セキュリティ・バイ・デザイン

導入指南書

２０２２年８月

産業サイバーセキュリティセンター　中核人材人材育成プログラム

5 期生 「システム開発のセキュリティ向上」プロジェクト

目次

はじめに *3*

免責事項 *3*

1. セキュリティ・バイ・デザイン *4*

1.1 開発現場を取り巻く環境 4

1.2 セキュリティ・バイ・デザインとは 4

1.3 OWASP SAMM とは 5

1.4 セキュリティ・バイ・デザインは何をやれば良いのか 6

2. 脅威分析 *8*

2.1 脅威分析のフレームワーク 8

2.2 ビジネススコープと事業被害 9

2.3 脅威の特定 12

2.4 ステークホルダー 14

2.5 戦略的シナリオによるステークホルダーリスクの削減 16

2.6 攻撃手順シナリオ 19

2.7 リスクへの対応 21

3. セキュリティ要件 *25*

3.1 セキュリティ要件の定義方法 25

3.2 ガイドラインに準拠する 29

3.3 対応するセキュリティ要件を決定する 30

3.4 セキュリティ要件の決定はスタートラインに過ぎない 31

4. セキュリティアーキテクチャ *33*

4.1 セキュリティ原則の適用 33

4.2 ツールや技術の特定 33

5. 下流工程のセキュア開発 *36*

5.1 シフトレフトとは 36

5.2 シフトレフトの実践 37

6. 参考文献 *41*

6.1 主なセキュリティガイドライン 41

6.2 本書執筆における参考文献 45

おわりに *46*

巻末コメント *47*

謝辞.....*53*

はじめに

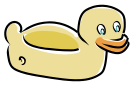
セキュリティ・バイ・デザインという言葉を聞いたことがあるだろうか。セキュリティ・バイ・デザインとは、製品の企画や設計のフェーズからセキュリティ対策を組み込むことで、サイバーセキュリティを確保しておく考え方である。システム開発やソフトウェア開発においてセキュリティ・バイ・デザインを実践する組織が増えてきているが、言葉は理解できるものの何をすれば良いのか、何から始めれば良いのかがわからない人も多いだろう。

このドキュメントはセキュリティ・バイ・デザイン実践の入門書である。セキュリティ・バイ・デザインを実践するためには、開発プロセスや開発手法、ネットワークやシステムアーキテクチャ設計などの開発技術、脅威分析やセキュリティ対策手法など幅広い知識を必要とするが、本文書を通してセキュリティ・バイ・デザインそのものに対する理解を深め、セキュアな開発を実践するための第一歩を踏み出してほしい。

尚、セキュリティ・バイ・デザインを実施する上で、技術以外にも組織的対応、人的な対応が必要となるが、今回それらはスコープ外とする。

本文書はシステム開発のプロジェクト管理者および開発担当者を対象読者として想定している。

本文書の登場人物

 ：読者の気持ちを代弁するアヒル

免責事項

- このドキュメントは単に情報として提供され，内容は予告なしに変更される場合がある。
- このドキュメントに誤りが無いことの保証や，商品性又は特定目的への適合性の黙示的な保証や条件を含め明示的又は黙示的な保証や条件は一切無いものとする。
- 本書に記載の内容は，独立行政法人情報処理推進機構および産業サイバーセキュリティセンターの意見を代表するものではなく，作成者の見解に基づいている。
- 本書の利用によるトラブルに対し，本書作成者ならびに監修者は一切の責任を負わないものとする。
- 本書の有効期限は，発行日から２年間とする。

1. セキュリティ・バイ・デザイン

1.1　開発現場を取り巻く環境

近年、システム開発の現場はビジネス環境の変化が激しく、開発を開始してから顧客に納品するまでスピードが要求されるようになった。ビジネスニーズにいち早く対応し、素早くシステムを提供するために、DevOps（開発担当と運用担当が連携・協力し、フレキシブルかつスピーディーに開発する開発手法）やアジャイル開発といった手法を取り入れながら、ビジネス環境の変化に対応する企業が増えてきている。

一方、ソフトウェア製品やウェブサイトの脆弱性は毎年多数報告されている。脆弱性は様々なサイバーインシデントに繋がる可能性があり、ソフトウェアの脆弱性を減らすためにも、セキュア開発の重要性は益々高まっている。企業やシステム利用者を様々なサイバー攻撃から守るためにも、開発に関わる全ての人がセキュア開発に取り組まなければならない。

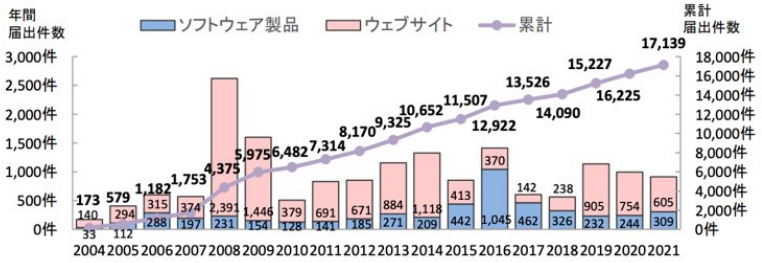
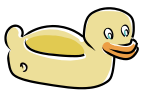


図 1 脆弱性の年間届出件数の推移[1]



「取り組まなければならない」と言うが、
何をすればいいのか分からんのや

～重要ポイント～

✓　**利用者をサイバー攻撃から守るためにセキュア開発に取り組む必要がある**

1.2　セキュリティ・バイ・デザインとは

セキュリティ・バイ・デザインとは「情報セキュリティを企画、設計段階から組み込むた

[1] ソフトウェア等の脆弱性関連情報に関する届出状況 2021 年第 4 四半期（10 月～12 月）参照(https://www.ipa.go.jp/security/vuln/report/vuln2021q4.html)

めの方策」で内閣サイバーセキュリティセンター(NISC)により定義されている。開発プロセスの早い段階からセキュリティを考慮することで、開発するシステムのセキュリティを確保するという考え方である。

従来のような後付けでセキュリティ機能を追加したり、出荷直前になってセキュリティツールを実行していては、手戻りが多発したり、結果的に開発コストが多くかかってしまう可能性がある。開発の早い段階でセキュリティ対策を行うことで、手戻りが少なく、コスト削減に繋がり、保守性の良いシステム・ソフトウェアの作成につながる。

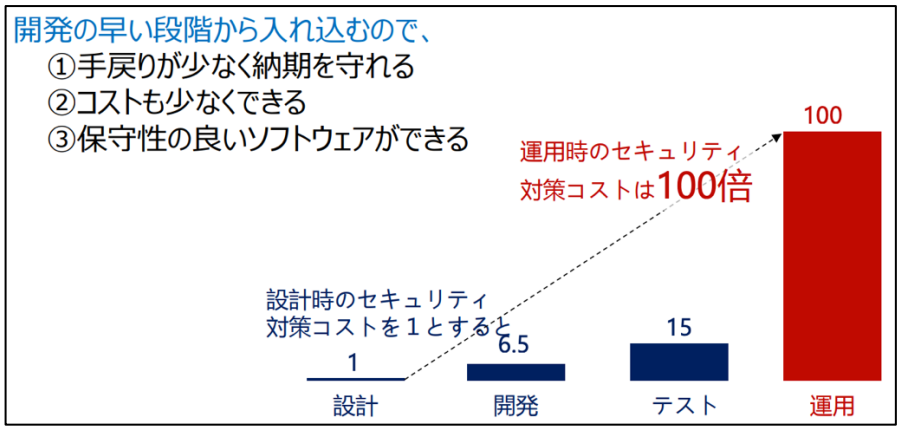
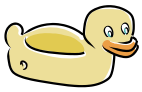


図 2 セキュリティ対策の実施タイミングと対策コスト[2]



なるほど、完全に理解した（棒）

～重要ポイント～

- ✓ **セキュリティ・バイ・デザインとは、脆弱性に対して上流工程で先手を打つこと**
- ✓ **セキュリティ・バイ・デザインにより、ソフトウェアの「信頼性の向上」「コスト削減」「保守性の良いシステムが開発できる」が得られる**

1.3 OWASP SAMM とは

セキュリティ・バイ・デザインをやろうと思い立ったとしても、まず直面するのが「なにから始めればいいのだろう」という疑問だ。

[2] セキュリティ・バイ・デザイン入門(https://www.ipa.go.jp/files/000055823.pdf)

本文書では OWASP SAMM 2.0 をベースに、セキュリティ・バイ・デザインで実施すべき事項を具体例も交えて紹介する。SAMM はソフトウエア保証成熟度モデル(Software Assurance Maturity Model)の頭文字で、ソフトウェ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である。Web アプリケーション開発においてやるべきことを、要件定義からテスト、保守工程まで様々な観点で記載している。SAMM は Web アプリケーションのフレームワークだがセキュア開発の全体像を掴むためには有益なドキュメントである。

SAMM は以下の 5 つのビジネス機能に対して成熟度を定義している。

・ガバナンス(Governance)
・設計(Design)
・実装(Implementation)
・検証(Verification)
・運用(Operations)

本文書では、開発現場におけるセキュリティ・バイ・デザインの実施に該当する「設計」をベースに上流工程で実施すべき事項を記載している。

～重要ポイント～

✓ **OWASP SAMM にはセキュリティ対策でやるべきことが網羅されている**
✓ **本文書でセキュリティ・バイ・デザインの中心となる「設計」を紹介する**

1.4　セキュリティ・バイ・デザインは何をやれば良いのか

OWASP SAMM の「設計」では、以下の 3 つのセキュリティ対策を実施することを推奨している。

・脅威分析(Threat Assessment)
・セキュリティ要件(Security Requirements)
・セキュリティアーキテクチャ(Security Architecture)

「脅威分析」ではソフトウェアが直面する脅威や想定される攻撃を明らかにする。ソフトウェアを「何から守るのか」明らかにするのだ。

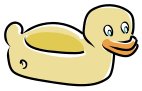
「セキュリティ要件」ではソフトウェア自身のセキュアな動作を定義する。要件の種類にはシステムの機能に関する要件や、可用性、保守性、性能などの要件がある。セキュリティ要件とは、システム要件のうちセキュリティに関する要件のことであり、システムを安全に運用するために必要な目標を定義する。システム要件定義書の一部として、またはセキュリティ要件定義書として、セキュリティ要件を記述する。

「セキュリティアーキテクチャ」は少し難しい。どのソフトウェアにも通用する安全な

アーキテクチャは存在しないからだ。SAMM においては、フレームワークやデザインパターンを利用することを推奨している。手作業で作った新規のソースコードよりも、広く普及している信頼されたプラットフォームやライブラリの方が信頼できるという考え方だ。自組織で独自のアーキテクチャを考えるよりも、プラットフォームの提供元が推奨しているアーキテクチャをカスタマイズしつつ利用する。

～重要ポイント～

✓ **設計では「脅威分析」「セキュリティ要件」「セキュリティアーキテクチャ」の３つの取り組みを実施する**



上司からセキュリティ・バイ・デザインをやれと言われて
困っていたが、この３つをやればいいのか！！

2. 脅威分析

本章では、OWASP SAMM の設計で定義されている脅威分析の実践方法を紹介する。

SAMM における脅威分析では、2 つのストリームに 3 段階の成熟度レベルが定義されている。

ストリーム A：アプリケーションリスクプロファイル(Application Risk Profile)

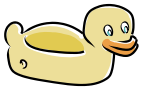
ストリーム B：脅威モデリング(Threat Modeling)

これらを参照することで、脅威分析において実施すべき事項が分かる。しかし、具体的な実施手順やコツが明記されておらず、SAMM を参照するだけでは脅威分析の実施は難しい。

本章では、脅威分析を実施するための具体的な手順の一例を紹介する。

～重要ポイント～

- ✓ **SAMM の脅威分析は 2 つのストリームで定義されている**
- ✓ **本章では、脅威分析の具体的な実施手順を紹介する**



SAMM は「やるべきこと」なので、「やり方」の説明が少ない。
ここで教えてくれるのはありがたい。

2.1　脅威分析のフレームワーク

前述の通り、SAMM には実施手順が具体的に明記されていない。このため、脅威分析を実施する際の指標として、別のフレームワークを活用する。

脅威分析の実施において、公開されているフレームワークをいくつか取り上げる。

・OWASP Threat Modeling / Threat Modeling Process
・STRIDE
・Trike
・攻撃ツリー
・EBIOS
・制御システムのセキュリティリスク分析ガイド（IPA）
　他

本章では、これらの中から EBIOS を利用した脅威分析の手法を紹介する。外部の脅威の列挙から具体的攻撃手法の想定まで実施することができ、これを実施するだけで SAMM の推奨事項の大半を満たすことができるためだ。

EBIOS はフランスの国家情報システムセキュリティ庁（ANSSI）が提唱するリスク分析手法で 5 つの Workshop で構成されている。

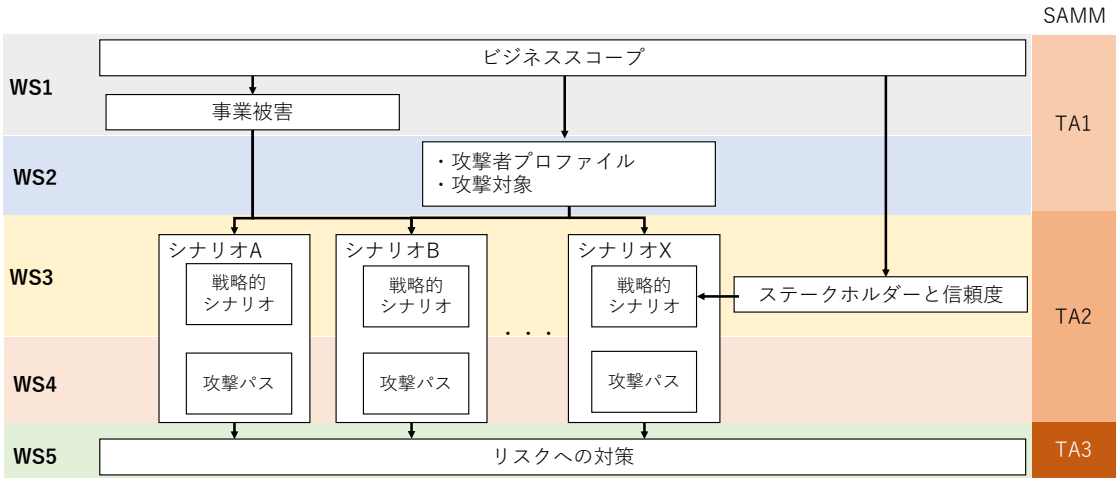
Workshop1：スコープとセキュリティベースライン（Scope and security baseline）

