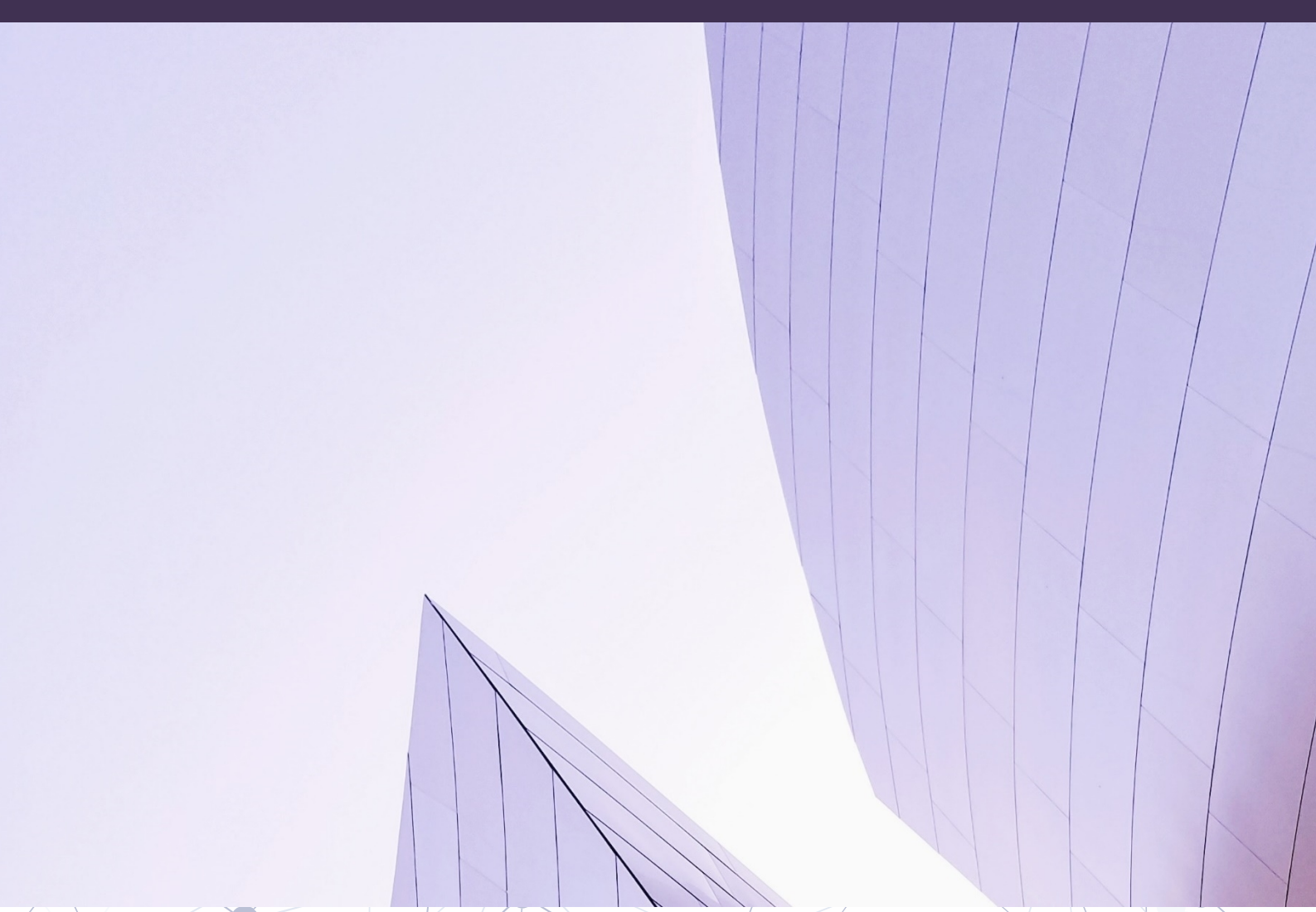


セキュリティの勘所

～上流設計編～

Secure SketCH Books VOL.6

Secure SketCH

Secure SketCH Books VOL.6

セキュリティの勘所

～上流設計編～

日々高度化するサイバー攻撃に対し、システムに最適なセキュリティ対策を施すことは難しい。完璧なセキュリティ品質を追い求めればコストが度外視され、リリース間近でのセキュリティ要件追加や脆弱性修正は開発遅延に繋がる。こういった事態を避けるために、上流工程からのメリハリをつけた設計、即ちセキュリティ上流設計が有効となる。

