Java を搭載する携帯端末機器におけるコンピュータウイルスの危険性に関する調査・検討結果報告書

平成 13 年 2 月

情報処理振興事業協会

目次

1 はじめに

本報告書は、Java 実行環境を搭載する携帯端末機器において、現在インターネット上で存在が確認されている不正 Java アプリケーション(以下、Java ウイルスと呼ぶ)の動作実験を行い、各種携帯端末機器における Java ウイルスの危険性を検証し、その結果を報告するものである。

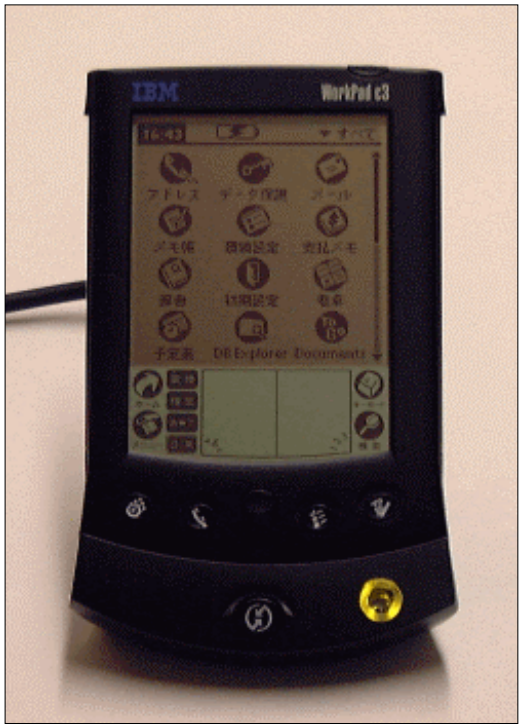
実験対象とする Java ウイルスは以下の 5 種類である。

- Homer
- Hijacker
- Attacker
- StrangeBrew
- Bean Hive

携帯端末機器としては、OS の異なる代表的な下記の 3 種類を選定した。

- Windows CE マシン: NEC 社製 Mobile Gear II MC/R730F、Windows CE Version 2.11, Handheld PC Edition Version 3.01、RAM 32MB

- PalmOS マシン: IBM 社製 WorkPad c3、Palm OS Version 3.5 日本語版、RAM 8MB



- Java (i アプリ) 対応携帯電話: 富士通社製 デジタルムーバ F503i HYPER、RAM 200KB[1]

[1] 1 つの i アプリの最大サイズは 10KB まで。

2 背景と目的

近年、携帯端末機器の普及が急激に拡大しているが、その多くに Java 実行環境が提供され、Java アプリケーションが実行可能となってきている。これらの Java 対応端末機器上で共通に動作する Java ウイルスが出現した場合は、爆発的に感染被害が拡大することが予想されるため、適切な対策を構築することが必要不可欠である。

Microsoft Windows 95/98 等や UNIX では、インターネット上で公開されている不正 Java プログラムが動作し被害を受ける可能性があることが確認されているが、Java アプリケーションを実行可能な携帯端末機器でもその被害が懸念される。特に、代表的な携帯端末機器である Windows CE マシンや PalmOS マシン用に Java 実行環境が提供され、また、Java アプリケーションを実行可能な携帯電話も発売されているが、それらを用いた場合の不正 Java プログラムの動作は明らかにされておらず、どのような影響が出るかも不明である。

本調査は、Java 実行環境を搭載するまたは搭載可能な携帯端末機器における不正 Java プログラムの動作内容を調査し、その動作確認やその被害が及ぶ範囲を明らかにするものであり、その結果を、それら携帯端末機器に関連したウイルス対策を構築するための基礎資料とすることを目的としている。

3 Windows CE マシンでの感染実験

実験に用いた Windows CE マシンは NEC 社製の Mobile Gear II であり、その OS は Windows CE Version 2.11, Handheld PC Edition Version 3.01 である。このマシンには Java 実行環境は入っていないが、Sun Microsystems 社が Web ページで公開している Personal Java Runtime Environment[2] for Windows CE 2.11 Version 1.0 をインストールして Java アプリケーションを実行可能とした[3]。なお、Personal Java では普通にコンパイルした Java アプリケーションファイル(class ファイル)をそのまま利用可能である。

実験を行う Java ウイルスは Homer、Hijacker、Attacker、StrangeBrew、Bean Hive の 5 種類とする。ただし、ソースファイルの状態で公開されていた Homer.java、Hijacker.java、Attacker.java の 3 つは、パーソナルコンピュータ(PC)上の JDK 1.2 でコンパイルした。これらのファイルを携帯端末に転送するために、PC 側に ActiveSync Version 3.1 をインストールし、両者をシリアルケーブルで接続して通信を行った。

以上により、Windows CE の Personal Java Runtime Environment(以下、PJRE と呼ぶ)上で上記ウイルスの動作実験を行う。

実験手順

1. PJRE を Windows CE マシンにインストールする。
2. Java ウイルスファイルを ActiveSync により Windows CE マシンの任意のフォルダに転送する。
3. Windows CE マシン上で Java ウイルスファイルをダブルクリックして実行(図 1)し、その挙動を確認する。

図1: Java アプリケーションの起動画面(Personal Java)

[2] Personal Java は携帯型情報端末向けのサブセット版 Java 実行環境である。

[3] Windows CE 上の"pjava.exe"が Java 仮想マシンとなって Java アプリケーションを実行する。

3.1 実験結果

ここでは、各 Java ウイルスの本来の働きと、Windows CE マシン上での実験の結果を述べる。

3.1.1 Homer

Homer.java は不正な UNIX シェルスクリプト[4]を生成する、わずか 20 行足らずの Java プログラムである。Homer は、ユーザのホームディレクトリ上に".homer.sh"[5]というファイルを作成し、実行可能な属性を設定し、シェルプログラム"/bin/sh"でそれを実行させようとする。".homer.sh"は Homer 内に文字列定数として定義されている UNIX シェルスクリプトであり、UNIX システムにおいて他のシェルスクリプトに感染する機能を持っている。これを UNIX 上で実行すると、"Java is safe, and UNIX viruses do not exist."というメッセージを表示し、ホームディレクトリ上のすべてのシェルスクリプトファイルを再帰的に検索し、それが感染済みかどうかを検査し、未感染であれば自分自身(.homer.sh)のコードをそのシェルスクリプトの末尾に追加(感染)する。