Workshop2：脅威の特定（Risk Origins）

Workshop3：戦略的シナリオ（Strategic Scenarios）

Workshop4：攻撃手順シナリオ（Operating Scenarios）

Workshop5：リスクへの対策（Risk Treatment）



※WS: Workshop

図 3 EBIOS Workshop の流れ

5 つの Workshop を実施することで、SAMM の設計の実施内容をほぼカバーすることができる。このため、本章では EBIOS の Workshop 実施の流れに沿って脅威分析の実践手順を紹介する。

～重要ポイント～

- ✓ **EBIOS は脅威分析のためのフレームワーク**
- ✓ **EBIOS を実施することで、SAMM の脅威分析をカバーできる**

2.2 ビジネススコープと事業被害

本節は EBIOS の Workshop1「スコープとセキュリティベースライン(Scope and security

baseline)」の実践方法を紹介する。

図 3 に示すように、Workshop1 ではビジネススコープと事業被害を定義する。ビジネススコープとは、ソフトウェアが使用されるビジネス上の目的や周辺環境のことを示す。以下に例示するような項目を定義していくと、脅威分析を行う範囲が明らかになっていく。

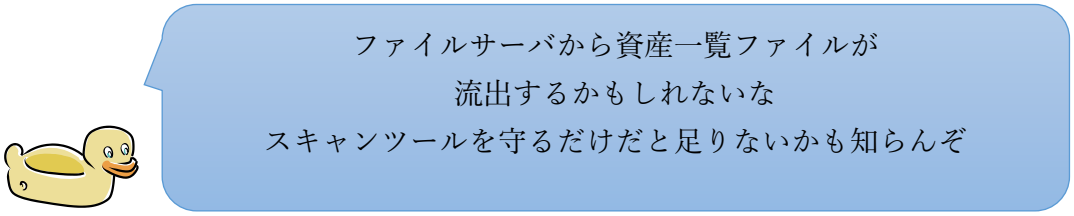
・目的　　：ビジネスにおいて達成すべき目的
・手順 or 情報：業務上の手順か情報資産かの分類
・使用資産　　：業務手順で使用するツールや情報を保管する機器など
・担当　　：手順を実施したり情報を管理する担当者

本文書ではこれ以降、ICS[3]資産を把握するためのネットワークスキャンツールの開発を題材として、セキュリティ・バイ・デザイン実施の具体例を提示する。

表 1 ビジネススコープの定義例

目的	ネットワークに接続されている機器を調査し、資産リストを作成する。	
ビジネス分類	資産管理	資産一覧ファイル
手順 or 情報	手順	情報
詳細	・ネットワーク上でスキャンツールを実行 ・資産一覧ファイルを出力	・IPアドレス ・OS ・使用プロトコル ・ネットワーク図
担当	ICSシステム担当	ICSシステム担当
使用資産	スキャンツール	ファイルサーバ
資産詳細	・スキャンツールソフトウェア ・ツール用小型PC	・Windowsファイル共有サーバ
部署	ICS運用部	ICS運用部

表 1 は ICS 資産を把握するためのネットワークスキャンツールを題材としたビジネススコープの具体例である。このソフトウェアが何を目的にして利用されるのか、ツールを使用する業務手順、情報の保管場所などが分かる。



[3] ICS(Industrial Control System)：産業用制御システム。工場やビルなどの産業インフラを制御するシステム

ビジネススコープ定義の次には、想定される事業被害を定義する。以下のカテゴリの被害がよく挙げられる。

・情報流出
・事業停止
・資産の破損
・人命の危険
・信用失墜
・金銭被害

定義したビジネススコープ内で自組織にとって避けたい事業被害を定義し、深刻度を設定する。深刻度は表 2 に示す 4 段階で分類する。

表 2 事業被害の深刻度

深刻度	概要
Level 4　致命的	・安全や資産に致命的な影響が生じる ・発生した被害を克服できず、企業存続が脅かされる
Level 3　深刻	・安全や資産に深刻な影響が生じる ・その状況を脱するのが困難で、企業活動に大きなダメージが生じる
Level 2　重要	・安全や資産への影響は生じない ・ある程度の被害が生じるが、復旧可能
Level 1　影響あり	・安全や資産への影響は生じない ・復旧作業が必要であるが、直接的な被害は発生しない

今回のサンプルでは表 3 のとおり事業被害を想定し、事業に与える深刻度の評価を行った。

表 3 事業被害の定義例

事業被害	詳細	深刻度
ネットワーク上の機器の停止	顧客のインフラのサーバ類の停止	Level 2
ネットワーク上の機器の乗っ取り	ネットワーク上の機器が遠隔操作され、サイバー攻撃の起点となる	Level 3
資産一覧ファイルの流出	顧客のネットワークや資産情報が外部に流出する	Level 3

これに加え、ガイドラインやスタンダードを活用したセキュリティベースラインの定義も EBIOS では推奨されているが、ガイドラインやスタンダードの活用については 3.2 節で述べる。

～重要ポイント～

- ✓ **脅威分析を実施するビジネススコープを定義する。**
- ✓ **ビジネススコープ内で起こってほしくない事業被害を定義する。**

2.3　脅威の特定

本節では EBIOS の Workshop2「脅威の特定（Risk Origins)」の実践方法を紹介する。

セキュリティの分野では、リスクを発生させる要因を脅威と定義する。脅威には大別すると外部の脅威と内部の脅威が存在する。外部の脅威としてはサイバー犯罪組織によるサイバー攻撃などが挙げられる。内部の脅威は自組織の社員が金銭や転職目的で情報を持ち出したり、怨恨でシステムを停止させるようなパターンが挙げられる。

これらの脅威となるような攻撃者の代表的なカテゴリを挙げる。

・ハクティビスト【外部】
・サイバー犯罪組織【外部】
・国家支援型組織【外部】
・産業スパイ【外部】
・個人犯罪者【外部】
・内部犯【内部】
・サプライチェーン事業者（仕入れ先、下請け業者など）【内部、外部】

これらのカテゴリから、自組織に対しての脅威となる攻撃者を選定する。選定するのが難しい場合、全て列挙しておいてもよい。

列挙した攻撃者からの攻撃可能性を評価するために、以下の 3 つの要素を付け加える。

・攻撃者が狙う攻撃目標　：攻撃者が狙う資産や情報。攻撃者毎に複数存在しうる
・攻撃者のモチベーション：活動の活発さを 4 段階で評価
・攻撃者のリソース　　：使える資金や人数、時間、技術力を 4 段階で評価

攻撃目標は 2.2 で定義したビジネススコープを参考にするとよい。資産詳細などは攻撃者の攻撃目標になりやすい。モチベーションとリソースについては、以下の表 4 および表 5 を参考に設定する。

表 4 攻撃者のモチベーション基準

レベル	モチベーション
Level 4	最大の目的が自組織への攻撃で、日々活動している
Level 3	攻撃行動を主要な活動としており、意欲が高い
Level 2	攻撃を意識しているが、積極的に活動していない
Level 1	攻撃行動をとる可能性はあるが、活動の兆候がない

表 5 攻撃者のリソース基準

レベル	リソース
Level 4	攻撃に必要な技術力があり、資金や人員が充足している
Level 3	攻撃に必要な技術力があるが、資金や人員が限定的
Level 2	高度な技術を持たないが、資金や人員が豊富
Level 1	高度な技術を持たず、資金や人員が限定的

ここまでで必要な要素が出揃ったため、各攻撃目標への攻撃可能性を評価する。以下の表 6 を参考に、攻撃目標毎のモチベーションとリソースから攻撃可能性を評価する。

表 6 攻撃の発生可能性

		リソース			
		Level 1	Level 2	Level 3	Level 4
モチベーション	Level 4	攻撃可能性がある	比較的攻撃可能性が高い	攻撃可能性が高い	攻撃可能性が高い
	Level 3	攻撃可能性がある	比較的攻撃可能性が高い	比較的攻撃可能性が高い	攻撃可能性が高い
	Level 2	攻撃可能性が低い	攻撃可能性がある	比較的攻撃可能性が高い	比較的攻撃可能性が高い
	Level 1	攻撃可能性が低い	攻撃可能性が低い	攻撃可能性がある	攻撃可能性がある

ネットワークスキャンツールの具体例を以下の表 7 に提示する。

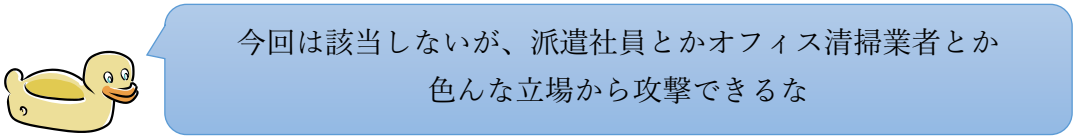
表 7 ネットワークスキャンツールの攻撃発生可能性

攻撃者	攻撃対象	モチベーション	リソース	発生可能性
サイバー犯罪組織	資産一覧ファイル	Level 3	Level 3	比較的攻撃可能性が高い
	NW上の機器停止	Level 1	Level 3	攻撃可能性がある
サプライチェーン事業者(開発委託先社員)	ツールにマルウェアを混入	Level 2	Level 3	比較的攻撃可能性が高い
サプライチェーン事業者(OSS開発者)	ツールにマルウェアを混入	Level 1	Level 3	攻撃可能性がある
内部犯(ICS作業員)	NW上の機器停止	Level 1	Level 1	攻撃可能性が低い

モチベーションの Level 判断には、自組織が攻撃のターゲットかどうかが一つの指標となる。例えば、開発委託先社員は自組織の案件を受託しているため、自組織との関連が強い。

これに対し、OSS 開発者は不特定多数の組織に対してソフトウェアを提供しているため、特に自組織を狙っての攻撃活動を行う可能性が低い。

リソースの Level 判断では、対象の技術力に加え、対象システムへ精通しているか否かが判断材料になる。開発委託先社員は開発中のシステムを詳細まで理解しており、攻撃リソース Level を高く判断できる。



～重要ポイント～

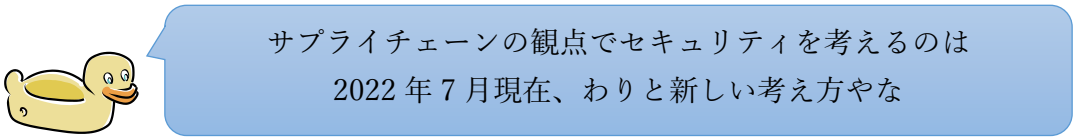
- ✓ **自組織の脅威となる攻撃者を設定する**
- ✓ **攻撃対象、モチベーション、リソースから攻撃の発生可能性を導き出す**

2.4 ステークホルダー

本節では EBIOS の Workshop3「戦略的シナリオ（Strategic Scenarios)」前半の実践方法を紹介する。

サイバー攻撃に対する自組織の防御が十全であっても、攻撃を受けてしまうケースが存在する。機器ベンダーや顧客といった、ビジネスのサプライチェーン上におけるステークホルダーが攻撃されることにより、自組織が間接的な攻撃を受けてしまう場合がある。

ステークホルダーのリスク分析にあたって、まずは自組織のステークホルダーを列挙する。サプライチェーンの上流にあたるサービスプロバイダや下流にあたる顧客、派遣社員などのパートナーや下請け業者など、関わりのある立場を列挙する。



こうして列挙したステークホルダーに脅威度のスコア付けを実施する。表 8 のスコア基準と数式 1 を使用して算出する。

$$脅威度 = \frac{危険性}{安全性} = \frac{依存性 \times アクセス権限}{セキュリティ成熟度 \times 信頼性}$$

数式 1 ステークホルダーの脅威度

表 8 ステークホルダー脅威度のスコア基準

スコア	依存度	スコア	アクセス権限	スコア	セキュリティ成熟度	スコア	信頼性
1	必要がない関係性	1	アクセス権限がないもしくはユーザー権限でアクセスできる	1	ルール化されておらず、個人単位で対応している。	1	調査されていないステークホルダー
2	ビジネスにとって有用	2	管理者権限でユーザー端末にアクセスできる。もしくは物理オフィスに入場できる。	2	ITルールを現場毎に設定している。	2	中立的なステークホルダー
3	重要だが代替可能	3	ビジネス用サーバーに管理者権限でアクセスできる。	3	チーム単位で全体ポリシーを適用しており、リスクに先回りして反応的に動ける	3	善意を持って協業していると判明している
4	重要で代替が効かない	4	インフラ環境に管理者権限でアクセスできる。もしくはサーバールームに入場可能。	4	リスクマネジメントポリシーを適用しており、随時更新している。	4	ステークホルダーと自組織の考えが完全に一致していると判明している