Topic

1．認識されつつある、上流設計の重要性

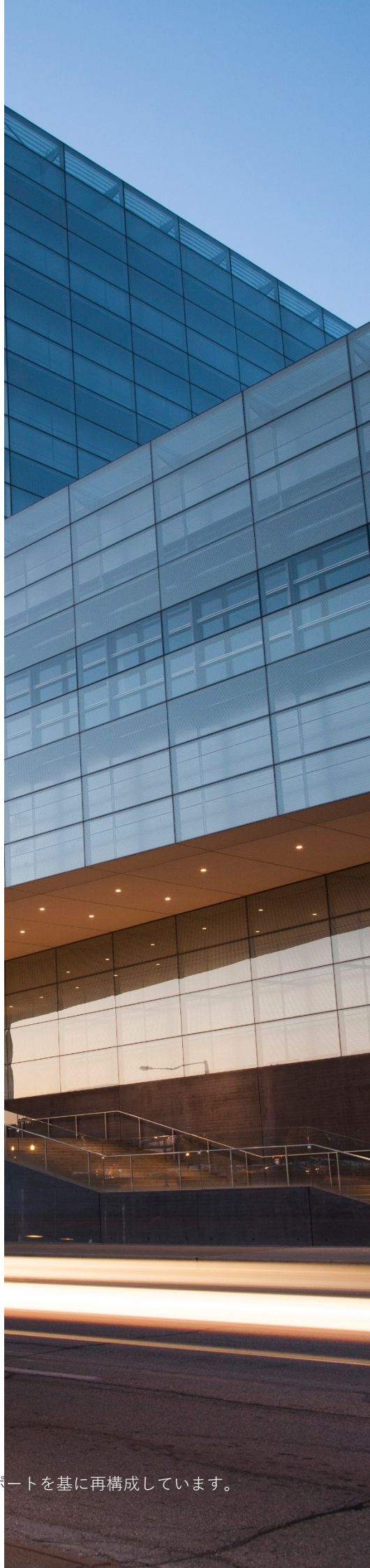
2．上流設計での十分な検討がQCDの最適解に

　　2-1．企画・構想

　　2-2．要件定義

　　2-3．基本設計

3．チェックリストを活用して抜け漏れなく検討・チェック



※本記事は2018 年 2 月に発行したNRIセキュア ニュースレターで掲載されたレポートを基に再構成しています。

1．認識されつつある、上流設計の重要性

システム開発において「上流設計」とは、システム開発の早い段階で実施される、自由度の高い一連の設計活動のことである。設計とは「ある目的を達成する要件や手段を具体化し、可視化する作業」の総称であり、実装に移る前、すなわちシステムの企画から詳細設計までの間に、いかに顧客要求や業務要件を適切な粒度で言語化できるかがポイントとなる。

設計のうち企画・構想、要件定義、基本設計を特に上流設計とよぶ。（図1）一般的にシステム開発のライフサイクルコストは工程とともに推移し（図2）、上流設計終了時点でその66%が決まるといわれる[*1]。これは、詳細設計以降での仕様変更が大きな 手戻り（バックトラック）を生じ、コストとスケジュールの両面で影響を与えることを意味する。従って、後続工程での手戻りを減少させるべく上流工程を戦略的・効率的に実施し、可能な限り要件を固めることがシステム開発の成功の鍵であるといえる。

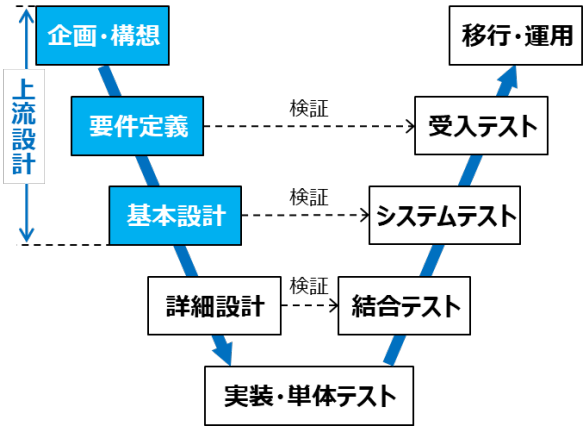


図1 システム開発の V 字モデル

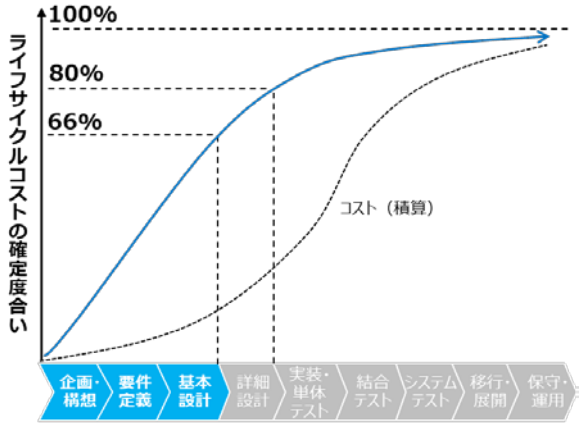


図2 コスト実績とコスト確定度合いの推移

特に、セキュリティ要件の定義や設計においてはこの点をより強く意識する必要がある。なぜなら、従来のシステムは可用性を重視して設計されてきたこと、セキュリティは非機能要件として見られてきた（詳細後述）ことなどにより、上流工程でのセキュリティ設計に関 する標準的な取組みがこれまで確立されていないからである。

*1 Fabrycky, W. J. ; Blanchard, B. S. Life-Cycle Cost and Economic Analysis. 1991.

