社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－DPS－138 (41)
2009－CSEC－ 44 (41)
2009／3／6

セキュアなアプリケーション開発のための要求・デザインパターンの提案

大久保 隆夫 †‡　　田中 英彦 ‡

† 株式会社 富士通研究所
211-8588 川崎市中原区上小田中 4-1-1
okubo@jp.fujitsu.com

‡ 情報セキュリティ大学院大学
221-0835 横浜市神奈川区鶴屋町 2-14-1
tanaka@iisec.ac.jp

あらまし セキュアなアプリケーション開発のためには，開発の最上流からセキュリティを考慮した開発が必要になるが，セキュリティ知識不足やセキュリティ分析，設計作業が重いことなどが原因で，適切に行われず，脆弱性を生む一因となっている．筆者らは，最小限のセキュリティ知識でも効率的なセキュアな開発を可能にする手法として，アスペクト指向の概念に基づいた開発手法を研究している．本稿では，要求分析および設計段階において，ミスユースケースを拡張した記法を用いてセキュリティ要求が，設計時にセキュリティを挿入する箇所をパターン化したセキュリティ設計がそれぞれパターン化可能であり，それらによってセキュリティ作業の省力化，再利用が実現できることを示す．

A Proposal of requirement patterns and design patterns of secure application development

Takao Okubo†　　Hidehiko Tanaka‡

†Fujitsu Laboratories limited
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki 211-8588, Japan
okubo@jp.fujitsu.com
‡Institute of Information Security
2-14-1, Tsuruyacho, Kanagawa-ku, Yokohama 221-0835, Japan
tanaka@iisec.ac.jp

Abstract Security consideration in early development stages is important for secure application software development. However, lack of security knowledge and heavy security tasks make it difficult that is one of the reasons of software vulnerabilities. In this paper, we propose two kinds of security pattern: secuirty requirement patterns and security design patterns. We present that these pattern make security design laborsaving and reusable.

1 はじめに

本稿では，脆弱性のないアプリケーションを開発する手法として，要求分析で用いるセキュリティ要求パターン，設計で用いるセキュリティ設計パターンを提案する．要求パターン，設計パターンはそれぞれ要求分析，および設計それぞれについてその過程をパターン化し，典型的な分析，設計を行う場合にパターンを適用することによりセキュリティ分析，設計作業の手間を省くことを可能にするものである．これらのパターンは，筆者らが提案するアスペクト指向の概念に基づいた開発手法を支援する技術であり，パターンを適用することで必要なセキュリ

ティ知識の最小化も実現している．本稿では要求，設計それぞれのパターンを示し，その実効性を評価する．

2 背景と従来技術

アプリケーションの開発において，脆弱性の排除は重要な問題と認識されているにも関わらず，従来技術ではその排除は有効に機能しているとはいえない．セキュアなアプリケーション開発のために重要なのは，開発の最上流である要求分析の段階からセキュリティ要求を明確に定義し，その要求に沿った設計，開発を行うことである．しかし，筆者らのセキュリティ開発支援の経験では，セキュリティ要求定義が行われず，要求の存在しないまま開発を行っている場合がしばしばみられた．その主な理由は次に挙げるものであった．

(1) 要求分析においてセキュリティが重要視されていない

(2) セキュリティ要求分析のためのノウハウを持つ開発者がいない

(3) 要求分析のために用意された時間が短く，セキュリティ要求分析を行う時間が確保できない

上記のうち，(1) に関しては開発に関わるステークホルダ (利害関係者) のセキュリティに対する意識や知識不足，(2)(3) についても，開発体制の中でセキュリティの開発に必要な知識を持つ人材が不足していることが原因と考えられる．また，(3) については，セキュリティ要求分析に必要な脅威分析などの作業が，大きな工数を要するため，従来の要求分析のために与えられた時間や体制では分析の完遂が困難になってしまっていることが挙げられる．

アプリケーションのセキュリティ要求分析では，対象とするアプリケーションにどのような脅威があるかをまず識別する必要がある (脅威分析)．脅威分析の従来技術としては，脅威モデリング [1] がある．脅威モデリングは，アプリケーションのコンポーネント間のデータの流れを分析し，それを基に脅威を抽出する手法である．脅威モデリングは，アプリケーションのアーキテクチャが明らかになっているほど効果が高いため，既存ソフトウェアの分析に用いられる．しかし，一つ一つの細かいデータの流れを詳細に把握した上で分析を行うため，大きな工数が必要となる．また，逆にアーキテクチャ情報が詳細化されていない要求分析段階ではあまり有効な分析が行えない．

筆者らは，セキュリティ有識者が少数である現状の体制においても，効率的なセキュリティ開発を可能にする手法として，アスペクト指向開発の概念を導入したフレームワーク「Security Injector」の研究を行っている．次節では，「Security Injector」の要求分析段階，設計段階において，セキュリティ要求分析および設計の過程をパターン化する試みについて述べる．