ネットワークスキャンツールの具体例では、以下の表 9 のような数値となる。

表 9 ネットワークスキャンツールのステークホルダー

ステークホルダー	依存度	アクセス権限	セキュリティ成熟度	信頼性	脅威度
OSS開発者	3	1	4	2	0.375
情報システム部	4	1	3	4	0.333
ICS保守ベンダ	3	4	3	4	1.000
派遣社員	2	3	2	2	1.500
オフィス清掃業者	1	2	1	2	1.000
ツール開発委託企業	3	2	2	3	1.000

ステークホルダーに脅威度を設定した後は、脅威度に応じて脅威レベルを設定する。脅威レベルは 3 段階存在する。ステークホルダーに設定した脅威度の最大値を 100%として、0-50%,50-90%,90%-100%で区切ることで、脅威レベルを設定する。

表 10 ステークホルダーの脅威レベル

脅威レベル	リスク受容	推奨対応	対象脅威度（最大値からの%）
危険	受容不可	・対象ステークホルダーにセキュリティ対策を施す ・ビジネスから除外する	90% < x <=100%
注意	受容度合いをコントロールする必要あり	・対象を特別に監視する ・セキュリティ認証を取得させる ・リスクを低減させる対策を実施	50% < x <=90%
観察	受容可能	・特になし	0% < x <=50%

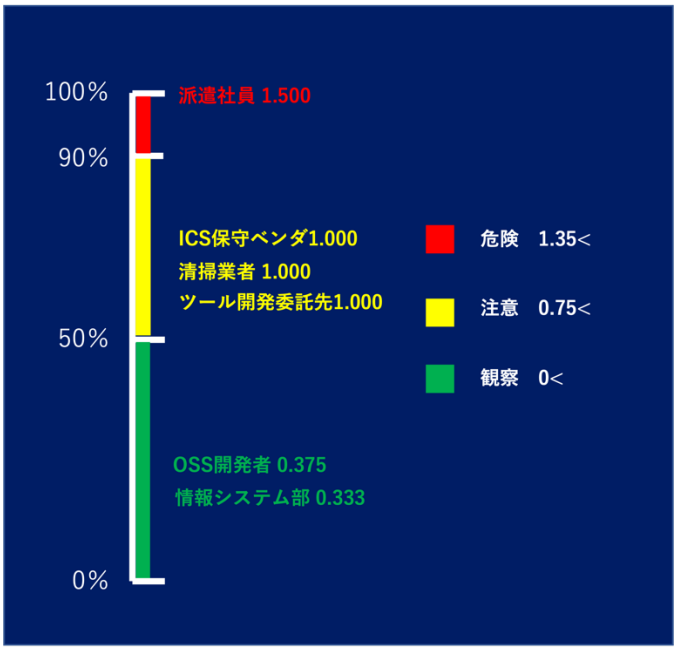
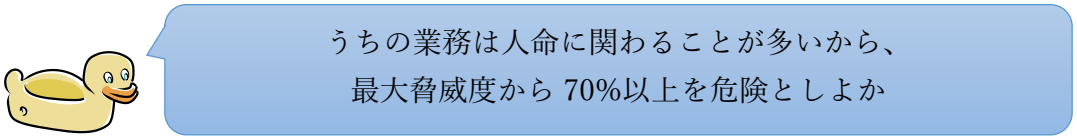


図 4 ネットワークスキャンツールのステークホルダーマッピング

ここまでステークホルダーの脅威レベルを示す基準を示してきたが、これらは標準的な基準である。実際に脅威分析を実施する際には、各組織の事情に合わせてカスタマイズすることを推奨する。



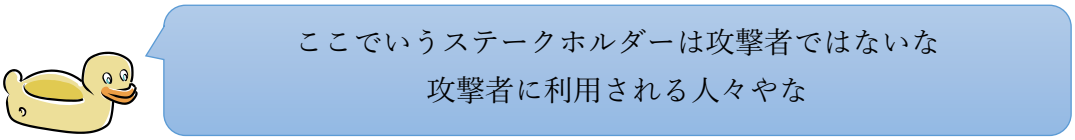
～重要ポイント～

- ✓ **ステークホルダーにもリスクが存在する**
- ✓ **ステークホルダーの危険性と安全性からリスク受容の可否を決定する**

2.5 戦略的シナリオによるステークホルダーリスクの削減

本節では EBIOS の Workshop3「戦略的シナリオ（Strategic Scenarios)」後半の実践方法を紹介する。

前節ではステークホルダーとその脅威度を定義した。本節においては、ステークホルダーを媒介する攻撃シナリオを作成し、攻撃を成立させないためのセキュリティ対策(Security Measure)を提案する。



攻撃シナリオの作成手順

1. 事業被害(2.2 節)と攻撃目標(2.3 節)の組み合わせを列挙する
2. 組み合わせのうち、事業被害と攻撃目標が噛み合っているものを抽出する
3. 事業被害と攻撃者の組み合わせ毎に、攻撃パスのフローチャートを作成する
   - 3.1. 社外、ステークホルダー、社内の 3 つにエリアを区切る
   - 3.2. 攻撃のスタート地点に攻撃者、ゴール地点に攻撃対象を記述
   - 3.3. 攻撃対象の設置箇所や保管場所を明記
   - 3.4. 攻撃対象に影響を与えたり、アクセスできるステークホルダーを追記
   - 3.5. 攻撃者から攻撃対象まで、アクセスできるルートを線で結び、説明を追記

以下にネットワークスキャンツールの攻撃シナリオのサンプルを提示する。

■シナリオ A　サイバー犯罪組織による資産一覧ファイルの窃取

① ネットワーク機器がハッキングされ、ファイルサーバにアクセスされる
② 攻撃者から企業内部の作業者に向けて、機密情報の買取募集が発行される
③ 貸与されたアカウントでファイルサーバからファイルを窃取する。または資産ツールから直接資産一覧ファイルを入手する

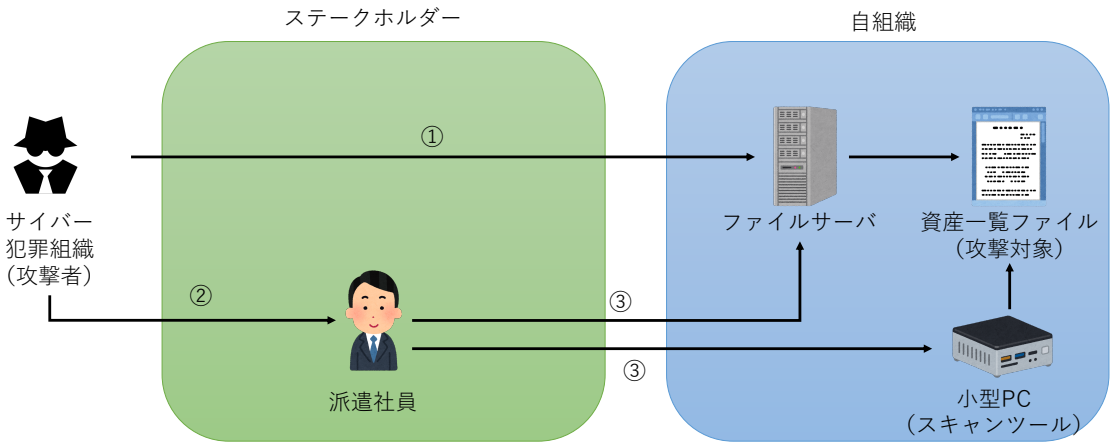


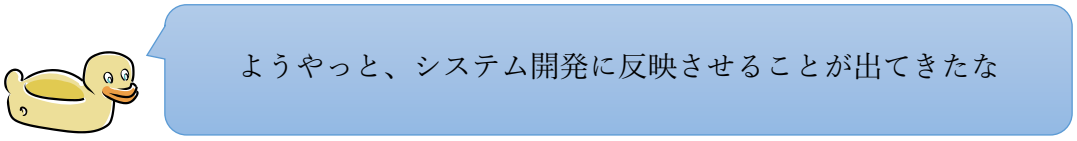
図 5 サイバー犯罪者から資産一覧ファイルへの攻撃シナリオ

シナリオが出来上がると、各ステークホルダーが脅威となっている要因を読み取ることができる。脅威レベルが高いステークホルダーに対して対策を講じることで脅威レベルを「観察」にまで低減させる。

具体例の中では、アカウントの認可権限を最小化することで、派遣社員のアクセス権限を 3 から 2 へ改善。セキュリティ教育を受講させることでセキュリティ成熟度を 2 から 3 へ改善する。これらのセキュリティ対策により、脅威度が 1.500 から 0.666 に低下し、派遣社員のステークホルダーリスクが受容可能なレベルにまで低減される。

表 11 ステークホルダーのセキュリティ対策と脅威度の低減

ステークホルダー	攻撃ルート	セキュリティ対策	脅威度（対策前）	脅威度（対策後）
派遣社員	・貸与アカウントでファイルサーバから資産一覧ファイルを窃取 ・持ち込んだUSBを端末に接続してマルウェアをインストール	・派遣社員に与えるアカウントの認可権限を最小限に設定する ・エンドポイントセキュリティ製品を導入し、不許可USBの接続を防止 ・自組織のセキュリティ教育を受講させ、組織のセキュリティ基準を遵守させる ・スキャンツールに認証機能を実装し、派遣社員から利用できないようにする。 ・スキャンツールのハードウェアを施錠できるようにする	1.500	0.666



この段階までたどり着くと、システム開発においてやるべきことが見えてきた。シナリオ A では、スキャンツールをインストールした小型 PC の設置箇所に侵入されることがリスクであると判明した。このため、スキャンツールには侵入されても不正操作されないようにする機能が必要であることが分かる。今回の具体例では、パスワードなどの認証機能の実装、及び施錠によるハードウェアの保護をセキュリティ対策とした。

これらのセキュリティ対策はシステム開発におけるセキュリティ機能として開発工程に組み込む必要がある。

～重要ポイント～

- ✓ **事業被害と攻撃目標の組み合わせでシナリオを作成する**
- ✓ **ステークホルダーのリスクを明確化し、低減する対策を提案する**
- ✓ **システム開発に組み込むセキュリティ機能を明らかにする**

2.6　攻撃手順シナリオ

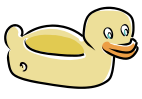
本節では EBIOS の Workshop4「攻撃手順シナリオ（Operating Scenarios)」の実践方法を紹介する。

2.5 節では攻撃者とステークホルダーの観点から攻撃シナリオを作成したが、本節ではシナリオをさらに詳細化する。攻撃の手順を 4 段階に分けて記述し、それに対して実現可能性と難易度の 2 観点で脅威度のスコア付けを行う。こうすることで、攻撃行動のうち、どこを防げば効率的なリスク対策となるかを明らかにすることができる。

攻撃手順の 4 段階
調査：ポートスキャン、OSINT、内部協力者の確保など
侵入：フィッシングメール、不正 USB デバイスの設置、脆弱性攻撃など
探索：ネットワークスキャン、ラテラルムーブメント、C2 通信など
実行：マルウェアの実行、機密情報の外部送信、システムの停止など

シナリオの作成手順

1. 攻撃者が目的を達成するまでの手順をフローチャートとして図に書き起こす。
   このとき、順番に実行する手順は直列、どれか一つでも成功すればよい手順を並列に記述する。
2. 各手順に実現性と難易度のスコアをつける。（表 11 参照）
3. 各手順に対し、そこに辿り着くまでの手順を含めた実現性と難易度のスコアをつける。
4. 攻撃パス毎の実現性と難易度から、攻撃の発生可能性を導出する。



「ファイルサーバにログイン」は 3 つの攻撃パスが通過するな。
なんとなく重要な気がする。

サイバー犯罪者が、機密情報である資産情報の一覧を窃取するフローを図 6 に記載する。

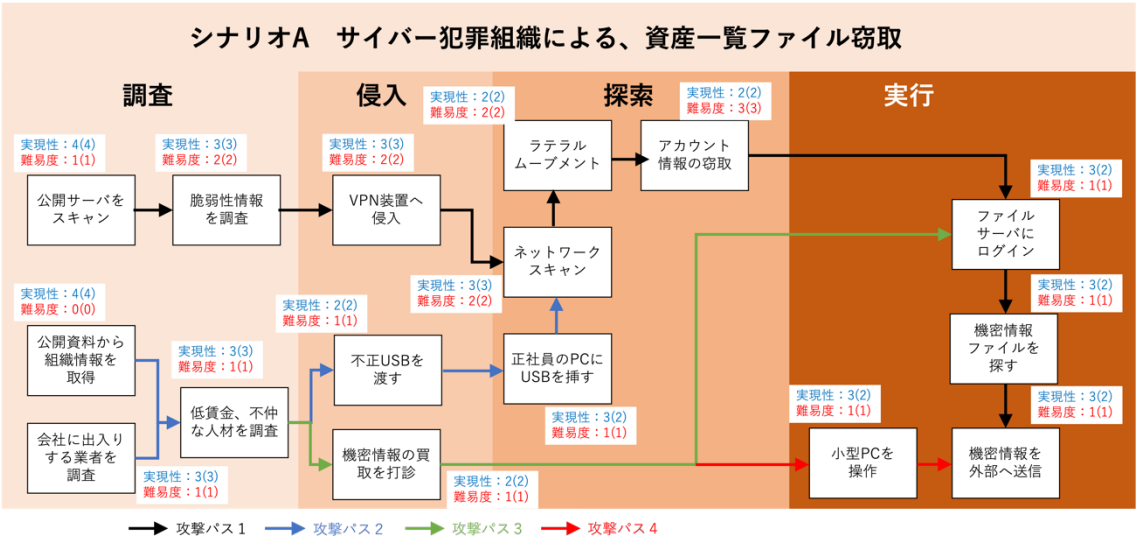


図 6 攻撃手順シナリオサンプル

表 12 実現性と難易度基準

	実現可能性	難易度
Level4	90%<	ゼロデイ攻撃など、高度な攻撃技術が必要
Level3	60%<	攻撃技術への一定の理解が必要
Level2	20%<	攻撃ツールや既知の脆弱性を活用することで攻撃可能
Level1	<=20%	特定のシステム操作や設定変更など、攻撃技術の深い理解がなくとも攻撃可能
Level0	<3%	侵入経路やパスワードが明瞭で、誰でも攻撃可能