セキュリティ上流設計を軽んじたことによる開発の失敗事例

ある広告企業 A 社で、新サービス立ち上げのため Web システムを開発することになった。当該サービスは大量の個人情報と金銭処理を扱うため、厳格なセキュリティ要件が必須となる。

しかし、他社に遅れをとらないようにと企画・構想工程でのシステムの目的や安全性の検討が不足した状態で設計工程へ進み、急いで実装を完了させた。テスト工程に入り、設計にないセキュリティ不備がいくつも見つかったが構わず開発を前へ進めた。

しかし、リリース間近の脆弱性診断によって重大な脆弱性が大量に見つかり、プログラム構造レベルの見直しが必要になった。さらに悪いことに、当初の構成にしがみついて小手先の改修をかけたために処理やデータの不整合が次々に生じた。

結果、多大な工数・時間の投入に反してリリース時期が大幅に遅れ、システムはその目的を達成することができなかった。

脆弱性を作りこまないセキュアコーディングやリリース前に検出する脆弱性診断は年々一般的になっているが、これらは詳細設計以降がメインとなる。また、テスト工程で検出された脆弱性を修正する工数・期間が上流工程で考慮されていないことも多い（セキュリティ以外の機能の場合、不具合対応はプロジェクトスケジュールに含まれるが、脆弱性修正、すなわちセキュリティ機能の場合は含まれておらず追加工数・期間で対応せざるを得ないことが多々ある）。そのため、詳細設計以降リリース前までに生じた手戻りや、リリース後の脆弱性対応によってシステム開発・運用の採算がとれなくなるケースも散見される。

左の事例からもわかるように、近年セキュリティ目線での開発最適化と従来のシステム開発プロセスへの組み込みが各企業で課題と認識されており、そのために上流設計で要件を漏れなく捉えて具体化する標準の設計フレームワークや品質管理手法を確立する必要がある。

2．上流設計での十分な検討がQCDの最適解に

昨今のビジネスは時間との勝負であり、顧客要求を実現するサービスをいかに早くリリースできるかが競争優位の鍵を握る。また、システムの複雑化に伴い詳細設計以降の仕様変更コストが何倍にも跳ね上がる。これらをふまえ上流設計に重点を置いたシステム開発プロセスを、従来のものと対比する（図3）。

ここで重要なことは、顧客要求（Q）実現に必要な開発期間（D）を短縮するためにある程度の工数負荷（C）を容認している点である。

しかし、上述のとおりセキュリティ上流設計は標準化が遅れている（＝担当者の知見・能力に依存する）分野でもある。ゆえに、セキュリティを組み込んだシステム開発を効率的に行うために、各工程で何をすべきか、具体的な指針、仕組み、手法が模索されている。

実際、筆者が過去に対応した多くの顧客において、通常のシステム開発における設計基準やチェック項目は標準化されていても、セキュリティに関しては明確なものはなかった。

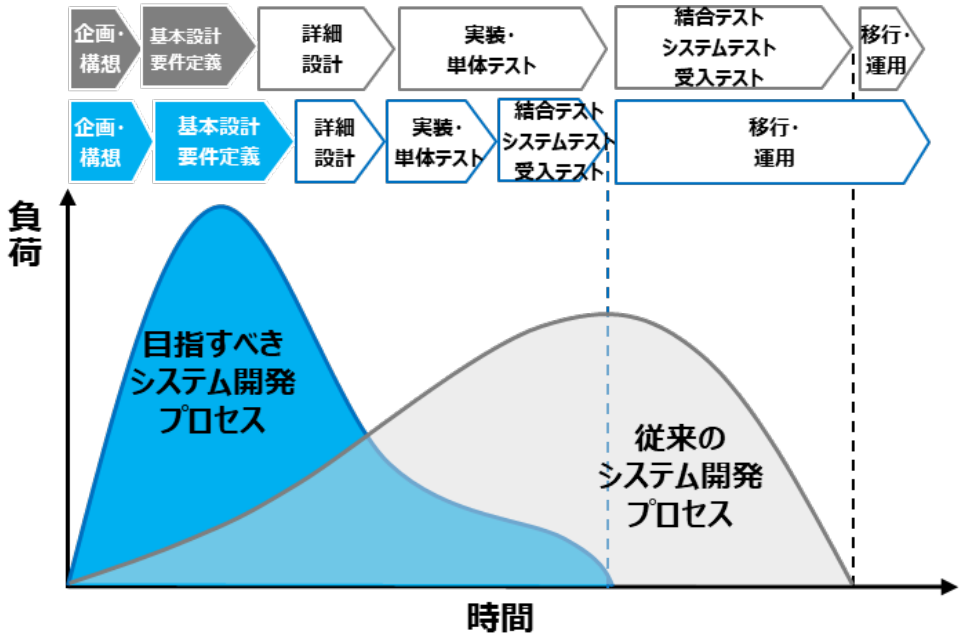


図3 上流設計に注力した開発プロセス

次章からは、戦略的開発のためにセキュリティ要件を具体化する上流設計のフレームワークとツールを紹介する。それに則り、**企画・構想**、**要件定義**、**基本設計**の各工程において、論理的に要件を解析し、重要度をにらんで絞込み、対応方針を立てる。

