Coverity：PCI DSS コンプライアンスのためのツール

通常の業務プロセスの一部としてセキュリティとコンプライアンスを確立

概要

ソフトウェア・セキュリティ・イニシアティブ（SSI）の最大の目的は情報アシュアランスにあり、セキュア・ソフトウェア開発ライフサイクル（SSDLC）を定義し、その実現に向けたソリューションを選択する際は、常にこの原則に従う必要があります。多くの場合、高いレベルの情報アシュアランスを達成することが、PCI DSS（Payment Card Industry Data Security Standard）などのスタンダードへのコンプライアンスを達成する近道となります。

PCI DSS は、小売店、金融機関、POS ベンダー、および決済処理インフラの開発と運用を手がけるハードウェア / ソフトウェア開発者などペイメント・カード処理に携わるすべての事業者に適用されるデータ・セキュリティ基準です。

PCI DSS には 12 のコンプライアンス要件があり、これらは以下の 6 つのグループに分類されています。

1. 安全なネットワークとシステムの構築と維持
2. カード会員データの保護
3. 脆弱性管理プログラムの維持
4. 強力なアクセス制御手法の導入
5. ネットワークの定期的な監視およびテスト
6. 情報セキュリティ・ポリシーの維持

PCI DSS へのコンプライアンス業務における Coverity の利点

静的解析ツールの Coverity は、品質上の重大な不具合や潜在的なセキュリティ脆弱性をソフトウェア開発ライフサイクル（SDLC）の早い段階で特定し、信頼できる具体的な修正ガイダンスを開発者に提示することにより、リスクの軽減と全体的なプロジェクト・コストの削減に貢献します。このため、PCI DSS へのコンプライアンスを求められる組織にとって、Coverity は理想的なソリューションとなります。

解析の深さと精度

PCI DSS では、アプリケーションに含まれていてはならない不具合の種類が定義されています。十分な不具合検出率を達成できないと、二次的なコンプライアンスの問題が生じることになるため、不具合は組織にとって非常に高いリスク領域となります。

Coverity は、幅広いセキュリティ脆弱性を検知できる強力な解析アルゴリズムと、その抜群の精度により、業界トップクラスの不具合検出率を達成しており、業界をリードする SAST ソリューションに認定されています（Forrester 社調査）。

重要な個人データの追跡

Coverity の洗練された重要データ漏洩チェッカーは、PCI DSS へのコンプライアンスにおいて特に重要な役割を果たします。このチェッカーにより、すべてのカード会員データおよび個人情報（医療情報を含む）を適切に処理できるようになります。

Coverity は、国民識別番号、カード会員データ、口座データ、トランザクション情報、医療情報、生体データ、地理データなど 25 種類の重要データを追跡します。この種の情報を適切に処理できていないことが情報漏洩の大きな要因であり、監査不合格の最も一般的な理由にもなっています。

問題の効率的な管理と修正

Coverity はコーディング中にその場で結果が得られ、正確かつ効率的な修正方法も具体的な修正アドバイスとして提示されるため、世界中の開発者が利用しています。しかも、Coverity は SDLC の早い段階に導入されるため、下流でのコストとリスクが大幅に軽減し、企業にとって時間と費用の節約につながります。

豊富なデータを柔軟なカスタマイズが可能なレポートとして生成

Coverity が保存する情報には、CWE や OWASP Top 10 などの各種スタンダードへのマッピングなどメタデータが豊富に含まれるため、個々のニーズに応じたレポートを簡単に自動生成できます。Coverity には、PCI DSS へのコンプライアンスの実証に必要なレポートを柔軟に生成する機能があります。

- Coverity のレポート生成パッケージは、PCI 認定審査機関（QSA）への提出用レポートを含め、一般的に要求されるレポートを PDF などいくつかのフォーマットで生成します。
- Coverity で生成されるデータはすべて、REST API を利用して CSV、XML、および JSON 形式で利用できるため、Coverity の実行結果を自由なフォーマットとレイアウトで表示できます。

