富士通 Java ブログ

セキュリティマネージャー廃止への準備を始めましょう ～ 非推奨（Java SE 17）から廃止（Java SE 24）となる Java Security Manager ～

2025 年 3 月 3 日 初版

数村 憲治

2025 年 3 月に Java SE 24 がリリースされる予定です。このリリースの中でもっとも影響の大きい変更の一つに、 [セキュリティマネージャーの廃止] があります。本稿では、アプリケーションに与える影響、代替方法、そして、セキュリティマネージャー廃止の背景と今後の Java のセキュリティ強化の方向性について紹介します。

なお、本文で使用している Java プログラムの実行例は、特に断りがない限り OpenJDK 24 のベータ版を用いています。

セキュリティマネージャー（Java Security Manager）廃止の影響

セキュリティマネージャーとは

セキュリティマネージャーを簡単に言うと、特定の API 使用を特定のコードに限定許可する機能です。これはサンドボックスとも言われています。
例えば、ローカルマシンのディスクにパスワード情報があった場合、ファイルアクセス関連 API を信頼できるコードだけに許可することで、 パスワード情報へのアクセスを制限することができるようになります。 では、「信頼できるコード」とはどのようなコードでしょうか？
自分自身で作成したコードはそれを自分で動作させる場合には信頼できるコードと言えるでしょう。それに対して、インターネット経由で作者不明のコードをダウンロードしてきた場合は、一般的には信頼できないと考えられます。
このような信頼できないコードが使われてしまう例としては、ブラウザ上で動くアプレットが典型的でした。

セキュリティマネージャー廃止で既存コードはどうなるか

既存コードがセキュリティマネージャーを使っていなければ何も影響はなく、ここでこのページを閉じていただいてかまいません。

本稿では「廃止」という言葉を使っていますが、Java SE 24 で実際に行われるのは、セキュリティマネージャー関連の API（クラスファイル）が JDK のランタイムから削除されるのではなく、セキュリティマネージャーをどうやっても有効にできなくなるということです。

以下の Hello という Java プログラムを例にして、みていきましょう。

```
class Hello {
  public static void main(String ... args) {
    System.out.println("Hello");
  }
}
```

セキュリティマネージャーを有効にするにはいくつか方法がありますが、まず、コマンドラインでの有効化を試します。

```
$ java -Djava.security.manager Hello
Error occurred during initialization of VM
java.lang.Error: A command line option has attempted to allow or enable the Security Manager. Enabling a Security
Manager is not supported.
        at java.lang.System.initPhase3(java.base@24-beta/System.java:1947)
```

このように、java.lang.Error が発生し、プログラムが動作しません。「Hello」という文字は出力されません。

参考ですが、Java21 で実行した場合には、

```
$ java -Djava.security.manager Hello
WARNING: A command line option has enabled the Security Manager
WARNING: The Security Manager is deprecated and will be removed in a future release
Hello
```

ワーニングは出るものの、プログラムの実行は継続されます。「Hello」という文字は出力されます。
しかし、Java SE 24 以降では、プログラムの起動すらできなくなります。

その他のコマンドライン指定方法でも同様で、

```
-Djava.security.manager=""
```

```
-Djava.security.manager=com.fujitsu.example.SM
```

これらはすべてエラーとなります。
ただし、唯一、

```
-Djava.security.manager=disallow
```

この方法だけ、エラーなくプログラムを起動できます。ただし、"disallow" は許可しないという意味なので、セキュリティマネージャーは有効になりません。

次に、プログラム内で Java API を利用しセキュリティマネージャーを有効にする場合を見ていきます。
前掲の Hello プログラムを少し改造し、System::setSecurityManager の呼び出しを追加しました。

```
class HelloSM {
  public static void main(String ... args) {
    System.out.println(”Hello”);
    System.setSecurityManager(null);
    System.out.println(”complete setting Security Manager”);
  }
}
```

このプログラムを Java24 で実行すると

```
$ java HelloSM
Hello
Exception in thread ”main” java.lang.UnsupportedOperationException: Setting a Security Manager is not supported
        at java.base/java.lang.System.setSecurityManager(System.java:286)
        at HelloSM.main(HelloSM.java:4)
```