2-1．企画・構想
システムの目的と守るべき情報産を理解し、リスクへの対応方針を設定する

企画･構想工程は大きく二つに分かれる。最初に、システムの目的・用途、利用者、アクセス経路と、システムが保有する「守るべき情報資産」を整理し、以下の 4つの観点で定義した重要度（図4）に基づいて、システムが目指すべきセキュリティ水準を設定する。

① 情報資産の重要度・量

システムが保有する情報資産の重要度を影響度（漏洩・改竄・破壊が起きた場合の被害範囲と深刻さ）と量から定義する。特に顧客の個人情報を大量に処理するシステムは、被害時の影響度が非常に大きく、一般的に「重要システム」と位置づけられることが多い。

② システム・情報資産の場所

システムの設置場所、情報資産の保管場所を、インターネットからの接続可否、アクセス可能なユーザの拠点・属性などで分類する。特にインターネットから接続可能な場合、外部から不正アクセスされやすいため、リスクの洗い出しを入念に行う必要がある。

③ ユーザの種別・数

システムを利用するユーザの属性と規模を整理する。システム停止やデータ改竄・漏洩が起こった際に、当該システムが提供するサービスに関する影響（顧客のレピュテーション低下、パフォーマンス低下、営業担当者の機会損失など）をもとに損害を推定する。

④ 他システムへの影響

連携先のシステムに与える影響を整理する。システム停止やデータ改竄が起こった際に、当該システムのデータの入出力が連携しているシステムの動作やデータに与える影響（重要処理の停止、データの不整合など）が生じる範囲と、深刻度合いを推定する。

No.	システム名	主管部	システムの重要度	システム・情報資産の場所	重要資産の重要度・量	ユーザの種別・数	他システムへの影響	影響度	発生確率
1	Aシステム	A事業部	S	DMZ上 (公開Web)	ランク1 (顧客データ) 約800万ユーザ	外部顧客(約600万ユーザ) グループ社員(3000ユーザ)	影響なし	大	大
2	Bシステム	B事業部	A	**DMZ上 (公開Web)**	ランク2(取引先データ) 約200社の取引先	取引先(約200ユーザ) グループ社員(約50ユーザ)	影響あり (取引先のシステム)	中	大
3	Cシステム	C事業部	S	イントラネット内 (非公開)	ランク1(顧客データ) 約1,500万ユーザ	グループ社員(3000ユーザ)	影響あり (顧客情報を利用する20システム)	大	中

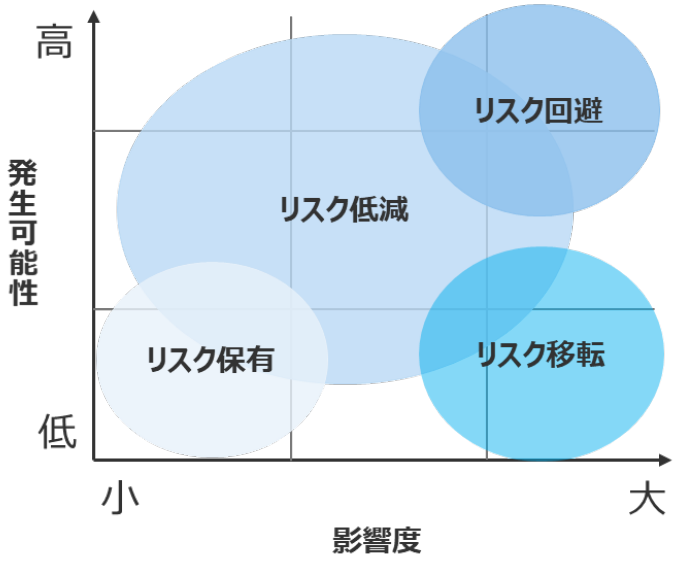
図4 システムの重要度定義一覧

次に、システムと情報資産に想定されるリスクを脅威・脆弱性の両面から洗い出し、リスクの発生確率、顕在化した際の影響度などを可視化する。

個々のリスクは情報資産の重要度やシステム特性を踏まえ評価・分析し、費用対効果を考慮して対応方針を設定する*2。

例えば、優先度の高いリスクは回避・低減するための対策を要件として抽出して設計書に盛り込み、優先度の低いリスクは、保有・転嫁するために実装すべき要件から除外する。

設定した「リスク対応方針」を、要件義以降セキュリティ要件として具体化していく。



リスク回避： 脅威の要因を停止もしくは別の方法を採用し、脅威の発生をなくす

リスク低減： 脆弱性への情報セキュリティ対策を講じ、リスク顕在化の可能性を下げる

リスク保有： リスクを低減するセキュリティ対策を行わず、許容範囲内として受容する

リスク移転： リスク負担を第三者に移し、リスク顕在化時に自身が被る影響をなくす

図5 リスク対応方針

*2 NIST Cybersecurity Framework 1.1（https://www.nist.gov/cyberframework）IPA リスクへの対応（概念図）（https://www.ipa.go.jp/security/manager/protect/pdca/risk.html）

