Fortify ScanCentral SASTで完全な分析をリモート実行する新機能

ソフトウェアリリース/ドキュメントバージョン	変更点
	• "高度なサンプル" ページ192 - webgoat、riches.java、riches.netの3つのサンプルを削除しました • "fortify-sca.properties" ページ196 - Fortify License and Infrastructure Manager (LIM)で管理されるライセンスの使用と正規表現分析に関する新しいプロパティの説明を追加しました
21.1.0	追加: • "Kotlinスクリプトの変換" ページ63 - Kotlinスクリプトの変換が新たにサポートされるようになりました • "BIRTReportGeneratorのトラブルシューティング" ページ155 - メモリ不足の問題を解決するために追加された情報 更新: • "Javaバイトコードの変換" ページ58 - 変換フェーズに含めるJavaバイトコードを逆コンパイルする新しいオプション • "Visual StudioおよびMSBuildプロジェクトの変換" ページ65 - MSBuild統合のために行われた更新 • "Goコマンドラインオプション" ページ86 - GoプロキシURLを指定するために追加されたオプション • "COBOLコードの変換" ページ91 - Micro Focus Visual COBOLのサポートに関して更新しました • "Dockerfileの変換" ページ114 - Dockerfilesとして分析されるファイルの説明を更新しました • "分析オプション" ページ128および"環境設定オプション" ページ194 - FindBugsに関連するオプションとプロパティを削除しました • "コマンドラインからのFPRファイルの操作" ページ144 - FPRUtilityに対して行われた機能拡張 • "BIRTレポートの生成" ページ152 - 新しいレポートバージョンが使用可能になりました • "短縮ダイヤルを使用したスキャン速度の設定" ページ165 - 新しい精度レベル3と4を追加しました 削除: • FindBugsの使用 - サポートされなくなりました

ソフトウェアリリース/ドキュメントバージョン	変更点
20.2.0	追加: • "Fortify Static Code Analyzerとアプリケーションのインストールについて" ページ29および"Dockerを使用したFortify Static Code Analyzerのインストールと実行" ページ36 • "Dockerfileの変換" ページ114 • "短縮ダイヤルを使用したスキャン速度の設定" ページ165 • "Fortify Javaの注釈" ページ225 - 以前にjavaAnnotationsサンプルの`README.txt`で入手可能だった情報をこのガイドに取り込みました 更新: • "Visual StudioおよびMSBuildプロジェクトの変換" ページ65 - これまでのいくつかのリリースで行われた変換の改善点を反映して更新しました(以前の章タイトル: .NETコードの変換) • "COBOLコードの変換" ページ91 - COBOLコードを分析するために導入された変更点について説明します • "Fortify ABAP Extractorの実行" ページ104 - カスタムコードに加えてSAP標準コードをエクスポートする新しいオプション • "BIRTレポートの生成" ページ152 - 新たにサポートされたレポートテンプレート(OWASP ASVS 4.0)を追加しました • "サンプルプロジェクト" ページ190 - 2つの新しいサンプルを追加しました • Fortify ScanCentralのすべての参照をFortify ScanCentral SASTに置き換えました(製品名の変更)
20.1.2	追加: • "Kotlinコードの変換" ページ61 更新: • "Gradleの統合" ページ120 - Gradle Wrapperの使用に関する情報を追加しました
20.1.0	更新: • "Fortify Static Code Analyzerとアプリケーションのインストールについて" ページ29 - Solarisがサポートされなくなったため、このオペレーティングシステムに関する記述をすべて削除しました

ソフトウェアリリース/ドキュメントバージョン	変更点
	• "Fortify Static Code Analyzerとアプリケーションのサイレント(無人)インストール" ページ32 - さまざまなオペレーティングシステムの手順に詳しい情報を追加しました • "BIRTレポートの生成" ページ152 - 新しいレポート(CWE Top 25 2019)のサポートを追加しました • "レガシレポートの生成" ページ155 - 可能な出力形式としてRTFを削除しました • Fortify CloudScanのすべての参照をFortify ScanCentralに置き換えました(製品名の変更) 削除: • 増分分析 - 次回のリリースで削除される機能

第1章: はじめに

このガイドでは、Micro Focus Fortify Static Code Analyzer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。このガイドの対象は、セキュリティの監査およびセキュアなコーディングの責任者です。

このセクションでは、次のトピックについて説明します。

Fortify Static Code Analyzer

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。Fortify Static Code Analyzer言語技術では、迅速で正確に修正できるように、アナライザで違反を特定して優先順位を付けることができる豊富なデータが提供されています。Fortify Static Code Analyzer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。この設計によって、顧客固有のセキュリティルールを組み込むことができます。

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、ドキュメント『*Micro Focus Fortifyソフトウェアシステム要件*』を参照してください。

最高レベルでは、Fortify Static Code Analyzerを使用して次の処理を実行します。

1. スタンドアロンプロセスとしてFortify Static Code Analyzerを実行するか、ビルドツールにFortify Static Code Analyzerを統合する
2. ソースコードを中間変換形式に変換する
3. 変換されたコードをスキャンし、セキュリティ脆弱性レポートを生成する
4. Micro Focus Fortify Audit Workbenchで結果(通常はFPRファイル)を開くか、分析用にMicro Focus Fortify Software Security Centerをアップロードしてスキャンの結果を監査する、または画面に表示された結果を直接操作する。

注: Fortify Audit WorkbenchまたはFortify Software Security Centerで結果を開いて表示する方法については、それぞれ『*Micro Focus Fortify Audit Workbenchユーザガイド*』または『*Micro Focus Fortify Software Security Centerユーザガイド*』を参照してください。

アナライザについて

Fortify Static Code Analyzerは、7つの脆弱性アナライザ(Buffer、Configuration、Content、Control Flow、Dataflow、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、各アナライザのリストを表示して説明します。

アナライザ	説明
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、Fortify Static Code Analyzer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テイント(ユーザ制御)されている場合は、その値もFortify Static Code Analyzerでレポートされ、どのように変数がテイント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。
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" ページ51」を参照)。
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。
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では、チェックの時刻/使用時刻の問題や初期化されていない変数の時刻が検出され、XMLリーダ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がチェックされます。

アナライザ	説 明
Dataflow	Dataflow Analyzerでは、テイントデータ(ユーザ制 御 入 力)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では、グローバルなプロシージャ間 テイント伝 播 分 析 を使 用 して、ソース(ユーザ入 力 のサイト)とシンク(危 険 な関 数 呼 び出 しや操 作)間 のデータフローが検 出 されます。たとえば、Dataflow Analyzerでは、ユーザが制 御 する無 制 限 の長 さの入 力 文 字 列 が静 的 サイズのバッファにコピーされているかどうかが検 出 され、ユーザが制 御 する文 字 列 を使 用 してSQLクエリテキストが作 成 されるかどうかが検 出 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。
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。この特 殊 なロジックでは、バッファオーバーフロー、フォーマット文 字 列 、および実 行 パスに関 する問 題 が検 索 されますが、これらのカテゴリに限 定 されません。たとえば、Semantic AnalyzerではJavaで非 推 奨 になった関 数 が検 出 され、Javaの関 数 およびC/C++の安 全 でない関 数 (gets()など)が検 出 されます。
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では、Javaサーブレット内 のメンバー変 数 への割 り当 てが検 出 され、static final宣 言 されていないロガーの使 用 が特 定 され、述 語 が常 にfalseであるために実 行 されないデッドコードのインスタンスにフラグが付 けられます。

ライセンス

Fortify Static Code Analyzer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" ページ46」を参 照)。Fortify Static Code Analyzerのライセンスを取 得 する方 法 の詳 細 については、ドキュメント『*Micro Focus Fortifyソフトウェアシステム要 件* 』を参 照 してください。

Fortifyのライセンスファイル(fortify.license)が必 要 です。オプションで、Micro Focus Fortify License and Infrastructure Manager (LIM)を使 用 してFortify Static Code Analyzer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Fortify Static Code Analyzerのインストールで単 一 のライセンスを共 有 できます(詳 細 については、『*Micro Focus Fortify License and Infrastructure Managerインストールおよび使 用 ガイド*』を参 照)。LIMライセンスの管 理 の詳 細 については、「"LIMライセンスディレクティブ" ページ137」を参 照 してください。

Fortify Software Security Content

Fortify Static Code Analyzer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。Micro Focus Fortify Software Security Contentは、変 換 と分 析 の両 方 に必 要 です。Fortify Static Code Analyzer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(「"Fortify Static Code Analyzerのインストール" ページ27」を参 照)。また、インストール後 のタスクとしてfortifyupdateユーティリティを使 用 して、以 前 にダウンロードしたFortify Security Contentをダウンロードまたはインポートすることもできます(「"Fortify Security Contentの手 動 インストール" ページ35」を参 照)。

Fortify Software Security Content (セキュリティコンテンツ)は、Secure Coding Rulepacksと外 部 メタデータで構 成 されています。

- 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。

Fortifyでは、Fortify Static Code Analyzerと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 が提 供 されています。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、『*Micro Focus Fortify Static Code Analyzerカスタムルールガイド*』を参 照 してください。

Fortifyでは、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ユーティリティを使 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 " ページ142」を参 照 してください。

Fortify ScanCentral SAST

Micro Focus Fortify ScanCentral SASTを使 用 すると、Fortify Static Code Analyzerのスキャン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、Fortify 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。Micro Focus Fortify Software Security Centerのユーザは、FPRファイルをサーバに直 接 出 力 するようにFortify ScanCentral SASTに指 示 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。

- 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、Fortify 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、システムを簡単に拡張できます。さらに、
- Fortify 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Fortify ScanCentral SASTにオフロードできます。サポートされている特定の言語については、ドキュメント『*Micro Focus Fortifyソフトウェアシステム要件*』を参照してください。

Fortify ScanCentral SASTを使用する方法の詳細については、『*Micro Focus Fortify 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。

Fortify Scan Wizard

Micro Focus Fortify Scan Wizardは、WindowsまたはLinux/macOSシステム用のFortify Static Code Analyzerコマンドを実行するスクリプトを簡単に生成できるユーティリティです。この生成されたスクリプトを実行すると、Fortify Static Code Analyzerを使用してコードを分析できます 。分析をローカルで実行するように、またはMicro Focus Fortify ScanCentral SASTを使用して分析のすべてまたは一部をリモートで実行するように指定することもできます。詳細については、「"Fortify Scan Wizard" ページ187」を参照してください。

関連ドキュメント

このトピックでは、Micro Focus Fortifyソフトウェア製品に関する情報を提供するドキュメントについて説明します。

注 : Micro Focus Fortifyの製品ドキュメントは、https://www.microfocus.com/support/documentationにあります。ほとんどのガイドは、PDF形式とHTML形式の両方で提供されています。製品ヘルプは、Fortify LIM製品内で利用できます。

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Micro Focus製 品ドキュメントWebサイトで利 用 できます。

ドキュメント/ファイル名	説 明
Micro Focus Fortify製 品 ソフトウェアのドキュメントについて About_Fortify_Docs_<*version*>.pdf	この文 書 では、Micro Focus Fortify製 品 マニュアルにアクセスする方 法 について説 明します。 **注 :** 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。
Micro Focus Fortify License and Infrastructure Managerインストールおよび使 用 ガイド LIM_Guide_<*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。
Micro Focus Fortifyソフトウェアシステム要 件 Fortify_Sys_Reqs_<*version*>.pdf	このドキュメントでは、Fortifyソフトウェアのこのバージョンでサポートされている環 境 と製 品 について詳 しく説 明します。
Micro Focus Fortifyソフトウェアリリースノート FortifySW_RN_<*version*>.pdf	このドキュメントでは、Fortifyソフトウェア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。
Micro Focus Fortifyソフトウェア<*version*>の新 機 能 Fortify_Whats_New_<*version*>.pdf	このドキュメントでは、Fortifyソフトウェア製 品 の新 機 能 について説 明します。

Micro Focus Fortify ScanCentral SAST

以 下 のドキュメントには、Fortify ScanCentral SAST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Micro Focus製 品 マニュアルのWebサイト(https://www.microfocus.com/documentation/fortify-software-security-center)で利 用 できます。

ドキュメント/ファイル名	説 明
Micro Focus Fortify ScanCentral SASTインストール、設 定 、使 用 ガイド SC_SAST_Guide_*<version>*.pdf	このドキュメントでは、Fortify ScanCentral SASTをインストール、設 定 、使 用 して、静 的コード分 析 のプロセスを合 理 化 する方 法 について説 明 します。これは、リソースを大 量 に消 費 するFortify Static Code Analyzerプロセスの変 換 およびスキャンフェーズをオフロードするためにFortify ScanCentral SASTをインストール、設 定 、または使 用 するユーザを対 象 にしています。

Micro Focus Fortify Software Security Center

以 下 のドキュメントには、Fortify Software Security Center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Micro Focus製 品 マニュアルのWebサイト(https://www.microfocus.com/documentation/fortify-software-security-center)で利 用 できます。

ドキュメント/ファイル名	説 明
Micro Focus Fortify Software Security Centerユーザガイド SSC_Guide_*<version>*.pdf	このドキュメントでは、Software Security Centerのユーザ向 けに、Software Security Centerを展 開 および使 用 する方 法 について詳 しく説 明 します。Software Security Centerの取 得 、インストール、設 定 、使 用 に必 要 なすべての情 報 を提 供 します。 これは、システムおよびインスタンス管 理 者 、データベース管 理 者 (DBA)、エンタープライズセキュリティリード、開 発 チームマネージャ、および開 発 者 による使 用 を目 的 としています。Software Security Centerは、セキュリティチームのリードにプロジェクトの履 歴 と現 在 のステータスの概 要 を提 供 します。

Micro Focus Fortify Static Code Analyzer

以 下 のドキュメントには、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Micro Focus製 品 マニュアルのWebサイト(https://www.microfocus.com/documentation/fortify-static-code)で利 用 できます。

ドキュメント/ファイル名	説 明
Micro Focus Fortify Static Code Analyzerユーザガイド SCA_Guide_<*version*>.pdf	このドキュメントでは、Static Code Analyzerをインストールおよび使 用して、多くの主 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。
Micro Focus Static Code Analyzerカスタムルールガイド SCA_Cust_Rules_Guide_<*version*>.zip	このドキュメントでは、Static Code Analyzer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。 **注 :** 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。
Micro Focus Audit Workbenchユーザガイド AWB_Guide_<*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。
Micro Focus Fortify Eclipse用プラグインユーザガイド Eclipse_Plugins_Guide_<*version*>.pdf	このドキュメントでは、Fortify Eclipse用 修 復プラグインをインストールして使 用する方 法について説 明します。
Micro Focus Fortify JetBrains IDEおよびAndroid Studio用プラグインユーザガイド JetBrains_AndStud_Plugins_Guide_<*version*>.pdf	このドキュメントでは、IntelliJ IDEAおよびAndroid Studio用 のFortify分 析 プラグインと、IntelliJ IDEA、Android Studio、およびその他 のJetBrains IDE用 のFortify修 復プラグインの両 方をインストールして使 用する方 法について説 明します。

ドキュメント/ファイル名	説明
Micro Focus Fortify Extension for Visual Studioユーザガイド VS_Ext_Guide_*<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。
Micro Focus Fortify Static Code Analyzerツールのプロパティリファレンスガイド SCA_Tools_Props_Ref_*<version>*.pdf	このドキュメントでは、Fortify Static Code Analyzerツールで使用されるプロパティについて説明します。

第2章: Fortify Static Code Analyzerのインストール

この章では、Fortify Static Code AnalyzerとFortify Static Code Analyzerのツール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。この章では、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。ドキュメント『*Micro Focus Fortifyソフトウェアシステム要件*』を参照して、各ソフトウェアコンポーネントのインストールに関する最小要件をシステムが満たしていることを確認してください。

このセクションでは、次のトピックについて説明します。

Fortify Static Code Analyzerツール

このインストールは、実行される分析のタイプに必要な情報を提供するように特別に調整された一連のルールに従ってビルドコードを分析するFortify Static Code Analyzerで構成されます。Fortify Static Code Analyzerのインストールには、1つ以上のアプリケーションが含まれる場合があります。

次の表では、Fortify Static Code Analyzerおよびアプリケーションのインストーラを使用してインストールできるツールについて説明します。

ツール	説明
Micro Focus 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、Fortify Static Code Analyzerのグラフィカルユーザインタフェースを提供します。
Micro Focus 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。

ツール	説明
Micro Focus Fortify Analysis Plugin for IntelliJ and Android Studio	プロジェクトのコードベース全体でFortify Static Code Analyzerスキャンを実行し、IntelliJおよびAndroid Studio IDE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。
Micro Focus 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、Micro Focus Fortify Software Security Center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。
Micro Focus Fortify Custom Rules Editor	カスタムルールを作成および編集するためのツール。
Micro Focus Fortify Scan Wizard	Fortify Static Code Analyzerでコードをスキャンするために使用できるスクリプトをすばやく準備し、必要に応じてFortify Software Security Centerに結果を直接アップロードするためのツール。 **注:** このツールは、Fortify Static Code Analyzerと一緒に自動的にインストールされます。
コマンドラインユーティリティ	Fortify Static Code Analyzerと一緒に自動的にインストールされるコマンドラインユーティリティが複数あります。詳しくは、"コマンドラインユーティリティ" ページ140を参照してください。

次の表では、Fortify Static Code AnalyzerおよびアプリケーションのパッケージにContainされているツールについて説明します。これらのツールは、Fortify Static Code AnalyzerおよびアプリケーションのインストーラとはContainにインストールします。

ツール	説明
Micro Focus Fortify Remediation Plugin for Eclipse	Eclipse IDEからソースコードで検出された問題を修正する必要がある開発者のために、Fortify Software Security Centerと連携して動作します。
Micro Focus Fortify Remediation Plugin for	複数のJetBrains IDEおよびAndroid Studioで、セキュリティ分析に修正機能を追加するためにFortify Software

ツール	説明
JetBrains IDEs and Android Studio	Security Centerと連携して動作します。
Micro Focus Fortify Security Assistant Plugin for Eclipse	Javaコードの作成時に潜在的なセキュリティの問題に対する警告を表示します。セキュリティリスクに関する詳細情報と潜在的な問題から保護する方法に関する推奨事項を提供します。

Fortify Static Code Analyzerとアプリケーションのインストールについて

このセクションでは、Fortify SCAとアプリケーションをインストールする方法について説明します。Fortify Static Code Analyzer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。次の表には、Fortify SCAとアプリケーションをインストールする方法を示します。

インストール方法	指示
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	"Fortify Static Code Analyzerとアプリケーションのインストール" 次のページ
サイレント(無人)でインストールを実行する	"Fortify Static Code Analyzerとアプリケーションのサイレント(無人)インストール" ページ32
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のFortify Static Code Analyzerとアプリケーションのテキストベースモードのインストール" ページ35
Dockerを使用してインストールを実行する	"Dockerを使用したFortify Static Code Analyzerのインストールと実行" ページ36

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Fortify Static Code Analyzerをインストールします。

注: 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Fortify SCAとアプリケーション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Fortify SCAとアプリケーション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" ページ41」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Fortify Static Code Analyzerとそのコンポーネントのパフォーマンスを設定することもできます。Fortify Static Code Analyzerの環境設定オプションについては、「"環境設定オプション" ページ194」を参照してください。
Fortify Static Code Analyzerコンポーネントアプリケーションの環境設定オプションについては、『*Micro Focus Fortify Static Code Analyzerツールプロパティリファレンスガイド*』を参照してください。

Fortify Static Code Analyzerとアプリケーションのインストール

Fortify Static Code Analyzerとアプリケーションをインストールするには、次の手順に従います。

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ます。
   - Windows: `Fortify_SCA_and_Apps_<version>_windows_x64.exe`
   - Linux: `Fortify_SCA_and_Apps_<version>_linux_x64.run`
   - macOS: `Fortify_SCA_and_Apps_<version>_osx_x64.app.zip`
   - AIX: `Fortify_SCA_<version>_aix_x64.run`
   - Solaris: `Fortify_SCA_<version>_solaris_x86.run`または`Fortify_SCA_<version>_solaris10_sparc.run`

   ここで、<*version*>はソフトウェアリリースのバージョンです。
2. ライセンス契約に受諾し、 [**Next**]をクリックします。
3. Fortify Static Code Analyzerとアプリケーションのインストール先を選択し、 [**Next**]をクリックします。

   注: Micro Focus Fortify ScanCentral SASTを使用している場合は、パスにスペースが含まれていない場所を指定する必要があります。
4. (オプション)インストールするコンポーネントを選択し、 [**Next**]をクリックします。

   注: 一部のオペレーティングシステムではコンポーネントを選択できません。
5. `fortify.license`ファイルへのパスを指定し、 [**Next**]をクリックします。
6. (オプション) [**LIM license**]ページで、同時ライセンスの管理にFortify License and Infrastructure Manager (LIM)が使用されるように [**Yes**]を選択してから、 [**Next**]をクリックします。

   注: Fortify Static Code Analyzerでライセンスが必要なタスクが実行されたら、Fortify Static Code Analyzerでライセンスプールからの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Fortify Static Code Analyzer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、`fortify-sca.properties`ファイルの`com.fortify.sca.lim.WaitForInitialLicense`を使用します(「"fortify-sca.properties" ページ196」を参照)。

a. LIMサーバのURL、ライセンスプール名、およびプールパスワードを入力します。
b. [Next]をクリックします。[LIM Proxy Settings]ページが開きます。
c. 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(オプション)を入力します。
d. [Next]をクリックします。

7. セキュリティコンテンツを更新するために必要な設定を指定します。

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、次の手順に従います。

注: Windows以外のプラットフォームにインストールする場合、および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である場合は、fortifyupdateユーティリティを使用してセキュリティコンテンツを更新できます。「"Fortify Security Contentの手動インストール" ページ35」を参照してください。

a. 更新サーバのURLを入力します。セキュリティコンテンツの更新にFortifyルールパック更新サーバを使用するには、URLを`https://update.fortify.com`のままにします。
b. (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。
c. [Next]をクリックします。

8. システムで以前のFortify Static Code Analyzer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Fortify Static Code Analyzerのインストールから移行すると、Fortify Static Code Analyzer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、「"Fortify Static Code Analyzerとアプリケーションのアップグレードについて" ページ38」を参照してください。

注: `scapostinstall`コマンドラインユーティリティを使用してFortify Static Code Analyzer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Fortify Static Code Analyzer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" ページ41」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。

a. [SCA Migration]ページで[Yes]を選択し、[Next]をクリックします。
b. システム上の既存のFortify Static Code Analyzerのインストール場所を指定し、[Next]をクリックします。

9. Visual Studio用のFortify拡張機能をインストールする場合は、現在のインストールユーザ向けの拡張機能をインストールするのか、すべてのユーザ向けの拡張機能をインストールするのかを指定するように求めるプロンプトが表示されます。

デフォルトでは、現在のインストールユーザ向けの拡張機能がインストールされます。

10. サンプルのソースコードプロジェクトをインストールするかどうかを指定し、[Next]をクリックします。

これらのサンプルの説明については、「"サンプルプロジェクト" ページ190」を参照してください。

注: サンプルをインストールせずに、後でインストールする場合は、Fortify Static Code Analyzerとアプリケーションをアンインストールしてから再インストールする必要がありま

す。

11. [**Next**]をクリックして、Fortify Static Code Analyzerとアプリケーションのインストールを続行します。
12. Fortify Static Code Analyzerがインストールされ後に、セキュリティコンテンツを更新する場合は、[**Update security content after installation**]を選択し、[**Finish**]をクリックします。

    [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。

Fortify Static Code Analyzerとアプリケーションのサイレント(無人)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。Fortify Static Code Analyzerとアプリケーションをサイレントでインストールすると、インストーラでMicro Focus Fortify Software Security Contentがダウンロードされません。Fortify Security Contentをインストールする方法については、「"Fortify Security Contentの手動インストール" ページ35」を参照してください。

Fortify Static Code Analyzerをサイレントでインストールするには、次の手順に従います。

1. オプションファイルを作成します。
   a. 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。
   b. 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>
      ```

   c. `https://update.fortify.com`のデフォルトとは異なる場所をFortify Security Contentの更新に使用している場合は、次の行を追加します。

      ```
      UpdateServer=<update_server_url>
      ```

      注: すでに説明したように、サイレントインストールではFortify Security Contentがダウンロードされません。ただし、*<sca_install_dir>*`/Core/config/server.properties`を手動でインストールすると、この情報および次のステップのプロキシ情報が使用するFortify Security Contentファイルに追加されます。

d. プロキシサーバが必要な場合は、次の行を追加します。

```
UpdateProxyServer=<proxy_server> UpdateProxyPort=<port_number>
```

e. サンプルのソースコードプロジェクトをインストールしない場合は、次の行を追加します。

```
InstallSamples=0
```

f. 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。

Windowsの`enable-components`オプションでは、`AWB_group`パラメータを指定して、Fortify Audit Workbench、Fortify Custom Rules Editorをインストールし、FPRファイルをFortify Audit Workbenchと関連付けます。特定のプラグインをインストールするには、各プラグインをパラメータ名でリストに表示します(`Plugins_group`パラメータでは一部のプラグインがインストール***されず***、追加する必要はありません)。

