フレームワークを利用したWebアプリケーション開発時における

セッション汚染対策の検討

Security enhancement against session taint in Web application development

木村 勇一† 後藤 厚宏†
Yuichi Kimura Atsuhiro Goto

1. はじめに

Webアプリケーションの安全性を脅かすものの一つに，セッション汚染[1]がある．これは，Webアプリケーション上の情報の格納先であるセッションに，不正なHTTPリクエストが格納されることで生じる．セッション汚染を考慮したチェック処理が行われていないと，Webアプリケーション内で二次的な障害を引き起こす可能性がある．一方，Webアプリケーションの開発現場に目を向けると，短納期やリソース不足，スキル不足といった問題を抱えることが多く，セキュリティ設計が十分に実施できない状況も見受けられる．フレームワークやライブラリのセキュリティ機能により対策可能な脆弱性はあるが，セッション内の業務情報は開発者がセキュリティを確保する必要がある．

そこで本研究では，フレームワークを利用したWebアプリケーション開発を対象に，業務情報の保護に特化したセッション管理方法を提案し、開発者の漏れ防止と負担軽減を図る．

2. 用語の整理

2.1 セッション管理

セッションとはユーザごとにサーバ側に作成されるWebアプリケーションが扱う情報の格納先である．セッションIDのみをクライアント側に保持させて，サーバ側にあるセッション情報と紐付けることで，業務情報をサーバのみで保持する．

本稿ではセッションで保持する情報のうち，Webアプリケーションごとに定義される業務固有の情報を業務情報と定義する．また，クライアント側から送信されたHTTPリクエストをセッションに格納する処理をリクエスト処理と呼ぶ．

2.2 Webアプリケーションフレームワーク

本稿ではフレームワークをWebアプリケーションに特化した基本的な枠組みと機能を実装したプログラム群として定義する[2].

フレームワークの例として，Java言語のフレームワークであるStrutsやSpringMVCがある．これらはMVCモデル[3]を採用し，この枠組みに従って開発者が実装を行うことで，プログラム全体の機能分割と再利用性の向上を実現している．また，基本的な機能として，リクエスト処理，セッション管理，DBアクセス処理などを提供し，開発者の作業範囲を業務固有のロジック処理のみに集約している．

†情報セキュリティ大学院大学
INSTITUTE of INFORMATION SECURITY

3. 先行事例・関連研究と課題

3.1 入力値検証ツール

不正な入力値の検知技術としてWAF（Web Application Firewall）がある．Webアプリケーションに HTTP リクエストが渡される前にチェックを実施して，不正な構文や文字列パターンを検知する．また，SecureBlocker [4]というWAFと同機能を持つフィルタ処理ライブラリがある．これは Web アプリケーション内のリクエスト処理の一環として不正情報を検知する．

これらの入力値検証ツールの多くは，HTTP リクエストに含まれる脆弱性攻撃特有の異常な構文や文字列パターンをチェックすることにより，SQL インジェクションや OS コマンドインジェクション，XSS（Cross Site Scripting ）などを検知する．しかし，入力値の意味的な検証は実施することができない．なぜなら意味的な検証は入力値が業務情報として正当かどうかのチェックであり，業務要件を加味したチェックルールが必要となるためである．

業務情報が不正な値に書き換えられた場合，例えば権限の書き換えによる権限昇格やユーザ ID の書き換えによる他ユーザの情報更新が発生する可能性がある．これらについては，汎用的なチェックツールでは対処することは難しい．

3.2 セッションに着目した研究

セッションハイジャックやセッション固定攻撃の対策は既存研究が多い．これらの攻撃は，攻撃者がユーザのセッション ID を不正に入手・操作することで，セッションを乗っ取る攻撃である．基本的な対策として，XSS などの脆弱性によってセッション ID を盗まれないようにすること，セッション ID を任意に指定されないようにログイン処理後に再発行することが必要である[5]．学術研究としては，Web アプリケーションのシステム構成やユーザをモデル化して，モデル検査によって脆弱性を早期検知する方法[6]や構築されたシステムを解析し自動的に攻撃コードを生成・実行して，解析を行う手法[7]が提案されている．

このように，セッション ID の保護については議論が行われているが，セッション汚染や業務情報の保護についての議論は少ない．また，対策については設計時のチェックや試験の自動化ツールの活用が提案されているが，実装レベルでの対策となる方法は少ない．

4. フレームワーク利用の課題

4.1 リクエスト処理

リクエスト処理について，フレームワークの多くが開発者の作業負担を軽減するため，HTTP リクエストをセッション内のプロパティに自動マッピングする機能を提供している．自動的にマッピングを行うので，入力値が正規の方法で送信されたものか，不正な方法で差し込まれたかどうかを判断することはできない．これに対して，指定したプロパティのみを取り込む拡張機能[8]やユーザ情報をセッション上の外部入力値の影響を受けない領域に設定する機能[9]を持つフレームワークがある．

ただし，フレームワークは目的ごとに特化されたものであるため，初期構築時に提供されていない機能については，設計工程にて吸収する前提で採用される場合がある．また，既に構築された Web アプリケーションはフレームワークを別のものに切り替えることが難しい．

