# CVE-2022-43484について

Table of Contents



## 1. はじめに

本ドキュメントは、TERASOLUNA フレームワークユーザの皆様に、CVE-2022-43484に関する情報提供を行う目的で公開するものです。 本ドキュメントの内容を参考に各システムにおける対策立案を行ってください。

## 2. 本脆弱性の概要

本脆弱性はクラスローダマニピュレーション(クラスローダを操作可能な脆弱性)であり、これを利用した攻撃方法や影響は実行環境によって異なります。

APサーバがTomcatである場合、リモートコード実行(RCE)が可能であることが確認されています。これ以外のAPサーバにおける影響は明確ではありませんが、クラスローダマニピュレーションはAPサーバの種類に関係なく可能であるため、Tomcat以外のAPサーバにおいても何らかの攻撃を受ける可能性があります。

本脆弱性はCVE-2022-22965(Spring4Shell)と同様のものですが、特定のSpringのバージョンの範囲では、利用するJDKのバージョンに関係なく本脆弱性を利用した攻撃が可能である点に注意が必要です。

## 3. 本脆弱性の影響範囲

本脆弱性は、Spring 3.2.6よりも古いバージョンを利用する、以下のTERASOLUNA フレームワークのバージョンで影響を受ける可能性があります。

- TERASOLUNA Global Framework 1.0.0(Public review version)
- TERASOLUNA Server Framework for Java(Rich版) 2.0.5.1以下のすべてのバージョン

本ドキュメントでは、TERASOLUNA Global Framework 1.0.0(Public review version)に関する内容を記載します。TERASOLUNA Server Framework for Java(Rich版)については、

- https://osdn.net/projects/terasoluna/wiki/cve-2022-43484

をご確認ください。

## 4. 本脆弱性の対策方法

本脆弱性は、以下のいずれかの方法で対処可能です。

- フレームワークのバージョンアップ
- 原因となるパラメータやリクエストの無視・遮断(ワークアラウンド)

### 4.1. フレームワークのバージョンアップ

以下のバージョンアップを行うことで対処可能です。可能な限りこちらの方法を採用することを検討してください。

- TERASOLUNA Global Framework 1.0.0(Public review version)を利用している場合
    - TERASOLUNA Server Framework for Java 5.7.1.SP1(Spring Framework 5.3.18を利用)にアップデートする
        - 本脆弱性のみであれば、TERASOLUNA Global Framework 1.0.1(Spring Framework 3.2.10を利用)以降へのアップデートのみで対策できますが、TERASOLUNA Global Framework の後継の TERASOLUNA Server Framework for Java 5.x では、これまでにその他の脆弱性対策も行っているため、最新のバージョンを利用することを推奨します。

#### 4.1.1. TERASOLUNA Global Framework 1.x のバージョンアップ方法

https://github.com/terasolunaorg/terasoluna-gfw/wiki#1x で公開しているため、こちらをご参照ください。

### 4.2. 原因となるパラメータやリクエストの無視・遮断(ワークアラウンド)

バージョンアップが行えない場合、名前が

- class.~
- Class.~
- ~.class.~
- ~.Class.~

に該当するパラメータをすべて無視するか、上記のいずれかのパラメータを持つリクエストを遮断することで本脆弱性の影響を回避することができます。

これは、CVE-2022-22965に対するものと同様であり、CVE-2022-22965の対応として上記の対策を行っている場合は本脆弱性への対処は不要です。 ただし、本脆弱性はCVE-2022-22965と異なり、JDKに関係なく影響を受けるため、JDK8以前へのバージョンダウンは本脆弱性への対処にはならないことに注意してください。

また、TERASOLUNA Global Framework 1.0.x は、JDK9以降には対応していませんが、もしも、JDK9以降で運用している場合は、本脆弱性の影響を受けない TERASOLUNA Global Framework 1.0.1 以降であっても、CVE-2022-22965への対処(後述の本脆弱性の対策と同じ方法)が必要となるため、ご注意ください。

なお、原因となるパラメータやリクエストを無視・遮断する方針には、以下のようなものが考えられます。以降では、これらの具体的な実現方法について記載します。

- WAF(ウェブアプリケーションファイアウォール)を利用して対策する
- アプリケーションの実装で対策する
- Tomcatをバージョンアップする(暫定対処)

#### 4.2.1. WAF(ウェブアプリケーションファイアウォール)を利用して対策する

上記のいずれかのパラメータを持つリクエストを遮断するよう設定してください。実際の設定方法は製品ごとに異なるため、ベンダにお問い合わせください。

#### 4.2.2. アプリケーションの実装で対策する

##### TERASOLUNA Global Frameworkのアプリケーションで対策を行う場合

コンポーネントスキャン範囲に以下のようなControllerAdviceクラスを配置することで、対策可能です。

*対策用ControllerAdviceクラスの実装例*

```java
package xxx;

import org.springframework.core.Ordered;
import org.springframework.core.annotation.Order;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.InitBinder;

@ControllerAdvice
@Order(Ordered.LOWEST_PRECEDENCE)
public class BinderControllerAdvice {

    @InitBinder
    public void setAllowedFields(WebDataBinder dataBinder) {
        String[] denylist = new String[]{"class.*", "Class.*", "*.class.*", "*.Class.*"};
        dataBinder.setDisallowedFields(denylist);
    }
}
```