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、Fortify Security Contentを取得するための場所とプロキシ情報、以前のリリースから移行する要求、Audit Workbenchのインストール、すべてのユーザ向けのMicro Focus Fortify Extension for Visual Studio 2019のインストール、Fortify SCAとアプリケーションのインストールディレクトリの場所が指定されています。

```
fortify_license_path=C:\Users\admin\Desktop\fortify.license
UpdateServer=https://internalserver.abc.com
UpdateProxyServer=webproxy.abc.company.com
UpdateProxyPort=8080
MigrateSCA=1
enable-components=AWB_group,VS2019 VS_all_users=1
installdir=C:\Fortify
```

LinuxおよびmacOSのオプションファイルの例は、次のとおりです。

```
fortify_license_path=/opt/Fortify/fortify.license
UpdateServer=https://internalserver.abc.com
UpdateProxyServer=webproxy.abc.company.com UpdateProxyPort=8080
MigrateSCA=1 installdir=/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。

注: 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	`Fortify_SCA_and_Apps_<version>_windows_x64.exe--mode unattended--optionfile <full_path_to_options_file>`
Linux	`./Fortify_SCA_and_Apps_<version>_linux_x64.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。 `Fortify_SCA_and_Apps_<version>_osx_x64.app/Contents/MacOS/installbuilder.sh --mode unattended --optionfile <full_path_to_options_file>`
AIXおよびSolaris	`./Fortify_SCA_<version>_<platform>.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\FortifySCAandApps-<version>-install.log`
Linux	`/tmp/FortifySCAandApps-<version>-install.log`
macOS	`/tmp/FortifySCAandApps-<version>-install.log`
AIXおよびSolaris	`/tmp/FortifySCA-<version>-install.log`

Windows以外のプラットフォームでのFortify Static Code Analyzerとアプリケーションのテキストベースモードのインストール

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。

Fortify Static Code Analyzerとアプリケーション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	`./Fortify_SCA_and_Apps_<version>_linux_x64.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。 `Fortify_SCA_and_Apps_<version>_osx_x64.app/Contents/MacOS/installbuilder.sh --mode text`
AIXおよびSolaris	`./Fortify_SCA_<version>_<platform>.run --mode text`

Fortify Security Contentの手動インストール

Windowsのインストール手順時に、Micro Focus Fortify Software Security Content (Secure Coding Rulepacksとメタデータ)を自動的にインストールできます。ただし、Fortifyルールパック更新サーバからFortify Security Contentをダウンロードしてから、fortifyupdateユーティリティ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、Windows以外のプラットフォームへのインストール、および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Fortify Security Contentをインストールするには、fortifyupdateユーティリティを使用します。

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ます。
2. `<sca_install_dir>/bin`ディレクトリに移動します。
3. コマンドプロンプトで「`fortifyupdate`」と入力します。
   以前にFortifyルールパック更新サーバからFortify Security Contentをダウンロードした場合は、`-import`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`fortifyupdate`を実行します。

この同じユーティリティを使用して、セキュリティコンテンツを更新することもできます。fortifyupdateユーティリティの詳細については、「"セキュリティコンテンツの更新" ページ142」を参照してください。

Dockerを使用したFortify Static Code Analyzerのインストールと実行

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

注: Fortify Static Code Analyzerは、Dockerでサポートされているプラットフォームでのみ実行できます。

Fortify Static Code Analyzer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Fortify Static Code Analyzer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。

   注: Fortify Static Code Analyzer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、「"ビルド統合" ページ118」を参照してください。

2. COPY命令を使用して、Fortify SCAとアプリケーションインストーラ、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を作成する方法については、「"Fortify Static Code Analyzerとアプリケーションのサイレント(無人)インストール" ページ32」を参照してください。

3. RUN命令を使用してFortify SCAとアプリケーション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、「"Fortify Static Code Analyzerとアプリケーションのサイレント(無人)インストール" ページ32」を参照してください。

4. fortifyupdateを実行して、RUN命令を使用してFortify Security Contentをダウンロードします。

   このユーティリティの詳細については、「"Fortify Security Contentの手動インストール" 前のページ」を参照してください。

5. Fortify Static Code Analyzer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/Fortify_SCA_and_Apps_<version>/bin/sourceanalyzerです。

Fortify SCAとアプリケーションをインストールするDockerfileの例は、次のとおりです。

```
FROM registry.suse.com/suse/sles12sp4 COPY fortify.license ./ COPY Fortify_SCA_and_Apps_21.1.0_
linux_x64.run ./ COPY installerSettings ./ RUN zypper -n install rpm-build RUN ./Fortify_SCA_and_
Apps_21.1.0_linux_x64.run --mode unattended \ --optionfile ./installerSettings && \
/opt/Fortify/Fortify_SCA_and_Apps_21.1.0/bin/fortifyupdate && \ rm Fortify_SCA_and_Apps_21.1.0_
linux_x64.run fortify.license installerSettings ENTRYPOINT [ "/opt/Fortify/Fortify_SCA_and_Apps_
21.1.0/bin/sourceanalyzer" ]
```

注: SUSEのrpmでビルドされたパッケージがインストーラで必要です。

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 build -t <image_name>
```

コンテナの実行

このトピックでは、Fortify Static Code Analyzer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。

注: コンテナでFortify Static Code Analyzer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、Fortify Static Code Analyzer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。

Fortify Static Code Analyzer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SCAビルドセッションを保存し、出力ファイル(ログ、FPR)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、Fortify Static Code Analyzer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、/srcに入力ディレクトリ/sources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-scaです。

重要 Fortify Static Code Analyzerでコンテナ専用のメモリのみが検出および使用されるように、Fortify Static Code Analyzerの-fcontainerオプションを変換コマンドとスキャンコマンドの両方に含めます。それ以外の場合は、-autoheapが有効になるため、デフォルトでシステムメモリの合計がFortify Static Code Analyzerで検出されます。

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Fortify Static Code Analyzerを実行します。

```
docker run -v /scratch_local/:/scratch_docker -v /sources/:/src -it
fortify-sca –b MyProject -project-root /scratch_docker -fcontainer [<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Fortify Static Code Analyzerを実行します。

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

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

Fortify Static Code Analyzerとアプリケーションのアップグレードについて

Fortify Static Code Analyzerとアプリケーション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、*<sca_install_dir>*`/Core/config`ディレクトリにあるFortify Static Code Analyzer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。

- *<sca_install_dir>*`/Core/config/rules`フォルダ
- *<sca_install_dir>*`/Core/config/customrules`フォルダ
- *<sca_install_dir>*`/Core/config/ExternalMetadata`フォルダ
- *<sca_install_dir>*`/Core/config/CustomExternalMetadata`フォルダ
- *<sca_install_dir>*`/Core/config/server.properties`ファイル
- *<sca_install_dir>*`/Core/config/scales`フォルダ

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、「"Fortify Static Code Analyzerとアプリケーションのアンインストールについて" 次のページ」を参照してください。

注: 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起動されます。Fortify Secure Code Pluginsからソースコードをスキャンしても、最近インストールされたバージョンのFortify Static Code

Analyzerが使用されます。

Fortify Extension for Visual Studioのアップグレードに関するメモ

管理特権を持っているユーザがサポートされているバージョンのVisual Studio用に以前のバージョンのFortify Static Code Analyzerからアップグレードする場合は、インストーラで既存のMicro Focus Fortify Extension for Visual Studioが上書きされます。管理特権なしで以前のバージョンがインストールされている場合は、Fortify Extension for Visual Studio管理特権を必要とせずにインストーラで既存のバージョンが上書きされます。

注: 管理特権を持っていないユーザが管理特権が付与されたユーザアカウントを使用して以前にインストールされたFortify Extension for Visual Studioをアップグレードする場合は、まず管理特権アカウントを使用してVisual StudioからFortify Extension for Visual Studioをアンインストールする必要があります。

Fortify Static Code Analyzerとアプリケーションのアンインストールについて

このセクションでは、Fortify Static Code Analyzerとアプリケーション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、アンインストールをサイレントで実行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。

Fortify Static Code Analyzerおよびアプリケーションのアンインストール

Windowsプラットフォームでのアンインストール

Fortify Static Code Analyzerおよびアプリケーションソフトウェアをアンインストールするには、次の手順を実行します。

1. [**スタート**]> [**コントロールパネル**]> [**プログラムの追加と削除**]の順に選択します。
2. プログラムのリストから [**Fortify SCA and Applications <*version*>**]を選択し、 [**削除**]をクリックします。
3. 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。
   - [**Yes**]をクリックして、アンインストールするFortify Static Code Analyzerのバージョンとともにインストールされたツールのアプリケーション設定フォルダを削除します。Fortify Static Code Analyzer (`sca<version>`)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。

他のプラットフォームでのアンインストール

LinuxおよびFortify Static Code Analyzerプラットフォーム上のmacOSソフトウェアをアンインストールするには、次の手順を実行します。

1. 作成した重要なファイルも含めて、設定をバックアップします。
2. ご使用のオペレーティングシステムの*<sca_install_dir>*にあるアンインストールコマンドを実行します。

Linux	`Uninstall_FortifySCAandApps_<version>`
macOS	`Uninstall_FortifySCAandApps_<version>.app`

3. 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。
   - 「**Yes**」をクリックして、アンインストールするFortify Static Code Analyzerのバージョンとともにインストールされたツールのアプリケーション設定フォルダを削除します。Fortify Static Code Analyzer (`sca<version>`)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。

Fortify Static Code Analyzerとアプリケーションのサイレントアンインストール

Fortify Static Code Analyzer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. オペレーティングシステムに基づいて、次のコマンドのいずれかを入力します。

Windows	`Uninstall_FortifySCAandApps_<version>.exe --mode unattended`
Linux	`./Uninstall_FortifySCAandApps_<version> --mode unattended`
macOS	`Uninstall_FortifySCAandApps_<version>.app/Contents/MacOS/installbuilder.sh --mode unattended`
AIXおよびSolaris	`./Uninstall_FortifySCA_<version> --mode unattended`

注: Windows、Linux、およびmacOSの場合は、アンインストールするFortify Static Code Analyzerバージョンでインストールされたツールのアプリケーション設定フォルダがアンインストーラで削除されます。

Windows以外のプラットフォームでのFortify Static Code Analyzerとアプリケーションのテキストベースモードのアンインストール

Fortify Static Code Analyzerをテキストベースモードでアンインストールするには、次のように、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。

1. インストールディレクトリに移動します。
2. オペレーティングシステムに基づいて、次のコマンドのいずれかを入力します。

Linux	`./Uninstall_FortifySCAandApps_<version> --mode text`
macOS	`Uninstall_FortifySCAandApps_<version>.app/Contents/MacOS/installbuilder.sh --mode text`
AIXおよびSolaris	`./Uninstall_FortifySCA_<version> --mode text`

インストール後のタスク

インストール後のタスクでは、Fortify Static Code Analyzerとツールの使用を開始する準備をします。

インストール後処理ツールの実行

Fortify Static Code Analyzerインストール後処理ツールを実行するには、次の手順を実行します。

1. コマンドラインから*<sca_install_dir>*/binディレクトリに移動します。
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。

プロパティファイルの移行

以前のバージョンのFortify Static Code Analyzer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のFortify Static Code Analyzer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. コマンドラインから*<sca_install_dir>*/binディレクトリに移動します。
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、SCA 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。

ロケールの指定

Fortify Static Code Analyzerインストールのデフォルトロケールは英語です。

Fortify Static Code Analyzer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. コマンドラインからbinディレクトリに移動します。
2. コマンドプロンプトで、「scapostinstall」と入力します。
3. 「2」を入力してSettingsを選択します。
4. 「1」を入力してGeneralを選択します。
5. 「1」を入力して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。
   - 英語: en
   - スペイン語: es
   - 日本語: ja
   - 韓国語: ko
   - ポルトガル語(ブラジル): pt_BR
   - 簡体字中国語: zh_CN
   - 繁体字中国語: zh_TW

セキュリティコンテンツ更新の設定

Micro Focus Fortify Software 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。

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

1. コマンドラインからbinディレクトリに移動します。
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ルールパック更新サーバのURLを変更するには、「1」と入力してから、URLを入力します。
   Fortifyルールパック更新サーバのデフォルトのは`https://update.fortify.com`です。
6. Fortify 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。
   a. 「2」と入力してProxy Server Hostを選択してから、プロキシサーバの名前を入力します。
   b. 「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。
   c. (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。

Fortify Software Security Centerへの接続の設定

Micro Focus Fortify Software Security Centerに接続する方法を指定します。ネットワークでFortify Software Security Center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。

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

1. コマンドラインからbinディレクトリに移動します。
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、Fortify Software Security Center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。
   a. 「2」と入力してProxy Serverを選択してから、プロキシサーバのパスを入力します。
   b. 「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。
   c. 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。
   - Fortify Software Security Centerサーバからセキュリティコンテンツを更新するかどうか(オプション6)
   - Fortify Software Security Centerのユーザ名(オプション7)

プロキシサーバ設定の削除

以前にFortify Security Contentアップデートサーバまたは Micro Focus Fortify Software Security Center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。

Fortify Security ContentアップデートまたはFortify Software Security Centerのプロキシ設定を削除するには、次の手順を実行します。

1. コマンドラインからbinディレクトリに移動します。
2. コマンドプロンプトで、「scapostinstall」と入力します。
3. 「2」を入力してSettingsを選択します。
4. 「2」を入力してFortify Updateを選択するか、「Software Security Center Settings」を入力して3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、「-」(ハイフン)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。

信頼された証明書の追加

Fortify Static Code AnalyzerとそのツールからFortifyの他の製品および外部システムに接続するために、HTTPS経由の通信が必要である場合があります。次に例をいくつか示します。

- Fortify Static Code Analyzer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、「"fortify-sca.properties" ページ196」を参照してください。
- fortifyupdateユーティリティでは、Windowsシステムのインストール時に自動的にHTTPS接続を使用するか、手動でHTTPS接続を使用して(「"Fortify Security Contentの手動インストール" ページ35」を参照)、Fortify Security Contentが更新されます。
- 通常、Fortify Static Code Analyzerツール(Fortify Audit Workbench、Fortify Scan Wizard、Fortify Extension for Visual Studioなど)とFortify Software Security Centerが通信するには、HTTPS接続が必要です。デフォルトでは、これらのツールで自己署名された証明書またはローカルで署名された証明書が信頼されません。
- ScanCentral SASTセンサとして設定されたFortify Static Code Analyzerとコントローラは、HTTPS接続を使用して通信します。

HTTPSを使用すると、Fortify Static Code AnalyzerとそのツールではデフォルトでSSLサーバ証明書に標準のチェック(証明書が信頼されるかどうかを確認するチェックなど)が適用されます。組織が独自の認証局(CA)を実行し、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Fortify Static Code Analyzerツールで信頼される必要がある場合は、そのCAが信頼されるようにFortify Static Code Analyzerツール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Fortify Static Code Analyzerキーストアに追加する必要があります。Fortify Static Code Analyzerキーストアは、*<sca_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Fortify Static Code Analyzer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。

   ```
   <sca_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。

   注: 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。

第3章: 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。

分析プロセス

分析プロセスは、次の4つの個別のフェーズで構成されます。

1. **ビルド統合** - Fortify Static Code Analyzerをビルドツールに統合するかどうかを選択します。ビルド統合のオプションについては、「"ビルドへの統合" ページ118」を参照してください。
2. **変換** - 一連のコマンドを使用してソースコードを収集し、ビルドIDに関連付けられた中間形式に変換します。通常、ビルドIDは変換するプロジェクトの名前です。詳細については、「"変換フェーズ" 次のページ」を参照してください。
3. **分析** - 変換フェーズで特定されたソースファイルをスキャンし、分析結果ファイルを通常はFortify Project Results (FPR)形式で生成します。FPRファイルのファイル拡張子は`.fpr`です。詳細については、「"分析フェーズ" ページ49」を参照してください。
4. **変換と分析の検証** - ソースファイルが適切なルールパックを使用してスキャンされ、エラーがレポートされていないことを検証します。詳細については、「"変換フェーズと分析フェーズの検証" ページ52」を参照してください。

コードの変換と分析に使用する一連のコマンドの例は次のとおりです。

```
sourceanalyzer -b <build_id> -clean
sourceanalyzer -b <build_id> ...
sourceanalyzer -b <build_id> -scan -f MyResults.fpr
```

この例の3つのコマンドは、分析プロセスの次のステップを示しています。

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

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

2. プロジェクトコードを変換します。
   このステップは、同じビルドIDを持つsourceanalyzerの複数の呼び出しで構成できます(JavaScript/TypeScript、PHP、Python、Rubyなどの動的言語を除く)。
3. プロジェクトコードを分析し、Fortify Project Resultsファイル(FPR)を生成します。

パラレル処理

大規模なプロジェクトのスキャン時間を短縮するために、Fortify Static Code Analyzerは並行分析モードで実行されます。これにより、システムで使用できるCPUコアをすべて活用できます。Fortify Static Code Analyzer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、Fortify Static Code Analyzer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。

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。各ソースコードタイプの章では、特定の要件について説明します。

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Fortify Static Code Analyzerの呼び出しで構成されます。Fortify Static Code Analyzer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。

注意 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。Fortify Static Code Analyzerでは、その後の呼び出し時にビルドIDに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。

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
```

次の章では、さまざまなソースコードタイプを変換する方法について説明します。

モバイルビルドセッション

Fortify Static Code Analyzer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、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。スキャン時間を短縮するために、ビルドコンピュータで変換を実行してから、よりスキャンに適したコンピュータにビルドセッション(MBSファイル)を移動できます。開発者は自分のコンピュータで変換を実行し、1台の強力なコンピュータのみを使用して大規模なスキャンを実行できます。

変換を実行しているシステムと分析を実行しているシステムの両方に同じバージョンのFortify Security Content (Rulepack)がインストールされている必要があります。

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Fortify Static Code Analyzerバージョンと分析コンピュータ上のFortify Static Code Analyzer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>* (21.2.0.0240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Fortify Static Code Analyzer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、21.2.0と21.1.xには互換性があります。

Fortify Static Code Analyzer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。

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Fortify Static Code Analyzer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。

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Fortify Static Code Analyzerのプロジェクトルートディレクトリにインポートします。

注: 必要に応じて、次のコマンドを使用してMBSファイルからビルドIDとFortify Static Code Analyzerバージョンを取得できます。

```
sourceanalyzer -import-build-session <file>.mbs -
Dcom.fortify.sca.ExtractMobileInfo=true
```

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
```

Fortify Static Code Analyzer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Fortify Static Code Analyzer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" 下」を参照)。

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。

分析フェーズは、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" ページ128」および「"出力オプション" ページ131」を参照)。

分析フェーズの基本的なコマンドライン構文の例は次のとおりです。

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

注: デフォルトでは、Fortify Static Code Analyzer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
```

ウィルス対策ソフトウェアを使用すると、Fortify Static Code Analyzerのパフォーマンスが悪影響を受ける可能性があります。Fortifyでは、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Fortify Static Code Analyzer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、Fortify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、Fortify Static Code Analyzerの内部ファイルが次の場所に作成されます。

- Windowsの場合: `c:\Users\<username>\AppData\Local\Fortify\sca<version>`
- Windows以外の場合: `<userhome>/.fortify/sca<version>`

ここで、*<version>*は使用しているFortify Static Code Analyzerのバージョンです。

高次分析

高次分析(HOA)では、高次コードを使用してデータフローを追跡する機能が改善されます。高次コードでは、関数が値として操作され、匿名関数式(ラムダ式)を使用して関数が生成され、引数として渡され、値として返され、オブジェクトの変数およびフィールドに割り当てられます。これらのコードパターンは、JavaScript、TypeScript、Python、Ruby、Swiftなどの最新の動的言語で共通です。

デフォルトでは、JavaScript、TypeScript、Python、Ruby、Swiftの各コードをスキャンすると、Fortify Static Code Analyzerで高次分析が実行されます。高次分析のプロパティについては、「"fortify-sca.properties" ページ196」を参照し、「高次分析」を検索してください。

モジュラー分析

このリリースには、モジュラー分析の技術プレビューが含まれています。モジュラー分析では、コアプロジェクトとは別にライブラリ(とサブライブラリ)を事前にスキャンできます。その後、コアプロジェクトをスキャンするときに、これらの事前にスキャンされたライブラリを含めることができます。これにより、コアプロジェクトをスキャンするたびにライブラリを再スキャンしないため、コアプロジェクト分析のパフォーマンスが改善される可能性があります。モジュラー分析では、ライブラリのソースコード、Fortify Static Code Analyzerで変換されたファイル、またはライブラリのスキャンに使用されるカスタムルールを必要とせずに、ライブラリが参照されるプロジェクトをスキャンすることもできます。これにより、必要なことはコアアプリケーションで問題を監査することのみであるという追加の利点が得られます。分析結果は、直接制御するコードに合わせてより合理化されるため、所有していないコードの問題を心配する必要がありません。

現在、モジュラー分析はJavaおよびJava EEで開発されたライブラリおよびアプリケーションで使用できます。

注: このリリースでは、モジュラー分析によるパフォーマンスの改善が見られない可能性もあります。Fortifyでは、今後のリリースでモジュラー分析のパフォーマンスが最適化される予定です。

次の操作のたびに、ライブラリを再スキャンする必要があります。

- 新しいバージョンのFortify Static Code Analyzerに更新する
- Fortify Security Contentを更新する
- ライブラリを変更する

モジュラーコマンドラインの例

ライブラリを個別に変換およびスキャンするには、次のコマンドを入力します。

```
sourceanalyzer -b LibA MyLibs/A/*.java

sourceanalyzer -b LibA -scan-module
```

コアプロジェクトを変換およびスキャンし、事前にスキャンされた複数のライブラリを含めるには、次のコマンドを入力します。

```
sourceanalyzer -b MyProj MyProj/*.java

sourceanalyzer -b MyProj -scan -include-modules LibA,LibB
```

この例で示したオプションについては、「"分析オプション" ページ128」を参照してください。

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する機能が提供されます。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。Configuration Analyzerには、正規表現分析機能が含まれています。

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。特に変換から除外されていなければ、検出された各ディレクトリのファイルはすべて分析されます。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" ページ126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
```

変換フェーズと分析フェーズの検証

Micro Focus Fortify Audit Workbenchの結果証明書には、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、Fortify Static Code Analyzer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
- Fortify Static Code Analyzer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

注: 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、*Micro Focus Fortify Audit Workbenchユーザガイド*を参照してください。

第4章: Javaコードの変換

このセクションでは、Javaコードを変換する方法について説明します。

Fortify Static Code Analyzerでは、Java EEアプリケーション(JSPファイル、環境設定ファイル、展開ディスクリプタを含む)、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。

Javaコマンドライン構文

Javaコードを変換するには、ライブラリで定義され、コード内で参照されるすべてのタイプが、ソースコード、クラスファイル、またはJARファイルに対応する定義を持っている必要があります。すべてのソースファイルをFortify Static Code Analyzerコマンドラインに含めてください。

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Fortify Static Code Analyzer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コマンドラインオプション" 次のページ」を参照してください。

Java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> <files>
```

Javaコードを使用すると、Fortify Static Code Analyzerではのいずれかを実行できます。

- コンパイラをエミュレートする。ビルドの統合に便利な場合があります
- ソースファイルを直接受け入れる。コマンドラインスキャンで便利です

AntとFortify Static Code Analyzerの統合の詳細については、「"Antの統合" ページ120」を参照してください。

Fortify Static Code Analyzer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>]
```

ファイルをFortify Static Code Analyzerに直接渡すには、次のコマンドを入力します。

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

ここで:

- *<translation_options>*は、コンパイラに渡されるオプションです。
- `-cp` *<classpath>*は、Javaソースコード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、Fortify Static Code Analyzer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、Fortify Static Code Analyzer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、Fortify Static Code Analyzerでは`A.jar`からの`MyData.class`を使用します。

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

  Fortifyでは、`-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。
  Fortify Static Code AnalyzerではJARファイルを次の順序でロードします。
  a. `-cp`オプションから
  b. `jre/lib`から
  c. *<sca_install_dir>*`/Core/default_jars`から
  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コマンドラインオプション" 下を参照してください。

Java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Java EE用)について説明します。

Java/Java EEオプション	説明
`-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。 • 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。

Java/Java EEオプション	説明
	• 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`

`-cp <dirs>` |
`-classpath <dirs>`
	Java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。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、Fortify Static Code Analyzer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" ページ138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。
`-source <version>` | `-jdk <version>`	Javaコードを記述する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、*Micro Focus Fortifyソフトウェアシステム要件*のドキュメントを参照してください。デフォルトはJava 1.8です。 **同等のプロパティ名:** `com.fortify.sca.JdkVersion`
`-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

Java/Java EEオプション	説明
	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のコマンドライン例

クラスパスとして`javaee.jar`が指定された1つのファイル`MyServlet.java`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
```

`lib`ディレクトリ内のすべてのJARファイルをクラスパスとして使用して`src`ディレクトリ内のすべての`.java`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
```

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
```

Javaの警告

Javaに関する次の警告が表示される場合があります。

```
Unable to resolve type...

Unable to resolve function...

Unable to resolve field...

Unable to locate import...
```

```
Unable to resolve symbol...

Multiple definitions found for function...