4.2 フレームワーク利用を踏まえたセキュリティ設計

フレームワークを利用しなければ漏れがなく実装される処理も，フレームワークの利用によって見落とされる場合がある．リクエスト処理にて自動マッピングを行うフレームワークを利用する際は，開発者は業務仕様だけではなく，フレームワークの動作を把握し，サポートしていない動作を補完する設計を行う必要がある．

5. 提案

5.1 コンセプト

研究の全体像を図 1 に示す．3 章，4 章で述べたように，開発者は業務情報の正当性チェックとフレームワークがサポートしていない動作の補完について，その処理設計を行う必要がある．

そこで開発者の漏れ防止と負担軽減を図るため，フレームワークが行うリクエスト処理機能を拡張して，セッション情報を保護する汎用的な仕組みを Web アプリケーションの枠組みに追加する方法を提案する．提案方法では，業務情報の管理を明確にして，チェック処理が見落とされることを防止する．またこの処理を自動化し，システム全体に動作させることで，作業の切り分けと軽量化を推進する．

5.2 機能概要

5.2.1 セッションの保護

セッション汚染の防止機能と検証機能を用意する．防止機能とは，セッションに格納される情報のうち汚染を防止したいプロパティについて HTTP リクエストを取り込まないようにする機能である．検証機能とは，選択したプロパティについてリクエスト処理時に任意のチェック処理を差し込めるようにする機能である．

5.2.2 アノテーションによる保護対象識別

Java 言語のメタデータとして記述されるアノテーション[10]を本提案用に定義する．それをプロパティに設定することで保護対象を識別し，セッションの保護を実現する．プログラムにアノテーションを記述する作業は，処理ロジックの記述と比較して，より作業量が少ないと見込んでいる．

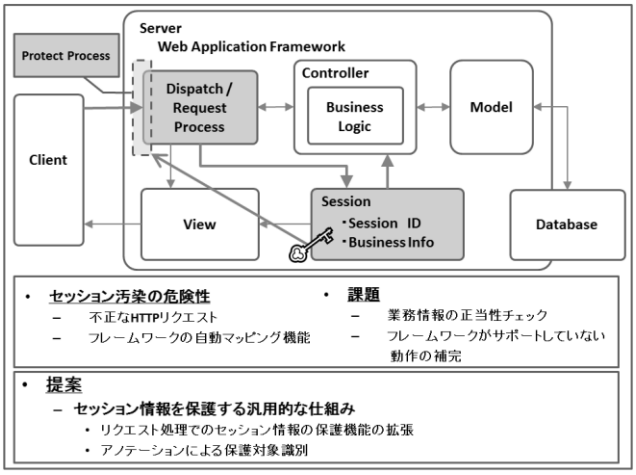

図 1　研究の全体像

6. まとめ

本稿では，フレームワークを利用した Web アプリケーション開発を対象として，セッション管理と入出力検証の先行事例・関連研究とフレームワーク利用の課題の考察を行った．その対処策として，セッション情報を保護する方法を提案した．

今後，引き続き提案方法の開発現場での有用性を考察し，具体的な実装と検証を行う．

検証は，異なるフレームワークで構築された脆弱性を持つ検証用 EC サイトに対して，本提案機能を設定し，脆弱性の減少効果を確認する．特に異なる権限ごとの振る舞いが正しく行えることに重点を置いて検証を行う．また，同様の機能を持つ Spring MVC のバインダ機能[8]と比較し，機能性や修正の容易性を検証する．

参考文献

[1] Carnegie Mellon University, "Taint Analysis", http://users.ece.cmu.edu/~dbrumley/courses/18487-f10/files/taint-analysis-overview.pdf, (accessed 2013-06-30)

[2] Ralph E. Johnson, "Frameworks=(Components+Patterns) ", Communications of the ACM Volume 40 Issue 10, Oct. 1997

[3] Oracle, "Java SE Application Design With MVC", http://www.oracle.com/technetwork/articles/javase/index-142890.html, (accessed 2013-06-30)

[4] 株式会社 NTT データ, "2005 年ニュースリリース：簡単，Web アプリケーション脆弱性対策ツール『SecureBlocker』を展開", http://www.nttdata.com/jp/ja/news/release/2005/101700.html, (accessed 2013-06-30)

[5] IPA 独立行政法人 情報処理推進機構, "安全な ウェブサイトの作り方", http://www.ipa.go.jp/files/000017316.pdf, (accessed 2013-06-30)

[6] 森川, 山岡, 中山, "モデル検査による Web セッションのセキュリティ欠陥検出手法", 信学技報 KBSE, 知能ソフトウェア工学 107(5), 29-34, 2007-04-12

[7] 高松, 小菅, 河野, "セッション管理の脆弱性検査の自動化", コンピュータセキュリティシンポジウム 2011 論文集

[8] GoPivotal, "Spring Framework API DataBinder : setAllowedFields", http://static.springsource.org/spring/docs/1.2.x/api/org/springframework/validation/DataBinder.html, (accessed 2013-06-30)

[9] 株式会社 NTT データ, "TERASOLUNA Server Framework for Java（Web 版） 機能説明書 ：ユーザ情報保持機能"

[10] Oracle, "The Java Tutorials Lesson: Annotations", http://docs.oracle.com/javase/tutorial/java/annotations/, (accessed 2013-06-30)