コンピュータセキュリティ 15－3
（２００１．１２．２１）

セキュアな Java ソースコードの作成とその監査手法

兒島 尚*　鳥居 悟*
*（株）富士通研究所

Web アプリケーションに対するセキュリティ被害が増加している。それらの原因の多くはアプリケーションの脆弱性にあり、セキュアなアプリケーション構築技術が急務である。今後、Web アプリケーションの開発言語は、Java に移っていくと予想され、セキュアな Java アプリケーション構築技術が必要となる。我々は既存の構築技術を、ガイドライン、監査ツールという観点で分類し、その問題点を明らかにする。そして、これらの問題点を解決するガイドラインを紹介し、これに沿った監査手法について提案する。

Guidelines and Technique of Auditing for Secure Java Code

Hisashi Kojima*　Satoru Torii*
* Fujitsu Laboratories Ltd.

The number of security breaches continues to grow steadily. Software vulnerabilities are the most common form of such breaches, therefore the technique to build non-vulnerable software is a pressing need. Java is likely to be the main programming language of web application development, and then the technique to build secure Java software is needed. We classify current technique of building secure software into guidelines and auditing tools, and explain their problems. Then we present our guidelines, and suggest the technique of auditing in accordance with such guidelines.

1. はじめに

セキュリティ被害が増加している[1]。Web を用いた電子商取引は広く一般的なものとなっているが、増加するセキュリティ被害がその基盤を大きく揺るがしかねない状況になってきている。セキュリティ被害の主要な原因となっているものは、Web アプリケーションの脆弱性である[2]。脆弱性のない、セキュアな Web アプリケーションの構築が急務であるといえよう。

Web アプリケーションは様々な言語で開発されている。例えば、CGI アプリケーションの多くは Perl スクリプトで記述されているし、性能が求められるバックエンドのアプリケーションは C、C++で記述されていることが多い。しかし、今後の Web アプリケーションは、Java[3]が中心になっていくと考えられる。クライアント側アプリケーションではあまり普及しなかったが、その生産性の高さやマルチプラットフォーム性から、サーバー側アプリケーションの分野では、サーブレット[4]、JSP[5]、そして Enterprise JavaBeans[6]など、Java は広く用いられている。また、最近特に注目されている、Web サービスの分野でも、XML との親和性の高い Java が有力な開発プラットフォームとして期待されている。こうした現状から、我々は、セキュアな Java アプリケーションの構築技法が今後必要になっていくと考える。

本論文は次のように構成されている。第 2 章では、セキュアな Java アプリケーション構築技法の現状について総括し、問題点を明らかにする。第 3 章では、我々の作成した Java のガイドラインである、Java 版セキュアソースガイドラインおよびべからず集について解説し、第 2 章で述べた問題点を解決できることを示す。第 4 章では、第 2 章で明らかになった問題点に対して、問題点に対応する、我々の提案する監査手法の要件を示し、独自のアクセス権解析手法について解説する。最後に、第 5 章においてまとめを述べる。

2. セキュアな Java アプリケーション構築技法の現状と問題点

まず、セキュアな Java アプリケーション構築技法の現状について総括し、それぞれの特徴、および問題点を明らかにする。ここではセキュアなアプリケーション構築技法を、ガイドラインと監査に分類して説明する。

2.1. ガイドライン

長いプログラミング経験の中で、頻繁に発生するセキュリティホールの典型的な例と、その原因および対策が分かってきた。これらをまとめたものがガイドラインである。

ここでは、一般的なガイドラインの代表例を簡単に評価し、現状のガイドラインに不足しているものを示す。そして Java における既存の各ガイドラインを同様に評価して、要求されるガイドラインの要件を考える。

2.1.1. 一般的なガイドラインの評価

C 言語やシェルスクリプトの歴史は比較的長く、多くのガイドラインが存在し、*Practical UNIX & Internet Security* が有名である[7]。CGI アプリケーションに対する脆弱性の事例も多いため、W3C により *The World Wide Web Security FAQ* が公開されている[8]。また、Linux の普及に伴い、Linux 上でのプログラミング全般におけるガイドラインとして、*Secure Programming for Linux and Unix HOWTO* が公開されている[9]。

