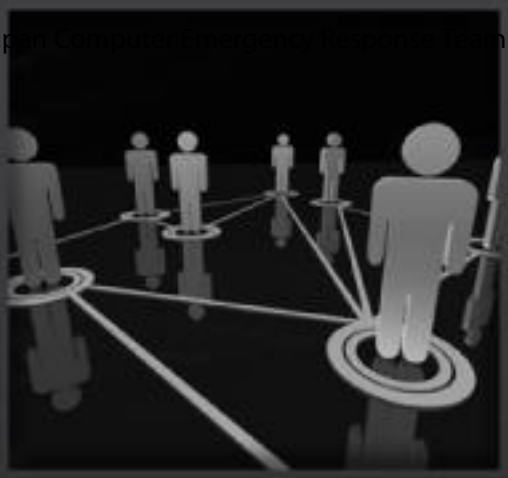


ソフトウエア セキュリティ保証 成熟度モデル

ソフトウエア開発にセキュリティを組み込むための手引き

第 1.0 版

本書の最新版及び追加情報については、Web サイト（http://www.opensamm.org）**を参照のこと**。

謝辞

ソフトウエアセキュリティ保証成熟度モデル（Software Assurance Maturity Model: SAMM）は、元々は独立ソフトウエアセキュリティコンサルタントである Pravir Chandra（chandra@owasp.org）により開発・設計・執筆された。最初の草案の作成は Fortify Software, Inc.の財政的支援を得て実現した。現在、本書の維持管理・改訂は OpenSAMM Project（代表 Pravir Chandra）が行っている。SAMM の公開当初から、同プロジェクトは Open Web Application Security Project（OWASP）に組み込まれた。巻末に示した各協賛組織にも感謝の意を表したい。

協力者・査読者一覧

本書の作成は、多数の査読者や専門家の支援と貴重な意見なくしては成し得なかった。ここに協力者の一覧を示す（アルファベット順、敬称略）。

Fabio Arciniegas	Brian Chess	Matteo Meucci	John Steven
Matt Bartoldus	Dinis Cruz	Jeff Payne	Chad Thunberg
Sebastien Deleersnyder	Justin Derry	Gunnar Peterson	Colin Watson
Jonathan Carter	Bart De Win	Jeff Piper	Jeff Williams
Darren Challey	James McGovern	Andy Steingruebl	

本プロジェクトは OWASP Project のプロジェクトである

OWASP

The Open Web Applecation Security Project

Open Web Application Security Project（OWASP）は、アプリケーションソフトウエアのセキュリティ向上に関する活動を展開する、フリーでオープンな世界規模のコミュニティである。OWASP の使命は、一般利用者や組織がアプリケーションのセキュリティリスクについての意思決定を行う際に十分な判断材料が得られるように、アプリケーションのセキュリティを「可視化」することである。OWASP には誰もが自由に参加でき、OWASP が作成する資料は、無償のオープンソフトウエアライセンスに基づいて公開される。OWASP Foundation は、米国国税庁（IRS）規定 501 条（c）項 3 号の認定を受けた非営利のボランティア組織として OpenSAMM Project の成果物を継続的に公開し、支援している。詳細については、OWASP の Web サイト（http://www.owasp.org）を参照のこと。

本書は経済産業省の委託事業として一般社団法人 JPCERT コーディネーションセンターが翻訳したものです。

ライセンス

エグゼクティブサマリ

ソフトウエア保証成熟度モデル（Software Assurance Maturity Model：SAMM）は、組織が直面する固有のリスクに応じたソフトウエアセキュリティ対策のための戦略の策定・実施を支援するフレームワークとして公開されている。SAMM が提供するリソースは、以下の作業に役立つ。

- 評価：組織の既存のソフトウエアセキュリティ対策の評価
- 構築：明確に定められた反復型作業による、バランスの取れたソフトウエアセキュリティ保証プログラムの構築
- 実証：セキュリティ保証プログラムに関する具体的な改善効果の実証
- 定義と測定：組織全体にわたるセキュリティ関連活動の定義と測定

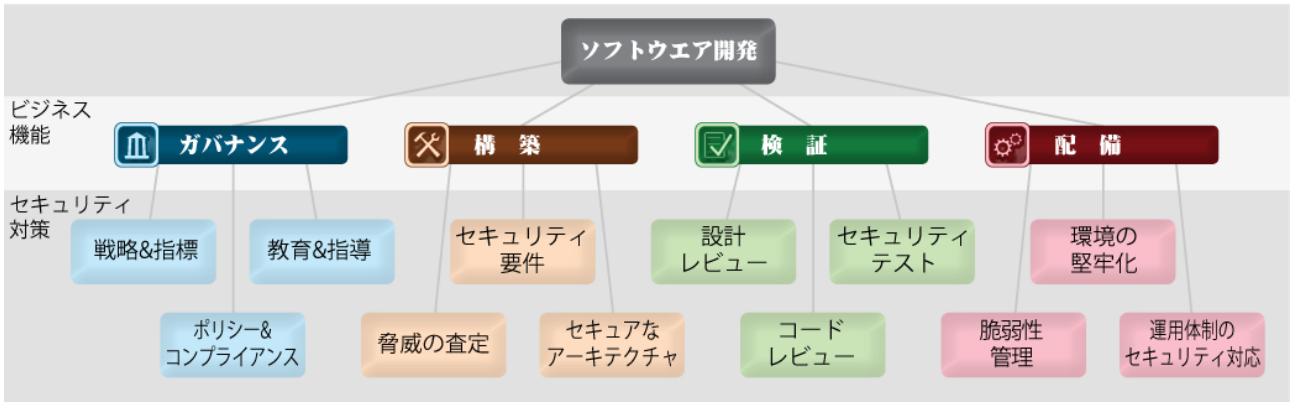
SAMM は、規模の大・中・小を問わずあらゆる組織のあらゆる開発スタイルにおいて活用できるように、柔軟性を持たせて定義された。また、組織全体だけでなく部門やプロジェクト単位の適用も可能である。さらに、SAMM は以下の原則の上に成り立っている。

- 組織の振る舞いは時間をかけてゆるやかに変化する ― ソフトウエアセキュリティプログラムを成功させるには、セキュリティ保証上の改善効果をもたらす短いサイクルの反復型作業と、長期的な目標に向けた段階的な取り組みとを組み合わせてプログラムを定めるものとする。
- すべての組織に効果のあるたった 1 つの秘策というものは存在しない ― ソフトウエアセキュリティのフレームワークには、組織それぞれのリスク耐性とソフトウエア構築・利用方法に応じた方法を選択できる柔軟性が必要である。
- セキュリティに関わる活動の指針は規範的なものであること ― セキュリティ保証プログラムの作成と査定に関する手順はすべて、簡潔かつ明確に定義され、結果を測定できるものでなくてはならない。SAMM は、一般的な組織向けのロードマップテンプレートも用意している。

このモデルの土台は、ソフトウエア開発の主要なビジネス機能とセキュリティ対策を相互に結び付けたものの上に成り立っている（下図）。モデルを構成する部品として、12 のセキュリティ対策それぞれに 3 段階の成熟度レベルが定義されている。これらは、セキュリティ上のリスク軽減とソフトウエアセキュリティ保証の向上のために組織が取り組むことのできる広範囲の活動を定義している。その他の詳細事項は、成功した取り組みの成果の測定、関連するセキュリティ保証上のメリットの理解、人員などのコストの見積もりのために盛り込まれている。

公開プロジェクトである SAMM のコンテンツは、常にベンダ中立であり誰もが自由に利用できる。

SAMMの全体像



目次

目的別目次

既存のソフトウエアセキュリティ保証対策を査定する

組織の戦略ロードマップを作成する

セキュリティ対策を実装または実施する

◆ *必読*

◇ *任意*

SAMM とは

ソフトウエアセキュリティ保証成熟度モデルの全容

SAMM は、ソフトウエア開発に関わる主要な「ビジネス機能」と結び付いた一連の「セキュリティ対策」の上に成り立っている。この節では、これらの「ビジネス機能」とそれぞれに対応する「セキュリティ対策」を紹介する。フレームワークの全体像を紹介した後は、個々の「セキュリティ対策」をどのようにすれば徐々に反復的に改善できるかをイメージしやすいように、「セキュリティ対策」ごとの「成熟度レベル」についても簡単に説明する。

ビジネス機能

SAMM の最上位には、もっとも概略的なレベルとして、必要不可欠な 4 つの「ビジネス機能」が定義されている。基本的なソフトウエア開発作業に関連する各種の活動は、これらの「ビジネス機能」（下記）のいずれかに分類される。言い換えれば、ソフトウエア開発に携わる組織はみな、これら「ビジネス機能」のそれぞれをある程度満たしている必要がある。

「ビジネス機能」ごとに、SAMM では 3 つの「セキュリティ対策」を定義している。個々の「セキュリティ対策」（右ページ）は、関連する「ビジネス機能」のセキュリティ保証を実現するセキュリティ関連活動の 1 分野である。それぞれ独立にセキュリティ改善にかかわる合計 12 の「セキュリティ対策」が、ソフトウエア開発の「ビジネス機能」の配下に展開されている。

「セキュリティ対策」ごとに、SAMM では 3 つの「成熟度レベル」を「目的」として定義している。ある「セキュリティ対策」における「成熟度レベル」はそれぞれ、一連の具体的な活動で定義される 1 つの「目的」によって特徴付けられる。「目的」の内容は段階的に洗練度を増すようになっており、上位レベルの目的を達成するには下位よりも厳しい基準を満たす必要がある。さらに、個々の「セキュリティ対策」における改善はそれぞれ独立に達成することもできるが、関連性のある活動を並行して実施することで最適化につなげることができる。

ガバナンス

ガバナンスは、組織におけるソフトウエア開発活動全体の管理に関するプロセスと活動を中心に据えている。より具体的には、開発に関与する各種グループに横断的に適用される問題や、組織全体レベルで確立されるビジネスプロセスが含まれる。

構 築

構築は、組織における目標設定の方法と開発プロジェクト内でのソフトウエア作成方法に関連したプロセスと活動に関するビジネス機能である。一般的には、製品管理、要件収集、概略レベルのアーキテクチャ仕様、詳細設計、実装が含まれる。

検 証

検証は、組織がソフトウエア開発を通して得た成果物をチェック、テストする方法に関するプロセスと活動に着目したビジネス機能である。テストなどの品質保証作業がこの典型であるが、その他のレビューや評価も含まれる。

配 備

配備は、作成したソフトウエアのリリースに関する組織としての管理方法に関連したプロセスと活動を意味するビジネス機能である。エンドユーザへの製品出荷、内部・外部ホストへの製品配備、実行時環境におけるソフトウエアの通常運用が含まれる。

ガバナンス

戦略&指標には、ソフトウエアセキュリティ保証プログラム全体の戦略的な方針と、セキュリティに対する組織の姿勢に関する指標を収集するプロセスと活動の計測が含まれる。

ポリシー&コンプライアンスには、開発中及び運用中のソフトウエアのセキュリティ保証を高めるための、セキュリティ及びコンプライアンスの管理、並びに組織全体を対象とする監査フレームワークの確立が含まれる。

教育&指導には、個別の職能に応じたセキュリティ題目の教育・指導による、ソフトウエア開発に携わる人員のセキュリティ知識の向上が含まれる。

脅威の査定には、組織のソフトウエアに対する潜在的な攻撃を正確に識別して特性を把握し、リスクに対する理解を深めてリスク管理を促進することが含まれる。

セキュリティ要件には、適切な機能を開発当初から仕様化するために、ソフトウエア開発工程にセキュリティ関連の要件を含めるよう促すことが含まれる。

セキュアなアーキテクチャには、設計プロセスを強化することが含まれる。その方法は、「デフォルト設定でセキュア」な設計を促進する活動と、ソフトウエア構築の基盤となる技術やフレームワークの管理である。

構築

設計レビューには、設計プロセスから得られる成果物を検査することによって、セキュリティの十分な仕組みが確実に装備され、セキュリティに対して組織が期待する水準が確実に満たされるようにすることが含まれる。

コードレビューには、組織のソースコードを査定することによる、脆弱性の発見及び関連するリスク緩和活動の支援と、セキュアなコーディングに対する期待値の基本水準の確立が含まれる。

セキュリティテストには、組織のソフトウエアを実行時環境内でテストすることによる脆弱性の発見と、ソフトウエアリリースに関する最低限の標準の確立が含まれる。

検証

脆弱性管理には、内部・外部の脆弱性レポートを管理する一貫したプロセスを確立することで、脅威にさらされる範囲を限定し、セキュリティ保証プログラム強化のためのデータを収集することが含まれる。

環境の堅牢化には、組織のソフトウエアを取り巻く運用環境の管理体制を整え、配備されたアプリケーションのセキュリティ状況を改善することが含まれる。

運用体制のセキュリティ対応には、オペレータに必要とされるセキュリティ関連情報を識別・捕捉し、組織のソフトウエアを適切に構成・配備・実行できるようにすることが含まれる。

配備

成熟度レベル

上記 12 種類の「セキュリティ対策」には、それぞれ 3 段階の「成熟度レベル」と、暗黙の開始点であるレベル 0 が定義されている。個々のレベルの詳細は「セキュリティ対策」の種類ごとに異なるが、大まかに言えばレベル分けは次のとおりである。

当該「セキュリティ対策」の活動が満たされていない状態を表す暗黙の開始点

当該「セキュリティ対策」に関する初歩的な理解とその場限りの対応

当該「セキュリティ対策」の効率や効果の向上

当該「セキュリティ対策」についての総合的な熟達

表記について

本書全体を通じて、次の括弧付きの用語は、この節で定義した SAMM の構成要素を指す。本文中でこれらの用語が括弧付きでなく用いられる場合は、文脈に応じて解釈されるものとする。

- 「ビジネス機能」または「機能」
- 「セキュリティ対策」または「対策」
- 「成熟度レベル」または「レベル」「目的」

ガバナンス

「セキュリティ対策」の説明

戦略&指標（Security and Metrics：SM）は、ソフトウエアセキュリティ保証プログラムのためのフレームワークを組織内に確立することに主眼を置いた「セキュリティ対策」である。測定可能なセキュリティ目標を、組織のビジネスにおける現実のリスクに対応した形で定義するうえで、これはもっとも基本的な手順である。

最初は簡易なリスクプロファイルの導入から着手し、その後でアプリケーションとデータ資産に関するリスク分類方法を徐々に高度化させていくとよい。相対的リスク指標についての見識が高まるに従って、プロジェクトレベルのセキュリティ目標をより組織に適したものに調整し、きめ細かなロードマップを設定してセキュリティプログラムの効率を高められるようになる。

この「セキュリティ対策」のより上位のレベルでは、組織は内部・外部のさまざまな情報源を利用して、セキュリティプログラムに関する指標と質的なフィードバックを収集する。そうすることで、実際に得られるメリットに対するコストをプログラムレベルで微調整できる。

ポリシー&コンプライアンス（Policy and Compliance：PC）は、法律・規制上の外的な要件を理解しつつ内部のセキュリティ標準を整備し、組織としてのビジネス上の目的に整合する形でのコンプライアンスを実現することに主眼を置いた「セキュリティ対策」である。

この「対策」を推進するにあたっては、プロジェクトレベルの監査によって組織の行動についての情報を収集し、期待される水準が満たされるかどうかをチェックすることが重要である。定形的な監査を、まず簡易な内容で導入し、徐々に高度化させていくことで組織に反復的な変化がもたらされる。

この「対策」の洗練度が向上すると、内部標準及び外的コンプライアンス推進要因の両方が組織全体に理解され、また、潜在化することの少ないチェックポイントがプロジェクトチームとの間に維持されるため、期待水準を満たさないプロジェクト運営の見落としがなくなる。

教育&指導（Education and Guidance：EG）は、ソフトウエアのライフサイクルに関与する人員に対し、セキュアなソフトウエアを設計・開発・配備するための知識及びリソースを提供することに主眼を置いた「セキュリティ対策」である。こうした情報へのアクセス性が向上すると、プロジェクトチームでは、当該組織の直面する具体的なセキュリティリスクを事前に識別・緩和しやすくなる。

3 段階の「目的」に共通する大きなテーマは、講師の指導によるセッションまたは PC を使ったモジュールによって、従業員のトレーニングを提供することである。段階が進むに従って、まず開発者向け、それから組織内のほかの職務向けと広範なトレーニングが蓄積されていく。最終的には、履修内容を完全に習得したことを示す職務別の証明書が発行される。

トレーニング以外にも、この「対策」においては、セキュリティ関連情報をまとめ、要員の拠りどころとなる指針を作成することも必要である。これにより、組織内のセキュリティ対策に対する期待値の基本水準を定めるための基盤を確立し、ひいては、指針採用後の段階的な改善を可能にすることができる。

ガバナンス

活動の概要

戦略&指標

	SM 1	SM 2	SM 3
目的	**組織内のソフトウエアセキュリティに関する統一的な戦略ロードマップを設定する**	**データ及びソフトウエア資産の相対的価値を測定し、リスク耐性を選択する**	**セキュリティ関連の支出を、適切なビジネス指標及び資産価値に整合させる**
実施内容	A. ビジネス全体のリスクプロファイルを見積もる B. セキュリティ保証プログラムのロードマップを設定し、維持管理する	A. ビジネス上のリスクに基づいてデータとアプリケーションを分類する B. 分類ごとのセキュリティ目標を設定し、測定する	A. 業界全体を対象としたコスト比較を定期的に実施する B. セキュリティ支出の推移に関する指標を収集する

ポリシー&コンプライアンス

	PC 1	PC 2	PC 3
目的	**ガバナンスとコンプライアンスについての当該組織に該当する推進要因を把握する**	**セキュリティとコンプライアンスの基本水準を設定し、プロジェクトごとのリスクを把握する**	**コンプライアンスを義務付け、組織全体のポリシーと標準に照らしてプロジェクトを測定する**
実施内容	A. 外的なコンプライアンス推進要因を特定する B. コンプライアンスの指針を設定し、維持管理する	A. セキュリティ及びコンプライアンスのポリシーと標準を策定する B. プロジェクト監査の方法を確立する	A. プロジェクトのためのコンプライアンスゲートを作成する B. 監査データ収集の方策を採択する

教育&指導

	EG 1	EG 2	EG 3
目的	**開発要員に、セキュアなプログラミングと配備の話題を扱っている参考情報へのアクセスを提供する**	**ソフトウエアのライフサイクルに関与するすべての人員に対し、セキュアな開発について職務に特化した指導内容の教育を実施する**	**総合的なセキュリティ教育を義務付け、基本水準を満たす知識を備えた人員を認定する**
実施内容	A. 技術的なセキュリティ意識向上トレーニングを実施する B. 技術的な指針を設定し、維持管理する	A. アプリケーションセキュリティについて職務に特化したトレーニングを実施する B. セキュリティコーチを利用してプロジェクトチームを強化する	A. アプリケーションセキュリティをサポートする公式ポータルを作成する B. 職務に応じた試験や認定制度を確立する

構 築

「セキュリティ対策」の説明

脅威の査定

脅威の査定（Threat Assessment：TA）は、開発対象ソフトウエアの機能と実行時環境の特性に基づいてプロジェクトレベルのリスクを特定･把握することに主眼を置いた「セキュリティ対策」である。個々のプロジェクトが直面する脅威や想定される攻撃について詳細な情報が得られると、組織は全体として、セキュリティに関する意思決定や方策の優先順位付けをより効果的に行うことができる。また、リスク受容に関する意思決定のための判断材料が充実するため、よりよくビジネスと整合した決定が可能になる。

最初は単純な脅威モデルの導入から着手し、脅威の分析・重み付け手法を徐々に高度化させるようにすると、時間の経過とともに組織としての対応を向上させていくことができる。洗練度を究極的に高めるには、たとえば、リスクの相殺要因や外部から持ち込まれるリスクと密接に対応する形でこの情報を維持管理することが考えられる。このようにすると、下流においてセキュリティ問題が及ぼしうる影響の内容をより広範に認識しつつ、当該組織が既知の脅威に対応する能力の現状を常に詳しく監視できる。

セキュリティ要件

セキュリティ要件（Security Requirements：SR）は、セキュリティに関してソフトウエアに期待される動作を事前予防的に定めることに主眼を置いた「セキュリティ対策」である。プロジェクトレベルで分析活動を導入することにより、ソフトウエアの大局的なビジネス目的に基づいてセキュリティ要件を初期段階に収集できるようになる。段階が進むとテクニックが高度化し、たとえば、開発当初には明らかでない新しいセキュリティ要件を発見するためにアクセス制御の仕様を定めておくといった手法が使用される。

この「対策」の洗練度が向上すると、組織のセキュリティ要件が供給元業者との関係にまで波及し、その上で、セキュリティ要件の仕様に期待されるすべての要素を満たすようなプロジェクト監査が行われる。

セキュアなアーキテクチャ

セキュアなアーキテクチャ（Secure Architecture：SA）は、「デフォルト設定でセキュア」なソフトウエアを設計・構築するために組織が実施する事前予防的な作業に主眼を置いた「セキュリティ対策」である。再利用可能なサービスやコンポーネントを使ってソフトウエア設計プロセスを強化すると、ソフトウエア開発に由来するセキュリティリスクを全体として劇的に減らすことができる。

最初はソフトウエアのフレームワークに関する単純な推奨事項と、セキュアな設計原則に関する明示的な考慮事項を設定することから着手し、セキュリティ機能のためのデザインパターンを一貫して使用する体制を目指して変革を進めていく。また、集中化したセキュリティサービス及びインフラストラクチャの利用増をプロジェクトチームに推奨する活動も実施する。

この「対策」の段階が進むことは、当該組織で開発するソフトウエアの基になる各種の原型を網羅したリファレンスプラットフォームを構築することを意味する。それらの原型は、カスタムソフトウエア開発時の脆弱性リスクを低減するためのフレームワークとして機能する。

構 築

活動の概要

脅威の査定

	TA 1	TA 2	TA 3
目的	**組織及び個々のプロジェクトにとっての脅威を概要レベルで特定し、理解する**	**脅威の査定の精度を高め、プロジェクト別のきめ細かな理解を進める**	**内製及びサードパーティ製ソフトウエアに存在する個々の脅威に対し、相殺管理策を的確に対応付ける**
実施内容	A. アプリケーションごとに特化した脅威モデルを設定し、維持管理する B. ソフトウエアキテクチャに基づく攻撃者プロファイルを作成する	A. プロジェクトごとの悪用ケースモデルを設定し、維持管理する B. 脅威測定のための重み付けシステムを採用する	A. サードパーティ製コンポーネントからもたらされるリスクを明示的に評価する B. 相殺管理策を盛り込んで脅威モデルの内容を洗練させる

セキュリティ要件

	SR 1	SR 2	SR 3
目的	**ソフトウエア要件の設定作業中に明示的なセキュリティ検討の機会を設ける**	**ビジネスロジックと既知のリスクに基づくセキュリティ要件をより緻密に設定する**	**すべてのソフトウエアプロジェクト及び依存対象サードパーティ要素に対し、セキュリティ要件のプロセス遵守を義務付ける**
実施内容	A. ビジネス機能に基づくセキュリティ要件を明確化する B. セキュリティとコンプライアンスのベストプラクティスを評価し、用件として採用する	A. リソースと機能に関するアクセス制御マトリックスを作成する B. 既知のリスクに基づくセキュリティ要件を策定する	A. サプライヤとの契約にセキュリティ要件を盛り込む B. セキュリティ要件の監査プログラムを拡張する

セキュアなアーキテクチャ

	SA 1	SA 2	SA 3
目的	**事前予防的なセキュリティ指導についての検討機会をソフトウエア設計プロセスに含める**	**既知のセキュアサービスと「デフォルト設定でセキュア」な設計を採用する方向にソフトウエア設計プロセスを誘導する**	**ソフトウエア設計プロセスを公式に管理し、セキュアなコンポーネントの利用について認可制度を実施する**
実施内容	A. 推奨ソフトウエアフレームワークのリストを維持管理する B. セキュリティに関する原則を設計に対して明示的に適用する	A. セキュリティに関するサービスやインフラストラクチャを具体的に指定し、使用を奨励する B. セキュリティに関するデザインパターンをアーキテクチャに基づいて指定する	A. 公式なリファレンスアーキテクチャとリファレンスプラットフォームを確立する B. フレームワーク、パターン、プラットフォームの使用について認可制度を実施する

検 証

「セキュリティ対策」の説明

設計レビュー（Design Review：DR）は、セキュリティ関連の問題を発見するためにソフトウエアの設計とアーキテクチャを査定することに主眼を置いた「セキュリティ対策」である。これにより、アーキテクチャレベルの問題をソフトウエア開発の早い段階で発見でき、後でセキュリティ問題が発覚してリファクタリングのために大きなコストが発生するのを回避できる。

最初は、アーキテクチャにおけるセキュリティ関連の詳細事項についての理解を醸成するための簡易な活動から着手し、その後セキュリティ対策が完備されているかを検証する公式の検査手法を整備していく。設計レビューのためのサービスは組織全体のレベルに構築され、利害関係者に提供される。

この「対策」の洗練度が向上すると、詳細かつデータレベルの設計検査が行われ、また、リリース承認前の設計査定及び発見内容のレビューにより、期待されるセキュリティの基本水準が確実に保たれる。

コードレビュー（Code Review：CR）は、セキュリティ脆弱性を発見するためにソフトウエアをソースコードレベルで検査することに主眼を置いた「セキュリティ対策」である。コードレベルの脆弱性は概念的にはわかりやすいものが多いが、知識の豊富な開発者でさえ、ソフトウエアにセキュリティ侵害の可能性を残してしまうミスを犯すことはよくある。

最初は効率を重視して簡易なチェックリストを使用し、非常に重要なソフトウエアモジュールのみを検査対象にする。後の段階では、自動化技術を使用することにより、検査の対象範囲及びコードレビュー活動の有効性を劇的に拡大する。

この「対策」の洗練度が向上すると、コードレビュー活動が開発プロセスの中に緊密に組み込まれ、プロジェクトチームがより早期に問題を発見できるようになる。また、組織として行うコードレビューの質が向上し、ソフトウエアリリース前のコードレビューによる発見内容について期待水準をより的確に設定できるようになる。

セキュリティテスト（Security Testing：ST）は、セキュリティ問題を発見するためにソフトウエアを実行時環境内で検査することに主眼を置いた「セキュリティ対策」である。こうしたテスト活動は、実行時に想定される状況下でチェックすることによりソフトウエアのセキュリティ保証ケースを補強するものであり、実際の運用状況でないと発覚しにくい運用上の構成ミスやビジネスロジック上のミスを明らかにすることができる。

最初はソフトウエアの機能に基づく侵入テストや高レベルテストケースの導入から着手し、段階が進むに従って、システムの脆弱性を実証するさまざまなテストケースを網羅した自動セキュリティテストを採用していく。

この「対策」の洗練度が向上すると、自動テストがカスタマイズされ、アプリケーションに特化した懸念事項を細かく網羅したセキュリティテスト群が構築される。組織レベルの可視性が向上するため、プロジェクトリリース承認前のセキュリティテスト結果について最低限の期待水準を設定できるようになる。

検 証

活動の概要

設計レビュー

			
目的	**ソフトウエア設計に対する当座のレビューをサポートし、既知のリスクに対して基本水準の緩和策を確実に適用する**	**総合的なセキュリティベストプラクティスに照らした、ソフトウエア設計レビューのための査定サービスを提供する**	**査定を義務付け成果物の認可制度を実施して、保護機構についての詳しい理解を広める**
実施内容	A. ソフトウエアが攻撃にさらされる箇所を特定する B. 既知のセキュリティ要件に照らして設計を分析する	A. セキュリティの仕組みが完備されているかどうかを検査する B. プロジェクトチーム向け設計レビューサービスを配備する	A. 扱いに注意を要するリソースについてのデータフロー図を作成する B. 設計レビューを行うためのリリースゲートを設定する

コードレビュー

			
目的	**取り組みやすい範囲で、基本的なコードレベルの脆弱性やその他のセキュリティ問題を検出する**	**開発中に行うコードレビューの精度と効率が自動化によって向上する**	**総合的なコードレビュー作業を義務付け、言語レベルのリスクとアプリケーションに特化したリスクを検出する**
実施内容	A. 既知のセキュリティ要件に基づいてレビューチェックリストを作成する B. リスクの高いコードに対して重点レビューを実施する	A. 自動コード分析ツールを利用する B. コード分析作業を開発プロセスに組み込む	A. アプリケーションに特化した問題に対応するようコード分析をカスタマイズする B. コードレビューを行うためのリリースゲートを設定する

セキュリティテスト

			
目的	**実装とソフトウエアの要件に基づく基本的なセキュリティテストを実行するためのプロセスを確立する**	**開発中に行うセキュリティテストの完全性と効率が自動化によって向上する**	**配備前に基本水準のセキュリティを確保するため、アプリケーション固有のセキュリティを要求する**
実施内容	A. 既知のセキュリティ要件に基づいてテストケースを作成する B. ソフトウエアリリースに対する侵入テストを実施する	A. 自動セキュリティテストツールを利用する B. セキュリティテスト作業を開発プロセスに組み込む	A. アプリケーションに特化したセキュリティテスト自動化手法を採用する B. セキュリティテストを行うためのリリースゲートを設定する

配 備

「セキュリティ対策」の説明

脆弱性管理

脆弱性管理（Vulnerability Management：VM）は、組織内における脆弱性レポートや運用中インシデントの処理方法に関するプロセスに主眼を置いた「セキュリティ対策」である。それらのプロセスを導入すると、こうした事象の取扱いに関して一定の期待水準を維持し、効率が上がり、不十分な情報に基づいてむやみな対応をとる必要がなくなる。

最初はインシデント発生時の役割分担を簡易的に決めることから着手し、段階が進むに従って、発生する問題についての可視化と追跡管理を確実にする公式なインシデント対応プロセスを整備していく。また、情報伝達を改善することでプロセスの全体的な理解を向上させる。

この「対策」の洗練度が向上すると、インシデント切り分けの徹底と脆弱性レポートにより、詳細な評価指標と根本原因に関するその他の情報を収集し、以後の組織としての対応に反映できるようになる。

環境の堅牢化

環境の堅牢化（Environment Hardening：EH）は、組織のソフトウエアをホストする実行時環境についてのセキュリティ保証を実現することに主眼を置いた「セキュリティ対策」である。外部コンポーネントに問題があるとアプリケーション運用のセキュリティは低下する。運用の基礎となるインフラストラクチャも一種の外部コンポーネントであり、これを堅牢化することには、ソフトウエアの全体的なセキュリティ状況を直接的に改善する効果がある。

最初は運用環境の情報を簡単な方法で追跡管理及び周知することにより開発チームへの情報発信を改善することから着手し、段階が進むに従って、運用環境に対するセキュリティパッチの配備管理や、被害が生じる前に潜在的なセキュリティ問題の警告を発する早期検出の仕組みを運用環境に組み込むための、大規模対応が可能な方法を採用していく。

この対策の洗練度が向上すると、脆弱性が悪用された場合も被害が小規模で済むように、運用環境の見直しが行われ、防御策とセーフティネットの層を増強する保護ツールを配備することで堅牢化がさらに徹底される。

運用体制のセキュリティ対応

運用体制のセキュリティ対応（Operation Enablement：OE）は、ソフトウエア構築プロジェクトチームからセキュリティ上重要な情報を収集し、その情報をソフトウエアのユーザやオペレータに発信することに主眼を置いた「セキュリティ対策」である。この情報がないと、たとえソフトウエアの設計がきわめてセキュアであっても、セキュリティ上の重要な特性や選択肢が配備先サイトに知らされず不必要なリスクが生じることになる。

最初はユーザやオペレータにもっとも重大な影響を及ぼす詳細事項のみを記載した簡易な文書を作成することから着手し、段階が進むに従って、完成された運用セキュリティガイドをリリースごとに提供する体制を整備していく。

この「対策」の洗練度が向上すると、個々のプロジェクトチームに対する組織レベルでのチェックが行われ、各種の期待水準に従って情報の収集・共有が確実に行われるようになる。

配 備

活動の概要

脆弱性管理

目的	**脆弱性レポートまたはインシデントへの対応に関する高レベル計画を理解する**	**対応プロセスに期待される内容を細かく検討し、一貫性と情報発信の方法を改善する**	**対応プロセスにおける分析とデータ収集の方法を改善し、事前予防の計画に反映する**
実施内容	A. セキュリティ問題の連絡窓口を決定する B. 非公式のセキュリティ対応チームを編成する	A. 一貫したインシデント対応プロセスを確立する B. セキュリティ問題の情報公開プロセスを採択する	A. インシデントの根本原因分析を実施する B. インシデントごとに指標を収集する

環境の堅牢化

目的	**アプリケーションとソフトウエアコンポーネントに関する運用環境の基本水準を理解する**	**運用環境の堅牢化によってアプリケーション運用の信頼度を向上させる**	**既知のベストプラクティスに照らしてアプリケーションの状況及び状態の妥当性を確認する**
実施内容	A. 運用環境に関する仕様を維持管理する B. 重要なセキュリティアップグレード及びパッチを特定し、インストールする	A. 定形的なパッチ管理プロセスを確立する B. 環境の基本水準に関する構成状態を監視する	A. 適切な運用保護ツールを指定し、配備する B. 環境構成の監査プログラムを展開する

運用体制のセキュリティ対応

目的	**重要なセキュリティ関連データについて、開発チームとオペレータとの情報交換を可能にする**	**詳細な手続きを整備し、セキュアな運用の継続性に関する期待水準を引き上げる**	**セキュリティ情報の発信を義務付け、作成資料の完成度を確認する**
実施内容	A. 配備のために重要なセキュリティ情報を収集する B. 一般的なアプリケーション警告への対応手順を文書化する	A. リリースごとの変更管理手順を策定する B. 公式の運用セキュリティガイドを維持管理する	A. 運用情報の監査プログラムを展開する B. アプリケーションコンポーネントのコード署名を実行する