Windows CE マシン上で Homer.class を実行すると、Java コンソール[6]に Unsupported Operation 例外のメッセージが表示され、プログラムが異常終了した(

)。ただし、シェルスクリプト".homer.sh"は PJRE をインストールしたディレクトリに生成された[7]。異常終了したのは、シェルスクリプトに実行可能な属性を設定したりそれを実行したりするための Runtime オブジェクトの exec メソッドが PJRE ではサポートされていないためである。なお、生成された".homer.sh"は、通常の Windows CE ではダブルクリックしたとしても実行できないため**被害はない**。しかしながら、ファイルを生成する機能は何らかの攻撃に悪用できそうであり、注意する必要があるだろう。

---

[4] UNIX シェル上で動作する簡易プログラム。MS-DOS でのバッチファイルに相当する。

[5] ファイル名の先頭を.(ピリオド)にすると、ls コマンドに-a オプションを付けないと表示されない隠しファイルとなる。

[6] Java アプリケーションの実行時に標準出力装置に出力される各種情報が表示される。この場合は例外メッセージが出力された。

[7] Windows CE にはホームディレクトリという概念はない。

図2: Homer の実行画面

3.1.2 Hijacker

Hijacker.java は Sun Microsystems 社の Java コンパイラ"javac"を攻撃する。javac コマンドはシェルスクリプトであり、クラスを探す際に、まずユーザのクラスパス"CLASSPATH"に設定されたディレクトリを探し、それから Java ホームディレクトリ(以下"$J_HOME"と記す)の"$J_HOME/classes"を探し、最後に"$J_HOME/lib/classes.zip"から探す。Hijacker はこのクラス探索手順を悪用している。つまり、ユーザの CLASSPATH に"classes.zip"が設定されていなければ、javac は classes.zip を使用する前に$J_HOME/classes で Java クラスを探す。Hijacker はこの事実を利用して、不正な Java コンパイラメインクラス"$J_Home/classes/sun/tools/javac/Main.class"を生成し、コンパイラの動作をハイジャックするのである。ハイジャックされたコンパイラは実行するたびに"Your Java compiler has been hijacked!"というメッセージを表示し、コンパイルされるクラスファイルには"Hijacked!"という文字列が追加される。

Windows CE マシン上で Hijacker.class を実行すると、PJRE をインストールしたディレクトリに"classes¥sun¥tools¥javac¥Main.class"を生成(図 3)し、正常終了した。ただし、Windows CE にインストールされているのは Java の実行環境のみであるため、このような**Java コンパイラへの攻撃は無意味で被害はない**。

図3: 生成された不正 Main.class

3.1.3 Attacker

Attacker.java は、他の Java クラスファイルを改ざんし、その機能を変更する Java プログラムである。改ざんの対象となるクラスファイルは Attacker 内で指定されている"Beginner.class"であり、このクラスファイルの finally ブロック[8]の末尾に、プログラムの先頭まで制御を返す"goto"命令を挿入する。対象となる Beginner.java は下記のプログラムである。Beginner は存在しないファイル"none"を開こうとして発生する入出力例外

```
import java.io.*;
class Beginner {
        public static void main(String[] argv) {
                try {
                        FileInputStream inner = new FileInputStream("none");
                }
                catch (IOException ioe) {System.out.println("Oops!");}
                finally {System.out.println("Whew!");}
        }
}
```

"IOException"をキャッチし、エラーメッセージ"Oops!"を表示し、最終的にはメッセージ"Whew!"を表示して終了する。Attacker は、Java のバイトコードベリファイアによるコードの検証をパスするように、コードの長さなどの情報を調整している。

Windows CE マシン上の"IPA"ディレクトリに Beginner.class を置いて Attacker.class を実行すると正しく動作しなかったが、Beginner.class をルートディレクトリ[9]に置いて実行した場合は上記の通り動作し、ファイルの改ざんが成功した(図 4, 図 5)。置き場所によって動作が異なる理由は、Attacker はディレクトリを指定せずに Beginner.class にアクセスしようとするが、PC の Windows にあるようなカレントディレクトリという概念が Windows CE には無いようで、ディレクトリを指定しなかった場合はルートディレクトリが対象になるためであろう。

Attacker によって改ざんされた Beginner.class は問題なく実行されるが、それは本来の動作とは異なり、"Oops!"と"Whew!"というメッセージの表示を無限に繰り返してしまう[10](図 6)。このように、Windows CE 環境では、ファイル名を指定して File オブジェクトを生成する際の**デフォルトのディレクトリがカレントディレクトリではなくルートディレクトリになる点以外は、PC の Windows と同等の機能を持つ**ことがわかった。

---

[8] 例外発生の有無に関わらず try ブロックの実行後に実行されるルーチン。

[9] Mobile Gear II の場合は"ﾏｲ ﾊﾝﾄﾞﾍﾙﾄﾞ PC"という名前のディレクトリ。

[10] これを停止させるには、タスクマネージャから「タスク終了」を選択するとよい。

図4: 元の Beginner.class

図5: 改ざんされた Beginner.class

図6: 改ざんされた Beginner.class の暴走

3.1.4 StrangeBrew

StrangeBrew は他の Java アプリケーション(class ファイル)に感染する Java ウイルスである。これは、感染時に対象ファイルのメソッド領域に格納されているプログラムコードを改変してウイルスコードを挿入し、また、ウイルス自身が持つ定数をコンスタントプールに追加する。これによって、対象ファイルの中にはウイルスコードが実行可能な形で含まれることになり、対象ファイルを実行した場合はそのウイルスが活動を開始し、さらに感染・拡散することになる。