3 アスペクト指向の概念に基づくセキュアな開発の効率化

アスペクト指向開発 (Aspect-oriented software development： AOSD) は，関心事を分離し，関心事ごとに開発を行う開発手法である [2]．また，AOSD と関連の深い概念として，依存性注入 (Dependency Injection: DI) がある [?]．DI は，通常と依存性を逆にすることにより関心事間のモジュール性，再利用性を高めるという概念である． 筆者らは，AOSD と DI の概念をセキュリティ開発に活用することで，2 節で挙げた知識不足の問題点を解決できるのではないかと考えた．AOSD，DI の導入により，次の効果が期待される．非セキュリティ関心事の開発はセキュリティ関心事とは分離され (AOSD)，かつ非セキュリティからセキュリティへの依存関係は排除されるため，非セキュリティ関心事の開発においてセキュリティ知識の必要性が最小化され，一般の開発者が担当することができるようになる．一方セキュリティ関心事はセキュリティ有識者が開発し，DI の仕組みにより，非セキュリティ関心事中の適切な箇所に挿入されるようになることが期待される．

従来 AOSD のセキュリティへの適用は，実

装工程のみへの適用など，限定的に行われてきており，開発工程全体によるセキュアなAOSDの手法は存在しない．そこで筆者らは新規に開発工程全体を通してAOSDとDIの概念を導入したフレームワーク「Secureity Injector」の体系化を目標にした研究を行っている．Security Injectorの構成を図1に示す．

Security Injectorのねらいは，要求分析段階でセキュリティ関心事とその他の関心事が分離され，以降の工程ではそれぞれの関心事ごとに開発を行わせることによって，セキュリティ知識不足の問題が解決されることである．

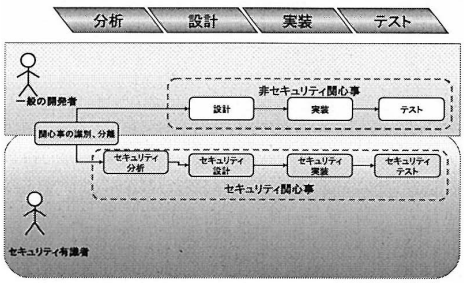

図 1: Security Injectorの構成

Security Injectorは必要なセキュリティ知識の限定には有効であることが期待されるが，別の問題であるセキュリティ作業の効率化は，上記の構成だけでは達成できないため，各工程においてセキュリティ作業を効率化する技術が別途必要になる．本稿で提案するセキュリティ要求パターン，セキュリティ設計パターンは，これらの作業の効率化を目的とする要素技術である．

3.1 セキュリティ要求のパターン化

3.1.1 拡張ミスユースケース記法 AsseMis

筆者らは，ミスユースケース記法[3]を拡張し，資産の概念を導入することで脅威抽出を容易に拡張した，資産ベースミスユースケース(Asset-based Misuse case:AsseMis)記法を提案している[4][5]．AsseMisの記述例を図2に示す．

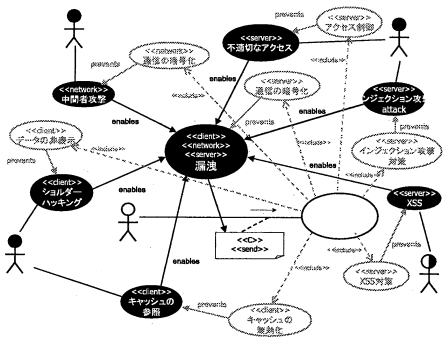

図 2: AsseMisの例

AsseMisの特徴は，ユースケースとアクタの間にデータ資産とその流れの表記の追加，およびセキュリティプロパティの追加を行うことにより，保護資産を明確にし，セキュリティ有識者による脅威抽出を容易にしたことである．また，AsseMisでは大まかなアーキテクチャ情報を追加することにより，要求分析において可能な粒度での脅威の特定を可能にしている．

3.2 Webセキュリティ要求パターン

セキュリティ要求は一般にアプリケーションのアーキテクチャや資産により異なるため，通常はAsseMisにおいてもアプリケーションごとに次のような手順で分析を行うことになる．

(1) ユースケース図の作図 (一般の開発者)

(2) 資産情報の付与 (共同作業)

(3) 脅威の抽出 (セキュリティ有識者)

(4) 脅威の評価 (セキュリティ有識者)

(5) 対策の抽出 (セキュリティ有識者)

しかし，ブラウザをクライアントとし，httpでサーバと通信を行うWebアプリケーションのような限定されたドメインでは，アプリケーションに共通する次のような特徴が存在する．