SAMM の適用

このモデルを実際に使うには

この節では、重要かつ有用な SAMM の応用方法をいくつか示す。このモデルの基本設計は、組織のセキュリティ保証プログラムを測定してスコアカードを作成するためのベンチマークとして使用できるようになっている。スコアカードは、セキュリティ保証プログラム策定の反復作業を通して得られる改善効果を実証するのに有効である。また、もっとも重要な点は、組織がセキュリティ保証の取り組みにおける増強や改良の方針を設定する目安として、SAMM ロードマップテンプレートを使用できることである。

成熟度レベルの利用法

12 の「セキュリティ対策」には、それぞれ 3 段階の「成熟度レベル」が定められている。各レベルは、当該レベルを理解・達成する上で不可欠の要因を規定する複数の要素から成り立っている。さらに、規範的な詳細情報が示されているため、SAMM を使ったソフトウエア保証プログラム構築という文脈以外で「セキュリティ対策」の定義を使用することもできる。

目的

「目的」は、当該の「レベル」を達成するセキュリティ保証上の目標を一般的な文で表現したものである。当該の「対策」における「レベル」が上がるに従って、「目的」には、ソフトウエア開発・配備のセキュリティ保証実現に関する、より洗練度の高い目標が述べられるようになる。

実施内容

実施内容とは、当該「レベル」の達成に必要な主要な要件である。これには組織全体での実行を想定したものもあれば、個々のプロジェクトチームにおける作業に対応するものもある。いずれにしても、実施内容は主要なセキュリティ機能を表現しているのであり、実施内容をどのように満たすかは組織としての判断に委ねられる。

成果

成果とは、当該「レベル」の達成によって得られる能力や成果物を表すものである。これには具体的なものもあれば、能力向上に関する質的な内容を示すものもある。

成功指標

成功指標とは、当該「レベル」において組織が実行している内容のチェックに使用できる尺度の例である。データ収集・管理の方法は各組織の判断によるが、推奨される情報源と基準値が示されている。

コスト

コストとは、当該「レベル」を達成する際に組織に発生する支出についての質的な説明である。具体的な額は組織によって異なるが、当該「レベル」の運用にまつわる一時コスト及び継続的なコストについて目安が示されている。

人員

特定「レベル」におけるこれらの特性は、当該「レベル」における運用のための人的リソースに関する継続的なオーバヘッドの見積もりを示している。

- 開発者 — ソフトウエアの詳細設計及び実装を行う人員
- アーキテクト — 概要設計作業及び大規模システムエンジニアリングを行う人員
- マネージャー — 開発要員の日常的な管理を行う人員
- QA テスト担当者 — ソフトウエアの品質保証テスト及びリリース前検証を行う人員
- セキュリティ監査員 — 作成するソフトウエアに関連した技術的なセキュリティ知識を持つ人員
- ビジネスオーナー — ソフトウエア及びそのビジネス上の要件について最終的な意思決定を行う人員
- サポート担当者 — 顧客サポートまたは直接の技術的な運用サポートを行う人員

関連レベル

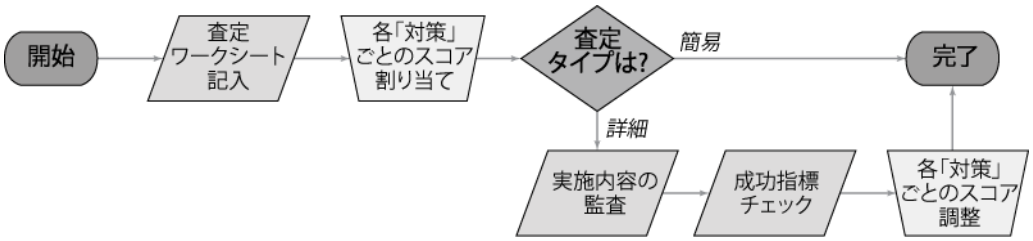
関連レベルとは、組織の構造やセキュリティ保証プログラム構築の進捗によってある程度重複する可能性があると考えられる、ほかの「対策」の「レベル」である。機能的には、その関連レベルが同時に目標として掲げられている場合やすでに施行されている場合、実施内容の履行において相乗効果または最適化効果が生じることを示す。

査定の実施

定義されている各種「セキュリティ対策」に照らして組織の状況を測定すると、組織に組み込まれているセキュリティ保証活動の全体像が描き出される。この種の査定は、組織でどの程度幅広いセキュリティ活動が行われているかの現状を把握するために有用である。また、当該組織で SAMM を利用し、今後反復的な向上を続けていくためのロードマップを作成することも、この査定によって可能になる。

査定は、単純に組織を評価し、現状行われている行動の成熟度レベルを判定することにより実施される。組織のパフォーマンスをどの程度までチェックするかは、その査定の実施を促した背景要因によって異なるが、一般論としては次の 2 つのタイプが推奨される。

- 簡易 — 各「対策」の査定ワークシートを評価し、回答に基づいてスコアを割り当てる。組織が現状のセキュリティ保証プログラムを SAMM に対応付け、状況の大まかな全体像を把握することが目的の場合、この種の査定を実施すれば普通は十分である。
- 詳細 — 査定ワークシートを記入した後、追加の監査作業を実行し、「対策」ごとに示されている規範的な実施内容が当該組織で履行されていることを確認する。さらに、各「対策」に示されている成功指標のデータを収集し、組織のパフォーマンスが期待水準を満たしていることも確認すべきである。



査定ワークシートを使用して組織のスコアを評価する方法は明快である。設問に回答してから、回答列を評価して「レベル」を判定すればよい。「レベル」は回答列の右にあるマーカで示されている。あるマーカより上にある設問の回答がすべて「はい」であれば当該「レベル」を満たしたことになる。

既存のセキュリティ保証プログラムを構成する活動内容は、いずれかの「成熟度レベル」に完全に当てはまるとは限らない。たとえば、ある「対策」についての査定が「レベル 1」を満たし、さらに、ほかの内容の活動が行われているものの査定が「レベル 2」には満たないといった可能性がある。そのような場合は組織のスコアに「+」記号を付記することで、判定された「レベル」のほかにもセキュリティ保証の活動が行われていることを示す。たとえば、ある組織が運用体制のセキュリティ対応について「レベル 1」の実施内容をすべて満たすのに加え、「レベル 2」または「レベル 3」に該当する実施内容を 1 つ実施している場合は、「1+」というスコアが割り当てられる。同様に、ある「セキュリティ対策」の実施内容をすべて満たすのに加えて SAMM に規定されていない活動も実施している場合のスコアは「3+」となる。

ガバナンス

査定ワークシート

戦略&指標

	Yes/No
すでに施行しているソフトウエアセキュリティ保証プログラムがある	
ほとんどのビジネス利害関係者が組織のリスクプロファイルを理解している	
ほとんどの開発要員がセキュリティ保証プログラムの将来計画を認識している	
ほとんどのアプリケーションとリソースがリスクに基づいて分類される	SM 1
必要なセキュリティ保証活動をカスタマイズするためにリスク評価が使われる	
リスク評価に基づいて、必要な事項を組織の人員のほとんどが理解している	
セキュリティ保証活動のコストに関するプロジェクトごとのデータが収集される	SM 2
組織としてのセキュリティ支出が、ほかの組織と随時比較される	SM 3

ポリシー&コンプライアンス

	Yes/No
ほとんどのプロジェクト利害関係者が、関係するプロジェクトのコンプライアンス状況を知っている	
コンプライアンス要件がプロジェクトチームで明示的に検討される	
ソフトウエア開発の管理に、組織としてポリシーと標準を利用している	PC 1
ポリシーと標準に対するコンプライアンスの監査をプロジェクトチームが要求できる	
プロジェクトの監査が定期的に実施され、ポリシーと標準に対するコンプライアンスの基本水準が確認される	PC 2
組織として、監査を体系的に使用することでコンプライアンスの証跡を収集・管理している	PC 3

教育&指導

	Yes/No
ほとんどの開発者が概略的なセキュリティ意識向上トレーニングを受講済みである	
セキュアな開発に関するベストプラクティスや手引書が、すべてのプロジェクトチームで利用できるようになっている	
開発プロセスにかかわるほとんどの職務に対し、職務に特化したトレーニングや指導が実施される	EG 1
ほとんどの利害関係者が、プロジェクトのためにセキュリティコーチを招くことができる	
セキュリティ関連の手引書が一元管理され、組織全体に一貫した方法で配布される	EG 2
ほとんどの人員について、セキュアな開発の実践に関する基本水準の技能を持つことがテストにより確認される	EG 3

構 築

査定ワークシート

脅威の査定

	Yes/No	
◆ 組織内のほとんどのプロジェクトで、直面する可能性が大きい脅威についての検討と文書化が行われる		
◆ 直面している攻撃の種類が組織として理解され、文書化される		TA 1
◆ プロジェクトチームが、悪用の可能性がないか機能要件を随時分析する		
◆ プロジェクトチームが、何らかの脅威評価手法を使用して相対的な比較を行う		
◆ 利害関係者が、該当する脅威及び評価について認識している		TA 2
◆ 外部ソフトウエアからもたらされるリスクがプロジェクトチームで明示的に検討される		
◆ あらゆる保護機構や保護策の情報が収集され、脅威への対応付けが行われる		TA 3

セキュリティ要件

	Yes/No	
◆ ほとんどのプロジェクトチームが開発時に何らかのセキュリティ要件を策定する		
◆ プロジェクトチームが、ベストプラクティス及びコンプライアンス手引書に基づく要件を抽出する		SR 1
◆ ほとんどの利害関係者が、関係するプロジェクトについてアクセス制御マトリックスを確認する		
◆ プロジェクトチームが、ほかのセキュリティ活動から得られたフィードバックに基づいて要件を策定する		SR 2
◆ ほとんどの利害関係者が、ベンダ契約に盛り込まれるセキュリティ要件を確認する		
◆ プロジェクトチームの策定するセキュリティ要件に対して監査が行われる		SR 3

セキュアなアーキテクチャ

	Yes/No	
◆ 推奨されるサードパーティ製コンポーネントのリストがプロジェクトチームに提供される		
◆ ほとんどのプロジェクトチームがセキュアな設計の原則を認識し、適用する		SA 1
◆ 共有のセキュリティサービスに関する情報が、手引きでプロジェクトチームに周知されている		
◆ プロジェクトチームに対し、該当するアプリケーションのアーキテクチャに基づく規範的なデザインパターンが提供される		SA 2
◆ プロジェクトチームでのソフトウエア構築が、一元管理されたプラットフォームやフレームワークを使って行われる		
◆ プロジェクトチームに対し、セキュアなアーキテクチャ構成要素の使用に関する監査が行われる		SA 3

検 証

査定ワークシート

設計レビュー

	Yes/No	
◆ ソフトウエアの設計上、攻撃者の侵入口となり得る箇所についての情報が、プロジェクトチームによって文書化されている		
◆ ソフトウエア設計が、プロジェクトチームにより既知のセキュリティリスクに照らしてチェックされている		DR 1
◆ ほとんどのプロジェクトチームが、設計要素に対し、セキュリティの仕組みに関する分析作業を明示的に行っている		
◆ ほとんどのプロジェクト利害関係者が、正式な設計レビューを受ける方法を認識している		DR 2
◆ 設計レビューのプロセスに詳細なデータレベルの分析作業が組み込まれている		
◆ 定形的なプロジェクト監査において、設計レビューの結果が一定の基本水準を満たすことが義務付けられている		DR 3

コードレビュー

	Yes/No	
◆ ほとんどのプロジェクトチームが、一般的な問題についてのレビューチェックリストを用意している		
◆ プロジェクトチームが、選出された高リスクのコードに対しておおむねレビューを実施している		CR 1
◆ セキュリティの問題を発見するための自動コード分析ツールが、ほとんどのプロジェクトチームで利用できるようになっている		
◆ ほとんどの利害関係者がコードレビューの結果を常に要求し、確認している		CR 2
◆ プロジェクトチームで、アプリケーション固有のコーディング標準に照らしたコードチェックが自動化手法を利用して行われている		
◆ 定形的なプロジェクト監査において、コードレビューの結果がリリース前に一定の基本水準を満たすことが義務付けられている		CR 3

セキュリティテスト

	Yes/No	
◆ プロジェクトにおいて、要件に基づく何らかのセキュリティテストを定めている		
◆ ほとんどのプロジェクトがリリース前の侵入テストを実施している		
◆ ほとんどの利害関係者がリリース前のセキュリティテストの状況について認識している		ST 1
◆ プロジェクトによるセキュリティテストケースの評価に自動化手法が使われている		
◆ ほとんどのプロジェクトが、セキュリティテストの評価と利害関係者への報告を一貫したプロセスに従って行っている		ST 2
◆ アプリケーション固有のロジックに対して総合的なセキュリティテストケースが作られている		
◆ 定形的なプロジェクト監査において、セキュリティテストの結果が最低基準を満たすことが要求されている		ST 3

配 備

査定ワークシート

脆弱性管理

	Yes/No	
◆ ほとんどのプロジェクトにセキュリティ問題の連絡担当者がいる		
◆ セキュリティ対応を担当するチームが組織に存在する		
◆ ほとんどのプロジェクトチームが、セキュリティ問題に関する各部署の連絡担当者及び対応チームについて認識している		VM1
◆ インシデントの報告と対応に関して、組織内に一貫したプロセスが設定されている		
◆ ほとんどのプロジェクト利害関係者が、関係するソフトウエアプロジェクトに関連する適切なセキュリティ情報公開について認識している		VM2
◆ ほとんどのインシデントについて根本原因の調査が実行され、以後の推奨事項が作成される		
◆ ほとんどのプロジェクトにおいて、インシデントに関連するデータや指標の収集・報告が一貫した方法で行われる		VM3

環境の堅牢化

	Yes/No	
◆ 大半のプロジェクトにおいて、運用環境に関する何らかの要件が文書化されている		
◆ ほとんどのプロジェクトで、サードパーティ製ソフトウエアコンポーネントのセキュリティ更新プログラムのチェックが行われている		EH1
◆ 重要な依存対象要素に対するアップグレードやパッチの適用が、一貫した手順で実行されている		
◆ ほとんどのプロジェクトが、アプリケーションと環境の正常性チェックに自動化手法を採用している		EH2
◆ 利害関係者が、運用時に動作中のソフトウエアを保護するための追加ツールの選択肢を認識している		
◆ 定形的な監査で、環境の正常性が基本水準を満たすことがほとんどのプロジェクトについてチェックされている		EH3

運用体制のセキュリティ対応

	Yes/No	
◆ 大半のソフトウエアリリース時に、セキュリティに関するメモもあわせて配布している		
◆ セキュリティ関連の警告やエラー条件が、ほとんどのプロジェクトについて文書化されている		OE1
◆ ほとんどのプロジェクトが、十分な理解を得た変更管理プロセスを利用している		
◆ プロジェクトチームが、運用のセキュリティに関するガイドを製品リリースごとに提供している		OE2
◆ リリースのたびに適切な運用セキュリティ情報が提供されているかをチェックする監査がほとんどのプロジェクトに対して実行されている		
◆ ソフトウエアコンポーネントに対するコード署名が、一貫したプロセスによって定形的に実行されている		OE3

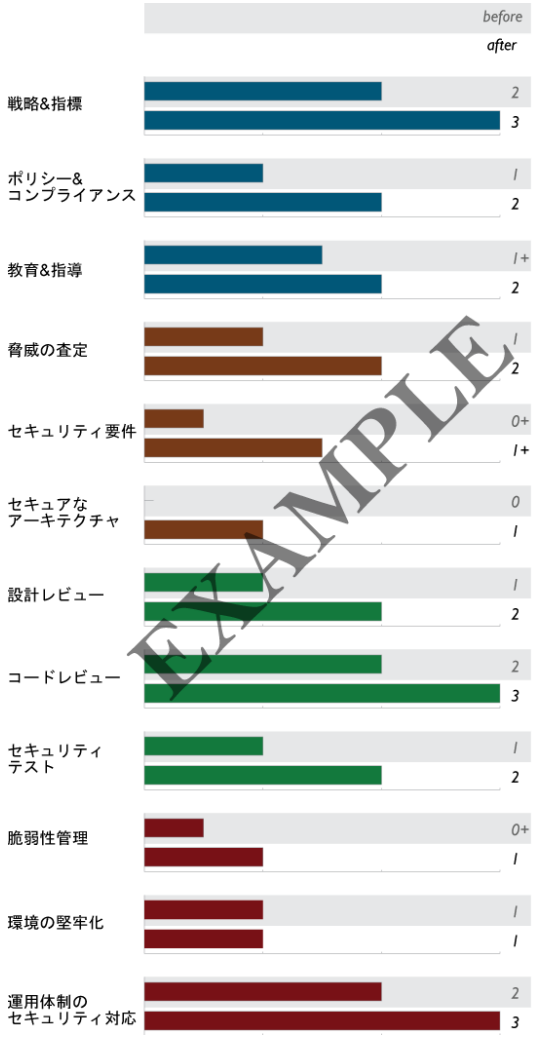
スコアカードの作成

各「セキュリティ対策」に割り当てたスコアに基づき、それらの値を記録するスコアカードを作成することができる。スコアカードは、機能的には単に特定時点における 12 のスコアを集めたものに過ぎないが、一定の間隔を決めてスコアカードを作成すると、その時間枠内にセキュリティ保証プログラム全体に生じた変化を理解しやすくなる。

間隔をおいてスコアカードを作成することが推奨されるのは次のような場合である。

- ギャップ分析 — 詳細な査定から得られるスコアと、期待されるパフォーマンス水準とを比較する
- 改善効果の実証 — セキュリティ保証プログラムを構築する反復作業を開始する前と実施した後のスコアを比較する
- 経過の測定 — 実施中のセキュリティ保証プログラムについて、一定の時間枠ごとにスコアを確認する

右図は、ある組織のセキュリティ保証プログラムに 1 年間に生じた変化を示すスコアカードの例である。もし、この組織が開始時点に計画していた 1 年後の状況に関するデータが残っているとすれば、いっそう興味深い図を作成できる。その情報を加えることで、1 年のうちに計画をどの程度修正する必要が生じたかを確認しやすくなる。



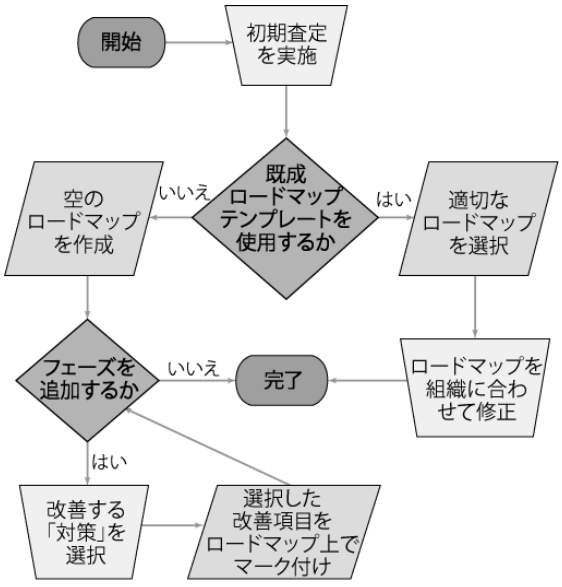
セキュリティ保証プログラムの構築

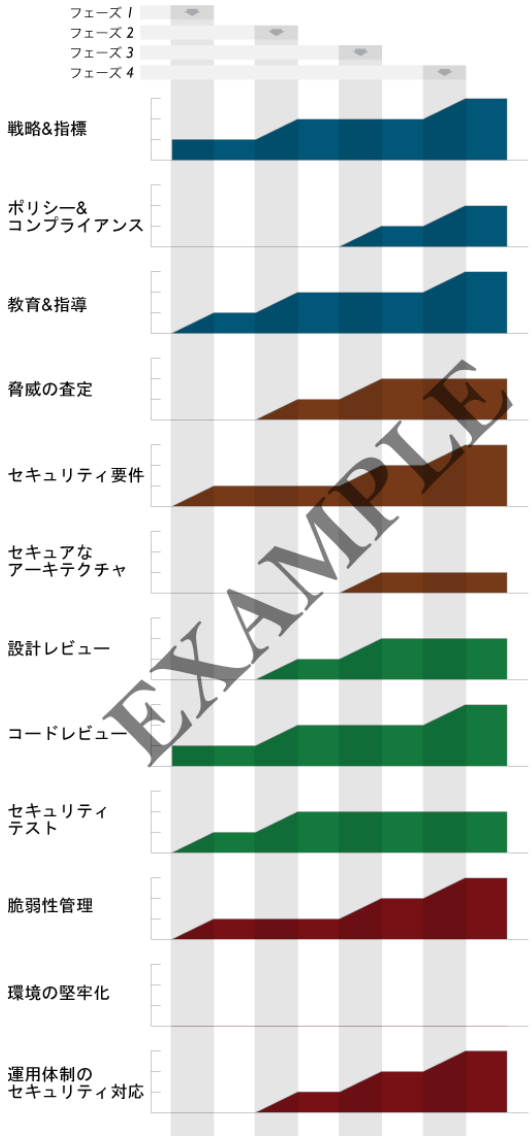
SAMM の主要な用途の 1 つは、組織におけるソフトウエアセキュリティ保証プログラムの構築を支援することである。そのプロセスは明快であり、当該組織で何らかのセキュリティ保証活動をすでに実施しているかどうか査定する作業から始めるのが一般的である。

一般的な種類の組織に合わせたロードマップテンプレートが数種類用意されている。多くの組織では、自組織に合った適切なロードマップテンプレートを選び、必要に応じた変更を加えて使用できる。その他の種類の組織では、ロードマップを独自に作成することが必要な場合がある。

ロードマップ（右図）は、各「対策」における「レベル」の進行段階を示すフェーズ（縦列）で構成されている。したがって、ロードマップの構築作業においては、計画のいずれのフェーズでいずれの「対策」を改善するのかを選択していくことになる。計画にどこまで遠い将来を含めるかは組織の自由であるが、セキュリティ保証目標をビジネス上の目標とリスク耐性に見合ったものにするには、ビジネス上の推進要因や組織に特有の情報に基づいて反復的にプログラムを進めることが推奨される。

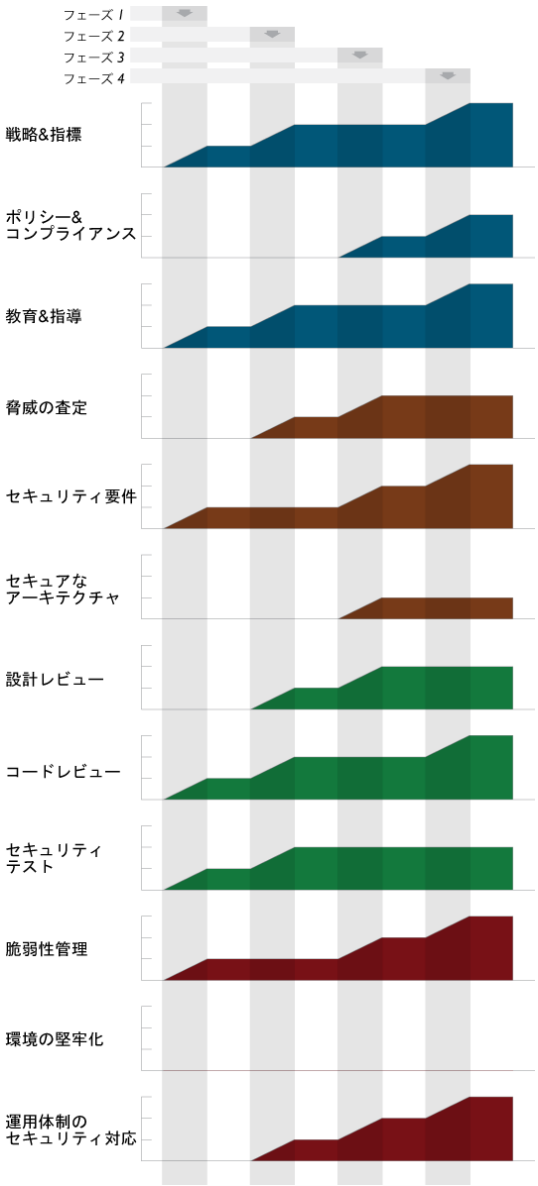
ロードマップを設定した後のセキュリティ保証プログラムの構築は簡単である。単純に 1 つの改善フェーズを開始し、計画されている「レベル」を達成するための所定の活動を実施するだけである。フェーズの終わりには、実際に達成できた内容に応じてロードマップを修正し、その上で次のフェーズを開始する。





独立系ソフトウエアベンダ

ロードマップテンプレート



方針

独立系ソフトウエアベンダにおいては、ソフトウエアコンポーネントやアプリケーションの構築と販売が主要なビジネス機能である。

顧客やユーザに影響を及ぼす一般的な脆弱性を限定することが初期の推進要因になるため、当初はコードレビュー及びセキュリティテストに注力する。

フェーズが進むと、セキュリティエラーを製品仕様において事前予防することに対する関心が強まるため、セキュリティ要件に関する実施内容が加わる。

また、セキュリティ問題が発見された場合の影響を最小化するために、脆弱性管理に関する実施内容が充実してくる。

組織の成熟度が上がると、ソフトウエアのセキュアな運用に関する情報を顧客やユーザによりよく伝えるために、運用体制のセキュリティ対応に関する実施内容が追加される。

その他の考慮事項

開発の外注化

外部の開発リソースを利用する組織の場合は、コードへのアクセスを制限する必要性から、コードレビューよりもセキュリティ要件に関する実施内容の優先順位が高くなるのが一般的と考えられる。また、脅威の査定を早期のフェーズで進めると、外注先の開発者にセキュリティの必要性をより明確に説明できる可能性がある。ソフトウエアの構成に関する専門知識は、開発の外注先となるグループにおいてもっとも充実すると考えられる。したがって、契約条件は運用体制のセキュリティ対応に関する実施内容を考慮して作成すべきである。

インターネット接続アプリケーション

オンラインリソースを利用するアプリケーションを構築する組織の場合、インターネットに向けたシステムをホストするために主要インフラストラクチャがインターネットに面して設置されることから、より大きなリスクが生じる。このリスクを考慮して、環境の堅牢化に関する実施内容をロードマップに加えるべきである。

ドライバ及び組み込みシステム開発

低レベルドライバや組み込みシステムを構築する組織の場合、ソフトウエアの設計にセキュリティ脆弱性が発見されたときの被害の大きさと修復コストがさらに大きくなる可能性がある。したがって、セキュアなアーキテクチャ及び設計レビューに関する実施内容を早期のフェーズで重視するようにロードマップを修正すべきである。

企業買収による組織規模の拡大

買収による組織規模拡大の際には、開発モデルもセキュリティ関連活動の導入度合いも異なるいくつかのプロジェクトチームが併存する状況になることが多い。このような場合、部門またはプロジェクトチームごとに当初の状況が異なることや、開発対象ソフトウエアの種類が多様化してプロジェクトごとに特有の考慮事項が生じることから、別々のロードマップの使用を余儀なくされる可能性がある。

オンラインサービスプロバイダ

ロードマップテンプレート

方針

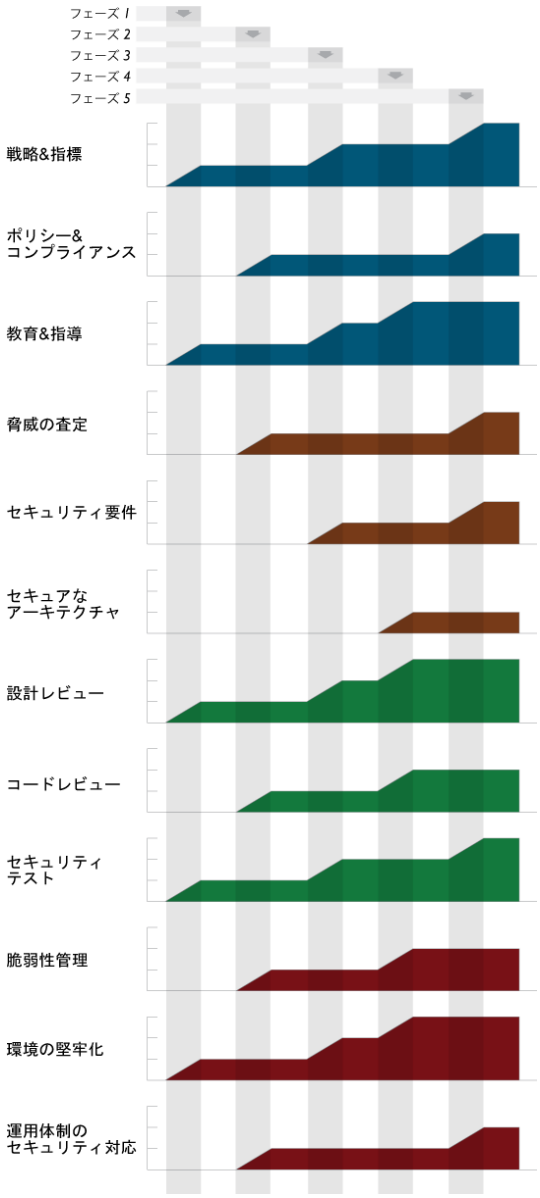
オンラインサービスプロバイダにおいては、Web アプリケーションやその他ネットワーク対応インターフェイスの構築が、主要なビジネス機能である。

技術革新を妨げることなく設計全体の健全性を証明することが初期の推進要因になるため、当初は設計レビュー及びセキュリティテストに注力する。

また、重要なシステムがネットワークに接することから、ホスト環境に起因するリスクを考慮して、環境の堅牢化に関する実施内容が早期に追加される。

組織の主要なビジネスの内容にもよるが、ポリシー&コンプライアンスに関する実施内容に早いフェーズで着手し、外的なコンプライアンス推進要因の重要度に応じて段階を進めるべきであると考えられる。

組織の成熟度が上がるにつれて、脅威の査定、セキュリティ要件、セキュアなアーキテクチャのそれぞれに関する実施内容が徐々に追加される。これは、セキュリティに関する期待値の基本水準がある程度確立した後の事前予防的なセキュリティを補強するためである。



その他の考慮事項

開発の外注化

外部の開発リソースを利用する組織の場合は、コードへのアクセスを制限する必要性から、コードレビューよりもセキュリティ要件に関する実施内容の優先順位が高くなるのが一般的と考えられる。また、脅威の査定を早期のフェーズで進めると、外注先の開発者にセキュリティの必要性をより明確に説明できる可能性がある。ソフトウエアの構成に関する専門知識は、開発の外注先となるグループにおいてもっとも充実すると考えられる。したがって、契約条件は運用体制のセキュリティ対応に関する実施内容を考慮して作成すべきである。

オンライン支払い決済処理

クレジットカードの国際セキュリティ基準である PCI-DSS（Payment Card Industry Data Security Standard）や、オンライン決済に関するその他の標準を遵守することが要求される組織の場合、ポリシー&コンプライアンスに関する実施内容をロードマップの早期のフェーズに配置すべきである。そうすれば、コンプライアンスを保証するための活動を可能な範囲で確立しながら、進捗に応じて将来のロードマップを修正していくことができる。

Web サービスプラットフォーム

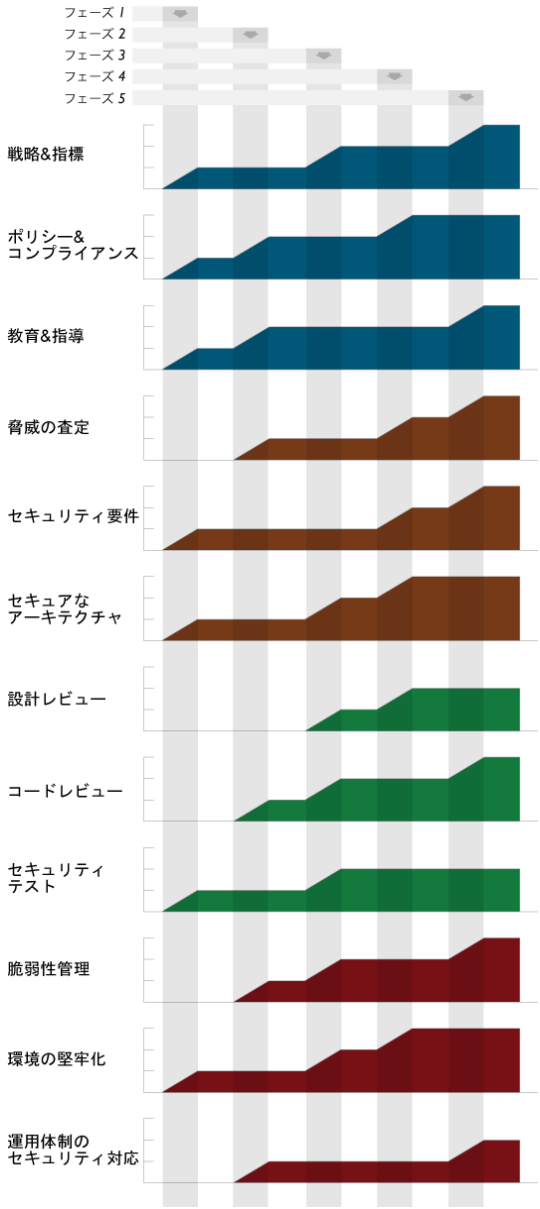
Web サービスのプラットフォームを構築する組織の場合、設計上の誤りがより大きなリスクを生み、問題を緩和するためのコストが大きくなる可能性がある。したがって、脅威の査定、セキュリティ要件、セキュアなアーキテクチャのそれぞれに関する実施内容をロードマップの早期のフェーズに配置すべきである。

企業買収による組織規模の拡大

買収による組織規模拡大の際には、開発モデルもセキュリティ関連活動の導入度合いも異なるいくつかのプロジェクトチームが併存する状況になることが多い。このような場合、部門またはプロジェクトチームごとに当初の状況が異なることや、開発対象ソフトウエアの種類が多様化してプロジェクトごとに特有の考慮事項が生じることから、別々のロードマップの使用を余儀なくされる可能性がある。

金融サービス機関

ロードマップテンプレート



方針

金融サービス機関においては、金融上の取引・処理を支援するシステムの構築が主要なビジネス機能である。通常、この種のシステムでは、まったく異なる種類の外部データプロバイダとやり取りする内部システムやバックエンドシステムの集中化の度合いが大きい。