Windows CE マシン上での StrangeBrew ウイルスの感染動作の実験結果を以下に示す。

図 7は StrangeBrew ウイルス"Virus.class"の実行前の状態であり、これを実行したことによって図 8の状態になった。これらの図から、正常だったクラスファイル"Welcome.class"、"Welcome$1.class"、"WelcomeCanvas.class"の 3 つが感染し、ファイルサイズが増加したことが確認できる。"Welcome.class"の場合、元のサイズが 2.12KB であり、感染後は約 4KB 増加して 6.01KB(正確には 6,161B)になった。このサイズ 6,161 は 101 の倍数であるが、このウイルスはこれを感染済みであることを表す目印としている。したがって、これらのファイルは次回の感染対象からは外されることになる。

図7: StrangeBrew 実行前

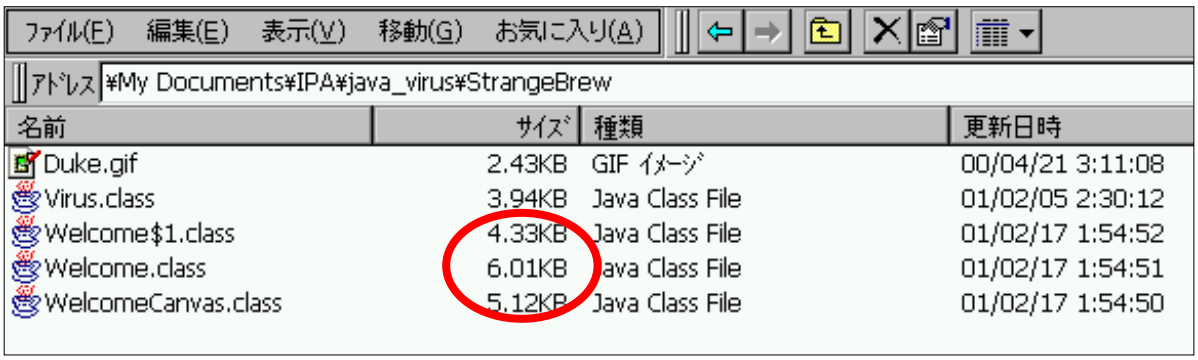


図8: StrangeBrew 実行後

このディレクトリに 562B の"HelloWorld.class"を追加し(図 9)、感染した"Welcome.class"を起動した(図 10)。Welcome アプリケーションは PJRE が正しくインストールできているというメッセージを表示するものだが、感染している状態でも正しく動作し、また、ウイルス

の感染機能が働いて"HelloWorld.class"に 2 次感染した。

図9: HelloWorld.class を追加

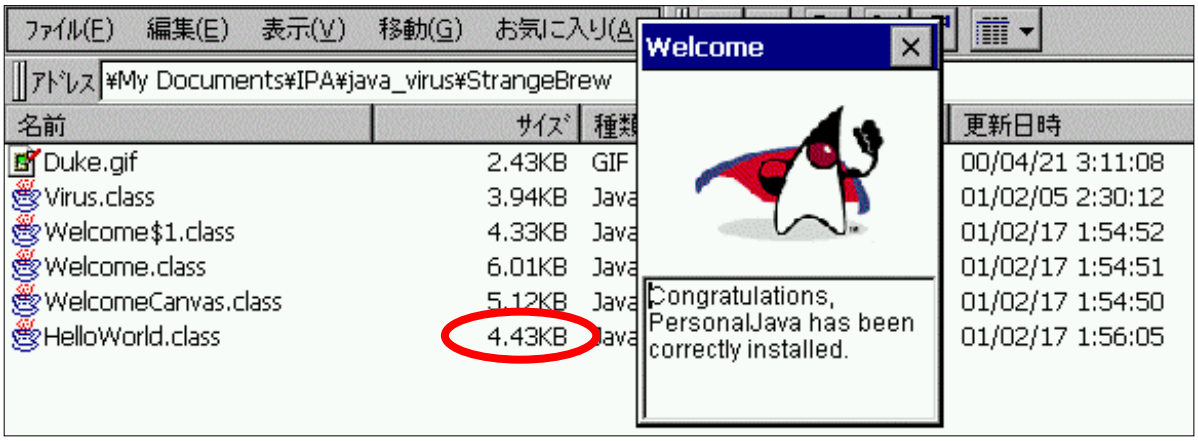


図10: 感染している Welcome.class を実行

以上の通り、StrangeBrew ウイルスは本来の動作を実行でき、他のファイルへの感染が成功した。すなわち、**Windows CE はファイル感染型の Java ウイルスの標的となり得、その危険性は Windows 95/98 等と同等である**と言える。Windows CE マシンで Java アプリケーションを扱う場合は、ウイルスチェックを十分に行う必要がある。

なお、StrangeBrew は Attacker と違い、ディレクトリを指定して File オブジェクトを生成しているため、正しくカレントディレクトリのファイルに感染する。

3.1.5 Bean Hive

Bean Hive は複数の Java プログラムで構成されており、ウイルスプログラムをネットワークからダウンロードして感染動作を行う、ネットワーク専用の新しいウイルスである。このウイルスは、StrangeBrew と同様の手法を用いて感染対象ファイルに不正なクラスローダを追加書き込みする。感染したクラスが実行されると不正なクラスローダが呼び出され、ネットワーク経由でウイルスプログラム本体をダウンロードし、それが実行されて新たな感染動作を行う。

Windows CE マシン上での Bean Hive の動作実験を以下に示す。

図 11、図 12は、正常なクラスファイル"Test.class"が、"BeanHive.class"の実行によって感染したことを示している。ファイルサイズが 281B から 1,382B 増加し、1,683B になった。

図11: BeanHive 実行前

図12: BeanHive 実行後

感染した Test.class を実行すると、本来はあるサイトに接続してウイルス本体をダウンロードしようとするのだが、現在はそのサイトが閉鎖されているためか何も起きなかった。しかしながら、プログラム上は動作するような仕組みになっているため、今後同様の Java ウイルスが出現した場合は、インターネットに接続中の Windows CE マシン上でも動作する可能性は十分あるだろう。