Multiple definitions found for class...
```

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。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。

Java EEアプリケーションの変換

Java EEアプリケーションを変換するには、Fortify Static Code AnalyzerでJavaソースファイルおよびJava EEコンポーネント(JSPファイル、展開ディスクリプタ、および環境設定ファイルなど)を処理します。Java 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。

Javaファイルの変換

Java 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" 前のページを参照してください。

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

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、および展開ディ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。

Java EE変換の警告

Java 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`.jar`ファイルおよび`.class`ファイルが正しい`WEB-INF/lib`および`WEB-INF/classes`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。

Javaバイトコードの変換

Fortifyでは、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。

ソースコードの変換に加えて、プロジェクト内のJavaバイトコードを変換できます。バイトコードを変換するには、次の2つのオプションがあります。

- Fortify Static Code Analyzer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。

  バイトコードを変換用に逆コンパイルするには、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コマンドラインにこのプロパティを含めます)。

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

- Fortify Static Code Analyzer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、Fortifyでは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。

  バイトコードをFortify Static Code Analyzer変換に含めるには:

  a. `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、Fortify Static Code Analyzerで`.class`および`.jar`ファイルを処理する方法を指定します。

b. Fortify Static Code Analyzer変換フェーズで、変換するJavaバイトコードファイルを指定します。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"
```

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の変換とスキャン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

Fortify Static Code Analyzer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。Fortify Static Code Analyzer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Fortify Static Code Analyzer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Fortify Static Code Analyzerを再実行します。

Java EEアプリケーションのスキャンの詳細については、"Java EEアプリケーションの変換" ページ57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Fortify Static Code Analyzer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。

第5章: Kotlinコードの変換

このセクションでは、Kotlinコードを変換する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。

Kotlinコマンドライン構文

Kotlinコードの変換は、Javaコードの変換と同様です。Kotlinコードを変換するには、ライブラリで定義され、コード内で参照されるすべてのタイプが、ソースコード、クラスファイル、またはJARファイルに対応する定義を持っている必要があります。すべてのソースファイルをFortify Static Code Analyzerコマンドラインに含めてください。

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>
```

ここで

- `-cp <classpath>`は、Kotlinソースコード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。
  Fortify Static Code Analyzer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、Fortify Static Code Analyzer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、Fortify Static Code Analyzerでは`A.jar`からの`MyData.class`を使用します。

  ```
  sourceanalyzer -cp "A.jar:B.jar" myfile.kt
  ```

  `-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。

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、「"Kotlinコマンドラインオプション" 次のページ」を参照してください。

Kotlinコマンドラインオプション

次の表では、Kotlin固有のコマンドラインオプションについて説明します。

Kotlinオプション	説明

`-cp <paths>` | `-classpath <dirs>`	Kotlinソースコードの変換に使用するクラスパスを指定します。これは、セミコロン区切りまたはコロン区切りのディレクトリのリストです。次の例に示すように、Fortify Static Code Analyzer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" ページ138を参照してください。
	同等のプロパティ名:
	`com.fortify.sca.JavaClasspath`

`-source <version>` | `-jdk <version>`	Kotlinコードを記述する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、*Micro Focus Fortifyソフトウェアシステム要件*のドキュメントを参照してください。デフォルトはJava 1.8です。 **同等のプロパティ名:** `com.fortify.sca.JdkVersion`
`-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Java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`

Kotlinコマンドラインの例

クラスパスとしてA.jarが指定された1つのファイルMyKotlin.kt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
```

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

```
sourceanalyzer -b MyProject gradlew clean assemble

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

lib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src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"
```

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Fortify Static Code Analyzer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、「"Kotlinコマンドラインオプション" 前のページ」を参照してください。

Kotlinスクリプトの変換

Fortify Static Code Analyzer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`ファイル拡張子に基づいてスクリプトに適用されます。Fortify Static Code Analyzer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。

第6章: Visual StudioおよびMSBuildプロジェクトの変換

Fortify Static Code Analyzerは、Visual StudioまたはMSBuildでビルドされた次のタイプのプロジェクトの変換をサポートします。

- C/C++プロジェクト
- C#またはVisual Basicで作成された.NETプロジェクト(VB.NET)

  これには、.NET、.NET Framework、.NET Core、および.NET Standardをターゲットとするプロジェクトが含まれます。
- ASP.NETアプリケーション

  これには、ASP.NET Core Frameworkを使用するアプリケーションが含まれます。
- 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のサポート対象バージョンの一覧とVisual StudioおよびMSBuildについては、*Micro Focus Fortifyソフトウェアシステム要件*ドキュメント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。

Visual StudioおよびMSBuildプロジェクト変換の前提条件

重要 Fortify Static Code Analyzerは、Windowsシステム上のVisual StudioおよびMSBuildプロジェクトの変換のみをサポートしています。

Fortifyでは、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。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。

Visual StudioおよびMSBuilプロジェクト変 換コマンドライン構 文

Visual StudioまたはMSBuildプロジェクトを変 換 する基 本 的 な構 文 は、プロジェクトをビルドするMSBuildコマンドをFortify Static Code Analyzerコマンドに追 加 します。次 のコマンドは、`Sample.sln`という名 前 のVisual Studioソリューションを変 換 します。

```
sourceanalyzer –b <build_id> msbuild /t:rebuild Sample.sln
```

このコマンドは、ソリューションまたはプロジェクトをビルドおよび変 換 します。Fortifyでは、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。

重 要 Visual Studio環 境 の開 発 者 コマンドプロンプトから変 換 する場 合 、Fortifyでは、Fortify Static Code Analyzer変 換 を実 行 する前 に`dotnet restore`コマンドを実 行 するよう推 奨 しています。このコマンドは、プロジェクトの最 上 位 フォルダから実 行 する必 要 があります。これにより、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 できます。

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

Visual StudioおよびMSBuildプロジェクトの変 換 に関する特 殊 なケースの処 理

スクリプトからの変 換 の実 行

前 に説 明 したように、Fortifyでは、Visual StudioおよびMSBuildプロジェクトのFortify Static Code Analyzer変 換 をVisual Studioの開 発 者 コマンドプロンプトから実 行 することをお勧 めします。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、Fortify Static Code Analyzer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。

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。

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Micro Focus Fortify Extension for Visual Studioから変換する必要があります。

注: 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。

空白を含む設定を持つプロジェクトの変換

プロジェクトが空白を含む環境設定ファイルまたはその他の設定ファイルを使用してビルドされている場合は、MSBuildコマンドで必ず次のように指定してください。

- 設定値を引用符で囲む(該当するコマンドラインオプションを囲む引用符に加えて)
- 引用符をエスケープする

たとえば、環境設定`My Configuration`を使用してビルドされたVisual Studioソリューション`Sample.sln`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
```

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
```

MSBuildコマンドで複数のターゲットやプロジェクトを使用する

MSBuildの最新バージョンでは、/tオプションの拡張構文を使用して複数のターゲットや特定のプロジェクトをビルドできます。しかし、Fortifyでは、Visual StudioまたはMSBuildプロジェクトを変換する場合にこの構文を使用しないことをお勧めします。サポートされているVisual StudioおよびMSBuildプロジェクトを変換するには、1つのrebuildターゲットを指定するだけで十分です。

1つのrebuildターゲットを使用してプロジェクトを変換できない場合、FortifyではMSBuildコマンドに指定された複数のターゲットおよびプロジェクトを限定的にサポートしています。この機能を使用するには、次の例に示すように、Fortify Static Code Analyzerコマンドに-multiple-msbuild-targetsオプションを追加します。

```
sourceanalyzer –b MySampleProj -multiple-msbuild-targets msbuild
/t:Project1:build;Project2:build Sample.sln
```

注: この変換モードのサポートは限定的であり、Fortify Static Code Analyzerでターゲットとプロジェクトのすべての組み合わせを適切に処理できるとは限りません。

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(ファイル名の拡張子*.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。Fortify 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Sample2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Sample.sln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。

```
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 -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" ページ128を参照してください。

Visual StudioおよびMSBuild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およびMSBuildプロジェクトを変換する別の方法について説明します。

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別の2つの変換方法を次に示します。

- Micro Focus 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。
- Fortify Static Code Analyzerコマンドにdevenvコマンドを追加する
  次のコマンドは、`Sample.sln`という名前のVisual Studioソリューションを変換します。

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

  Fortify Static Code Analyzer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。

Fortify Static Code Analyzerを明示的に実行せずに変換する

Fortify Static Code Analyzerを直接起動せずにVisual StudioまたはMSBuild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、*<sca_install_dir>*`\Core\private-bin\sca\MSBuildPlugin`ディレクトリ内にある`Fortify.targets`ファイルが必要です。プロジェクトをビルドするMSBuild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。例:

```
msbuild /t:rebuild /p:CustomAfterMicrosoftCommonTargets=<sca_install_
dir>\Core\private-bin\sca\MSBuildPlugin\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、Fortify Static Code Analyzer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	デフォルト値
FORTIFY_MSBUILD_BUILDID	変換のFortify Static Code Analyzer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、Fortify Static Code Analyzer `-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、Fortify Static Code Analyzer `–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、Fortify Static Code Analyzer `–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	メモリ要件の変換を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
FORTIFY_MSBUILD_SCALOG	Fortify Static Code Analyzerログファイルの場所(絶対パス)を指定します。 これは、Fortify Static Code Analyzer `-logfile`オプションと同じです。	`%LOCALAPPDATA%/Fortify/sca/log/sca.log`

第7章: CおよびC++コードの変換

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。

重要 この章では、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" ページ65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。

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。Fortify Static Code Analyzer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。

注: Fortify Static Code Analyzerは、必ずしもすべての標準以外のC++構造体をサポートしていません。

Gradleを使用してC++プロジェクトをビルドする場合は、C++アプリケーションプラグインが次のいずれかの形式でGradleファイルに追加されていることを確認してください。

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

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

Gradleとの統合について詳しくは、"ビルド統合" ページ118を参照してください。

CおよびC++のコマンドライン構文

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。Fortify Static Code Analyzer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>*は、Fortify Static Code Analyzerに渡 されるオプションです。
- *<compiler>*は、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、*Micro Focus Fortifyソフト ウェアシステム要 件* のド キュメントを参 照 してください。
- *<compiler_options>*は、C/C++コンパイラに渡 されるオプションです。
- *<file>*.`c`は、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。

注 : Fortify Static Code Analyzer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Fortify Static Code Analyzer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
```

Fortify Static Code Analyzer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、`gcc`を実 行 して得 られるオ ブジェクトファイル`hello.o`の両 方 が生 成 されます。Fortify Static Code Analyzer `-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。

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Fortify Static Code Analyzer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Fortify Static Code Analyzer変 換 を呼 び出 す必 要 があります。Fortify Static Code Analyzer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。

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 、Fortify Static Code Analyzer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。

第8章: JavaScriptおよびTypeScriptコードの変換

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(ソースからソースにコンパイル)されます。この生成されたコードは最適化、圧縮、または両方が行われます。したがって、Fortify Static Code Analyzerによってレポートされるこのコードの脆弱性を修正するには困難であるため、変換から除外する必要がある場合があります。`-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手動で除外します。

Fortify Static Code Analyzerでは、圧縮されたJavaScript (*.min.js)を変換しません。

注: 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。Fortify Static Code Analyzerでは、その後の呼び出し時にビルドIDに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。

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_file_or_dir>*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。

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、fortify-sca.propertiesファイル内 の com.fortify.sca.skip.libraries.jQueryプロパティにjquery-us.jsが含 まれるため、Fortify Static Code Analyzer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。Fortify Static Code Analyzerでは com.fortify.sca.skip.libraries.jQuery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 '(-?\d+\.\d+\.\d+)?'を自 動 的 に挿 入します。

注 : -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" ページ126を 参 照 してください。

NPM依 存 関 係 の除 外

デフォルト で、Fortify Static Code Analyzer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Fortify Static Code Analyzer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、特 に com.fortify.sca.follow.importsプロパティによって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Fortify Static Code Analyzer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。

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、fortify-sca.propertiesファイルまたはコマンドラインで com.fortify.sca.EnableDOMModeling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。

注 : このオプションを有 効 にすると、Fortify Static Code Analyzer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、Fortify Static Code Analyzer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、Fortify Static Code Analyzerでは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
```

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、Fortify Static Code Analyzerでダウンロードして変 換 フェーズに含 めることができるドメインを指 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。

注 : 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"
```

注 : プロパティ値 では、ドメインからwww.プレフィクスを省 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。

第9章: Pythonコードの変換

Fortify Static Code Analyzerでは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。

このセクションでは、次のトピックについて説明します。

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>
```

注: 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。Fortify Static Code Analyzerでは、その後の呼び出し時にビルドIDに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。

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、Fortify Static Code Analyzer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。Fortify Static Code Analyzer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。

Fortify Static Code Analyzer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。Fortify Static Code Analyzerによって自動的に計算されます。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。

   Fortify Static Code Analyzer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)。Fortify Static Code Analyzerでは最 初 にFortify Static Code Analyzer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。Fortify Static Code Analyzer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. Fortify Static Code Analyzerが変 換 中 のファイルを含 む現 在 のディレクトリ。たとえば、Fortify Static Code Analyzer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。

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`オプションに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、次 の例 のように、複 数 のフォルダにネームスペースパッケージ`package_name`用 のサブパッケージが2つ含 まれるとします。

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

`-python-path`オプションには、`/path_1;/path_2`を含 めます。

PythonでのDjangoフレームワークの使 用

Fortify Static Code Analyzerでは、Djangoフレームワークをサポートしています。Djangoフレームワークを使 用して作 成 されたコードを変 換 するには、*<sca_install_dir>*`/Core/config/fortify-sca.properties`設 定 ファイルに次 のプロパティを追 加 します。

```
com.fortify.sca.limiters.MaxPassthroughChainDepth=8

com.fortify.sca.limiters.MaxChainDepth=8
```

デフォルト で、Fortify Static Code Analyzerではプロジェクト ルートフォルダ内 のDjangoテンプレートを検 出 しようとします。検 出 されたDjangoテンプレート は、自 動 的 に変 換 に追 加 されます。Fortify Static Code AnalyzerでDjangoテンプレートを自 動 的 に検 出 しないようにする場 合 は、`-django-disable-autodiscover`オプションを使 用 します。プロジェクト にDjangoテンプレート が必 要 なのに、Django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 は、`-django-disable-autodiscover`オプションに加 えて`-django-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コマンド に`-django-template-dirs`オプションを追 加して、Djangoテンプレートファイルの追 加 の場 所 を指 定 できます。

```
-django-template-dirs <dirs>
```

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
`-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`2`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。Fortify Static Code Analyzer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`
`-django-disable-autodiscover`	Fortify Static Code Analyzer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`
`-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

Pythonオプション	説 明
	(Windows)またはコロン区 切り(Windows以 外)リストを指 定しま す。Fortify Static Code Analyzer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`

Pythonのコマンドライン例

Python 3コードを変 換 するには、次 のコマンドを入 力します。

```
sourceanalyzer -b Python3Proj -python-version 3 -python-path
/usr/lib/python3.4:/usr/local/lib/python3.4/site-packages src/*.py
```

Python 2コードを変 換 するには、次 のコマンドを入 力します。

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

第10章: モバイルプラットフォームのコードの変換

Fortify Static Code Analyzerでは、次のモバイルアプリケーションソース言語の分析をサポートしています。

- Xcodeを使用して開発されるiOSアプリケーション用のSwift、Objective-C、およびObjective-C++
- Androidアプリケーション用のJava

Xamarinアプリケーションの変換の詳細については、"Visual StudioおよびMSBuildプロジェクトの変換" ページ65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。

Apple iOSプロジェクトの変換

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、Fortify Static Code Analyzer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。

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。Fortify Static Code Analyzer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" 次のページを参照)。
- 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。

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。

注 : 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" ページ 126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
```

その後、次の例に示すように、分析フェーズを実行できます。

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

Androidプロジェクトの変換

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。Fortify Static Code Analyzer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。

注 : Micro Focus Fortify Analysis Plugin for IntelliJ 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、*Micro Focus Fortify Plugins for JetBrains IDEsおよびAndroid Studioユーザガイド*を参照してください。

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。

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" ページ120を参照してください。

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、Fortify Static Code Analyzer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。

Fortifyでは、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、これらの問題が偽陽性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、"分析のフィルタリング" ページ184の説明に従って、問題をフィルタできます。インスタンスIDに基づいて問題をフィルタできます。

第11章: Goコードの変換

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。

このセクションでは、次のトピックについて説明します。

Goコマンドライン構文

最善の結果を得るため、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>
```

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
`-gopath <dir>`	プロジェクトのルートディレクトリを指定します。ディレクトリ構造がGoワークスペース階層に従っていることを確認してください (https://golang.org/doc/gopath_code.html)。このオプションを指定しない場合は、GOPATHシステム環境変数が使用されます。 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`

Goオプション	説 明
	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。 **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 、Fortify Static Code AnalyzerではFortify Static Code Analyzer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
-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 、Fortify Static Code Analyzerでは https://proxy.golang.org,directを使 用 します。 **同 等 のプロパティ名** com.fortify.sca.GOPROXY

依存関係の解決

Fortify Static Code Analyzer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

  Fortify Static Code Analyzer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。Fortify Static Code Analyzer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" ページ86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。
- 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。

第12章: Rubyコードの変換

このセクションでは、次のトピックについて説明します。

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
```

注: 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。Fortify Static Code Analyzerでは、その後の呼び出し時にビルドIDに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。

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
`-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" 次のページを参照)。 **同等のプロパティ名:** `com.fortify.sca.RubyLibraryPaths`

Rubyオプション	説明
`-rubygem-path <dirs>`	RubyGemsの場所へのパスを指定します("Gemパスの追加" 下を参照)。 **同等のプロパティ名:** `com.fortify.sca.RubyGemPaths`

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、`/usr/share/ruby/myPersonalLibrary`ディレクトリ内に`utils.rb`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
```

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
```

注: 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。

第13章: COBOLコードの変換

Fortify Static Code Analyzerでは、以前のリリースで更新されたCOBOLコードの変換が導入され、それがデフォルトの変換方法になりました。以前の変換方法(現在はレガシーCOBOL変換と呼ばれる)は、コマンドラインオプションを指定することで引き続き使用できます。次のいずれかの条件に当てはまる場合は、レガシーCOBOL変換方法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Fortify Static Code Analyzerを実行している
- 使用するCOBOL方言がサポートされていない

以下のセクションでは、デフォルトのCOBOLコード変換について説明します。レガシーCOBOL変換にのみ関連する情報については、その旨を明示します。

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、*Micro Focus Fortifyソフトウェアシステム要件*ドキュメントを参照してください。Fortify Static Code Analyzer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。

注: Fortify Static Code AnalyzerでCOBOLをスキャンするには、COBOLスキャン機能を明確に含むFortify Static Code Analyzerライセンスファイルが必要です。COBOLのスキャンと必要なライセンスファイルについて詳しくは、Micro Focus Fortify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。

COBOLソースファイルとコピーブックファイルの変換準備

Fortify Static Code Analyzerでは、Windowsシステム上でのCOBOLソースファイルの変換のみをサポートしています。

レガシーCOBOL変換: Fortify Static Code Analyzerでは、*Micro Focus Fortifyソフトウェアシステム要件*ドキュメントに記載されているサポート対象プラットフォームおよびアーキテクチャでのCOBOLソースファイルの変換をサポートしています。

COBOLプログラムを分析する前に、Fortify Static Code Analyzer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

  Fortifyでは、COBOLソースコードファイルに拡張子 .CBL、.COB、.cbl、または.cobを使用することを推奨します。したがって、「"ファイル拡張子を持たないCOBOLソースファイルの変換" 次のページ」および「"ファイルとディレクトリの指定" ページ138」で説明されているとおり、コマンドラインから拡張子の指定を管理する必要はありません。

  注: Fortify Static Code Analyzerでは、ファイル拡張子の有無に関係なく、COBOLソースコードファイルを変換します。

- 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
- COBOLソースコードで参照する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)

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

レガシーCOBOL変換: Fortify Static Code Analyzer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変換します。

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。

レガシーCOBOL変換:

- 対応するコピーブックファイルは、ファイル拡張子の有無に関係なく、名前がFOOになります。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。詳しくは、"レガシーCOBOL変換コマンドラインオプション" ページ94を参照してください。
- COPYコマンドでは、ファイル名の代わりにディレクトリ-ファイル-パス構造を受け入れることもできます。

Fortifyでは、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。

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
```

レガシーCOBOL変換: 自由形式のCOBOLがデフォルトの変換モードです。Fortify Static Code Analyzerは固定形式COBOLの変換をサポートしています。固定形式のCOBOLを変換するには、`-fixed-format`コマンドラインオプションを指定する必要があります。詳しくは、"レガシーCOBOL変換コマンドラインオプション" 次のページを参照してください。

参照情報

"ファイルとディレクトリの指定" ページ138

ファイル拡張子を持たないCOBOLソースファイルの変換

(COBOLファイル名として一般的な)`.COB`または`.CBL`ファイル拡張子を持たないCOBOLソースファイルをメインフレームから取得した場合は、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。

```
-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
```

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" ページ138を参照)。

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。

COBOLオプション	説明
`-copydirs <dirs>`	Fortify Static Code Analyzer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。 **同等のプロパティ名:** `com.fortify.sca.CobolCopyDirs`
`-cobol-dialect <dialect>`	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。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。 **注:** このオプションは、Micro Focus Server Expressの上級ユーザが対象です。 **同等のプロパティ名:** `com.fortify.sca.CobolCheckerDirectives`

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
`-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>`	Fortify Static Code Analyzer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

レガシーCOBOLオプション	説 明
	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`
`-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Fortify Static Code Analyzerに指 示 する、固 定 形 式 のCOBOLを指 定 します。 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 • Fortify Static Code Analyzer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`

第14章: ApexおよびVisualforceコードの変換

このセクションでは、次のトピックについて説明します。

Apex変換の前提条件

- Fortify Static Code Analyzerをインストールした同じコンピュータ上で、スキャンするすべてのソースコードが使用可能であること。

  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

  Salesforce Webサイトで入手できるForce.com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。
- アプリをサポートできるように標準のSalesforceデータベース構造をカスタマイズした場合は、変更点の説明もダウンロードして、変更されたバージョンのSalesforceとアプリのやり取りをFortify Static Code Analyzerが認識できるようにする必要があります。"カスタマイズされたSalesforceデータベース構造情報のダウンロード" ページ98を参照してください。

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> -apex <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。

重 要 ソースコードファイルとしてサポートされているファイル拡 張 子 は、.cls、.trigger、.page、および.componentです。

すべてのApexおよびVisualforce固 有 のコマンドラインオプションについて詳 しくは、"ApexおよびVisualforceのコマンドラインオプション" 下 を参 照 してください。

ApexおよびVisualforceのコマンドラインオプション

次 の表 では、ApexおよびVisualforce変 換 のコマンドラインオプションについて説 明 します。

ApexまたはVisualforceのオプション	説 明
-apex	.cls拡 張 子 を持 つファイルに対 してApexおよびVisualforce変 換 を使 用 するようにFortify Static Code Analyzerに指 示 します。このオプションを指 定 しない場 合 、Fortify Static Code Analyzerは*.clsファイルをVisual Basicコードとして変 換 します。 **注 :** または、コマンドライン(-Dcom.fortify.sca.fileextensions.cls=APEXを含 める)または*<sca_install_dir>*/Core/config/fortify-sca.propertiesファイルでcom.fortify.sca.fileextensions.clsプロパティをAPEXに設 定 することもできます。 **同 等 のプロパティ名 :** com.fortify.sca.Apex
-apex-sobject-path *<path>*	カスタムsObject JSONファイル(sobjects.json)の場 所 を指 定 します。 sf_extractorツールの使 用 方 法 については、"カスタマイズされたSalesforceデータベース構 造 情 報 のダウンロード" 次 のページを参 照 してください。

ApexまたはVisualforceのオプション	説 明
	同 等 のプロパティ名 : `com.fortify.sca.ApexObjectPath`

カスタマイズされたSalesforceデータベース構 造 情 報 のダウンロード

カスタマイズされたSalesforceデータベース構 造 の説 明 をダウンロード するには、sf_extractor ツールを使 用します。Fortify Static Code Analyzerでより詳 しい分 析 を実 行 するには、この情 報 が必 要 です。sf_extractorは、sourceanalyzerの変 換 フェーズに含 めるカスタムsObject JSONファイルを作 成します。(この情 報 をFortify Static Code Analyzerに提 供 する方 法 については、"ApexおよびVisualforceのコマンドラインオプション" 前 のページを参 照してください)。

次 の表 では、*<sca_install_dir>*/Toolsにある`sf_extractor.zip`ファイルの内 容 について説 明しています。

フォルダまたはファイル名	説 明
lib	JAR依 存 関 係 を含 むフォルダ
src	ソースコード
partner.wsdl	Partner WSDLファイルバージョン37.0
sf_extractor.jar	コンパイルされたJARファイル(依 存 関 係 を含 む)

sf_extractorを実 行 するためのコマンドラインは次 のとおりです。

```
java -jar sf_extractor.jar <username> <password> <security_token> <org>
```

ここで:

- *<username>*は、Salesforceクラウド のユーザ名 です。たとえば、`test@test.test`などです。
- *<password>*は、Salesforceクラウド のパスワード です。
- *<security_token>*は、25文 字 の英 数 字 からなるセキュリティトークンです。
- *<org>*は、サンドボックス組 織 を使 用している場 合 はy、運 用 組 織 を使 用している場 合 はn です。

sf_extractorツールは、資 格 情 報 を使 用してSalesforceのSOAP APIにアクセスします。現 在 の組 織 から追 加 情 報 を含 むすべてのsObjectをダウンロードした後 、sObject内 のフィールド に関 する情 報 をダウンロードします。これは、現 在 の組 織 で表 されている型 を適 切 に解 決 するために必 要 です。

このツールは、変 換コマンド で`-apex-sobject-path`オプションを使 用してFortify Static Code Analyzerに提 供 する`sobjects.json`ファイルを生 成します。

第15章: その他の言語および設定の変換

このセクションでは、次のトピックについて説明します。

PHPコードの変換

`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" 次のページ」の説明を参照してください。

注: 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。Fortify Static Code Analyzerでは、その後の呼び出し時にビルドIDに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。

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
`-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7.4です。有 効 なバージョンのリストについては、*Micro Focus Fortifyソフトウェアシステム要 件* のドキュメントを参 照してください。 **同 等 のプロパティ名 :** `com.fortify.sca.PHPVersion`

ABAPコードの変 換

ABAPコードの変 換 は、その他 のオペレーティング言 語 コードの変 換 に似 ています。ただし、SAPデータベースからコードを抽 出してスキャン用 のコードを作 成する追 加 の手 順 が必 要 です。詳しくは、"移 送 依 頼 のインポート" 次 のページを参 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。

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Fortify Static Code Analyzerを呼 び出します。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`例 外 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`)を増 加 させます。

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 LIKEprogramname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。この方 法 では、レコードのREPORTSとともにクラスとメソッドも読 み出 されます。

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。このファイルセットに対 してFortify Static Code Analyzerが変 換 とスキャンを行 い、Micro Focus Fortify Audit Workbenchで開 くことができるFPRファイルを生 成 します。

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。

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。Fortifyの移 送 依 頼 は*<sca_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。

- K900XXX.S9S(「XXX」はリリース番 号)
- R900XXX.S9S(「XXX」はリリース番 号)

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。

S95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Fortify Static Code Analyzer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

Fortify ABAP Extractorの移 送 依 頼 は、SAPリリース7.02、SPレベル0006が実 行 されているシステムでサポートされています。別 のSAPバージョンを実 行 していて、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 は、移 送 依 頼 のインストールに失 敗 しています。

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。

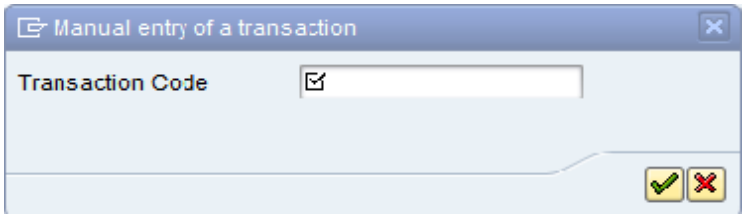
それでも問 題 が解 決 しない場 合 や、異 なるテーブル構 造 を持 つSAPバージョンがシステムで実 行 されている場 合 、Fortifyでは、Fortify Static Code AnalyzerがABAPコードをスキャンできるように、独 自 の技 術 を使 用 してABAPファイル構 造 をエクスポートすることをお勧 めします。

Fortify Static Code Analyzerのお気 に入 りリスト への追 加

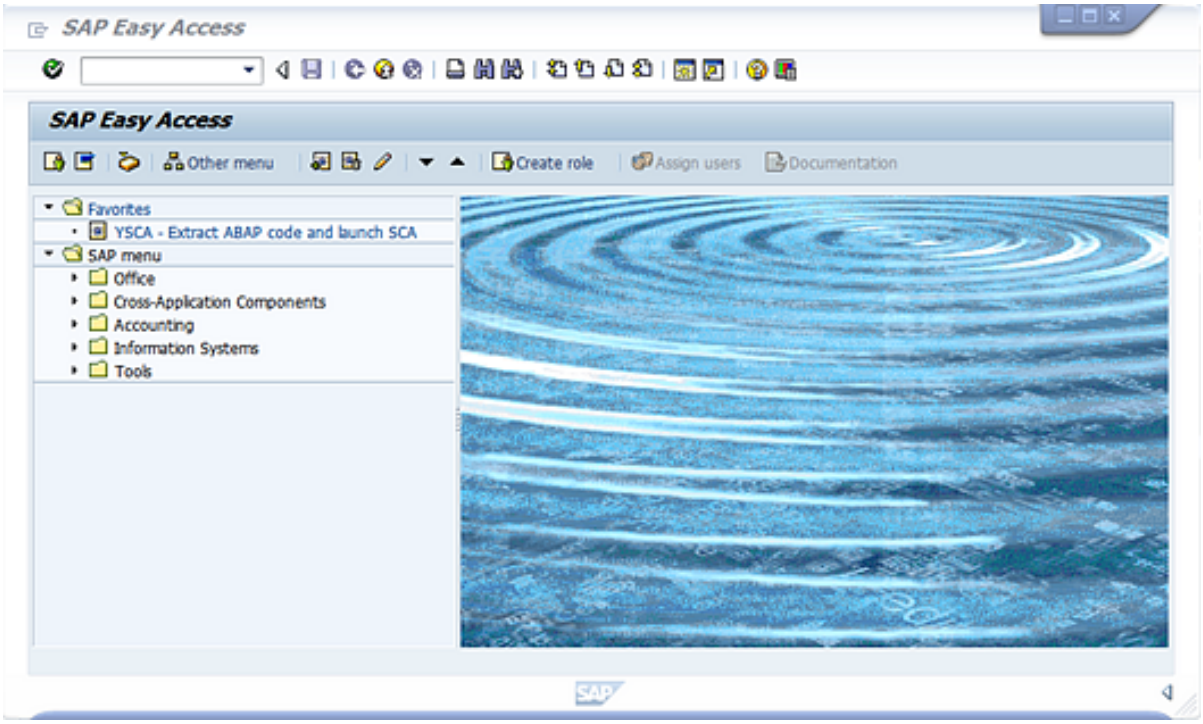
Fortify Static Code Analyzerのお気 に入 りリスト への追 加 はオプションですが、追 加 すると、Fortify Static Code Analyzerスキャンにすばやくアクセスして起 動 できます。次 の手 順 では、日 常 業 務 でユーザメニューを使 用 すると仮 定 しています。別 のメニューから作 業 を行 う場 合 は、使 用 するメニューにお気 に入 りリンクを追 加 します。Fortify Static Code Analyzerのエントリを作 成 する前 に、SAPサーバが実 行 中 であり、WebベースのクライアントのSAP Easy Accessエリアにアクセスしていることを確 認 してください。

Fortify Static Code Analyzer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. [Transaction Code]ボックスに「YSCA」と入 力 します。
4. 緑 色 のチェックマークボタンをクリックします。
   [Favorites]リストに [Extract ABAP code and launch SCA]項 目 が表 示 されます。

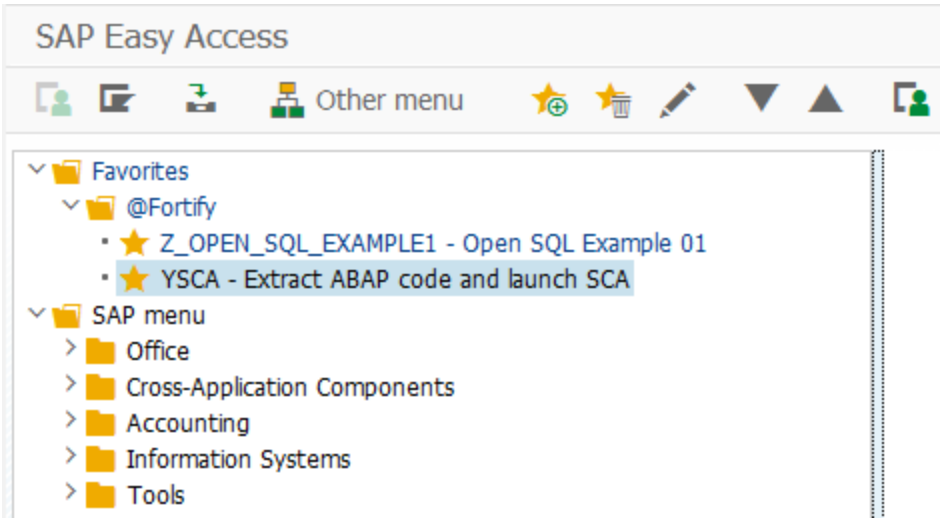


5. 『**Extract ABAP code and launch SCA**』リンクをクリックしてFortify ABAP Extractorを起動します。

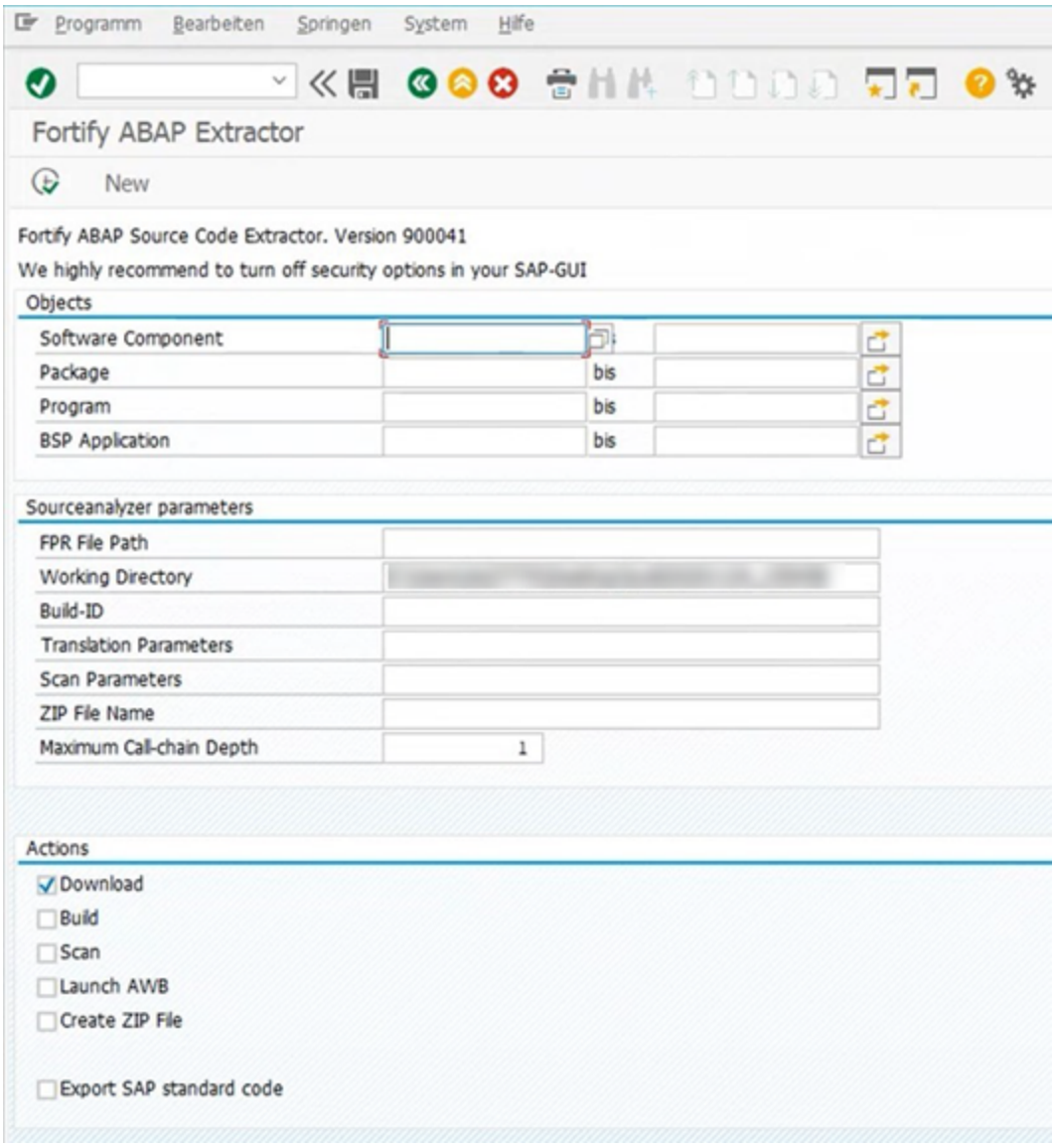
Fortify ABAP Extractorの実 行

Fortify ABAP Extractorを実 行 するには、次 の手 順 を実 行します。

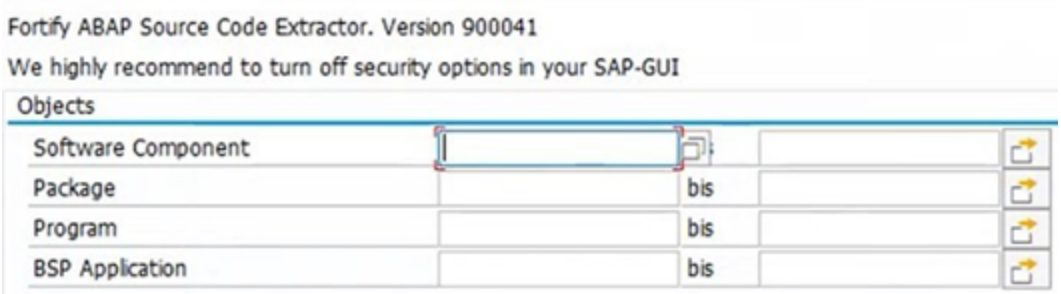
1. 『**Favorites**』リンクまたはトランザクションコード からプログラムを起 動 するか、手 動 で Extractorオブジェクトを起 動します。



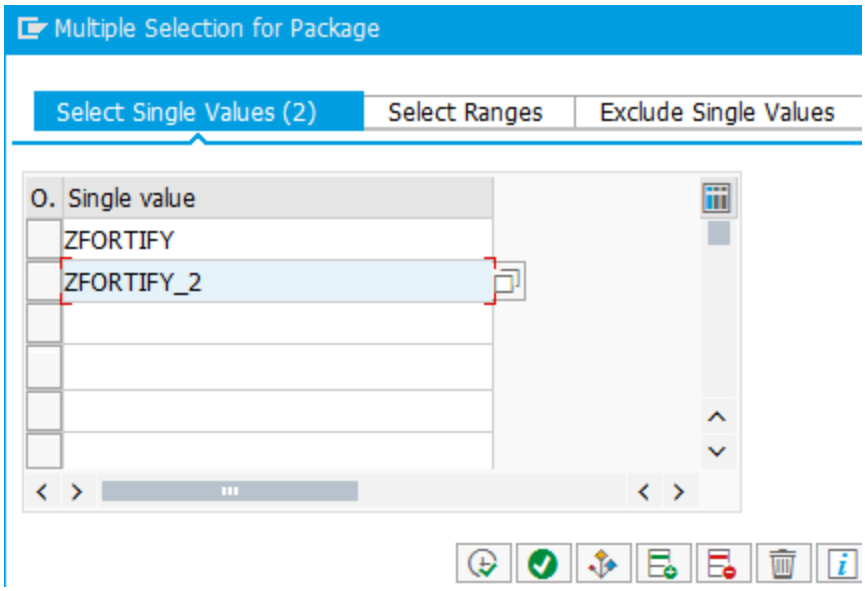
これにより、Fortify ABAP Extractorが開きます。



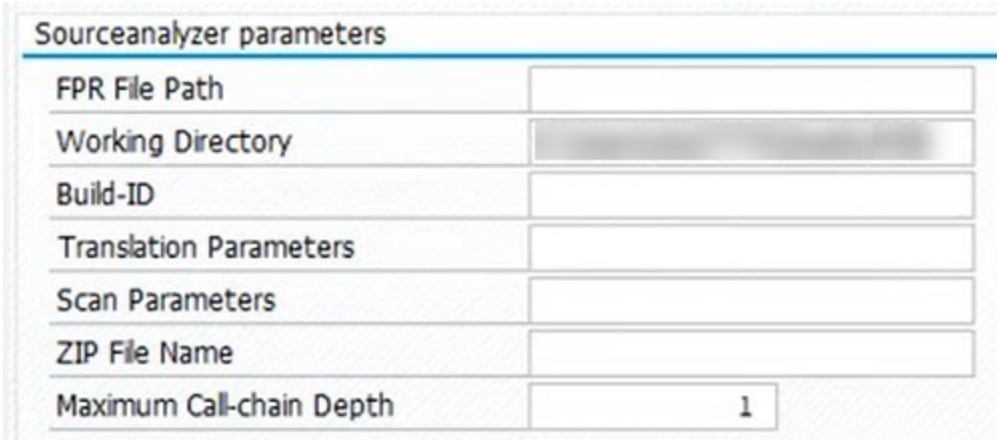
2. ダウンロードするコードを選 択します。
   スキャンするソフトウェアコンポーネント、パッケージ、プログラム、またはBSPアプリケーション の範 囲を開 始 名 と終 了 名 で指 定します。



注: 複 数 のオブジェクトまたは範 囲 を指 定 できます。



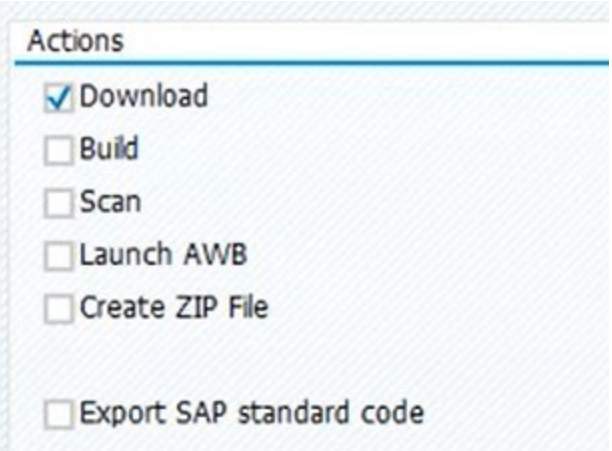
3. 次 の表 に示 すFortify Static Code Analyzer固 有 の情 報 を指 定 します。



フィールド	説 明
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 にスキャンする場 合 は、FPRファイルパスを指 定 する必 要 があります。
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。Fortify Static Code Analyzer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

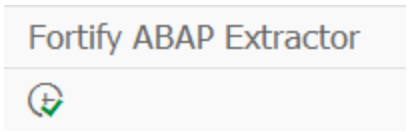
フィールド	説 明
	を実 行 している同 じコンピュータにダウンロードしたコードを自 動 的 に変 換 する場 合 は、ビルドIDを指 定 する必 要 があります。
Translation Parameters	(オプション)追 加 のFortify Static Code Analyzerコマンドライン変 換 オプションを入 力 します。抽 出 プロセスを実 行 している同 じコンピュータにダウンロードしたコードを自 動 的 に変 換 する場 合 、または変 換 オプションをカスタマイズする場 合 は、変 換 パラメータを指 定 する必 要 があります。
Scan Parameters	(オプション)Fortify Static Code Analyzerコマンドラインスキャンオプションを入 力 します。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 にスキャンする場 合 、またはスキャンオプションをカスタマイズする場 合 は、スキャンパラメータを指 定 する必 要 があります。
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。Fortifyでは、Micro Focus Fortify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
Download	SAPデータベースから抽 出 したソースコードをFortify Static Code Analyzerでダウンロード するには、このチェックボックスをオンにしま す。
Build	Fortify Static Code Analyzer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、この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Fortify Static Code Analyzer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Fortify Static Code Analyzerコンピュータに移 動 する方 が簡 単 です。
Scan	指 定 したビルドIDのスキャンをFortify Static Code Analyzerで実 行 するには、この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Fortify Static Code Analyzer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Fortify Static Code Analyzerコンピュータに移 動 する方 が簡 単 です。
Launch AWB	Micro Focus Fortify Audit Workbenchを起 動 して指 定 したFPR ファイルを起 動 するには、このチェックボックスをオンにします。
Create ZIP File	出 力 を圧 縮 するには、この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 コードをエクスポート するには、この チェックボックスをオンにします。

5. [**Execute**]をクリックします。



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

FlexおよびActionScriptの変 換

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 のシステム)「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。

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**
`-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

FlexおよびActionScriptオプション	説 明
	す。 **注:** 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-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`com.fortify.manager.util`内 の`Foo`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`

注: `-flex-sdk-root`および`–flex-source-roots`は主 にMXML変 換 用 であり、純 粋 なActionScriptをスキャンする場 合 はオプションです。`–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。

Fortify Static Code Analyzer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、Micro Focus Fortifyカスタマサポートまでお知 らせください。

ActionScriptのコマンドライン例

次 の例 は、ActionScriptの変 換 に関 する一 般 的 なシナリオのコマンドライン構 文 を示 しています。

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パッケージ内 に位 置 するFlexAppという1つのActionScriptクラスとしてMXMLアプリケーションを変 換します。

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、Fortify Static Code Analyzerファイル指 定 子 を使 用 してsrcフォルダ内 の.asファイルと.mxmlファイルを含 めます。この例 では、-flex-sdk-rootにある.SWC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。Fortify Static Code Analyzer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ルートライブラリと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、データの入 力 には時 間 がかかり、データは一 般 的 に一 定 なものだからです。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srcフォルダ内 にあるすべての.mxmlおよび.as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、パッケージcom.foo.util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"
```

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、Fortify Static Code Analyzer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。Fortify Static Code Analyzer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。

ColdFusionコードの変 換

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、*<sca_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。

ColdFusionコマンドライン構 文

ColdFusionソースコードを変 換 するには、次 のコマンドを入 力 します。

```
sourceanalyzer -b <build_id> -source-base-dir <dir> <files> | <file_
specifiers>
```

ここで:

- *<build_id>*はプロジェクトのビルドIDを指 定 します
- *<dir>*は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、"ファイルとディレクトリの指 定 " ページ138を参 照 してください。

注 : Fortify Static Code Analyzerでは、各 CFMLソースファイルへの相 対 パスを計 算 するために
`-source-base-dir` ディレクトリを開 始 点 として使 用 します。Fortify Static Code Analyzer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、Fortify Static Code Analyzerで生 成 されるインスタンスIDは同 じままです。

ColdFusionコマンドラインオプション

次 の表 は、ColdFusionのオプションについて説 明しています。

ColdFusionのオプション	説 明
`-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`

SQLの変 換

WindowsプラットフォームのFortify Static Code Analyzer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Fortify Static Code Analyzerを設 定 する必 要 があります。

WindowsプラットフォームでのSQL変 換 にSQLタイプを指 定 するには、次 のいずれかの変 換 コマンドを入 力します。

```
sourceanalyzer -b <build_id> -sql-language TSQL <files>
```

または

```
sourceanalyzer -b <build_id> -sql-language PL/SQL <files>
```

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`または`TSQL`に設 定 します。

PL/SQLのコマンドライン例

次 の例 は、2つのPL/SQLファイルを変 換 する構 文 を示しています。

```
sourceanalyzer -b MyProject x.pks y.pks
```

次 の例 は、`sources`ディレクトリ内 のすべてのPL/SQLファイルを変 換 する方 法 を示しています。

```
sourceanalyzer -b MyProject "sources/**/*.pks"
```

T-SQLのコマンドライン例

次 の例 は、2つのT-SQLファイルを変 換 するコマンドを示 しています。

```
sourceanalyzer -b MyProject x.sql y.sql
```

次 の例 は、sourcesディレクトリ内 のすべてのT-SQLファイルを変 換 する方 法 を示 しています。

```
sourceanalyzer -b MyProject "sources\**\*.sql"
```

注: この例 では、fortify-sca.properties内 のcom.fortify.sca.fileextensions.sqlプロパティがTSQLに設 定 されている場 合 を想 定 しています。

Scalaコードの変 換

Scalaコード の変 換 に必 要 なものは次 のとおりです。

- Scala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。
- Lightbendライセンスファイル
  このライセンスファイルは、Fortify Static Code Analyzerのインストールに含 まれ、*<sca_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、https://developer.lightbend.com/guides/fortifyにあるLightbendのドキュメントを参 照 してください。

重 要 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。

Dockerfileの変 換

デフォルト で、Fortify Static Code AnalyzerではDockerfile*、dockerfile*、*.Dockerfile、および*.dockerfileファイルがDockerfileとして返 還 されます。

注: 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 されるファイル拡 張 子 を変 更 できます。"fortify-sca.properties" ページ196を参 照 してください。

Fortify Static Code Analyzer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 、Fortify Static Code Analyzer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 、Fortify Static Code Analyzerではファイルを解 析 できないことを示 すエラーをログファイルに書 き込 み、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
```

ASP/VBScript仮 想 ルートの変 換

Fortify Static Code Analyzer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 、Fortify Static Code Analyzer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Fortify Static Code Analyzerに提 供 するには、次 の例 に示 すように、Fortify Static Code Analyzer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>
```

注 : 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Fortify Static Code Analyzerに渡 します。

```
-Dcom.fortify.sca.ASPVirtualRoots.Include="C:\WebServer\CustomerOne\inc"

-Dcom.fortify.sca.ASPVirtualRoots.Library="C:\WebServer\CustomerTwo\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。