これらはそれぞれターゲットごとに絞り込まれている。よって、ターゲットごとの網羅性は高いが、全体を網羅するガイドラインがない。また、ガイドラインごとに対象が異なっている。一般プログラマ向けの FAQ という形式のものと、セキュリティ専門家向けの詳細な解説書という形式のものがある。

よって、網羅性が高く、様々な対象に対応したガイドラインが、現状では不足しているといえる。

2.1.2. Java における既存のガイドライン

ここでは、Java における既存の各ガイドラインを説明し、前述の議論を踏まえて、網羅性の高さ、対象の範囲、という観点から整理する。

2.1.2.1. *SecuringJAVA*: プログラマ向け 12 ヶ条

Cigital 研究所の Gary McGraw と Princeton 大学の Edward W. Felten が、*SecuringJAVA* において提唱した[10]。プログラマ向けのガイドラインとして、12 個の項目に対して、危険性とその対策について解説している。12 個の項目しかないので、網羅性はあまり高くない。プログラマ向けのガイドラインであるので、簡潔な記述で統一されている。以下の項目がある：

1.初期化に依存しない 2.すべてを private に 3.すべてを final に 4.デフォルトアクセスに依存しない 5.ネストしたクラスは使わない 6.コードに署名しない 7.まとめて署名する 8.オブジェクトの複製を禁止 9.シリアライズを禁止 10.デシリアライズを禁止 11.文字列でクラスを比較しない 12.秘密の情報をコードに埋め込まない

2.1.2.2. *Security Code Guidelines*

Sun Microsystems, Inc.が Web サイト上で公開している[11]。簡潔な記述の 20 個程度の項目で構成されている。網羅性は *SecuringJAVA* よりは少し高いといえるが、十分ではない。簡潔な記述が多く、プログラマ向けという対象は明確になっている。以下の項目がある：

- 特権コードに関する注意点 … *特権コードはできるだけ少なくする・汚染された変数（プログラム外部から変更される可能性のある変数）を特権コードで参照してはいけない・特権コードが必要とされる場合の例（システムプロパティの読み込み・ファイルの読み込み等）*
- Java 一般の注意点 … *static フィールドはなるべく使わない・スコープはできるだけ狭くする・setter/getter の推奨・デフォルトアクセスの危険性・不変オブジェクトの推奨・配列の危険性・シリアライズおよびデシリアライズの危険性・native メソッドの危険性・メモリからの情報漏洩*

2.1.2.3. *Inside Java™2: Platform Security*

Java2 のセキュリティ機構設計者である Li Gong による、Java2 セキュリティの解説書である[12]。特権コード、アクセス制御、オブジェクトのセキュリティについて、詳細な解説が記述されている。

2.1.2.4. *Java™2 SDK Documentation*

Java2 SDK に付属するドキュメントである[13]。特権コード、アクセス制御、オブジェクト直列化、ネストしたクラスに関して、機能の詳細な仕様とともに、脆弱性に関する情報が詳細に記述されている。

2.1.3. Java における各ガイドラインの比較評価

Java における各ガイドラインを比較評価して、個々の特徴、欠点を評価する。

ここでは、網羅性の評価として、*1.特権コード 2.デフォルトアクセス 3.オブジェクト直列化 4.ネストしたクラス 5.リバースエンジニアリング 6.一般的な解説* の 6 分野を定義する。また対象として、一般プログラマ向け、セキュリティ専門家向けの、2 つの対象を考える。

表 1 に評価結果を示す：

表 1 各ガイドラインの比較評価

ガイドライン	1	2	3	4	5	6	対象
SecuringJAVA	△	○	○	○	△	◎	一般
Security Code	○	○	○	×	×	◎	一般
Inside Java	◎	×	○	×	×	○	専門
Java2 Doc	◎	×	◎	◎	×	○	専門
要求される ガイドライン	◎	◎	◎	◎	◎	◎	一般･専門