3.2 機能の検討

ここでは、以上の実験結果をまとめ、Windows CE 上での Java ウイルスの動作に関する考察を行う。

3.2.1 PJRE の機能

Personal Java では Java の一部の機能が実装されていない。Personal Java が完全にサポートしているのは下記のパッケージのみである。

- java.awt
- java.io
- java.math
- java.rmi
- java.sql
- java.util.zip

ウイルスに使われ得るファイルアクセス関係(java.io)はすべて動作することが保証されている。すなわち、ファイルの改ざん・破壊・感染などの活動を行うウイルスは動作してしまう。

なお、上記以外のパッケージも一部は機能するが、完全にはサポートされていない。java.lang.Runtime クラスの exec メソッド、すなわち、外部プログラムの実行機能が動作しなかったのはこのためである。

3.2.2 ファイルの生成

Windows 95/98 等の Java と同じ機能を持っている。

Homer や Hijacker のように FileOutputStream クラスを利用すれば、ファイル名を指定してファイルを新規作成することができる。また、既に存在するファイル名を指定した場合はそのファイルに上書きすることも可能である。ファイルへの書き込みは write メソッドで行う。

このように、Windows CE の Java も**不正なファイルを新規に作成**し、または**既存のファイルを破壊できる**という危険性を持っている。

3.2.3 外部プログラムの実行

Homer は Java 仮想マシンの Runtime オブジェクトの exec メソッドで外部プログラムを実行しようとするが、exec メソッドはサポートされていない、という例外メッセージが出て動作しなかった。

3.2.4 ファイルの改ざん

Windows 95/98 等の Java と同じ機能を持っている。

Attacker のように、改ざん対象のファイルが書き込み可能かどうかを File オブジェクトの canWrite メソッドで検査したり、RandomAccessFile クラスを利用して読み書きモードでフ

ァイルを開き、改ざんしたりすることが可能である。

3.2.5 **ウイルスの感染**

Windows 95/98 等の Java と同じ機能を持っている。

StrangeBrew のように、RandomAccessFile クラスを使ってファイルを読み書きすることによって、他の Java クラスファイルに感染するウイルス Java アプリケーションが作成可能である。前述のように、Windows CE マシンの Java ウイルスに対する安全性は Windows 95/98 等と全く変わらないため、**Windows CE マシンでも PC と同等の対策が必要である**。

3.3 対策手法

Windows 95/98 等を搭載する PC と同様に、Windows CE マシンでもファイルの内容を検査するパターンマッチング法が有効である。もちろん、PC と Windows CE マシンの双方で検査することが望ましい。

なお、Windows CE マシンにはさまざまなタイプの CPU が用いられているため、ワクチンソフトウェアを作成するときは CPU 別にコンパイルし、各機種のユーザに間違いのないよう提供しなければならない。しかしながら、このような機種間の互換性を実現したのが Java であるから、ワクチンを Java で作成するという手もある。Java ウイルスが動作する環境であれば、そのワクチンも確実に動作するからである。

4 PalmOS マシンでの感染実験

実験に用いた PalmOS マシンは IBM 社製の WorkPad c3 であり、その OS は PalmOS Version 3.5 である。このマシンには Java 実行環境は入っていないが、Sun Microsystems 社が Web ページで公開している Java 2 Platform Micro Edition の Connected, Limited Device Configuration Version 1.0 (J2ME CLDC)に含まれる仮想マシンプログラムファイル "KVM.prc"、"KVMutil.prc"の 2 つをインストールして Java アプリケーションを実行可能とした。ただし、PalmOS 用の KVM では普通にコンパイルした Java アプリケーションファイル(class ファイル)をそのまま利用することはできないため、PalmOS 用 J2ME CLDC に含まれる変換ツールを用いて class ファイルから PalmOS アプリケーションファイル(PRC ファイル)へ変換した。

実験を行う Java ウイルスは Homer、Hijacker、Attacker、StrangeBrew、Bean Hive の 5 種類とする。ただし、ソースファイルの状態で公開されていた Homer.java、Hijacker.java、Attacker.java の 3 つは、PC 上の JDK 1.2 で J2ME CLDC と PalmOS 用ライブラリを利用してコンパイルした。これらのファイルを PRC ファイルに変換した後、シリアルケーブルで接続した携帯端末に HotSync Version 3.5 で転送した。

以上により、PalmOS の KVM 上で上記ウイルスの動作実験を行う。

PRC ファイルへの変換方法

Java アプリケーションが 1 つの class ファイルであれば、まず、J2ME CLDC に含まれるプリベリファイア"preverify.exe"を使い、class ファイルの安全性をチェックし、PRC ファイルを生成するためのベリファイ済み class ファイルを出力する[11]。

次に、PalmOS 用 J2ME CLDC に含まれる Java アプリケーション"MakePalmApp.class"でベリファイ済み class ファイルを PRC ファイルに変換する。なお、プリベリファイアを通していない class ファイルを PRC ファイルに変換すると、Palm 上で実行したときに「Error verifying class クラス名」と表示され、実行できない。

もし、Java アプリケーションが複数の class ファイルで構成されているなら、それらをすべて"preverify.exe"でベリファイした後、1 つの jar ファイルにまとめてから、それを"MakePalmApp.class"で PRC ファイルに変換する。

実験手順

1. J2ME CLDC の KVM を PalmOS マシンにインストールする。
2. Java ウイルスの PRC ファイルを HotSync により PalmOS マシンに転送する。
3. PalmOS マシン上で Java ウイルス PRC ファイルを実行(図 13)し、その挙動を確認す

[11] J2ME CLDC 環境では、Java アプリケーション実行時の KVM の負担を軽くするために、プリベリファイアによってスタックマップ情報を生成し class ファイルに追加している。これにより、Palm 環境の少ないメモリ容量でも実行時の検証ができるのである。

る。

図13: Java アプリケーションの起動画面(JVM)

4.1 実験結果

ここでは、各 Java ウイルスの本来の働きの概要と、PalmOS マシン上での実験の結果を述べる。

4.1.1 Homer

Homer は、ユーザのホームディレクトリ上に".homer.sh"というファイルを作成し、実行可能な属性を設定し、シェルプログラム"/bin/sh"でそれを実行させようとする。

