B-005

既存 Web アプリケーションの入力処理の脆弱性調査と対策
Input processing vulnerability of Web application and its countermeasures

木村 勇一† 大久保 隆夫† 後藤 厚宏†
Yuichi Kimura Takao Okubo Atsuhiro Goto

1. 概要

既存 Web アプリケーション（WebAP）の入力処理の脆弱性を調査し，対策を考察した．

2014 年 4 月に広く認知された Struts2，及び Struts1 の脆弱性（Struts 脆弱性）は，Java の基本的な枠組みで発生する脆弱性である．課題は WebAP の内部構造のブラックボックス化とパラメタの正否を識別する仕組みの欠如である．対策としてホワイトリストのパラメタチェックフィルタがある．ホワイトリストは必要なパラメタのみ取り込み，その他のパラメタを除外するので，ブラックボックスとなるコンポーネント利用時の未知の脆弱性の防止に有効である．対策となるチェックルールの設定方法 3 案を提示する．

2. 既存 WebAP のセキュリティ対策

近年の Java による WebAP 開発では，ライブラリや WebAP フレームワーク（WebAP FW）などのコンポーネントが利用されている．その代表が Apache Foundation から提供されている Apache Commons や Struts1 である．これらを利用した WebAP は数多くある[1]．例えば，株式会社 NTT データの全社標準の WebAP FW「TERASOLUNA Server Framework for Java(Web 版)」（TERASOLUNA FW）は，これらのコンポーネントを内蔵したものである．

既に実績のあるコンポーネントを再利用することは，ソフトウェア工学の観点，開発現場の効率化やコスト意識の観点に合致する．しかし，コンポーネントに欠陥がある場合，コンポーネントを利用して構築された全ての AP は，欠陥を起因とする故障の可能性が生じる．そのため，本稿では既に構築された WebAP を対象とするセキュリティ対策について述べる．

3. Struts と脆弱性対策

3.1 Struts 脆弱性

2014 年 4 月に IPA より，Apache Struts2 の脆弱性(CVE-2014-0094)，また，これに類似する脆弱性が Struts1 にも含まれているとの注意喚起が行われた．この脆弱性は，パラメタの差し込みにより ClassLoader を操作され，WebAP の動作権限内で情報の窃取や特定ファイルの操作，及び WebAP を一時的に使用不可となる可能性がある緊急性の高い脆弱性である．

この脆弱性が Struts1 にも存在することは，Struts2 に脆弱性が存在することより影響の大きいものであった．理由は 3 点ある．まず Struts1,2 は内部構造の異なる実質的に別の WebAP FW であるため，類似の脆弱性があることは，より根本的な箇所での不具合と考えられること．次に，普及率の点では，Struts2 ベースの WebAP よりも，Struts1 ベースの WebAP が現在多く存在すること[1]．最後に，Struts1 は既に提供元からのサポートが終了しており，開発者が独自に対策を取る必要があることである．

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

3.2 Struts 脆弱性の原因

Struts 脆弱性の原因の一つは，Struts1,2 が利用するコンポーネントに脆弱性が存在することである．Struts1,2 が利用する Apache Commons の BeanUtils というオブジェクト間のパラメタマッピングを行うコンポーネントは，オブジェクトの親クラスが持つ getClassLoader メソッドを行可能である．AP サーバ Apache Tomcat8 系の場合，取得された ClassLoader に AP のリソースを操作するメソッドが含まれるため，情報の窃取や特定ファイルの操作を行うことが可能となる．

脆弱性のもう一つの原因は，Struts1,2 の Web 上で動作する AP としての考慮の欠如である．本来，Apache Commons や Struts1 は，技術者の学習教材として利用されるほど理想的な Java の実装パターンであるが，その実装パターンに WebAP としての考慮不足がある．すなわち，必要なパラメタのみを取り込み，それ以外のパラメタを除外する仕組みが WebAP FW にないことは脆弱性の原因の一つである．

3.3 先行研究

Struts 脆弱性を突いた攻撃をパラメタ改ざん攻撃として分類する．Skrupsky の研究[2]では，負のパラメタ改ざん攻撃を挙げ，PHP で構築された CMS を対象に，対策の自動化を図っている．木村の研究 [3]では，WebAP FW の自動的なパラメタマッピングと，内部の保持情報の定義をまとめることの組み合わせを原因とする入力処理の脆弱性の調査と対策検討を行った．木村は検証用 WebAP を作成して入力処理の脆弱性の説明を行ったが，本稿では Struts 脆弱性を類似のものと考え，Struts 脆弱性と入力処理の脆弱性とを関連付けた説明を行う．

4. Java の WebAP の課題

4.1 WebAP の内部構造のブラックボックス化

WebAP の開発者は，通常 WebAP FW やライブラリのソースコードを解読して，その正当性を確認することはしない．利用実績からそれらのコンポーネントを信頼しているためである．このとき WebAP の内部コンポーネントはブラックボックスとなり，結果として WebAP は開発者の意図しない動作をする可能性が生じる． このため，コンポーネントを活用するためには，必要な操作のみを許可し，それ以外を実行させない入口対策が重要となる．

4.2 パラメタの正否を識別する仕組みの欠如

Web 上で動作する AP は，信頼されたクライアントとの通信の他に，不特定のクライアントからの不正なパラメタを受け取る可能性を持つものである．

Java は，C 言語ライクのオブジェクト指向言語として開発され，コンポーネントの再利用を想定した大規模 AP の開発に適している．その延長として Web AP のための仕様 JSP/Servlet が整理され，WebAP 向けの機能を提供する WebAP FW の Struts1 が利用されるようになった．