- アーキテクチャ構成はブラウザ，ネットワーク，サーバ

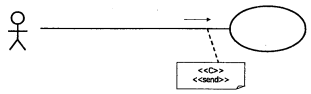

図 3: 「機密データの送信」パターン：Context

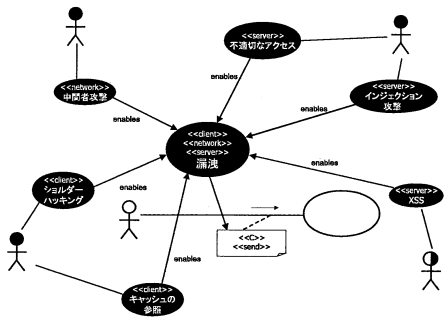

図 4: 「機密データの送信」パターン：Problem

- Web アプリケーションに特有の典型的な攻撃が存在する
- 上記の典型的な攻撃は，決まったデータフロー，ユースケースの形態の場合に脅威となる

例えば，インジェクション攻撃は，データをクライアントからサーバに送信するデータを改竄することにより発生するため，開発においてそれを防止する対策が必要であることが分かっている．この特徴を利用すると，なんらかのデータ資産を (アクタからユースケースに) 送る Web アプリケーションでは，インジェクション攻撃によるデータの改竄や漏洩の危険が存在し，対策を行う必要が常に存在することになる．筆者らはこの特徴を利用し，資産の形態 (データの流れ，セキュリティプロパティ) ごとに起こりうる脅威，対策をパターン化し，8 つ (機能の悪用，機密データの送信，要完全性データの送信，共有データの受信，プライバシー情報の受信，要完全性データの受信，ID，パスワード認証，セッション管理) の Web セキュリティ要求パターンを作成した．

要求パターンの例 (「機密データの送信」パターン) を図 3，4，2 に示す[1]．パターンは Context(図 3),Problem(図 4),Solution(図 2) で構成される．Context は，その要求パターンを適用すべき状況をあらわし，Problem は Context の状況において想定される脅威を，Solution は Problem で示された想定脅威に対して，推奨される対策案を記述する．

これらの要求パターンは，AsseMis の手順 (2) までを行った結果のユースケース図上で，各パターンの Context の特徴を検索することで，のうち Web に共通する典型的な脅威 (Problem) および対策 (Solution) を自動的に抽出することが可能となり，セキュリティ有識者による脅威分析 ((3) 脅威抽出 (4) 脅威評価 (5) 対策抽出) を省力化することが可能となる．

3.3 設計のパターン化

AsseMis を用いた要求分析を行うことにより，一般の (非セキュリティ) 関心事の他に，セキュリティ関心事が対策として抽出されたので，Security Injector の設計においては，それぞれの関心事ごとに設計を行う．

3.4 セキュリティを挿入する特徴点の設計手法

Security Injector では，非セキュリティ関心事からセキュリティ関心事への依存性を排除する形で開発を進める必要がある．そのため，非セキュリティ関心事上にセキュリティを挿入するべき特徴が存在する必要があり，設計においてはその特徴を定義し，設計しなければならない．例えば図 2 の分析結果においては，対象となるユースケースに要機密性のデータが送信される場合，利用者識別認証の対策が必要になることがわかる．設計では，利用者識別認証のセ

[1]すべてのパターンの詳細については文献 [5] を参照

キュリティ関心事を設計し,「要機密性のデータを送信」する箇所に挿入する必要がある．このように，セキュリティ処理を挿入するための特徴点として，対象となるクラスやメッセージ引数がセキュリティプロパティの存在を利用することができる．そこで，非セキュリティ側のクラスを，AsseMis のユースケースよりセキュリティプロパティを継承する形で自動生成させることを提案する．例えば，図 2 における要機密性のデータに対応するクラスは,「C」というステレオタイプを持つクラスを生成すればよい．

3.5 セキュリティ関心事設計のパターン化

セキュリティ関心事側の設計では，上記の特徴点を利用し，シーケンス図などにおいてセキュリティを挿入する箇所を決定し，挿入のための特徴検索 (AOSD におけるポイントカット) 処理を設計する．この挿入箇所は，各シーケンスの設計仕様に依存するように見える．しかし，多くのセキュリティ対策においてはアプリケーションに依存しない形で再利用可能な横断点が存在することが分かった．例えば，利用者識別，認証やアクセス制御においては,「C/I/P のセキュリティプロパティを持つデータがメッセージの引数か返値になっている，クラス外より呼ばれるメッセージの実行か，または，A のセキュリティプロパティを持つクラスのクラス外から呼ばれるメッセージの実行のそれぞれ実行前」と定義することができる．従って，上記特徴を検出して処理を実行するアスペクトとして，AspectJ などのアスペクト指向言語で実装をすれば，同様のセキュリティを必要とするアプリケーションに対して再利用することが可能になる．