PalmOS マシン上で Homer.prc を実行しようとするとエラーが発生して実行できなかった(図 14)。

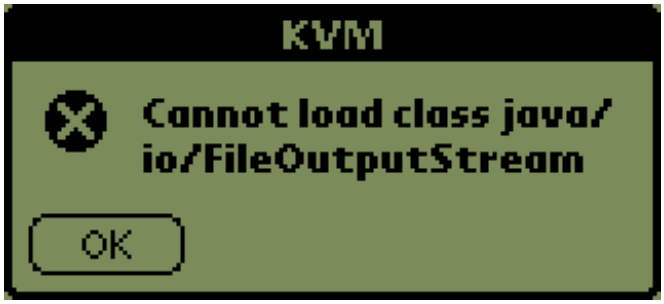


図14: Homer の実行エラー

PalmOS 用の KVM は java.io パッケージを完全にはサポートしておらず、Homer が利用している java.io.FileOutputStream クラスは存在しないため、実行できないのである。すなわち、PalmOS 上では **Homer ウイルスの被害はない**。なお、クラスのロード中にエラーが発生して終了したため確認できなかったが、外部プログラムの実行もできないようである。

4.1.2 Hijacker

Hijacker は不正な Java コンパイラメインクラス"$J_Home/classes/sun/tools/javac/Main.class"を生成し、コンパイラの動作をハイジャックする。

実験のために Hijacker.class を PRC ファイルへ変換する際、プリベリファイ時に下記のエラーが表示され、ベリファイ済み PRC ファイルは出力されなかった。

```
Error preverifying class Hijacker
        VERIFIER ERROR Hijacker.main([Ljava/lang/String;]V:
Cannot find class java/io/FileOutputStream
```

念のためプリベリファイ前の class ファイルを直接 PRC ファイルに変換して転送を試みたが、PRC ファイルに変換する時点で「WARNING: Resource "Clas 6613" may be too large. Hotsync may fail.」という警告が表示された。生成された PRC ファイルを HotSync で Palm に転送しようとしたが、やはり警告通り「本体の無効なファイルが削除されました。」というメッセージが出て送ることはできなかった。ちなみに、"Clas 6613"とは PRC ファイル中のリソースの 1 つであり、この場合は Hijacker.class のバイトコードが含まれている。

なお、プリベリファイ時の「Cannot find class java/io/FileOutputStream」というメッセージは矛盾しているように思われる。同じ FileOutputStream を用いている前述の Homer では、プリベリファイ時にはエラーが出なかった。恐らく FileOutputStream の使い方の違いが原因であろうが、それを修正したとしても、サイズの問題で転送できないのであるから、**Hijacker の被害はない**。

4.1.3 Attacker

Attacker は、ファイル"Beginner.class"を改ざんし、その機能を変更する Java プログラムであるが、ここでは Palm 用に、対象ファイルを"Beginner.prc"に変更して実験を行った。

PalmOS マシン上で Attacker.prc を実行しようとするとエラーが表示され、終了する(図15)。当然、Beginner.prc は変更されなかった。

図15: Attacker の実行エラー

Homer と同様、File クラスを利用する **Attacker ウイルスによる被害はない**。ちなみに、Beginner.prc も FileInputStream クラスを利用しているため実行できない(図 16)。PalmOS の KVM では、ファイル操作全般が実行できない。

図16: Beginner の実行エラー

4.1.4 StrangeBrew

StrangeBrew は他の class ファイルに感染する Java ウイルスである。

PalmOS マシン上で StrangeBrew ウイルスを起動しようとすると、やはりエラーが表示され終了した(図 17)。Attacker らと同様、ファイル操作を行う **StrangeBrew ウイルスによる被害はない**。

図17: StrangeBrew の実行エラー

4.1.5 Bean Hive

Bean Hive は複数の Java プログラムで構成されており、StrangeBrew と同様の手法を用いて感染対象ファイルに不正なクラスローダを追加書き込みするウイルスである。

PalmOS マシン上で Bean Hive を起動しようとしても、やはりエラーが表示され、終了する(図 18)。Attacker らと同様、ファイル操作を行う **Bean Hive ウイルスによる被害はない**。

図18: Bean Hive の実行エラー

4.2 機能の検討

ここでは、以上の実験結果をまとめ、PalmOS 上での Java ウイルスの動作に関する考察を行う。

4.2.1 J2ME CLDC / KVM の機能

J2ME CLDC は、メモリ等に制約のある小型の装置のために設計された Java プラットフォームであり、その仮想マシン KVM には Java の一部の機能しか実装されていない。サポートされていない機能のうち、主なものを以下に示す。

- Java Native Interface (JNI)
  これは OS ネイティブなプログラム(関数)を呼び出すための機能であるが、これがサポートされていないということは、Java と PalmOS 用プログラムを組み合わせた不正プログラムは実現不可能であることを意味する。
- ユーザ定義のクラスローダ
  これはクラスを読み込む方法を独自に作成できる機能であるが、これがサポートされていないということは、Bean Hive のようにクラスを外部のネットワークから優先的に読み込むような不正クラスローダを含むプログラムは実現不可能であることを意味する。

また、CLDC のライブラリには java.io パッケージのうち下記のものしか含まれていない。

java.io.InputStream
java.io.OutputStream
java.io.ByteArrayInputStream
java.io.ByteArrayOutputStream
java.io.DataInput (interface)
java.io.DataOutput (interface)
java.io.DataInputStream
java.io.DataOutputStream
java.io.Reader
java.io.Writer
java.io.InputStreamReader
java.io.OutputStreamWriter
java.io.PrintStream

すなわち、ファイルを操作する File, FileOutputStream, RandomAccessFile などのクラスが存在しないため、ファイルを改ざんしたり感染したりする不正プログラムは動作できないことになる。

他にも、java.lang.Runtime クラスは存在するが、外部プログラムを実行する exec メソッ

ド、すなわち、外部プログラムの実行関数がサポートされておらず、この機能も動作しないことが保証される。

4.2.2 ファイルの生成

上述の通り、Homer や Hijacker のような FileOutputStream クラスを利用する不正プログラムは、その機能がライブラリに存在しないため、実行前のロード・ベリファイ時にエラーとなって終了する。