Java にはアクセス修飾子という，フィールドやクラス，メソッドに対するアクセス制御の仕様がある．外部からのパラメタに対する WebAP のためのアクセス制御の仕組みは，アクセス修飾子と同様に Java の設計思想に合致する．JSP/Servlet には外部からのパラメタへの標準化されたアクセス制御の仕組みはない．そのため，WebAP FW にはパラメタマッピング機能と併せて，この仕組みが必要である．

5. チェックフィルタによる対策

既存 WebAP のセキュリティ対策に有効なパラメタチェックフィルタについて，対策箇所とチェックルールの方針，チェックルールの設定方法を述べる．

5.1 対策箇所

ServletFilter にチェックフィルタを実装する方法が，外部機器との依存性がなく汎用的であるため，適している．

ServletFilter 以外の対策候補に，WAF によるシグニチャ検知，WebAP FW の修正がある．前者は，短期間での対策適用が可能で WebAP の修正が不要となるが，機器の調達コストが発生し，WebAP と機器との間に依存関係が生じるため，開発者は適用以降，機器に設定したチェックルールのメンテナンス作業が必要となる．このため開発者が WebAP の修正による既存 WebAP のセキュア化を目指す場合は適さない．後者は， TERASOLUNA FW ではこの方法が採用され[4]，サポートの終了した Struts1 に改修を加え，TERASOLUNA FW が内蔵している Struts1 を修正版に置き換えることで対策を行っている．ただし，複数の異なる WebAP FW を対象とし，汎用的で WebAP への影響が少ない対策を目指すため，適さない．

5.2 チェックルールの方針

方針として，ホワイトリストとブラックリストがある．ホワイトリストのチェックルールは，必要なパラメタのみ取り込み，不要なパラメタを除外するため，開発者の想定外の動作の低減に繋がる．ただし，ホワイトリストは WebAP 毎に必要となる全てのパラメタを一覧化し，パラメタが追加／変更となる場合は，一覧の更新が必要なので柔軟性は低い．その点ではブラックリストが除外対象のみを一覧化するので柔軟性に優れる．TERASOLUNA FW の修正パッチは，必要最小限のチェックを行うポリシーのもとブラックリストのチェックルールを採用している[4].

Struts 脆弱性に当てはめると，事後対策ならばブラックリストで十分だが，未然に防止可能なのはホワイトリストである．このため，ブラックボックスとなるコンポーネント利用時の未知の脆弱なパラメタを考慮するので，ホワイトリストでなければ対処できない．

5.3 ホワイトリストのチェックルールの設定方法

内部処理や画面処理とは別に外部に切り出した設定ファイルを利用する方法，内部情報の格納クラスに直接アノテーションを設定する方法，画面を生成する HTTP レスポンスからチェックルールを自動生成する方法がある．

設定ファイルは，全ての呼び出しタイミング毎に必要なパラメタを定義するため，作業量は膨大となるが，他処理への依存性が少なく，直感的に分かりやすい．

アノテーション[5]は，クラスに画面処理のパラメタ名を注釈として記述するため，依存性が生じ，再利用性が低下する．ただし，ソースコードの自動生成ツールの利用によっては，再利用の考慮は不要となる場合もある．

チェックルールの自動生成は，動的にレスポンス情報から生成するため，ルールの作成作業が不要となり，修正作業量が最小化される．サーバサイドでの動的な入力項目の変化にも対応可能である．しかし，開発者自身がチェックルールの内容を把握できない，HTTP レスポンスから生成されるチェックルールはクライアントサイドで動的に入力項目を変更された場合に正否を判定できないという制約がある．ajax や HTML5 などのクライアントサイドでのプログラミングを考慮すると，この制約は大きな欠点である．

各方法は，WebAP 毎の特性や構成により採択が異なると考えるため，適用パターンとして整理するまでとする．

6. まとめと課題

本稿では，Struts 脆弱性を取り上げ WebAP の課題として内部構造のブラックボックス化とパラメタ取り込み可否を識別する仕組みの欠如を挙げた．課題に対する有効な対策として，WebAP にはホワイトリストのチェックフィルタがあり，未知の脆弱性の防止に繋がることを述べ，そのためのチェックルールの設定方法 3 案を提示した．今後の課題として，異なる WebAP FW やアーキテクチャを対象に調査を継続し，個々の具体的な検証を実施していくことが重要である．

参考文献

[1] 日経産業新聞," サポート切れ企業に脅威、システム構築ソフト「ストラッツ１」、自社対策へ。",
"https://messe.nikkei.co.jp/ss/news/123126.html" ,
(accessed 2014-06-29)

[2] Nazari Skrupsky, Prithvi Bisht, Timothy Hinrichs, V. N. Venkatakrishnan, Lenore Zuck," TamperProof: A Server-Agnostic Defense for Parameter Tampering Attacks on Web Applications", CODASPY 2013 ,(2013).

[3] 木村 勇一, 後藤 厚宏, "既存 Web アプリケーションの入力処理の脆弱性調査と対策", 情報処理学会研究報告. ソフトウェア工学研究会報告 2014-SE-183(2), 1-8, (2014)

[4] 株式会社 NTT データ,
"Apache Struts 1.2.9 with SP1 by TERASOLUNA",
"http://sourceforge.jp/projects/terasoluna/wiki/StrutsPatch1-JP",
(accessed 2014-06-29)

[5] IBM," Annotations in Tiger, Part 2: Custom annotations",
" https://www.ibm.com/developerworks/library/j-annotate2/" ,
(accessed 2014-06-29)