3.6 ロールベースアクセス制御 (RBAC) パターン

RBAC では，上記で示した特徴点の他に，次の情報を事前情報として用意する．

- 役割単位の権限定義ファイル
  利用者の役割とデータ，機能に対する権限の対応の表を記述．資産つきユースケース図から,「アクタがユースケースにリンク⇒「実行権あり」」などのルールを用いて自動生成することが可能．
- セキュリティプロパティを持つデータクラス
  資産名に対応するクラス名を付与するとともに，データの所有者のフィールド (システムにおける利用者の識別名と役割名) を付与して，ユースケースより自動生成

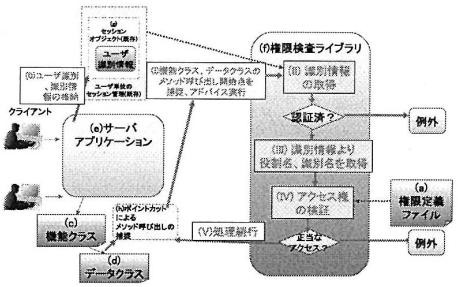

図 5: RBAC パターンの構成

次に，RBAC パターンの構成とアルゴリズムを図 5 に示す．

RBAC パターンは，以下のポイントカットを補足し，現利用者の情報と予め定義された権限定義ファイルの情報を検査するアドバイスが動作することにより動作する．

- C(機密性) プロパティを持つデータの参照メソッドの実行前
- I(完全性) プロパティを持つデータの更新メソッドの実行前
- P(プライバシー) プロパティを持つデータの参照実行前
- A(可用性) プロパティを持つクラスのメソッドの，クラス外からの実行前

4 効率化の評価

セキュリティ要求分析の作業量 (開発者，有識者による共同作業を除く) について，従来の

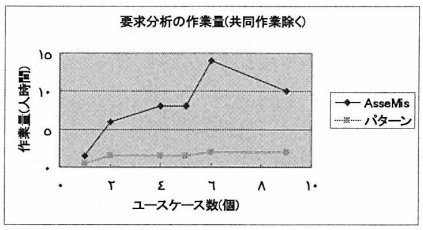

図 6: AsseMis とパターン適用の場合の作業量比較

AsseMis と，パターンを適用した場合を，同一アプリケーションの要求分析で計測した結果を図6に示す．対象としたWebアプリケーションはいずれも，実際に稼働し利用されるものである．AsseMis では，アプリケーションのユースケース量に比例して作業量が大きくなっているが，パターンにおいてはあまりユースケース量の影響を受けず，相対的に作業量が少なく済んでいることが分かる．

5 おわりに

本稿では，脆弱性のないセキュアなアプリケーションを開発する際の問題として，セキュリティ知識不足とセキュリティ作業に工数がかかる問題に着目し，関心事を分離する開発において，セキュリティ関心事の要求分析と設計作業を省力化，再利用するための要求パターンおよび設計パターンを開発し，提案した．要求パターンは，Webアプリケーションの場合，想定される(実装上の脅威も含む)脅威と対策を抽出することができ，かつ AsseMis を用いればパターンを適用すべき状況をパターンマッチングによって行うことが可能になっている．作業量においても，AsseMis のみを用いる場合よりも，パターンを用いた方が少ない作業量になることが確認できた．また，従来RBACをアプリケーション上で実現させるためには開発者によるアドホックな実装しか方法がなかったが，提案した設計パターンの適用により，開発者が実装しなくても，外部から権限検査処理を挿入することが可能になった．また，その処理を挿入されるのに必要な特徴は要求分析の結果によって自動的にあ与えられるため，特徴設計の省力化も可能になった．本稿では，RBACのパターンのみを扱った．他のセキュリティ機能のパターン化については今後の課題である．

参考文献

[1] Swiderski, F. and Snyder, W.: *Threat Modeling*, Microsoft Press (2004).

[2] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J. and Irwin, J.: Aspect-Oriented Programming, *the European Conference on Object-Oriented Programming, vol.1241*, pp. 120–131 (1997).

[3] Sindre, G. and Opdahl, A. L. I.: Eliciting Security Requirements by Misuse Cases, *Proc. TOOLS Pacific 2000*, pp. 120–131 (2000).

[4] Okubo, T. and Tanaka, H.: Identifying Security Aspects in Early Development Stages, *Proc. International Conference on Availability, Reliability and Security (ARES'08)*, Barcelona, Spain, pp. 1148–1155 (2008).

[5] Okubo, T. and Tanaka, H.: Web Security Patterns for Analysis and Design, *15th Cpmference on Pattern Languages of Programs (PLoP 2008)*, Nashville, TN, USA (2008).