このように、PalmOS の Java は**不正なファイルの作成や既存のファイルの破壊はできない**。

4.2.3 外部プログラムの実行

Homer は Java 仮想マシンの Runtime オブジェクトの exec メソッドで外部プログラムを実行しようとするが、**KVM では exec メソッドはサポートされておらず、実行できない**。

4.2.4 ファイルの改ざん

上述の通り、Attacker のような File オブジェクトを利用するプログラムも実行不可能である。

4.2.5 ウイルスの感染

同様に、StrangeBrew のような RandomAccessFile クラスを使う Java ウイルスも実行不可能である。

以上のように、機能の貧弱な携帯端末に合わせて Java の実行環境が構築されているため、Java 仮想マシンのセキュリティは非常に高く、**PalmOS マシンでは Java ウイルスの心配は全くないと言える**。

4.3 対策手法

PalmOS では、メモリ等に制約のある小型の装置のために設計された J2ME CLDC が利用可能であるが、これには Java の一部の機能しか実装されておらず、特にファイル操作機能やプログラム実行機能がサポートされていないため、ウイルス等の不正プログラムは PalmOS マシンに悪影響を与えることができない。

J2ME CLDC の KVM が仕様通りに実装されていれば、既存の Java ウイルスに対しても、これから出現する不正な Java プログラムに対しても、PalmOS マシンのユーザは何の対策も必要ではない。

5 Java 対応携帯電話機での感染実験

実験に用いた Java 対応携帯電話機は富士通社製のデジタルムーバ F503i HYPER である。この携帯電話は i アプリに対応、すなわち、i モード対応の Java 実行環境を搭載しており、Java 2 Platform Micro Edition の Connected, Limited Device Configuration (J2ME CLDC)をサポートしている。以下、Java 対応携帯電話機を i アプリ携帯と呼ぶ。

実験を行う Java ウイルスは Homer、Hijacker、Attacker、StrangeBrew、Bean Hive の 5 種類とする。ただし、ソースファイルの状態で公開されていた Homer.java、Hijacker.java、Attacker.java の 3 つは、PC 上の JDK 1.3 で、J2ME CLDC と i アプリ用のライブラリを利用してコンパイルした。これらのファイルを Web サーバ上に置き、i アプリ携帯にダウンロードすることにした。

以上により、i アプリ携帯上で上記ウイルスの動作実験を行う。

i アプリ作成方法

まず、アプリケーションを構成するすべての class ファイルについて、J2ME CLDC に含まれるプリベリファイア"preverify.exe"を使って class ファイルの安全性をチェックし、ベリファイ済み class ファイルを出力する。そして、それらを 1 つの jar ファイルにまとめる。

次に、アプリケーション記述ファイル(ADF、拡張子は*.jam)を作成する。これはアプリケーション情報を記述したテキストファイルであり、Web サイトから i アプリをダウンロードする際、まず ADF の情報に基づいて i アプリ携帯上にダウンロード・実行可能かどうかの判定を行うためのものである。ADF の書式を以下に示す。

```
AppName = アプリケーション名
PackageURL = jar ファイルの URL
AppSize = jar ファイルのサイズ(最大 10,240 バイト)
AppClass = アプリケーションの起動に使われるメインクラス名
LastModified = アプリケーションの最終更新日時
```

また、i アプリをダウンロードするための HTML ファイルを作成する。HTML ファイルでは、ADF(*.jam)の URL を OBJECT タグで定義し、A タグの ijam 属性で ADF とリンクさせ、ダウンロードができるようにする。なお、i アプリ携帯では ijam 属性でリンクされ、A タグの href 属性は i アプリ携帯以外の環境のためのリンク先[12]を指定する。HTML ファイルの書式を次に示す。

[12] 例えば、i アプリ携帯でアクセスするよう指示するページなど。

```
<OBJECT declare
  id="i アプリ ID"
  data=" ADF(jam ファイル)の URL"
  type="application/x-jam">
</OBJECT>
<A ijam="# i アプリ ID"
  href="Dummy.html"> i アプリ名</A>
```

最後に、作成した i アプリ(*.jar, *.jam)とそれにアクセスするための HTML ファイルを Web ページ上にアップロードする。

実験手順

1. Java ウイルスを i アプリに変換し、Web サーバ上に置く。
2. i アプリ携帯で上記ページにアクセスし、i アプリをダウンロードする。
3. i アプリ携帯で Java ウイルス i アプリを実行(図 19)し、その挙動を確認する。

図19: i アプリの起動画面(F503i)

5.1 実験結果

ここでは、各 Java ウイルスの本来の働きの概要と、Java 対応携帯電話(i アプリ携帯)上での実験の結果を述べる。

5.1.1 Homer

Homer は、ユーザのホームディレクトリ上に".homer.sh"というファイルを作成し、実行可能な属性を設定し、シェルプログラム"/bin/sh"でそれを実行させようとする。

実験のために Homer.java で i アプリを作成しようとすると、コンパイルの時点でエラーが発生して class ファイルを生成できなかった(図 20)。これは、i アプリ用の開発環境の CLDC ライブラリおよび i アプリ用ライブラリが、i アプリでサポートされているクラスファイル以外を含まないためである。