2-2．要件定義
セキュリティ要件を対策領域と対策ベクトルで分解する

要件定義工程では、中期計画やセキュリティ方針を参照しリスク対応方針に沿って対策の「広さ」と「方向」でセキュリティ要件を分解する。その前に、2つの原則を紹介する。

○原則１「セキュリティ要件は機能・非機能の両面をもつ」

セキュリティは非機能要件と解釈されることが多いが、その実現に必要な機能要件としても捉えるべきで、特に開発者担当者とのコミュニケーションに注意が必要である。

例えば正規ユーザのみをシステムに接続させる「認証」は設計・実装が必要な一つの機能だが、開発はベンダ任せとなり認証の要件や機構に不備があって手戻りを招いたケースもある。

○原則２「技術対策要件はセキュリティ要件の一部である」

セキュリティ対策は、技術対策だけでなく組織・規程、リスク管理、法令遵守、物理などの対策と共に全体を捉えるべきである[*3]。例えば、入退室管理が徹底されていれば内部不正リスクが低くなるため、ユーザアクセス制御は業務を阻害しない程度に留められる。

また、CSIRT が整備された企業ではサイバー攻撃の検知・対応力が相対的に高いため、技術対策では防御に徹する選択もある。

本稿は、単純のため技術対策のみを想定する。システムの設置場所、情報資産の保管場所を、インターネットからの接続可否、アクセス可能なユーザの拠点・属性などで分類する。特にインターネットから接続可能な場合、外部から不正アクセスされやすいため、リスクの洗い出しを入念に行う必要がある。

*3 NRI Secure Framework（https://www.nri-secure.co.jp/service/consulting/security_visualization.html）

■対策の「広さ」：対策すべき技術領域を網羅的に把握

要件を、技術対策領域（ユーザから情報資産の流れ（識別→認証→認可→保護→複製）に 沿った基本領域と、基本機能を補完する共通領域）に展開する。これらはバリューチェーンに倣って「情報資産（事業で創出された価値）に対するリスクへの技術対策を網羅的につなぎ合わせたもの」と表現できる（図6）。

図6 セキュリティ技術対策領域

① アカウント管理

適切なユーザのみがシステムに利用登録されるように、アカウントの申請・発行・削除、ポリシー制御、棚卸、モニタリングなどを行う。例えば、高権限を有する、または長時間ログインしたままのアカウントは内外からの不正なアクセスに利用されやすいため、一定期間で利用停止・削除や強制ログアウトさせる機能を持たせる。

ブルートフォース攻撃などで突破されにくい強固なパスワードポリシー設定も有効である。

② 認証

正規ユーザのみをシステムにログインさせるため、認証アカウントの登録・管理、認証方式の指定、ログイン管理、モニタリングなどを行う。例えば、認証メカニズムが脆弱な場合はアカウント管理を徹底しても盗聴やパスワード推測で不正ログインを許してしまうため、強い認証アルゴリズムや認証基盤を採用する。リモートログインや重要システムへのアクセスでは、生体認証や端末認証を併用した多要素認証も検討対象となる。

③ アクセス制御

認証されたユーザを必要なデータのみにアクセスさせるため、境界防御、役割・権限割当て、重要データの分離や棚卸、モニタリングなどを行う。例えば、重要情報の参照・更新を全ユーザに許可していると、認証突破された場合の不正アクセス被害が拡大するため、最小権限の原則に従って操作権限をデータベースや OS で制限する。また、ファイアウォールなどネットワーク機器による接続先／元の制限も重要である。

④ データ保護

非正規のユーザまたは権限によりデータにアクセスされた場合に備え、データ・ファイルシステム、通信経路の暗号化、保護対象の棚卸、モニタリングなどを行う。例えば、認証やアクセス制御が突破された場合に攻撃者が目的とする情報が暗号化されていれば解読に時間がかかり、その間に検知・対応ができる。また、特権アカウントへのなりすましを防ぐためにパスワードファイルの暗号化やハッシュ化も有効である。

⑤ バックアップ

システムの、機密性・完全性・可用性が阻害された場合にデータおよびシステムを元の状態へ復旧させるため、バックアップの計画、取得・保管、復元、システム復旧などを行う。例えば、攻撃者によりデータの破壊や改竄などを受けた場合にデータが消失しかねないため、平時から計画的なバックアップを実施して世代管理を行う。

さらに、定期的な復旧テストの計画・実施や保管場所についての配慮も必要である。

⑥ 標準構成

システムを構成するコンポーネント群を、標準化された健全な状態に保ち有効に機能させるために、システムの標準設定、システムの構成管理、変更管理、棚卸、モニタリングなどを行う。例えば、場当たり的に追加機能が実装されるとデグレードや不整合が生じやすくなるため、プログラムや構成のバージョンを管理する仕組みを導入する。OS 導入時に脆弱性を組み込まない標準イメージを使ったキッティングも有効である。