Fortify Static Code Analyzerで`#include`ディレクティブが出 現 する場 合 :

```
<!-- #include virtual="Include/Task1/foo.inc" -->
```

Fortify Static Code Analyzer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 、Fortify Static Code Analyzerで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 、Fortify Static Code Analyzerでは、`C:\WebServer\CustomerOne\inc\Task1\foo.inc`ファイルを探します。

または、*<sca_install_dir>*`\Core\config`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\\CustomerTwo\\Stuff

com.fortify.sca.ASPVirtualRoots.Include=C:\\WebServer\\CustomerOne\\inc
```

注 : 以 前 のバージョンのASPVirtualRootプロパティは引 き続 き有 効 です。Fortify Static Code Analyzerコマンドラインで次 のように使 用 できます。

```
-Dcom.fortify.sca.ASPVirtualRoots=C:\WebServer\CustomerTwo\Stuff;
C:\WebServer\CustomerOne\inc
```

これはFortify Static Code Analyzer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 、Fortify Static Code Analyzer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。

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 のコマンドを入 力 します。

```
sourceanalyzer -b mybuild "MyASP.asp"
```

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"
```

第16章: ビルドへの統合

サポートされているビルドツールに分析を統合できます。

このセクションでは、次のトピックについて説明します。

ビルド統合

プロジェクト全体を1回の操作で変換できます。元のビルド操作の前に`sourceanalyzer`コマンドをFortify Static Code Analyzerオプション付きで追加します。

完全なプロジェクト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> [<sca_options>] <build_tool> [<build_tool_
options>]
```

ここで、*<build_tool>*はビルドツール(make、gmake、msbuild、devenv、gbuildなど)の名前です。サポートされているビルドツールのリストについては、ドキュメント『*Micro Focus Fortifyソフトウェアシステム要件*』を参照してください。Fortify Static Code Analyzerでは、ビルドツールが実行され、すべてのコンパイラ操作を傍受して各入力に使用される特定のコマンドラインが収集されます。

注: Fortify Static Code Analyzerでは、ビルドツールで実行されるコンパイラコマンドのみが処理されます。ビルドを実行する前にプロジェクトをクリーンアップしない場合は、ビルドツールで再コンパイルされるファイルのみがFortify Static Code Analyzerで処理されます。

Xcodebuildとの統合については、「"iOSコード分析のコマンドライン構文" ページ83」を参照してください。MSBuildとの統合については、「"Visual StudioおよびMSBuildプロジェクトの変換" ページ65」を参照してください。

ビルドを正常に統合するには、ビルドツールで次が実行される必要があります。

- Fortify Static Code Analyzerでサポートされているコンパイラを実行する
- ハードコードされたパスではなく、オペレーティングシステムパスの検索でコンパイラを実行する(xcodebuildの統合には、この要件が必ずしも適用されません。)
- 後でコンパイラが実行されるサブプロセスを実行するのではなく、コンパイラを実行する

これらの要 件 を環 境 で満 たすことができない場 合 は、「"Fortify Static Code Analyzerが起 動 されるようにビルド スクリプトを変 更 " 下 」を参 照 してください。

makeの例

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 :

```
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
```

Fortify Static Code Analyzerが起 動 されるようにビルド スクリプトを変 更

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Fortify Static Code Analyzerオプションを追 加 できます。

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

各 操 作 に同 じビルド IDを使 用 する場 合 は、Fortify Static Code Analyzer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。

touchlessビルド 統 合

Fortify Static Code Analyzerには、Fortify Static Code Analyzerで直 接 サポートされていないビルド システムを使 用 してプロジェクトを変 換 できるtouchlessという汎 用 ビルド ツールが含 まれています。touchlessビルド 統 合 のコマンド ライン構 文 は次 のとおりです。

```
sourceanalyzer -b <build_id> touchless <build_command>
```

たとえば、`build.py`というPythonスクリプトを使用して依存関係を計算し、適切に順序付けられたCコンパイラ操作を実行する場合があります。ビルドを実行するには、次のコマンドを実行します。

```
python build.py
```

Fortify Static Code Analyzerでは、このようなビルド設計がネイティブサポートされていません。ただし、touchlessビルドツールを使用して、単一のコマンドでプロジェクト全体を変換およびビルドできます。

```
sourceanalyzer -b <build_id> touchless python build.py
```

この章の前半で説明したサポートされているビルドシステムにビルドを正常に統合するための同じ要件(「"ビルド統合" ページ118」を参照)は、サポートされていないビルドシステムとのtouchlessの統合にも適用されます。

Antの統合

Fortify Static Code Analyzerでは、Antビルドファイルを使用するプロジェクト用にJavaソースファイルを簡単に変換する方法が提供されています。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、Fortify Static Code Analyzer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。

注: 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`システムにインストールされ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> ant [<ant_options>]
```

Gradleの統合

`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、Fortify Static Code Analyzerでソースファイルがコンパイル時に変換されます。特にGradleの統合で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、ドキュメント『*Micro Focus Fortifyソフトウェアシステム要件*』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。

Fortify Static Code AnalyzerをGradleビルドに統合するには、sourceanalyzer実行可能ファイルがシステムのパス上にあることを確認します。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>
```

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>]
```

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コマンドを使用して変換します。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>
```

gradleまたはgradlewでプロジェクトが変換されたら、次の例に示すように分析フェーズを実行できます。

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

詳細オプションとデバッグオプションを含める

Fortify Static Code Analyzerの-verboseオプションを使用する場合は、-gradleオプションも含める必要があります。このオプションの使用は、GradleとGradle Wrapperの両方に適用されます。例:

```
sourceanalyzer -b MyProject -gradle -verbose gradle assemble
```

Gradleの統合の一環として、Fortify Static Code Analyzerで元のビルドファイルbuild.gradleが一時的に更新されます。-debugオプションを含める場合は、Fortify Static Code Analyzerに元のビルドファイルのコピーがbuild.gradle.origとして保存されます。-debugオプションを使用した分析が完了したら、build.gradle.origファイルの名前をbuild.gradleに戻し、-debugオプションを付けずにsourceanalyzerを再度実行します。

Mavenの統 合

Fortify Static Code Analyzer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。

- Fortify Static Code Analyzerで消 去 、変 換 、スキャンする
- Fortify Static Code Analyzerで変 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 Fortify Static Code Analyzerからエクスポート する
- 変 換 されたコードをMicro Focus Fortify ScanCentral SASTに送 信 する
- 結 果 をMicro Focus Fortify Software Security Center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。

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、*<sca_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でサポート されているバージョンについては、ドキュメント『*Micro Focus Fortifyソフト ウェアシステム要 件* 』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。

Fortify Mavenプラグイン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:/<sca_install_dir>/bin
   ```

   または

   ```
   set PATH=%PATH%;<sca_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、Fortify Static Code Analyzer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。

   注: 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*<root_dir>*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。次のエラー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。

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Fortify分析を実行します。

- 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Fortify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:<ver>: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プラグインが正しくインストールされた後にドキュメントが見つか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

- Fortify Static Code Analyzer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Fortify Static Code Analyzerオプションを追加します。Fortify Static Code Analyzer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。
  a. 以前のビルドを消去します。

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

  b. コードを変換します。

     ```
     sourceanalyzer -b MyProject [<sca_options>] [<mvn_command_with_
     options>]
     ```

     例:

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

     次の追加の例には、選択したファイルを分析から除外するFortify Static Code Analyzerオプションが含まれています。除外するファイルを指定するには、次の例に示すように、変換ステップに-excludeオプションを追加します。

     ```
     sourceanalyzer -b MyProject -exclude "fileA;fileB;fileC;" mvn package
     ```

注 : 複 数 のファイル名 はセミコロン(Windows)またはコロン(Windows以 外)で区 切り ます。

使 用 可 能 なFortify Static Code Analyzerオプションについては、「"コマンド ラインインタ フェース" ページ 126」を参 照してください。

c. 次 の例 に示 すように、スキャンを実 行して分 析 を完 了します。

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

第 17章 : コマンドラインインタフェース

この章では、一般的なFortify Static Code Analyzer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、その言語の章で説明します。

このセクションでは、次のトピックについて説明します。

変換オプション

次の表では、変換オプションについて説明します。

変換オプション	説明
`-b <build_id>`	ビルドIDを指定します。Fortify Static Code Analyzer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`	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`abap`、`actionscript`、`apex`、`cfml`、`cobol`、`configuration`、`cpp`、`dotnet`、`golang`、`java`、`javascript`、`jsp`、`kotlin`、`objc`、`php`、`plsql`、`python`、`ruby`、`scala`、`sql`、`swift`、`tsql`、`typescript`、`vb`です。 **同等のプロパティ名 :** `com.fortify.sca.DISabledLanguages`

変換オプション	説明
`-enable-language`	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`abap`、`actionscript`、`apex`、`cfml`、`cobol`、`configuration`、`cpp`、`dotnet`、`golang`、`java`、`javascript`、`jsp`、`kotlin`、`objc`、`php`、`plsql`、`python`、`ruby`、`scala`、`sql`、`swift`、`tsql`、`typescript`、`vb`です。 **同等のプロパティ名 :** `com.fortify.sca.EnabledLanguages`

`-exclude <file_specifiers>`	変換するファイルのリストからファイルを削除します。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。例 :
	`sourceanalyzer -cp "**/*.jar" "**/*" -exclude "**/Test/*.java"`
	この例では、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。ファイル指定子を使用する方法の詳細については、「"ファイルとディレクトリの指定" ページ138」を参照してください。
	注 : ほとんどのコンパイラまたはビルドツールと変換を統合すると、このオプションで除外するように指定されている場合でも、コンパイラまたはビルドツールで処理されるソースファイルがすべてFortify Static Code Analyzerで変換されます。ただし、Fortify Static Code AnalyzerのxcodebuildおよびMSBuildでは、`-exclude`オプションがサポートされています。
	同等のプロパティ名 :
	`com.fortify.sca.exclude`

`-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。Fortify Static Code Analyzer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。複数のエンコードを含むプロジェクトを処理するには、Fortify Static Code Analyzerで最初にソースコードファイルを読み込む際に、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。Fortify Static Code Analyzer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。

変換オプション	説明
	通常、エンコーディングタイプを指定しないと、Fortify Static Code Analyzerではエンコーディングパラメータなしで `java.io.InputStreamReader`コンストラクタから`file.encoding`が使用されます。一部のケース(ActionScriptパーサの場合など)では、Fortify Static Code Analyzerのデフォルトが`UTF-8`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、Fortify Static Code Analyzer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。
`-noextension-type <file_type>`	ファイル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_TRIGGER、ARCHIVE、ASPNET、ASP、ASPX、BITCODE、BSP、BYTECODE、CFML、COBOL、CSHARP、DOCKERFILE、GENERIC、GO、HOCON、HTML、INI、JAVA、JAVA_PROPERTIES、JAVASCRIPT、JSP、JSPX、KOTLIN、MSIL、MXML、OBJECT、PHP、PLSQL、PYTHON、RUBY、RUBY_ERB、SCALA、SWIFT、SWC、SWF、TLD、SQL、TSQL、TYPESCRIPT、VB、VB6、VBSCRIPT、VISUAL_FORCE、XMLです。

分析オプション

次の表では、分析オプションについて説明します。

分析オプション	説明
`-scan`	Fortify Static Code Analyzerで指定したビルドIDの分析が実行されます。 **注 :** このオプションは、同じsourceanalyzerコマンド内で`-scan-module`オプションと同時に使用しないでください。
`-scan-module`	Fortify Static Code Analyzerで指定したビルドIDの分析が個別のモジュールとして実行されます。 **注 :** このオプションは、同じsourceanalyzerコマンド内で`-scan`オ

分析オプション	説明
	プションと同時に使用しないでください。 **同等のプロパティ名 :** com.fortify.sca.ScanScaModule
-include-modules	プロジェクトスキャンに含めるビルドIDのカンマ区切りリストまたはコロン区切りリストで、以前に個別のモジュールとしてスキャンされたライブラリを指定します。 **同等のプロパティ名 :** com.fortify.sca.IncludeScaModules
-analyzers	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
-b *<build_id>*	ビルドIDを指定します。 **同等のプロパティ名 :** com.fortify.sca.BuildID
-p *<level>* | -scan-precision *<level>*	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。精度レベルが低いスキャンほど高速に実行されます。有効な値は、1、2、3、および4です。詳細については、「"短縮ダイヤルを使用したスキャン速度の設定" ページ165」を参照してください。 **同等のプロパティ名 :** com.fortify.sca.PrecisionLevel
-quick	fortify-sca-quickscan.propertiesファイルを使用して、プロジェクトをクイックスキャンモードでスキャンします。詳細さは低下します。デフォルトでは、Buffer AnalyzerとControl Flow Analyzerが無効になっています。さらに、Quick Viewフィルタセットも適用されます。 **同等のプロパティ名 :** com.fortify.sca.QuickScanMode
-bin *<binary>* | -binary-name	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まれ

分析オプション	説明
<binary>	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`
`-disable-default-rule-type` *<type>*	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。
`-filter` *<file>*	結果フィルタファイルを指定します。このオプションの詳細については、「"分析のフィルタリング" ページ184」を参照してください。 **同等のプロパティ名 :** `com.fortify.sca.FilterFile`
`-no-default-issue-rules`	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。関数の動作を示すルールもロードします。 **注 :** 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`	デフォルトのRulepackからルールがロードされないように指定します。Fortify Static Code Analyzer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`
`-no-default-source-rules`	デフォルトのRulepackでソースルールを無効にします。

分析オプション	説明
	注 : 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`	デフォルトのRulepackでシンクルールを無効にします。 **注 :** 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`
`-project-template`	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Micro Focus Fortify Software Security Centerサーバ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`
`-rules <file>` | `<dir>`	カスタムのRulepack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、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まれます。 **同等のプロパティ名 :** `com.fortify.sca.RulesFile`

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。

出力オプション	説明
`-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、Fortify Static Code Analyzer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`
`-format <format>`	出力形式を制御します。有効なオプションは、`fpr`、`fvdl`、

出力オプション	説明
	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拡張子に基づいて出力形式が選択されます。 FVDLは、Fortify Static Code Analyzer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。Micro Focus Fortify Audit Workbenchは、自動的に.fprファイル拡張子に関連付けられます。 **注:** 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、『*Micro Focus 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「"fortify-sca.properties" ページ196」を参照してください。 **同等のプロパティ名:** com.fortify.sca.Renderer
-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)Fortify 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、Fortify Static Code Analyzer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、Fortifyでは、-append生成された結果が証明されません。

出力オプション	説明
	このオプションを指定しない場合は、Fortify Static Code Analyzer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
-build-label *<label>*	スキャンされるプロジェクトのラベルを指定します。Fortify Static Code Analyzerでは、このラベルが使用されませんが、分析結果には含まれます。 **同等のプロパティ名 :** com.fortify.sca.BuildLabel
-build-project *<project>*	スキャンされるプロジェクトの名前を指定します。Fortify Static Code Analyzerでは、この名前が使用されませんが、分析結果には含まれます。 **同等のプロパティ名 :** com.fortify.sca.BuildProject
-build-version *<version>*	スキャンされるプロジェクトのバージョンを指定します。Fortify Static Code Analyzerでは、このバージョンが使用されませんが、分析結果には含まれます。 **同等のプロパティ名 :** com.fortify.sca.BuildVersion
-disable-source-bundling	ソースファイルをFPRファイルから除外します。 **同等のプロパティ名 :** com.fortify.sca.FPRDisableSourceBundling
-fvdl-no-descriptions	Fortify 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、Fortify 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Fortify Static Code Analyzerの実行に関するその他の情報が含まれています。

出力オプション	説明
	同等のプロパティ名 : `com.fortify.sca.FVDLDisableEngineData`
`-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`

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
@<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。 **注 :** デフォルトでは、このファイルでJVMシステムエンコーディングが使用されます。`fortify-sca.properties`ファイルで指定された`com.fortify.sca.CmdlineOptionsFileEncoding`プロパティを使用して、エンコーディングを変更できます。このプロパティの詳細については、「"fortify-sca.properties" ページ 196」を参照してください。
`-h`| `-?`| `-help`	コマンドラインオプションの概要を出力します。
`-debug`	Fortify Supportログファイルにデバッグ情報を含めます。この情報は、Micro Focus Fortifyカスタマサポートの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。

その他 のオプション	説 明
	同 等 のプロパティ名 : `com.fortify.sca.DebugVerbose`
`-verbose`	詳 細 ステータスメッセージをコンソールとFortify 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>`	Fortify Static Code Analyzerで作 成されるログファイルを指 定します。 **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Fortify Static Code Analyzerに指 示します。このオプションを指 定しない場 合 は、Fortify Static Code Analyzer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`
`-version` | `-v`	Fortify Static Code AnalyzerのバージョンとFortify Static Code Analyzer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、Fortify Static Code Analyzer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
`-Xmx<size>M | G`	Fortify Static Code Analyzerで使 用されるメモリ使 用 量 の最 大 値 を指 定します。 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32 GBを超 えるサイズが必 要 な場 合 は、64 GB以 上 が必 要 になる可 能 性 があります。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。 **注 :** このオプションを指 定 すると、`-autoheap`オプションで取 得 したデフォルトのメモリ割り当 てが上 書 きされます。

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
`-clean`	すべてのFortify Static Code Analyzer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。

ディレクティブ	説明
	注 : 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。
`-show-build-warnings`	`-b <build_id>`とともに使用すると、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。 **注 :** 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のファイルのリストが表示されます。`-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`	変換されるコードの行数が表示されます。

LIMライセンスディレクティブ

Fortify Static Code Analyzer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。

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。

ディレクティブ	説明

`-store-license-pool-credentials "<lim_url>|<lim_pool_name>|<lim_pool_pwd>|<proxy_`	Fortify Static Code Analyzer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。Fortify Static Code Analyzer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。Fortify Static Code Analyzer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>/LIM.REST.API|TeamA|mypassword"`
	関連付けられたプロパティ名 :
	`com.fortify.sca.lim.Url`

ディレクティブ	説明
url> | *<proxy_user>* | *<proxy_pwd>*"	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。
-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Fortify Static Code Analyzerを実行できます。 **注 :** 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。

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Fortify Static Code Analyzerに渡すことができる式です。Fortify Static Code Analyzer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。

<files> | <file_dir_specifiers>

注 : ファイル指定子は、C、C++、またはObjective-C++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/ディレクトリ指定子	説明
<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。
/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。
**/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

注: 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。

第18章: コマンドラインユーティリティ

このセクションでは、次のトピックについて説明します。

Fortify Static Code Analyzerユーティリティ

Fortify Static Code Analyzerコマンドラインユーティリティを使用すると、Fortify Security ContentおよびFPRファイルの管理、レポートの実行、インストール後の設定、スキャンの監視などを実行できます。これらのユーティリティは、*<sca_install_dir>*/binにあります。Windows用のユーティリティは、.batまたは.cmdファイルとして提供されます。次の表では、ユーティリティについて説明します。

注: デフォルトでは、ほとんどのユーティリティ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の場合: C:\Users*<username>*\AppData\Local\Fortify*<utility_name>*-*<version>*\log
- LinuxおよびmacOSの場合: *<userhome>*/.fortify/*<utility_name>*-*<version>*\log

ユーティリティ	説明	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	"セキュリティコンテンツの更新について" 次のページ
FPRUtility	このユーティリティを使用すると、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 • 移行済みプロジェクトのマッピングを表示する • FPRに関連するエラーを表示する	"コマンドラインからのFPRファイルの操作" ページ144

ユーティリティ	説明	詳細情報
	• FPR内の問題の数を表示する • 問題のフィルタされたリストをさまざまな形式で表示する • 分析された関数の表を表示する • FPRを変更する • 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	
BIRTReportGenerator ReportGenerator	FPRファイルからBIRTレポートやレガシレポートを生成します	"コマンドラインからのレポートの生成" ページ151
scapostinstall	Fortify Static Code Analyzerのインストール後にこのユーティリティを使用すると、Fortify Static Code Analyzer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Fortify Software Security Center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" ページ41
SCAState	スキャンフェーズ中にJVMに関する状態分析情報を提供します。	"Fortify Static Code Analyzerスキャンステータスの確認" ページ157

セキュリティコンテンツの更新について

`fortifyupdate`ユーティリティを使用して、最新のFortify Secure Coding Rulepacksメタデータを Fortifyからダウンロードできます。

`fortifyupdate`ユーティリティは、Fortifyインストール中の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ルールパック更新サーバに問い合わせします。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Fortify Static Code Analyzer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。

セキュリティコンテンツの更新

fortifyupdateユーティリティを使用して、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。このユーティリティは*<sca_install_dir>*/binディレクトリにあります。

Fortifyルールパック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Fortify Static Code Analyzer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 [<options>]
```

fortifyupdateコマンドラインオプション

次の表に、fortifyupdateオプションを示します。

fortifyUpdateオプション	説明
-import *<file>*.zip	アーカイブされた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。ルールパックは*<sca_install_dir>*/Core/config/rulesディレクトリに抽出されます。
-coreDir *<dir>*	更新が保存されるコアディレクトリを指定します。これが指定されていない場合、更新は*<sca_install_dir>*で実行されます。 **重要** *<sca_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。
-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。デフォルトは、fortify.properties環境設定ファイルのロケールプロパティに設定された値です。 fortify.properties環境設定ファイルの詳細については、『*Micro Focus Fortify Static Code Analyzerツールプロパティリファレンスガイド*』を参照してください。

fortifyUpdateオプション	説明
`-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。
`-proxyUsername <username>`	プロキシサーバで認証が必要な場合に、ユーザ名を指定します。
`-proxyPassword <password>`	プロキシサーバで認証が必要な場合に、パスワードを指定します。
`-showInstalledRules`	カスタムルールまたはメタデータを含む、現在インストールされているルールパックを表示します。
`-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、『*Micro Focus Fortify Static Code Analyzerツールプロパティリファレンスガイド*』を参照してください。
`-acceptKey`	公開鍵を受諾します。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。非標準の場所(`-url`オプションで指定)から更新する場合は、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証明書を使用します。

コマンドラインからのFPRファイルの操 作

Fortify Static Code Analyzerインストールの`bin`ディレクトリにあるFPRUtilityを使 用して、次 のタスクを実 行します。

FPRファイルのマージ

FPRUtilityの`-merge`オプションを使 用 すると、プライマリプロジェクトの値 を使 用して2つのFPRファイルの分 析 情 報 を1つのFPRファイルに結 合して、競 合 を解 決します。

FPRファイルをマージするには、次 のコマンドを入 力します。

```
FPRUtility -merge -project <primary>.fpr -source <secondary>.fpr \ -f
<output>.fpr
```

FPRファイルをマージし、インスタンスIDのマイグレートオプションを設 定 するには、次 のコマンドを入 力します。

```
FPRUtility -merge -project <primary>.fpr -source <secondary>.fpr \ -f
<output>.fpr -iidmigratorOptions "<iidmigrator_options>"
```

FPRUtilityデータマージオプション

次 の表 は、データのマージに適 用されるFPRUtilityオプションを一 覧 表 示しています。

FPRUtilityのオプション	説 明
`-merge`	指 定したプロジェクトとソースFPRファイルをマージします。
`-project <primary>.fpr`	マージするプライマリFPRファイルを指 定します。競 合 は、このファイルの値 を使 用して解 決されます。
`-source <secondary>.fpr`	マージするセカンダリFPRファイルを指 定します。競 合 が存 在 する場 合 、プライマリプロジェクトによって値 が上 書 きされます。
`-f <output>.fpr`	マージされた出 力 ファイルの名 前 を指 定します。このファイ

FPRUtilityのオプション	説明
	ルは、マージされたファイルの結果です。 **注 :** このオプションを指定すると、元のFPRファイルはどちらも変更されません。このオプションを使用しない場合、マージされた結果でプライマリFPRが上書きされます。
`-forceMigration`	2つのプロジェクトでFortify Static Code Analyzerおよびルールパックバージョンが同じ場合でも、強制的にマイグレーションを実行します。
`-ignoreAnalysisDates`	マージでプライマリおよびセカンダリFPRファイルの分析日を無視するように指定します。そうしないと、セカンダリFPRは常にプライマリFPRで更新されます。
`-useSourceIssueTemplate`	セカンダリFPRで問題テンプレートのフィルタセットとフォルダを使用します。
`-useMigrationFile <mapping_file>`	インスタンスIDマッピングファイルを指定します。これにより、マイグレーション結果を使用せずに、マッピングを手動で変更できます。独自のインスタンスIDマッピングファイルを指定します。
`-iidmigratorOptions <iidmigrator_options>`	インスタンスIDマイグレートオプションを指定します。含まれるオプションはスペースで区切り、引用符で囲む必要があります。有効なオプションの例は次のとおりです。 • `-i`は、マージされたファイルの大文字と小文字を区別してファイル名を比較します。 • `-u <scheme_file>`は、インスタンスIDマイグレーション用に照合スキームを`<scheme_file>`から読み込むようiidmigratorに指示します。 **注 :** Cygwinコマンドプロンプトで操作する場合は、`<-iidmigrator_options>`を一重引用符で囲みます(`'-u <scheme_file>'`)。 Windowsの例 : `FPRUtility -merge -project <primary>.fpr -source <secondary>.fpr -f <output>.fpr -iidmigratorOptions "-u scheme_file"`