表 1 から、既存のガイドラインには以下の点において不備な部分があることが分かる：

- 網羅性がたりない … ガイドラインごとに対象にする分野に偏りがあり、利用者は様々なガイドラインを参照して、目的の情報を探し出さなければならない。
- 対象とする利用者にあった記述がない … ガイドラインを参照する利用者は、セキュリティの専門家である場合もあれば、一般のプログラマである場合もある。そうした、様々な知識の違いをもつ利用者に対応するようなガイドラインがない。

2.2. 監査

ガイドラインだけでは、アプリケーションがガイドラインに従っているかどうかを必ずしも保証できない。それを保証するのが監査である。ここでは主に機械的な監査技術、監査ツールについて紹介する。そして、これらを比較評価して特長と問題点を明らかにして、要求される監査技術･ツールの要件を考える。

2.2.1. 既存の監査ツール

2.2.1.1. ソースコード監査ツール

ソースコードを解析して、バグが発生する可能性のあるコーディング、例えば C 言語における型の不一致や、Java におけるクラス名と同名のメソッドなどを検出して報告する。脆弱性に関連するものも、若干検出できる。

- 一般向けのツール … Lint 系のツール（LCLint[14]等）、規約チェックツール
- Java 向けのツール … 規約チェックツール（J.Taster[15]、SIMPLIA/JF Kiyacker[16]等）

2.2.1.2. ソースコード脆弱性検出ツール

ソースコードを解析して、脆弱性の可能性のある部分を検出する。

- 一般向けのツール … slint[17]、ITS4[18]（パターンマッチングによる脆弱性関数の検出、TOCTOU の検出）
- Java 向けのツール … jslint[19]（*SecuringJAVA* における 12 ヶ条の規則に対応した監査ツール。1.から 11.の規則に対応する監査を行い、1.から4.および8.から11.の規則に対しては自動的にソースコードの修正を行う）

2.2.1.3. バッファオーバーフロー対策

C 言語によるアプリケーションでは、バッファオーバーフローがもっとも多くみられる脆弱性であることが分かっている。そのため、バッファオーバーフローへの対策が多く提案されている。Java ではバッファオーバーフローによる影響は受けないが、参考としてこれも紹介しておく。

- StackGuard[20]、ProPolice[21] … アクティベーションレコードの不正な上書きを防ぐ、コンパイラへのエクステンション
- LibSafe[22] …バッファオーバーフローの危険のある関数（`getc()`等）を、内部的に境界チェックを行うようにカスタマイズしたライブラリ
- Purify[23] … 実行時にメモリ管理を監視して、境界チェックを行うように、実行コードを変換
- FIST[24] … アプリケーションを仮想環境で様々な入力を与えて実行させて、脆弱性を検出する

2.2.2. 各監査手法の比較評価

既に述べたように、ソースコード監査ツールは、ソフトウェア一般のバグを防ぐものであり、脆弱性に関連するバグのためのツールではない。そのために、大量のエラー報告が発生して、脆弱性に関連する報告の特定が困難である場合が多い。

ソースコード脆弱性検出ツールは、脆弱性の検出に大変有効であるが、しばしば限定した対象の脆弱性しか発見できない傾向がある。特に、Java 向けのツールが不足している。既に述べた jslint では、2.1.2.1 で説明した 12 ヶ条の規則に対応した脆弱性しか検出できない。

3. Java版セキュアソースガイドラインとべからず集

我々は、2.1 であげた問題点を解決するために、Java 版セキュアソースガイドラインと、Java 版セキュアソースべからず集という、2 つの性質の異なるガイドラインを作成した。我々はまず始めに、主に CGI アプリケーションに関する注意点をまとめたガイドラインである、セキュアソースべからず集について紹介する。

そして、我々の作成した Java 版セキュアソースガ

イドラインとべからず集を紹介し、これらが既存ガイドラインの問題点を解決することを示す。

3.1. セキュアソースべからず集

セキュアソースべからず集は、C/C++/Perl 等の脆弱性の原因となるようなコーディングの例をあげた、一問一答形式のガイドラインである。入出力や、ネットワーク、関数と、カテゴリごとに分類されており、各項目は「～べし」「～べからず」という、するべきこと、してはいけないこと、という形で、簡単な解説と対策法を示している。各項目に記述されている解説はそれほど専門的ではなく、一般プログラマにとって理解しやすい内容になっている。