図 6 の通り、攻撃手順シナリオには 4 つのパスが存在し、それぞれの手順に実現性と難易度のスコア付けを実施している。括弧内の数値はその手順に至るまでの手順を含めたトータルスコアである。注意しなければならないのは、実現性は数値が小さいほど、難易度は数値が大きいほど攻撃可能性が低い。実現性のトータルスコアをつける際は、通過した手順のうち最も小さい数値をつける。難易度のトータルスコアの場合は通過した手順のうち最も大きい数値をつける。

上記の例では、機密情報が送信される実現性が 2、難易度が 1 となる。これを表 5 の評価基準を基に評価（難易度の数値は反転させる）すると、「比較的攻撃可能性が高い」となり、2 番目に発生可能性が高い評価となる。

事業被害と攻撃者を組み合わせた各シナリオで同様のシナリオを作成し、攻撃の発生可能性を導出する。

～重要ポイント～

- ✓ **攻撃手順シナリオ：攻撃者の目的達成までの手順のフローチャート**
- ✓ **各手順を実現性と難易度で評価することで、シナリオ全体の発生可能性が分かる**

2.7　リスクへの対応

本節では EBIOS の Workshop5「リスクへの対策（Risk Treatment)」の実践方法を紹介する。

ここまでの作業により、自組織及びシステムに想定される脅威や事業被害、発生可能性を導き出すことができた。本節ではこれらのリスクに対する対応方法を決定する。

まず行う手順は、各攻撃シナリオのリスクの大きさの評価だ。攻撃シナリオのリスクは以下の計算式で表すことができる。

$$\textit{リスク} = \textit{事業被害の深刻度} \times \textit{シナリオの発生可能性}$$

数式 2 リスク算出式

事業被害の深刻度は 2.2 節で定義した値を使用する。シナリオの発生可能性は 2.6 節で導出した発生可能性を使用する。ただし、これらの値は数段階に評価されているだけで、単位のある数値ではない。このため、単純な乗算でリスクの大きさを数値化することはできない。このため、深刻度と発生可能性でマトリックスを作成し、自組織の評価基準を作成する。作成した評価基準の一例を以下に示す。

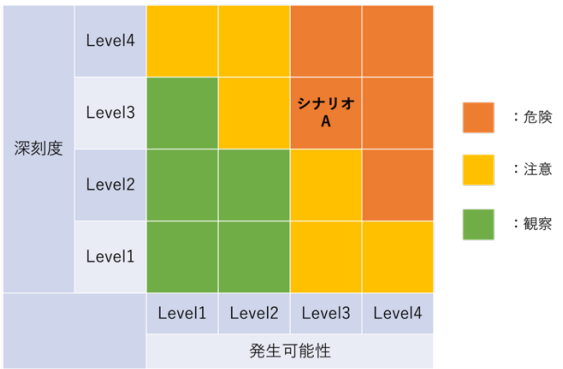


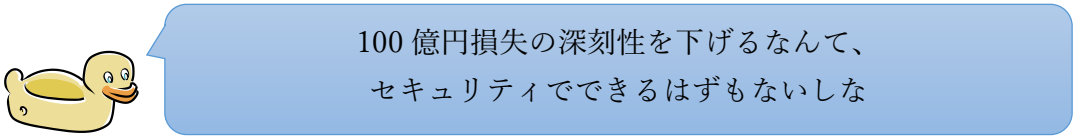
図 7 深刻度と発生可能性に対するリスクの大きさ

この例では発生可能性の比重をやや高くしているが、Level4 の深刻度を重くみる組織では深刻度 Level4、発生可能性 Level2 のリスクを危険と評価しても良い。

ここで定義したリスクの大きさはリスクの受容度に繋がる。表 9 の基準に照らし合わせると、シナリオ A のリスクは「受容不可」となる。つまり、シナリオ A に対するリスク低減策が必要であることが分かる。「注意」や「観察」のリスクに対してはリスク低減に代わ

り、対象の監視等で対応することもできる。

リスクの低減のためには、発生可能性を下げるのが原則だ。シナリオ発生時の深刻度はビジネス要件に起因するため、セキュリティ対策で低減することができない。深刻度を下げるためには、ビジネス要件の撤廃や縮小といったセキュリティ以外の施策が必要となる。



シナリオ A のリスクを受容可能にまで低減するには、図 8 のように発生可能性を Level1 にまで低減する必要がある。図 6 の全ての攻撃パスにおいて、実現性と難易度が変わるようなセキュリティ対策を導入するのである。今回のケースでは以下の対策とする。

・VPN 装置など、外部との接点となる機器の脆弱性管理を実施する
・エンドポイント製品[4]を導入し、不許可 USB の接続を検知・防御
・機密情報のアクセス権限を最小化し、閲覧する必要のある社員にのみアクセスを付与
・委託先業者に作業員リストと誓約書の提出を義務付ける
・小型 PC に認証機能を実装する

脆弱性管理を実施すれば、VPN 装置への侵入の難易度は変化しないが、攻撃の成功率が低下するため、実現性が下がる。実現性が 3 から 1 に落ちると想定すると、この攻撃パスの発生可能性が Level1 まで下がり、リスクが受容できるようになる。その他の攻撃パスも含め、どこまでセキュリティ対策を実施すればシナリオ全体でリスクが受容できるようになるのか、この取り組みによって明らかになる。

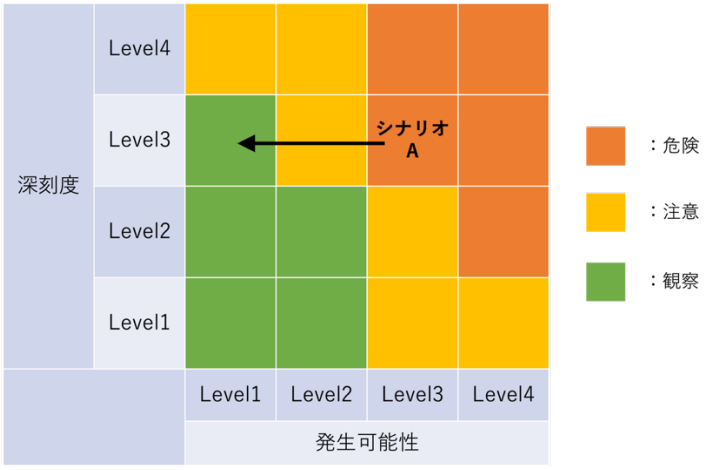


図 8 セキュリティ対策によるリスクの軽減

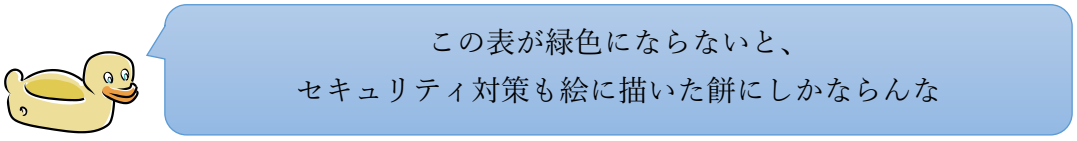
[4] エンドポイント製品自体の脆弱性をつき、攻撃に利用される可能性もある。導入前に十分な検証を行う、製品の評価実績を入手するなどの対応を考えるべきである。

最後の作業として、セキュリティ対策状況の管理を実施する。前述の手順で必要なセキュリティ対策を列挙することができたが、各対策の優先度と予算や体制を考慮しつつ、対策の実施計実施状況を管理する必要がある。

戦略シナリオ、攻撃手順シナリオの対策毎にカテゴリ、該当シナリオ、担当部署、コスト、実施期間、対策状況を列挙し、現状を把握しながら管理すると良い。以下にサンプルを示す。

表 13 セキュリティ対策状況

対策	シナリオ	担当部署	実施障害	コスト	期間	進捗
外部接点のある装置の脆弱性管理	A	情報システム部	部署毎に設置した野良サーバの資産特定が困難	++	6ヶ月	実施中
エンドポイント製品の導入	A	情報システム部		+++	12ヶ月	未実施
アクセス権限の最小化	A	情報システム部		+	2ヶ月	完了
委託先に作業員リストと誓約書の提出を要求	A	購買部		+	3ヶ月	実施中
共連れ入室のしづらいゲート導入	A	総務部		++	6ヶ月	実施中
小型PCに認証機能を実装	A	開発委託先		++	3ヶ月	実施中
小型PCのハードを施錠可能にする	A	開発委託先		++	3ヶ月	実施中



外部接点のある装置の脆弱性管理では、野良サーバが実施の障害となっている。進捗やコストへの影響もあるが、野良サーバが増え続けると管理外の装置がいつまでも残り続けてしまう。野良サーバの設置ルールや検知の仕組みの導入を検討すべきだ。

また、エンドポイント製品導入の着手ができておらず、進捗の遅れが想定される。脆弱性管理と同じく情報システム部が担当しているため、リソース不足が原因と考えられる。リソースの増強、実施時期の先送り検討、並行作業の計画などの対策が必要と思われる。

小型 PC の認証機能とハードの施錠機能は、開発対象システムのセキュリティ要件に盛り込むべき対策だ。

～重要ポイント～

- ✓ **リスク＝事業被害の深刻度×シナリオの発生可能性**
- ✓ **セキュリティ対策は発生可能性を低減する**
- ✓ **セキュリティ対策状況を管理し、適切に実施されていることを確認する**

3. セキュリティ要件

本章では、OWASP SAMM の設計で定義されているセキュリティ要件の実践方法を紹介する。SAMM におけるセキュリティ要件では、２つのストリームに３段階の成熟度レベルが定義されている。

ストリーム A：ソフトウェア要件(Software Requirements)

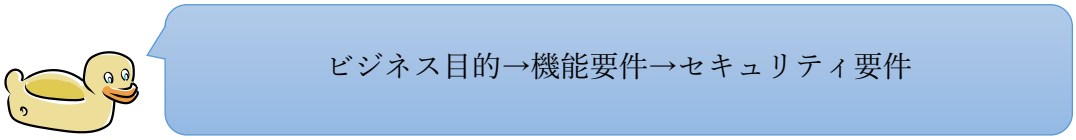
ストリーム B：サプライヤセキュリティ(Supplier Security)

本文書は開発チームのセキュリティ・バイ・デザイン実践の助けとなることをスコープとしているため、ストリーム A の実践に絞って紹介する。また、ストリーム A のレベル３、SR3(Software Requirements)はセキュリティ要件を組織全体で運用する内容であり、開発チーム内に収まらない内容のため、スコープ外とする。

3.1　セキュリティ要件の定義方法

本節では SAMM における SR1 セキュリティ要件定義の実施方法を紹介する。

セキュリティ要件は開発するシステムの機能要件を基に定義する。そもそもシステムにはユーザの利便性を向上したい、新たな価値を提供したい、業務を効率化したいといったビジネス上の目的がある。セキュリティ要件を定義するためには、ビジネス目的に沿った機能要件が定義されていることが重要である。セキュリティ要件を定義する目的はシステムを守るだけではなく、システムを利用するユーザや顧客のビジネスや資産を守ることである。機能要件がビジネス上の目的を達成するための要件となっているかあらためて確認し、機能要件を精査した上でセキュリティ要件を定義すべきである。



２章（脅威分析）で使用した ICS 資産を把握するためのネットワークスキャンツール(図9)を例としてセキュリティ要件定義の具体的な手法を以下に記載する。

注意して欲しいのは、ここで記載する手法は多数ある手法のうちの一部であることだ。自組織で使用している基準や、他のフレームワークから適切なものを選択したり、複数の手法を組み合わせてセキュリティ要件を定義してほしい。

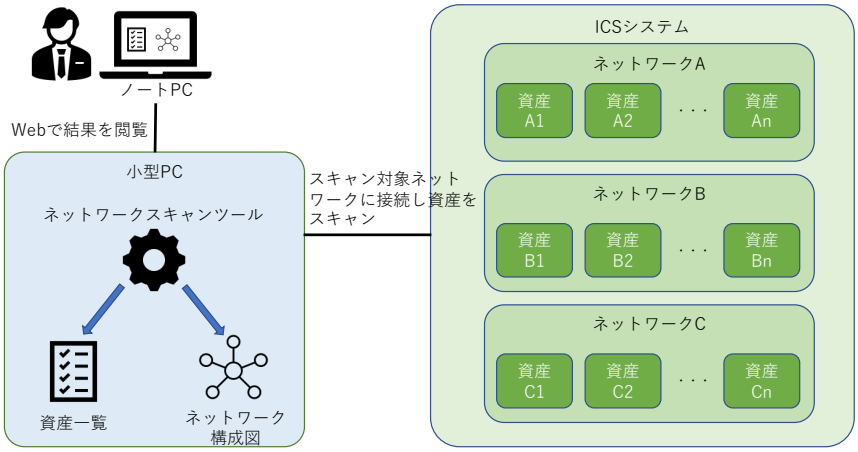
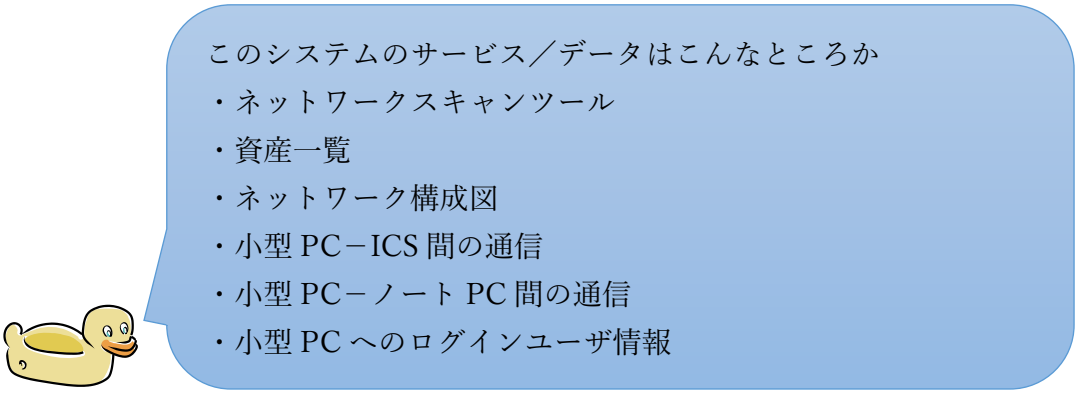