きわめて重要な性質のサービスであることから、最初はガバナンスに関する実施内容の改善に注力する。これによってセキュリティ保証プログラムの基本水準が決まり、組織のコンプライアンス要件が満たされやすくなる。

セキュアかつ信頼性の高いソフトウエアを事前予防的に構築することが全体的な目標とされることから、構築に関する実施内容には早期のフェーズで着手し、プログラムの成熟につれて急速に改善を進める。

また、レガシーシステム対応が無理のない期待水準で行われるように、検証に関する実施内容はロードマップの全期間を通してスムーズに改善を進める。このことは、構築に十分なサイクルを費やし、より事前予防効果の高い「対策」を実現するためにも有効である。

金融サービス機関の運用は独自に構築したソフトウエアを使用して行われることが多いため、初期のガバナンス対策がある程度実現された後、ロードマップの中期には配備に関する実施内容を重視する。後期には、構築に関する事前予防的な実施内容を非常に重視する。

その他の考慮事項

開発の外注化

外部の開発リソースを利用する組織の場合は、コードへのアクセスを制限する必要性から、コードレビューよりもセキュリティ要件に関する実施内容の優先順位が高くなるのが一般的と考えられる。また、脅威の査定を早期のフェーズで進めると、外注先の開発者にセキュリティの必要性をより明確に説明できる可能性がある。ソフトウエアの構成に関する専門知識は、開発の外注先となるグループにおいてもっとも充実すると考えられる。したがって、契約条件は運用体制のセキュリティ対応に関する実施内容を考慮して作成すべきである。

Web サービスプラットフォーム

Web サービスのプラットフォームを構築する組織の場合、設計上の誤りがより大きなリスクを生み、問題を緩和するためのコストが大きくなる可能性がある。したがって、脅威の査定、セキュリティ要件、セキュアなアーキテクチャのそれぞれに関する実施内容をロードマップの早期のフェーズに配置すべきである。

企業買収による組織規模の拡大

買収による組織規模拡大の際には、開発モデルもセキュリティ関連活動の導入度合いも異なるいくつかのプロジェクトチームが併存する状況になることが多い。このような場合、部門またはプロジェクトチームごとに当初の状況が異なることや、開発対象ソフトウエアの種類が多様化してプロジェクトごとに特有の考慮事項が生じることから、別々のロードマップの使用を余儀なくされる可能性がある。

政府機関

ロードマップテンプレート

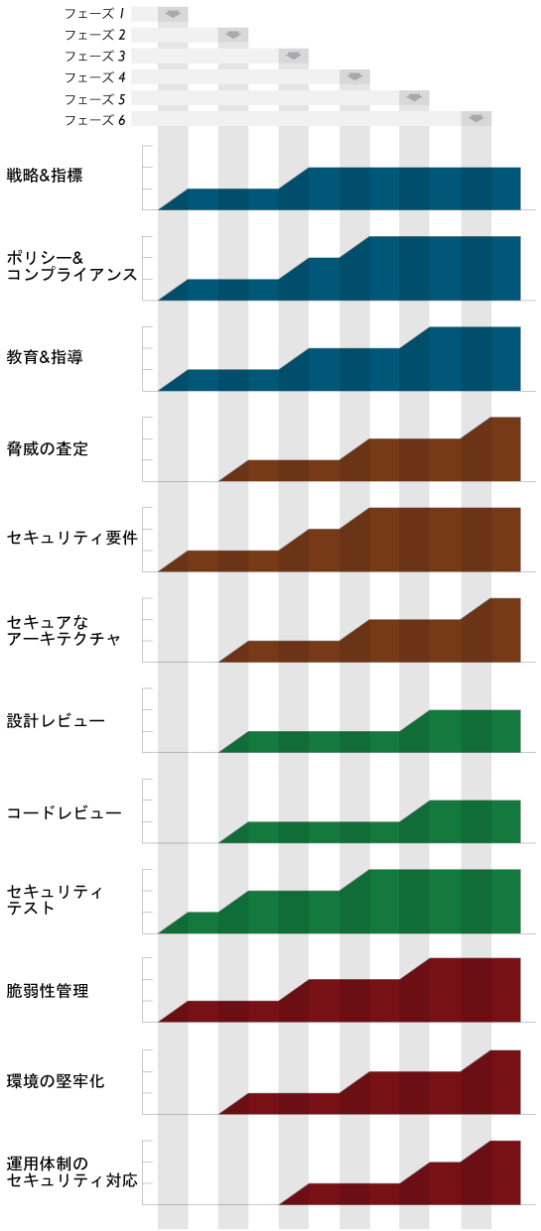
方針

政府機関においては、公共部門のプロジェクトを支援するソフトウエアの構築が主要なビジネス機能である。

具体的なロードマップに従って改善計画を進める中で組織に課される全体的なコンプライアンス負担を把握する目的などのために、当初はガバナンスに関する実施内容を確立する。

公共のアクセスにさらされるリスクと、大量のレガシーコードが稼働していると考えられる点を考慮して、検証機能のうちセキュリティテストに関する実施内容を早期のフェーズでは重視し、その後、より詳細なコードレビュー及び設計レビューに関する実施内容を充実させていく。

構築及び配備に関する実施内容についても、同様の考え方で重点を移していく。これは、組織として脆弱性の管理体制を確立しやすくするため、また、運用環境のセキュリティ保護体制を強化していくためである。同時に、開発中のソフトウエアに新たな問題が発生するのを防ぐため、構築に関する事前予防的な実施内容を進める。



その他の考慮事項

開発の外注化

外部の開発リソースを利用する組織の場合は、コードへのアクセスを制限する必要性から、コードレビューよりもセキュリティ要件に関する実施内容の優先順位が高くなるのが一般的と考えられる。また、脅威の査定を早期のフェーズで進めると、外注先の開発者にセキュリティの必要性をより明確に説明できる可能性がある。ソフトウエアの構成に関する専門知識は、開発の外注先となるグループにおいてもっとも充実すると考えられる。したがって、契約条件は運用体制のセキュリティ対応に関する実施内容を考慮して作成すべきである。

Web サービスプラットフォーム

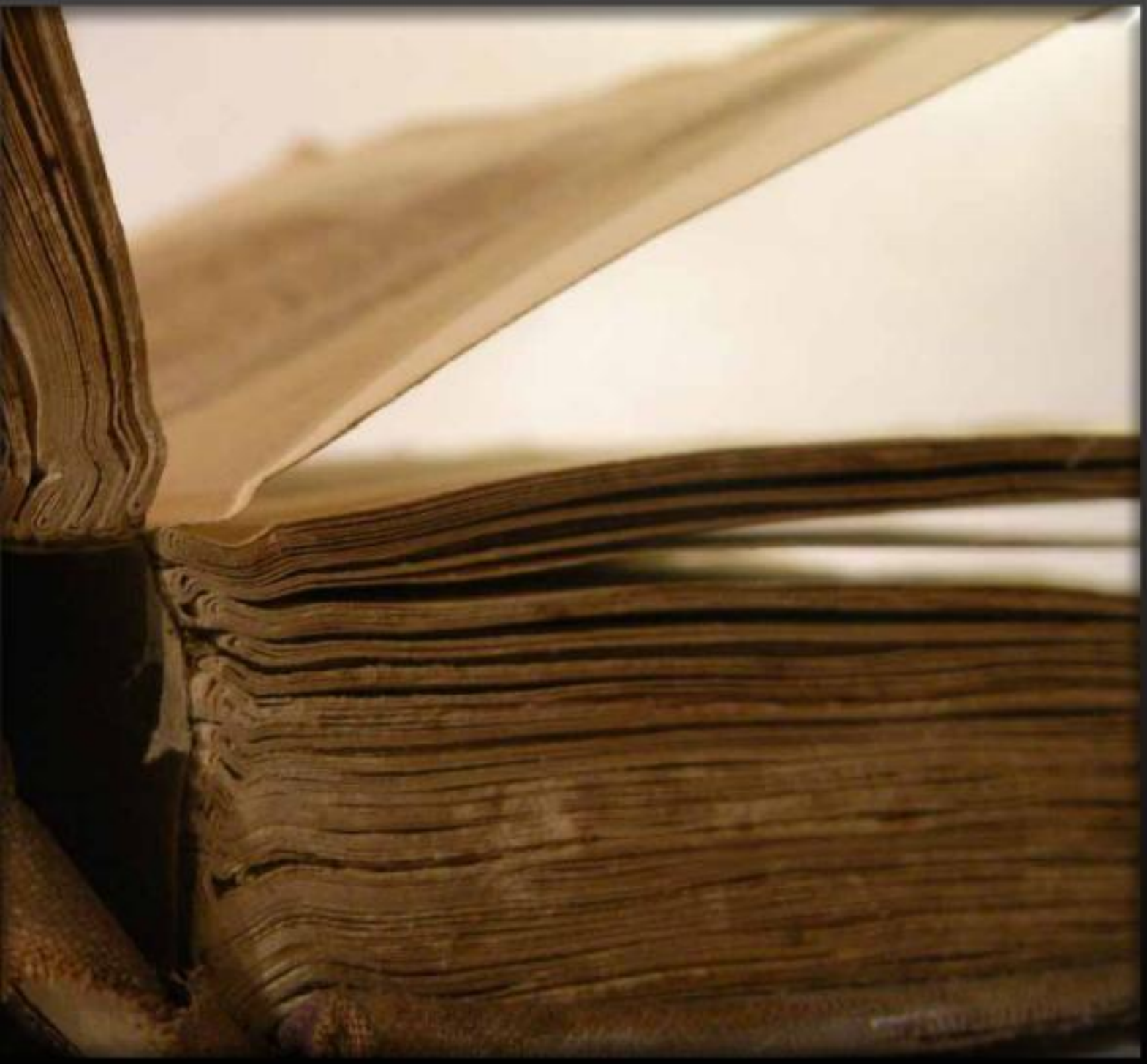
Web サービスのプラットフォームを構築する組織の場合、設計上の誤りがより大きなリスクを生み、問題を緩和するためのコストが大きくなる可能性がある。したがって、脅威の査定、セキュリティ要件、セキュアなアーキテクチャのそれぞれに関する実施内容をロードマップの早期のフェーズに配置すべきである。

規制に対するコンプライアンス

ビジネスプロセスに影響する厳しい規制の下で運営される組織の場合、そうした外的な推進要因に対応するように、ポリシー&コンプライアンスに関する実施内容の構築を修正すべきである。逆に、コンプライアンス負担が軽微な組織の場合は、そのことを利用してポリシー&コンプライアンスに関する実施内容の構築を遅らせ、代わりにほかの実施内容を進めるべきである。

セキュリティ対策

詳細内容の説明



この節では、SAMM の構成要素となる「セキュリティ対策」の「成熟度レベル」を定義する。まず、各「対策」に関する 3 段階の「レベル」の要約を表にまとめて示す。その後、「レベル」ごとに必要とされる実施内容、達成によって得られるもの、パフォーマンスを測定するための成功指標、継続的に必要とされる人的投資、及びその他の関連コストについて説明する。

戦略&指標

	SM 1	SM 2	SM 3
目的	**組織内のソフトウエアセキュリティに関する統一的な戦略ロードマップを設定する**	**データ及びソフトウエア資産の相対的価値を測定し、リスク耐性を選択する**	**セキュリティ関連の支出を、適切なビジネス指標及び資産価値に整合させる**
実施内容	A. ビジネス全体のリスクプロファイルを見積もる B. セキュリティ保証プログラムのロードマップを設定し、維持管理する	A. ビジネス上のリスクに基づいてデータとアプリケーションを分類する B. 分類ごとの目標を設定し、測定する	A. 業界全体を対象としたコスト比較を定期的に実施する B. セキュリティ支出の推移に関する指標を収集する
査定項目	◆ すでに施行しているソフトウエアセキュリティ保証プログラムがある ◆ ほとんどのビジネス利害関係者が組織のリスクプロファイルを理解している ◆ ほとんどの開発要員がセキュリティ保証プログラムの将来計画を認識している	◆ ほとんどのアプリケーションとリソースがリスクに基づいて分類される ◆ 必要なセキュリティ保証活動をカスタマイズするためにリスク評価が使われる ◆ リスク評価に基づいて、必要な事項を組織の人員のほとんどが理解している	◆ セキュリティ保証活動のコストに関するプロジェクトごとのデータが収集される ◆ 組織としてのセキュリティ支出が、ほかの組織と随時比較される
成果	◆ ソフトウエアによってビジネスレベルに生じるもっとも重大なリスクを具体的に列挙したリストができる ◆ 当該組織のセキュリティニーズに最小限のオーバヘッドで対応するカスタマイズされたロードマップができる ◆ 時間の経過に伴って自組織のセキュリティ保証プログラムがどう発展していくのかを組織全体が理解する	◆ 当該組織の主要なビジネス価値に基づく、プロジェクトごとにカスタマイズされたセキュリティ保証計画が策定される ◆ データ及びアプリケーション資産のセキュリティ対応状況を組織全体が理解する ◆ リスクの理解と受容について利害関係者の知識が深まる	◆ セキュリティ支出の案件ごとに適切な意思決定が可能な判断材料を入手できる ◆ セキュリティ問題によって生じた過去の損失の見積もりができる ◆ セキュリティ支出と損失可能性のバランスをプロジェクトごとに検討できる ◆ セキュリティに関して業界全体に通用するデューディリジェンスが行われる

戦略&指標

SM 1

組織内のソフトウエアセキュリティに関する統一的な戦略ロードマップを設定する

実施内容

A. ビジネス全体のリスクプロファイルを見積もる

ビジネスオーナーと利害関係者に対する聞き取り調査を実施し、組織のさまざまなアプリケーション及びデータ資産に関するワーストケースシナリオのリストを作成する。どのような方法でソフトウエアを構築・使用・販売する組織であるかによってワーストケースシナリオのリストは大きく異なるが、共通的な問題として、データの盗難や破損、サービス提供の中断、金銭的損害、リバースエンジニアリング、アカウント情報の漏洩などを挙げることができる。

幅広いワーストケースシナリオを収集した後、集めた情報と主たる業務の知識に基づいて各種シナリオを分析し、特に重要なものを選び出す。選ぶ件数はいくつでもよいが、時間を効率よく使うことと作業のポイントを明確に保つことを考慮して 3 件以上 7 件以下の範囲にするのが望ましい。

選択した各項目の記述内容を充実・洗練させ、当該組織において関連するワーストケースシナリオ、関連しうる要因、及び問題を緩和する方向に作用しうる要因を詳細に文書化する。

最終的に作成されるビジネスリスクプロファイルについては、ビジネスオーナー及びその他利害関係者の査読を受けて理解を得る。

B. セキュリティ保証プログラムのロードマップを設定し、維持管理する

組織の直面する主要なビジネスリスクを理解し、12 の「対策」それぞれについて組織の現状のパフォーマンスを評価する。各「対策」に関する累積的な成功指標の達成状況に応じて、「目的」に対応する 1 点、2 点または 3 点のスコアを割り当てる。満たしている成功指標がない場合、その「対策」についてのスコアは 0 である。

現状を十分に把握した後は、次回の反復作業においてどの「対策」を改善するかを決めることが次の目標である。この判断は、ビジネスリスクプロファイル、その他のビジネス推進要因、コンプライアンス要件、予算の許容範囲などに基づいて行われる。対象とする「対策」を選択した後は、各「対策」に関する次の「目的」を達成することがその反復における目標となる。

セキュリティ保証プログラムの反復的な改善作業は約 3〜6 カ月かけて行う。ただし、少なくとも 3 カ月に 1 回はセキュリティ保証戦略セッションを開催して、実施内容の進捗と、成功指標その他ビジネス推進要因に照らしたパフォーマンスをレビューし、プログラムの修正が必要かどうかを検討する。

成果

- ソフトウエアによってビジネスレベルに生じるもっとも重大なリスクを具体的に列挙したリストができる
- 当該組織のセキュリティニーズに最小限のオーバヘッドで対応するカスタマイズされたロードマップができる
- 時間の経過に伴って自組織のセキュリティ保証プログラムがどう発展していくのかを組織全体が理解する

成功指標

- 直近 6 カ月以内に、80%超の利害関係者に対してビジネスリスクプロファイルの概要を説明した
- 直近 3 カ月以内に、80%超の要員に対してセキュリティ保証プログラムのロードマップの概要を説明した
- 直近 3 カ月以内に複数回のプログラム戦略セッションを開催した

コスト

- ビジネスリスクプロファイルの構築と維持管理
- 四半期ごとのセキュリティ保証プログラム評価

人員

- 開発者（1 日/年）
- アーキテクト（4 日/年）
- マネージャー（4 日/年）
- ビジネスオーナー（4 日/年）
- QA テスト担当者（1 日/年）
- セキュリティ監査員（4 日/年）

関連レベル

- ポリシー&コンプライアンス - 1
- 脅威の査定 - 1
- セキュリティ要件 - 2

データ及びソフトウエア資産の相対的価値を測定し、リスク耐性を選択する

成果

- 当該組織の主要なビジネス価値に基づく、プロジェクトごとにカスタマイズされたセキュリティ保証計画が策定される
- データ及びアプリケーション資産のセキュリティ対応状況を組織全体が理解する
- リスクの理解と受容について利害関係者の知識が深まる

追加成功指標

- 直近 12 カ月以内に、90%超のアプリケーション及びデータ資産についてリスク分類のための評価を実施した
- 直近 6 カ月以内に、80%超の要員に対して、関係するアプリケーション及びデータのリスク評価の概要を説明した
- 直近 3 カ月以内に、80%超の要員に対して、関係するセキュリティ保証プログラムのロードマップの概要を説明した

追加コスト

- アプリケーション及びデータに関するリスク分類手法の構築またはライセンス
- ロードマップ計画の緻密度が上がることによるプログラムオーバヘッド

追加人員

- アーキテクト（2 日/年）
- マネージャー（2 日/年）
- ビジネスオーナー（2 日/年）
- セキュリティ監査員（2 日/年）

関連レベル

- ポリシー&コンプライアンス - 2
- 脅威の査定 - 2
- 設計レビュー - 2

実施内容

A. ビジネス上のリスクに基づいてデータとアプリケーションを分類する

アプリケーションのリスク階層を表す単純な分類システムを確立する。たとえば、もっとも単純な形であれば「高/中/低」の分類にする。もっと複雑な分類を使用してもよいが、数は 7 段階以下とし、また、ビジネスリスクに影響する程度の高低を大まかに示すような段階設定とする。

まず組織のビジネスリスクプロファイルに関する作業から始めて、個々のプロジェクトといずれか 1 つのリスク分類とを対応付けるプロジェクト評価条件を作成する。データ資産についても同様の分類方法を、別途作成し、ビジネスリスクに及ぼしうる影響の大きさに基づいて各項目の重み付けと分類を決定する。

各アプリケーションについて収集した情報を評価し、全体的な評価条件及び使用中のデータ資産のリスク分類に基づいて、各アプリケーションのリスク分類を決定する。これは、必要な情報を収集するためにカスタマイズした調査票を使用して、セキュリティグループで一元的に行うことも、あるいはプロジェクトチームが個別に行うこともできる。

アプリケーション及びデータ資産のリスクを分類する継続的なプロセスを確立することで、新しく加わる資産を分類するとともに、既存の情報を年に 2 回以上は更新する。

B. 分類ごとの目標を設定し、測定する

組織における各種アプリケーションの使用状況に対して一定の分類方法を適用することで、直接的なセキュリティ目標とセキュリティ保証プログラムのロードマップに関して、よりきめ細かな選択ができるようになる。

個々のアプリケーションリスク分類においてどの具体的な「対策」を重視するかを決定し、それらの分類を考慮して、セキュリティ保証プログラムのロードマップに修正を加える。典型的な方法としては、セキュリティ保証プログラムにおける反復作業の 1 回ごとに、リスクがもっとも大きいアプリケーション層の高レベルな「目的」に高い優先順位を設定し、ほかの分類にはリスクが小さくなるに従って要求の低い「目的」を設定していく形が考えられる。

このプロセスを実行すると、個々のリスク分類に属するアプリケーションに期待される具体的な「目的」について能動的な意思決定が必要になるため、当該組織のリスク耐性が明確になる。低リスクのアプリケーションを「セキュリティ対策」に関するパフォーマンスが低いレベルにとどめることを選択すれば、重み付けされたリスクを受容するのと引き換えにリソースを節約できる。しかし、リスク分類ごとのロードマップを別々に適宜作成する必要はない。そのようにするとセキュリティ保証プログラム自体の管理が非効率になりかねない。

戦略&指標

セキュリティ関連の支出を、適切なビジネス指標及び資産価値に整合させる

実施内容

A. 業界全体を対象としたコスト比較を定期的に実施する

業界内の情報交換フォーラム、ビジネスアナリスト、コンサルティング会社、その他の外部情報源を利用して、セキュリティコストに関する情報を研究・収集する。中でも、重要ないくつかの要因について特定する必要がある。

まず、収集した情報を使用して、同じ業界に属する同様の組織内で行われているセキュリティに対する取り組みの平均的な量を特定する。これは、合計額が予算や売上に占める割合の見積もりからトップダウンで実行してもよいし、該当する種類の組織において一般的と考えられるセキュリティ関連活動を特定することからボトムアップで実行してもよい。この作業の対象となる情報の測定は業界によって困難な場合があるので、利用できる限り多くの情報源を利用して情報収集活動を行う。

セキュリティコストの調査における次の目標は、組織で現在使用しているサードパーティ製セキュリティ製品及びサービスについて、コスト節減の可能性があるかどうかを確認することである。異なるベンダへの乗り換えを検討する場合は、要員の再教育やその他のプログラムオーバヘッドなど、見えないコストを考慮する必要がある。

以上のようなコスト比較作業を、少なくとも年に１回、以下で述べるセキュリティ保証プログラム戦略セッションの前に実施する。また、セキュリティ保証プログラムと実際のビジネスとの整合性を高めるために、比較の情報を利害関係者に示す。

B. セキュリティ支出の推移に関する指標を収集する

過去のセキュリティインシデントで発生したコストについて、プロジェクトごとの固有の情報を収集する。たとえば、攻撃を受けた後の収拾作業に要した時間・費用、システムの停止による金銭的な損失、規制当局から課された罰金・料金、当該プロジェクトに限ってツールやサービスに支払った単発のセキュリティ費用などがこれに該当する。

各アプリケーションに関するセキュリティコスト基本水準については、アプリケーションリスク分類とそれぞれの分類に対応した規範的に示されたセキュリティ保証プログラムロードマップを使用し、対応するリスク分類の関連コストから初期の見積もりを作成できる。

アプリケーション固有のコスト情報とリスク分類に基づく一般的なコストモデルを組み合わせ、その上で、異常値（リスク分類にふさわしくない合計値など）が生じたプロジェクトについて評価する。異常値は、リスク評価や分類に誤りがあったことや、セキュリティコストの根本原因をより効率よく追求するために組織のセキュリティ保証プログラムを再調整する必要があることを示すものである。

プロジェクト単位のセキュリティ支出については、四半期ごとにセキュリティ保証プログラム戦略セッションにおいて状況を確認し、少なくとも年に１回は利害関係者によるレビュー及び評価を行う。異常値や想定外のコストについては、セキュリティ保証プログラムのロードマップに影響する可能性を検討する。

成果

- セキュリティ支出の案件ごとに適切な意思決定が可能な判断材料を入手できる
- セキュリティ問題によって生じた過去の損失の見積もりができる
- セキュリティ支出と損失可能性のバランスをプロジェクトごとに検討できる
- セキュリティに関して業界全体に通用するデューディリジェンスが行われる

追加成功指標

- 直近３カ月以内に、80%超のプロジェクトについてセキュリティコストが報告された
- 直近１年以内に、業界全体についてのコスト比較作業が複数回行われた
- 直近１年以内に、過去のセキュリティ支出動向についての評価作業が複数回行われた

追加コスト

- セキュリティプログラムに関する業界の情報を調査研究する能力の構築またはライセンス
- コストの見積もり・追跡管理・評価によるプログラムオーバヘッド

追加人員

- アーキテクト（1 日/年）
- マネージャー（1 日/年）
- ビジネスオーナー（1 日/年）
- セキュリティ監査員（1 日/年）

関連レベル

- 脆弱性管理 – 1

ポリシー&コンプライアンス

	PC1	PC2	PC3
目的	**ガバナンスとコンプライアンスについての当該組織に該当する推進要因を把握する**	**セキュリティとコンプライアンスの基本水準を設定し、プロジェクトごとのリスクを把握する**	**コンプライアンスを義務付け、組織全体のポリシーと標準に照らしてプロジェクトを測定する**
実施内容	A. 外的なコンプライアンス推進要因を特定する B. コンプライアンスの指針を設定し、維持管理する	A. セキュリティ及びコンプライアンスのポリシーと標準を策定する B. プロジェクト監査の方法を確立する	A. プロジェクトのためのコンプライアンスゲートを作成する B. 監査データ収集の方策を採択する
査定項目	◆ ほとんどのプロジェクト利害関係者が関係するプロジェクトのコンプライアンス状況を知っている ◆ コンプライアンス要件がプロジェクトチームで明示的に検討される	◆ ソフトウエア開発の管理に、組織としてポリシーと標準を利用している ◆ ポリシーと標準に対するコンプライアンスの監査をプロジェクトチームが要求できる	◆ プロジェクトの監査が定期的に実施され、ポリシーと標準に対するコンプライアンスの基本水準が確認される ◆ 組織として、監査を体系的に利用することでコンプライアンスの証跡を収集・管理している
成果	◆ 第三者による監査対応で良好な成果を得る可能性が高まる ◆ 内部リソースがコンプライアンス要件に基づいて整理される ◆ 当該組織に影響するような変化が規制要件に生じたことを迅速に認識できる	◆ セキュリティ及びコンプライアンスの両方に関して期待される内容について、プロジェクトチームの意識が向上する ◆ 実際の製品ラインに該当する具体的なコンプライアンスリスクがビジネスオーナーによりよく理解される ◆ 最適化されたアプローチにより、可能な範囲でセキュリティの改善を進めつつコンプライアンスを満たすことができる	◆ コンプライアンス違反によるリスクの受容状況が組織レベルで可視化される ◆ コンプライアンスについて具体的な保証がプロジェクトレベルで行われる ◆ 過去のプロジェクトのコンプライアンス実績が正確に追跡管理される ◆ 人手での作業量を軽減するツールを活用した効率的な監査プロセスが実施される

ポリシー&コンプライアンス

PC 1

ガバナンスとコンプライアンスについての当該組織に該当する推進要因を把握する

実施内容

A. 外的なコンプライアンス推進要因を特定する

組織に課される可能性のあるコンプライアンス要件は多種多様であるが、この活動は、当該組織がソフトウエアやデータを構築・利用する方法に直接または間接的な影響を及ぼす要件に関する事項を特に対象とする。コンプライアンスに特化した要員が組織内にいる場合は、その要員を活用する。

第三者による規制基準を調査し、組織の主要なビジネスに基づいて、コンプライアンスが要求される基準や業界の規範と考えられる基準を特定する。これに該当しうる基準の例としては､米 SOX 法(サーベンス･オクスリー法)､PCI-DSS（クレジットカードの国際セキュリティ基準)、米 HIPAA 法（医療保険の相互運用性と説明責任に関する法律）などがある。これらを始め個々の第三者基準を読み、内容を理解した後で、ソフトウエア及びデータに関連する具体的な要件を収集し、個々の推進要因（第三者基準）をセキュリティに関する個々の具体的要件に対応付けた 1 つのリストに統合する。この段階で、必須ではないと考えられる要件や単なる推奨事項を除外し、要件の数量をできるだけ少なくする。

第三者基準に生じる変更を組織として確実に把握しておくために、少なくとも年 2 回は調査を実施する。組織が属している業界やコンプライアンスの重要性に応じて、必要な作業量や人的関与の程度は異なるが、この作業は常に明示的に行うべきものである。

B. コンプライアンスの指針を設定し、維持管理する

コンプライアンス推進要因に応じたソフトウエア及びデータ関連の要件について、統合リストに記載した各要件への対応方法を説明文にし、それらの文を盛り込むことで統合リストの内容を拡充する。対応方法の説明文はコントロールステートメントとも呼ばれ、その内容は、当該組織が当該要件を満たすために何を実行するか（または、当該要件が該当しない理由）を概念的に説明するものである。

監査は、コントロールステートメントの記述が十分かどうかをチェックし、さらに、そのコントロールステートメント自体に照らして当該組織をチェックする形で実施されるのが一般的である。したがって、コントロールステートメントには当該組織における実際の対策が正確に示されている必要がある。また、多くの要件は、組織を変革してセキュリティ保証をさらに改善していく前に、コンプライアンスの基本水準を満たす単純かつ簡易なプロセス要素を実施して満たすことができる。

統合リストを基にして主要なギャップを特定し、その情報を、セキュリティ保証プログラム構築の将来計画を立案する際に役立てる。コンプライアンスとのギャップに関する情報を利害関係者に伝達し、コンプライアンス違反によるリスクが確実に意識されるようにする。

コントロールステートメントは、少なくとも年に 2 回更新して利害関係者とのレビューを実施する。コンプライアンス推進要因の数によっては、更新をさらに頻繁に行うことが合理的である場合もある。

成果
- 第三者による監査処理で良好な結果を得る可能性が高まる
- 内部リソースがコンプライアンス要件に基づいて整理される
- 当該組織に影響するような変化が規制要件に生じたことを迅速に認識できる

成功指標
- 直近 6 カ月以内に、コンプライアンス洗い出しミーティングを複数回開催した
- 直近 6 カ月以内に、コンプライアンスチェックリストを完成し更新した
- 直近 6 カ月以内に、利害関係者とのコンプライアンスレビューミーティングを複数回開催した

コスト
- コンプライアンスチェックリストの新規作成及び継続的な維持管理

人員
- アーキテクト（1 日/年）
- マネージャー（2 日/年）
- ビジネスオーナー（1～2 日/年）

関連レベル
- 戦略&指標 - 1

PC2 ポリシー&コンプライアンス

セキュリティとコンプライアンスの基本水準を設定し、プロジェクトごとのリスクを把握する

成果
- セキュリティ及びコンプライアンスの両方に関して期待される内容について、プロジェクトチームの意識が向上する
- 実際の製品ラインに該当する具体的なコンプライアンスリスクがビジネスオーナーによりよく理解される
- 最適化されたアプローチにより、可能な範囲でセキュリティの改善を進めつつコンプライアンスを満たすことができる

追加成功指標
- 直近 6 カ月以内に、75%超の要員に対してポリシーと標準の概要を説明した
- 80%超の利害関係者が、ポリシーと標準に関するコンプライアンス状況を認識している

追加コスト
- 内部標準の構築またはライセンス
- 内部標準及び監査へのコンプライアンスによるプロジェクトごとのオーバヘッド

追加人員
- アーキテクト（1 日/年）
- マネージャー（1 日/年）
- セキュリティ監査員（2 日/プロジェクト・年）

関連レベル
- 教育&指導 - 1 及び 3
- 戦略&指標 - 2
- セキュリティ要件 - 1 及び 3
- セキュアなアーキテクチャ - 3
- コードレビュー - 3
- 設計レビュー - 3
- 環境の堅牢化 - 3

実施内容

A. セキュリティ及びコンプライアンスのポリシーと標準を策定する

現在のコンプライアンスガイドラインを基にして規制基準をレビューし、必須でないセキュリティ要件や推奨事項レベルのセキュリティ要件を抽出する。また、適用されるコンプライアンス要件に将来変更が生じる可能性について小規模な調査を実施する。

セキュリティに関して既知のビジネス推進要因に基づく要件があればリストに追加する。この作業は、開発要員に提供されている既存の手引書を確認することや、ベストプラクティスを収集することで実施するのがもっとも簡単である場合が多い。

共通の要件や類似の要件をグループ化し、個々のグループをより一般化または単純化したステートメントに書き直して、すべてのコンプライアンス推進要因を満たしつつ付加的なセキュリティ価値を提供する内容にする。このプロセスを各グループについて実施する。その際、コンプライアンス推進要因やベストプラクティスに対する直接の対応付けが可能な内部ポリシー及び標準の策定を目標とする。

ポリシーと標準には、プロジェクトチームでそれを達成することが困難すぎる、またはコストがかかりすぎるような要件を含めないようにすることが重要である。目安として、80%程度のプロジェクトが大きな混乱を伴わず遵守できるような要件を設定するとよい。そのためには良質な情報発信プログラムを実施して、新しいポリシーや標準を周知させ、必要に応じてチームのコンプライアンス達成を支援することが要求される。

B. プロジェクト監査の方法を確立する

内部標準に照らした監査をプロジェクトチームが要求・受審できる簡単な監査プロセスを策定する。監査を担当するのは一般にセキュリティ監査員であるが、内部標準の知識がありセキュリティに詳しい要員が実施してもよい。

監査待ちプロジェクトの選別と並行して、既知のビジネスリスク尺度に基づいてプロジェクトの優先順位を設定することで、高リスクのソフトウエアに対する査定を迅速に、より頻繁に実行できる。また、監査のコスト効率を上げるために、低リスクのプロジェクトについては内部監査要件を緩和することも考えられる。

全体として、活動中のプロジェクトは必ず年に 2 回以上の監査を受けるべきである。対象アプリケーションに関する十分な監査情報が保管されていれば、初回の監査と比べて 2 回目以降の監査は簡単に実施できることが多い。

ビジネスオーナーやその他の利害関係者にもこのサービスの存在を告知し、関係するプロジェクトの監査を要求できるようにする。内部標準に含まれる個々の要件ごとに、合否の詳しい結果をプロジェクト利害関係者に通知し、評価を受ける。また、そうすることが役立つのであれば、影響内容や推奨される改善策も結果に含める。

ポリシー&コンプライアンス

コンプライアンスを義務付け、組織全体のポリシーと標準に照らしてプロジェクトを測定する

実施内容

A. プロジェクトのためのコンプライアンスゲートを作成する

組織においてセキュリティの内部標準を確立した後は、次のレベルとして、プロジェクトライフサイクル内の特定時点にコンプライアンスゲートを設定し、それらの時点での監査を義務化する。プロジェクトはゲートにおいて内部標準に照らした監査を受け、適合が認められるまで工程を進めることができない。