3.2. Java 版セキュアソースガイドラインとべからず集

セキュアソースべからず集の Java 版という位置付けで作成されたガイドラインである。セキュアソースべからず集に対応する、Java 版セキュアソースべからず集と、それに付属する Java 版セキュアソースガイドラインの2つにより構成されている。

Java 版セキュアソースガイドライン

- Java 版セキュアソースべからず集の、参考書的な位置付けで、べからず集の各項目に対応した解説が記述されている。べからず集のように、独立した項目による構成ではなく、カテゴリごとに1つの解説文書という構成になっている。Java のセキュリティに比較的詳しい、セキュリティ専門化が参照することを想定している。
- Java の機能ごとにカテゴリ分けされており、*1.Java2 のセキュリティ*、*2.特権コード*、*3.デフォルトアクセス*、*4.オブジェクト直列化*、*5.ネストしたクラス*、*6.リバースエンジニアリング*、*7.一般的な解説*の全 7 章で構成されている。

Java 版セキュアソースべからず集

- CGI 版セキュアソースべからず集の Java 版という位置付けである。Java の機能ごとに分けられたカテゴリごとに、一問一答形式の項目が設けられている。各項目は「～べし」「～べからず」という形式で、するべきこと、してはいけないことが明確に示されている。各項目には簡単な説明と、問題への対処法が示されており、より詳しい解説を参照できるように、対応する Java 版セキュアソースガイドラインへのポインタも示されている。Java のセキュリティに対する専門的知識の少ない、一般の Java プログラマが参照することを想定している。
- *1.特権コード*、*2.デフォルトアクセス*、*3.オブジェクト直列化*、*4.ネストしたクラス*、*5.リバースエンジニアリング*、*6.一般的な解説*の全 6 章で構成されている。

これらの 2 つのガイドラインは、以下の既存ガイドラインの問題点を解消するものである。

- 網羅性が高い … 既存ガイドラインの内容を 1 つのガイドラインとしてまとめて、さらに不足していると思われる情報を独自に追加しているので、既存ガイドラインに比べて網羅性が高い。Java の主な機能の多くの部分を含んでいる（特権コード、デフォルトアクセス、オブジェクト直列化、ネストしたクラス、リバースエンジニアリング、一般的な解説）。
- 対象とする利用者にあった記述 … 1 つの文書で、必要とする知識の異なるような内容を同時に記述することは難しい。そこで、利用者として想定される対象を、Java のセキュリティに比較的詳しいセキュリティの専門家と、Java のセキュリティについてはあまり知らない一般のプログラマの二種類を定義した。そして、セキュリティの専門家向けに Java 版セキュアソースガイドラインを、一般のプログラマ向けに Java 版セキュアソースべからず集を作成した。さらに、べからず集からガイドラインを参照することもできる。

4. 監査手法の提案

ここでは、前述のガイドラインに沿った監査ツールの要件を提案し、重要な機能の一つである、アクセス権解析技術について解説する。

4.1. 監査手法が満たすべき要件の考察

監査手法が満たすべき要件として、Java 版セキュアソースべからず集におけるカテゴリごとに、以下のような要件が考えられる。

1. 特権コード … 特権コードの検出機能がまず必要である。そして、特権コードの実体は、特定のメソッド呼び出しであるので、特定のメソッド呼び出しの検出機能が求められる。また、メソッド呼び出し引数の外部依存関係を調べるために、データフロー解析も必要になる。そして、特権コードに関連することの多い、アクセス権の解析も行わなければならない。アクセス権解析手法については、4.2 において説明する。
2. デフォルトアクセス … デフォルトアクセスかどうか判定するために、クラスやメソッドやフィールドのスコープを解析する必要がある。また、