となり、setSecurityManager の呼び出し前までは正常に実行できますが、 setSecurityManager の呼び出しはエラーとなり、Java プログラムはそこで終了します。

セキュリティマネージャー使用有無の調べ方

セキュリティマネージャー関連 API を使用しているかどうかは、すべてのプログラムを自身・自社で開発し、すべてのソースコードが利用可能であれば、ソースコードを関連 API で検索すればわかります。しかし、サードパーティープログラムの場合や、自社の資産でもソースがなくなっている場合、ソースコードを検索することができません。
このような場合に有用なツールとして、JDK には、jdeprscan というコマンドが提供されています。このコマンドは、jar ファイル、フォルダー、および単一クラスを対象として、使用している deprecation 機能を検出します。
以下は、前掲の HelloSM プログラムを対象に jdeprscan コマンドを実行した結果です。

```
$ jdeprscan --class-path . HelloSM
class HelloSM uses deprecated method java/lang/System::setSecurityManager(Ljava/lang/SecurityManager;)V
(forRemoval=true)
```


このように、どのクラスで、どの deprecate API を使用しているかがわかるようになります。
なお、セキュリティマネージャー関連 API については、JDK17 以降の jdeprscan コマンドで検出できます。

セキュリティマネージャー（Java Security Manager）の代替

セキュリティマネージャーの主な用途はサンドボックス、すなわち、コードの実行を制限することですが、これは OS などが提供している機能を使っても同様のことができます。これらの機能説明は本稿の範囲を超えているため、ここではその紹介のみにとどめます。

Secure Computing Mode（seccomp）

seccomp は Linux カーネルが提供している機能で、使用できるシステムコールを制限することができます。 詳細は、 [カーネルドキュメント] などを参照ください。

AppArmor

AppArmor も Linux カーネルが提供している機能で、使用できるリソースを制限することができます。詳細は、 [カーネルドキュメント] などを参照ください。

仮想化技術

コンテナやハイパーバイザーは、アプリケーションに独立した環境を用意し、リソースアクセスなどを制限することができます。特に、Docker では、seccomp との[組み合わせ] で柔軟なサンドボックスを実現できます。

セキュリティマネージャー（Java Security Manager）廃止の背景と Java における今後のセキュリティ強化方針

セキュリティマネージャーは、サンドボックスを目的に JDK1.0 より提供されてきました。Java が注目されるきっかけとなったのは、ブラウザ上で動くアプレットでしたが、その仕組み上、ネットワークから信頼できない任意のコードをダウンロードしローカルで実行できてしまうため、サンドボックスの提供は必然的であったと考えられます。
しかし、時代は進み、クライアント技術が進化したことでアプレット自体が廃止となりました。もちろん、サーバーサイドでサンドボックスを使うことも可能ですが、サーバーサイドでは出自不明の信頼できないコードが動くことはほとんどありません。したがって、現在のサーバーサイドでのセキュリティを考えるときに、サンドボックスは有効なセキュリティ技術ではなくなりました。
現在のセキュリティリスクは、信頼できないコードが動作することではありません。巧妙に細工したり改ざんしたデータを入力することでプログラムが意図しない動作になるような脆弱性が、信頼できるコードに含まれうることがセキュリティリスクとなってきています。そのため、Java における今後のセキュリティ強化分野は、以下のようなネットワークや暗号化関連技術にフォーカスしていくことになります。

- TLS1.3 や HTTP / 3.0 などの信頼性の高いネットワークプロトコル
- HSS / LMS や SHA-3 などの強度の高い暗号化アルゴリズム
- ポスト量子暗号のベースとなる鍵の扱いに関する API（ [JEP452] / [JEP478] ）

おわりに

本稿では、Java の黎明期から存在したセキュリティマネージャーの廃止について紹介してきました。
実際に、セキュリティマネージャーが使えなくなるのは、Java SE 24 導入後からとなりますが、特に、古い資産を活用している場合には、セキュリティマネージャーの使用箇所の洗い出しや修正に少なからずコストがかかることが予想されるため、早めに準備を始めるとともに、その際には本記事を参考にしてください。