以下のページでは、PCI DSS の要件を達成する上で Coverity が果たす役割について詳しく説明します。

スタンダードへのコンプライアンスと脆弱性検知を幅広くサポート

PCI DSS の要件	コンプライアンスへの対策
6.1：セキュリティ脆弱性情報の信頼できる社外提供元を使ってセキュリティの脆弱性を特定し、新たに発見されたセキュリティの脆弱性にリスクのランク（「高」、「中」、「低」など）を割り当てるプロセスを確立する。	Coverity により、すべての開発プロジェクトで継続的な脆弱性の検知 / 修正プロセスを確立できます。 各リスクには、業界ベスト・プラクティスや潜在的影響度を考慮してランクが割り当てられます。たとえば脆弱性のランク付けには、CVSS ベース・スコアやブラック・ダックによる分類が基準として使用されます。新しいセキュリティ脆弱性が見つかった場合、Coverity によってこれらのリスクは「高」、「中」、「低」に分類されるため、開発者は影響度 / リスクに基づいてただちに不具合にアクセスできます。
6.3：内部および外部ソフトウェアアプリケーション（アプリケーションへの Web ベースの管理アクセスを含む）を次のように開発する。 • PCI DSS（安全な認証やロギングなど）に従って。 • 業界基準やベスト・プラクティスに基づいて。 • ソフトウェア開発ライフサイクル全体に情報セキュリティを組み込む。	ソフトウェアが内部アプリケーション用か外部アプリケーション用かを問わず、Coverity は業界スタンダードおよびベスト・プラクティスに基づいてソフトウェア開発ライフサイクルにセキュリティを組み込めるように設計されています。また、Coverity で生成される強力な文書は、PCI DSS へのコンプライアンス業務に役立ちます。
6.4.3：テストまたは開発に本番環境データ（実際の PAN）を使用しない。	データ・ソース・タイプを CardHolderData として Coverity の SENSITIVE_DATA_LEAK チェッカーを実行し、開発、テスト、本番環境のソース・コードに PAN が含まれていないかどうかを調べます。PAN が見つかった場合は削除します。
6.4.4：テスト・データとテスト・アカウントは、システムがアクティブになる前、または本番稼働の前にシステム・コンポーネントから削除する。	この要件を満たすには、テスト・データとテスト・アカウントに対するコードを解析し、Coverity のコンポーネントを使用してこれらの情報をフィルタリングすることにより、システムを本番環境に移す前にこれらを削除します。
6.4.5.3：変更がシステムのセキュリティに悪影響を与えないことを確認するための機能テスト。	この要件を満たすには、Coverity を使用してコードのチェンジ・セットを検索し、セキュリティ脆弱性を見つけます。Coverity はコードの抽象構文木（AST）を理解できるため、より正確な解析が可能です。また、コードが変更された関数を、変更されていないコードから呼び出す「波及効果」も Coverity で解析できます。
6.5：ソフトウェア開発プロセスにおいて次のようにして一般的なコーディングの脆弱性に対応する。 • 開発者に対して一般的なコーディングの脆弱性を回避する方法を含む安全なコーディング技法について少なくとも年に一度トレーニングを実施する。 • 安全なコーディング・ガイドラインに基づいてアプリケーションを開発する。	コーディング脆弱性への対処方法を開発者が学ぶには、ある脆弱性がなぜ悪用可能と分類されたかを調べ、ソースからシンクまでの汚染データのフローを見るのが効果的です。Coverity では、こうした情報に加え詳細な修正アドバイスも得られるため、セキュア・コーディング・ガイドラインに基づいて不具合を修正できます。 別途 e ラーニング契約を結んでいただくと、コード中に見つかった CWE に関連するコースへのリンクが Coverity に表示されるため、使用しているプログラミング言語に関する最新のセキュア・コーディング・プラクティスを身につけることができます。