FPRUtilityのオプション	説明
-debug	FPRUtilityの問題のトラブルシューティングに役立つデバッグ情報を表示します。

FPRUtilityデータマージ終了コード

-mergeコマンドが完了すると、次の表に示す終了コードのいずれかがFPRUtilityによって渡されます。

終了コード	説明
0	マージは正常に完了しました。
5	マージに失敗しました。

FPRファイルからの分析結果情報の表示

FPRUtilityの-informationオプションを使用すると、分析結果に関する情報が表示されます。次の目的で情報を取得できます。

- 署名を検証する
- FPRに関連するエラーを調べる
- 各アナライザ、脆弱性カテゴリ、またはカスタムグループ化に関する問題の数を取得する
- 問題のリスト(一部の基本情報を含む)を取得する。これらのリストはフィルタできます。

プロジェクト署名情報を表示するには、次のコマンドを入力します。

```
FPRUtility -information -signature -project <project>.fpr -f <output>.txt
```

FPRの完全な分析エラーレポートを表示するには、次のコマンドを入力します。

```
FPRUtility -information -errors -project <project>.fpr -f <output>.txt
```

脆弱性カテゴリまたはアナライザごとの問題の数を表示するには、次のコマンドを入力します。

```
FPRUtility -information -categoryIssueCounts -project <project>.fpr
FPRUtility -information -analyzerIssueCounts -project <project>.fpr
```

検索に基づいてカスタムグループ化に関する問題の数を表示するには、次のコマンドを入力します。

```
FPRUtility -information -search -query <search_expression> \ [-
categoryIssueCounts] [-analyzerIssueCounts] \ [-includeSuppressed] [-
```

```
includeRemoved] \ -project <project>.fpr -f <output>.txt
```

注 : デフォルトでは、抑止された問題や削除された問題は結果に含まれません。抑止または削除された問題を含めるには、`-includeSuppressed`または`-includeRemoved`オプションを使用します。

問題に関する情報をCSV形式で表示するには、次のコマンドを入力します。

```
FPRUtility -information -listIssues \ -search [-queryAll | -query <search_
expression>] \ [-categoryIssueCounts] [-analyzerIssueCouts] \ [-
includeSuppressed] [-includeRemoved] \ -project <project>.fpr -f
<output>.csv -outputFormat CSV
```

クイックビューフィルタセットを使用して最新のスキャンのすべての問題(未抑止の問題と削除された問題を除く)に関する情報を表示するには、次のコマンドを入力します。

```
FPRUtility -information -listIssues \ -search -
queryAllExistingUnsuppressed \ -filterSet "Quick View" \ [-
categoryIssueCounts] [-analyzerIssueCouts] \ -project <project>.fpr -f
<output>.txt
```

FPRUtility情報オプション

次の表は、プロジェクト情報に適用されるFPRUtilityオプションを一覧表示しています。

FPRUtilityのオプション	説明
`-information`	プロジェクトの情報を表示します。
次のいずれかのオプションを指定して、表示する情報を指定します。	
`-signature`	署名を表示します。
`-mappings`	マイグレーションマッピングレポートを表示します。
`-errors`	FPRの完全なエラーレポートを表示します。
`-versions`	静的スキャンで使用されるFortify Static Code Analyzerおよびルールパックのバージョンを表示します。
`-functionsMeta`	静的アナライザで検出したすべての関数をCSV形式で表示します。表示される関数をフィルタするには、`-excludeCoveredByRules`および`-excludeFunctionsWithSource`を含めます。
`-categoryIssueCounts`	各脆弱性カテゴリの問題の数を表示します。
`-`	各アナライザの問題の数を表示します。

FPRUtilityのオプション	説明
analyzerIssueCounts	
-search *<query_option>*	• -search -query *<search_expression>*を使用すると、指定した検索式の結果における問題の数が表示されます。脆弱性カテゴリまたはアナライザごとの問題の数を表示するには、-categoryIssueCountsおよび-analyzerIssueCountsオプションをsearchオプションに追加します。抑止または削除された問題を含めるには、-includeSuppressedおよび-includeRemovedオプションを使用します。 • 抑止された問題と削除された問題を含む、FPRのすべての問題を検索するには、-search -queryAllを使用します。 • 抑止された問題と削除された問題を除く、FPRのすべての問題を検索するには、-search -queryAllExistingUnsuppressedを使用します。
-project *<project>*.fpr	結果情報を抽出するFPRを指定します。
-listIssues	各問題の場所を次のいずれかの形式で表示します。 *<sink_filename>*:*<line_num>*または *<sink_filename>*:*<line_num>* (*<category>* | *<analyzer>*) -listIssuesオプションは、-searchとも、両方のissueCountsグループ化オプションとも一緒に使用することもできます。-categoryIssueCountsでグループ化する場合、出力には(*<analyzer>*)が含まれます。-analyzerIssueCountsでグループ化する場合、出力には(*<category>*)が含まれます。 -outputFormat CSVオプションを指定した場合、各問題は次の形式で1行に表示されます。 "*<instanceid>*", "*<category>*", "*<sink_filename>*:*<line_num>*", "*<analyzer>*"
-filterSet *<filterset_name>*	フィルタセットで指定されたフィルタを通過する問題とカウントのみを表示します。このオプションを指定しない場合、フィルタセットは無視されます。 **重要** このオプションと一緒に-searchを使用する必要があります。
-f *<output>*	出力ファイルを指定します。デフォルトはSystem.outです。
-outputformat *<format>*	出力形式を指定します。有効な値は、TEXTおよびCSVです。デフォルト値はTEXTです。
-debug	FPRUtilityの問題のトラブルシューティングに役立つデバッグ情報を表示します。

FPRUtility署名終了コード

`-information -signature`コマンドが完了すると、次の表に示す終了コードのいずれかがFPRUtilityによって渡されます。

終了コード	説明
0	プロジェクトは署名され、すべての署名が有効です。
1	プロジェクトは署名され、署名の一部(すべてではない)は有効性テストに合格しました。
2	プロジェクトは署名されますが、どの署名も有効ではありません。
3	プロジェクトには検証する署名がありませんでした。

FPRファイルからのソースアーカイブの抽出

FPRUtilityの`-sourceArchive`オプションを使用すると、指定したFPRファイルからソースアーカイブ(FSA)ファイルが作成され、FPRファイルからソースコードが削除されます。FPRファイルからソースコードを抽出したり、既存のソースアーカイブ(FSA)をFPRファイルにマージしたり、ソースアーカイブからソースファイルを回復したりすることができます。

データをアーカイブするには、次のコマンドを入力します。

```
FPRUtility -sourceArchive -extract -project <project>.fpr -f <output_
archive>.fsa
```

データをディレクトリにアーカイブするには、次のコマンドを入力します。

```
FPRUtility -sourceArchive -extract -project <project>.fpr \ -
recoverSourceDirectory -f <output_dir>
```

アーカイブをFPRファイルに追加するには、次のコマンドを入力します。

```
FPRUtility -sourceArchive -mergeArchive -project <project>.fpr \ -source
<old_source_archive>.fsa -f <project_with_archive>.fpr
```

FPRファイルから失われたファイルを回復するには、次のコマンドを入力します。

```
FPRUtility -sourceArchive -fixSecondaryFileSources \ -payload <source_
archive>.zip -project <project>.fpr -f <output>.fpr
```

FPRUtilityソースアーカイブオプション

次 の表 は、ソースアーカイブの操 作 に適 用 されるFPRUtilityオプションを一 覧 表 示 しています。

FPRUtilityのオプション	説 明
-sourceArchive	FSAファイルを作 成 して、ソースアーカイブを抽 出 できるようにします。
次 のいずれか: -extract -mergeArchive -fixSecondaryFileSources	-extractオプションを使 用 すると、FPRファイルのコンテンツを抽 出 します。
	-mergeArchiveオプションを使 用 すると、FPRファイルのコンテンツを既 存 のアーカイブファイル(-sourceオプション)とマージします。
	-fixSecondaryFileSourcesオプションを使 用 すると、FPRファイルで見 つからないソースアーカイブ(-payloadオプション)からソースファイルを回 復 します。
-project *<project>*.fpr	アーカイブするFPRを指 定 します。
-recoverSourceDirectory	-extractオプションと一 緒 に使 用 して、復 元 されたソースファイルを含 むディレクトリとしてソースを抽 出 します。
-source *<old_source_archive>*.fsa	既 存 のアーカイブの名 前 を指 定 します。FPRファイルを既 存 のアーカイブとマージする(-mergeArchiveオプション)場 合 にのみ使 用 します。
-payload *<source_archive>*.zip	-fixSecondaryFileSourcesオプションと一 緒 に使 用 して、ソースファイルの回 復 元 となるソースアーカイブを指 定 します。
-f *<project_with_archive>*.fpr | *<output_archive>*.fsa | *<output_dir>*	出 力 ファイルを指 定 します。FPR、ディレクトリ、またはFSAファイルを生 成 できます。
-debug	FPRUtilityの問 題 のトラブルシューティングに役 立 つデバッグ情 報 を表 示 します。

FPRファイルの変更

FPRUtilityの-trimToLastScanオプションを使用すると、マージされたプロジェクト(FPR)から以前のスキャン結果が削除されます。これにより、以前のスキャン結果が不要になった場合にFPRファイルのサイズを小さくすることができます。また、Fortify Audit WorkbenchでFPRを開くのにかかる時間を短縮できます。

FPRから以前のスキャンを削除するには、次のコマンドを入力します。

```
FPRUtility -trimToLastScan -project <merged_project>.fpr [-f <output>.fpr]
```

FPRUtilityのFPRファイルの変更オプション

FPRUtilityのオプション	説明
-trimToLastScan	マージされたプロジェクトから以前のスキャン結果を削除します。
-project <merged_project>.fpr	変更するマージされたFPRを指定します。このプロジェクトがマージされたプロジェクトではない場合、FPRファイルは変更されません。
-f <output>.fpr	変更された出力ファイルの名前を指定します。このオプションを指定しない場合、マージされたFPRが変更されます。

FPRUtilityに対してメモリの割り当て量を増やす

大きな複雑なFPRファイルを使用してタスクを実行すると、メモリ不足エラーが発生する可能性があります。デフォルトでは、1000MBがFPRUtilityに割り当て済みです。メモリを増やすには、コマンドラインに-Xmxオプションを追加します。たとえば、FPRUtilityに2GBを割り当てるには、次のコマンドを使用します。

```
FPRUtility -Xmx2G -merge -project <primary>.fpr -source <secondary>.fpr \
-f <output>.fpr
```

コマンドラインからのレポートの生成

レポートを生成するには、次の2つのコマンドラインユーティリティがあります。

- BIRTReportGenerator–Business Intelligence and Reporting Technology (BIRT)システムに基づくレポートをFPRファイルから生成します。

注 : OpenJDKを実行しているテキストベースのLinuxシステムを使用している場合、BIRTレポートを正常に生成するには、DejaVu SansフォントとDejaVu Serifフォントをインストールする必要があります。

- ReportGenerator–レガシレポートをFPRファイルから生成します。レポートテンプレートを指定できます。指定しない場合は、デフォルトのレポートテンプレートが使用されます。使用可能なレポートテンプレートの説明については、*Micro Focus Fortify Audit Workbenchユーザガイド*を参照してください。

BIRTレポートの生成

BIRTレポートを生成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。

```
BIRTReportGenerator -template <template_name> -source <audited_proj>.fpr -
format <format> -output <report_file>
```

次に、追加オプションを指定してOWASP Top 10 2017レポートを生成する方法の例を示します。

```
BIRTReportGenerator -template "owasp top 10" -source auditedProj.fpr -
format pdf -showSuppressed --Version "owasp top 10 2017" --
UseFortifyPriorityOrder -output MyOWASP_Top10_Rpt.pdf
```

BIRTReportGeneratorコマンドラインオプション

次の表に、BIRTReportGeneratorオプションを示します。

BIRTReportGeneratorオプション	説明
`-template <template_name>`	(必須)レポートテンプレート名を指定します。*<template_name>*の有効な値は、"CWE Top 25"、"CWE/SANS Top 25"、"Developer Workbook"、"DISA CCI 2"、"DISA STIG"、"FISMA Compliance"、GDPR、MISRA、"OWASP ASVS 4.0"、"OWASP Mobile Top 10"、"OWASP Top 10"、"PCI DSS Compliance"、および"PCI SSF Compliance"です。 **注 :** *<template_name>*にスペースが存在する場合は、レポートテンプレート名を引用符で囲むだけですみます。テンプレート名の値では、大文字と小文字を区別しません。
`-source <audited_proj>.fpr`	(必須)レポートのベースとなる監査されたプロジェク

BIRTReportGeneratorオプション	説明
	トを指定します。
`-format pdf | doc | html | xls`	(必須)生成されるレポート形式を指定します。 **注:** フォーマットの値では、大文字と小文字を区別しません。
`-output <report_file.***>`	(必須)レポートの書き込みファイルを指定します。 **注:** すでに存在するファイルを指定すると、そのファイルは上書きされます。
`-searchQuery <query>`	レポートを生成する前に、問題をフィルタするための検索クエリを指定します。例: `-searchQuery audited:false` 検索クエリ構文の詳細については、*Micro Focus Fortify Audit Workbenchユーザガイド*を参照してください。
`-showSuppressed`	抑制済みとしてマークされている問題を含めます。
`-showRemoved`	削除済みとしてマークされている問題を含めます。
`-showHidden`	非表示としてマークされている問題を含めます。
`-filterSet <filterset_name>`	レポートの生成に使用するフィルタセットを指定します(たとえば`-filterSet "Quick View"`)。
`--Version <version>`	テンプレートのバージョンを指定します。テンプレートバージョンの有効な値を次に示します。 **注:** ここにリストされていないテンプレートで使用可能なバージョンは1つのみです。 複数のバージョンが使用可能なときにバージョンを指定しない場合は、FPRの作成時に使用された外部メタデータに基づいた最新バージョンがデフォルトで使用されます。テンプレートバージョンの値では、大文字と小文字を区別しません。

BIRTReportGeneratorオプション	説 明
	• 「CWE Top 25」テンプレートの場 合 、バージョンは"CWE Top 25 *<year>*"です(例 : "CWE Top 25 2020") • 「CWE/SANS Top 25」テンプレートの場 合 、バージョンは"*<year>* CWE/SANS Top 25"です(例 : "2011 CWE/SANS Top 25") • 「DISA STIG」テンプレートの場 合 、バージョンは"DISA STIG *<version>*"です(例 : "DISA STIG 5.1") • 「FISMA Compliance」テンプレートの場 合 、バージョンは"NIST 800-53 Rev *<version>*"です(例 : "NIST 800-53 Rev 5") • MISRAテンプレートの場 合 、使 用 可 能 なバージョンは"MISRA C 2012"または "MISRA C++ 2008"です • 「OWASP Top 10」テンプレートの場 合 、バージョンは"OWASP Top 10 *<year>*"です(例 : "OWASP Top 10 2017") • 「PCI DSS Compliance」テンプレートの場 合 、バージョンは"*<version>* Compliance"です(例 : "3.2 Compliance") • 「PCI SSF Compliance」テンプレートの場 合 、バージョンは"PCI SSF *<version>*"です(例 : "PCI SSF 1.1")
--IncludeDescOfKeyTerminology	レポートに［*Description of Key Terminology*］セクションを含 めます。
--IncludeAboutFortify	レポートに［*About Fortify Solutions*］セクションを含 めます。
--SecurityIssueDetails	報 告 された問 題 の詳 細 な説 明 を提 供 します。このオプションは、Developer Workbookテンプレートでは使 用 できません。
--UseFortifyPriorityOrder	問 題 をカテゴリ分 けするには、フォルダ名 の代 わりに［Fortify Priority Order］を使 用 します。このオプションは、Developer WorkbookおよびPCI

BIRTReportGeneratorオプション	説 明
	Complianceテンプレートでは使 用 できません。
-h | -help	オプションの詳 細 情 報を表 示します。
-debug	BIRTReportGeneratorの問 題 のトラブルシューティングに役 立 つデバッグ情 報を表 示します。

BIRTReportGeneratorのトラブルシューティング

場 合 によっては、レポートを生 成 する際 に、メモリ不 足 エラーが発 生します。コマンドライン出力 に次 のようなメッセージが表 示される場 合 があります。

```
java.lang.OutOfMemoryError: GC overhead limit exceeded
```

BIRTReportGeneratorに割り当 てられたメモリを増 やすには、次 の手 順 を実 行します。

1. バックアップとして*<sca_install_dir>*/bin/BIRTReportGeneratorファイルのコピーを作 成 します。
2. テキストエディタで*<sca_install_dir>*/bin/BIRTReportGeneratorファイルを開きます。
   ファイルの最 後 の行 には-Xmx1500Mがあります。これは、BIRTReportGeneratorに提 供 されるヒープメモリの合 計 です。この行 は次 のようになります。

   ```
   "%FORTIFY_CORE%\private-bin\awb\eclipse\eclipsec.exe" -vm "%JAVA_CMD%" ...
   -Xmx1500M -Dcom.fortify.log.console=%CONSOLE_LOG% %USER_VM_OPTS%
   ```

3. この行 の-Xmxオプションの値 を変 更して、BIRTReportGeneratorのヒープメモリ量 を増 やします。たとえば、32GBをBIRTReportGeneratorに割り当 てるには、オプションを-Xmx32Gに設 定します。

   注 : 物 理 的に使 用 可 能 な量 より多 くのメモリを割り当 てないでください。

4. ファイルを保 存して閉じてから、レポートを再 実 行します。

レガシレポートの生 成

PDFレポートを生 成 するには、次 のコマンドを入 力します。

```
ReportGenerator -format pdf -f <myreport>.pdf -source <myresults>.fpr
```

XMLレポートを生 成 するには、次 のコマンドを入 力します。

```
ReportGenerator -format xml -f <myreport>.xml -source <myresults>.fpr
```

ReportGeneratorコマンド ラインオプション

次 の表 に、ReportGeneratorオプションを示 します。

ReportGeneratorオプション	説 明
`-format pdf | xml`	(必 須)生 成 されるレポート形 式 を指 定 します。
`-source <audited_proj>.fpr`	(必 須)レポート のベースとなる監 査 されたプロジェクトを指 定 します。
`-f <report_file.***>`	(必 須)レポート の書 き込 みファイルを指 定 します。
`-template <template_name>`	レポートテンプレートを指 定 します。指 定 されていない場 合 、ReportGeneratorではデフォルト のテンプレートを使 用 します。デフォルト のテンプレートは`<sca_install_dir>/Core/config/reports/DefaultReportDefinition.xml`にあります。 **注 :** 空 白 が含 まれる場 合 は、*<template_name>*を引 用 符 で囲 む必 要 があります。
`-user <username>`	レポートに追 加 するユーザ名 を指 定 します。
`-showSuppressed`	抑 制 済 みとしてマークされている問 題 を含 めます。
`-showRemoved`	削 除 済 みとしてマークされている問 題 を含 めます。
`-showHidden`	非 表 示 としてマークされている問 題 を含 めます。
`-filterSet <filterset_name>`	レポート の生 成 に使 用 するフィルタセットを指 定 します (たとえば`-filterset "Quick View"`)。
`-verbose`	コンソールにステータスメッセージを表 示 します。
`-debug`	ReportGeneratorの問 題 のトラブルシューティングに役 立 つデバッグ情 報 を表 示 します。
`-h`	オプションの詳 細 情 報 を表 示 します。

Fortify Static Code Analyzerスキャンステータスの確認

SCAStateユーティリティを使用して、スキャンフェーズ中に最新の状態分析情報を確認します。

Fortify Static Code Analyzerの状態を確認するには、次の手順を実行します。

1. Fortify Static Code Analyzerスキャンを実行します。
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>]
```

SCAStateユーティリティのコマンドラインオプション

次の表に、SCAStateユーティリティオプションを示します。

SCAStateオプション	説明
`-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ユーティリティ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Fortify Static Code Analyzerの状態情報を表示します。

SCAStateオプション	説明
`-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。
`-pid <process_id>`	現在実行中のFortify Static Code AnalyzerプロセスIDを指定します。複数のFortify Static Code Analyzer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Fortify Static Code Analyzerバージョン番号を表示します。
`-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`	Fortify Static Code Analyzer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

SCAStateオプション	説明
	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)。

注 : Fortify Static Code Analyzer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。Fortifyでは、SCAStateをWindows管理者として実行することを推奨します。

第 19章 : パフォーマンスの改 善

この章 では、Fortify Static Code Analyzer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。

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)

Fortify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)	説明
単純	4	16	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ユーティリティなど)。
中間	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)。
複雑	16	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)。
非常に複雑	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)。

注 : 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、ドキュメント『*Micro Focus Fortifyソフトウェアシステム要件*』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、Fortify Static Code Analyzer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** - Fortify Static Code Analyzer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。Fortifyでは、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、「"メモリのチューニング" ページ 176」を参照してください。
- **CPU** - Fortifyでは、2.1 GHz以上のプロセッサが推奨されています。

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Fortify Static Code Analyzerバージョン 21.2.0を使 用して実 行 されました。これらのスキャンは、Micro Focus Fortify Software Security Content 2021 Update 3を使 用して実 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。

プロジェクト名	言 語	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
nasm 0.98.38	C/C++	00:30	04:03	1,145	21,449	8個 のCPUと32GBの RAMを搭 載 したLinux VM
WebGoat 8	Java	00:39	01:00	568	5,223	8個 のCPUと32GBの RAMを搭 載 したLinux VM
WordPress for Android	Java	00:24	01:13	430	10,034	
CakePHP	PHP	00:26	01:50	2,368	55,048	
phpBB 3	PHP	00:33	01:48	1,285	39,610	
SharpZipLib	.NET (C#)	01:54	02:59	1,491	12,185	8個 のCPUと32GBの RAMを搭 載 した Windows VM
Hackademic-next	JavaScript	01:37	03:37	464	44,586	8個 のCPUと32GBの RAMを搭 載 したLinux VM
prisma	TypeScript	00:50	02:21	58	23,739	
numpy-1.13.3	Python 3	02:44	09:39	261	90,921	
MediaBrowser	Swift	00:40	01:44	27	6,225	2個 のCPUと8GBの RAMを搭 載 した macOS VM

チューニングオプション

Fortify Static Code Analyzer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

Fortify Static Code Analyzer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Micro Focus Fortify Software Security Center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	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" ページ 48
分 析	`-quick`	クイックスキャンの実 行	"クイックスキャン" 下
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 " ページ165
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 " ページ166
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" ページ176
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" ページ176
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" ページ168
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 " ページ170

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。Fortify Static Code Analyzerは、スキャンの深 さを減 らし、クイックビューフィルタセットを適 用 することで、スキャンを高 速 に実 行 します。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" ページ221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

Fortify Static Code Analyzer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。Fortify Static Code Analyzer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、Fortify Static Code Analyzer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。

Fortify Static Code Analyzerでは、ユーザがFortify Static Code Analyzer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" ページ221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、`-scan`オプションとともに`-quick`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、Fortify Static Code Analyzerは標 準 の*<sca_install_dir>*`/Core/config/fortify-sca.properties`ファイルに加 えて*<sca_install_dir>*`/Core/config/fortify-sca-quickscan.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、Fortify Static Code Analyzer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。Fortifyでは、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について詳 しくは、"Fortify Static Code Analyzerのプロパティファイル" ページ194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。Micro Focus 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Micro Focus Fortify Software Security Centerサーバ** - フルスキャンの結 果 が上 書 きされるのを防 ぐため、Fortify Software Security Centerでは、クイックスキャンモードでスキャンされ、アップロードされたFPRファイルはデフォルトで無 視 されます。ただし、Fortify Software Security Center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、*Micro Focus Fortify Software Security Center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。

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、`<sca_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。

重 要 設 定 ファイルを変 更した場 合 、これらのファイルはFortify Static Code Analyzerのアップグレード によって上 書 きされる可 能 性 があることに注 意 してください。

メモ:

- Micro Focus Fortify Software Security Center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、Fortify Software Security Center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
```

注 : 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
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場 合 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

精度レベル	説明
	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。

fortify-sca.propertiesファイルのcom.fortify.sca.PrecisionLevel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
```

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ほとんどない場合は、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)。Fortify Static Code Analyzer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Fortify Static Code Analyzer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
-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。

オプション	説 明
`-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 、Fortify Static Code Analyzerではバイナリごとにツリーが表 示 されます。

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Fortify Static Code Analyzerで変 換 する言 語 を制 限 できます。

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Fortify Static Code Analyzer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、`buffercontentconfiguration`、`controlflow`、`dataflow`、`nullptr`、`structural`、および`semantic`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
```

Fortify Static Code Analyzerのプロパティファイル(*<sca_install_dir>*`/Core/config/fortify-sca.properties`)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
```

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、`abap`、`actionscript`、`apex`、`cfml`、`cobol`、`configuration`、`cpp`、`dotnet`、`golang`、`java`、`javascript`、`jsp`、`kotlin`、`objc`、`php`、`plsql`、`python`、`ruby`、`scala`、`sql`、`swift`、`tsql`、`typescript`、`vb`です。

たとえば、SQL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 sql:php
```