デフォルトアクセスに関係するアクセス権を制御する、特定のメソッド呼び出しの検出も必要となる。

3. オブジェクト直列化 … シリアライズ・デシリアライズに関するクラスやメソッドは決められているので、それらの検出が必要になる。また、オブジェクト直列化を禁止するように、特定のメソッドを上書きする機能も必要である。
4. ネストしたクラス … ネストしたクラスの解析のために、クラスの階層構造の解析が必要である。また、ネストしたクラスの仕様では、ネストしたクラスのスコープは、トップレベルクラスのスコープに依存することがあるので、トップレベルクラス全体の依存関係を解析する必要もある。
5. リバースエンジニアリング … このカテゴリは監査が難しい。プログラムに秘密の情報が埋め込まれていたとしても、それを機械的に判定することは非常に困難である。
6. 一般的な注意点 … ここでは、カテゴリ分けされないような、雑多な注意点について述べられている。clone()を禁止するために関係するメソッドを上書きする機能、public static フィールドを検出するためのフィールドスコープの解析、不変オブジェクト検出のための配列変数受け渡しの解析、などが必要である。

4.2. アクセス権解析手法

すべての項目を機械的に監査することは難しいと思われる。そこで、特権コードの監査において必要となるアクセス権解析手法について説明する。

これは、jslint などの脆弱性解析ツールやソースコード監査ツールなどには存在しない機能である。

本手法は Java の持つアクセス制御機能を対象とするものである。アクセス制御機能は、Enterprise JavaBeans 等のサーバーサイドアプリケーションにおいては、認証関連の各技術とともに欠かせないものになっている機能である。

4.2.1. 概要

アプリケーションが必要とするアクセス権は、通常アプリケーションの設計時に決定されるべきものである。しかし、しばしば実装の段階で、設計時に定義されたアクセス権より大きな、もしくは小さなアクセス権を使用するように、誤ってコーディングしてしまうことがある。この場合、実行時の予期しないセキュリティ例外発生や、脆弱性の原因になる可能性がある。

これを防ぐために、実装されたソースコードを解析して必要とするアクセス権を求め、設計時に定義されたアクセス権との矛盾を検出する手法が、ここで説明するアクセス権解析手法である。

4.2.2. アクセス権の解析

Java においては、アクセス権は対応するメソッドの呼び出しと同等である。よって、メソッドの呼び出しを検出することにより、必要とするアクセス権が決定される。以下に例を示す：

```
// “/tmp/hoge”をオープンして
// 入力ストリームを得る
fis = new
  java.io.FileInputStream(“/tmp/hoge”);

    // “/tmp/hoge”の読み込み
    // アクセス権が必要
  -> java.io.FilePermission
                  “/tmp/hoge”, “read”;
```

メソッドの呼び出しからアクセス権を求める際には、メソッドの呼び出しと必要なアクセス権が対応した表を参照するので、表のデータを更新することにより、新たなメソッド呼び出しとアクセス権の定義を拡張することができる。

4.2.3. アクセス権の比較

アクセス権を解析した後は、設計時に定義されたアクセス権と比較して、矛盾がないかどうか調べる。前節の例で、設計時に定義されたアクセス権が、

```
java.io.FilePermission
                “/tmp/foo”, “read”;
```

であったとすると、下記のように、実装されたアクセス権の方が、あらかじめ定義されたアクセス権と矛盾するので、警告が発せられる。

```
java.io.FilePermission
                “/tmp/hoge”, “read”;

    java.io.FilePermission
                 “/tmp/foo”, “read”;
  // “/tmp/hoge”が”/tmp/foo”に矛盾する
```

アクセス権の比較についても、アクセス権ごとに比較する規則が表として定義されているので、表のデータを更新することにより、新たなアクセス権の比較方法の定義を拡張することができる。

5. まとめ

本論文では、セキュアな Java アプリケーション構築技法の現状について、ガイドラインと監査ツールという観点から整理し、各技法を比較評価した。網羅性・対象の範囲の広さという点において、既存のガイドラインは十分とはいえない。また、監査ツールについても、Java 向けのツールの機能が不足している。

我々は、Java の既存ガイドラインの問題点を解決

するために、Java 版セキュアソースガイドラインとべからず集を作成した。このガイドラインは、既存ガイドラインを総括し、不足している情報を追加しているため、網羅性が高い。また、セキュリティ専門家向けに詳しく記述されたガイドラインと、一般プログラマ向けに簡潔に一問一答形式で記述されたべからず集を別々に作成することにより、より広い対象の利用者が参照することができる。