PCI DSS の要件	コンプライアンスへの対策
6.5.1：インジェクションの不具合（特に SQL インジェクション）。OS コマンドインジェクション、LDAP および Xpath のインジェクションの不具合、その他のインジェクションの不具合も考慮する。	Coverity には、次のように多くのインジェクション・チェッカーが含まれます。 ANGULAR_EXPRESSION_INJECTION DISTRUSTED_DATA_DESERIALIZATION EL_INJECTION FORMAT_STRING_INJECTION HEADER_INJECTION JAVA_CODE_INJECTION JCR_INJECTION JSP_DYNAMIC_INCLUDE JSP_SQL_INJECTION LDAP_INJECTION NOSQL_QUERY_INJECTION OGNL_INJECTION OS_CMD_INJECTION REGEX_INJECTION SCRIPT_CODE_INJECTION SQLI TAINTED_ENVIRONMENT_WITH_EXECUTION TEMPLATE_INJECTION UNKNOWN_LANGUAGE_INJECTION UNSAFE_DESERIALIZATION UNSAFE_JNI UNSAFE_REFLECTION XML_INJECTION XPATH_INJECTION
6.5.2：バッファ・オーバーフロー	Coverity には、次のように多くのバッファ・オーバーフロー・チェッカーがあります。 ARRAY_VS_SINGLETON BAD_SIZEOF BUFFER_SIZE BUFFER_SIZE_WARNING DC.STREAM_BUFFER DC.STRING_BUFFER INTEGER_OVERFLOW OVERLAPPING_COPY OVERRUN READLINK SIZECHECK STRING_NULL STRING_OVERFLOW STRING_SIZE
6.5.3：安全でない暗号化保存	この要件は、Coverity の以下のチェッカーで満たすことができます。 DC.WEAK_CRYPTO INSECURE_RANDOM INSECURE_SALT RISKY_CRYPTO UNRESTRICTED_ACCESS_TO_FILE

PCI DSS の要件	コンプライアンスへの対策
6.5.4：安全でない通信	この要件は、Coverity の以下のチェッカーで満たすことができます。 AUDIT.SPECULATIVE_EXECUTION_DATA_LEAK BAD_CERT_VERIFICATION CONFIG.CONNECTION_STRING_PASSWORD CONFIG.DYNAMIC_DATA_HTML_COMMENT CONFIG.MISSING_JSF2_SECURITY_CONSTRAINT CONFIG.SPRING_SECURITY_DEBUG_MODE CONFIG.STRUTS2_ENABLED_DEV_MODE CUSTOM_KEYBOARD_DATA_LEAK EXPOSED_PREFERENCES HARDCODED_CREDENTIALS INSECURE_COMMUNICATION INSECURE_MULTIPEER_CONNECTION JSP_DYNAMIC_INCLUDE MISSING_PERMISSION_FOR_BROADCAST MISSING_PERMISSION_ON_EXPORTED_COMPONENT PATH_MANIPULATION RISKY_CRYPTO SENSITIVE_DATA_LEAK TAINTED_SCALAR UNENCRYPTED_SENSITIVE_DATA URL_MANIPULATION USER_POINTER
6.5.5：不適切なエラー処理	この要件は、Coverity の以下のチェッカーで満たすことができます。 CONFIG.MISSING_GLOBAL_EXCEPTION_HANDLER MISSING_THROW UNCAUGHT_EXCEPT UNLOGGED_SECURITY_EXCEPTION
6.5.6：脆弱性特定プロセス（PCI DSS 要件 6.1 で定義）で特定された、すべての「高リスク」脆弱性	Coverity は、PCI DSS 要件 6.1 の定義に従って特定の脆弱性を「高影響度 / 高リスク」に分類します。
6.5.7：クロスサイト・スクリプティング（XSS）	この要件は、Coverity の以下のチェッカーで満たすことができます。 DOM_XSS XSS
6.5.8：不適切なアクセス制御（安全でないオブジェクトの直接参照、URL アクセス制限の失敗、ディレクトリ・トラバーサル、機能へのユーザアクセス制限の失敗など）	この要件は、Coverity の以下のチェッカーで満たすことができます。 CONFIG.DEAD_AUTHORIZATION_RULE CONFIG.STRUTS2_DYNAMIC_METHOD_I JSP_DYNAMIC_INCLUDE OPEN_REDIRECT PATH_MANIPULATION UNRESTRICTED_DISPATCH
6.5.9：クロスサイト・リクエスト・フォージェリ（CSRF）	この要件は、Coverity の以下のチェッカーで満たすことができます。 CONFIG.HANA_XS_PREVENT_XSRF_DISA CONFIG.SYMFONY_CSRF_PROTECTION_CSRF