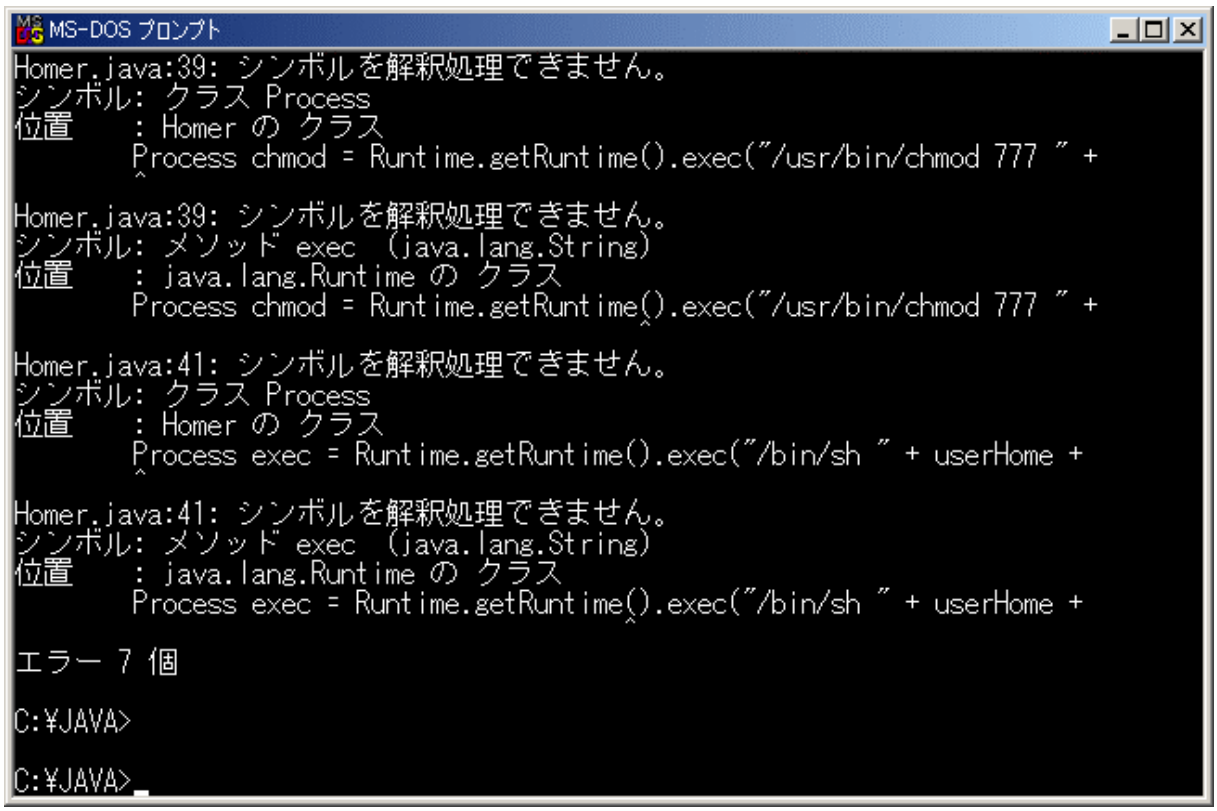


図20: Homer のコンパイルエラー

発生したコンパイルエラーはすべて「シンボルを解釈処理できません。」というものであり、そのシンボルとは、クラス FileOutputStream 2 個、String クラスのメソッド getBytes 1 個、クラス Process 2 個、Runtime クラスのメソッド exec 2 個、合計 7 個であった。なお、getBytes は CLDC にも含まれているが、Homer が利用している引数 4 個の getBytes はサポートされていない。

次に、これを Java 2 Platform Standard Edition (J2SE)の該当するクラスファイル群を用いて強制的にコンパイルしてみた。この場合はプリベリファイ次に下記のエラーが出たが、

J2SE のクラス群を用いて強制的にプリベリファイを行うことはできる。

```
Error preverifying class Homer
    java/lang/NoClassDefFoundError: java/lang/Object
```

しかしながら、i アプリでは、最初に起動されるクラスは com.nttdocomo.ui.IApplication クラスから派生させることになっており、また、IApplication クラスの start メソッドを再定義しなければならない。このため、J2SE 環境でコンパイルした通常の Java アプリケーションは、そのままでは i アプリとして実行することはできず、エラーが表示され終了した(図 21)。

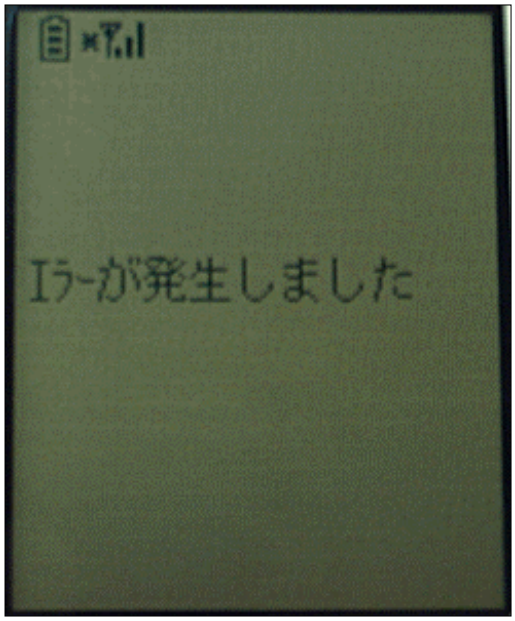


図21: i アプリでない場合の実行エラー

また、仮に IApplication クラスから派生させ start メソッドの多重定義を行ったとしても、i アプリの実行環境は J2ME CLDC の KVM であるため、PalmOS マシンと同様に、FileOutputStream クラスを利用できず、外部プログラムの実行もできない(図 21と同じエラーが発生した)。したがって、**Homer ウイルスの被害はない**。

5.1.2 Hijacker

Hijacker は不正な Java コンパイラメインクラス"$J_Home/classes/sun/tools/javac/Main.class"を生成し、コンパイラの動作をハイジャックする。

実験のために Hijacker.java をコンパイルすると、Homer と同様のエラーが表示され、class ファイルは出力されなかった。発生したエラーはすべて「シンボルを解釈処理できません。」というものであり、そのシンボルとは、クラス File 4 個、クラス FileOutputStream 1 個、合計 5 個であった。

これを、J2SE を用いて強制的にコンパイルし強制的にプリベリファイを行っても、Homer と同様、i アプリとして実行することはできない。

それ以前に、i アプリでは最大 10KB というサイズ制限があり、Hijacker を J2SE でコンパイルすると 79.6KB と巨大であるため、例え ADF の AppSize を 10KB 以下に書き換えたとしてもダウンロード中に失敗してしまうのである(図 22)。したがって、**i アプリ携帯では Hijacker の被害はあり得ない**。