また、Java の既存の監査ツールに不足している機能を補うものとして、前述の Java 版セキュアソースべからず集に沿った監査ツールの要件を考察した。そして、特権コードの監査において必要となる、アクセス権解析手法について説明した。Enterprise JavaBeans 等のサーバーサイドアプリケーションにおいて、我々のアクセス権解析手法は特に有効であると考える。

今後の課題としては、提案した監査ツールの要件の、実現可能性の検討があげられる。要件によっては、機械的な方法で実現することが難しい場合があり、人為的な方法と組み合わせることも必要である。実際に監査ツールを実装するために、機械的にできる方法を明らかにしなければなない。将来的には、Enterprise JavaBeans、JSP、サーブレットといった、サーバーサイドアプリケーションに特化したガイドライン・監査ツールの実装について検討していく予定である。

参考文献

[1] "CERT/CC Statistics 1988-2001," http://www.cert.org/stats/cert_stats.html, CERT/CC.

[2] http://www.securityfocus.com/, SecurityFocus.

[3] James Gosling, et al, "The Java™ Language Specification Second Edition," Sun Microsystems, Inc., 2000.

[4] "Java™ Servlet 2.3 Specification," Sun Microsystems, Inc., 2001/8/13.

[5] "JavaServer Pages™ Specification Version1.2," Sun Microsystems, Inc., 2001/8/27.

[6] "Enterprise JavaBeans™ Specification, Version 2.0," Sun Microsystems, Inc., 2001/8/14.

[7] Simson Garfinkel, Gene Spafford, "Practical UNIX & Internet Security, 2nd Edition," O'Reilly & Associates, Inc., 1996/4.

[8] Lincoln D. Stein, John N. Stewart, "The World Wide Web Security FAQ," W3C, 2001/9/12.

[9] David A. Wheeler, "Secure Programming for Linux and Unix HOWTO," http://www.linux.org/docs/ldp/howto/Secure-Programs-HOWTO/index.html, 2001.

[10] Gary McGraw, Edward W. Felten, "Securing JAVA™," Wiley Computer Publishing, 1999.

[11] "Security Code Guidelines," http://java.sun.com/security/seccodeguide.html, Sun Microsystems, Inc., 2000/2/2.

[12] Li Gong, "Inside Java™2 Platform Security," Addison Wesley, 1999.

[13] "Java™ 2 SDK, Standard Edition Documentation Version 1.4.0," http://java.sun.com/j2se/1.4/docs/index.html, Sun Microsystems, Inc., 2001.

[14] "LCLint Home Page," http://lclint.cs.virginia.edu/.

[15] "Java コード解析システム J.Taster," http://www.toyo.co.jp/ss/jtaster/index.html, Toyo Corporation.

[16] "SIMPLIA/JF Kiyacker," http://software.fujitsu.com/jp/simplia/syoukai/jf-kiyacker_pc.html, Fujitsu Limited.

[17] mudge, "The slint web page," http://www.l0pht.com/slint.html.

[18] John Viega, et al, "ITS4: A Static Vulnerability Scanner for C and C++ Code," Proc. Ann. Computer Security Applications Conf. 2000, 2000/12.

[19] John Viega, et al, "Statically Scanning Java Code: Finding Security Vulnerabilities," IEEE Software 17(5) 2000.

[20] Crispin Cowan, et al, "StackGuard: Automatic Adaptive Detection and Prevention of Buffer Overflow Attacks," In 7th USENIX Security Conference, 1998/1.

[21] Hiroaki Etoh, "ProPolice", http://www.trl.ibm.com/projects/security/ssp/, IBM Research, 2001/7/28.

[22] Arash Baratloo, et al, "Transparent Run-Time Defense Against Stack Smashing Attacks," In Proceedings of the USENIX Annual Technical Conference, 2000/6.

[23] "Purify: Fast Detection of Memory Leaks and Access Errors," http://www.rational.com/products/whitepapers/319.jsp, Rational Software Corporation, 2001.

[24] A. Ghosh, et al, "An Automated Approach for Identifying Potential Vulnerabilities in Software," Proceedings of the IEEE Symposium on Security and Privacy, 1998/5.