図 9 ネットワークスキャンツール概要図



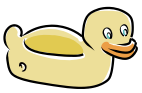
① サービスやデータの機密性(C)、完全性(I)、可用性(A)を考える

本手法は機能要件で利用されるサービスやデータに対し機密性、完全性、可用性の侵害を検討し、そこからセキュリティ要件を抽出するものである。この手法で定義したセキュリティ要件の具体例を表 14 に示す。

表 14 CIA から導き出したセキュリティ要件

サービス/データ	侵害されるCIA	セキュリティ要件
ネットワークスキャンツール	• ツールのAPI情報窃取(C) • ツールの不正操作によるツールやICSの停止(A)	• API情報の管理 • ツールへのアクセス制限や認証
ICSネットワーク構成図	• ICSネットワーク構成図の窃取(C) • ICSネットワーク構成図の改竄(I)	• ネットワーク構成図の暗号化 • ネットワーク構成図の適切なアクセス権設定
ICS資産リスト	• ICS資産リストの窃取(C) • ICS資産リストの改竄(I)	• ICS資産リストの暗号化 • ICS資産リストの適切なアクセス権設定
ノートPC/小型PC間の通信データ	• ツールのAPI情報の盗聴 (C) • ツールのAPIを利用した小型PCの不正操作(A)	• API情報の管理 • ツールへのアクセス制限や認証
小型PC/ICSネットワーク間の通信データ	• スキャン用パケットの盗聴によるICS資産情報やICSネットワーク情報の窃取 (C) • 小型PCやICSネットワークへの不正パケット送信 (I/A)	• 物理的対策（ICS環境への入退場管理やスイッチ等の物理的な接続制限等）
ログインユーザ情報(ID, パスワード)	• ユーザ情報の窃取(C) • ユーザ情報の改竄(I/A)	• ユーザ情報格納データベースのアクセス権限管理

機能要件からサービスやデータを抽出し（今回は図 9 の概要図から抽出)、各サービスやデータから侵害される CIA とセキュリティ要件を定義した。機密性、完全性、可用性は情報セキュリティの 3 要素と呼ばれるが、他にも真正性、信頼性、責任追跡性、否認防止を含む情報セキュリティの 7 要素で考えることで更に網羅的にセキュリティ要件を抽出することができる。

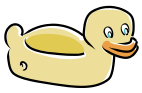


機能要件の各項目に対して「C:情報流出しそうか？」
「I:改竄されそうか？」「A:機能停止しないか？」と考えるべし

② 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、稼働時間に関する期待を示し、それに対するセキュリティ要件を考える。

本手法は機能要件に対して以下の観点で機能要件を細分化し、その結果を踏まえてセキュリティ要件を定義するものである。

・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



「金融システムだから計算ミスは絶対ダメ」
「広告システムだから少し止まっても倍動かせば取り戻せる」
みたいなシステムへの期待を出せばいいのやな。
あんまり期待が大きいと作るの大変やで

①と同様にネットワークスキャンツールを例に機能要件を細分化した例を表 15 に示す。

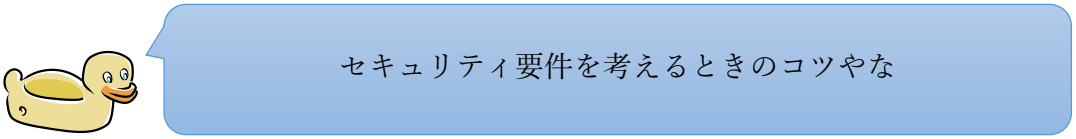
表 15 機能要件の細分化とセキュリティ要件

観点	要件
データセキュリティ	ID とパスワード情報が保護されていること
アクセス制御	—
トランザクションの完全性	ID とパスワードでユーザの正当性確認を行い、サービスにログインすること
ビジネス機能の重大度	重大度：高 （不正ログインされた場合、ICS ネットワークへの侵入や業務停止を引き起こすため）
職務分掌	許可された者のみ認証情報所有できること 担当者の異動時はパスワードを変更する運用とすること
稼働時間	—

細分化した機能要件から導き出されるセキュリティ要件の例として以下のようなものがある。

- ID/パスワードによる認証機能を有し、管理されたユーザ以外ログインできないこと
- ID/パスワードに単純な文字列は使用できないこと
- パスワード変更機能をつけること

機能要件毎にセキュリティ要件を検討するため、網羅的にセキュリティ要件を検討することができる。また、機能要件を細分化することでセキュリティ要件として実装すべき機能が具体的に抽出できる。本手法のポイントとして、ポイントとして機能要件は「～できること」を考えるが、セキュリティ要件は「～以外できないこと」を考えると良い。



どのような観点でセキュリティ要件を抽出するとしても、共通かつ重要なポイントは要件の具体化である。定義した要件が設計や実装のインプットとなるが、後工程で認識の相違が発生しないようにしなければならない。アジャイル開発モデルであれば、スプリントごとに修正できるが、特にウォータフォール開発モデルでは認識の違いが後工程に大きく響いてしまう。機能要件が曖昧だと、セキュリティ要件も曖昧になってしまうため、明確なセキュリティ要件を定義するためにも機能要件の曖昧さを排除することが重要だ。

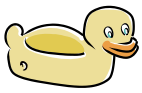
～重要ポイント～

- ✓ **機能要件からセキュリティ要件を定義する手法は多数ある**
- ✓ **定義するセキュリティ要件が適切に実装されるように、曖昧さを排除した明確なセキュリティ要件を定義する**

3.2　ガイドラインに準拠する

本節では SAMM における SR2 セキュリティ要件定義の実施方法を紹介する。SR2 ではポリシーや法令、既知の脆弱性や過去のフィードバックなど、機能要件以外の要素からセキュリティ要件を定義するとしている。これらの要素はガイドラインという形で示されていいることが多く、適切なガイドラインを選定して準拠するべきである。

世の中には公開されているセキュリティガイドラインや標準が数多く存在する。例えば業界毎のガイドライン（電力、ビル、制御システム、Web アプリケーション、金融、医療）や、Web アプリケーション開発におけるガイドライン（OWASP ASVS）、法令や法的ガイドラインなどがある。また、組織においてもセキュリティポリシーやセキュリティガイドラインが既に定義されているかもしれない。参照するガイドラインを選定し、それらのガイドラインからプロダクトのセキュリティ要件として採用しよう。



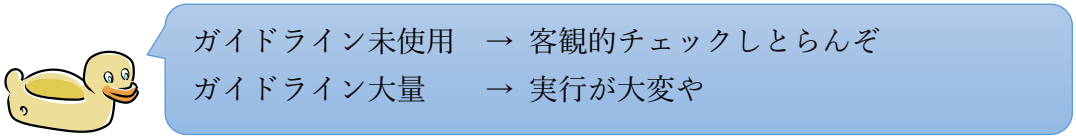
遵守義務のあるガイドライン　→ ちゃんとやろうな
任意遵守のガイドライン　　→ 適当につまみ食い

採用にあたり、事業継続やビジネス環境により必ず遵守しなければならないガイドラインと任意の（遵守することが望ましい）ガイドラインがある。例えば、金融系のシステムの場合は PCI DSS は遵守しなければならない。任意のガイドラインについては様々なものがあるが、セキュリティ要件への採用指針として、以下を満たすものが望ましいと考える。

- 実装が可能であり、コスト効率の良いもの
- 特定の技術やホスティング環境に依存しないもの

上記は組織内の他のプロジェクトからのフィードバックも有益な情報となるため、組織内での利用実績などの情報を踏まえて選定すると良い。

ただ、必要以上に選びすぎるのも良くない。業界等準拠すべきものを開発元のセキュリティ担当者や自社の品質管理部門、セキュリティ担当部門(PSIRT 等)と相談のうえ選択しよう。6.1 に主なガイドラインを記載しているため、参考にすると良い。



Web アプリケーションセキュリティガイドラインの一つである OWASP ASVS に準拠してセキュリティ要件を定義した例を以下に示す。OWASP ASVS は利用する等業界を絞っておらず、アプリケーションの設計、開発、脆弱性診断などにおいて必要となるセキュリティ要件を標準化しているため、非常に有効なガイドラインの一つである。

表 16 ASVS を用いたセキュリティ要件の検討

ch	chapter_name	section_i	req_id	req_description	導入	要件詳細
V2	Authentication	V2.1	V2.1.1	Verify that user set passwords are at least 12 characters in length (at	○	Web IFへのログインパスワードは最低12文字とする
V2	Authentication	V2.1	V2.1.5	Verify users can change their password.	○	Web IFのパスワード変更機能を実装する
V2	Authentication	V2.1	V2.1.6	Verify that password change functionality requires the user's current	○	パスワード変更には現在のパスワードと新規パスワードが必要
V2	Authentication	V2.1	V2.1.7	Verify that passwords submitted during account registration, login, ar	○	開発時点での一般的なパスワードリスト上位１万件を保持し、該当するもの
V2	Authentication	V2.1	V2.1.10	Verify that there are no periodic credential rotation or password histo	○	パスワードに有効期限を設けない
V2	Authentication	V2.1	V2.1.11	Verify that "paste" functionality, browser password helpers, and exter	×	Webログインフォームへのコピー＆ペーストを無効化する
V2	Authentication	V2.1	V2.1.12	Verify that the user can choose to either temporarily view the entire m	○	マスクされたパスワードの可視化ボタンを実装する
V2	Authentication	V2.5	V2.5.2	Verify password hints or knowledge-based authentication (so-called '	○	いわゆる秘密の質問を実装しない
V3	Session Management	V3.1	V3.1.1	Verify the application never reveals session tokens in URL parameters	○	

ASVS のチェックリストを基に導入する項目をチェックする。このとき、導入しないものに対して理由を明記する。そうすることでレビューにおける判断の妥当性を確認しやすくなる。また、ASVS の各項目にはレベルが設定されているため、プロジェクトや組織のセキュリティ基準に従い、導入する項目を決定する。導入対象とした項目は、セキュリティ要件としてシステムの要件に追加する。

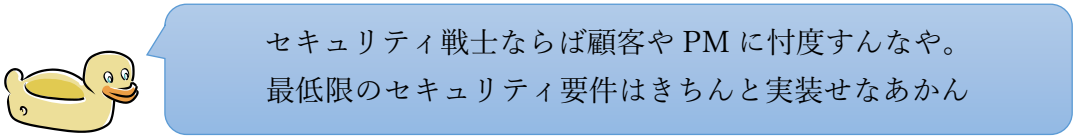
～重要ポイント～

✓　**準拠するガイドラインを選定し、セキュリティ要件を定義しよう**

3.3　対応するセキュリティ要件を決定する

機能要件やガイドラインからセキュリティ要件を定義したが、本当に全部実装できるのだろうか。現実には予算や納期など、開発する上で様々な制約があるため、難しい場合がある。そのようなときはビジネスリスクや脅威を元に最低限実装すべきセキュリティ要件を決めることが重要になる。

実装すべき「最低限」を決定するには、2 章の脅威分析を参考にすると良い。リスクが許容可能なまでに低減されるセキュリティ要件が「最低限」のセキュリティ要件だ。



以下に機能要件やガイドラインから抽出したセキュリティ要件をリスト化し、優先順位を付けた例を表 17 に示す。

表 17 セキュリティ要件の優先順位付け

分類	セキュリティ要件	優先順位
ツールへのアクセス制限や認証	Web IFへのログインパスワードは最低12文字とする	高
	Web IFのパスワード変更機能を実装する	中
	パスワード変更には現在のパスワードと新規パスワードが必要	中
	一般的なパスワードリスト上位1万件を保持し、該当するパスワード設定はできないようにする	中
	パスワードに有効期限を設けない	中
ネットワーク構成図の暗号化	独自実装された暗号化方式ではなく、業界で実績のある、または政府が承認した暗号化アルゴリズム、モード、およびライブラリが使用されている。	高
	最新の勧告を使用して、暗号初期化ベクトル、暗号設定およびブロックモードが安全に構成されている。	高
・・・	・・・	

優先順位は、プロジェクトや組織のセキュリティ基準、セキュリティサービスやデータの重要度、攻撃リスクなどを踏まえてつけ、重要な対策を見える化する。

～重要ポイント～

- ✓ **セキュリティ要件を全て実装するにはコストや時間がかかる。優先順位をつけよう。**
- ✓ **どこまでのセキュリティ要件を実装するのか、脅威分析を参考に決定しよう**

3.4 セキュリティ要件の決定はスタートラインに過ぎない

ここまでセキュリティ要件をどのように決めるかを見てきたが、これはセキュアなシステムを開発するための第一歩に過ぎない。開発プロセス全体でセキュリティを確保していく必要がある。そのためのポイントを以下に紹介する。

設計工程でセキュリティ要件をブラッシュアップする

設計工程でシステムのアーキテクチャ、具体的なデータ、利用するフレームワークなどが決まるが、設計工程の成果物を元に再度脅威分析やセキュリティ要件を見直したり、要件に対してフィードバックすることが重要である。ビジネス要件とアーキテ