通常、ソフトウエアのリリース時点にはコンプライアンスゲートを設定し、リリースを公開する前にコンプライアンスチェックに合格することを義務付ける。監査の実施と問題があった場合の対策には十分な時間を確保することが重要であるから、たとえばリリースが QA に引き渡される時点など、早いタイミングで監査が開始されるようにする。

コンプライアンスゲートでは断固とした監査を行う必要がある反面、レガシープロジェクトや特殊なプロジェクトでは適合が不可能な場合も考えられるため、例外を承認するプロセスの策定も必須である。例外承認の対象とするプロジェクト件数は全体の 20%程度以下にする。

B. 監査データ収集の方策を採択する

定期監査を実施する組織では､大量の監査データが作成されていくことになる。このため、データ収集や照合時の保管・抽出管理を容易にし、機密性のある監査データに対する個人のアクセスを制限する自動化の方策が必要である。

内部標準に含まれる具体的な要件の多くについては、コードアナライザ、アプリケーション侵入テストツール、監視ソフトウエアなどの既存ツールをカスタマイズして利用することにより、内部標準へのコンプライアンスチェックを自動化できる。コンプライアンスチェック自動化の目的は、監査処理を効率化することと、公式な監査の実施前に要員が自己チェックを行えるようにすることである。また、自動的なチェックではミスが発生しにくく、問題発見までの時間も短縮される可能性がある。

情報の保管機能は、プロジェクトごとの現在及び過去の監査データに対する一元化されたアクセスが可能であるべきである。また、自動化によって詳細なアクセス制御機能を提供し、ビジネス上正当な理由のある人員だけが監査データへのアクセスを認められるようにする必要がある。

各プロジェクトチームに対して、コンプライアンスデータのアクセス方法及びアクセス許可の申請方法に関する手順・手続きを告知する。初期においては、セキュリティ監査員がプロジェクトチームに長めの時間を与えて対応体制を整えさせることが必要な場合があると考えられる。

成果

- コンプライアンス違反によるリスクの受容状況が組織レベルで可視化される
- コンプライアンスについての具体的な保証がプロジェクトレベルで行われる
- 過去のプロジェクトのコンプライアンス実績が正確に追跡管理される
- 人手での作業量を軽減するツールを活用した効率的な監査プロセスが実施される

追加成功指標

- 80%超のプロジェクトが、監査を受ける時点でポリシーと標準に適合している
- 監査 1 件あたりの所要時間が、人手による場合の 50%未満である

追加コスト

- 内部標準に関する監査を自動化するツールの構築またはライセンス
- 監査ゲート及び例外プロセスの継続的な維持管理

追加人員

- 開発者（1 日/年）
- アーキテクト（1 日/年）
- マネージャー（1 日/年）

関連レベル

- 教育&指導 - 3
- コードレビュー - 2
- セキュリティテスト - 2

教育&指導

	EG 1	EG 2	EG 3
目的	**開発要員に、セキュアなプログラミングと配備の話題を扱っている参考情報へのアクセスを提供する**	**ソフトウエアのライフサイクルに関与するすべての人員に対し、セキュアな開発について職務に特化した指導内容の教育を実施する**	**総合的なセキュリティ教育を義務付け、基本水準を満たす知識を備えた人員を認定する**
実施内容	A. 技術的なセキュリティ意識向上トレーニングを実施する B. 技術的な指針を設定し、維持管理する	A. アプリケーションセキュリティについて職務に特化したトレーニングを実施する B. セキュリティコーチを利用してプロジェクトチームを強化する	A. アプリケーションセキュリティをサポートする公式ポータルを作成する B. 職務に応じた試験や認定制度を確立する
査定項目	◆ ほとんどの開発者が概略的なセキュリティ意識向上トレーニングを受講済みである ◆ セキュアな開発に関するベストプラクティスや手引書が、すべてのプロジェクトチームで利用できるようになっている	◆ 開発プロセスにかかわるほとんどの職務に対し、職務に特化したトレーニングや指導が実施される ◆ ほとんどの利害関係者が、プロジェクトのためにセキュリティコーチを招くことができる	◆ セキュリティ関連の手引書が一元管理され、組織全体に一貫した方法で配布される ◆ ほとんどの人員について、セキュアな開発・配備の実践に関する基本水準の技能を持つことがテストにより確認される
成果	◆ コードレベルの非常に一般的な問題について開発者の意識が向上する ◆ 初歩のベストプラクティスが実践されている環境でソフトウエアの維持管理が行われる ◆ 技術要員のセキュリティノウハウについて基本水準が確立される ◆ 基本水準のセキュリティ知識に関する質的なセキュリティチェックが可能になる	◆ 製品・設計・コードの各レベルでセキュリティ脆弱性の原因となる問題が、すべての工程において意識されるようになる ◆ 進行中のプロジェクトに関する脆弱性や設計上の欠陥を修正するための計画が策定される ◆ 要件・設計・開発の各段階に質的なセキュリティチェックポイントを設定できるようになる ◆ セキュリティ問題の理解が深まることで、セキュリティ計画立案時に事前予防がいっそう考慮されやすくなる	◆ 進行中のコードとレガシーコードの両方において、脆弱性の是正作業の効率が上がる ◆ 新種の攻撃や脅威が短期間で理解され、緩和されるようになる ◆ 要員のセキュリティ知識を判定し、一般的な標準に照らして測定できる ◆ セキュリティ意識向上に対する公正なインセンティブが確立する

教育&指導

EG 1

開発要員に、セキュアなプログラミングと配備の話題を扱っている参考情報へのアクセスを提供する

実施内容

A. 技術的なセキュリティ意識向上トレーニングを実施する

技術要員に、アプリケーションセキュリティの基本原理に関するセキュリティトレーニングを内部または外部の手段によって受講させる。一般にこれは、講師の指導する 1～2 日間のトレーニングか、コンピュータ上のトレーニング（開発者各人がほぼ同じ時間をかけてモジュールを学習するもの）によって達成できる。

コース内容は、概念的な情報と技術的な情報の両方を学習できるものとする。たとえば、入力検証、出力エンコーディング、エラー処理、ログの記録、認証、権限に関する概略的なベストプラクティスの話題などがふさわしい。また、開発対象ソフトウエア（Web アプリケーション、組み込みデバイス、クライアントサーバアプリケーション、バックエンドトランザクションシステムなど）に応じた、よくあるソフトウエア脆弱性のトップ 10 を紹介することなども望ましいと考えられる。可能な限り、適用される具体的なプログラミング言語を使用したコード例やラボ実習を含めるようにする。

実施にあたっては、年ごとのセキュリティトレーニングを義務付けた上で、開発人員数に応じた必要な頻度でコース（講師またはコンピュータによる）を開講することが推奨される。

B. 技術的な指針を設定し、維持管理する

開発要員のために、技術面に特化したセキュリティ上のアドバイスが得られる承認済み文書、Web ページ、技術メモのリストを作成する。こうした参考資料は、インターネット上に多数公開されているリソースを基にして揃えることができる。非常に特殊な技術や独自仕様の技術を主体にして開発環境全体が構成されているようなプロジェクトの場合は、職務経験が長くセキュリティに詳しい要員を起用して、徐々にセキュリティノートを整備し、当該環境に特化したナレッジベースを作成する。

管理者は用意したリソースについて認識し、期待されている使用方法を新たな要員に説明することが必要である。混乱及び内容と実態とのかい離を防ぐため、ガイドラインはできるだけ簡易な内容にとどめ、随時改訂して最新情報を反映させることが望ましい。十分なレベルに仕上がったら、これを質的なチェックリストとして利用し、ガイドラインの通読・理解と開発プロセスにおける遵守を徹底させることができる。

成果

- コードレベルの非常に一般的な問題について開発者の意識が向上する
- 初歩のベストプラクティスが実践されている環境でソフトウエアの維持管理が行われる
- 技術要員のセキュリティノウハウについて基本水準が確立される
- 基本水準のセキュリティ知識に関する質的なセキュリティチェックが可能になる。

成功指標

- 直近 1 年以内に、50%超の開発要員に対してセキュリティ問題の概要を説明した
- 直近 1 年以内に、75%超の上級開発者及びアーキテクトに対してセキュリティ問題の概要を説明した
- 初回トレーニングから 3 カ月以内に技術的指針の運用を開始した

コスト

- トレーニングコースの構築またはライセンス
- 技術的指針の継続的な維持管理

人員

- 開発者（1～2 日/年）
- アーキテクト（1～2 日/年）

関連レベル

- ポリシー&コンプライアンス - 2
- セキュリティ要件 - 1
- セキュアなアーキテクチャ - 1

EG 2 教育&指導

ソフトウエアのライフサイクルに関与するすべての人員に対し、セキュアな開発について職務に特化した指導内容の教育を実施する

成果
- 製品･設計･コードの各レベルでセキュリティ脆弱性の原因となる問題が、すべての工程において意識されるようになる
- 進行中のプロジェクトに関する脆弱性や設計上の欠陥を修正するための計画が策定される
- 要件・設計・開発の各段階に質的なセキュリティチェックポイントを設定できるようになる
- セキュリティ問題の理解が深まることで、セキュリティ計画立案時に事前予防がいっそう考慮されやすくなる

追加成功指標
- 直近 1 年以内に、60%超の開発要員に対してトレーニングを実施した
- 直近 1 年以内に、50%超のマネージャー及びアナリストに対してトレーニングを実施した
- 直近 1 年以内に、80%超の上級開発者及びアーキテクトに対してトレーニングを実施した
- トレーニングコースの有用性について、リッカート尺度による評価が 5 段階評価において 3.0 を超える

追加コスト
- トレーニングライブラリの構築またはライセンス
- セキュリティに熟達した要員による実地指導

追加人員
- 開発者（2 日/年）
- アーキテクト（2 日/年）
- マネージャー（1〜2 日/年）
- ビジネスオーナー（1〜2 日/年）
- QA テスト担当者（1〜2 日/年）
- セキュリティ監査員（1〜2 日/年）

関連レベル
- 脆弱性管理 - 1
- 設計レビュー - 2
- セキュアなアーキテクチャ - 2

実施内容

A. アプリケーションセキュリティについて職務に特化したトレーニングを実施する

要員に対し、それぞれの役割の職務におけるアプリケーションセキュリティに着目したセキュリティトレーニングを実施する。一般に、講師の指導する 1〜2 日間のトレーニングか、コンピュータ上のトレーニング（各人がほぼ同じ時間をかけてモジュールを学習するもの）によって達成できる。

マネージャー及び要件設定担当者のために、コース内容にはセキュリティ要件計画、脆弱性及びインシデント管理、脅威モデリング、誤用・悪用ケース設計を含める。

テスト担当者及び監査員に対するトレーニングの主眼は、セキュリティ関連の問題を理解しソフトウエアをより効果的に分析することに関する要員トレーニングである。したがって、その主な内容は、コードレビュー、アーキテクチャ及び設計分析、実行時分析、効果的なセキュリティテスト計画といったものになる。

開発者及びアーキテクトに対する技術トレーニングを拡張して、セキュリティデザインパターン、特定ツールに関するトレーニング、脅威モデリング、ソフトウエア査定技法など、ほかの適切な話題を対象範囲に加える。

実施にあたっては、毎年のセキュリティ意識向上トレーニングと、特定の話題に関する定期的なトレーニングを義務付けることが推奨される。コース（講師またはコンピュータによる）の開講頻度については、役割ごとの人員数に応じて必要な程度を確保する。

B. セキュリティコーチを利用してプロジェクトチームを強化する

内部または外部の専門家を利用して、プロジェクトチームの相談を受けるセキュリティに詳しい要員を確保する。さらに、このコーチ要員を利用できることが要員に知れ渡るよう、存在を組織内に告知する。

コーチ要員を確保するには、組織内の経験豊富な人材を登用し、勤務時間の一部（最大 10%程度）をコーチ活動に割り当てる方法がある。コーチ同士では情報交換によって各員の得意分野を相互に認識させ、疑問点を適切な者に問い合わせて効率よく解決できるようにする。

コーチの利用はソフトウエアライフサイクル内の任意時点に導入可能であるが、適切なタイミングとしては、初期の製品コンセプト確立時や、機能仕様または詳細設計仕様の決定前、開発作業中の問題発生時、テスト計画時、及び運用中のセキュリティインシデント発生時などが考えられる。

やがて、コーチ要員の相互のネットワークを内部に形成し、組織全体にセキュリティ関連情報を発信する場合の連絡窓口として機能できるようになる。また、セキュリティチームを完全に一元化した場合には得られない、進行中のプロジェクトチームによく精通したローカル要員にもなる。

教育&指導

EG 3

総合的なセキュリティ教育を義務付け、基本水準を満たす知識を備えた人員を認定する

実施内容

A. アプリケーションセキュリティをサポートする公式ポータルを作成する

アプリケーションセキュリティの話題に関する文書化された資料を使って、一元的なリポジトリ（普通は内部 Web サイト）を作成し、そのことを告知する。ガイドライン自体は、当該組織にとって合理的であればどのような方法で作成してもよい。ただし、承認委員会と明快な変更管理プロセスを確立することは必須である。

ベストプラクティスのリスト、特定ツールに関するガイド、FAQ、その他の記事を静的コンテンツとして提供するだけでなく、サポートポータルにはメーリングリスト､Web フォーラム､Wiki などのインタラクティブな要素を用意して、セキュリティ関連の話題について内部の要員同士が情報交換することや、後で参照したい情報をカタログ化することを可能にする。

プラットフォーム、プログラミング言語、特定サードパーティ製ライブラリやフレームワークとの関連性、ライフサイクル上の段階など、共通的な要素に基づいて、内容をカタログ化し、検索しやすいようにする。ソフトウエアを作成するプロジェクトチームは、従うことになる具体的なガイドラインに合った体制を製品開発の早期に整える。製品査定においては、該当するガイドラインや製品に関連する議論のリストを監査の基準として使用する。

B. 職務に応じた試験や認定制度を確立する

職務ごと、またはトレーニングクラスやモジュールごとに、セキュリティ知識に関する人員の理解度・応用力を調べる試験を作成し、運用する。正答率 75%程度を合格ラインとして、職務に応じたカリキュラムに基づく試験を作成するのが一般的と考えられる。要員には、年 1 回の適切なトレーニングまたは再教育コースを受講することとし、認定試験は年に 2 回以上受けることを義務付ける。

合否の基準や突出したパフォーマンスに基づいて、要員の階級を判定する。これを利用すると、たとえば、特定の認定レベルに達した人員に一部のセキュリティ関連活動を承認させることができる。具体的には、認定を取得していない開発者が設計から実装へと工程を進める場合は認定済みアーキテクトの明示的な承認を受けさせるといったことが考えられる。すると、プロジェクトごとのセキュリティに関する意思決定が各人の責任能力に応じて行われる状況を細かく把握できるようになる。要するに、アプリケーションセキュリティに関してビジネス上的確な判断を下すための賞罰を行う基盤を作るということである。

成果

- 進行中のコードとレガシーコードの両方において、脆弱性の是正作業の効率が上がる
- 新種の攻撃や脅威が短期間で理解され、緩和されるようになる
- 要員のセキュリティ知識を判定し、一般的な標準に照らして測定できる
- セキュリティ意識向上に対する公正なインセンティブが確立する

追加成功指標

- 直近 1 年以内に、80%超の要員が認定を取得した

追加コスト

- 認定試験の構築またはライセンス
- アプリケーションセキュリティのサポート用ポータルに関する維持管理及び変更管理
- 従業員認定制度の実施による人事コスト及びオーバヘッドコスト

追加人員

- 開発者（1 日/年）
- アーキテクト（1 日/年）
- マネージャー（1 日/年）
- ビジネスオーナー（1 日/年）
- QA テスト担当者（1 日/年）
- セキュリティ監査員（1 日/年）

関連レベル

- ポリシー&コンプライアンス - 2 及び 3

脅威の査定

			
目的	**組織及び個々のプロジェクトにとっての脅威を概要レベルで特定し、理解する**	**脅威の査定の精度を高め、プロジェクト別のきめ細かな理解を進める**	**内製及びサードパーティ製ソフトウエアに存在する個々の脅威に対し、相殺管理策を的確に対応付ける**
実施内容	A. アプリケーションごとに特化した脅威モデルを設定し、維持管理する B. ソフトウエアキテクチャに基づく攻撃者プロファイルを作成する	A. プロジェクトごとの悪用ケースモデルを設定し、維持管理する B. 脅威測定のための重み付けシステムを採用する	A. サードパーティ製コンポーネントからもたらされるリスクを明示的に評価する B. 相殺管理策を盛り込んで脅威モデルの内容を洗練させる
査定項目	◆ 組織内のほとんどのプロジェクトで、直面する可能性が大きい脅威についての検討と文書化が行われている ◆ 直面している攻撃の種類が組織として理解され、文書化されている	◆ プロジェクトチームが、何らかの脅威評価手法を使用して相対的な比較をう ◆ 利害関係者が、該当する脅威及び評価について認識している ◆ プロジェクトチームが、悪用の可能性がないか機能要件を随時分析する	◆ 外部ソフトウエアからもたらされるリスクがプロジェクトチームで明示的に検討されている ◆ あらゆる保護機構や保護策の情報が収集され、脅威への対応付けが行われている
成果	◆ 望ましくない結果をもたらす要因の概略が理解される ◆ 脅威についてプロジェクトチームの意識が向上する ◆ 脅威に関する情報が組織内に蓄積される	◆ 個々のプロジェクトごとに、直面する可能性の大きい脅威が細かく理解される ◆ プロジェクトチームにおけるトレードオフ判断の精度を向上させるフレームワークができる ◆ プロジェクトチーム内の開発作業の優先順位をリスクの重み付けに基づいて設定できるようになる	◆ 各ソフトウエアプロジェクトの脅威プロファイル全体がより深く考慮される ◆ 各ソフトウエアプロジェクトについて、各種のセキュリティ保証機能と確定した脅威との細かい対応付けが行われる ◆ 各ソフトウエアプロジェクトのビジネス機能に基づいてデューディリジェンスを文書化するための資料ができる

脅威の査定

TA 1

組織及び個別プロジェクトにとっての脅威を大まかなレベルで特定し、理解する

実施内容

A. アプリケーションごとに特化した脅威モデルを設定し、維持管理する

各プロジェクトのビジネス上の目的と、ビジネスリスクプロファイル（ある場合）のみに基づき、各プロジェクトチームで開発中のソフトウエアに発生する可能性が大きいワーストケースシナリオを特定する。この作業は、単純な攻撃ツリーあるいは脅威モデリングプロセス（Microsoft の STRIDE、Trike など）を使って実施できる。

攻撃ツリーを作成するには、各ワーストケースシナリオを 1 つの文に記述し、攻撃者の目標を大まかに示すラベルにする。こうして明らかにした個々の目標について、達成のために必要な前提条件を特定する。この情報を、各目標から下に伸びる枝として追加する。1 本の枝は、その下の説明文に対する論理 AND または論理 OR の意味を持つ。AND の枝は、直下にあるすべての子ノードがそれぞれ「真」にならないと親ノードが達成されないことを意味する。OR の枝は、直下にある子ノードのいずれか 1 つが「真」にならないと親ノードが達成されないことを意味する。

使用する脅威モデリング手法にかかわらず、現在及び過去の機能要件を個々にレビューし、それぞれに該当するセキュリティ障害を示せるよう攻撃ツリーに改良を加える。ブレインストーミングで個々のセキュリティ障害シナリオを反復的に分解していき、攻撃者が目標のいずれかを達成しうると考えられる方法をすべて挙げる。アプリケーションの脅威モデルを作成した後、そのソフトウエアに大幅な変更が加えられたときには脅威モデルの内容を更新する。この査定作業は上級開発者、アーキテクト、及び 1 名～複数名のセキュリティ監査員が実施する。

B. ソフトウエアキテクチャに基づく攻撃者プロファイルを作成する

最初に査定を実施し、組織が直面する可能性の高い脅威すべてをソフトウエアプロジェクトに基づいて特定する。この査定においては、悪意のあるエージェントによる脅威だけを検討対象とし、既知の脆弱性や潜在的な弱点などほかのリスクは除外する。

最初に、外部のエージェントとその攻撃の動機について大まかに検討する。このリストに、組織内にあって被害を発生させうる職務と、それら内部の者が攻撃を行う動機を追加する。アーキテクチャやビジネス上の目的が同じアプリケーションは、概してどれも同じような脅威に直面すると考えられる。したがって、検討対象となるソフトウエアプロジェクトのアーキテクチャに基づき、この分析を 1 種類のアーキテクチャについては 1 回だけ実施するようにすると、プロジェクトごとに実施するよりも効率を上げられる可能性がある。

この査定作業はビジネスオーナー及びその他の利害関係者が実施するが、脅威をより多面的に捉えられるように 1 名～複数名のセキュリティ監査員を加えてもよい。最終的な目標は、各種の脅威エージェント及びそれらの者が攻撃を行う動機を簡潔にまとめたリストを作成することである。

成果
- 望ましくない結果をもたらす要因の概略が理解される
- 脅威についてプロジェクトチームの意識が向上する
- 脅威に関する情報が組織内に蓄積される

成功指標
- 直近 12 カ月以内に、50%超のプロジェクト利害関係者に対して、関係するプロジェクトの脅威モデルの概要を説明した
- 75%超のプロジェクト利害関係者に対して、該当するアーキテクチャの攻撃プロファイルの概要を説明した

コスト
- 脅威モデルに関するプロジェクト作成資料の構築と維持管理

人員
- ビジネスオーナー（1 日/年）
- 開発者（1 日/年）
- アーキテクト（1 日/年）
- セキュリティ監査員（2 日/年）
- マネージャー（1 日/年）

関連レベル
- 戦略&指標 - 1
- セキュリティ要件 - 2

TA 2 脅威の査定

脅威の査定の精度を高め、プロジェクト別のきめ細かな理解を進める

成果
- 個々のプロジェクトごとに、直面する可能性の大きい脅威が細かく理解される
- プロジェクトチームにおけるトレードオフ判断の精度を向上させるフレームワークができる
- プロジェクトチーム内の開発作業の優先順位をリスクの重み付けに基づいて設定できるようになる

追加成功指標
- 75%超のプロジェクトチームにおいて脅威の特定と評価が行われた
- 直近 6 カ月以内に、75%超のプロジェクト利害関係者に対して、関係するプロジェクトの脅威・悪用モデルの概要を説明した

追加コスト
- 脅威モデル及び攻撃プロファイルの維持管理によるプロジェクトオーバヘッド

追加人員
- セキュリティ監査員（1 日/年）
- ビジネスオーナー（1 日/年）
- マネージャー（1 日/年）

関連レベル
- 戦略&指標 - 2
- セキュアなアーキテクチャ - 2

実施内容

A. プロジェクトごとの悪用ケースモデルを設定し、維持管理する

組織に対する脅威についての検討をさらに進め、より形式的な分析作業を実施して、機能が誤用・悪用される可能性を判定する。このプロセスは平常時の使用シナリオを特定する作業から始められることが多い。たとえば、ユースケース図があればそれを利用する。

形式的な悪用ケース手法を用いない場合は、各シナリオの悪用ケース群を作成する。これを行うには、通常の使い方を説明する文を基にブレインストーミングを行い、その説明の一部または全部が否定されるような場合を挙げていく。もっとも単純な着手方法は、文中の名詞・動詞の前後などに、できるだけ多くのやり方で「ではない」「しない」といった語句を挿入してみることである。そうして、考えられる悪用ケースの説明文を 1 つの使用シナリオにつき数個程度作成する。

ソフトウエアのビジネス機能に基づいて、アプリケーション固有の考慮事項を盛り込み、悪用ケース説明文の内容を拡充する。最終的な目標は、さまざまな悪用ケース群の説明文を網羅的に揃え、ソフトウエアによって禁止されるべき使用パターンのモデルを形成することである。それらの悪用ケースは、必要に応じて既存の脅威モデルと組み合わせることもできる。

悪用ケースモデルを作成した後、進行中のプロジェクトの場合は、設計フェーズにおいて悪用ケースモデルに修正を加える。既存プロジェクトの場合は、新たな要件を分析して潜在的な悪用の可能性を探す。また、すでに確立した機能の悪用ケース作成作業を、有用な部分について、取り組める範囲で実行する。

B. 脅威測定のための重み付けシステムを採用する

確定した攻撃者プロファイルに基づいて、脅威と脅威との相対的な比較を可能にするための評価システムを決定する。最初はビジネスリスクに基づく単純な「高/中/低」評価システムでよいが、5 段階以内の評価ができればどのような尺度を使っても差し支えない。

評価システムを決定した後は、個々の脅威に評価を付けるための評価基準を策定する。これを適切に行うには、個々の脅威について、動機以外のほかの要因を考慮する必要がある。重要な要因としては、資本及び人的資源、本来のアクセス権限、技術能力、脅威モデル上該当する目標、攻撃成功の可能性などを挙げることができる。

個々の脅威に評価を付けた後は、その情報に基づいて、開発ライフサイクル内のリスク緩和活動に優先順位を設定する。新機能の設計時やリファクタリング作業時には、当該プロジェクトチームにおける優先順位に修正を加える。

脅威の査定

TA 3

内製及びサードパーティ製ソフトウエアに存在する個々の脅威に対し、相殺管理策を的確に対応付ける

実施内容

A. サードパーティ製コンポーネントからもたらされるリスクを明示的に評価する

ソフトウエアコードベースの査定を実施し、外部に由来するコンポーネントをすべて特定する。これには、オープンソースプロジェクトや購入した COTS（商用既製品）ソフトウエアと、組織のソフトウエアで使用しているオンラインサービスなどが含まれる。

サードパーティ製コンポーネントからセキュリティ侵害が発生する潜在的な可能性に基づいて、当該ソフトウエアプロジェクトの攻撃者プロファイルに、特定した個々のコンポーネントに関する情報を追加する。新たに特定した攻撃者プロファイルに基づき、攻撃者の新たな目標や能力によって生じる可能性が大きいリスクを加味してソフトウエア脅威モデルを修正する。

脅威シナリオだけでなく、サードパーティ製ソフトウエアに含まれる脆弱性や設計上の欠陥がどのような形で組織のコードと設計に影響しうるかについても検討する。脆弱性による潜在リスクと、更新した攻撃者プロファイルの知識に従って、脅威モデルの内容を拡充する。

この作業をプロジェクトに対して最初に実施した後は、設計フェーズや毎回の開発サイクルにおいて修正とレビューを行う必要がある。この活動は、セキュリティ監査員と、技術面及びビジネス面の利害関係者が実施する。

B. 相殺管理策を盛り込んで脅威モデルの内容を洗練させる

脅威モデルに表現されているセキュリティ侵害の前提条件の直接の阻害要因となるものを形式的な方法で特定するための査定を実施する。それらの緩和要因は、ソフトウエアに直接起因するリスクに形式的に対応する相殺策である。ソフトウエア自体の技術的な機能だけでなく、開発ライフサイクル、インフラストラクチャ機能などもこの要因になる可能性がある。

攻撃ツリーを使用する場合、個々の枝によって表現される論理的な関係は AND と OR のいずれかである。AND の枝については、前提条件を 1 つだけ緩和すれば親ノードが成立しなくなり、その他すべての葉ノードも緩和されたと見なすことができる。一方、OR の枝については、すべての子ノードを阻害しないと親ノードの攻撃を緩和したことにならない。

相殺策を特定して脅威モデルに直接注釈を加える作業は、使用する脅威モデリング手法にかかわらず実施する。目標は、脅威モデル上で緩和される範囲が最大になるような相殺策を適用することである。攻撃を実行できる経路が残っている場合は、それらすべてに関して相殺策の必要性を指摘し、組織の戦略に反映させる。

この作業をプロジェクトに対して最初に実施した後は、設計フェーズや毎回の開発サイクルにおいて修正とレビューを行う必要がある。この活動は、セキュリティ監査員と、技術面及びビジネス面の利害関係者が実施する。

成果

- 各ソフトウエアプロジェクトの脅威プロファイル全体がより深く考慮される
- 各ソフトウエアプロジェクトについて、各種のセキュリティ保証機能と確定した脅威との細かい対応付けが行われる
- 各ソフトウエアプロジェクトのビジネス機能に基づいてデューディリジェンスを文書化するための資料ができる

追加成功指標

- 80%超のプロジェクトチームが、毎回の実装サイクルに先立って脅威モデルを更新した
- 80%超のプロジェクトチームが、毎回のリリースに先立ってサードパーティ製コンポーネントの調査リストを更新した
- 直近 12 カ月間に発生した全セキュリティインシデントのうち、50%超が脅威モデルによって事前に特定されていた

追加コスト

- 詳細な脅威モデル及び拡張した攻撃プロファイルの維持管理によるプロジェクトオーバヘッド
- すべてのサードパーティ依存箇所の発見

追加人員

- ビジネスオーナー（1 日/年）
- 開発者（1 日/年）
- アーキテクト（1 日/年）
- セキュリティ監査員（2 日/年）
- マネージャー（1 日/年）

関連レベル

- セキュリティ要件 - 2 及び 3

セキュリティ要件

	SR 1	SR 2	SR 3
目的	**ソフトウエア要件の設定作業中に明示的なセキュリティ検討の機会を設ける**	**ビジネスロジックと既知のリスクに基づくセキュリティ要件をより緻密に設定する**	**すべてのソフトウエアプロジェクト及び依存対象サードパーティ要素に対し、セキュリティ要件のプロセス遵守を義務付ける**
実施内容	A. ビジネス機能に基づくセキュリティ要件を明確化する B. セキュリティとコンプライアンスのベストプラクティスを評価し、要件として採用する	A. リソースと機能に関するアクセス制御マトリックスを作成する B. 既知のリスクに基づくセキュリティ要件を策定する	A. サプライヤとの契約にセキュリティ要件を盛り込む B. セキュリティ要件の監査プログラムを拡張する
査定項目	◆ ほとんどのプロジェクトチームが開発時に何らかのセキュリティ要件を策定する ◆ プロジェクトチームが、ベストプラクティス及びコンプライアンス手引書に基づく要件を抽出する	◆ ほとんどの利害関係者が、関係するプロジェクトについてアクセス制御マトリックスを確認する ◆ プロジェクトチームが、ほかのセキュリティ活動から得られたフィードバックに基づいて要件を策定する	◆ ほとんどの利害関係者が、ベンダ契約に盛り込まれるセキュリティ要件を確認する ◆ プロジェクトチームの策定するセキュリティ要件に対して監査が行われる
成果	◆ 開発作業の体制が概略レベルでビジネスリスクと整合する ◆ 業界のセキュリティベストプラクティスが、プロジェクト固有の明示的な要件として取り入れられる ◆ ソフトウエアに由来するリスクを緩和するために行う方策が利害関係者に意識される	◆ ビジネスロジックに対する攻撃シナリオが詳細に理解される ◆ 実行される可能性の大きい攻撃に基づいて、セキュリティ機能に関する開発作業の優先順位が設定される ◆ 機能とセキュリティ対策のトレードオフに関する意思決定が、より充実した知識を背景に行われるようになる ◆ 利害関係者が、本質的にセキュリティ上の欠陥を抱えた機能要件を避けやすくなる	◆ 外部コードに関するセキュリティ期待の基本水準が公式に設定される ◆ 各プロジェクトチームで行われるセキュリティ対策についての情報を一元的に把握できる ◆ アプリケーションリスク及び希望するセキュリティ要件に基づいてリソースをプロジェクトに整合させることが可能になる

セキュリティ要件

SR 1

ソフトウエア要件の設定作業中に明示的なセキュリティ検討の機会を設ける

実施内容

A. ビジネス機能に基づくセキュリティ要件を明確化する

個々のソフトウエアプロジェクトについて、ビジネスロジック及び全体的な動作を指定する機能要件のレビューを実施する。プロジェクトの要件を収集した後に、査定を実施して、該当するセキュリティ要件を明確化する。ソフトウエアの構築作業をサードパーティに行わせる場合であっても、セキュリティ要件を決定した後は、ベンダに伝える機能要件にそれらを盛り込む。

セキュリティ監査員は、セキュリティに関するすべての期待事項を明示的に利害関係者へと伝えるプロセスを、個々の機能要件について実行する。各要件を明確化するためのポイントとして、データセキュリティ、アクセス制御、トランザクションの完全性、ビジネス機能の重大度、職務分掌、稼働時間などに関する期待を示すのが一般的である。

すべてのセキュリティ要件が、適切な要件を作成するための一般原則に従っていることが重要である。つまり、具体的であり、評価可能であり、合理的であることが求められる。

このプロセスを、進行中のプロジェクトに関するすべての新たな要件について実施する。既存機能については、同じプロセスでギャップ分析を実施し、セキュリティに関するリファクタリング作業を将来実行する際の参考とする。

B. セキュリティとコンプライアンスのベストプラクティスを評価し、要件として採用する

業界のベストプラクティスから、プロジェクトチームの要件として扱うべきものを選定する。評価の対象となるのは、公開されているガイドライン、内部または外部のガイドライン/標準/ポリシー、策定されたコンプライアンス要件などである。

設計と実装の間には時間的なトレードオフが存在するので、開発の各サイクルにはあまり多くのベストプラクティス要件を持ち込まないことが重要である。開発サイクルごとに少しずつベストプラクティスを追加し、当該ソフトウエアの全体的なセキュリティ保証プロファイルを時間の経過とともに強化していく方法が推奨される。

既存システムの場合、セキュリティベストプラクティスのためのリファクタリングは複雑な作業になる可能性がある。したがって、機能追加の際に対応できる範囲でセキュリティ要件を追加していく形をとる。最低限、どのベストプラクティスが該当するかを確認する分析作業を実施して将来計画の参考とする。

このレビューは、セキュリティ監査員が、ビジネス利害関係者から情報を得て実施する。また、設計・実装に特有の知識が意思決定プロセスに反映されるよう、上級開発者、アーキテクト、その他の技術的な利害関係者も関与すべきである。