PCI DSS の要件	コンプライアンスへの対策
6.5.10：不完全な認証管理とセッション管理	この要件は、Coverity の以下のチェッカーで満たすことができます。 CONFIG.MISSING_JSF2_SECURITY_CONS CONFIG.SPRING_SECURITY_DISABLE_AU CONFIG.SPRING_SECURITY_HARDCODED CONFIG.SPRING_SECURITY_REMEMBER_ CONFIG.SPRING_SECURITY_SESSION_FIX HARDCODED_CREDENTIALS JSP_DYNAMIC_INCLUDE MISSING_AUTHZ MOBILE_ID_MISUSE SESSION_FIXATION WEAK_BIOMETRIC_AUTH WEAK_GUARD WEAK_PASSWORD_HASH
6.6：一般公開されている Web アプリケーションで、継続的に新たな脅威や脆弱性に対処し、これらのアプリケーションが、次のいずれかの方法によって、既知の攻撃から保護されていることを確実にする。 ・一般公開されている Web アプリケーションは、アプリケーションのセキュリティ脆弱性を手動 / 自動で評価するツールまたは手法によって、少なくとも年 1 回および何らかの変更を加えた後にレビューする。 ・Web ベースの攻撃を検知および回避するために、一般公開されている Web アプリケーションの手前に、Web ベースの攻撃を自動的に検出・防止する技術的な解決策（Web アプリケーションファイアウォールなど）をインストールする。	Web ベースのアプリケーションは常に脅威にさらされており、いつでも攻撃を受ける可能性があります。このような攻撃がしばしば成功してしまうのは、セキュアでないコーディング・プラクティスに一因があります。したがって、攻撃を阻止するには、これらアプリケーションのレビューを定期的に実施する必要があります。 PCI DSS では 2 つの手法が推奨されていますが、導入と利用が容易なのは静的解析ツールを利用したコード・レビューです。これには 2 つの理由があります。1 つは、どのソフトウェア・プロジェクトにもレビュー可能なコードが存在すること、そしてもう 1 つは、高度なツールを使用すればコード・レビューを部分的に自動化できることです。 Web アプリケーションに対しては、コード・レビューを実施するだけでなく、インタラクティブ・アプリケーション・セキュリティ・テストや WAF など、Web ベースの攻撃を検知して防止する自動化されたテクニカル・ソリューションを使用することも重要です。

ブラック・ダックについて

ブラック・ダックは、業界で最も包括的かつ強力で信頼できるアプリケーション・セキュリティ・ソリューション・ポートフォリオを提供します。ブラック・ダックには、世界中の組織がソフトウェアを迅速に保護し、開発環境にセキュリティを効率的に統合し、新しいテクノロジーで安全に革新できるよう支援してきた比類なき実績があります。ソフトウェア・セキュリティのリーダー、専門家、イノベーターとして認められているブラック・ダックは、ソフトウェアの信頼を築くために必要な要素をすべて備えています。詳しくは www.blackduck.com/jp をご覧ください。

ブラック・ダック・ソフトウェア合同会社

www.blackduck.com/jp