⑦ マルウェア対策

悪意あるソフトウェアの侵入・感染拡大からシステムを守るため、マルウェア情報収集と対策ツール管理、アプリケーション実行制御などを行う。例えば、悪意あるサイトから端末にマルウェアがダウンロードされることが想定される場合、ファイル拡張子によるアプリケーション実行制御や、感染時のネットワーク遮断が有効となる。また、ウィルス対策ソフトのパターンファイルの検証と定期的な更新も必要である。

⑧ 脆弱性管理

システム構築過程で脆弱性を作りこまない／見逃さないように、脆弱性情報収集と評価、修正プログラム適用、セキュアコーディング、脆弱性診断、モニタリングなどを行う。例えば、データベースを呼び出すロジックに不備があるとSQL インジェクションなどの危険があるが、入出力制御徹底や標準ライブラリ使用で防ぐことも可能だ。脆弱性を検証するためにアプリケーションやプラットフォームの診断も有効である。

⑨ ログ管理

平時はシステムやユーザの挙動を正確に把握し、有事は問題へ迅速に対処するため、ログの時刻同期や見読性向上、保管・保護、モニタリング、分析、監査などを行う。例えば、不正アクセスで情報漏洩や改竄が生じた場合も、監査ログを取得していれば攻撃者の行動・操作履歴をトレースして被害の拡大防止や犯人の追跡に役立つ。また、ログの集中管理やSIEM などログ分析ソリューションとの連携も視野に入れる。

■対策の「方向」：対策ベクトルごとに必要な機能を考える

技術対策領域ごとの要件を、時系列に沿って**対策ベクトル**（Cybersecurity Framework[*4]のFunctionに相当）に分解する。順に**「特定(Identify)」**：守るべき情報資産を把握する、**「防御(Protect)」**：サイバー攻撃を事前に予防する、**「検知(Detect)」**：防御を突破された際に事象を検知する、**「対応(Respond)」**：緊急対応を施し、被害の極小化を行う、**「復旧(Recovery)」**：システム・サービスを平時の状態に戻す、となる。

例えば、「アカウント管理」の「特定」では「正規利用者のみ登録可能なワークフローや申請フォームを採用する」、「防御」では「強固なパスワードルールを AD のポリシーで強制適用する」などになる。また、「標準構成」の「検知」では「構成情報の改竄有無を日次チェックする」、「対応」では「改竄検知した構成の不整合箇所を即時特定し設定変更できる」「復旧」では「指定した世代へ構成情報を復元できる」などが考えられる。