成果

- 開発作業の体制が概略レベルでビジネスリスクと整合する
- 業界のセキュリティベストプラクティスが、プロジェクトに特有の明示的な要件として取り入れられる
- ソフトウエアに由来するリスクを緩和するために行う方策が利害関係者に意識される

成功指標

- 50%超のプロジェクトチームが、明示的に定義されたセキュリティ要件を採用した

コスト

- 毎回の開発サイクルに対するセキュリティ要件追加によるプロジェクトオーバヘッド

人員

- セキュリティ監査員（2 日/年）
- ビジネスオーナー（1 日/年）
- マネージャー（1 日/年）
- アーキテクト（1 日/年）

関連レベル

- 教育&指導 - 1
- ポリシー&コンプライアンス - 2
- 設計レビュー - 1
- コードレビュー - 1
- セキュリティテスト - 1

SR 2 セキュリティ要件

ビジネスロジックと既知のリスクに基づくセキュリティ要件をより緻密に設定する

成果
- ビジネスロジックに対する攻撃シナリオが詳細に理解される
- 実行される可能性の大きい攻撃に基づいて、セキュリティ機能に関する開発作業の優先順位が設定される
- 機能とセキュリティ対策のトレードオフに関する意思決定が、より充実した知識を背景に行われるようになる
- 利害関係者が、本質的にセキュリティ上の欠陥を抱えた機能要件を避けやすくなる

追加成功指標
- 直近 6 カ月以内に、全プロジェクトの 75%超において悪用ケースモデルを更新した

追加コスト
- 悪用ケースモデルの構築及び維持管理によるプロジェクトオーバヘッド

追加人員
- セキュリティ監査員（2 日/年）
- マネージャー（1 日/年）
- アーキテクト（2 日/年）
- ビジネスオーナー（1 日/年）

関連レベル
- 脅威の査定 - 1 及び 3
- 戦略&指標 - 1

実施内容

A. リソースと機能に関するアクセス制御マトリックスを作成する

アプリケーションのビジネス上の目的に基づいて、ユーザ及びオペレータの役割を明確にする。また、何らかのアクセス制御による保護の対象となるデータ資産及びアプリケーション特有の機能をすべて抽出し、リソース及び機能のリストを作成する。

役割の軸とリソースの軸を持つ単純なマトリックスを使って、個々の役割と個々のリソースとの関係を検討し、利害関係者によるアクセス制御の観点から見た正しいシステム動作を各交点に記入する。

データリソースについては、作成、読み取りアクセス、更新、削除の各操作に必要なアクセス権限を記入することが重要である。機能的なリソースについては、アクセス権限の分布はアプリケーションに特有の形になる可能性が大きいが、どの機能に対するアクセスがどの役割に許可されるかを少なくとも記入する。

このアクセス許可マトリックスは、システム全体のビジネスロジックに関する正しいアクセス制御権限を文書化した資料として機能することになる。したがって、作成作業は、ビジネス利害関係者から情報を得てプロジェクトチームが行う。作成後は毎回のリリースに先立ってビジネス利害関係者が内容を更新するが、修正は設計フェーズの開始に先立って行うのが一般的である。

B. 既知のリスクに基づくセキュリティ要件を策定する

当該ソフトウエアのリスクプロファイル全体をよりよく理解するために、組織またはプロジェクトに特有のセキュリティリスクを示した既存の資料を明示的にレビューする。概要レベルのビジネスリスクプロファイル、アプリケーション個別の脅威モデル、設計レビュー/コードレビュー/セキュリティテストの発見内容などがある場合は、これらのリソースを利用する。

既存の資料をレビューするだけでなく、悪用シナリオを直接・間接に緩和する具体的なセキュリティ要件を特定するための参考として、アプリケーションの悪用ケースモデルを使用する。

このプロセスは、ビジネスオーナー及びセキュリティ監査員が必要に応じて実施する。新たなセキュリティ要件の設定につながるリスクを認識する活動は、究極的には計画フェーズ内の手順として組み込まれるべきである。その手順において、新しく発見されたリスクはプロジェクトチームによる詳細な査定を受ける。

セキュリティ要件

すべてのソフトウエアプロジェクト及び依存対象サードパーティ要素に対し、セキュリティ要件のプロセス遵守を義務付ける

実施内容

A. サプライヤとの契約にセキュリティ要件を盛り込む

サードパーティとの契約によって、過去の分析作業で特定した各種セキュリティ要件とは別のセキュリティ上のメリットを享受する方法がある。その場合、要件（及び場合によって概要設計）は内部で作成し、詳細設計及び実装をサプライヤに担当させることが多い。

開発を外注するコンポーネントごとの作業分担に基づいて、ベンダ契約に盛り込む具体的なセキュリティ活動及び技術査定基準を特定する。その内容は、設計レビュー、コードレビュー、セキュリティテストの各「対策」に関する実施内容を組み合わせたものになるのが一般的である。

要件の追加は一般にコスト増を伴うため、契約文言の修正はサプライヤごとに個別処理すべきである。発生しうるセキュリティ活動のコストは、当該コンポーネントまたはシステムの使用方法に応じて、実施内容のメリットとのバランスにより検討する必要がある。

B. セキュリティ要件の監査プログラムを拡張する

セキュリティ要件の完成度チェックを、定期的なプロジェクト監査作業の中に組み込む。これを行うにはプロジェクト固有の知識がないと難しい場合があるので、監査においては、適切な種類の分析が実施されたことの証跡となる要件定義文書や設計文書など、プロジェクト成果物資料のチェックを重視する。

とりわけ、個々の機能要件に対しては、ビジネス推進要因及び想定される悪用シナリオに基づく注釈が付けられているべきである。全体のプロジェクト要件には、ガイドラインや標準に含まれているベストプラクティスに基づいて作成された要件のリストが含まれていなければならない。また、満たせないセキュリティ要件及び将来リリースにおける解決見込みスケジュールを明記したリストも必要である。

この監査は、開発サイクルごとに実施する。タイミングは要件定義プロセスの前が最適であり、リリース以前に実行することが必須である。

成果

- 外部コードに関するセキュリティ期待の基本水準が公式に設定される
- 各プロジェクトチームで行われるセキュリティ対策についての情報を一元的に把握できる
- アプリケーションリスク及び希望するセキュリティ要件に基づいてリソースをプロジェクトに整合させることが可能になる

追加成功指標

- 直近 6 カ月以内に、80%超のプロジェクトがセキュリティ要件監査に合格した
- 直近 12 カ月以内に、80%超のベンダ契約について契約上のセキュリティ要件の分析を実施した

追加コスト

- 外注開発のセキュリティ要件追加によるコスト増
- セキュリティ要件監査のリリースゲートによる継続的なプロジェクトオーバヘッド

追加人員

- セキュリティ監査員（2 日/年）
- マネージャー（2 日/年）
- ビジネスオーナー（1 日/年）

関連レベル

- 脅威の査定 – 3
- ポリシー&コンプライアンス - 2

セキュアなアーキテクチャ

	SA 1	SA 2	SA 3
目的	**事前予防的なセキュリティ指導についての検討機会をソフトウエア設計プロセスに含める**	**既知のセキュアサービスと「デフォルト設定でセキュア」な設計を採用する方向にソフトウエア設計プロセスを誘導する**	**ソフトウエア設計プロセスを公式に管理し、セキュアなコンポーネントの利用について認可制度を実施する**
実施内容	A. 推奨ソフトウエアフレームワークのリストを維持管理する B. セキュリティに関する原則を設計に対して明示的に適用する	A. セキュリティに関するサービスやインフラストラクチャを具体的に指定し、使用を奨励する B. セキュリティに関するデザインパターンをアーキテクチャに基づいて指定する	A. 公式なリファレンスアーキテクチャとリファレンスプラットフォームを確立する B. フレームワーク、パターン、プラットフォームの使用について認可制度を実施する
査定項目	◆ 推奨されるサードパーティ製コンポーネントのリストがプロジェクトチームに提供される ◆ ほとんどのプロジェクトチームがセキュアな設計の原則を認識し、適用する	◆ 共有のセキュリティサービスに関する情報が、手引きでプロジェクトチームに周知されている ◆ プロジェクトチームに対し、該当するアプリケーションのアーキテクチャに基づく規範的なデザインパターンが提供される	◆ プロジェクトチームでのソフトウエア構築が、一元管理されたプラットフォームやフレームワークを使って行われる ◆ プロジェクトチームに対し、セキュアなアーキテクチャ構成要素の使用に関する監査が行われる
成果	◆ 想定外の依存性や、実装上の1回限りの選択が発生することを、プロジェクトに特有の方法で予防できる ◆ ライブラリやフレームワークの選択によってプロジェクトのリスクが増大することを利害関係者が認識する ◆ 開発作業時、事前予防的なセキュリティメカニズムを設計に含めるためのプロトコルが確立する	◆ 資産とユーザ役割との詳細な対応付けが定義され、設計の区画化が促進される ◆ セキュリティの保護や機能を提供する再利用可能な設計構築部品ができる ◆ セキュリティに関する確立された設計手法を使用することで、ソフトウエアプロジェクトの信頼度が向上する	◆ セキュリティ保護を組み込んだカスタマイズ済みアプリケーション開発プラットフォームができる ◆ 開発時に事前予防的なセキュリティ対策が行われることを組織全体が期待するようになる ◆ セキュアな設計に対するビジネス上のニーズに基づくトレードオフの意思決定を、利害関係者がより的確に行えるようになる

セキュアなアーキテクチャ

SA 1

事前予防的なセキュリティ指導についての検討機会をソフトウエア設計プロセスに含める

実施内容

A. 推奨ソフトウエアフレームワークのリストを維持管理する

組織内の各種ソフトウエアプロジェクトで共通的に使われているサードパーティ製ソフトウエアライブラリやフレームワークを明らかにする。一般に、これを行うために依存性の網羅的な調査を行う必要はなく、よく使われている高水準コンポーネントに着目する。

サードパーティ製コンポーネントの提供する主要機能に基づいて、リスト内のコンポーネントを機能ごとに分類する。また、プロジェクトチームで各コンポーネントがどれだけ広く使用されているかに着目し、サードパーティ製コードに対する依存度の重み付けを行う。この重み付けリストを参考にして、開発組織全体に推奨コンポーネントとして告知するコンポーネントのリストを作成する。

推奨リストに含めるかどうかを判断する際には、いくつかの要因を考慮する。この目的に特化した調査を実施しなくてもリストの作成は可能であるが、個々のサードパーティ製コンポーネントについて、インシデント発生履歴、脆弱性への対応実績、当該組織に対する機能の適合性、また、使用方法に特別複雑な点があるかどうかなどを調べることが望ましい。

このリスト作成作業は、上級開発者及びアーキテクトが、マネージャー及びセキュリティ監査員から情報を得て実施する。作成後、この推奨リストと機能分類とを対応付けて、開発組織内に告知する。究極的な目標は、すべてのプロジェクトチームのために既知のデフォルトを提供することである。

B. セキュリティに関する原則を設計に対して明示的に適用する

設計作業の際、プロジェクトチームの技術要員は、目安となるセキュリティ原則の簡潔なリストを、詳細なシステム設計に対するチェックリストとして使用する。このリストに含まれる原則としては、多重防御、最弱リンクの保護、セキュアなデフォルトの使用、単純なセキュリティ機能設計、セキュアな障害時動作、セキュリティとユーザビリティのバランス、最小権限による実行、隠蔽によるセキュリティの不使用、などが考えられる。

特に周縁部インターフェイスに関して、設計チームは個々の原則をシステム全体の文脈において考慮し、どのような機能を追加するとこの種のインターフェイス個々のセキュリティを補強できるかを明らかにする。総じて、機能追加は、機能要件を通常の形で実装する場合のコストを若干上回る程度の作業で実現できるものにとどめるべきである。この範囲を超える内容については、将来リリースの計画に盛り込む事項として記録に残す。

このプロセスは、各プロジェクトチームがセキュリティ意識向上トレーニングを受けた上で実施する。セキュリティに精通した要員が加わって設計上の意思決定を助けると効果的である。

成果

- 想定外の依存性や、実装上の１回限りの選択が発生することを、プロジェクトに特有の方法で予防できる
- ライブラリやフレームワークの選択によってプロジェクトのリスクが増大することを利害関係者が認識する
- 開発作業時、事前予防的なセキュリティメカニズムを設計に含めるためのプロトコルが確立する

成功指標

- 直近１年以内に、80%超の開発要員に対してソフトウエアフレームワークに関する推奨事項の概要を説明した
- 50%超のプロジェクトが、設計にセキュリティ原則を適用した旨を自主的に報告した

コスト

- ソフトウエアフレームワークに関する推奨事項の構築、維持管理、意識形成
- セキュリティ原則の分析及び適用による継続的なプロジェクトオーバヘッド

人員

- アーキテクト（2～4 日/年）
- 開発者（2～4 日/年）
- セキュリティ監査員（2～4 日/年）
- マネージャー（2 日/年）

関連レベル

- 教育&指導 - 1

SA 2 セキュアなアーキテクチャ

既知のセキュアサービスと「デフォルト設定でセキュア」な設計を採用する方向にソフトウエア設計プロセスを誘導する

成果
- 資産とユーザ役割との詳細な対応付けが定義され、設計の区画化が促進される
- セキュリティの保護や機能を提供する再利用可能な設計構築部品ができる
- セキュリティに関する確立された設計手法を使用することで、ソフトウエアプロジェクトの信頼度が向上する

追加成功指標
- 直近 6 カ月以内に、80%超のプロジェクトがアクセス許可マトリックスを更新した
- 直近 6 カ月以内に、適用可能なセキュリティパターンの概要を 80%超のプロジェクトチームに説明した

追加コスト
- 適用可能なセキュリティパターンの構築またはライセンス
- アクセス許可マトリックスの維持管理による継続的なプロジェクトオーバヘッド

追加人員
- アーキテクト（2〜4 日/年）
- 開発者（1〜2 日/年）
- マネージャー（1〜2 日/年）
- ビジネスオーナー（1 日/年）
- セキュリティ監査員（1〜2 日/年）

関連レベル
- 教育&指導 - 1

実施内容

A. セキュリティに関するサービスやインフラストラクチャを具体的に指定し、使用を奨励する

組織は、セキュリティ機能を提供する共有インフラストラクチャやサービスの存在を明らかにすべきである。これに該当するものとしては、シングルサインオンサービス、企業内ディレクトリシステム、アクセス制御または資格付与サービス、認証システムなどがある。再利用可能なシステムを収集・評価して、そのようなリソースのリストをまとめ、提供されるセキュリティメカニズムに応じて分類する。また、開発チームが共有リソースを統合したいと考える理由、つまり使用するメリットは何かという観点から各リソースを検討することも有用である。

同じ分類に属するリソースが複数ある場合は、分類ごとに 1 つまたは複数の共有サービスを選び出し、組織としての標準を決定する。以後は選択したサービスに依存してソフトウエア開発を行うので、個々のリソースについては徹底した監査を実施し、セキュリティ状況の基本水準を確実に把握しておく。選択した個々のサービスについて、システムへの統合方法を開発チームに理解させるための設計ガイドを作成する。その後、トレーニング、指導、ガイドライン、標準を通して開発チームが設計ガイドを利用できるようにする。

この方法のメリットは、セキュアであると確認されているシステムを利用できること、プロジェクト設計チーム向けのセキュリティ指導を単純化できること、共有セキュリティサービスを利用するアプリケーションの周辺にセキュリティ保証を確立する手順がより明快になることなどである。

B. セキュリティに関するデザインパターンをアーキテクチャに基づいて指定する

組織内の各種ソフトウエアプロジェクトを、アーキテクチャの一般的な種類によって分類する。分類としては、クライアントサーバアプリケーション、組み込みシステム、デスクトップアプリケーション、Web に対するインターフェイスを備えたアプリケーション、Web サービスプラットフォーム、トランザクションミドルウェアシステム、メインフレームアプリケーションなどがある。組織のビジネス分野によっては、言語、プロセッサアーキテクチャ、あるいは配備時期などに基づいて、より細かな分類を定義することが必要な場合もある。

ソフトウエアキテクチャの大まかな種類に対して、堅実なセキュリティ機能実装方法を表す一般的なデザインパターン群を設定し、組織のソフトウエアプロジェクト個別の設計に適用する。これらのセキュリティデザインパターンは、研究や購入が可能な汎用設計要素の一般的な定義を表している。組織に合わせてカスタマイズするとさらに効果的なものになることが多い。パターンの例としては、シングルサインオンサブシステム、階層間の委任モデル、セキュリティ強化されたインターフェイス設計、職務分掌に基づく権限モデル、ログの一元記録パターンなどが考えられる。

適用可能かつ適切なパターンを特定するプロセスは、アーキテクト、上級開発者、その他の技術的な利害関係者が、設計フェーズにおいて実施する。

セキュアなアーキテクチャ

SA 3

ソフトウエア設計プロセスを公式に管理し、セキュアなコンポーネントの利用について認可制度を実施する

実施内容

A. 公式なリファレンスアーキテクチャとリファレンスプラットフォームを確立する

共有セキュリティサービスの統合を促し、アーキテクチャの種類ごとに特化したセキュリティパターンを揃えた後は、それらの機能を実装したコード集をプロジェクトチームから選び出し、共有コードベースの基礎として使用する。この共有コードベースは、最初は各プロジェクトで共通に必要となる推奨ライブラリ群として提供を始め、しだいに内容を拡充して、プロジェクトチームによるソフトウエア構築作業のリファレンスプラットフォームとなる 1 つまたは複数のソフトウエアフレームワークとして整備していくとよい。たとえば、モデル/ビュー/コントローラ型 Web アプリケーション用フレームワーク、トランザクションバックエンドシステム用サポートライブラリ、Web サービスプラットフォーム用フレームワーク、クライアントサーバアプリケーションの基礎、プラグ可能ビジネスロジック用ミドルウェアなどのリファレンスプラットフォームを提供することが考えられる。

初期のリファレンスプラットフォームを構築するもう 1 つの方法として、ライフサイクルの早期段階にある特定のプロジェクトを選び、セキュリティに精通した要員と共同で汎用的なセキュリティ機能を構築させる方法がある。その機能をプロジェクトから抽出し、組織内の別のプロジェクトで利用できるようにするのである。

どのような手法で作成するかにかかわらず、リファレンスプラットフォームには、監査及びセキュリティ関連レビューの迅速化、開発効率の向上、維持管理オーバヘッドの低減といったメリットがある。

リファレンスプラットフォームの設計・作成には、アーキテクト、上級開発者、その他の技術的な利害関係者が関与する。作成後には、継続的なサポート及び更新活動をチームが維持する必要がある。

B. フレームワーク、パターン、プラットフォームの使用について認可制度を実施する

定期的なプロジェクト監査の際に、付加的なプロジェクト成果物資料分析作業を実施し、推奨フレームワーク、デザインパターン、共有セキュリティサービス、リファレンスプラットフォームの利用状況を測定する。定期的な監査作業の中で実施するとはいえ、この活動の目標は、プロジェクトチームからのフィードバックを収集して個別的な事前予防セキュリティ対策の取り組み状況を調べることである。

全体として、プロジェクトチームに関するいくつかの要素を確認することが重要である。推奨しないフレームワークの使用状況を特定し、推奨事項と組織内の機能ニーズとの間にギャップがないかどうかを確認する。デザインパターンやリファレンスプラットフォームモジュールの不使用または誤用について調べ、更新が必要かどうかを確認する。また、組織が変化するとリファレンスプラットフォームに対するニーズも変化し、プロジェクトチームが、より充実した機能または別の機能を実装したプラットフォームを必要とするようになる可能性がある。

この分析作業は、セキュリティに精通した要員であれば誰でも実施できる。各プロジェクトから収集した尺度は、照合され、マネージャー及び利害関係者による分析の対象となる。

成果

- セキュリティ保護を組み込んだカスタマイズ済みアプリケーション開発プラットフォームができる
- 開発時に事前予防的なセキュリティ対策が行われることを組織全体が期待するようになる
- セキュアな設計に対するビジネス上のニーズに基づくトレードオフの意思決定を、利害関係者がより的確に行えるようになる

追加成功指標

- 進行中のプロジェクトの 50%超がリファレンスプラットフォームを使用した
- 直近 6 カ月以内に、80%超のプロジェクトからフレームワーク、パターン、プラットフォームの使用状況に関するフィードバックを得た
- ガイドやプラットフォームの有用性に関するプロジェクトチームからの報告で、リッカート尺度による評価が 5 段評価において 3.0 を超えた

追加コスト

- リファレンスプラットフォームの構築またはライセンス
- リファレンスプラットフォームの継続的な維持管理及びサポート
- 監査時に実施する使用状況検証作業による継続的なプロジェクトオーバヘッド

追加人員

- マネージャー（1 日/年）
- ビジネスオーナー（1 日/年）
- アーキテクト（3～4 日/年）
- 開発者（2～3 日/年）
- セキュリティ監査員（2 日/年）

関連レベル

- ポリシー&コンプライアンス - 2
- 設計レビュー - 3
- コードレビュー - 3
- セキュリティテスト - 3

設計レビュー

			
目的	**ソフトウエア設計に対する当座のレビューをサポートし、既知のリスクに対して基本水準の緩和策を確実に適用する**	**総合的なセキュリティベストプラクティスに照らした、ソフトウエア設計レビューのための査定サービスを提供する**	**査定を義務付け、成果物の認可制度を実施して、保護機構についての詳しい理解を広める**
実施内容	A. ソフトウエアが攻撃にさらされる箇所を特定する B. 既知のセキュリティ要件に照らして設計を分析する	A. セキュリティの仕組みが完備されているかどうかを検査する B. プロジェクトチーム向け設計レビューサービスを配備する	A. 扱いに注意を要するリソースについてのデータフロー図を作成する B. 設計レビューを行うためのリリースゲートを設定する
査定項目	◆ ソフトウエアの設計上、攻撃者の侵入口となりうる箇所についての情報が、プロジェクトチームによって文書化されている ◆ ソフトウエア設計が、プロジェクトチームにより既知のセキュリティリスクに照らしてチェックされている	◆ ほとんどのプロジェクトチームが、設計要素に対し、セキュリティメカニズムに関する分析作業を明示的に行っている ◆ ほとんどのプロジェクト利害関係者が、公式の設計レビューを受ける方法を認識している	◆ 設計レビューのプロセスに詳細なデータレベルの分析作業が組み込まれている ◆ 定期的なプロジェクト監査において、設計レビューの結果が一定の基本水準を満たすことが義務付けられている
成果	◆ I/OやI/Fなど周辺のアーキテクチャがセキュリティ上どのような意味を持つかが大まかに理解される ◆ 開発チームが設計に対してセキュリティベストプラクティスの自己チェックを実行できるようになる ◆ プロジェクトレベルの設計レビューを実施するための簡易なプロセスが確立する	◆ セキュリティに関するアーキテクチャレビューを一貫した方法で実施するための公式な査定サービスが提供される ◆ 維持モードに入ったシステムやレガシーシステム内にあるセキュリティ上の欠陥が正確に特定される ◆ 当該ソフトウエアにおいてセキュリティ保証的な保護がどのように提供されているかについて、プロジェクト利害関係者の理解が深まる	◆ システム設計上の弱点を細かく把握できるようになり、区画化が促進される ◆ アーキテクチャに対して期待されるセキュリティ基本水準に照らしたプロジェクトの現在位置が組織レベルで意識される ◆ 既知の弱点の緩和について、効率性や進捗状況をプロジェクト間で比較できるようになる

設計レビュー

ソフトウエア設計に対する当座のレビューをサポートし、既知のリスクに対して基本水準の緩和策を確実に適用する

実施内容

A. ソフトウエアが攻撃にさらされる箇所を特定する

個々のソフトウエアプロジェクトについて、アーキテクチャ全体を簡単に見渡せる概略図を作成する。これは、概略的な要件定義文書や設計文書、技術要員に対する聞き取り調査、モジュールレベルでのコードベースレビューなど、各種のプロジェクト資料に基づいて作成するのが一般的である。システムを構成する大まかなモジュールを把握することが重要であり、レビュー対象システム全体が 1 ページの概略図に収まる程度が最適な詳細度の目安と考えられる。

アーキテクチャ概略図に基づいて、承認済みユーザ、匿名ユーザ、オペレータ、アプリケーションに特有の各種役割などによるインターフェイスへのアクセス性の観点から、個々のコンポーネントを分析する。また、インターフェイスを提供する各種コンポーネントを概略図の文脈において検討し、図中にあるほかのコンポーネントに対して機能の委任やデータの引き渡しが行われるポイントを発見する。アクセス性の特性が類似しているインターフェイスやコンポーネントをグループにまとめ、それらのグループを当該ソフトウエアの「攻撃対象領域」として捉える。

概略図に示された個々のインターフェイスに、セキュリティ関連機能の情報を追加する。攻撃対象領域を構成するインターフェイスグループの情報に基づいてモデルをチェックし、類似のアクセス特性を持つ各種インターフェイスのセキュリティ確保の方法が設計レベルで一貫性があるかどうかを確認する。一貫性が破綻している部分があれば、査定の発見内容としてすべて記録する。

この分析作業は、プロジェクトチーム内部または外部の、セキュリティに精通した要員が実施する。作成した図と攻撃対象領域の情報は、普通、システム外縁のインターフェイスに機能追加や変更が生じた場合に限り、設計フェーズにおいて更新する必要がある。

B. 既知のセキュリティ要件に照らして設計を分析する

セキュリティ要件（公式に確認されたもの、または非公式に知られたもの）を特定し、収集する。また、システムの安全な運用の前提となっているセキュリティ上の想定条件があれば特定し、収集した情報に加える。

既知のセキュリティ要件のリストに含まれる個々の項目を、システムアーキテクチャ概略図に照らしてレビューする。個々のセキュリティ要件に対応する設計レベルの機能に関する情報を概略図に追加する。システムが大規模または複雑である場合は、この情報収集作業を実行しやすいように、より詳細な図を別途作成してもよい。全体的な目標は、既知のセキュリティ要件すべてについてシステム設計上の対策が行われていることを確認することである。設計レベルで明確な対策がなされていないセキュリティ要件があれば、査定の発見内容としてすべて記録する。

この分析作業は、セキュリティに精通した技術要員が、アーキテクト、開発者、マネージャー、ビジネスオーナーから情報を得て実施する。作成した情報は、セキュリティ要件またはシステム概略設計に変更が生じた場合、設計フェーズにおいて更新する必要がある。

成果
- 周縁部のアーキテクチャがセキュリティ上どのような意味を持つかが大まかに理解される
- 開発チームが設計に対してセキュリティベストプラクティスの自己チェックを実行できるようになる
- プロジェクトレベルの設計レビューを実施するための簡易なプロセスが確立する

成功指標
- 直近 12 カ月以内に、50%超のプロジェクトが攻撃対象領域の分析情報を更新した
- 直近 12 カ月以内に、50%超のプロジェクトがセキュリティ要件の設計レベル分析情報を更新した

コスト
- 各プロジェクトに関するアーキテクチャ概略図の構築と維持管理
- 攻撃対象領域及びセキュリティ要件についての設計検査による継続的なプロジェクトオーバヘッド

人員
- アーキテクト（2〜3 日/年）
- 開発者（1〜2 日/年）
- マネージャー（1 日/年）
- セキュリティ監査員（1 日/年）

関連レベル
- セキュリティ要件 - 1

設計レビュー

総合的なセキュリティベストプラクティスに照らした、ソフトウエア設計レビューのための査定サービスを提供する

成果
- セキュリティに関するアーキテクチャレビューを一貫した方法で実施するための公式な査定サービスが提供される
- 維持モードに入ったシステムやレガシーシステム内にあるセキュリティ上の欠陥が正確に特定される
- 当該ソフトウエアにおいてセキュリティ保証的な保護がどのように提供されているかについて、プロジェクト利害関係者の理解が深まる

追加成功指標
- 直近 6 カ月以内に、80%超の利害関係者に対してレビュー要求状況の概要を説明した
- 直近 12 カ月以内に、75%超のプロジェクトについて設計レビューを実施した

追加コスト
- 設計レビューチームの構築、トレーニング、維持管理
- レビュー活動による継続的なプロジェクトオーバヘッド

追加人員
- アーキテクト（1〜2 日/年）
- 開発者（1 日/年）
- マネージャー（1 日/年）
- セキュリティ監査員（2〜3 日/年）

関連レベル
- 教育&指導 - 2
- 戦略&指標 - 2

実施内容

A. セキュリティの仕組みが完備されているかどうかを検査する

アーキテクチャ概略図上のモジュールに示された個々のインターフェイスについて、システムを分析し、セキュリティメカニズムのリストに記載したすべての項目の対策が盛り込まれているかどうかを確認する。この種の分析作業は、内部インターフェイス（階層間など）と外部インターフェイス（攻撃対象領域を構成するインターフェイスなど）の両方について実行すべきである。

検討を要する主要なセキュリティメカニズムは、認証、権限付与、入力検証、出力エンコーディング、エラー処理、ログ記録の 6 つである。また、該当する場合は暗号化及びセッション管理のメカニズムについても検討する。個々のインターフェイスについて、システム設計内のどこで各セキュリティメカニズムが提供されているかを確認し、機能に不足または不明確な点があれば発見内容として記録する。

この分析作業は、セキュリティに精通した要員が、アプリケーションに特有の知識をプロジェクトチームから得て実施する。この分析作業は、リリースごとに 1 回、普通は設計フェーズの終了前に実行する。初回の分析を行った後は、変更が生じる場合に、その内容に応じて開発サイクル内で情報を更新する必要がある。

B. プロジェクトチーム向け設計レビューサービスを配備する

プロジェクト利害関係者が設計レビューの実施を要求できるプロセスを策定する。このサービスは組織として一元的に提供してもよいし、既存要員に分散して提供してもよい。ただし、完全かつ一貫したレビュー実行方法のトレーニングをすべてのレビュー担当者に施すことが必須である。

レビューサービスは一元的に管理され、レビュー要求キューに並ぶ要求は、上級マネージャー、アーキテクト、及び、当該組織のビジネスリスクプロファイル全体に詳しい利害関係者によってトリアージが行われる。これにより、全体的なビジネスリスクと整合する形でプロジェクトレビューの優先順位が設定される。

設計レビューにおいて、レビューチームは、攻撃対象領域についての理解を形成し、プロジェクトに特有のセキュリティ要件と設計要素とを対応付け、モジュールインターフェイスのセキュリティメカニズムを確認するために必要な情報を、プロジェクトチームの協力の下で収集する必要がある。

設計レビュー

査定を義務付け、成果物の認可制度を実施して、保護機構についての詳しい理解を広める

実施内容

A. 扱いに注意を要するリソースについてのデータフロー図を作成する

ソフトウエアプロジェクトのビジネス上の機能に基づいて、高リスクの機能に関連したシステム動作を詳細に確認するための分析作業を実施する。高リスクの機能は、機密性のあるデータの作成、アクセス、更新、削除操作を実装した機能と関連していることが多い。また、データ以外に関しては、プロジェクトに特有で本質的に重大な性格を持つビジネスロジック（DoS 攻撃や侵入が発生した場合の被害が甚大であるもの）も高リスクの機能に含まれる。

特定された個々のデータまたはビジネス機能について、該当するソフトウエアモジュール、データソース、アクター、及びこれら相互の間でやり取りされるメッセージに関する情報を、統一された表記方法によって記録する。多くの場合、まず設計概略図のレベルから始めて、必要な要素については反復的に詳細事項を検討していき、機密性のリソースに関連しない要素を除外していくと作業しやすい。

プロジェクトのために作成したデータフロー図を基にして分析作業を実施し、内部に存在する設計上の隘路を発見する。設計上の隘路は、扱うデータの機密性レベルが一様でないソフトウエアモジュールや、ビジネス上の重要度が一様でない多数のビジネス機能に対するアクセスの関門となるソフトウエアモジュールに存在するのが一般的である。

B. 設計レビューを行うためのリリースゲートを設定する

一貫した設計レビュープログラムを確立した後は、次の手順として、ソフトウエア開発ライフサイクル内の特定時点にリリースゲートを設定する。プロジェクトは、このゲートにおいて設計レビューを実施し、レビュー結果の検討と承認を受けるまで工程を進めることができない。これを実現するには、期待の基本水準を設定する必要がある。たとえば、重大度の高い発見内容があるプロジェクトは合格できないようにし、また、その他の発見内容についてはビジネスオーナーの承認を得ることを義務付ける。

設計レビューは、セキュリティ問題の早期発見を目的として設計フェーズの終了前に実施するのが一般的であり、プロジェクトチームからのリリース前に実施することが必須である。

レガシーシステムや停止中のプロジェクトに対しては、運用を継続するための例外プロセスを策定する。ただし、プロジェクトごとにレビューの時間枠を明示的に設定し、既存システム内の隠れた脆弱性が発見されるようにする。例外に該当するプロジェクト件数は全体の 20%以下になるようにする。

成果

- システム設計上の弱点を細かく把握できるようになり、区画化が促進される
- アーキテクチャに対して期待されるセキュリティ基本水準に照らしたプロジェクトの現在位置が組織レベルで意識される
- 既知の弱点の緩和について、効率性や進捗状況をプロジェクト間で比較できるようになる

追加成功指標

- 直近 6 カ月以内に、80%超のプロジェクトがデータフロー図を更新した
- 直近 6 カ月以内に、75%超のプロジェクトが設計レビュー監査に合格した

追加コスト

- データフロー図の維持管理による継続的なプロジェクトオーバヘッド
- 設計レビュー監査の不合格によるプロジェクト遅延で生じる組織オーバヘッド

追加人員

- 開発者（2 日/年）
- アーキテクト（1 日/年）
- マネージャー（1〜2 日/年）
- ビジネスオーナー（1〜2 日/年）
- セキュリティ監査員（2〜3 日/年）