Fortify Static Code Analyzerのプロパティファイル(*<sca_install_dir>*`/Core/config/fortify-sca.properties`)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=sql:php
```

FPRファイルの最 適 化

この章 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。これには、スキャン時 間 の短 縮 、FPRファイルサイズの削 減 、および大 きなFPRファイルを開 くためのヒントが含 まれます。

フィルタファイル

フィルタファイルは、スキャンで`-filter`オプションを使 用 して指 定 できるフラットファイルです。フィルタファイルを使 用 して、特 定 の脆 弱 性 インスタンス、ルール、および脆 弱 性 カテゴリを除 外 します。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、これらのタイプの問 題 にフラグを付 けてFPRに追 加 しないようにFortify Static Code Analyzer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 み込 むだけの単 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 されている可 能 性 が高 いため、表 示 する必 要 はありません。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。スキャンの`-filter`オプションは、次 の例 のように使 用 します。

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

`MyResults.fpr`には、パス操 作 カテゴリの問 題 は含 まれません。

フィルタファイルについて詳しくは、"分 析 のフィルタリング" ページ 184を参 照してください。

フィルタセットによるFPRからの問 題 の除 外

問 題 テンプレートのフィルタによって、Fortify Static Code Analyzerの結 果 の表 示 方 法 が決 まります。たとえば、検 出 されたSQLインジェクションの問 題 を`SQL Injections`という名 前 の別 のディレクトリに入 れるフィルタや、信 頼 性 が一 定 のしきい値 を下 回 る問 題 を非 表 示 にするフィルタを使 用 できます。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、Micro Focus Fortify Audit Workbenchを使 用 してフィルタとフィルタセットを作 成 し、フィルタセットを使 用 してFortify Static Code Analyzer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、『*Micro Focus Fortify Audit Workbenchユーザガイド*』を参 照 してください。次 の例 では、スキャン時 間 フィルタを作 成 して使 用 する基 本 的 な手 順 について説 明 します。

1. この例 では、OWASP Top 10 2017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で次 のようなフィルタを作 成 します。

   ```
   [OWASP Top 10 2017]にAが含 まれていない場 合 は問 題 を非 表 示 にする
   ```

   このフィルタは、問 題 を検 索 して、名 前 に「A」を含 むOWASP Top 10 2017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 2017のカテゴリはすべて「A」(A1、A2、...、A10)で始 まるため、文 字 「A」を含 まないカテゴリはOWASP Top 10 2017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。
2. Fortify Audit Workbenchで、前 のフィルタを含 む`OWASP_Filter_Set`という名 前 の新 しいフィルタセットを作 成 し、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。
3. その後 、スキャン時 に次 のコマンドを使 用 してこのフィルタを指 定 できます。

   ```
   sourceanalyzer -b MyProject -scan -f myFilteredResults.fpr -project-
   template IssueTemplate.xml -Dcom.fortify.sca.FilterSet=OWASP_Filter_set
   ```

前 の例 では、`-Dcom.fortify.sca.FilterSet`プロパティを追 加 して、問 題 テンプレート`IssueTemplate.xml`の`OWASP_Filter_Set`フィルタセットを使 用 するようにFortify Static Code Analyzerに指 示 しています。ビューに問 題 を表 示 しないフィルタは削 除 され、FPRに書 き込 まれません。したがって、表 示 される問 題 の数 を減 らし、スキャン対 象 を大 幅 に絞 り込 み、結 果 のFPRファイルのサイズを削 減 することができます。

注 : 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。Fortify Static Code Analyzer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、Fortify Static Code Analyzerがロードするルールタイプが決 まります。

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ファイルのスキャン時 間とサイズを削 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。一 般 に、小さなソースファイルのスキャン時 間 を短 縮 する必 要 はありません。

Fortify Static Code AnalyzerでFPRにソースコードが含 まれないようにするには、2つの方 法 があります。*<sca_install_dir>*/Core/config/fortify-sca.propertiesファイル内 のプロパティを設 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
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 のコマンドラインの例 では、両 方 のオプションを使 用しています。

```
sourceanalyzer -b MyProject -disable-source-bundling -fvdl-no-snippets -
scan -f MySourcelessResults.fpr
```

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 " 上 で説 明しているように、FPRからソースコードを除 外 することです。"FPRファイルの変 更 " ページ151で説 明しているように、FPRUtilityを使 用して以 前 のスキャン結 果 を削 除 することで、マージされたFPRのサイズを削 減 することもできます。

FPRから除 外 する項 目 を選 択 するために使 用 できるオプションとプロパティが他 にもいくつかあります。これらのプロパティは、Fortify Static Code Analyzerのプロパティファイル(*<sca_install_dir>*/Core/config/fortify-sca.properties)で設 定 するか、スキャンフェーズで-D*<property_name>*=trueを使 用して指 定 することができます。これらのオプションのほとんどに、同 等 のコマンドラインオプションがあります。

プロパティ名	説 明
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。Micro Focus 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。 **注 :** FPRからエンジンデータを除 外 する場 合 は、FPRをMicro Focus Fortify Software Security Center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Fortify Static Code Analyzerのバージョンが含 まれていないため、Fortify Software Security Centerはサーバ上 でFPRをマージできません。
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。

大きなFPRファイルを開く

<sca_install_dir>/Core/config/fortify.properties環境設定ファイルには、大きなFPRファイルを開くのにかかる時間を短縮するために設定できるプロパティがあります。これらのプロパティについて詳しくは、『*Micro Focus Fortify Static Code Analyzerツールプロパティリファレンスガイド*』を参照してください。次の表では、これらのプロパティについて説明します。

プロパティ名	説明
com.fortify.model.DisableProgramInfo=true	この設定は、Micro Focus 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)
	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)	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に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。
com.fortify.model.MaxEngineErrorCount=*<num>*	このプロパティは、Fortify Static Code Analyzerによって報告され、FPRでロードされる警告の数を指

プロパティ名	説 明
	定します。スキャンの警 告 が多 いプロジェクトでは、これにより、Fortify Audit Workbenchでのロード時 間 とFPRを開くのに必 要 なメモリ量 の両 方 を削 減 できます。
`com.fortify.model.ExecMemorySetting`	Audit Workbenchでiidmigratorやfortifyupdateなどの外 部 ユーティリティを起 動 するためのJVMヒープメモリサイズを指 定します。

長時間実行されているスキャンの監視

Fortify Static Code Analyzer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。Fortifyでは、デバッグログをMicro Focus Fortifyカスタマサポートチームに提 供 することを推 奨していますが、Fortify Static Code Analyzer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。

SCAStateユーティリティの使 用

SCAStateコマンドラインユーティリティを使 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ユーティリティは*<sca_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、Fortify Static Code Analyzerがスキャン中 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ユーティリティの使 い方 について詳 しくは、*"Fortify Static Code Analyzerスキャンステータスの確 認 " ページ 157*を参 照 してください。

JMXツールの使 用

ツールを使 用 すると、JMXテクノロジでFortify Static Code Analyzerを監 視 できます。これらのツールでは、長 期 間 にわたってFortify Static Code Analyzer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、完 全 なOracleドキュメント (http://docs.oracle.comで入 手 可 能)を参 照 してください。

注 : これらはサードパーティのツールであり、Micro Focusでは提 供 またはサポートされていません。

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、Fortify Static Code Analyzerのスキャンを開 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Fortify Static Code Analyzerをローカルまたはリモートで監 視 します。

```
jconsole <host_name>:9090
```

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 " 前 のページ」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
```

第20章: 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。

終了コード

次の表に、取り得るFortify Static Code Analyzer終了コードを示します。

終了コード	説明
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、Fortify Static Code Analyzerでサポートされていないファイル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、Fortify Static Code Analyzerでは終了コード0、1、2、3、または7だけが返されます。

<sca_install_dir>/Core/Config/fortify-sca.properties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。

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、Fortify Static Code Analyzer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" ページ131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。

注: 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。

Javaヒープの枯渇

Javaヒープの枯渇は、Fortify Static Code Analyzer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、Fortify Static Code Analyzer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Fortify Static Code Analyzer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Fortify Static Code Analyzer 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、

Fortify Static Code Analyzer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。

32GB～48GB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、32GB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 になる可 能 性 があります。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Fortify Static Code Analyzer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。

注 : 常 駐 していても(Fortify Static Code Analyzer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Fortify Static Code Analyzer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。

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、Fortify Static Code Analyzer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。

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Fortify Static Code Analyzerログファイルに出現しますが、コマンドラインの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、`sourceanalyzer`コマンドに`-Xss`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。

複雑な関数のスキャン

Fortify Static Code Analyzer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

Fortify Static Code Analyzer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。Fortify Static Code Analyzerでは、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とります。このため、Fortify Static Code Analyzer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、Fortify Static Code Analyzer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、それまでのアクセスで学習した情報があれば結果に含まれます。

リミッタと呼ばれるFortify Static Code Analyzer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。

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、Fortify Static Code Analyzer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、Fortify Static Code Analyzerプロパティファイル*<sca_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。Fortify Static Code Analyzerで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ではその関数を(`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、Fortify Static Code Analyzerでは、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、Fortify Static Code Analyzer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。Fortify Static Code Analyzer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」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。

制御フローアナライザと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、制御フローアナライザとNullポインタアナライザは、非常に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、Fortify Static Code Analyzer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、Fortify Static Code Analyzerプロパティファイル*<sca_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、Fortify Static Code Analyzerの物理メモリを増やします。

注: 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。

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、Micro Focus Fortify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. *<sca_install_dir>*/core/config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
```

ログファイルの場 所 の確 認

デフォルトでは、Fortify Static Code Analyzerによって次 の場 所 に2つのログファイルが作 成 されます。

- Windowsの場 合 : C:\Users*<username>*\AppData\Local\Fortify\sca*<version>*\log
- その他 のプラットフォームの場 合 : *<userhome>*/.fortify/sca*<version>*/log

ここで、*<version>*は使 用 しているFortify Static Code Analyzerバージョンです。

次 の表 は、2つのログファイルについて説 明 しています。

デフォルトのファイル名	説 明
sca.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	Fortify Supportログの内 容 は次 のとおりです。 • 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、Micro Focus Fortifyカスタマサポートまたは開 発 チームが発 生 する可 能 性 のある問 題 をトラブルシューティングする場 合 にのみ役 立 ちます。

解 決 できない警 告 またはエラーが発 生 した場 合 は、Fortify SupportログファイルをMicro Focus Fortifyカスタマサポートにお送 りください。

ログファイルの設 定

Fortify Static Code Analyzer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 ("fortify-sca.properties" ページ196を参 照)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" 下 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`

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。*<sca_install_dir>*`/Core/config/fortify.sca.properties`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
DEBUG	Micro Focus Fortify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

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、Micro Focus Fortify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 : Fortify Static Code Analyzer
- Fortify Static Code Analyzerのバージョン番号および任意の独立したFortify Static Code Analyzer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。

付録A: 分析のフィルタリング

このセクションでは、次のトピックについて説明します。

フィルタファイル

`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。

注: Fortifyでは、上級ユーザである場合にのみフィルタファイルを使用することが推奨されています。通常、監査人はFortify Static Code Analyzerで検出されたすべての問題を確認して評価する必要があるため、標準監査ではフィルタファイルを使用しないでください。

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。各フィルタ項目は、フィルタファイル内の個別の行にあります。次のフィルタタイプを指定できます。

- カテゴリ
- インスタンスID
- ルールID

フィルタは、フィルタのタイプに基づいて分析プロセスで別々の時間に適用されます。Fortify Static Code Analyzerでは、分析が実行される前に初期化フェーズ時にカテゴリおよびルールIDフィルタが適用されます。一方、インスタンスIDフィルタは分析フェーズ後に適用されます。

フィルタファイルの例

たとえば、次の出力は`<sca_install_dir>/Samples/basic/eightball`ディレクトリにある`EightBall.java`のスキャンによる出力です。

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コンテンツの例 は、次 のとおりです。

- 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。

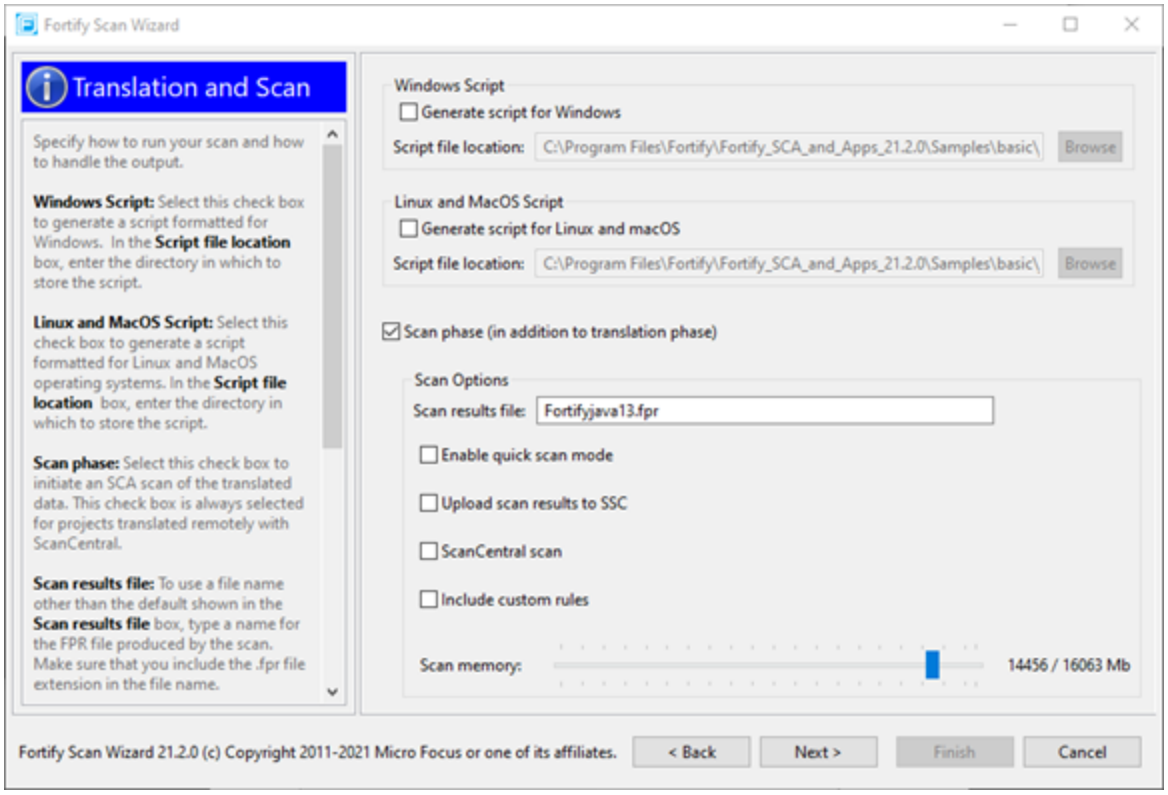
```
[176CC0B182267DD538992E87EF41815F : critical : Path Manipulation :
dataflow ] EightBall.java(12) : ->new FileReader(0) EightBall.java(6) :
<=> (filename) EightBall.java(4) : ->EightBall.main(0)
[E4B3ACF92911ED6D98AAC15876739EC7 : high : Unreleased Resource : Streams :
controlflow ] EightBall.java(12) : 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
```

付録B: Fortify Scan Wizard

このセクションでは、次のトピックについて説明します。

Fortify Scan Wizardを使用するための準備

Fortify Scan Wizardでは、ユーザが指定した情報を使用して、Fortify Static Code Analyzerがプロジェクトコードをスキャンし、必要に応じて分析結果をMicro Focus Fortify Software Security Centerにアップロードするためのコマンドを含むスクリプトを作成します。Fortify Scan Wizardを使用して、スキャンをローカルで実行するスクリプトや、スキャンをMicro Focus Fortify ScanCentral SASTに送信して分析のすべてまたは一部を実行するスクリプトを作成できます。



Fortify Scan Wizardを使用するには、スキャンするプロジェクトのビルドディレクトリにアクセスする必要があります。次の表では、プロジェクトの分析方法とスキャン結果をFortify Software Security Centerにアップロードするかどうかに応じて、必要となる情報の一部について説明しています。

重要 Fortify Software Security CenterまたはFortify ScanCentral SASTコントローラで内部認証局からのSSL接続または自己署名証明書を使用する場合は、証明書をFortify Static Code AnalyzerのJavaキーストアに追加する必要があります("信頼された証明書の追加" ページ44を参照)。

タスク	要件
Fortify ScanCentral SASTによるリモート分析(変換およびスキャンフェーズ)を実行する	Fortify ScanCentral SASTコントローラのURL **注:** Fortify Software Security Centerに分析結果もアップロードする場合、コントローラのURLを指定する必要はありません。この場合は、Fortify Software Security Centerサーバと統合されたFortify ScanCentral SASTが使用されます。
	プロジェクトの言語がFortify ScanCentral SASTで変換できる言語であることを確認します。サポートされている言語のリストについては、『*Micro Focus Fortifyソフトウェアシステム要件*』を参照してください。
ローカルのFortify Static Code Analyzer変換とFortify ScanCentral SASTによるリモートスキャンを実行する	Fortify ScanCentral SASTコントローラのURL
分析結果をFortify Software Security Centerにアップロードする	Fortify Software Security Centerサーバ のURL **注:** Fortify ScanCentral SASTを使用する場合は、Fortify Software Security Centerサーバ をFortify ScanCentral SASTコントローラと統合する必要があります。
	Fortify Software Security Centerのログインアカウント情報 **注:** Fortify Software Security Centerのログインアカウント情報がない場合は、Fortify Software Security Centerに存在するアプリケーション名とバージョンが必要になります。

タスク	要 件
	ToolsConnectTokenタイプの認 証トークン **注 :** トークンがない場 合 は、Fortify Scan Wizardを使 用 してトークンを生 成 できます。これを行うには、Fortify Software Security Centerのログインアカウント情 報 が必 要 です。

注 : Windowsシステム用 のスクリプトを生 成 する場 合 、そのスクリプトをWindows以 外 のシステムで実 行 することはできません。同 様 に、Windows以 外 のシステム用 のスクリプトを生 成 した場 合 、そのスクリプトをWindowsシステムで実 行 することはできません。

Fortify Scan Wizardの開 始

ローカルにインストールされたFortify SCAとアプリケーションでFortify Scan Wizardを開 始 するには、使 用 するオペレーティングシステムに応じて、次 のいずれかを実 行 します。

- Windowsでは、 **[スタート]> [すべてのプログラム]> [SCA and Applications *<version>*]> [Scan Wizard]**の順 に選 択 します。
  コマンドプロンプトウィンドウを開 いて、「`scanwizard`」と入 力 することもできます。
- Linuxでは、*`<sca_install_dir>`*`/bin`ディレクトリに移 動 して、コマンドラインから`ScanWizard`を実 行 します。
- macOSでは、*`<sca_install_dir>`*`/bin`ディレクトリに移 動 して、`ScanWizard`をダブルクリックします。

付録C: サンプルプロジェクト

Fortify SCAとアプリケーションのインストールには、Fortify Static Code Analyzerの使用方法を学習する際に使用できるコードサンプルがいくつか含まれている場合があります。サンプルファイルをインストールした場合は、次のディレクトリにあります。

<sca_install_dir>/Samples

Samplesディレクトリには、basicとadvancedの2つのサブディレクトリがあります。各コードサンプルには、Fortify Static Code Analyzerでコードをスキャンしてその結果をMicro Focus 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サブディレクトリには、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サブディレクトリには、Fortify Static Code Analyzerとバグトラッカーアプリケーションの統合に役立つソースコードを含む、より高度なサンプルが含まれています。バグトラッカーアプリケーションとFortify Audit Workbenchの統合については、『*Micro Focus Fortify Audit Workbench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。

基本的なサンプル

次の表では、*<sca_install_dir>*/Samples/basicディレクトリ内のサンプルファイルについて説明し、サンプルが示す脆弱性のリストを示します。サンプルの多くには、サンプルの使用に関する詳細と手順が記載されたREADME.txtファイルが含まれています。

フォルダ名	説明	脆弱性
cpp	C++サンプルファイルと単純なデータフロー脆弱性を含むコードを分析する手順。Fortifyで分析するには、gccまたはclコンパイラが必要です。	Command Injection Memory Leak
database	database.pksサンプルファイル。このSQLサンプルには、SQLコードの問題が含まれています。	Access Control: Database

フォルダ名	説 明	脆 弱 性
eightball	誤 ったエラー処 理 を示 すJavaアプリケーション (`EightBall.java`)。整 数 の引 数 が必 要 です。引 数 として整 数 ではなくファイル名 を指 定 すると、ファイルの内 容 が表 示 されます。	Path Manipulation Unreleased Resource: Streams Unchecked Return Value J2EE Bad Practices: Leftover Debug Code
formatstring	`formatstring.c`ファイル。Fortifyで分 析 するには、gccまたはclコンパイラが必 要 です。	Format String
java13	`Sample.java`ファイル。	Privacy Violation Insecure Randomness: Hardcoded Seed J2EE Bad Practices: Leftover Debug Code
javascript	`sample.js` JavaScriptファイル。	Cross-Site Scripting Open Redirect Privacy Violation
nullpointer	`NullPointerSample.java`ファイル。	Null Dereference
php	2つのPHPファイル(`sink.php`および `source.php`)。`source.php`を分 析 すると、単 純 なデータフローの脆 弱 性 と危 険 な関 数 が明 らかになります。	Cross-Site Scripting SQL Injection
sampleOutput	WebGoatプロジェクト のサンプル出 力 ファイル (`WebGoat5.0.fpr`)。	各 種
stackbuffer	`stackbuffer.cpp`ファイル。Fortify Static Code Analyzerでこのサンプルを分 析 するには、g++またはclコンパイラが必 要 です。	Buffer Overflow
toctou	`toctou.c`ファイル。Fortify Static Code Analyzerでこのサンプルを分 析 するには、gccまたはclコンパイラが必 要 です。	Time-of-Check/Time-of-Use (Race Condition)

フォルダ名	説 明	脆 弱 性
vb6	`command-injection.bas`ファイル。	Command Injection SQL Injection
vbscript	`source.asp`および`sink.asp`ファイル。	SQL Injection

高 度なサンプル

次 の表 では、*`<sca_install_dir>`*`/Samples/advanced`ディレクトリ内 のサンプルについて説 明 します。サンプルの多 くには、サンプルの分 析 方 法 に関 する追 加 の詳 細 と手 順 を提 供 する `README.txt`ファイルが含 まれています。

フォルダ名	説 明
BugTrackerPlugin *<bug_tracker>*	サポートされているバグトラッカープラグインのソースコード が含 まれています。
c++	Visual Studioの各 サポート対 象 バージョンのサンプルソリューション。 このサンプルを使 用 するには、次 のコマンド がインストールされている必 要 があります。 • Visual Studio Visual C/C++のサポート対 象 バージョン • Fortify Static Code Analyzer • README.txtファイルの説 明 に従 ってVisual Studioのサンプルを分 析 するには、使 用 するVisual Studioバージョン用 のMicro Focus Fortify Extension for Visual Studioがインストールされている必 要 があります。 このコード には、コマンド インジェクションの問 題 と未 チェックの戻 り値 の問 題 が含 まれています。
configuration	Webモジュール展 開 記 述 子 `web.xml`に脆 弱 性 があるサンプルJava EEアプリケーション。
crosstier	複 数 のアプリケーション技 術 (Java、PL/SQL、JSP、およびStruts) にまたがる脆 弱 性 を持 つサンプル。 この出 力 には、アクセス制 御 に関 する2つの脆 弱 性 を含 む、さまざまなタイプの問 題 が含 まれています。これらの問 題 の1つがクロスティアの結 果 です。Javaコード 内 のユーザ入 力 から取 得 したデータフローのトレースが含 まれており、PL/SQLの`SELECT`ステートメントに

フォルダ名	説 明
	影 響 を与 える可 能 性 があります。
csharp	SQLインジェクションの脆 弱 性 を含 む簡 単 なC#プログラム。Visual Studioの各 サポート対 象 バージョンに対 応 するバージョンが含 まれています。このサンプルの分 析 により、SQLインジェクション、未 公 開 リソース、およびパス操 作 の脆 弱 性 が明らかになります。スキャンで使 用 されるルールパックのバージョンによっては、他 のカテゴリも存 在 する可 能 性 があります。
customrules	4種 類 のアナライザ(Semantic、Dataflow、Control Flow、Configuration)によるルールの解 釈 方 法 を示 す単 純 なソースコードサンプルとルールパックファイル。このフォルダには、実 際 のアプリケーションをスキャンするために使 用 できる実 際 のルールの雑 多 なサンプルも含 まれています。
ejb	サーブレットとEJBを含 むJava EEのサンプルクロスティアアプリケーション。
filters	Fortify Static Code Analyzer `-filter`オプションを使 用 するサンプル。
javaAnnotations	使 用 時 に発 生 する可 能 性 がある問 題 と、FortifyのJava注 釈 に関 する問 題 の修 正 方 法 を示 すサンプルアプリケーション。 このサンプルは、Fortifyの注 釈 を使 用 して報 告 される脆 弱 性 の精 度 を向 上 させる方 法 を示 しています。`README.txt`ファイルには、このサンプルアプリケーションに関 連 する潜 在 的 な問 題 と解 決 策 が記 述 されています。
JavaDoc	`public-api`および`WSClient`のJavaDocディレクトリ。
swift	iGoat-Swiftフォルダには、Open Web Application Security Project (OWASP)によって提 供 されるiGoat-Swiftソースが含 まれています。このプロジェクトを分 析 するには、サポート対 象 バージョンのxcodebuildがインストールされている必 要 があります。`README.txt`ファイルには、このアプリケーションの分 析 で検 出 される脆 弱 性 が記 述 されています。

付録D: 環境設定オプション

Fortify SCAとアプリケーションインストーラでは、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、Micro Focus Fortify Static Code Analyzer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。

Fortify Static Code Analyzerのプロパティファイル

プロパティファイルは、`<sca_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。Fortifyでは、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。

次の表では、主要なプロパティファイルについて説明します。追加のプロパティファイルについては、『*Micro Focus Fortify Static Code Analyzerツールプロパティリファレンスガイド*』を参照してください。

プロパティファイル名	説明
`fortify-sca.properties`	Fortify Static Code Analyzer環境設定プロパティを定義します。
`fortify-sca-quickscan.properties`	Fortify Static Code Analyzer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。

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。

注: 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
```