このように、業務やシステムの「あるべき姿」をもとにシステム要件を漏れなく定義し「セキュリティ要件定義書」として言語化する（または通常の要件定義書に組み込む）。

追加検討すべきセキュリティ要件も含め、システムの所管部門や IT 企画部門、設計担当者と協議のうえ、定義された要件の妥当性を確認して工程を完了する。

*4 NIST Cybersecurity Framework 1.1（https://www.nist.gov/cyberframework）

ここから基本設計工程に移る。前章で「広さ」「方向」で分解して定義したセキュリティのシステム要件を、「深さ」の軸、すなわちシステムの各レイヤーに展開していく。

■対策の深さ：対策を実装すべきレイヤーを決定する

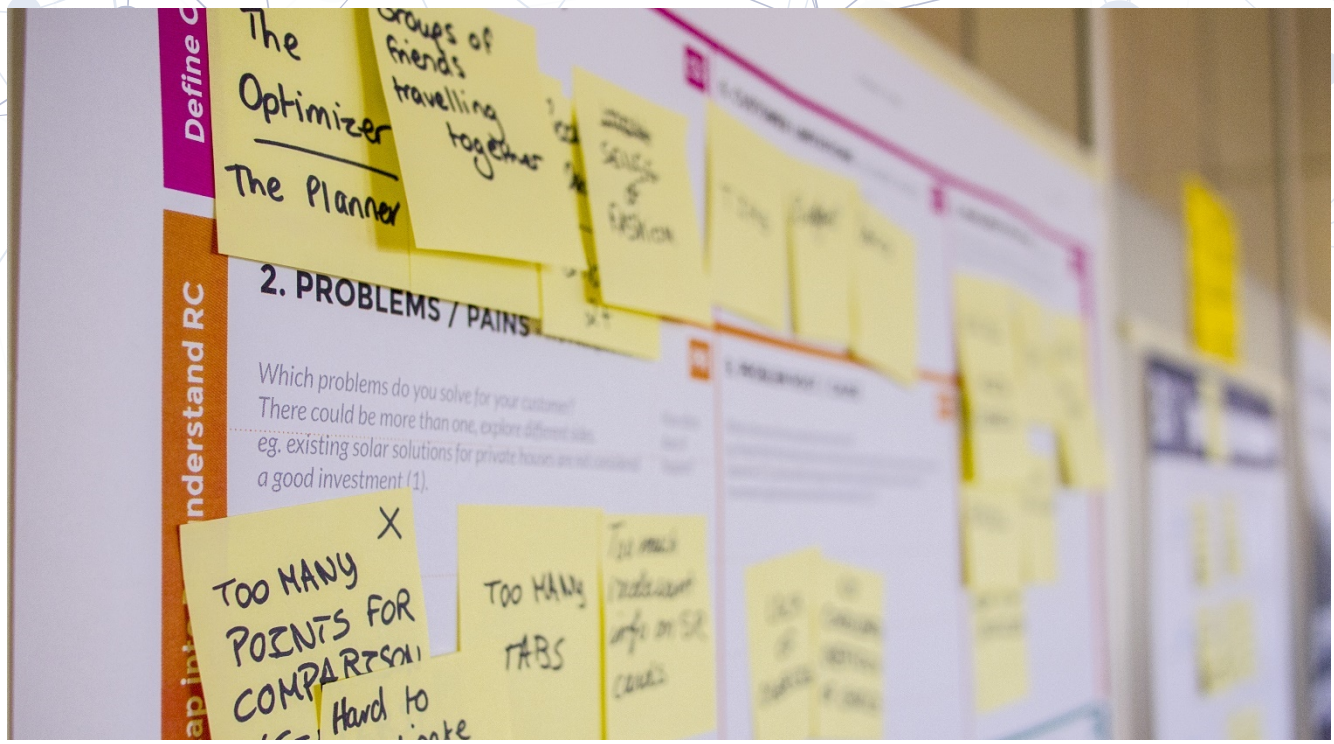
システムの各レイヤー（アプリケーション、ミドルウェア、OS、サーバ、ネットワーク）でどのセキュリティ要件を実装するか、レイヤー横断でどんな機能を持たせるかを考える。

そして、開発コスト・工数・期間や各レイヤーで想定される脅威、システム外での対策状況などを考慮したうえ、選択と集中の観点で各レイヤーの要件を決定する。

あらゆる対策を全レイヤーで実装すると対策の重複や煩雑化を生み、実装・運用コストの肥大や性能劣化に繋がるため、各レイヤーに必要最小限の要件を盛り込むべきである。

例えば、「データセンター内の環境にシステムを構築する場合は、強固な共通システム基盤が信頼できるため OS 以上のレイヤーの対策に集中する」「クラウド基盤上で OS までのセキュリティ対策をベンダが保証する場合は、ミドルウェアとアプリケーションのセキュリティ対策に専念する」「データベースのログ監査機能を使える人材がいない場合、監査ログは OS の機能で取得する」という方針が考えられる。

以上の検討結果を「セキュリティ基本設計書」として言語化する（または通常の基本設計書に組み込む）。基本設計で実装方針が決定しなかった要件や検討課題は、引継ぎ事項として一覧化して詳細設計以降へ引き継ぎ、所管部門の承認のもと工程を完了する。



3．チェックリストを活用して抜け漏れなく検討・チェック

最後に、企画・構想、要件定義、基本設計に亘りセキュリティ要件を漏れなく具体化・可視化して設計書に落とし込むのに有効なチェックリストと、その使い方を紹介する。

チェックリストは上流工程だけでなく、詳細設計のレビューや、各種テストでも有効だ。まず、要件定義工程では対策領域・対策ベクトルごとの「要件カテゴリ」を定義する（図7上）。そして、基本設計工程では、各要件カテゴリを各レイヤーごとに読み替え（図7下）、要件カテゴリごとに基本設計で定義すべき項目（機能要件・非機能要件）を洗い出して一覧化する。

このとき、国内外の専門書籍、最新の脆弱性・脅威情報、ガイドラインが参考になる。

これらの情報の取捨選択・加工と文書化も、セキュリティ担当者の腕の見せ所である。

要件定義		特定	防御	検知	対応	復旧
基本機能	アカウント管理	アカウント管理対象の特定	アカウント統制	アカウント監視	アカウント停止	アカウント復旧
	認証	認証対象の特定	認証	認証監視	認証の停止	認証の復旧
	アクセス制御	アクセス制御対象の特定	アクセス制御	アクセス監視	アクセスの遮断	アクセスの復旧
	データ保護	データ保護対象の特定	データ保護	データ改ざん・漏洩範囲の調査	データ改ざん・漏洩対処	データの復旧
	バックアップ	バックアップ対象の特定	バックアップ取得	バックアップ監視	バックアップ復元	システムの復旧
共通機能	標準構成	標準構成の特定	構成管理	構成監視	構成復元	標準構成の復旧
	脆弱性管理	脆弱性情報の特定	脆弱性管理	脆弱性診断・監視	脆弱性暫定対処	脆弱性監査・報告
	マルウエア対策	マルウエア情報の特定	マルウエア遮断	マルウエア監視	マルウエア駆除・封じ込め	マルウエア感染対象の復旧
	ログ管理	ログ管理対象の特定	ログ取得・保管	ログ監視	ログ調査・分析	ログ監査・報告