関連レベル

- セキュアなアーキテクチャ - 3
- コードレビュー - 3

コードレビュー

	CR 1	CR 2	CR 3
目的	**取り組みやすい範囲で、基本的なコードレベルの脆弱性やその他のセキュリティ問題を検出する**	**開発中に行うコードレビューの精度と効率が自動化によって向上する**	**総合的なコードレビュー作業を義務付け、言語レベルのリスクとアプリケーションに特化したリスクを検出する**
実施内容	A. 既知のセキュリティ要件に基づいてレビューチェックリストを作成する B. リスクの高いコードに対して重点レビューを実行する	A. 自動コード分析ツールを利用する B. コード分析作業を開発プロセスに組み込む	A. アプリケーションに特化した問題に対応するようコード分析をカスタマイズする B. コードレビューを行うためのリリースゲートを設定する
査定項目	◆ ほとんどのプロジェクトチームが、一般的な問題についてのレビューチェックリストを用意している ◆ プロジェクトチームが、選出された高リスクのコードに対しておおむねレビューを実施している	◆ セキュリティの問題を発見するための自動コード分析ツールが、ほとんどのプロジェクトチームで利用できるようになっている ◆ ほとんどの利害関係者がコードレビューの結果を常に要求し、確認している	◆ プロジェクトチームで、アプリケーション固有のコーディング標準に照らしたコードチェックが自動化手法を利用して行われている ◆ 定期的なプロジェクト監査において、コードレビューの結果がリリース前に一定の基本水準を満たすことが義務付けられている
成果	◆ 検査によって、システムの露見や攻撃につながる可能性が大きい一般的なコード脆弱性を発見できる ◆ 簡易なレビューによって、重大なセキュリティ被害につながるコーディングエラーを発見できる ◆ セキュリティ保証に関する基本的なデューディリジェンスがコードレベルで行われる	◆ 開発チームが、コードレベルに存在するセキュリティ脆弱性の自己チェックを随時実行できるようになる ◆ 定期的な分析作業によって、セキュアなコーディングに関するチームごとの慣行について過去の実績データが作成される ◆ 緩和されていない脆弱性が利害関係者に認識され、的確なトレードオフ分析を行いやすくなる	◆ コード分析結果の正確さと妥当性についての信頼度が向上する ◆ セキュアなコーディングについて組織全体から期待される基本水準が確立する ◆ プロジェクトチームに、コードレベルのセキュリティを判断するための客観的な目標が与えられる

コードレビュー

取り組みやすい範囲で、基本的なコードレベルの脆弱性やその他のセキュリティ問題を検出する

実施内容

A. 既知のセキュリティ要件に基づいてレビューチェックリストを作成する

特定プロジェクトに関する既知のセキュリティ要件から、セキュリティのための簡易なコードレビューチェックリストを作成する。リストの内容は、機能要件に関連するセキュリティ上の懸念に特化した項目や、実装言語、プラットフォーム、一般的な技術スタックなどに基づいてセキュアなコーディングのベストプラクティスを確認する項目などである。このため、組織内で複数種類のソフトウエア開発が行われている場合、それらをすべてチェックするには複数のチェックリストが必要になることが多い。

チェックリスト作成の基になったものが公開されているリソースか、購入したものであるかにかかわらず、技術的な利害関係者（開発マネージャー、アーキテクト、開発者、セキュリティ監査員など）がチェックリストをレビューして有効性や現実性を確認すべきである。リストの内容は、手作業や簡単な検索ツールでも見つかるような単純で優先度の高い問題の検出を重視して、項目数を増やさず簡潔にすることが重要である。自動コード分析ツールを使用して同じ目的を達成することもできるが、その場合も、カスタマイズしてチェック内容を全体的に減らし、重要な項目だけを残してスキャン及びレビューのプロセスを効率化すべきである。

開発者に対しては、職能に応じたチェックリストの目標について概要を説明する。

B. リスクの高いコードに対して重点レビューを実施する

コードレベルの脆弱性は、ソフトウエア内のセキュリティ上重要な部分に含まれていると被害が劇的に大きくなるおそれがある。このため、プロジェクトチームは、高リスクのモジュールを対象として一般的な脆弱性を見つけるレビューを実施する。高リスク機能の一般的な例としては、認証モジュール、アクセス制御適用ポイント、セッション管理の仕組み、外部インターフェイス、入力検証、データ解析などが挙げられる。

コードレビューチェックリストを利用すれば、変更が生じる場合はプロジェクトチームのメンバに対象モジュールを割り当ててレビューを行うというように、通常の開発プロセスの一環として分析作業を実施できる。また、セキュリティ監査ツールや自動レビューツールを利用することも可能である。

高リスクのコードの変更やレビューを含んだ開発サイクルでは、開発マネージャーが、ほかのプロジェクト利害関係者から情報を得て、発見内容のトリアージと緩和策の優先順位付けを適切に行う。

成果

- 検査によって、システムの露見や攻撃につながる可能性が大きい一般的なコード脆弱性を発見できる
- 簡易なレビューによって、重大なセキュリティ被害につながるコーディングエラーを発見できる
- セキュリティ保証に関する基本的なデューディリジェンスがコードレベルで行われる

成功指標

- 直近 6 カ月以内に、80%超のプロジェクトチームに対して、該当するコードレビューチェックリストの概要を説明した
- 直近 6 カ月以内に、50%超のプロジェクトチームが、高リスクのコードを対象としてコードレビューを実施した
- コードレビューチェックリストの有用性に関する開発者からの報告で、リッカート尺度による評価が 5 段階評価において 3.0 を超えた

コスト

- コードレビューチェックリストの構築またはライセンス
- 高リスクのコードを対象としたコードレビュー活動による継続的なプロジェクトオーバヘッド

人員

- 開発者（2〜4 日/年）
- アーキテクト（1〜2 日/年）
- マネージャー（1〜2 日/年）
- ビジネスオーナー（1 日/年）

関連レベル

- セキュリティ要件 - 1

コードレビュー

開発中に行うコードレビューの精度と効率が自動化によって向上する

成果

- 開発チームが、コードレベルに存在するセキュリティ脆弱性の自己チェックを随時実行できるようになる
- 定期的な分析作業によって、セキュアなコーディングに関するチームごとの慣行について過去の実績データが作成される
- 緩和されていない脆弱性が利害関係者に認識され、的確なトレードオフ分析を行いやすくなる

追加成功指標

- 直近 6 カ月以内に、50%超のプロジェクトがコードレビューを実施し、利害関係者の承認を得た
- 直近 1 カ月以内に、80%超のプロジェクトが自動化コードレビューの結果にアクセスした

追加コスト

- コード分析ソリューションの調査及び選定
- 自動化統合の初期コスト及び保守管理
- 自動化コードレビュー及び緩和策による継続的なプロジェクトオーバヘッド

追加人員

- 開発者（1〜2 日/年）
- アーキテクト（1 日/年）
- マネージャー（1〜2 日/年）
- セキュリティ監査員（3〜4 日/年）

実施内容

A. 自動コード分析ツールを利用する

コードレベルにおけるセキュリティ脆弱性の多くは、複雑であり、発見するには入念な検査が必要である。しかし、自動的にコードを分析してバグや脆弱性を発見する有用な自動化ソリューションが数多く提供されている。

広く普及しているプログラミング言語やフレームワークに対応した市販の製品及びオープンソースの製品がある。適切なコード分析ソリューションの選定は、検査の深さと精度、製品のユーザビリティと使用モデル、拡張性及びカスタマイズ機能、組織のアーキテクチャや技術スタックへの対応可能性など、いくつかの要因に基づいて行う。

選定プロセスではセキュリティに精通した技術要員、開発者、開発マネージャーから情報を得て利用し、全体的な結果については利害関係者とのレビューを実施する。

B. コード分析作業を開発プロセスに組み込む

コード分析ソリューションを選定した後は、そのソリューションを開発プロセスに統合し、機能をプロジェクトチームで利用するよう促す必要がある。効果的な方法としては、インフラストラクチャの設定によってビルド時にスキャンを自動実行させることや、プロジェクトのコードリポジトリにあるコードから自動実行させることが考えられる。この方法では結果が早期に入手できるため、開発チームがリリース前の作業と並行して自己チェックを実行できるようになる。

レガシーシステムや進行中の大規模なプロジェクトで発生しうる問題として、コードスキャナが当該リリースでは更新されないモジュール内の発見内容を多数報告することがある。自動スキャンを定期的に実行するように設定している場合、レビューのオーバヘッドを効果的に回避するには、発見の対象範囲を、前回のスキャン以降に追加、削除、変更が行われた箇所に限定するという方法がある。それ以外の範囲に関する結果を無視できない事情がある場合、開発マネージャーはセキュリティ監査員、利害関係者、プロジェクトチームからの情報を入力として、残りの発見内容に対応する具体的な計画を作成する。

コードレビューで発見されてからリリース時まで未対策のまま残っている問題がある場合は、それらについてプロジェクト利害関係者のレビューと承認を受ける必要がある。

コードレビュー

総合的なコードレビュー作業を義務付け、言語レベルのリスクとアプリケーションに特化したリスクを検出する

実施内容

A. アプリケーションに特化した問題に対応するようコード分析をカスタマイズする

コードスキャンツールは、言語 API 及び共通的に使用されるライブラリに基づくコードをチェックするための組み込みの規則ナレッジベースにより機能するが、カスタムの API や設計を理解して類似のチェックを適用する能力は限られている。しかし、カスタマイズすれば、組織やプロジェクトに特有のセキュリティ上の懸念を発見する強力かつ汎用的な分析エンジンとなりうる。

カスタム分析のしやすさや能力の詳細はツ―ルによって異なるが、コードスキャナのカスタマイズは、特定の API や関数呼び出し箇所に対して実行するチェックの内容を指定することにより行うのが一般的である。チェックに含まれる内容としては、内部コーディング標準への適合性の分析、カスタムインターフェイスへの未チェック汚染データの引き渡し、機密性のあるデータの扱いに関する追跡管理及び検証、内部 API の正しい使用法などがある。

作成したチェッカは複数のプロジェクトで利用できるため、共有コードベースの使用に関するチェッカは、スキャナのカスタマイズ作業に着手する最初の部分として効果的である。コードベースのためにツールをカスタマイズする場合、セキュリティ監査員は、コードと概略設計の両方を検査して候補チェッカを特定し、実装について開発要員及び利害関係者と話し合う。

B. コードレビューを行うためのリリースゲートを設定する

すべてのソフトウエアプロジェクトを対象としたコードレベルのセキュリティに関する基本水準を設定するには、ソフトウエア開発ライフサイクルにおける特定の時点にチェックポイントを設定し、その時点でコードレビューの結果が最低限の標準に達していることをリリースの必須条件とする。

最初、たとえば脆弱性の種類を 1 つか 2 つ選択するなどして単純な方法で満たせる基準を指定し、その基準に該当する発見内容がプロジェクトにある場合は合格できない設定にする。時間の経過とともに、チェックポイントを通過するための条件を追加し、この基本水準となる基準の内容を高度にしていく。

コードレビューのチェックポイントは実装フェーズの終了前に設定するのが一般的であり、リリース前に設定することが必須である。

レガシーシステムや停止中のプロジェクトに対しては、運用を継続するための例外プロセスを策定する。ただし、発見内容緩和の時間枠を明示的に設定する。例外に該当するプロジェクト件数は全体の 20%以下になるようにする。

成果

- コード分析結果の正確さと妥当性についての信頼度が向上する
- セキュアなコーディングについて組織全体から期待される基本水準が確立する
- プロジェクトチームに、コードレベルのセキュリティを判断するための客観的な目標が与えられる

追加成功指標

- プロジェクトの 50%超がコード分析のカスタマイズを使用した
- 直近 6 カ月以内に、75%超のプロジェクトがコードレビュー監査に合格した

追加コスト

- カスタムコードレビューチェックの構築と維持管理
- コードレビュー監査による継続的なプロジェクトオーバヘッド
- コードレビュー監査の不合格によるプロジェクト遅延で生じる組織オーバヘッド

追加人員

- アーキテクト（1 日/年）
- 開発者（1 日/年）
- セキュリティ監査員（1～2 日/年）
- ビジネスオーナー（1 日/年）
- マネージャー（1 日/年）

関連レベル

- ポリシー&コンプライアンス - 2
- セキュアなアーキテクチャ - 3

セキュリティテスト

			
目的	**実装とソフトウエアの要件に基づく基本的なセキュリティテストを実行するためのプロセスを確立する**	**開発中に行うセキュリティテストの完全性と効率が自動化によって向上する**	**配備前に基本水準のセキュリティを確保するため、アプリケーション固有のセキュリティテストを要求する**
実施内容	A. 既知のセキュリティ要件に基づいてテストケースを作成する B. ソフトウエアリリースに対する侵入テストを実施する	A. 自動セキュリティテストツールを利用する B. セキュリティテスト作業を開発プロセスに組み込む	A. アプリケーションに特化したセキュリティテスト自動化手法を採用する B. セキュリティテストを行うためのリリースゲートを設定する
査定項目	◆ プロジェクトにおいて、要件に基づく何らかのセキュリティテストを定めている ◆ ほとんどのプロジェクトがリリース前の侵入テストを実施している ◆ ほとんどの利害関係者がリリース前のセキュリティテストの状況について認識している	◆ プロジェクトによるセキュリティテストケースの評価に自動化手法が使われている ◆ ほとんどのプロジェクトが、セキュリティテストの評価と利害関係者への報告を一貫したプロセスに従って行っている	◆ アプリケーション固有のロジックに対して総合的なセキュリティテストケースが作られている ◆ 定期的なプロジェクト監査において、セキュリティテストの結果が最低限の基準を満たすことが要求されている
成果	◆ 重要なビジネス機能の周囲に期待されるセキュリティメカニズムを独立的に確認できる ◆ セキュリティテストに関する概略的なデューディリジェンスが行われる ◆ 個々のソフトウエアプロジェクトに対するセキュリティテスト群が当該目的に特化する形で充実する	◆ セキュリティに関するソフトウエア機能をより深く一貫した方法で確認できるようになる ◆ 開発チームがリリース前の自己チェックと問題修正を実行できるようになる ◆ リスク受容の意思決定を行う際、残された脆弱性を利害関係者がよりよく認識する	◆ アプリケーションが攻撃を受けた場合のパフォーマンスについて組織全体から期待される基本水準が確立する ◆ カスタマイズされたセキュリティテスト群によって自動分析の精度が向上する ◆ 攻撃耐性に関する客観的な目標がプロジェクトチームに意識される

セキュリティテスト

実装とソフトウエアの要件に基づく基本的なセキュリティテストを実行するためのプロセスを確立する

実施内容

A. 既知のセキュリティ要件に基づいてテストケースを作成する

プロジェクトに関する既知のセキュリティ要件から、当該ソフトウエアの機能が正しいことをチェックするテストケース群を作成する。これらのテストケースは一般に、当該システムの機能要件とビジネスロジックに関するセキュリティ上の懸念事項から導き出される。ただし、実装言語または技術スタックに基づく一般的な脆弱性に関する汎用的なテストも含んでいる必要がある。

たいていの場合、プロジェクトチームの時間を割いてアプリケーションに特有のテストケースを作成し、一般に公開されたリソースまたは購入したナレッジベースを利用してセキュリティに関する一般的なテストケースから適用可能なものを選択するのがもっとも効果的である。必須ではないが、自動セキュリティテストツールを利用することで一般的なセキュリティテストケースに対応してもよい。

このテストケース計画は要件フェーズまたは設計フェーズにおいて実施すべきであり、リリース前の最終テストに先立って実行することが必須である。候補テストケースについては、関係する開発/セキュリティ/品質保証要員がレビューを実施して、適用可能性、効果、現実性を確認すべきである。

B. ソフトウエアリリースに対する侵入テストを実施する

個々のプロジェクトのために特定したセキュリティテストケース群を使用して、各ケースにおけるシステムのパフォーマンスを評価するための侵入テストを実施する。これは、リリースに先立つテストフェーズにおいて実施するのが一般的である。

侵入テストケースには、ビジネスロジックの安定性を確認するアプリケーションに特化したテストと、設計及び実装をチェックする一般的な脆弱性テストの両方を含める。作成後、セキュリティテストケースはセキュリティに精通した品質保証要員または開発要員が実行する。ただし、セキュリティテストケースを初めて実行するプロジェクトチームには、チームメンバに対する支援・コーチのためにセキュリティ監査員が立ち会う。

リリースまたは配備に先立って、利害関係者はセキュリティテストの結果をレビューし、不合格になったセキュリティテストが示すリリース時のリスクを受け入れる必要がある。配備前の場合は、将来的にキャップを解決するための具体的なタイムラインを設定する。

成果

- 重要なビジネス機能の周囲に期待されるセキュリティメカニズムを独立的に確認できる
- セキュリティテストに関する概略的なデューディリジェンスが行われる
- 個々のソフトウエアプロジェクトに対するセキュリティテスト群が当該目的に特化する形で充実する

成功指標

- 直近 12 カ月以内に、50%超のプロジェクトに関するセキュリティテストケースが作成された
- 直近 6 カ月以内に、50%超の利害関係者に対して、プロジェクトのセキュリティテストに関する状況の概要を説明した

コスト

- セキュリティテストケースの構築またはライセンス
- セキュリティテストケースの維持管理及び評価による継続的なプロジェクトオーバヘッド

人員

- QA テスト担当者（1〜2 日/年）
- セキュリティ監査員（1〜2 日/年）
- 開発者（1 日/年）
- アーキテクト（1 日/年）
- ビジネスオーナー（1 日/年）

関連レベル

- セキュリティ要件 - 1

ST 2 セキュリティテスト

開発中に行うセキュリティテストの完全性と効率が自動化によって向上する

成果
- セキュリティに関するソフトウエア機能をより深く一貫した方法で確認できるようになる
- 開発チームがリリース前の自己チェックと問題修正を実行できるようになる
- リスク受容の意思決定を行う際、残された脆弱性を利害関係者がよりよく認識する

追加成功指標
- 直近 6 カ月以内に、50%超のプロジェクトがセキュリティテストを実施し、利害関係者の承認を得た
- 直近 1 カ月以内に、80%超のプロジェクトが自動セキュリティテストの結果にアクセスした

追加コスト
- 自動セキュリティテストソリューションの調査及び選定
- 自動化統合の初期コスト及び保守管理
- 自動セキュリティテスト及び緩和策による継続的なプロジェクトオーバヘッド

追加人員
- 開発者（1 日/年）
- アーキテクト（1 日/年）
- マネージャー（1〜2 日/年）
- セキュリティ監査員（2 日/年）
- QA テスト担当者（3〜4 日/年）

実施内容

A. 自動セキュリティテストツールを利用する

セキュリティ問題に関するテストを実施するには、個々のソフトウエアインターフェイスに対して多数の入力ケースをチェックしなくてはならない可能性がある。このため、テストケースを手作業で実装・実行して効果的なセキュリティテストを行うことは現実的でない場合がある。したがって、自動化セキュリティテストツールを使用して自動的にソフトウエアをテストし、セキュリティテストの効率と結果の品質を向上させるべきである。

利用できるツールには商用製品もオープンソース製品もあり、また、当該組織に適しているかどうかはレビューによって確認する必要がある。適切なツールの選定は、組み込まれているセキュリティテストケースの健全性及び正確さ、当該組織にとって重要な種類のアーキテクチャに対するテストの効果、テストケースを変更または追加するためのカスタマイズ、当該開発組織から見た発見内容の品質とユーザビリティなど、いくつかの要因に基づいて行う。

選定プロセスではセキュリティに精通した技術要員、開発要員、品質保証要員から情報を得て利用し、全体的な結果については利害関係者とのレビューを実施する。

B. セキュリティテスト作業を開発プロセスに組み込む

自動セキュリティテスト実行用のツールを使用して、組織内の各プロジェクトで開発中にセキュリティテストを定期的に実行し、その結果をレビューする。オーバヘッドを少なくし、大規模化にも対応できるように、セキュリティテストツールの定期的な自動実行（毎晩、毎週など）を構成し、発見内容を随時検査する。

要件定義フェーズまたは設計フェーズという早い段階でセキュリティテストを実施することにはメリットがある。機能テストケースでは以前から行われていることであるが、テストを軸にしたこの種の開発手法では、開発サイクルの早期（普通は設計時）に適切なセキュリティテストケースを特定し、実施する。自動実行されるセキュリティテストケースで多数の不合格が報告される状態でプロジェクトは実装フェーズに入る。不合格項目は未実装の機能を示しており、すべてのテストに合格すると実装が完了する。開発サイクルの早期に開発の目標がはっきりと示されるため、セキュリティ上の懸念によるリリース遅延や、プロジェクトの納期に合わせるためにやむを得ずリスクを受容する事態が発生するリスクが低減される。

プロジェクトのリリースごとに、自動・手動セキュリティテストの結果をマネージャー及びビジネス利害関係者に示し、レビューを実施する。発見内容が未対策のままリリース時の受容リスクとして残る場合は、利害関係者と開発マネージャーが協力して、それらへの対策を行う具体的なタイムフレームを決定する。

セキュリティテスト

ST 3

配備前に基本水準のセキュリティを確保するため、アプリケーション固有のセキュリティテストを要求

実施内容

A. アプリケーションに特化したセキュリティテストの自動化を採用する

プロジェクトチームは、セキュリティテストツールのカスタマイズ、汎用テストケース実行ツールの機能拡張、または独自のテストツールの構築のいずれかを通じ、正式にセキュリティ要件を繰り返し確認し、自動チェッカを作成して、実装済みビジネスロジックのセキュリティをテストする。

さらに、テスト中のプロジェクトにおいて、個々のソフトウエアインターフェイスをより詳細に考慮して自動セキュリティツールがカスタマイズされていれば、精度と対象の深さの点でこれらツールの多くを大幅に改善できる。コンプライアンスまたは技術標準に関する組織固有の問題点は、監査データの収集とプロジェクト単位の管理の可視性をより単純にするための、再利用可能な重要テスト集として体系化できる。

プロジェクトチームは、ソフトウエアのビジネス機能に基づいた大まかなセキュリティテストケースの構築に焦点を合わせるべきである。セキュリティ監査員が率いる組織レベルのチームは、コンプライアンスと内部標準に関する自動テストの仕様に焦点を合わせるべきである。

B. セキュリティテストを行うためのリリースゲートを設定する

セキュリティ上のバグが簡単に見つかる状態でソフトウエアがリリースされないようにするには、ソフトウエア開発のライフサイクルのある時点をチェックポイントとして定めるべきである。チェックポイントでは、プロジェクトからリリースするために、一連のセキュリティテストケースを確立し、それらに合格しなければならない。これにより、すべてのプロジェクトの合格が期待されるセキュリティテストの種類に関する基本水準が確立される。

最初に多くのテストケースを追加しすぎるとオーバヘッドコストが増加するおそれがあるため、最初は 1 つか 2 つのセキュリティ問題を選択し、テストに失敗しても合格してしまうプロジェクトがないよう、問題ごとに幅広いテストケースを設定する。この基本水準は、時間の経過とともにその他のセキュリティ問題を選択したり、相当するテストケースを追加したりして改善する必要がある。

一般に、このセキュリティテストのチェックポイントは、実装またはテストの終わりの方に設定すべきだが、リリース前には必ず設定しなければならない。

レガシーシステムや停止中のプロジェクトに対しては、運用を継続するための例外プロセスを策定する。ただし、発見内容緩和の時間枠を明示的に設定する。例外に該当するプロジェクト件数は全体の 20%以下になるようにする。

成果

- アプリケーションが攻撃を受けた場合のパフォーマンスについて組織全体から期待される基本水準が確立する
- カスタマイズされたセキュリティテスト群によって自動分析の精度が向上する
- 攻撃耐性に関する客観的な目標がプロジェクトチームに意識される

追加成功指標

- 50%超のプロジェクトがセキュリティテストのカスタマイズを利用した
- 直近 6 カ月以内に、75%超のプロジェクトがすべてのセキュリティテストに合格した

追加コスト

- セキュリティテスト自動化を目的としたカスタマイズのビルドアウトと保守
- セキュリティテストの監査プロセスに起因する、プロジェクトの継続的オーバヘッド
- 失敗したセキュリティテストの監査によって生じたプロジェクトの遅延に起因する、組織のオーバヘッド

追加人員

- アーキテクト（1 日/年）
- 開発者（1 日/年）
- セキュリティ監査員（1〜2 日/年）
- QA テスト担当者（1〜2 日/年）
- ビジネスオーナー（1 日/年）
- マネージャー（1 日/年）

関連レベル

- ポリシー&コンプライアンス – 2
- セキュアなアーキテクチャ - 3

脆弱性管理

目的	**脆弱性レポートまたはインシデントへの対応に関する高レベル計画を理解する**	**対応プロセスに期待される内容を細かく検討し、一貫性と情報発信の方法を改善する**	**対応プロセスにおける分析とデータ収集の方法を改善し、事前予防の計画に反映する**
実施内容	A. セキュリティ問題の連絡窓口を決定する B. 非公式のセキュリティ対応チームを編成する	A. 一貫したインシデント対応プロセスを確立する B. セキュリティ問題の情報公開プロセスを採択する	A. インシデントの根本原因分析を実施する B. インシデントごとに指標を収集する
査定項目	◆ ほとんどのプロジェクトにセキュリティ問題の連絡窓口が置かれている ◆ セキュリティ対応を担当するチームが組織に存在する ◆ ほとんどのプロジェクトチームが、セキュリティ問題に関する各部署の連絡窓口及び対応チームについて認識している	◆ インシデントの報告と対応に関して、組織内に一貫したプロセスが設定されている ◆ ほとんどのプロジェクト利害関係者が、関係するソフトウエアプロジェクトに関連する適切なセキュリティ情報公開について認識している	◆ ほとんどのインシデントについて根本原因の調査が実行され、以後の推奨事項が作られている ◆ ほとんどのプロジェクトにおいて、インシデントに関連するデータや指標の収集・報告が一貫した方法で行われる
成果	◆ 優先度の高い脆弱性またはインシデントに対応するための軽量プロセスが確立される ◆ 利害関係者に対する、セキュリティ上の影響がある事象の通知及び報告の体制が確立される ◆ セキュリティ問題に対処するための概略レベルのデューディリジェンスが実現する	◆ サードパーティからの脆弱性レポートに対処するコミュニケーション計画が策定される ◆ ソフトウエアオペレータにセキュリティパッチをリリースするための明確なプロセスが確立される ◆ インシデントの追跡と対処及びインシデントに関する内部のコミュニケーションに関する正式なプロセスが確立される	◆ 各インシデント後における、組織改善のための詳細なフィードバックが行われる ◆ 脆弱性及びセキュリティ侵害によって生じる損害の概算が得られる ◆ 利害関係者は、過去のインシデントの傾向に基づき、妥協点に関してより適切な判断を下せる

脆弱性管理

脆弱性レポートまたはインシデントへの対応に関する高レベル計画を理解する

実施内容

A. セキュリティ問題の連絡窓口を決定する

組織内の各部門、または各プロジェクトチームに対して、セキュリティ情報のコミュニケーションハブとして機能する連絡窓口を確立する。一般に、この任務において個人が多くの時間を要求されることはないが、事前に連絡窓口を定める目的は、脆弱性管理の体制とガバナンスを確立することにある。

連絡窓口の活用が必要となる可能性のあるインシデントの例としては、外部の主体からの脆弱性レポートの受領、現場で使用されているソフトウエアのセキュリティ侵害やその他のセキュリティ上の問題、または内部で発見した高リスクの脆弱性などが挙げられる。イベント発生時には、もっとも近い連絡窓口が、その影響を受けたプロジェクトチームの追加リソース及びアドバイザとして関与して技術的指針を示し、利害関係者には影響軽減のための取り組みの進行状況を説明する。

連絡窓口は、組織内のソフトウエアプロジェクトについて幅広い知識を持つ、セキュリティに精通した技術者または管理者から選択するべきである。選定されたセキュリティ連絡窓口のリストは一元的に維持し、少なくとも 6 カ月ごとに更新する。さらに、組織内の要員がセキュリティ上の問題について協力を要請したり、互いに直接連携したりできるように、このリストを公開、周知する。

B. 非公式のセキュリティ対応チームを編成する

セキュリティ連絡窓口の任務を割り当てられた人員のリスト、または専任のセキュリティ担当者の中から、技術面を集中的に担当するセキュリティ対応チームの役割を担う少人数のグループを選定する。このチームの任務には、セキュリティインシデントレポートまたは脆弱性レポートに対して直接的にオーナーシップを発揮することと、トリアージの実施、緩和策の実施、利害関係者への報告を担当することが含まれる。

インシデントが発生して活動が開始すると、セキュリティ対応チームのメンバは、経営陣向けのブリーフィングや上司への伝達も担当する。セキュリティ対応チームはほとんどの時間は、チームメンバとしての活動は実施していない可能性が高いが、いったん事が起きたら迅速に対応できるだけの柔軟性があるか、他のチームメンバにインシデント対応を委ねるスムーズなプロセスが存在していることが必要である。

セキュリティ対応チームは、対応プロセス、及びプロジェクトチームからのセキュリティ関連レポートに関する概略レベルの期待事項について、少なくとも年に 1 回、セキュリティ連絡窓口に概要を伝えるミーティングを開催する。

成果
- 優先度の高い脆弱性またはインシデントに対応するための軽量プロセスが確立される
- 利害関係者に対する、セキュリティ上の影響がある事象の通知及び報告の体制が確立される
- セキュリティ問題に対処するための概略レベルのデューディリジェンスが実現する

成功指標
- 直近 6 カ月以内に組織の 50%超に、もっとも近いセキュリティ連絡窓口について概要が説明された
- 直近 12 カ月に、セキュリティ対応チーム及びセキュリティ連絡窓口のミーティングが少なくとも 1 回開催された

コスト
- プロジェクトにおいてセキュリティ連絡窓口の役割を果たす要員の継続的で可変のオーバヘッド
- 適切なセキュリティ対応チームの特定

人員
- セキュリティ監査員（1 日/年）
- アーキテクト（1 日/年）
- マネージャー（1 日/年）
- ビジネスオーナー（1 日/年）

関連レベル
- 教育&指導 - 2
- 戦略&指標 - 3

脆弱性管理

対応プロセスに期待される内容を細かく検討し、一貫性と情報発信の方法を改善する

成果
- サードパーティからの脆弱性レポートに対処するコミュニケーション計画が策定される
- ソフトウエアオペレータにセキュリティパッチをリリースするための明確なプロセスが確立される
- インシデントの追跡と対処及びインシデントに関する内部のコミュニケーションに関する正式なプロセスが確立される

追加成功指標
- 直近 6 カ月以内に、80%超のプロジェクトチームにおいて、インシデント対応プロセスの概要が説明された
- 直近 6 カ月以内に、80%超の利害関係者がセキュリティ問題の開示内容について伝達された

追加コスト
- インシデント対応プロセスに起因する、組織の継続的オーバヘッド

追加人員
- セキュリティ監査員（3〜5 日/年）
- マネージャー（1〜2 日/年）
- ビジネスオーナー（1〜2 日/年）
- サポート/オペレータ（1〜2 日/年）

実施内容

A. 一貫したインシデント対応プロセスを確立する

非公式だったセキュリティ対応チームを拡張して、メンバが従うべき手続きのほか、組織のインシデント対応プロセスを明示的に文書化する。さらに、セキュリティ対応チームの各メンバは、少なくとも年に 1 回、この資料に基づいて訓練を受けなければならない。

堅実なインシデント対応プロセスにはいくつかの基本方針がある。たとえば、損害の拡大を防止するための初期トリアージ、変更管理とパッチ適用、プロジェクト担当者及びインシデントに関与する人々の管理、フォレンジックの証拠の収集と保管、インシデントに関するコミュニケーションを利害関係者に限定、利害関係者や連絡網への報告などがある。

セキュリティ対応担当者は、開発チームと連携し、技術的分析を行ってインシデントまたは脆弱性の各レポートに関する事実と仮説を検証する。同様に、プロジェクトチームはインシデントまたは高リスクの脆弱性を知った場合、セキュリティ対応チームのメンバと連絡をとる内部プロセスに従う。

B. セキュリティ問題の情報公開プロセスを採択する

ほとんどの組織にとって、セキュリティ問題のニュースが公になることは望ましくないが、セキュリティ問題に関して内部から外部へ連絡する重要な方法がいくつかある。

1 つ目は、もっともよくある方法で、組織が製作したソフトウエアに対し、セキュリティパッチの作成及び配布を通じて連絡する方法である。一般に、ソフトウエアプロジェクトのすべてが内部だけで使用される場合、この方法はそれほど重要ではなくなるが、外部の当事者がソフトウエアを運用するすべての場合に備え、パッチリリースプロセスが存在しなければならない。パッチリリースプロセスは、変更管理及びパッチリリース前の回帰テスト、パッチの重大度カテゴリ情報を含めた、オペレータまたはユーザに対する通知、限定的な技術詳細（攻撃手段が直接引き出されないようにするため）など、複数の要因に備えるべきである。

2 つ目の方法は、組織のソフトウエアに存在するセキュリティ脆弱性をレポートするサードパーティと連絡を取る方法である。対応のための時間枠を定めた期待されるプロセスを採用し、外部に公開することにより、脆弱性の報告者による責任ある開示対策の実施が促される。

最後は、個人識別情報及びその他の機密データの盗難が関係するインシデントに対し、多くの州及び国が法的に要求している外部とのコミュニケーションである。この種のインシデントが発生した場合、セキュリティ対応チームはマネージャー及び事業の利害関係者と協同で、その後の適切な方策を決定するべきである。

脆弱性管理

対応プロセスにおける分析とデータ収集の方法を改善し、事前予防の計画に反映する

実施内容

A. インシデントの根本原因分析を実施する

時間を要する可能性はあるが、インシデント対応プロセスには、インシデントの主たる原因となった根本的なセキュリティ障害を特定するための分析を追加するべきである。根本原因は、コードレベルの脆弱性や設定エラーなどの技術的問題の場合もあれば、ソーシャルエンジニアリングや手続き上の不手際など、人やプロセスの問題の場合もある。

インシデントの根本原因を特定したら、類似のインシデントが発生するおそれのある組織では、ほかの潜在的弱点を見つけ出すツールとしてその根本原因を利用すべきである。特定した弱点ごとに、影響を事前に緩和するための追加の推奨事項を、元々のインシデント対応の取り組みを終了する過程の中で伝えること。