クチャの両面から必要なセキュリティ機能を抽出しよう。

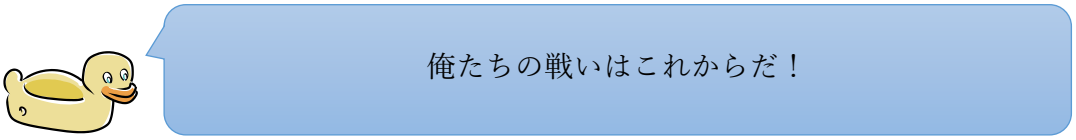
各開発工程でセキュリティ要件を反映する仕組みを作る

定義したセキュリティ要件がソフトウェアや稼働中のサーバの設定などに確実に反映されることが何よりも重要である。セキュリティ要件が設計や実装工程における成果物に反映されているか、テスト工程でセキュリティ機能が正しく動作しているかを漏れなく確認するために、要件に対するトレーサビリティをチェックしよう。また、組織で行われるプロジェクトの監査にセキュリティ要件の監査を含めるなどの手段も有効である。

セキュリティ要件は日々変化する

ソフトウェア技術やそれに伴う脆弱性は日進月歩のペースで進化していく。リリース時にはセキュアなソフトウェアでも 1 年後にはセキュアである保証はない。このため、段階的なリリースやバージョンアップの計画があるならば、その度にセキュリティ要件の再定義を行うべきである。しかし、中には 1 度リリースしたら数十年稼働し、メンテナンスやアップデートができない種類のソフトウェアも存在する。このような場合、その事情も考慮したセキュリティ要件の定義や運用を実施する必要がある。

セキュリティ要件を策定するだけでは意味がない。作りたいのは要件ではなく、ソースコードである。セキュアなソースコードを書きサーバに実装されて初めて意味がある。



～重要ポイント～

- ✓ **開発プロセス全体でセキュリティ要件の実装状況をチェックする仕組みを作ろう。**

4. セキュリティアーキテクチャ

「セキュリティアーキテクチャ」とは、IT 製品において、脅威に対抗するセキュリティ機能そのものを安全に実装するための、設計方針・プログラムの構造・しくみのことである。本章では、OWASP SAMM の設計で定義されているセキュリティアーキテクチャの実践方法を紹介する。SAMM におけるセキュリティアーキテクチャでは 2 つのストリームに 3 段階の成熟度レベルが定義されている。

ストリーム A：アーキテクチャ設計（Architecture Design）

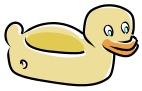
ストリーム B：技術マネジメント（Technology Management）

セキュリティアーキテクチャでは、セキュリティ原則の適用やセキュリティサービスの利用、ツールや技術の利用など、様々な面からセキュリティ設計のためのアプローチを記載しているが、本書ではストリーム A の SR1「セキュリティ原則の適用」およびストリーム B の SR1「ツールや技術の特定」の実践に絞って紹介する。

4.1　セキュリティ原則の適用

設計作業に入る前にセキュリティ原則を記述したリストを作成しよう。設計者はこのリストを常に参照しながらアーキテクチャに落とし込んでいく。また、レビュー時にもセキュリティ原則を遵守しているかチェックする。

このリストに記述する原則は OWASP SAMM、ASVS や情報セキュリティの 3 要素（機密性、完全性、可用性）、業界毎のガイドラインを参照して定義しよう。セキュリティの有識者を交えて作成することが望ましい。



要件に対して CIA チェックしたから、
設計に対しても同じことをやるということや

～重要ポイント～

- ✓ **設計に適用するセキュリティ原則を定める**
- ✓ **既存のガイドラインを活用するとよい**

4.2　ツールや技術の特定

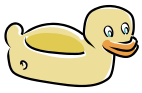
ソフトウェアにはそれぞれ独自の機能がある一方、入出力や通信、GUI など多くのソフトウェアに共通して実装される機能も多い。これらの汎用的な機能をそれぞれのソフトウ

ェアで独自に実装するのは効率が悪い。更に、バグや脆弱性が発生するリスクもある。

このため、SAMM では共通的な機能についてはサードパーティー製のライブラリやフレームワークを活用することを推奨している。

当然のことながら、サードパーティー製ソフトウェアにも脆弱性が存在しうるため、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、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。

また、サードパーティー製ソフトウェアの出自にも注意するべきである。製品としてサポートが提供されているもの、技術力のある団体が提供しており、広く使用されて信頼できる OSS、匿名の個人提供であまり更新されないライブラリ。出自によっても安全性が変わってくる。



OSS あるある
「品質部門が許可してくれない」「脆弱性調査が面倒→自作」
→けしからんじゃないか！！

本節ではこれらの実践例として、ソフトウェア・コンポジション解析ツールである Black Duck を活用した Docker コンテナの利用例を紹介する。Black Duck はアプリケーションやコンテナに含まれるオープン・ソースおよびサードパーティ製のコンポーネントのセキュリティ、品質、ライセンスなどのリスク管理を行うツールである。OSS を効率よく可視化でき、脆弱性情報のデータベースがタイムリーに更新されるため、継続的な監視を行うツールとして非常に有効である。

2 章および 3 章で具体例として挙げたネットワークスキャンツール開発において、デプロイ環境に Docker コンテナを採用し、Docker Hub の公式コンテナイメージを利用して実行環境を構築した。構築した Docker コンテナに Black Duck のコンテナスキャンをかけたところ、内包される OSS の脆弱性が 100 件近く検出された。

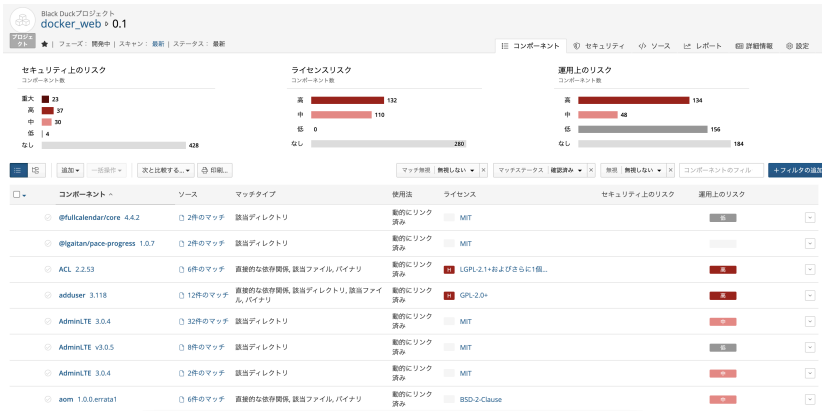


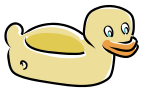
図 10 Black Duck のコンテナスキャン結果

これらの脆弱性に対する対策として、最小構成の Ubuntu コンテナイメージを取得し、開発するシステムに必要な OSS を一つずつインストールした。こうすることで、不要な OSS を含まない環境を構築し、脆弱性の件数を 7 件にまで削減できた。

この事例から、不用意な OSS の利用はシステムに脆弱性を混入してしまうこと、OSS の調査や検査が重要なことがわかる。

ただし、OSS の脆弱性は件数が少なければよいというものではない。対象システムが利用される環境において、起動しないサービス、権限が小さいプロセスなど、サイバー攻撃に利用されるリスクが低い脆弱性も存在する。脆弱性の母数を減らすことも重要だが、脅威度の高い脆弱性を個別に解消する取り組みも必要だ。

また、脆弱性は日々新しく発見されるため、出荷後も含めて定期的に調査・検査することを推奨する。



脆弱性調査はツールとか活用して定型業務か出来るとよいな。
ただし、セキュアかどうかの判断は定型化したらあかんで！！

～重要ポイント～

- ✓ **SAMM ではサードパーティー製ソフトウェアの活用を推奨**
- ✓ **使用する OSS の調査・検査を実施する必要がある**
- ✓ **脆弱性がもたらすリスクを考慮し、リスクが大きい脆弱性が残らないよう注意**
- ✓ **OSS の調査・検査は出荷後も継続して実施**

5. 下流工程のセキュア開発

１～４章まではセキュリティ・バイ・デザインで上流工程におけるセキュリティの確保について述べてきた。セキュリティの確保は上流だけではなく、下流工程におけるセキュリティの対策の実施も重要である。本章では上流だけではなく全ての開発プロセスにおけるセキュリティ確保のための手法であるシフトレフト、および下流工程での具体的なシフトレフトの導入事例を紹介する。

5.1　シフトレフトとは

セキュリティ・バイ・デザインでは主に上流工程おけるセキュリティの確保に関する概念に対して、シフトレフトは安全なものを開発するための現状把握や振り返り、対策へのフィードバック、改善を行うプロセスのことである。
これまで後工程で行っていたセキュリティに関する問題について発生原因まで遡り、その発生原因に対して根本的な解決を行うことで、従来よりも早い段階にセキュリティ対策を組み込むことができるというものである。

最近「動的な脆弱性スキャンを何回もかけること」自体をシフトレフトであると耳にするが、これは手段であってシフトレフトの本質とは少し異なる。

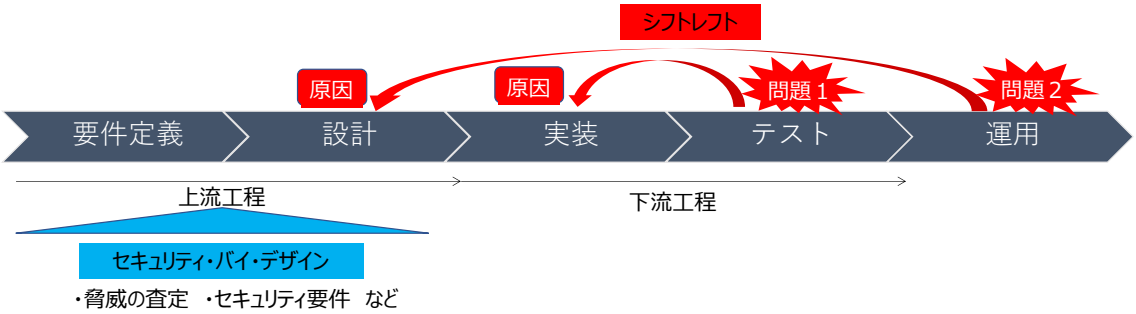
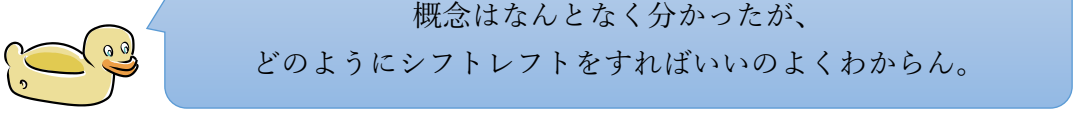


図 11 シフトレフトとセキュリティ・バイ・デザイン
12



～重要ポイント～

- ✓ **セキュリティの確保は上流だけじゃなくて、下流工程におけるセキュリティの対策の実施も重要である。**
- ✓ **シフトレフトは後工程で実施するセキュリティ対策を前工程に単純にシフトするのではなく、本質は発生原因の根本解決である。**

5.2　シフトレフトの実践

CI/CD による下流工程でのシフトレフトの実現方法を紹介する。より具体的な内容にするため、本章では卒業プロジェクトで実施した CI/CD 環境の導入事例を紹介する。CI/CD とセキュリティツールを組み合わせることで以下の効果が得られる。

- セキュリティツールの実施漏れを防ぐ
- スキルや知識が低い開発メンバーでも同じように実行できる
- 自動化により開発効率化に繋がる

尚、今回紹介する導入事例は一例であり、シフトレフトの本質は問題解決であるため、下流工程でのシフトレフトの実現方法は他にも存在する。環境依存なども踏まえつつ組織におけるセキュリティの問題点からシフトレフトの実現手段を検討する必要がある。

STEP 1　問題に関する発生原因を洗い出す

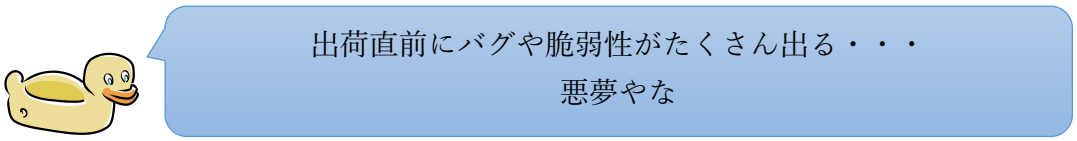
まず初めに、システム開発時に生じた問題や過去に起きたセキュリティインシデントからシフトレフトすべき問題を決める。※山のようにある解決すべき問題がある場合は、よりクリティカルでビジネスインパクトの大きい問題から着手するのが良いのである。
問題を決めたら、その問題に至った原因を洗い出し発生原因に遡る。

■実践例

問題例：ユーザ企業側の受け入れテスト時に、重大な脆弱性がたくさん出て、
　　大規模なソースコードの変更や、OSS のバージョンアップが必要になった。

原因１：　セキュア開発に導入に手を回す余裕がなかった。
原因２：　セキュア開発を実施できる従業員が少なかった。
原因３：　仕様変更により、セキュリティ検査をする時間がなかった。

原因４： 経験不足により、何からやればいいかわからなかった。



STEP 2　解決案の検討

発生原因から紐づく解決策を検討する。インシデント発生時は「開発手順書のフォーマットに○○を記載する」等アドホックな再発防止策をあげてしまいがちだが、シフトレフトでは問題を本質的に解決できるような解決案を検討する必要がある。