基本設計（アプリ）		特定	防御	検知	対応	復旧
基本機能	アカウント管理	アカウントの申請・登録	アカウントポリシー統制	アカウント棚卸・利用監視	アカウントの無効化	アカウントの有効化
	認証	認証方式・対象の定義	ID・端末・生体による認証	ログイン監視	ログインの無効化	ログインの有効化
	アクセス制御	役割の定義	役割によるアクセス制御	アクセス状況の監視	アクセス権限の無効化	アクセス権限の有効化
	データ保護	暗号化方式の定義	通信・処理の暗号化	—	—	—
	バックアップ	—	—	—	—	—
共通機能	標準構成	プログラム構成の把握	プログラム構成管理	プログラム構成監視	モジュールの復元	プログラム構成の復旧
	脆弱性管理	プログラムの脆弱性の把握	セキュアコーディング	アプリケーション診断	プログラム修正・パッチ適用	バージョンアップ
	マルウエア対策	—	—	—	—	—
	ログ管理	ログ出力要件の定義	アプリケーションログ出力	—	—	—

図7 セキュリティ上流設計フレームワーク

次に、設計項目毎に「工程」「設定値」「リスク」「優先度」「代替策」を検討する。

これらは後続工程における要件の微調整や変更管理の判断材料となり、システムのセキュリティ品質、コスト、期間、開発者のスキルなど制約条件の変更にも円滑に対応できる。

工程：「アクセス権限を付与するユーザの範囲は企画・構想工程で、アカウントの管理ポリシーは要件定義工程で、アカウントのモニタリング方法は基本設計工程で」のように、どの 工程で要件を検討して、各工程でのドキュメントに反映するかを記載する。

設定値：「アカウントの有効期限は 90 日、パスワードは英数字と記号を含む 8 文字以上」といった設定推奨値を記載する。PCI DSS5での具体的設定要件などが活用できる。

リスク：「開発時に標準ライブラリを用いない場合、ユーザ入力値のエスケープ処理に SQL インジェクションの脆弱性が残存する」など、実装しない場合の影響を記載する。

優先度：リスクや中長期計画や予算、納期、要員の工数・スキルなどを勘案して必須、推奨、任意の要件を色分けして実装方針を決める。「短期と長期：リリース前に実装すべきか、稼動後に追加すべきか」「平時と有事：定常運用で必須か、緊急時のみ必要か」などの観点で取捨選択するとよい。例えば、重要システムだが納期を優先して必須要件の実装に留め、推奨・任意の要件は稼動後のエンハンスの際に要件追加の是非を検討する。

代替策：設計項目を実現しない場合に要件を担保する別の方法を指す。ここでは「方式と運用：要件をシステムに実装するか、運用でカバーするか」を考える。例えば、システムに棚卸機能は実装せず、運用担当者が週次で不要なアカウント・権限を目視確認する。

以上を表形式で一覧化し、チェックリストとして機能させるため「判定」「備考」の欄を追加する。「判定」には OK、NG、N/A を記載し、その根拠を「備考」に記録する。

これによって後続工程やリリース後にどんな対応が必要かが明確になり、詳細設計工程、および各設計工程に対応するテスト工程（図 1）において、上流工程で言語化した要件のトレーサビリティを確保することができるようになる。

おわりに

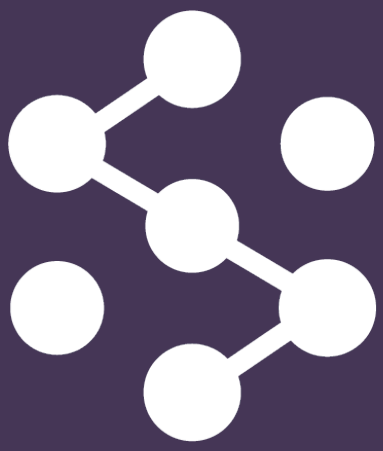
セキュリティ上流設計とは、セキュリティ要件のルーツをひとつひとつ紐解きながらシステムとして紡ぎあげる活動である。

経営・ビジネスの方針から守るべきものを明確にし、様々な制約条件下で要件を段階的に具体化し、運用を見据えて基本設計まで落とし込む。 こうして企業が目指すセキュリティ水準を達成し、経営・IT 戦略の実現に寄与することがセキュリティ上流設計の目的であり、根底にある *“Security by Design”* の体現といえる。

筆者略歴

岡部 拓也

外資系IT企業とコンサルティングファームで延べ9年システムの構想策定～設計・導入～運用、ITガバナンス、プロジェクト管理、リスクマネジメント等に従事。2016年に野村総合研究所入社後、NRIセキュアテクノロジーズへ出向し、中長期計画策定、セキュリティ上流設計、セキュリティリスク評価、IT組織改革等、数多くのコンサルティング案件に従事している。



Secure SketCH

セキュリティ経営をシンプルに

https://www.secure-sketch.com