DataBinder(WebDataBinder)のsetDisallowedFieldsをアプリケーション内で既に使用している場合、上記の方法では、

- 上記の対策が取り消される
- アプリケーションでもともと実装していた方が取り消される

という結果になる可能性があるため、無視するパラメータのパターン(setDisallowedFieldsの引数)に、"class.*", "Class.*", "*.class.*", "*.Class.*" をマージするよう対策を行ってください。

対策が有効になっているかについては、ControllerAdviceによる対策を行った後に、org.springframework.validation.DataBinderのデバッグログが出力されるよう設定した状態で、擬似的な攻撃を行うことで確認することができます。

*DataBinderのデバッグログを出力するためにlogback.xmlに追加する設定*

```xml
<logger name="org.springframework.validation.DataBinder">
    <level value="debug" />
</logger>
```

*疑似的な攻撃用パラメータ*

```
・class.a=a
・Class.b=b
・x.class.c=c
・x.Class.d=d
```

Formオブジェクトを引数に持つハンドラメソッドを呼び出すためのパスにリクエストを送信する際、上記の4つのパラメータを付与してください。

GETメソッドのリクエストを受け付けているパスであれば、アドレスバーにて、パスの後ろに以下のようなクエリ文字列を付与してください。

*付与するクエリ文字列*

```
?class.a=a&Class.b=b&x.class.c=c&x.Class.d=d
```

既にクエリ文字列がある場合は先頭の「?」を「&」に置き換えて連結してください。

検索画面やページネーションリンクからのリクエストで検索処理を行うパスであれば、

- ハンドラメソッドの引数にFormオブジェクトがある(必須条件)
- GETメソッドのリクエストを受け付けている(確認用のリクエストを送りやすい条件)

の両条件を満たしている可能性が高いです。

一方、POSTメソッドのリクエストしか受け付けていないパスの場合、アプリで元々扱っていないリクエストパラメータの追加を行うには、リクエストあるいは画面のHTMLを改ざんするツールを用いるなど、GETメソッドの場合と比較して、工夫が必要になります。

上記の疑似的な攻撃用パラメータによる攻撃を行い、以下のようなログが出力される場合、対策が有効になっています。

*上記の疑似的な攻撃用パラメータによる攻撃を防いだ際に出力される、DataBinderのデバッグログの例(メッセージ部分のみ抜粋)*

```
~Field [Class.b] has been removed from PropertyValues and will not be bound, because it has not been found in the list of allowed fields
~Field [class.a] has been removed from PropertyValues and will not be bound, because it has not been found in the list of allowed fields
~Field [x.Class.d] has been removed from PropertyValues and will not be bound, because it has not been found in the list of allowed fields
~Field [x.class.c] has been removed from PropertyValues and will not be bound, because it has not been found in the list of allowed fields
```

順不同で、すべてのログが出力されていることを確認してください。疑似攻撃を誤っていないにも関わらず、1つでも出力されていないログがある場合、対策誤りにより対策漏れがある状態です。

### 4.2.3. Tomcatをバージョンアップする(暫定対処)

本脆弱性と類似した脆弱性である、CVE-2022-22965 のPoCとして、Webアプリケーションが Tomcat 上で動作している場合にリモートコード実行が可能となる攻撃例が知られています。このPoCによる攻撃を成立させないように Tomcat 側で対策されたものが Tomcat の開発元である Apache から提供されています。

Tomcat側で取られた対策は本脆弱性に対しても有効であるため Tomcat を 9.0.62 もしくは 8.5.78 にバージョンアップすることでPoCに準ずる攻撃への対策が可能です。なお、Tomcat 10.0.20 でも同様の対策が行われていますが、少なくとも TERASOLUNA Server Framework for Java 5.7.x までは、Tomcat 10 では動作しない点に注意してください。

Tomcatをバージョンアップする対応は特にEOLとなっている旧バージョンのSpring Frameworkを使用している場合には回避策の一つとなり得ますが、 本脆弱性の根本的な原因を修正したものではなく、今後別の攻撃手法が公開される可能性も否定できないため、あくまで一時的な回避策であることにご留意ください。

Tomcatバージョンアップによる対策は、以下のような条件をすべて満たす場合における暫定対処として実施することをおすすめします。

- この回避策以外に記載している、いずれの回避策もすぐには実施できない
- その間システムを停止することができない
- Tomcatを使用している(直ちに攻撃を受ける可能性がある)
- Tomcatのバージョンアップであればすぐに実施できる