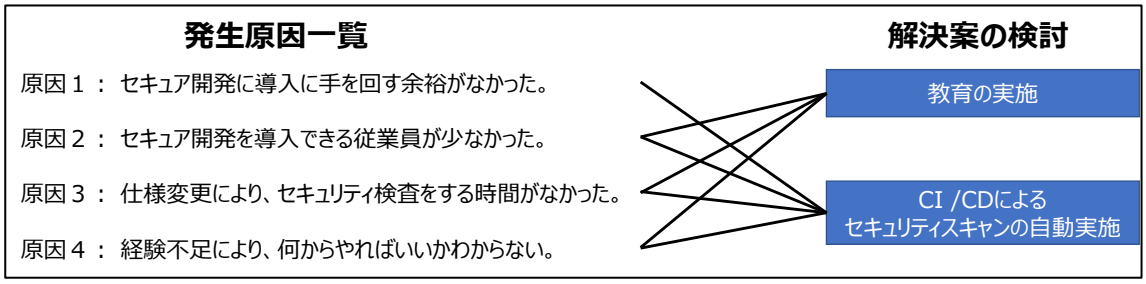
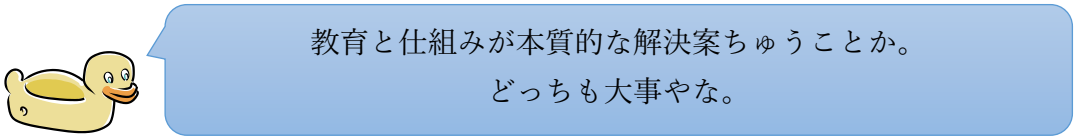


図 13 発生原因と解決案のマッピング例



STEP 3　解決策の導入

解決案の一つである CI/CD によるセキュリティスキャンの自動実施の方法を導入する。本卒業プロジェクトで導入した GitLab を用いたセキュリティスキャンを自動実行する環境の構成図を以下に示す。尚、本環境には Synopsys 社のセキュリティツールを利用し、GitLab の CI 機能と連携して自動実行する環境を構築した。

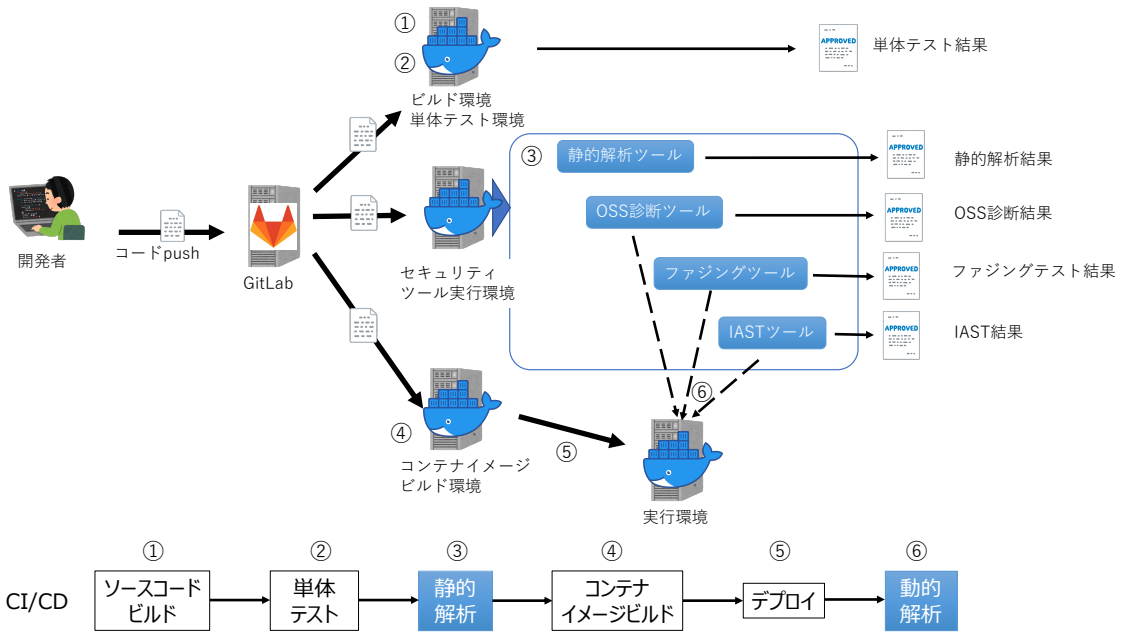


図 14 CI/CD 環境構成図

① ソースコードを Git で管理し、開発者がソースコードを Git リポジトリに push する。GitLab の CI 機能によりビルド用のコンテナを作成し、コンテナ上で自動ビルドを行う。

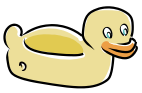
② ソースコードと同じく単体テストコードも Git で管理し、テストコードを Git リポジトリに push する。GitLab の CI 機能を用いてビルドと同時に単体テストを自動実行する。

③ 静的解析ツールを実行するための環境にソースコードをコピーし、GitLab の CI 機能により静的解析ツールを実行する。

④ デプロイ用のコンテナイメージを作成する。

⑤ コンテナイメージを実行環境にデプロイする。

⑥ 実行環境に対して OSS 診断ツール、ファジングテストツール、IAST(インタラクティブ・アプリケーション・セキュリティ・テスト)を CI 機能により実行する。



まずは問題ポイントを探してもらえるのは助かる！
あとは・・・取り組むだけだ！（白目）

～CI/CD とは～

・CI

「Continuous Integration」の略で、継続的インテグレーションと訳される。通常手作業で行っているソースコードの統合（インテグレーション）作業や、プログラムの不具合を見つけるテストを自動化し、より早く修正できる取り組みである。複数人でシステム開発している現場で考えると、ソースコードを共通のリポジトリに統合し、クラスや各種モジュールなどの構成要素に対するテストを自動化することができる。

・CD

「Continuous Delivery」の略で、「継続的デリバリー」と訳される。通常手動で行っているビルド、本番環境へのデプロイ作業を自動化し、いつでも最新のシステムをリリースできるという取り組みである。

6. 参考文献

6.1　主なセキュリティガイドライン

国際標準や業界毎のガイドライン

分類	ガイドライン名称	内容
国際標準	ISO 22301	事業継続マネジメントシステム（BCMS）に関する国際規格
	ISO 22313	ISO 22301 使用に関する手引
	IEC 62443	制御システムのセキュリティ
	ISO/IEC 27000 ファミリー	情報セキュリティマネジメントシステム
	ISO/IEC 15408	コモンクライテリア
	ISO/IEC 27030 ファミリー	セキュリティコントロールサービス
	ISO/IEC 24760 ファミリー	アイデンティティ管理
米国標準	NIST SP800 シリーズ	コンピュータセキュリティ関係のレポートやガイドライン
	FIPS	連邦情報処理標準
国内標準・ガイドライン	デジタル社会に向けた重点計画	デジタル社会の実現に向けた羅針盤としての重点計画
	政府機関等のサイバーセキュリティ対策のための統一基準群	ISO 27000 シリーズおよび NIST SP800-53 に対応する政府統一基準
	情報システムに係る政府調達におけるセキュリティ要件策定マニュアル	セキュリティ・バイ・デザインに関する政府公表文書
金融	金融機関等コンピュータシステムの安全対策基準・解説書	FISC が発行する金融機関等の情報システムのための安全対策基準
クレジットカー	PCI データ・セキュリティスタンダート	クレジットカード会員デー

ド業界	(PCI DSS)	タを安全に取り扱うためのセキュリティ基準
オープン POS	製品分野別セキュリティガイドライン：オープン POS 分野	POS 端末や POS サーバ等に関するセキュリティガイドライン
OT	制御システムのセキュリティリスク分析ガイド	IPA 発行の制御システム向けのセキュリティ対策におけるリスクアセスメントの実施と活用
	重要インフラにおける情報セキュリティ確保に関わる安全基準等策定指針	各重要インフラに共通して求められているセキュリティ対策の指針
IoT	IoT 開発におけるセキュリティ設計の手引	IoT 機器およびその使用環境で想定されるセキュリティ脅威と対策を整理した文書
	IoT セキュリティガイドライン	IoT 機器やシステム、サービスに対してリスクに応じた適切なサ イバーセキュリティ対策を検討するための考え方
	CCDS 製品分野別セキュリティガイドライン：IoT ゲートウェイ	個々の製品分野における具体的なセキュリティ設計・開発を行うためのガイドライン
電力	JESC Z0004 電力制御システムセキュリティガイドライン	電力制御システムにおいて実施すべきセキュリティ対策を規定したもの
	JESCZ003 スマートメータシステム・セキュリティガイドライン	スマートメータシステムにおいて実施すべきセキュリティ対策を規定したもの
原子力	IEC 61513	原子力分野の機能安全規格
自動車	ISO 26262	自動車分野の機能安全規格
	CCDS 製品分野別セキュリティガイドライン：車載分野	個々の製品分野における具体的なセキュリティ設計・開発を行うためのガイドラ

		イン
医療	医療情報システムの安全管理に関するガイドライン	医療機関等が遵守すべき情報セキュリティの規定
	IEC 60601	医療用を目的とした電気機器の基本的性能と安全性を定める一連の技術規格
	Content of Premarket Submissions for Management of Cybersecurity in Medical Devices	医療機器のサイバーセキュリティ管理のための販売前申請内容に関するガイダンス
	Postmarket Management of Cybersecurity in Medical Devices	医療機器のセキュリティ向上に関する推奨事項をまとめた文書
	IMDRF ガイダンス	医療機器のサイバーセキュリティに関する国際整合を図るための一般原則とベストプラクティス
	医療機器のサイバーセキュリティ導入に関する手引書	厚労省が IMDRF ガイダンスを参考に国内向けに作成した手引書
個人情報・プライバシー	GDPR	個人データ保護やその取り扱いについて詳細に定められた EU 域内の各国に適用される法令
	個人情報保護法	個人情報の取扱いに関連する日本の法律
	電気通信事業における個人情報保護に関するガイドライン	電気通信サービスのり弁性の向上を図るとともに利用者の権利利益を保護することを目的としたガイドライン
	JIS Q 15001	プライバシーマーク

技術要素毎のガイドライン

分類	ガイドライン名称	内容
Web アプリケーションセキュリティ	OWASP Top 10	Web アプリケーションのセキュリティに関する重大なリスクについてのレポート
	OWASP Application Security Verification Standard (ASVS)	Web アプリケーションと Web サービスの設計、開発、テストに必要なセキュリティ要件とコントロールのフレームワーク
	OWASP Proactive Controls	OWASP Top10 で示されている脆弱性を作り込まないようにするための事前の対策ガイドライン
	OWASP Software Assurance Maturity Model (SAMM)	ソフトウエアセキュリティ保証成熟度モデル
コンテナセキュリティ	NIST SP800-190	アプリケーションコンテナセキュリティガイド
クラウドセキュリティ	クラウドセキュリティガイドライン(経済産業省)	クラウドサービスの利用者およびクラウドサービス提供者向けのガイドライン
セキュアコーディング	CERT C コーディングスタンダード	
	MISRA C	
	Android アプリのセキュア設計・セキュアコーディングガイド	

6.2　本書執筆における参考文献

- OWASP, Software Assurance Maturity Model 1.0, 2009
- OWASP, Software Assurance Maturity Model 2.0, 2020
- ANSSI, EBIOS Risk Manager, 2018
- Greg Hoglund and Gary McGraw, セキュアソフトウェア, 2004
- 北山 晋吾, GitLab 実践ガイド, 2018
- 市川 豊, Docker コンテナ開発・環境構築の基本, 2021
- 古賀 政純, Docker 実践ガイド 第 2 版, 2019

おわりに

サイバー攻撃は年々増加し、世界中で被害が拡大している。社会のデジタルシフトによってインターネットに接続するシステムが増えており、またサイバー攻撃に利用するツールやサービスが簡単に入手でき、サイバー攻撃を受ける機会も増えている。開発したシステムに脆弱性が存在すると、そのシステムにはサイバー攻撃を受けるリスクが生じ、企業活動やブランドイメージに多大な影響を与える可能性がある。システムに脆弱性を作り込まないためにはセキュア開発は重要であり、セキュリティ・バイ・デザインを実践することで安全なシステムを開発することに繋がるだろう。

セキュリティ・バイ・デザインを導入、実践するためには技術や人、組織的対応など、実施すべきことは多岐に渡るが、GAFA を始めとする米国の主要 IT 企業は古くから取り組んでおり、今や常識になっている。日本のデジタル庁など政府の官公庁や主要 IT 企業など、様々な企業や団体も積極的に取り組んでいる。また、我々産業サイバーセキュリティセンター中核人材育成プログラムのメンバーもセキュリティ・バイ・デザインの考え方を学び、短い期間（約 1 ヶ月程度）でシステムのセキュリティ対策を検討することができた。

本書を通して、セキュリティ・バイ・デザインへの理解を深め、一つでも多くシステム開発の現場に導入するための助けになれば幸いである。

巻末コメント

セキュリティを「やってる風」で終わらせないために

奈良先端科学技術大学院大学
門林　雄基

このドキュメントを読んでいる開発者の方、あるいはリーダー層の方々は「セキュリティはどこまでやればいいのか」あるいは「セキュリティには既にかなりコストをかけている」と感じておられる方も多いだろう。

常に納期と仕様変更に追われる現場にとって、またクラウド対応、プロダクトライン開発、アジャイル開発など新たな技術的トレンドに事欠かないリーダー層にとって、セキュリティバイデザインに取り組むなどというのは夢物語に思われるかもしれない。

しかし急成長を遂げている企業ではセキュリティバイデザインというのは当たり前の話であり、クラウド対応、アジャイル開発なども当たり前の話なのだ。私は、「開発という言葉だけ同じで、住んでいる世界も使っているツールも全然違うよ」とこのドキュメントの執筆陣に最初に申し上げた。結局、営業より技術のほうが強い会社、開発リソースを消耗するだけの謎な顧客要望を突っぱねられる会社だけが余裕をもってセキュリティバイデザインに取り組めているのではないか。