プロパティ設定の優先順位

Fortify Static Code Analyzer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、Fortify Static Code Analyzer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
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	Fortify Static Code Analyzerのクイックスキャン環境設定ファイル	**注:** 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。
	Fortify Static Code Analyzer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。
3	Fortify Static Code Analyzer環境設定ファイル	Fortify Static Code Analyzer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、Fortify Static Code Analyzer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。

fortify-sca.properties

次の表に、fortify-sca.propertiesファイルで使用できるプロパティの概要を示します。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" ページ221を参照してください。各プロパティの説明には、値のタイプ、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)、および例が含まれています。

プロパティ名	説明
com.fortify.sca.lim.Url	LIMサーバの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。 **値のタイプ:** 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*>/LIM.REST.API
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。 **値のタイプ:** 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.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。 **値のタイプ:** 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。 **値のタイプ:** 文字列 **デフォルト:** (なし) **例:** http://proxy.example.com:8080 https://proxy.example.com

プロパティ名	説明
	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。 **値のタイプ:** 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.ProxyPassword`	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。 **値のタイプ:** 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警告メッセージが表示されます。 **値のタイプ:** 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、Fortify Static Code Analyzer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Fortify Static Code Analyzerが終了します。 **値のタイプ:** ブール値 **デフォルト:** `true`
`com.fortify.sca.BuildID`	ビルドのビルドIDを指定します。 **値のタイプ:** 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.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。Fortify Static Code Analyzerで

プロパティ名	説明
	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。 **値のタイプ:** 文字列(パス) **デフォルト(Windows):** `${win32.LocalAppdata}\Fortify` **注:** `${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特殊な変数です。 **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\`

`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`
注: 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ファイル拡張子を変換する方法を指定します。有効なファイル拡張子の種類は、ABAP、ACTIONSCRIPT、APEX、APEX_TRIGGER、ARCHIVE、ASPNET、ASP、ASPX、BITCODE、BSP、BYTECODE、CFML、COBOL、CSHARP、DOCKERFILE、GENERIC、GO、HOCON、HTML、INI、JAVA、JAVA_PROPERTIES、JAVASCRIPT、JSP、JSPX、KOTLIN、MSIL、MXML、OBJECT、PHP、PLSQL、PYTHON、RUBY、RUBY_ERB、SCALA、SWIFT、SWC、SWF、TLD、SQL、TSQL、TYPESCRIPT、VB、VB6、VBSCRIPT、VISUAL_FORCE、XMLです。
	値のタイプ: 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`
	`com.fortify.sca.fileextensions.rb=RUBY`
	`com.fortify.sca.fileextensions.aspx=ASPNET`
	`com.fortify.sca.fileextensions.php=PHP`

プロパティ名	説明
	`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。指定したファイル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Fortify Static Code Analyzer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Fortify Static Code Analyzerの警告が書き込まれます。 **例:** `com.fortify.sca.fileextensions.jsp=oracle:<path_to_script>`

`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`
注: 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。
	値のタイプ: 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Fortify Static Code Analyzer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。
	• gcc/g++コマンド名では、Apple LLVM clang/clang++の実行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`

`com.fortify.sca.UseAntListener`	trueに設定すると、Fortify Static Code Analyzer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。 **値のタイプ:** 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

プロパティ名	説明
	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" ページ138を参照してください。 **注:** Fortify Static Code Analyzerでは、ビルド統合のない変換時にのみこのプロパティが使用されます。ほとんどのコンパイラまたはビルドツールと統合すると、このプロパティで指定されている場合でも、コンパイラまたはビルドツールで処理されるすべてのソースファイルがFortify Static Code Analyzerによって変換されます。ただし、Fortify Static Code AnalyzerのxcodebuildおよびMSBuild統合では`-exclude`オプションがサポートされています。 **値のタイプ:** 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`
`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" ページ134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。 **注:** 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。 **値のタイプ:** 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`
`com.fortify.sca.InputFileEncoding`	ソースファイルのエンコードタイプを指定します。Fortify Static Code Analyzer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、Fortify Static Code Analyzer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。Fortify Static Code Analyzer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、Fortify Static Code Analyzerではエンコーディングパラメータなしで`java.io.InputStreamReader`コンストラクタから`file.encoding`が使用されます。一部のケース(ActionScriptパーサの場合など)では、Fortify Static Code Analyzerのデフォ

プロパティ名	説明
	ルトはUTF-8です。 **値のタイプ:** 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
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Fortify Static Code Analyzer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。 **値のタイプ:** 文字列(パス) **デフォルト:** reg **例:** com.fortify.sca.RegExecutable= C:\\Windows\\System32\\reg.exe
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Fortify Static Code Analyzerタッチレスビルドが停止します。trueにすると、Fortify Static Code Analyzer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Fortify Static Code Analyzer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。 **値のタイプ:** ブール値 **デフォルト:** false
com.fortify.sca. Apex	trueに設定すると、Fortify Static Code Analyzerで、.cls拡張子を持つファイルに対してApex変換が使用され、.component拡張子を持つファイルに対してVisualforce変換が使用されます。 **値のタイプ:** ブール値 **デフォルト:** false **コマンドラインオプション:** -apex
com.fortify.sca. ApexObjectPath	カスタムsObject JSONファイルsobjects.jsonの絶対パスを指定します。

プロパティ名	説明
	値のタイプ: 文字列 **デフォルト:** (なし) **コマンドラインオプション:** `-apex-sobject-path`
`com.fortify.sca.AddImpliedMethods`	trueに設定すると、Fortify Static Code Analyzer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。 **値のタイプ:** ブール値 **デフォルト:** `true`
`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。 **値のタイプ:** 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.DeadCodeFilter`	trueに設定すると、Fortify Static Code Analyzerで、コンパイラによってデッドコードが生成されるなどのデッドコードの問題が削除され、ソースコードに表示されません。 **値のタイプ:** ブール値 **デフォルト:** `true`
`com.fortify.sca.DisableDeadCodeElimination`	デッドコードは、常にfalseと評価される`if`ステートメントの本体内のコードなどの、絶対に実行されないコードです。このプロパティをtrueに設定すると、Fortify Static Code Analyzerでデッドコードが特定されず、デッドコードの問題も報告されません。ただし、デッドコード内の他の脆弱性は、実行中に到達不可能であっても報告されます。 **値のタイプ:** 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。 **値のタイプ:** 文字列 **デフォルト:** (なし)
`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" ページ175を参照してください。

プロパティ名	説明
com.fortify.sca. hoa.Enable	trueに設定すると、高次分析が有効になります。 **値のタイプ:** 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。有効な値は、python、swift、ruby、javascript、およびtypescriptです。 **値のタイプ:** 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。 Fortifyでは、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を推奨します。Fortifyでは、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。 **値のタイプ:** 数値 **デフォルト:** 2700
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および2です。 **値のタイプ:** 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. 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汚染パスの長さを指定します。 **値のタイプ:** 整数 **デフォルト:** 4
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。 **値のタイプ:** 文字列 **デフォルト:** javascript,python,ruby,typescript
com.fortify.sca. TypeInferencePhase0Timeout	フェーズ0(手続き間分析)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。 **値のタイプ:** 倍精度 **デフォルト:** 300

プロパティ名	説明
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。 **値のタイプ:** 倍精度 **デフォルト:** 60
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。 **値のタイプ:** ブール値 **デフォルト:** false
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*<sca_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。 **値のタイプ:** 文字列 **デフォルト:** .xml
com.fortify.sca. RulesFile	カスタム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。 **値のタイプ:** 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. 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。Fortify Static Code Analyzer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。 **値のタイプ:** 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
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。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。 **値のタイプ:** 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

プロパティ名	説明
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。 **注:** 特徴付けソースルールは無効になりません。 **値のタイプ:** 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`
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。 **注:** 特徴付けシンクルールは無効になりません。 **値のタイプ:** 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`
com.fortify.sca. DefaultRulesDir	Fortify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。 **値のタイプ:** 文字列(パス) **デフォルト:** `${com.fortify.Core}/config/rules`
com.fortify.sca. CustomRulesDir	カスタムルールの検索時に使用するディレクトリを設定します。 **値のタイプ:** 文字列(パス) **デフォルト:** `${com.fortify.Core}/config/customrules`
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。 **値のタイプ:** 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。 **値のタイプ:** 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。 **値のタイプ:** 数値 **デフォルト:** `10`

プロパティ名	説明
com.fortify.sca.SuppressLowSeverity	trueに設定すると、スキャンで検出された重大度の低い問題がFortify Static Code Analyzerによって無視されます。 **値のタイプ:** ブール値 **デフォルト:** true
com.fortify.sca.LowSeverityCutoff	重大度の抑制のカットオフレベルを指定します。Fortify Static Code Analyzerでは、このプロパティに指定された値よりも重大度が低い問題を検出した場合に無視します。 **値のタイプ:** 数値 **デフォルト:** 1.0
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。 **値のタイプ:** ブール値 **デフォルト:** true
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" ページ184を参照してください。 **値のタイプ:** 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
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。 **値のタイプ:** 文字列 **デフォルト:** (なし)
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。 **値のタイプ:** 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
com.fortify.sca.QuickScanMode	trueに設定すると、Fortify Static Code Analyzerでクイックスキャンが実行されます。Fortify Static Code Analyzerでは、fortify-sca.properties設定ファイルの代わりにfortify-sca-quickscan.propertiesの設定が使用されます。 **値のタイプ:** ブール値 **デフォルト:** (無効) **コマンドラインオプション:** -quick

プロパティ名	説明
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Micro Focus Fortify Software Security Centerサーバ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。 **値のタイプ:** 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`
com.fortify.sca. ScanScaModule	trueに設定すると、Fortify Static Code Analyzerでこのプロジェクトのモジュール式スキャンが実行されます。これにより、後続のスキャンで`include-modules`オプション(または`com.fortify.sca.IncludeScaModules`プロパティ)とともにこのライブラリのビルドIDを使用できるようになります。 -scanコマンドラインオプションが指定されている場合、このプロパティは無視されます。 **値のタイプ:** ブール値 **デフォルト:** false **コマンドラインオプション:** `-scan-module`
com.fortify.sca. IncludeScaModules	プロジェクトスキャンで使用する個別のモジュールとして事前にスキャンされたライブラリのビルドIDのカンマまたはコロン区切りリストを指定します。各ビルドIDが既存のスキャンされたライブラリを示している必要があります。 **値のタイプ:** 文字列(ビルドID) **デフォルト:** (なし) **コマンドラインオプション:** `-include-modules` **例:** `com.fortify.sca.IncludeScaModules=LibA,LibB`
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。 **値のタイプ:** ブール値 **デフォルト:** `true`
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。 **値のタイプ:** 文字列 **デフォルト:** `.*yyparse.*`

プロパティ名	説明
com.fortify.sca. MultithreadedAnalysis	Fortify Static Code Analyzerを並行分析モードで実行するかどうかを指定します。 **値のタイプ**: ブール値 **デフォルト**: true
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。 **値のタイプ**: 整数 **デフォルト**: (使用可能なプロセッサコア数)
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、abap、actionscript、apex、cfml、cobol、configuration、cpp、dotnet、golang、java、javascript、jsp、kotlin、objc、php、plsql、python、ruby、scala、sql、swift、tsql、typescript、vbです。 **値のタイプ**: 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、abap、actionscript、apex、cfml、cobol、configuration、cpp、dotnet、golang、java、javascript、jsp、kotlin、objc、php、plsql、python、ruby、scala、sql、swift、tsql、typescript、vbです。 **値のタイプ**: 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
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。 **値のタイプ**: 文字列 **デフォルト**: 1.8 **コマンドラインオプション**: -jdk
com.fortify.sca. JavaClasspath	Java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。 **値のタイプ**: 文字列(パス)

プロパティ名	説明
	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.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。 **値のタイプ:** 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。 **値のタイプ:** 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`
`com.fortify.sca.AppserverVersion`	アプリケーションサーバのバージョンを指定します。 **値のタイプ:** 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。 **値のタイプ:** 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.JavaSourcepath`	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。 **値のタイプ:** 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.JavaSourcepathSearch`	trueに設定すると、Fortify Static Code Analyzerで、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。それ以外の場合、Fortify Static Code Analyzer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。 **値のタイプ:** ブール値

プロパティ名	説明
	デフォルト: `true`
`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。 **値のタイプ:** 文字列 **デフォルト:** (なし)
`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。 **値のタイプ:** 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。 **値のタイプ:** 文字列(エンコーディング) **デフォルト:** `ISO-8859-1`
`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。 **値のタイプ:** 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`

`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。
	値のタイプ: 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。 **値のタイプ:** 文字列 **デフォルト:** (なし)

プロパティ名	説明

	例:
	`com.fortify.sca.ASPVirtualRoots.Library=c:\\WebServer\\CustomerTwo\\Stuff`
	`com.fortify.sca.ASPVirtualRoots.Include=c:\\WebServer\\CustomerOne\\inc`

`com.fortify.sca.DisableASPExternalEntries`	trueに設定すると、分析のASP外部エントリが無効になります。 **値のタイプ:** ブール値 **デフォルト:** `false`
`com.fortify.sca.EnableDOMModeling`	trueに設定すると、Fortify Static Code Analyzer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。 **注:** 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。 **値のタイプ:** 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、Fortify Static Code Analyzer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。 **値のタイプ:** 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`
`com.fortify.sca.JavaScript.src.domain.whitelist`	Fortify Static Code Analyzer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。 **値のタイプ:** 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。

プロパティ名	説明
	値のタイプ: ブール値 **デフォルト**: `false`
`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。`com.fortify.sca.skip.libraries.jQuery`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`'(-\d\.\d\.\d)?'`が自動的に挿入されます。 **値のタイプ**: 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`
`com.fortify.sca.follow.imports`	trueに設定すると、importステートメントに含まれるファイルがJavaScript変換に含められます。 **値のタイプ**: ブール値 **デフォルト**: `true`
`com.fortify.sca.exclude.unimported.node.modules`	trueに設定すると、インポートされたnode_modulesのみがJavaScript変換に含められます。 **値のタイプ**: ブール値 **デフォルト**: `true`
`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。 **値のタイプ**: 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。 **値のタイプ**: 文字列

プロパティ名	説明
	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。 **値のタイプ:** 文字列 **デフォルト:** (GOPROXYシステム環境変数)
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、*Micro Focus Fortifyソフトウェアシステム要件*のドキュメントを参照してください。 **値のタイプ:** 文字列 **デフォルト:** `7.4` **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。 **値のタイプ:** 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`
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。Fortify Static Code Analyzer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。 **値のタイプ:** 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。 **値のタイプ:** 数値 **デフォルト:** `2` **コマンドラインオプション:** `-python-version`
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" ページ78を参照してください。 **値のタイプ:** 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`

プロパティ名	説明
com.fortify.sca. DjangoTemplateDirs	Djangoテンプレートのパス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。Fortify Static Code Analyzerでは、Djangoの`settings.py`ファイルのTEMPLATE_DIRS設定は使用されません。 **値のタイプ**: 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	Fortify Static Code AnalyzerでDjangoテンプレートを自動検出しないことを指定します。 **値のタイプ**: 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`
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。 **値のタイプ**: 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。 **値のタイプ**: 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`
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。 **値のタイプ**: 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。`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。 **値のタイプ**: 文字列(パス)

プロパティ名	説明
	デフォルト: (なし) **コマンドラインオプション:** `-flex-sdk-root`
`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。 **値のタイプ:** 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`
`com.fortify.sca.AbapDebug`	trueに設定すると、Fortify Static Code Analyzerでメッセージをデバッグするためにステートメントが追加されます。 **値のタイプ:** ブール値 **デフォルト:** (なし)
`com.fortify.sca.AbapIncludes`	Fortify Static Code Analyzer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。 **値のタイプ:** 文字列(パス) **デフォルト:** (なし)
`com.fortify.sca.CobolCopyDirs`	Fortify Static Code Analyzer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。 **値のタイプ:** 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。dialect値では、大文字と小文字を区別しません。 **値のタイプ:** 文字列 **デフォルト:** COBOL390 **コマンドラインオプション:** `-cobol-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。 **値のタイプ:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** `-checker-directives`
`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。 **値のタイプ:** ブール値

プロパティ名	説明
	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Fortify Static Code Analyzerに指示されます(レガシーCOBOL変換のみ)。 **値のタイプ:** 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)。 **値のタイプ:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** `-copy-extensions`
`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。 **値のタイプ:** 文字列 **デフォルト:** `TSQL` **コマンドラインオプション:** `-sql-language`
`com.fortify.sca.CfmlUndefinedVariablesAreTainted`	trueに設定すると、Fortify Static Code Analyzer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。 **値のタイプ:** 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。 **値のタイプ:** ブール値 **デフォルト:** (無効)

プロパティ名	説明
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。 **値のタイプ:** 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
com.fortify.sca. FVDLDisableDescriptions	trueに設定すると、分析結果ファイル(FVDL)からFortify Security Contentの説明が除外されます。 **値のタイプ:** 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. 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。 **値のタイプ:** 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. 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。 **値のタイプ:** 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. 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。 **値のタイプ:** 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. 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。 **値のタイプ:** ブール値 **デフォルト:** false
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。 **値のタイプ:** 文字列(パス) **デフォルト:** ${com.fortify.Core}/resources/sca/fvdl2html.xsl
com.fortify.sca. ResultsFile	結果が書き込まれるファイル。 **値のタイプ:** 文字列

プロパティ名	説明
	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.OutputAppend`	trueに設定すると、Fortify Static Code Analyzerで既存の結果ファイルに結果が追加されます。 **値のタイプ:** ブール値 **デフォルト:** `false` **コマンドラインオプション:** `-append`
`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ファイル拡張子に基づいて出力形式が選択されます。 **値のタイプ:** 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.ResultsAsAvailable`	trueに設定すると、Fortify Static Code Analyzer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。 **値のタイプ:** ブール値 **デフォルト:** `false`
`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。Fortify Static Code Analyzerでは、この名前は使用されませんが、結果に含められます。 **値のタイプ:** 文字列 **デフォルト:** (なし) **コマンドラインオプション:** `-build-project`
`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。Fortify Static Code Analyzer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。 **値のタイプ:** 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.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。Fortify Static Code Analyzer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。 **値のタイプ:** 文字列

プロパティ名	説明
	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Fortify Static Code Analyzer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。 **値のタイプ:** ブール値 **デフォルト:** (無効)
`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。エラーが発生した行の両側にある2行のコードは常に含められます。デフォルトでは、5行が表示されます。 **値のタイプ:** 数値 **デフォルト:** `2`
`com.fortify.sca.MobileBuildSessions`	trueに設定すると、Fortify Static Code Analyzerでソースファイルがビルドセッションディレクトリにコピーされます。 **値のタイプ:** ブール値 **デフォルト:** `false`
`com.fortify.sca.ExtractMobileInfo`	trueに設定すると、Fortify Static Code Analyzerでモバイルビルドセッションからビルド IDとFortify Static Code Analyzerのバージョン番号が抽出されます。 **注:** Fortify Static Code Analyzer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。 **値のタイプ:** ブール値 **デフォルト:** `false`
`com.fortify.sca.ClobberLogFile`	trueに設定すると、Fortify Static Code Analyzerで sourceanalyzerを実行するたびにログファイルが上書きされます。 **値のタイプ:** ブール値 **デフォルト:** `false` **コマンドラインオプション:** `-clobber-log`
`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。 **値のタイプ:** 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`

プロパティ名	説明
	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、"ログファイルの場所の確認" ページ181および"ログファイルの設定" ページ181を参照してください。 **値のタイプ:** 文字列 **デフォルト:** `INFO`
`com.fortify.sca.PrintPerformanceDataAfterScan`	trueに設定すると、Fortify Static Code Analyzerでスキャンの完了後にパフォーマンス関連のデータがFortify 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。 **値のタイプ:** ブール値 **デフォルト:** `false`
`com.fortify.sca.Debug`	Fortify Supportログファイルにデバッグ情報を含めます。この情報は、Micro Focus Fortifyカスタマサポートのトラブルシューティングにのみ役立ちます。 **値のタイプ:** 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。 **値のタイプ:** 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、Fortify Supportログファイルに詳細メッセージが含められます。 **値のタイプ:** ブール値 **デフォルト:** `false` **コマンドラインオプション:** `-verbose`
`com.fortify.sca.DebugTrackMem`	trueに設定すると、パフォーマンス情報をFortify Supportログファイルに書き込む追加デバッグが有効になります。 **値のタイプ:** 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。

プロパティ名	説明
	値のタイプ: 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。 **値のタイプ**: 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、Fortify Static Code Analyzerのメモリ使用量が監視され、JVMガーベ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。 **値のタイプ**: ブール値 **デフォルト**: true

fortify-sca-quickscan.properties

Fortify Static Code Analyzer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。

注: 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
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。 **値のタイプ**: 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。 **値のタイプ**: 文字列

プロパティ名	説明
	クイックスキャンのデフォルト: `controlflow:buffer` **デフォルト:** (なし)
`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が含まれている問題テンプレートを指定するには、「"fortify-sca.properties" ページ196」で説明した「`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、『*Micro Focus Fortify Audit Workbenchユーザガイド*』を参照してください。 **値のタイプ:** 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)
`com.fortify.sca.FPRDisableMetatable`	Micro Focus 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。 **値のタイプ:** ブール **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Fortify Static Code Analyzer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Fortify Software Security Center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。 **値のタイプ:** ブール **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`
`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

プロパティ名	説明
	ると、スキャン時間全体が短縮されます。 **値のタイプ:** 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。 **値のタイプ:** 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。 **値のタイプ:** 整数 **クイックスキャンのデフォルト:** `10000` **デフォルト:** `500000`
`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。 **注:** 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。 **値のタイプ:** 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。 **値のタイプ:** 整数 **クイックスキャンのデフォルト:** `5` **デフォルト:** `50`
`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。

プロパティ名	説明
	注: Fortifyでは、スキャン時間が長くなる可能性があるため、このプロパティを5より大きい値に設定することは推奨されていません。 **値のタイプ:** 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。 **値のタイプ:** 整数 **クイックスキャンのデフォルト:** 250 **デフォルト:** 1000
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。 **値のタイプ:** 整数 **クイックスキャンのデフォルト:** 500 **デフォルト:** 4000

付録E: Fortify Javaの注釈

Fortifyでは、2つのバージョンのJava Fortify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の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の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。

(たとえば、Fortify on Demandの評価で)Fortify製品を使用してアプリケーションのバイトコードを分析する場合は、注釈の保持ポリシーがCLASSに設定されているバージョンを使用します。Fortify製品を使用し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、Fortifyでは、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。

重要 Fortifyの注釈を運用コードに残すと、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、セキュリティ上のリスクになります。Fortifyでは、保持ポリシーがCLASSに設定されている注釈はFortify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、*<sca_install_dir>*`/Samples/advanced/javaAnnotations`ディレクトリにサンプルのFortify SCAとアプリケーションインストールとともに含まれています。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の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。

Fortify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" 次のページ
- "フィールドと変数の注釈" ページ228
- "その他の注釈" ページ229

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、『Micro Focus Fortifyカスタマサポート』を参照してください。

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、Fortify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`

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。

負以外の注釈とゼロ以外の注釈

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。

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() { ... } }
```

ドキュメントのフィードバックを送信する

このドキュメントに関するご意見は、電子メールでドキュメントチームまでお寄せください。

注: 弊社製品に関する技術的な問題が発生した場合は、ドキュメントチームに電子メールを送信しないでください。代わりに、Micro Focus Fortifyカスタマサポート (https://www.microfocus.com/support)にご連絡いただくと、サポートを受けることができます。

このコンピュータに電子メールクライアントが設定されている場合は、前のドキュメントチームに連絡するためのリンクをクリックすると、表題の行に以下の情報が付いた状態で電子メールウィンドウが開きます。

ユーザガイド(Fortify Static Code Analyzer 21.2.0)に関するフィードバック

電子メールにフィードバックを追加して、 [送信]をクリックします。

電子メールクライアントが使用できない場合は、前の情報をWebメールクライアントの新しいメッセージにコピーして、fortifydocteam@microfocus.comにフィードバックを送信してください。

皆様のご意見をお待ちしております。