図22: 10KB を超える i アプリのダウンロードエラー

5.1.3 Attacker

Attacker は、ファイル"Beginner.class"を改ざんし、その機能を変更する Java プログラムであるが、ここでは i アプリ用に、対象ファイルを"Beginner.jar"に変更して実験を行った。

実験のために Attacker.java をコンパイルすると、Homer と同様のエラーが表示され、class ファイルは出力されなかった。発生したエラーはすべて「シンボルを解釈処理できません。」というものであり、そのシンボルとは、クラス File 2 個、クラス RandomAccessFile 2 個、合計 4 個であった。

Homer と同様、File クラスを利用する **Attacker ウイルスによる被害はない**。ちなみに、Beginner も FileInputStream クラスを利用しているためコンパイルできない。i アプリ携帯では、ファイル操作全般が実行できない。

5.1.4 StrangeBrew

StrangeBrew は他の class ファイルに感染する Java ウイルスである。

これはコンパイルされた状態で出回っているため、class ファイルを強制プリベリファイし、jar ファイルに変換して実験したが、やはり i アプリ携帯上でエラーが表示され終了した。Attacker らと同様、ファイル操作を行う **StrangeBrew ウイルスによる被害はない**。

5.1.5 Bean Hive

Bean Hive は複数の Java プログラムで構成されており、StrangeBrew と同様の手法を用いて感染対象ファイルに不正なクラスローダを追加書き込みするウイルスである。

実験のために Bean Hive ウイルスを構成するすべてのクラスをコンパイルすると、Homer と同様のエラーが表示され、class ファイルは出力されなかった。発生したエラーはすべて「シンボルを解釈処理できません。」というものであり、そのシンボルとは、クラス File 5 個、クラス RandomAccessFile 8 個、合計 13 個であった。

Bean Hive を強制的にコンパイル・プリベリファイして i アプリ携帯上で起動しようとしても、やはりエラーが表示され終了する。Attacker らと同様、ファイル操作を行う **Bean Hive ウイルスによる被害はない**。

5.2 機能の検討

ここでは、以上の実験結果をまとめ、i アプリ携帯上での Java ウイルスの動作に関する考察を行う。

5.2.1 J2ME CLDC / KVM の機能

i アプリ携帯では、PalmOS についての検討と同様のことが言える。すなわち、i アプリ携帯も制限のある J2ME CLDC に準拠しており、その仮想マシンには Java の一部の機能しか実装されていない。

CLDC のライブラリには、ファイル操作関連のクラスが含まれていないため、ファイルを操作する File, FileOutputStream, RandomAccessFile などを利用してファイルを改ざんしたり感染したりする不正プログラムは動作できないことになる。

他にも、java.lang.Runtime クラスは存在するが、外部プログラムを実行する exec メソッド、すなわち、外部プログラムの実行関数がサポートされておらず、この機能も動作しないことが保証される。

5.2.2 ファイルの生成

上述の通り、Homer や Hijacker のような FileOutputStream クラスを利用する不正プログラムは、その機能がライブラリに存在しないため、実行前[13]にエラーとなって終了する。

このように、i アプリ携帯の Java は**不正なファイルの作成や既存のファイルの破壊はできない**。

5.2.3 外部プログラムの実行

Homer は Java 仮想マシンの Runtime オブジェクトの exec メソッドで外部プログラムを実行しようとするが、i アプリ携帯に搭載されている KVM も PalmOS の KVM と同様、**exec メソッドはサポートされておらず、実行できない**。

5.2.4 ファイルの改ざん

上述の通り、Attacker のような File オブジェクトを利用するプログラムも実行不可能である。

5.2.5 ウイルスの感染

同様に、StrangeBrew のような RandomAccessFile クラスを使う Java ウイルスも実行不可能である。

以上のように、J2ME CLDC / KVM は機能の貧弱な携帯端末に合わせて構築された Java 実行環境であるため、Java 仮想マシンのセキュリティは非常に高く、**i アプリ携帯では Java ウイルスの心配は全くないと言える**。

---

[13] 特に表示はないが、恐らく実行直前のベリファイ時であろう。

5.3 対策手法

i アプリ携帯では、メモリ等に制約のある小型の装置のために設計された J2ME CLDC / KVM が搭載されているが、これには Java の一部の機能しか実装されておらず、特にファイル操作機能やプログラム実行機能がサポートされていないため、ウイルス等の不正プログラムは i アプリ携帯に悪影響を与えることができない。

i アプリ携帯の J2ME CLDC / KVM が仕様通りに実装されていれば、既存の Java ウイルスに対しても、これから出現する不正な Java プログラムに対しても、i アプリ携帯のユーザは何の対策も必要ではない。

6 まとめ

携帯端末が急速に普及しつつあり、また、それらの端末に Java 実行環境が搭載されるようになり、UNIX や Windows 95/98 等で問題となっている Java ウイルスにそれら携帯端末のユーザが遭遇する場合も増えてくると考えられる。

本報告書では、Java 実行環境を搭載可能な Windows CE マシン、PalmOS マシン、そして Java 実行環境を搭載している i アプリ対応携帯電話における不正 Java プログラムの動作を実験によって確認した。

その結果、Windows CE マシンでは Windows 95/98 等と同等の機能が提供されており、現存する Java ウイルスがほとんどそのまま実行可能であることを確認した。これは非常に危険なことであるが、幸いなことに Windows 95/98 等で培った Java ウイルス対策技術がほとんどそのまま適用可能であるため、Windows CE 用のワクチンを開発し、従来通りの対策を実行することで、Java ウイルスの被害を防ぐことは十分可能であろう。

また、J2ME CLDC を採用している PalmOS および i アプリ携帯については、CLDC の KVM 自体が高いセキュリティ機能を持っているため、PalmOS マシンや i アプリ携帯自体に危険が及ぶことはないことを確認した。

ただし、i アプリ携帯については、携帯電話機メーカー各社が独自に i アプリの仕様を拡張することが考えられる。すなわち、利便性を追求するあまり、携帯電話機のアドレス帳等のメモリへのアクセス手段を提供するなど、セキュリティホールになるような機能が追加されるのではないかという懸念はまだ残る。これについては、今後も各社の開発の動向を注目していかなければならないだろう。