安心、安全なソフトウェアを作るためにはセキュリティについてじっくり検討する時間的余裕と人的余裕が必要だと私は思う。優先順位の問題なのである。人命を預かるシステム、個人情報や国民の財産を預かるシステムが増えた今日、２０年前と同じ優先順位でソフトウェア開発をやっている会社は市場から淘汰されるべきではないか。これはたんなる一個人の私見ではなく、IoT セキュリティや重要インフラのセキュリティをめぐる各国の最新規制や判例をつぶさに見て感じている温度感である。

とはいえ、セキュリティは抽象的な話が多い。現状維持と細かな品質改良が得意な日本企業にとって、抽象的なアドバイスだけでルビコン川を渡るような大英断を下すのは難しいだろう。この導入指南書では抽象的なアドバイスを咀嚼し、具体例に落とし込んで担当者目線でやるべきことを書き連ねてある。読者諸兄は、ルビコン川の向こう岸の地図を手に入れたようなものである。

アーキテクトの視点では、開発者の微視的視点でのセキュリティだけに満足しないように気をつけたい。泥粘土をこねているだけでは安全な建物が手に入らないのと同様に、いくら const, private, final 等々書き連ねてみても構造的安全性を考慮しなければセキュアなシステムは手に入らない。我々は欧米のリスク分析フレームワークを俯瞰し、今

日時点で最も優れていると思われるフランスのリスク分析フレームワークを演習で用いており、２章はその一例となっている。

この導入指南書が、日本のソフトウェア産業の国際競争力向上の一助となれば幸いである。

セキュアな開発が目指すべきこと

OWASP Japan Chapter Leader
株式会社アスタリスク・リサーチ 代表取締役
岡田良太郎

「セキュアな開発を実践したい」という企業は多くなってきましたが、実際にこれを実現する手立てや、その全体像をイメージできる企業は多くありません。

2022 年現時点、ソフトウェア構築は、まだまだ、攻撃に対する耐性の低いシステムを作ってしまいやすい世の中です。もちろん、数々のフレームワークやプラットフォームを引き受けるクラウドサービス、またサードパーティーのコンポーネントが充実し、よくあるセキュリティ問題を軽減することに貢献していきたことは、否定しません。

しかし、同時に、それらが、システムのブラックボックス化を進めることになったため、実装レベルのばらつきが分析しづらくなっており、ユーザサイドの設定不足、モニタリング不足など、新たなシステムの問題を生じさせているのです。

さらに、同じ企業の中で起きていることがあります。システム開発に携わる方々がサイロ化されていると、それぞれの目的、情報、実践手段の大きなコンフリクトが生じ、この推進を複雑にしています。

ビジネス目的、すなわちシステムが出来上がった時に産まれる提供価値や世界観などが「共通目的」「共通のゴール」になると、チームはサイロ化したり対立したりすることなく、ひとつの目的に向かって協力し合うことが自然にできるはずです。

そこで、方法論に甘んじることなく、行動の目的や目標、つまり HOW の前に WHY を問い、そこで一致していかなければ、シフトレフトもセキュア開発も進んでいかないということです。

サイバー攻撃によって生じうるリスクによって、その価値が毀損されてしまうということへの強い抵抗感が、強くしなやかなシステムを作っていこう、そういうものを支えるソフトウェアにしていこうという、団結の動機付けになるはずでしょう。セキュリティが、アプリケーション開発の一部としてなじんでいくチームの特徴は、まさにこのようなものなのです。

その視点からすると、この研究成果のドキュメントが、ビジネス目的と脅威シナリオを検討し、その上で対策を具体的にディスカッションしていくべきであると論じていることには、本当に大きな価値があると思います。

ただ、稼働中のビジネス面での、またシステム運用面での課題をどのように取り込むか、新たな技術のリスクをどのように評価するか、そしてそれをいかに全員と共有していくかなど、まだまだ洞察や技術の進歩も必要な領域があり、それらはこれからもっと多くの人たちの中でプラクティスを発見していかなければなりません。

この成果が、そのような試行錯誤と、努力の成果の共有に大きな刺激となることを期待します。

ソフトウェアのセキュア開発の投資対効果と自由な試行錯誤ができる社内環境の実現

情報処理推進機構　産業サイバーセキュリティセンター　サイバー技術研究室
登 大遊

(1) セキュア開発は、価値の高いソフトウェアも生み出すので、一石二鳥である

ソフトウェアのセキュリティを保つことは、重要であるが、追加の時間がかかり、コスト増につながる。同じものを作るとき、いんちきなやっつけ仕事のプログラム（従来の外注型の日本組織の業務用アプリケーションによく見られる）と比較して、セキュリティをきちんと考えたプログラムは、開発コストがかさむ。そこで、セキュア開発は、攻撃に対するやむを得ない防御コストとして認識されることが多い。セキュア開発は損失原因になり、何ら実利益を生じさせないと考えてしまいがちである。

しかしながら、うまくセキュア開発を行なうことにより、セキュリティと無関係な、ソフトウェアの本質的価値において、かなりの組織的利益が生じるのである。以下、その理由について述べる。

そもそも、組織内のソフトウェアは、安定して動作し、不具合が少なく、業務ロジックの変化等の必要に応じて柔軟に拡張することができて初めて、結果として長期間使われて、大変高い価値が生じるのである。いずれかの要素に欠けるソフトウェアは、短時間で破綻する。そこで、どうすればこのような価値の高いソフトウェアを作ることができるか。これは、開発過程において、気を抜かずに隅々まで技術上の配慮をして高品質な開発をするという地味な作業の遂行が、唯一の解法である。ところが、これはとても難しいことである。ソフトウェアの品質や価値の追求を目指すとなると、青天井であり、どこまでやれば良いのか、具体的にどうすれば良いのかわからず、なかなか、モチベーションが生じないのである。

しかし、幸運にも、最近の社会はセキュリティが重要であるということになっていて、セキュア開発を行なわないといけないという社会的圧力も生じている。そこで、今、ソフトウェア開発においてセキュリティを重視した開発を行なおうとすることにより、ソフトウェアの価値の本質である安定性、品質、柔軟な拡張性なども自然に高まるという、良い副次的効果を得ることができる。ここで、副次的効果と言ったが、実際には副次的効果のほうが、我々が本当に欲しいものではなかろうか。

すなわち、高いレベルのセキュリティを実現しようとするために色々な勉強をし、いざ設計や実装をするときにもセキュリティを意識して注意して開発するようになると、自然に、ソフトウェアの動き、特に例外的な状態となったときの予期しない動作についてイメージが湧くようになる。そして、あれやこれやの例外的状態でも問題が生じないように、穴を塞いだり、粗い部分をなめらかにしたりしたくなる。手抜きをすると、気持ち悪いという気分が生じるくらいに、完全性を追求する意識が高まるのである。このような良い習性が一度身に付くと、セキュリティに一見無関係な部位の品質も自然に高まる。その結果、長期安定して動作するプログラムが、自然に出来上がる。セキュリティも高まり、作られたプログラムも価値が高いものになるから、これは、一石二鳥である。

(2) セキュア開発は絶妙なバランスを要する投資行為であり自組織で行なう必要がある

このように、セキュア開発は、一見、サイバー攻撃による損害を避けるためのやむを得ない対策という程度の払い損的コストのように見えるが、実は、同時にソフトウェアそのものの品質、安定性、柔軟性を実現するというもともと困難であった課題が自然に実現され、結果として長期的価値が生じることが期待されるので、大変能率の良い投資行為であるということができるのである。

そして、コードを一行書き、機能を１つ作るという都度に、隅々までどれくらい気をつけて仕事をすればよいか、という日々の投資判断のバランスが最も重要であるが、このバランスをとるという毎日の作業は、投資の主体である組織自らが実施しなければならない。他組織に金銭的報酬を払い、仕様だけ指示して、外注して任せると、あまり良い結果にならない。なぜならば、「このコードには、どの程度のコストをかけて、どの程度のリスクを採るべきか？」というバランス判断ことが、まさに投資行為であるが、その絶妙・最適な値は、自組織そのものにしか分からないからである。仕様書のような計画的文書に、予め、指示を書くことができない。自転車のハンドルを握ってでこぼこ道を走るとき、あらかじめハンドルの角度を仕様書に定義しても意味がないのと同じである。毎日ミーティングを開いて「本日の自転車のハンドルの角度」を決定しても意味がない。自転車の操作では、１秒１秒の頭脳内での演算と外界からのフィードバックをも

とに、ハンドルを仔細に操作する必要があるためである。ソフトウェア開発におけるバランス判断も、これとまったく同じである。

業務委託契約関係では、どうしても仕様と報酬に基づく義務の履行という性質が強いから、外注先のエンジニアにおいては、いくら良心的であっても、やはりできるだけ注意力を減らして一応の水準に達したと言えるギリギリのところで仕事を納品して楽をしたいという誘惑に負けてしまう。発注者がこれを外からコントロールすることは至難の業である。なぜならば、ソフトウェアのセキュリティや品質上の問題は、建造物でいうとネジ、クギ１本１本の質や施工方法のような一度組み上がって見えない部分に遍在的、確率的に存在する色々な隠れた問題に起因しているのであって、次第に時間が経ったり、周辺環境が変化したりすると、次第に顕在化してきて、内部から腐敗して傾いてくるものであるからである。そこで、表面的な綺麗さではなく、内部の健康状態が重要である。継続して動作するソフトウェアを実現するために必要な水準に達する高いレベルの健全性を実現するには、前記の投資バランスを日々取る必要があるから、その責任を持つ主体である組織の構成員そのものが実際に手を動かして取り組む必要がある。外注者に自社の投資判断を委ねることはできない。

(3) どのようにすれば自組織のソフトウェア能力を引き上げることができるか

このように考えると、投資対効果が組織の存続に直結するような重要なソフトウェアであればあるほど、社内で快適な開発環境を作り、内製開発をしたほうが良いということになる。そのような内製開発を行なうときに重要なのは、自社人材の育成である。そして、人材育成は、ソフトウェアやセキュリティに関係する試行錯誤を許容する環境があって初めて実現できる。まずは手作りで、あまり固いことを考えずにプログラムを作ってみて、仲間内や組織内、場合によってはインターネット上でわいわいと公開したりして、色々な不具合が生じたり、脆弱性が出たりして、その結果のフィードバックを得て切磋琢磨することにより、初めて、レベルが上がるのである。

このレベル上げ競争は、いわば、サイバー攻撃者との技術力競争でもある。そのレベルが、対象システムにおいて想定される脅威となるサイバー攻撃者のレベルを超えた状態を、何とか維持し続けている組織は、存続できる可能性が高い。そうでない組織は、早晩、サイバー被害に遭って破綻するのである。IT、サイバーというものは、このように、大変に恐い世界でもある。

(4) そもそも事業存続は自組織能力レベルの維持が必須条件である

しかしよく考えてみれば、このような能力レベルの違いで存続か破綻かが決まってしまう大変に恐い世界であるということは、何も IT、サイバーに限らず、すべての事業領域で、同じである。

すべて事業というものは、なんどきでも攻撃を加えてくるかもしれない自然の脅威および人的脅威よりも、自組織との能力レベルを常に高めておくという行為を継続的に行う営みである。したがって、日本の各組織における人材能力レベルを高め維持するために、試行錯誤を許容するための人材育成環境の成立は、IT の面だけみても、サイバーセキュリティの面と高品質なソフトウェアの面の両方の実現において、これからは、ほとんど必須的要素といって良いほど、極めて重要な条件となるのである。繰り返しになるが、IT 以外のすべての事業領域においても、これは同じことである。

(5) 楽しい試行錯誤環境を実現するためにどのようにして経営者を説得すれば良いか

それでは、IT の面における試行錯誤環境の実現のために、我々は、どのようにして組織経営者を説得すれば良いか。経営者というものは忙しいし、コンピュータのことはあまり良く分かっていないケースも多いのであるが、最近、経営者にとってはサイバーセキュリティ事故が発生することは大変に嫌な事であるから、それを避けるために必要な提案は真剣に聞き、一定の合理性と熱意が認められれば、何としてでもそれを実現しろという具合に、積極的姿勢に転じるであろう。

このように、今は、セキュア開発の実現という名目においてソフトウェア技術を磨きたい、そのためには環境と自由裁量が必要だと説明し、経営者を説得し易い、良い時代であるということができる。

そこで、このようなセキュリティの話と共に、上記の一石二鳥理論も併せて説得し、組織内に自由裁量が与えられた試行錯誤環境を作るとよい。そのような組織は日本ではまだ最近少ないから、よく目立つ。業界において、若者の間でも評判が良くなる。そうすると、そのうち、ソフトウェアに限らず、色々な面で良い技術者が集まってきて栄え、長年の繁栄が実現できるであろう。

謝辞

本書を作成するにあたって、産業サイバーセキュリティセンター中核人材育成プログラムの講師であられる、門林雄基先生、満永拓邦先生、登大遊先生、小林裕士先生には本書の元となるセキュア開発プロジェクトのメンター・講師として、ご指導・ご助言・ご支援を賜りました。改めて御礼申し上げます。

また，有識者として OWASP Japan　株式会社アスタリスク・リサーチ 代表取締役 岡田良太郎様にも有益なご助言、査読、巻末コメントをいただきました。改めて御礼申し上げます。

そして、本書の作成や本プロジェクトをともに実施した、下記メンバーの皆様にも感謝申し上げます。

＜システム開発のセキュリティ向上プロジェクト＞

リーダー　篠原　隆

小林　慧吾

メンバー　大沼田　俊一

浮田　裕基

横井　雄亮