根本原因の分析に基づく推奨事項は、影響緩和対策のスケジュールを設定、または容認されるリスクを認識するため、経営陣及び事業の利害関係者がレビューするべきである。

B. インシデントごとに指標を収集する

セキュリティ侵害及び優先度の高い脆弱性レポートのすべてに対処する一元化されたプロセスを用意することにより、所定の期間における傾向を測定し、セキュリティ保証の取り組みの影響や効率を判断することが可能となる。

過去のインシデントの記録は、少なくとも 6 カ月ごとに保存及びレビューを行う。類似のインシデントを分類してグループ分けし、問題の分類ごとに全体の件数を数えておく。インシデントから得られるその他の測定値として、ソフトウエアプロジェクトがインシデントの影響を受けた頻度、システムの停止時間と使用できないことによる損害額、インシデントの対応と整理に携わる人的資源、規制の違反金やブランド価値の低下など長期的損失の見積額が挙げられる。技術的問題が実際の根本原因である場合、根本原因は、事前の対策、レビュー、運用中の対策のどれによって問題をより早く検出できたか、または損害を軽減できたかを明らかにしておくと役立つ。

この情報は、組織が時間の経過とともに受けるセキュリティに関する現実の影響を表しているため、プログラム計画プロセスにとって具体的なフィードバックとなる。

成果

- 各インシデント後における、組織改善のための詳細なフィードバックが行われる
- 脆弱性及びセキュリティ侵害によって生じる損害の概算が得られる
- 利害関係者は、過去のインシデントの傾向に基づき、妥協点に関してより適切な判断を下せる

追加成功指標

- 直近 6 カ月以内に、80%超のインシデントが根本原因及びその他の推奨事項とともに文書化された
- 直近 6 カ月以内に、80%超のインシデントが評価指標を得るために整理された

追加コスト

- インシデントの詳細な調査と分析の実施による、組織の継続的オーバヘッド
- インシデント評価指標の収集とレビューによる、組織の継続的オーバヘッド

追加人員

- セキュリティ監査員（3 日/年）
- マネージャー（2 日/年）
- ビジネスオーナー（2 日/年）

関連レベル

- 戦略&指標 - 3

環境の堅牢化

	EH 1	EH 2	EH 3
目的	**アプリケーションとソフトウエアコンポーネントに関する運用環境の基本水準を理解する**	**運用環境の堅牢化によってアプリケーション運用の信頼度を向上させる**	**既知のベストプラクティスに照らしてアプリケーションの状況及び状態の妥当性を確認する**
実施内容	A. 運用環境に関する仕様を維持管理する B. 重要なセキュリティアップグレード及びパッチを特定し、インストールする	A. 定期的なパッチ管理プロセスを確立する B. 環境の基本水準に関する構成状態を監視する	A. 適切な運用保護ツールを指定し、配備する B. 環境構成の監査プログラムを展開する
査定項目	◆ プロジェクトの大多数が運用環境の一部の要件を文書化している ◆ ほとんどのプロジェクトがサードパーティ製ソフトウエアコンポーネントに対するセキュリティアップデートの有無を確認している	◆ 重要な依存対象要素に対するアップグレードやパッチの適用が、一貫した手順で実行される ◆ ほとんどのプロジェクトが、アプリケーションと環境の正常性チェックに自動化手法を採用する	◆ 利害関係者が、運用時に動作中のソフトウエアを保護するための追加ツールの選択肢を認識している ◆ 定期的な監査で、環境の正常性が基本水準を満たすことがほとんどのプロジェクトについてチェックされる
成果	◆ 開発チーム内で運用面の期待水準が明確に理解される ◆ インフラストラクチャに起因する優先度の高いリスクが、合意されたスケジュールに沿って緩和される ◆ ソフトウエアオペレータは、セキュリティ上重要なインフラストラクチャの保守のための高レベルの計画が立てられる	◆ 運用中システムのセキュリティ特性の細かな検証ができる ◆ インフラストラクチャのリスク緩和のためのスケジュールに関して公式の期待水準が設定される ◆ ソフトウエアプロジェクトの現在の運用状態が利害関係者に確実に認識される	◆ セキュリティを多層的にチェックすることで運用環境が強化される ◆ 運用上の保守とパフォーマンスに関する、確立され、測定に基づく目標値が設定される ◆ 外部に依存している要素の不備を経由する攻撃が成功する可能性が低減される

環境の堅牢化

アプリケーションとソフトウエアコンポーネントに関する運用環境の基本水準を理解する

実施内容

A. 運用環境に関する仕様を維持管理する

プロジェクトごとに、期待される運用プラットフォームを具体的に定め、維持する。組織によっては、この仕様を開発要員、利害関係者、サポートグループや運用グループとともに作成する。

この仕様は、ソフトウエアのビジネス機能に基づいて、運用環境について必ず当てはまる詳細事項をすべて把握することから始まる。具体的には、プロセッサのアーキテクチャ、オペレーティングシステムのバージョン、必須ソフトウエア、競合が生じるソフトウエアなどの要素を把握する。さらに、ソフトウエアの動作に影響する運用環境について、ユーザまたはオペレータが構成可能な既知のオプションにも注目する。

そのほかに、プロジェクトの設計と実装でなされた、運用環境に関する仮定を明確にし、それらの仮定を仕様内で把握する。

この仕様は、進行中のプロジェクトでは少なくとも 6 カ月ごと、ソフトウエアの設計または期待される運用環境に変更が加えられた場合はさらに高い頻度でレビュー、更新する。

B. 重要なセキュリティアップグレード及びパッチを特定し、インストールする

ほとんどのアプリケーションは、組み込みのプログラミング言語ライブラリ、サードパーティ製コンポーネントと開発フレームワーク、ベースのオペレーティングシステムなどで構成されたソフトウエアからなる独立の大規模なスタックの上で動作するソフトウエアである。大規模ソフトウエアスタック内のいずれかのモジュールにセキュリティ上の不備があれば、組織のソフトウエアのセキュリティ全体に影響が及ぶため、技術スタックの要素に対する重要なセキュリティアップデートはインストールしておかなければならない。

このため、リスクの高い依存対象要素について、定期的な調査または継続的監視を行い、セキュリティ上の不備に対し最新の修正が適用された状態を保つ必要がある。ソフトウエアプロジェクトのセキュリティ状況に影響する重要なアップグレードやパッチが明らかになったら、影響が及ぶユーザとオペレータに対象アプリケーションを更新させる計画を作成する。ソフトウエアプロジェクトの種類によって、計画の詳細は異なる可能性がある。

成果

- 開発チーム内で運用面の期待水準が明確に理解される
- インフラストラクチャに起因する優先度の高いリスクが、合意されたスケジュールに沿って緩和される
- ソフトウエアオペレータは、セキュリティ上重要なインフラストラクチャの保守のための高レベルの計画が立てられる

成功指標

- 直近 6 カ月以内に、50%超のプロジェクトの運用環境仕様が更新された
- 直近 6 カ月以内に、50%超のプロジェクトが関連する重要セキュリティパッチの最新リストを入手した

コスト

- 運用環境仕様の構築と保守に起因する、プロジェクトの継続的オーバヘッド
- 重要なセキュリティアップデートの監視とインストールに起因する、プロジェクトの継続的オーバヘッド

人員

- 開発者（1～2 日/年）
- アーキテクト（1～2 日/年）
- マネージャー（2～4 日/年）
- サポート/オペレータ（3～4 日/年）

関連レベル

- 運用準備 - 2

EH 2 環境の堅牢化

運用環境の堅牢化によってアプリケーション運用の信頼度を向上させる

成果
- 運用中システムのセキュリティ特性の細かな検証ができる
- インフラストラクチャのリスク緩和のためのスケジュールに関して公式の期待水準が設定される
- ソフトウエアプロジェクトの現在の運用状態が利害関係者に確実に認識される

追加成功指標
- 直近 12 カ月に、80%超のプロジェクトチームでパッチ管理プロセスについて概要が伝達された
- 直近 6 カ月以内に、80%超の利害関係者が最新のパッチ状態を認識した

追加コスト
- パッチの管理及び監視に起因する、組織の継続的オーバヘッド
- インフラストラクチャ監視ツールの構築またはライセンス

追加人員
- アーキテクト（1〜2 日/年）
- 開発者（1〜2 日/年）
- ビジネスオーナー（1〜2 日/年）
- マネージャー（1〜2 日/年）
- サポート/オペレータ（3〜4 日/年）

実施内容

A. 定期的なパッチ管理プロセスを確立する

重要なアップグレードやパッチをその時々に適用するプロセスから脱却して、より正式なプロセスへと移行し、運用環境において依存するソフトウエアに一貫してアップデートを適用するために継続的なプロセスを組織内に作成する。

このプロセスは、もっとも基本的な形では、セキュリティアップグレードやパッチがリリースされてから、それらが適用されるまでの間に経過する時間に関して保証することを目指すべきである。このプロセスを効率化するため、組織は通常、優先度の低いアップデートに関して長期の先送りを許容する。たとえば、重要なパッチに許容される猶予は最大 2 日間であるのに対し、優先度の低いパッチは最大 30 日間まで許容されるという具合である。

この実施内容は、サポート要員及び運用要員が主体となって実施すべきであるが、開発要員とも定期的にミーティングを実施し、過去の変更や予定されているアップグレードについて、プロジェクト全体で最新の情報が得られるようにする。

さらに、開発要員はソフトウエアプロジェクトが内部的に依存しているサードパーティコンポーネントのリストを共有し、サポート要員と運用要員がサードパーティ製コンポーネントの監視も行い、アップグレードの必要なタイミングを開発チームに知らせられるようにする。

B. 環境の基本水準に関する構成状態を監視する

ソフトウエアプロジェクトのインフラストラクチャを構成するさまざまなコンポーネントについてパッチだけでも監視及び管理するのは複雑なため、構成の状態についてシステムを自動的に監視するために、自動化ツールを利用するべきである。

この種の機能を提供する、商用及びオープンソースのツールがともに存在する。したがって、プロジェクトチームは組織の要求に合わせてソリューションを選択するべきである。一般的な選定基準として、配備及びカスタマイズのしやすさ、組織のプラットフォームや技術スタックへの適用可能性、変更管理や警告に関する内蔵機能、測定値の収集、傾向の追跡などが挙げられる。

ホスト及びプラットフォームのチェックに加え、監視の自動化をカスタマイズして、アプリケーション固有の状態チェックや構成の検証を実行する必要がある。サポート及び運用の担当者は、アーキテクトや開発者と連携してソフトウエアプロジェクトごとに最適な監視の程度を決定する。

最終的には、環境の構成状態を監視するソリューションが配備された後、予期しない警告や構成変更を収集し、それらの情報をプロジェクトの利害関係者が定期的に（毎週、または少なくとも四半期に 1 回）レビューする。

環境の堅牢化

EH 3

既知のベストプラクティスに照らしてアプリケーションの状況及び状態の妥当性を確認する

実施内容

A. 適切な運用保護ツールを指定し、配備する

運用環境にあるソフトウエアのセキュリティ保証を強化するために、ツールをさらに増やして、システム全体のセキュリティ状況を向上させることができる。運用環境は互いに大きく異なる可能性がある。そのため、利用する保護技術の妥当性は、当該プロジェクトの文脈で検討する必要がある。

一般に使用される保護ツールには、Web アプリケーションのファイアウォール、Web サービス用の XML セキュリティゲートウェイ、クライアント/組み込みシステム向けの改ざん防止及び難読化パッケージ、レガシーインフラストラクチャ向けのネットワーク侵入検知/防止システム、フォレンジックログ集計ツール、ホストベースの完全性検証ツールなどがある。

技術面の利害関係者は、組織及びプロジェクト固有の知識に基づき、サポート要員及び運用要員とともに、運用保護ツールを選択して、事業の利害関係者に推奨する。利害関係者は、リスクの軽減と実装コストを比較して投資の価値があると考えられる場合、試験運用、拡大展開及び継続的な保守の計画に同意すべきである。

B. 環境構成の監査プログラムを展開する

プロジェクトレベルの定期的な監査を実施する際に運用環境の強化に関連する成果物の検査を含めるよう、レビュー内容を拡張すること。監査では、運用環境の最新仕様に加え、前回の監査以降の最新のパッチの状況と履歴データを検査する。また、監視ツールを利用することにより、アプリケーションの構成管理と過去の変更に関する主要な要素を検証することもできる。監査では、当該ソフトウエアのアーキテクチャ向けに提供されているものを比較対象として、運用保護ツールの使用状況の検査も行うべきである。

インフラストラクチャの監査は、プロジェクトの初回リリースと配備後の任意の時点で行うことができるが、少なくとも 6 カ月に 1 回実施すべきである。レガシーシステムや開発が行われていないプロジェクトに対しても、インフラストラクチャの監査は依然として事業の利害関係者が実施し、レビューする必要がある。例外的なプロジェクトの運用を継続できるよう例外プロセスを策定する。ただし、発見内容緩和の時間枠を明示的に設定する。例外に該当するプロジェクト件数は全体の 20%以下になるようにする。

成果

- セキュリティを多層的にチェックすることで運用環境が強化される
- 運用上の保守とパフォーマンスに関する、確立され、測定に基づく目標値が設定される
- 外部に依存している要素の不備を経由する攻撃が成功する可能性が低減される

追加成功指標

- 直近 6 カ月以内に、80%超の利害関係者が、関連の運用保護ツールについて伝達された
- 直近 6 カ月以内に、75%超のプロジェクトがインフラストラクチャ監査に合格した

追加コスト

- 運用保護ソリューションの調査、選定
- 運用保護ツールの構築またはライセンス
- 保護ツールの保守に起因する、運用の継続的オーバヘッド
- インフラストラクチャ関連監査に起因する、プロジェクトの継続的オーバヘッド

追加人員

- ビジネスオーナー（1 日/年）
- マネージャー（1～2 日/年）
- サポート/オペレータ（3～4 日）

関連レベル

- ポリシー&コンプライアンス - 2

運用体制のセキュリティ対応

	OE 1	OE 2	OE 3
目的	**重要なセキュリティ関連データについて、開発チームとオペレータとの情報交換を可能にする**	**詳細な手続きを整備し、セキュアな運用の継続性に関する期待水準を引き上げる**	**セキュリティ情報の発信を義務付け、作成資料の完成度を確認する**
実施内容	A. 配備のために重要なセキュリティ情報を収集する B. 一般的なアプリケーション警告への対応手順を文書化する	A. リリースごとの変更管理手順を策定する B. 公式の運用セキュリティガイドを維持管理する	A. 運用情報の監査プログラムを展開する B. アプリケーションコンポーネントのコード署名を実行する
査定項目	◆ 大多数のソフトウエアリリースについてセキュリティノートを提供している ◆ ほとんどのプロジェクトについて、セキュリティ関連の警告とエラー条件が文書化されている	◆ ほとんどのプロジェクトで、明確な変更管理プロセスが利用されている ◆ プロジェクトチームは製品リリースごとに運用セキュリティガイドを提供している	◆ ほとんどのプロジェクトが、適切な運用セキュリティ情報について各リリースを調べる監査を受けている ◆ 一貫したプロセスを用いてコード署名をソフトウエアコンポーネントに対して定期的に実行している
成果	◆ 正しい運用をより深く理解することで、ソフトウエアセキュリティ体制の当座の改善が図られる ◆ オペレータとユーザがセキュアな配備を確実にする上での自身の役割を認識する ◆ セキュリティ上重要な情報に関する、ソフトウエア開発者とユーザの間のコミュニケーションが改善される	◆ ソフトウエアリリース時における、セキュリティ関連の変更点に関する詳細なガイドが提供される ◆ アプリケーションごとのセキュアな運用手続きに関して情報リポジトリが更新される ◆ 開発者、オペレータ及びユーザの間で運用に関する期待水準の認識が合う	◆ セキュリティ関連文書に対する期待水準が組織規模で理解される ◆ 配備と運用から得られたフィードバックに基づき、利害関係者がトレードオフについて適切に意思決定できる ◆ オペレータとユーザがそれぞれ独立にソフトウエアリリースの完全性を検証できる

運用体制のセキュリティ対応

重要なセキュリティ関連データについて、開発チームとオペレータとの情報交換を可能にする

実施内容

A. 配備のために重要なセキュリティ情報を収集する

プロジェクトチームは、ソフトウエア固有の知識とともに、セキュリティ関連の構成情報と運用情報を特定し、ユーザとオペレータにそれらの情報を伝達する。これにより、配備先でのソフトウエアの実際のセキュリティ状況を、プロジェクトチーム内の設計者の意図通りに機能させることができる。

この分析は、アーキテクトと開発者がソフトウエアに組み込むセキュリティ機能のリストを作成することから始める。このリストから、構成オプションに関する情報と構成オプションのセキュリティ上の影響も把握する。異なる複数の配備モデルを提供するプロジェクトの場合、それぞれのモデルの影響に関する情報がより的確にユーザとオペレータに伝わるよう、各配備モデルのセキュリティ区分に関する情報に注意を払う必要がある。

全体として、上記のリストは簡易的なものにしてより重要な情報を取得することを目指すべきである。リストを作成したら、プロジェクトチームと事業の利害関係者にそのリストをレビューしてもらい、同意を得る。さらに、情報がわかりやすく、実行可能であることを保証するために、選ばれたオペレータまたはユーザにリストをレビューしてもらうことも効果的である。プロジェクトチームはリストの情報をリリースごとにレビューして更新するものとし、また、少なくとも 6 カ月ごとに見直しと更新を行う。

B. 一般的なアプリケーション警告への対応手順を文書化する

ソフトウエアの動作形態を具体的に知っているプロジェクトチームは、ユーザ及びオペレータの注意を引く必要のあるもっとも重要なエラーメッセージと警告メッセージを特定する。その後、特定したそれぞれの事象について、ユーザまたはオペレータが各事象に対してとるべき適切な措置に関する情報を収集する。

ソフトウエアで発生する可能性のある事象は非常に多岐にわたる場合があるため、ソフトウエアのビジネス上の目的という観点から見た重要性に基づき、もっとも優先度の高い事象のセットを選択する。選択する事象には、セキュリティ関連の事象もすべて含まれるが、ソフトウエアの状態や構成に関連する重大なエラーや警告の場合もある。

次に実行する必要のある手順や事象の根本原因をユーザやオペレータが判断できるような情報を収集して、ユーザやオペレータに提供するべきである。これらの手続きはプロジェクトチームがレビューし、主要な製品リリースごとや 6 カ月おきに更新する必要がある。ただしリリースごとなど、もっと高い頻度で実施してもかまわない。

成果

- 正しい運用をより深く理解することで、ソフトウエアセキュリティ体制の当座の改善が図られる
- オペレータとユーザがセキュアな配備を確実にする上での自身の役割を認識する
- セキュリティ上重要な情報に関する、ソフトウエア開発者とユーザの間のコミュニケーションが改善される

成功指標

- 直近 6 カ月以内に、50%超のプロジェクトで配備に関するセキュリティ情報が更新された
- 直近 6 カ月以内に、50%超のプロジェクトで事象に関する運用手続きが更新された

コスト

- 配備に関するセキュリティ情報の保守に起因する、プロジェクトの継続的オーバヘッド
- 重要な運用手続きの保守に起因する、プロジェクトの継続的オーバヘッド

人員

- 開発者（1～2 日/年）
- アーキテクト（1～2 日/年）
- マネージャー（1 日/年）
- サポート/オペレータ（1 日/年）

運用体制のセキュリティ対応

詳細な手続きを整備し、セキュアな運用の継続性に関する期待水準を引き上げる

成果
- ソフトウエアリリース時における、セキュリティ関連の変更点に関する詳細なガイドが提供される
- アプリケーションごとのセキュアな運用手続きに関して情報リポジトリが更新される
- 開発者、オペレータ及びユーザの間で運用に関する期待水準の認識が合う

追加成功指標
- 直近 6 カ月以内に、50%超のプロジェクトの変更管理手続きが更新された
- 直近 6 カ月以内に、80%超の利害関係者が運用セキュリティガイドの状態について伝達された

追加コスト
- 変更管理手続きの保守に起因する、プロジェクトの継続的オーバヘッド
- 運用セキュリティガイドの保守に起因する、プロジェクトの継続的オーバヘッド

追加人員
- 開発者（1〜2 日/年）
- アーキテクト（1〜2 日/年）
- マネージャー（1 日/年）
- サポート/オペレータ（1 日/年）

関連レベル
- 環境の堅牢化 - 1

実施内容

A. リリースごとの変更管理手順を策定する

ソフトウエアにおける変更点について、より正式な形でユーザとオペレータに最新情報を伝えるため、各リリースにアップグレードインストールと初回インストールに対する変更管理手順を用意しておく。つまり、配備を成功させ、過剰なダウンタイムやセキュリティ状況の悪化を引き起こさないようにすることを目的として、期待される付随手順を入手する。

開発時にこれらの手順を確立するには、プロジェクトチームは配備に影響を及ぼしうる関連項目を把握するために簡易的な内部プロセスを組み立てる。このプロセスを開発サイクルの初期に導入し、この情報を要件定義、設計、実装のフェーズで特定してすぐに保持できるようにすることが効果的である。

各リリース前に、プロジェクトチームは、完全性と実現可能性についてリスト全体をレビューする。一部のプロジェクトについては、ある特定のリリースで生じることになる膨大な変更手続きによって、特別な対処（配備時の誤りを回避する自動アップグレードスクリプト作成など）が必要となる場合がある。

B. 公式の運用セキュリティガイドを維持管理する

プロジェクトチームは、重要なソフトウエア事象について得た情報や各事象に対処する手続きを手始めに、ユーザとオペレータが知る必要のあるセキュリティ関連情報のすべてを盛り込んだ正式なガイドを作成し、維持するべきある。

このガイドは最初に、セキュリティ関連の構成オプション、事象対応手順、インストールとアップグレードのガイド、運用環境仕様、配備環境に関するセキュリティ関連の仮定など、システムに関する既知の情報を元に作成する。その後、ガイドを拡張した正式な運用セキュリティガイドでは、ユーザとオペレータの大多数が持ちうるすべての疑問に答えるだけの詳細が盛り込まれるよう、既知の情報それぞれについて詳細に記述する。大規模なシステムや複雑なシステムでは、この作業は容易ではないことが考えられるため、プロジェクトチームは事業の利害関係者と連携して文書化の適切なレベルを決定する。さらに、セキュリティの強化につながる可能性のある、配備に関する推奨事項を文書化するべきである。

運用セキュリティガイドは、最初に作成した後はプロジェクトチームがレビューし、リリースごとに更新する。

運用体制のセキュリティ対応

セキュリティ情報の発信を義務付け、作成資料の完成度を確認する

実施内容

A. 運用情報の監査プログラムを展開する

プロジェクトレベルの定期的な監査を実施する際に、セキュリティの運用準備に関連する成果物資料を検査対象に含めるようにレビュー内容を拡張する。プロジェクトに、ソフトウエアの詳細に関する最新の運用セキュリティガイドが完備していることを点検する。

これらの監査は、開発サイクルの終盤でリリースが近づいた時点で開始すべきであるが、監査が完了して合格するまではリリースしてはならない。レガシーシステムや活動していないプロジェクトの場合には、この種の監査を実施し、発見結果に対処し、監査への適合性を検証する作業を 1 回だけ行い、以後は運用準備に関する監査を新たに行う必要はない。

監査結果は、リリース前に事業の利害関係者とともにレビューする。監査に合格しなかったプロジェクトがリリースへ進めるように、例外プロセスを策定するべきであるが、このようなプロジェクトには、監査結果の緩和策の具体的なスケジュールを定める必要がある。例外は、有効な全プロジェクト件数の 20%以下になるようにする。

B. アプリケーションコンポーネントのコード署名を実行する

コード署名は特別な目的のソフトウエアに使用することが多いが、コード署名を利用することで、ユーザやオペレータはモジュールやリリースの真正性を暗号学的に検証でき、ソフトウエアの完全性を確認できるようになる。ソフトウエアモジュールに署名することにより、プロジェクトチームは、運用環境に配備されたソフトウエアに破損や改変がないという強い確証をこれまで以上に持って、配備したアプリケーションを運用することが可能となる。

コードへの署名では、組織の署名証明書の管理に関するオーバヘッドが生じる。組織は、署名鍵の継続的な秘密性を確保するため、安全な鍵管理プロセスに従わなければならない。暗号鍵を扱う場合、プロジェクトの利害関係者は、鍵ローテーションや鍵の漏洩、紛失など、暗号化に関する運用面の一般的な問題を扱う計画も考慮しなければならない。

コード署名はすべてに適しているわけではないため、アーキテクトと開発者は、セキュリティ監査員及び事業の利害関係者とともに、ソフトウエアのどの部分に署名すべきかを決定するべきである。プロジェクトが進むのに応じて、署名する部分のリストをリリースごとにレビューする。特に、新規モジュールを追加したり、以前に署名したコンポーネントに変更を加えたりする場合にレビューする必要がある。

成果

- セキュリティ関連文書の期待水準が組織規模で理解される
- 配備と運用から得られたフィードバックに基づき、利害関係者がトレードオフについて適切に意思決定できる
- オペレータとユーザがそれぞれ独立にソフトウエアリリースの完全性を検証できる

追加成功指標

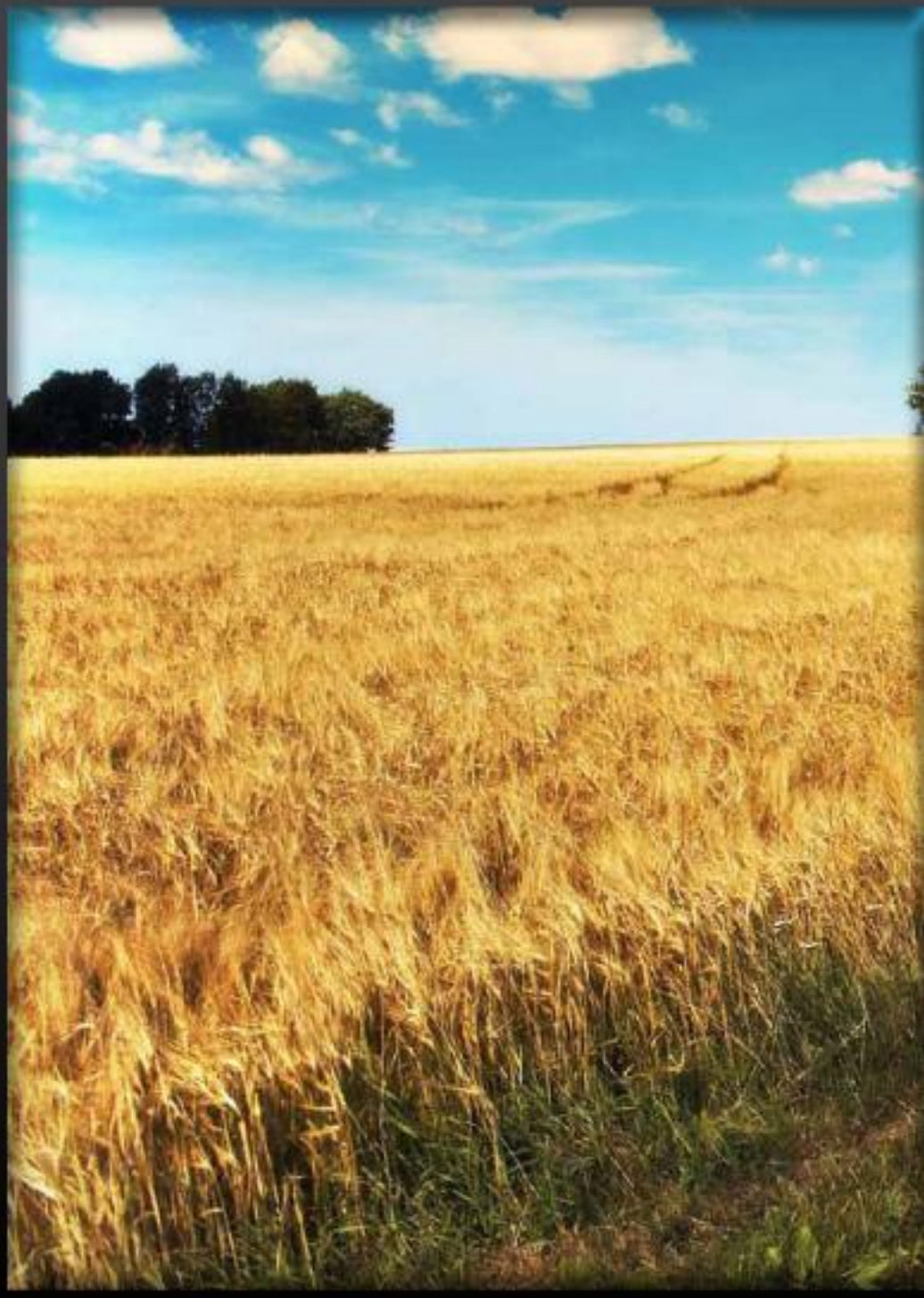
- 直近 6 カ月以内に、80%超のプロジェクトの運用セキュリティガイドが更新された
- 直近 6 カ月以内に、80%超の利害関係者がコード署名の選択肢と状態について説明された

追加コスト

- 運用ガイドの監査に起因する、プロジェクトの継続的オーバヘッド
- コードの署名証明書の管理に起因する、組織の継続的オーバヘッド
- コードモジュールの識別と署名に起因する、プロジェクトの継続的オーバヘッド

追加人員

- 開発者（1 日/年）
- アーキテクト（1 日/年）
- マネージャー（1 日/年）
- セキュリティ監査員（1〜2 日/年）



ケーススタディ

サンプルシナリオの実地検証

この節では、ある特定のビジネスケースにおけるシナリオをいくつか示して、SAMM の適用を解説する。このケーススタディでは、ロードマップのテンプレートをガイドとして利用し、ある組織がセキュリティ保証プログラムを作成するときに、どのようにしてベストプラクティスを採り入れ、組織固有のリスクを考慮するかを例示する。

VirtualWare 社

ケーススタディ：中規模の独立系ソフトウエアベンダ

事業の概要

VirtualWare 社は、統合型仮想化アプリケーションプラットフォーム市場のリーダーであり、組織向けにアプリケーションインターフェイスを 1 つの環境に集約する技術を提供している。同社の技術は、Microsoft、Apple、Linux プラットフォームなど、複数の環境向けのサーバアプリケーション及びデスクトップクライアントとして提供されている。

VirtualWare 社は中規模（従業員数 200～1000 人）の組織で、世界各国に事業を展開しており、ほとんどの主要国に支社がある。

組織

VirtualWare 社は 8 年以上にわたり、主力事業としてソフトウエアプラットフォーム開発を行っている。この間、同社は Web インターフェイスの使用を最小限に抑えることにより、一般的な Web の脆弱性に起因するリスクを限定してきた。VirtualWare 社のプラットフォームの大部分は、サーバシステムまたはデスクトップ上で実行されるシッククライアント（thick-client）を通じて稼働する。

最近 VirtualWare 社は、Web 技術を通じてクライアント/サーバインターフェイスを提供する新しいプロジェクトをいくつか立ち上げた。これを機に同社は、Web を介したよくあるさまざまな攻撃を踏まえ、同社に将来影響を及ぼす可能性のある潜在的脅威に確実に対応するために、ソフトウエアセキュリティ戦略の見直しを行った。

これまで同社は、アプリケーションコードの基本的なレビューを行いながら、セキュリティよりもパフォーマンスと機能を重視してきた。VirtualWare 社の開発者は、いくつかのコード品質分析ツールを使用してバグを特定し、コード内でバグに対処してきた。

上位の経営陣はこのことを念頭におきながら、同社のアプリケーションにおけるセキュリティの最新状態をレビューし、アプリケーションに存在する脆弱性を特定して、排除し、防止する最良の方法を決定するという戦略的目標を設定した。

環境

VirtualWare 社は、Java、C++及び Microsoft .NET が混合した技術に基づく仮想化技術を開発している。同社の中核的なアプリケーション仮想化技術は C++で開発されており、バグとセキュリティについて数回のレビューが行われたが、現時点では既知または未知のセキュリティ上のバグを特定して修正するための正式なプロセスは存在しない。

VirtualWare 社は、自社の Web 技術を Java でサポートすることを選択したが、バックエンドシステムは Microsoft と C++の技術を用いて構築されている。この新しい Web インターフェイスに注力する開発チームは、主に Java 開発者で構成されている。

VirtualWare 社は、300 人以上の開発者と、担当プロジェクトに基づいて複数のチームに分かれた要員を抱えている。12 のチームがあり、1 チームあたり 20～40 名の開発者が属している。各チーム内ではソフトウエアセキュリティの経験は最低限のレベルしかなく、コードの基本的な評価は上級開発者が行うものの、セキュリティは組織内の重要な目標とは考えられていない。

VirtualWare 社内の各チームが、それぞれ異なる開発モデルを採用している。現在主に使用されているのは、Agile SCRUM と反復型の Waterfall の 2 つの手法である。IT 部門またはプロジェクトアーキテクトによるソフトウエアセキュリティに関する指導は最低限のレベルしか行われていない。

主な課題

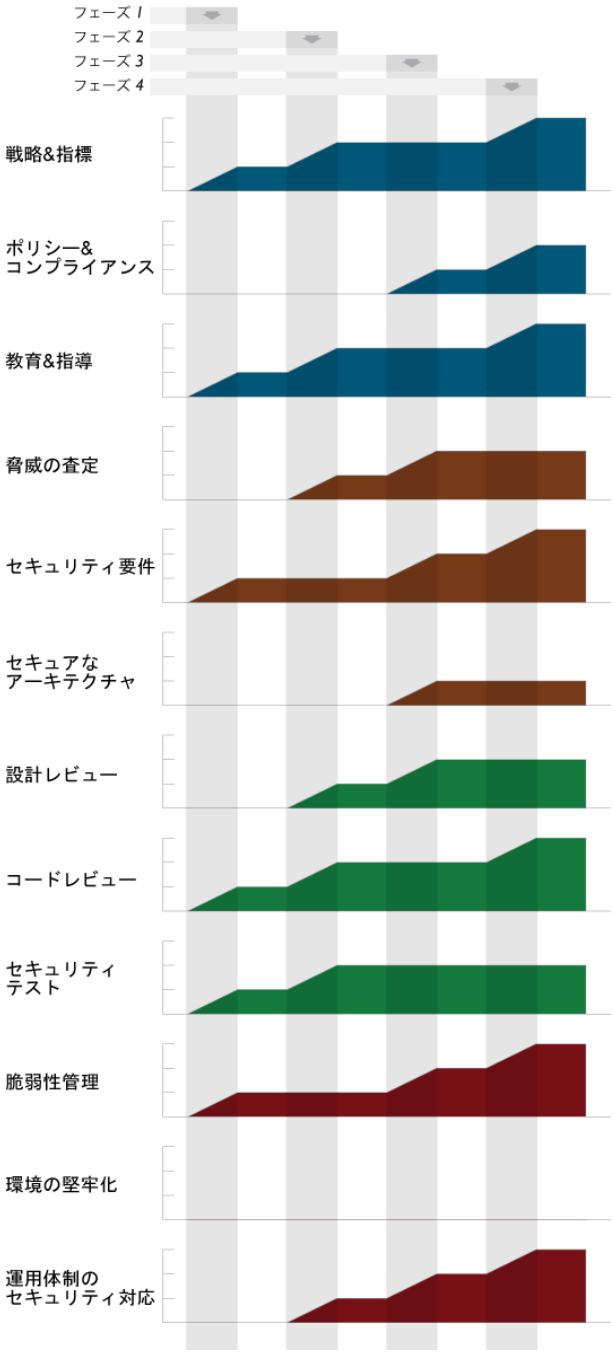
- 競合企業に勝る競争力を確実に維持するために、アプリケーション機能を短期でリリースすること
- ソフトウエアセキュリティの概念に関する知見が限られていること。現時点では、セキュリティ関連作業にかかわる活動は最小限である
- 開発者が離職し、より経験の浅い開発者に入れ替わっている
- アプリケーション内で複数の技術が使われており、最初に構築されて以来更新されていないレガシーアプリケーションが存在する
- 既存のセキュリティ状況や組織が直面するリスクについて把握されていない

VirtualWare 社は、新しい Web アプリケーションが確実に安全に顧客に提供されることに注力したいと考えた。そのため、セキュリティ保証プログラムの実施における最初の重点項目は、開発チームの教育と意識向上、及びセキュアコーディングとテスト基準に関する基本的な技術指導に置かれた。

VirtualWare 社はこれまで、support@virtualware.net というアドレスを通じて、バグへの対応要請とセキュリティ脆弱性の情報を受け取っていた。しかしこのアドレスはサポート全般に使われていたため、これまでの要請は必ずしも組織内の適切なチームに割り振られず、適切に処理されなかった。そのため、正式なセキュリティ脆弱性対応プログラムを実施する必要性も明らかにされた。

実施計画

組織におけるセキュリティ保証プログラムの採用は長期的な計画であり、開発者の文化や、業務アプリケーションの開発・提供の業務プロセスに大きな影響を与える。この計画は、12 カ月の期間をかけて導入することが設定されているが、同社の規模からすると、この期間での導入実施は比較的容易である。



VirtualWare – フェーズ 1

フェーズ 1（0～3 カ月目）－意識向上&計画

VirtualWare 社は、アプリケーションセキュリティに関して自社に対する脅威の知識や意識が乏しく、セキュアコーディングの経験も少ないことを明らかにした。VirtualWare 社内での戦略展開における最初のフェーズは、開発者をトレーニングすることと、現時点のセキュリティ脆弱性を特定するための指針とプログラムを実施することに焦点を合わせた。

VirtualWare 社内の開発チームは、セキュアコーディング技術の知見に乏しかったため、最初のトレーニングプログラムとして、社内の開発者が防御的プログラミングの技術を習得できるようなプログラムが開発された。

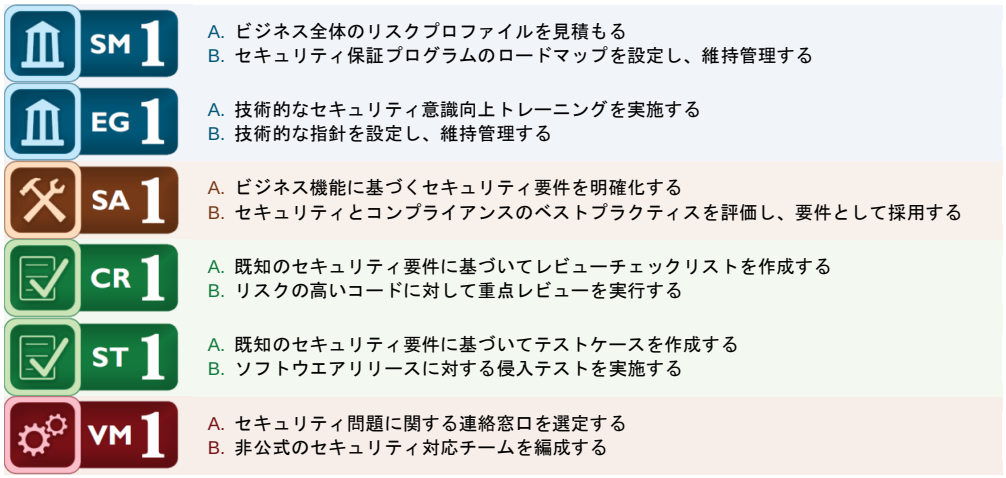
社内に 300 名以上の開発者を抱え、複数のプログラミング言語に対応している VirtualWare 社の主な課題は、セキュアコーディングの考え方の基礎を開発者に教育するのに十分に技術的な教育プログラムを用意することであった。この最初の教育コースの目的は、主にコーディング技術とテストツールについて習得することであった。社内で開発され提供されたこのコースは、1 日でセキュアコーディングの基本を取り上げていた。

VirtualWare 社は、自社のいくつかのアプリケーションに脆弱性があるものの、既存の脆弱性を特定して妥当な期間でリスクに対処する現実的な計画がないことを認識していた。基本的なリスク評価の手法が採用され、VirtualWare 社は既存のアプリケーションプラットフォームのレビューを行った。

このフェーズには、開発チームのセキュリティツールを強化するためのいくつかの概念を実装することも含まれていた。開発チームは、品質評価に利用できるツールをすでにいくつか保有していた。さらに、コードレビューとセキュリティテストツールの調査が行われた。

目標設定

プロジェクトのこのフェーズの間、VirtualWare 社は SAMM における次の対策と実施内容を実施した。



これらの成熟度レベルを達成するため、VirtualWare 社はこの展開のフェーズにおいていくつかの計画を導入した。採用された取り組みは以下のとおりである。

- 全開発者を対象とした 1 日間のセキュアコーディングコース（高レベル）
- 社内で使用される技術のアプリケーションセキュリティに関する技術指導白書の作成
- リスクプロセスの作成、アプリケーションプラットフォームに対する高水準のビジネスリスク評価、ビジネスリスクの見直し
- 開発者向けの初期の技術的ガイドラインと基準の策定
- 組織への甚大なリスクのあるアプリケーションプラットフォームでの短いコードレビュー
- プロジェクトのテストケースと使用事例の作成、及びアプリケーションに照らしたこれら事例の評価
- アプリケーションセキュリティの取り組みに対する役割の任命
- セキュリティ保証プログラムの次のフェーズで使用する戦略的ロードマップの草案の作成

VirtualWare 社内の専門知識が限られているため、同社は外部のセキュリティコンサルティンググループと契約し、トレーニングプログラムの作成、及び自社用の脅威のモデル化と戦略ロードマップ作成の支援を受けた。

このフェーズで直面した大きな課題の 1 つは、300 人の開発者全員に 1 日のトレーニングコースを受けさせることであった。これを達成するため、VirtualWare 社はコースを 20 日間実施し、各チームから一度に少人数の開発者がコースに出席するようにした。これにより、トレーニング期間における人的資源への全体的な影響が軽減された。

プロジェクトのこのフェーズの間、VirtualWare 社はリスクレビュープロセスの採用と、組織にとってのビジネスリスクの見直しに重点的にリソースを投入した。これらの作業に相当の労力が費やされたが、VirtualWare 社が進める次の段階と、同社が直面するビジネスリスクを確実にすり合わせる上で非常に重要な作業であった。

トレーニングプログラムに関して経営陣へ寄せられた社内の開発者からの反応は、ほとんどが良いものばかりだった。初期のトレーニングは、詳細ではないものの、日常のセキュアコードの記述ですぐに役立つ可能性のある基本技能を習得できたと開発者は感じた。

実施コスト

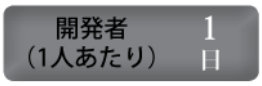
プロジェクトのこのフェーズでは、多くの内部リソースとコストが費やされた。このフェーズに関連するコストは 3 種類あった。

内部のリソース要求

このフェーズのアプリケーションセキュリティの取り組みについて、内容の作成、ワークショップ、及びレビューでは内部リソースの労力が費やされた。延べ日数で示した、役割ごとに費やされた労力は次のとおりである。

トレーニングのリソース要求（1 人あたりのトレーニング期間）

VirtualWare 社内の各開発者は、トレーニングコースへの参加が求められたため、開発者 1 人あたり 1 日がアプリケーションセキュリティプログラムに割り当てられた。



外部委託リソース

VirtualWare 社内の知識が不足しているため、外部リソースを利用し、内容作成の支援、及び開発者に向けたトレーニングプログラムの作成/提供を行った。

VirtualWare – フェーズ 2

フェーズ 2（3～6 カ月目）－教育&テスト

VirtualWare 社はフェーズ 1 において、同社のいくつかのアプリケーションに、外部の脅威により悪用される可能性のある脆弱性が含まれていることを確認した。そのため、このフェーズの主要な目標の 1 つは、基本的なテスト及びレビュー技術を導入し、脆弱性を特定してコード内で対処することであった。

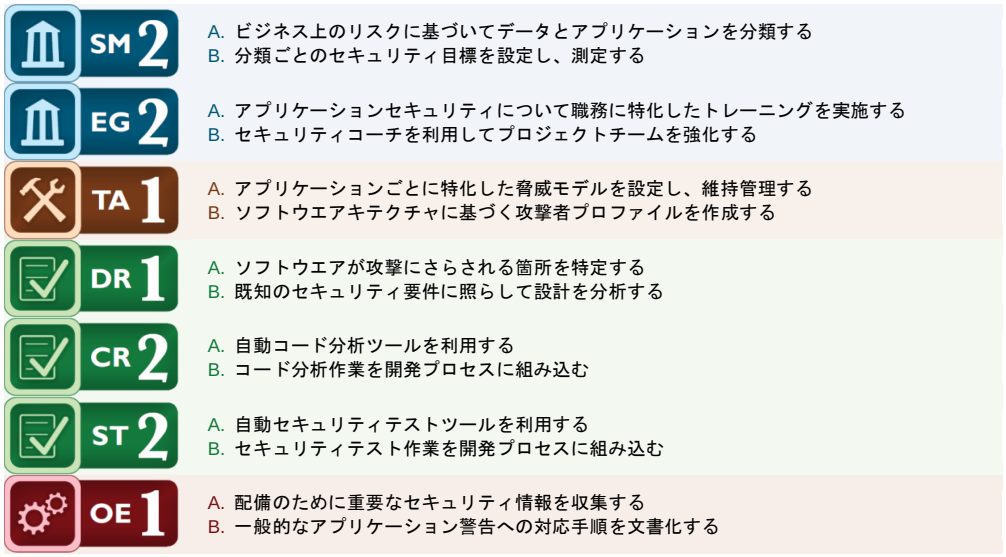
コードのカバレッジ及び弱点の発見を支援する自動ツールの導入が、この実装のフェーズにおけるもっとも大きな課題の 1 つであった。これまで伝統的に、開発者は非常に苦労しながら自動化ツールを使用していたため、新たなツールを導入することは大きな挑戦ととらえられていた。

社内での自動化ツールの展開を確実に成功させるため、VirtualWare 社は段階的な展開を進めた。自動化ツールはまず上位のチームリーダに与えられ、その他の開発者は一定期間をかけて利用を始める。チームはツールの採用を促されたが、ツールの使用に関する正式なプロセスは導入されなかった。

実施のこのフェーズでは、より正式な教育と意識向上プログラムの導入も行われた。前フェーズのトレーニングを受けた開発者からは、Web サービス及びデータ検証の領域に関する、より具体的なトレーニングの要請があった。これら 2 つの領域に関する、新たな 6 時間のトレーニングコースが開発された。VirtualWare 社はアーキテクトとマネージャー向けの追加のトレーニングプログラムも実施し、社内における意識向上のキャンペーンも展開した。

目標設定

プロジェクトのこのフェーズの間、VirtualWare 社は SAMM における次の対策と実施内容を実施した。



これらの成熟度レベルを達成するため、VirtualWare 社はこの展開のフェーズにおいていくつかの計画を導入した。採用された取り組みは以下のとおりである。

- QA テスト担当者、マネージャー、アーキテクトを対象とした追加の教育&トレーニングコース
- データ資産の分類及びセキュリティ目標の設定
- リスク評価方法を発展させ、攻撃ツリーと攻撃プロファイルを含んだ脅威モデル化手法を構築
- アプリケーションプラットフォームごとのセキュリティ要件のレビュー及び特定
- 既存のアプリケーション及び新規コードに対するコードカバレッジとセキュリティ分析を支援する自動ツールの導入
- 既存の侵入テストプログラムのレビュー及び強化
- セキュリティテストを開発工程に組み込むための、既存のソフトウエア開発サイクルの拡張

VirtualWare 社はアプリケーションセキュリティに関する既存のトレーニングプログラムを改良し、ビジネスアプリケーションのセキュリティ意識向上プログラムとして、より小規模な技術的でないバージョンを用意した。このトレーニングプログラムは短めの 4 時間のコースで、対象者は同社のマネージャー、ビジネスオーナーまで拡大された。

既存のコードレビューと侵入テストプログラムに関して概略レベルでレビューを行ったところ、アプリケーションのセキュリティ脆弱性に関してプロセスが不十分であり、より適切なテストを実施して結果を得るには拡張する必要があることが分かった。チームは、侵入テストとコードレビューを実行する新しいプログラムの実施に着手した。このプログラムの一部として、プログラムチームの上級開発者には、アプリケーションに対する概略レベルのソースコードレビューを実施するために一人あたり約 4 日間が割り当てられた。

VirtualWare 社の経営陣は、インフラストラクチャとアプリケーションが密接に統合されていることを理解し、このフェーズの間、アプリケーションプラットフォーム（インフラストラクチャ）の運用側に対するレビューが行われた。このフェーズでは、配備済みの推奨ハードウェアとアプリケーションインターフェイスの間のインフラストラクチャ要件とアプリケーション統合機能に着目した。

このフェーズでは、アプリケーションセキュリティのための戦略的ロードマップと方法論が、プロジェクトチームによってレビューされた。このレビューと更新の目的は、データ資産を正式に分類し、データ資産とアプリケーションに対応するビジネスリスクの適切なレベルを設定することであった。プロジェクトチームは、設定したレベルに基づき、これらアプリケーションのセキュリティ目標を設定することが可能となった。

実施コスト

プロジェクトのこのフェーズでは、多くの内部リソースとコストが費やされた。このフェーズに関連するコストは 3 種類あった。

内部のリソース要求

このフェーズのアプリケーションセキュリティの取り組みについて、内容の作成、ワークショップ、及びレビューでは内部リソースの労力が費やされた。延べ日数で示した、役割ごとに費やされた労力は次のとおりである。

トレーニングのリソース要求（1 人あたりのトレーニング期間）

VirtualWare 社内の要員が新たにトレーニングコースに参加を求められた。そのため、アプリケーションセキュリティのトレーニングについて、いくつかの役割に時間が割り当てられた。

外部委託リソース

VirtualWare 社内の知識が不足しているため、外部リソースを利用し、内容作成の支援、及び開発者に向けたトレーニングプログラムの作成/提供を行った。

VirtualWare – フェーズ3

フェーズ3（6～9 カ月目）－アーキテクチャ&インフラストラクチャ

VirtualWare 社におけるセキュリティ保証プログラム実施の第 3 フェーズは、前の実施フェーズを前提として構築されており、リスクのモデル化、アーキテクチャ、インフラストラクチャ、及び運用準備能力に重点が置かれている。

このフェーズの主な課題は、アプリケーションプラットフォームと同社の運用側との間の緊密な統合の実現であった。前のフェーズでは、VirtualWare 社のチームは脆弱性管理及びアプリケーションセキュリティの運用面を学んだ。今度のフェーズでは、VirtualWare 社はこれら領域の次のフェーズを採用し、明確なインシデント対応プロセスと詳細な変更管理手続きを導入した。

VirtualWare 社はこの実施において 2 つの新たな領域に着手した。VirtualWare 社はコンプライアンス遵守の影響は受けないが、同社の顧客の多くから、同社のプラットフォームがコンプライアンス遵守に役立ちうるかどうかに関する問い合わせが寄せられ始めた。VirtualWare 社内の少人数のチームが、コンプライアンス遵守に関する推進要因を特定し、推進要因のチェックリストを作成する作業に着手した。

前のフェーズで、VirtualWare 社は脆弱性のレビューと特定に関して、いくつかの新しい自動化ツールを導入した。今度のフェーズでは焦点が置かれていないが、開発チームは新たなツールを採用し、新しいツールの使用によるメリットがグループ内で得られ始めていると報告した。

目標設定

プロジェクトのこのフェーズの間、VirtualWare 社は SAMM における次の対策と実施内容を実施した。

PC 1	A. 外的なコンプライアンス推進要因を特定する B. コンプライアンスの指針を設定し、維持管理する
TA 2	A. プロジェクトごとの悪用ケースモデルを設定し、維持管理する B. 脅威測定のための重み付けシステムを導入する
SR 2	A. リソースと機能に関するアクセス制御マトリックスを作成する B. 既知のリスクに基づくセキュリティ要件を策定する
SA 1	A. 推奨ソフトウエアフレームワークのリストを維持管理する B. セキュリティに関する原則を設計に対して明示的に適用する
DR 2	A. セキュリティの仕組みが完備されているかどうかを検査する B. プロジェクトチーム向け設計レビューサービスを展開する
VM 2	A. 一貫したインシデント対応プロセスを確立する B. セキュリティ問題の情報公開プロセスを採択する
OE 2	A. リリースごとの変更管理手順を策定する B. 公式の運用セキュリティガイドを維持管理する

これらの成熟度レベルを達成するため、VirtualWare 社はこの展開のフェーズにおいていくつかの計画を導入した。採用された取り組みは以下のとおりである。

- 社内プロジェクトを対象としたセキュリティ要件とセキュアアーキテクチャに関する技術ガイドの定義と発行
- コンプライアンス及び規制に関する要件の特定と文書化
- アプリケーションインフラストラクチャのセキュリティガイドラインの内容の選定及びガイドライン作成
- 承認された開発フレームワークの定義済みリストの作成
- VirtualWare 社内で使用する、既存の脅威モデル化プロセスの拡張
- インシデント対応計画の採用及びセキュリティ情報開示プロセスの作成
- 全プロジェクトを対象とした変更管理手続き及び正式ガイドラインの導入

（前フェーズからの）開発用の自動化ツール導入のタイミングに合わせるため、セキュアコーディング技法についての正式な技術手引書が VirtualWare 社に導入された。これらの手引書は、言語とテクノロジに関する具体的な技術文書であり、各関連言語/アプリケーションにおけるセキュアコーディング技法の手引を提供するものであった。

VirtualWare 社は、教育プログラムと意識向上プログラム、技術手引書、そして開発者を支援する自動化ツールなどの取り組みの組み合わせの結果、アプリケーションの実稼働バージョンに提供されるコードには目に見える違いが現れ始めた。ツールに対する開発者からの反応は良く、このプログラムの下で開発者への教育が行われた。

VirtualWare 社では初めて、プロジェクトチームが同社のアプリケーションプラットフォームのセキュリティと設計の責任を負うことになった。このフェーズの間、各チームによって、ベストプラクティスに照らして正式なレビュープロセスと検証が行われた。いくつかのチームが、セキュリティとビジネス設計の両方に関して、レビューを要するギャップを明らかにした。ギャップへの対処を確実にするための正式な計画が実行に移された。

正式なインシデント対応計画と変更管理手続きが、プロジェクトのこのフェーズで導入された。このプロセスは実施が難しく、組織風土や事業の運営側への影響が大きかったため、VirtualWare 社のチームは当初、このプロセスに苦闘していた。しかし、時間が経つにつれ、チームメンバ各自がこの新しいプロセスの価値を認識し、変更は導入期間の間にチームに受け入れられた。

実施コスト

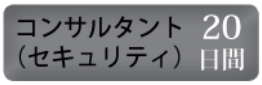
プロジェクトのこのフェーズでは、多くの内部リソースとコストが費やされた。このフェーズに関連するコストは 2 種類あった。

内部のリソース要求

このフェーズのアプリケーションセキュリティの取り組みについて、内容の作成、ワークショップ、及びレビューでは内部リソースの労力が費やされた。延べ日数で示した、役割ごとに費やされた労力は次のとおりである。

外部委託リソース

VirtualWare 社内の知識が不足しているため、外部リソースを利用し、内容作成の支援、及びプロセス、ガイドライン、支援チームの作成/提供を行った。



VirtualWare – フェーズ 4

フェーズ 4（9～12 カ月目）－ガバナンス&運用セキュリティ

VirtualWare 社におけるセキュリティ保証プログラム導入の 4 番目のフェーズは、前のフェーズを継続しており、同社内の既存のセキュリティ機能を強化する。これまでに、VirtualWare 社はアプリケーションの開発と保守が安全に行われるようにするため、アプリケーションセキュリティに関する重要なプロセスと仕組みをいくつか実施してきた。

今度のフェーズの中核は、整合及びガバナンスの規律の強化である。これら 3 つの機能は、長期にわたる効果的なアプリケーションセキュリティ戦略の基盤において重要な役割を果たす。完成された教育プログラムが導入され、同時に VirtualWare 社の長期にわたる戦略的ロードマップが実行に移される。

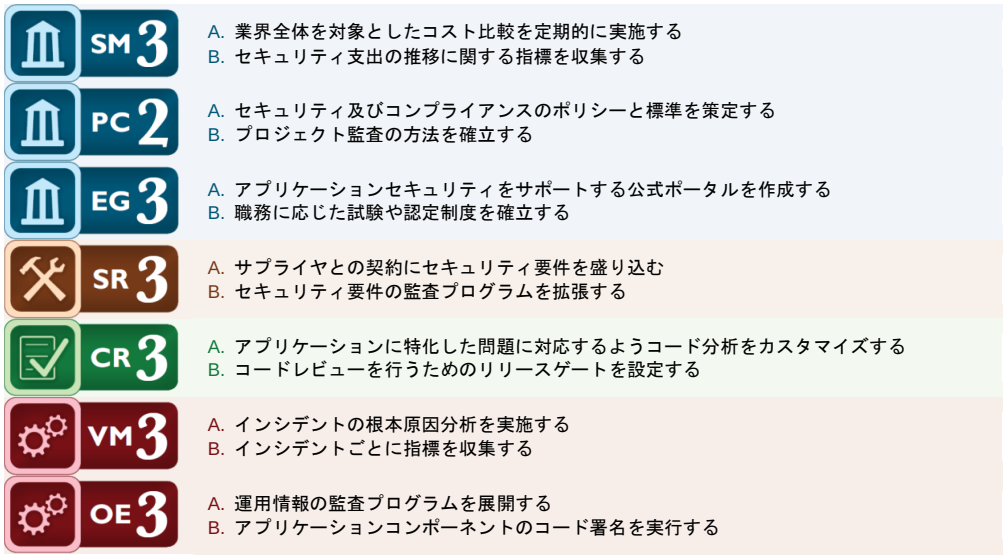
このフェーズのその他の重点は、導入の運用面に置かれている。VirtualWare 社の経営陣は、インシデント対応計画及び専用の変更管理プロセスが長期的戦略にとって重要であることをすでに明らかにしている。

VirtualWare 社はこのフェーズを、同社の長期的未来への足がかりとみなしていた。このフェーズでは、同社はそれまでのフェーズで設けてきた基本構成要素を結合するための最終的な手段を数多く導入した。これにより、同社のアプリケーションプラットフォームにとってもっとも安全な結果が得られるよう、プロセス、概念、管理体制が社内で機能し続けることが、長期的に確実となる。

VirtualWare 社はこのフェーズを選択して、同社の顧客にアプリケーションセキュリティに関する新たな取り組みを紹介し、アプリケーションセキュリティ、アプリケーションの安全な配備、同社のアプリケーションに含まれる脆弱性レポートに関して顧客に一連のプログラムの詳細を提供した。これらプログラムの主要な目標は、VirtualWare 社のアプリケーションはセキュリティを念頭において構築されており、同社は自らの技術を用いて顧客のアプリケーション環境を確固たるものにする支援が可能である、という確信を浸透させることである。

目標設定

プロジェクトのこのフェーズの間、VirtualWare 社は SAMM における次の対策と実施内容を実施した。



これらの成熟度レベルを達成するため、VirtualWare 社はこの展開のフェーズにおいていくつかの計画を導入した。採用された取り組みは以下のとおりである。

- すべてのプロジェクトに対して明確に定められたセキュリティ要件とテストプログラムの作成
- インシデント対応計画の作成と実装
- アプリケーションに関する既存の警告手続きのレビュー及び事象把握のプロセスの文書化
- アプリケーションセキュリティ配備に関する顧客向けセキュリティホワイトペーパーの作成
- プロジェクトの既存のセキュリティ関連支出のレビュー、及びセキュリティに関して各プロジェクトに適切な予算が割り当てられているかどうかの判断
- アプリケーションの役割に関する、最終的な教育/認識プログラムの実装
- 組織のためのアプリケーションセキュリティ戦略に関する長期的ロードマップの完成

これまでのフェーズにおいて、VirtualWare 社は自社のコードに関して見つかった脆弱性を顧客が報告できるように、顧客向けに正式なインシデント対応計画をリリースした。今度のフェーズの間、VirtualWare 社は報告された脆弱性がもたらす結果を踏まえ、問題が生じた理由と経過を考査し、報告された脆弱性の中から特定された共通のテーマを判断するため、一連の報告を試みた。

アプリケーションを内部だけでなく顧客のネットワーク上でも安全に配備するための継続的な取り組みの一環として、VirtualWare 社は推奨環境の業界標準に基づいた堅牢化に関して一連の白書を作成し、顧客に提供した。これらのガイドラインの目的は、アプリケーション配備のための最善策を示して顧客を支援することにある。

このフェーズの間、VirtualWare 社はコンピュータベースの短いトレーニングモジュールを作成し、既存及び新規の開発者がアプリケーションセキュリティにおける自身の技能を維持できるようにした。さらに、「アプリケーション」に関係するすべての役割が 1 年につき 1 日のコースを受けることが義務付けられた。これは、開発者に与えられた技能が失われないようにし、新たな開発者は在職中に技能を向上できるようにするため徹底された。

VirtualWare 社に導入された最終的機能の 1 つが、「現状」のギャップの評価とレビューを完了させ、直近 12 カ月にどれくらい効果が上がったかを判断することであった。この間、プログラムに関する短いアンケートが関わったチームメンバ全員に送付され、SAMM に照らした基本レビューが行われた。このレビューの間に明らかになった短所と長所は VirtualWare 社の最終的な戦略ロードマップとして文書化され、同社の次の 12 カ月の戦略が定められた。

実施コスト

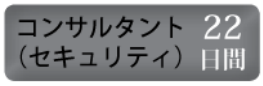
プロジェクトのこのフェーズでは、多くの内部リソースとコストが費やされた。このフェーズに関連するコストは 2 種類あった。

内部のリソース要求

このフェーズのアプリケーションセキュリティの取り組みについて、内容の作成、ワークショップ、及びレビューでは内部リソースの労力が費やされた。延べ日数で示した、役割ごとに費やされた労力は次のとおりである。

外部委託リソース

VirtualWare 社内の知識が不足しているため、外部リソースを利用し、文書化、プロセス、ワークショップを含め、このフェーズの実装を支援した。



VirtualWare – 進行中

継続中（13 カ月目以降）

直近 12 カ月において、VirtualWare 社はいくつかのトレーニングプログラムや教育プログラムの導入から着手し、内部用ガイドラインとポリシーの策定までを行った。VirtualWare 社は、セキュリティ保証プログラム導入の最終フェーズにおいて、顧客のアプリケーションプラットフォームのセキュリティを強化するため、外部に情報を提供し、顧客との連携を開始した。

VirtualWare 社の経営陣は、社内で開発したソフトウエアのセキュリティを確実に確保すること、同社のセキュリティへの取り組みが市場に確実に認識されること、アプリケーションプラットフォームのセキュリティ確保の面で顧客を確実に支援するという独自の指令を出した。

これら経営上の目標を達成するため、最初の 12 カ月は VirtualWare 社内の効果的戦略の道筋を定め、最終的には顧客のアプリケーション環境のセキュリティ確保において顧客の支援を開始した。VirtualWare 社は今後のため、同社が古い習慣に陥ることのないよう、数多くの取り組みに着手した。具体的なプログラムのいくつかは次のとおりである。

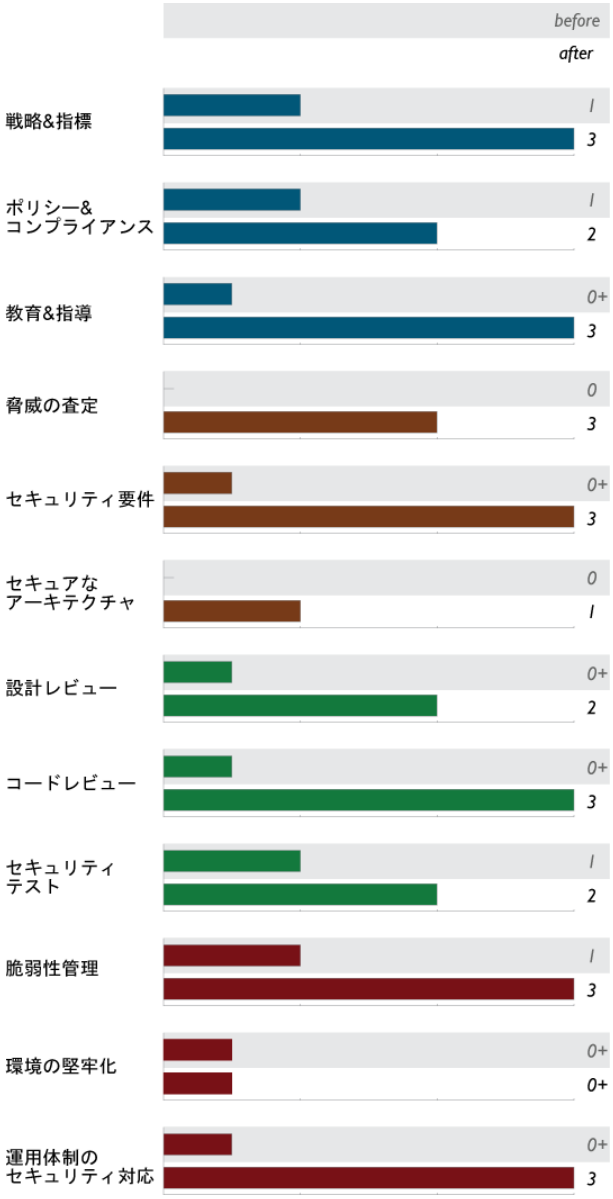
- ビジネスオーナー及びチームリーダは、自社のアプリケーションに関連するリスクを認識すること、そしてリリース前にアプリケーションのリリースを承認することが求められる。
- チームリーダはすべてのアプリケーションが正式にセキュリティプロセスを経ることを要求し、開発者はコードレビューを毎週実施する。
- 毎年継続的なトレーニングプログラムと教育プログラム（CBT を含む）がすべてのプロジェクト要員に提供され、開発者は少なくとも年に 1 回コースへの参加を求められる。
- アプリケーションセキュリティに関する専属のチームリーダを配置し、顧客とのコミュニケーション、顧客向けの技術文書やガイドラインに関する責任を負う。

将来を見据える VirtualWare 社は SDL の一部であるセキュリティの文化を浸透させ、顧客向けに開発し、提供されたアプリケーションの安全性と堅牢性を確保する。報告を受け、必要に応じて同社が対処する可能性のある脆弱性に対し、効果的なプロセスが導入された。

導入の最終フェーズの間、導入時に表面化した弱点を明らかにするために、プロジェクトのギャップ審査が実施された。特に要員の離職率が高いため、VirtualWare 社は新しい開発者が入社するたびにトレーニングを継続的に実施する必要があった。この問題を解決するために設定された主要目的は、ある入門プログラムを特に開発者向けに導入して、開発者たちが入社当初から正式なセキュリティトレーニングを受けられるようにすることだった。これは、セキュリティが重要であるという考え方を社内とその開発チーム内で生み出すことにも役立った。

成熟度スコアカード

この成熟度スコアカードは、VirtualWare 社によるソフトウエア保証プログラムの導入中に自己評価として記入されたものである。最終的なスコアカード（右側参照）は、4 つのフェーズからなる改善プロジェクトの開始時と終了時における VirtualWare 社の状態を表している。



最新版及びその他の情報については、プロジェクトのウェブサイト
(http://www.opensamm.org) を参照のこと。

Author & Project Lead

Pravir Chandra

Contributors/Reviewers

Fabio Arciniegas
Matt Bartoldus
Sebastien Deleersnyder
Jonathan Carter
Darren Challey
Brian Chess
Dinis Cruz
Justin Derry
Bart De Win
James McGovern
Matteo Meucci
Jeff Payne
Gunnar Peterson
Jeff Piper
Andy Steingruebl
John Steven
Chad Thunberg
Colin Watson
Jeff Williams

Sponsors

Thanks to the following organizations that have made significant contributions to the SAMM Project.

Supporters

Thanks to the following organizations for helping review and support the SAMM Project.

Note: OWASP and the SAMM Project do not endorse any commercial products or services