УДК 004.05

студент И.Ю. Маковецкий
ГВУЗ «Национальный Горный Университет»

БЕЗОПАСНОСТЬ НА ПЛАТФОРМЕ JAVA

Java, как язык программирования, чтобы привлечь внимание, должен был ввести определенные нововведения, по сравнению с другими языками. В этой статье рассматривается, что именно из себя представляют простейшие встроенные в Java стандарты безопасности, привносимые компилятором и Java-машиной.

Компилятором предъявляются определенные требования к программному коду, прежде чем он будет преобразован в байт-код. В разных компиляторах требования разные, но в целом все они имеют определенное сходство.

Собственно, проверка соответствия загружаемых классов требованиям Java Virtual Machine сводится к верификации байт-кода (англ. Bytecode Verification). Эта проверка препятствует возникновению ошибок, таких, как например несоответствие переменных типам данных, переполнение стэков и др. Многие пользователи отзываются о ней, как о ненужной мере предосторожности, так как любой программный код при компиляции проходит через эту проверку, но загружаемые из сети классы или библиотеки классов не всегда ее проходили, не говоря уже о вредоносном коде.

Верификация байт-кода отключается следующей командой в командной строке JVM:

```
java -Xverify:none ApplicationName
```

Где `ApplicationName` – название программы.

Проверим это на примере простейшей программы:

```
public static void main(String args[])
{int x = 1;}
```

После компиляции, если открыть скомпилированное редактором байт-кода, мы увидим следующее:

```
iconst_1
istore_1
return
```

Заменим «`istore_1`» на «`fstore_1`». При попытке выполнить эту программу, JVM выдает следующую ошибку:

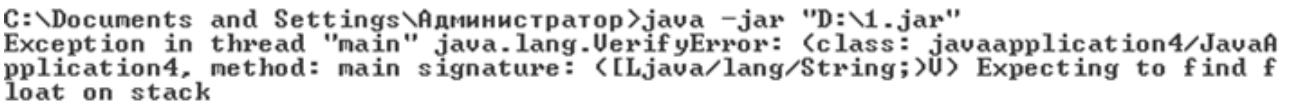

Рис. 1.

После отключения верификации байт-кода JVM выдала следующее сообщение, после чего прекратила свою работу:

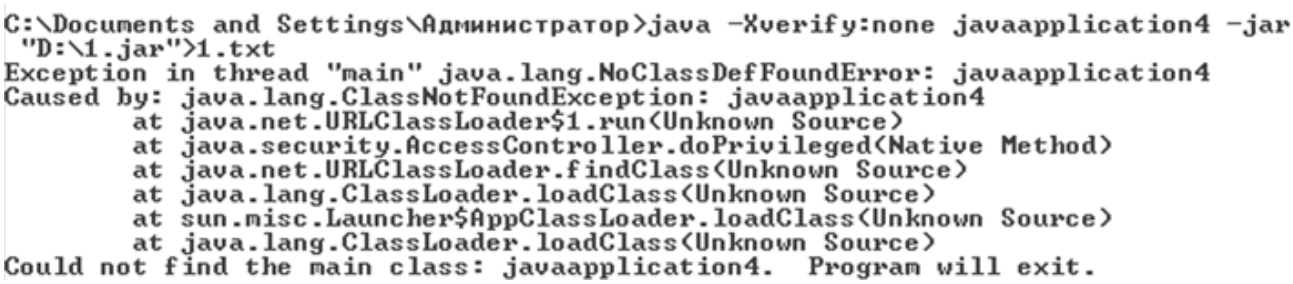

Рис. 2.

Что и подводит нас к выводу о том, что верификация байт-кода – необходимая мера предосторожности, и далеко не бессмысленна.

Повышение интереса к Java в основном обусловлено отсутствием надобности следить за выделяемой памятью – функция Java-машины часто называемая «сборщиком мусора» – автоматическое удаление динамических переменных, когда они больше не нужны, что, как следствие – делает любой написанный код компактней и проще для понимания. Однако это лишь часть функций элемента безопасности Java, называемого «Автоматическое Управление Памятью» (англ. «Automatic memory management»), и обладающего несколько большим потенциалом.

Также, язык Java является статическим языком – это означает, что проверка типов данных (англ. Type Checking) происходит не во время выполнения кода, как в динамических языках программирования, а во время компиляции – таким образом уменьшая количество ошибок в приложениях. Хотя в 7-ой версии Java добавлена поддержка для динамических языков (проект «Мультиязыковая Виртуальная Машина»).

Проверка типов данных осуществляется по определенному набору правил, называемых «Строгая типизация» (англ. Strong Data Typing) – обязательная

проверка соответствий сигнатур функций (методов) и операторов – типам данных, которые там используются.

Безопасная загрузка классов (англ. «Secure class loading») – пожалуй, важнейший элемент безопасности в Java. С увеличением количества доступных готовых библиотек классов для Java, не только в самой Java – но и по всей сети Интернет – некоторые люди стали делать публично-доступными замаскированные библиотеки классов, несущие кроме всего прочего вредоносный код – именно поэтому в Java получили реализацию такие элементы как «песочница» (англ. «Sandbox refresher») и усовершенствованный загрузчик классов (с возможностью создания своих объектов-загрузчиков).

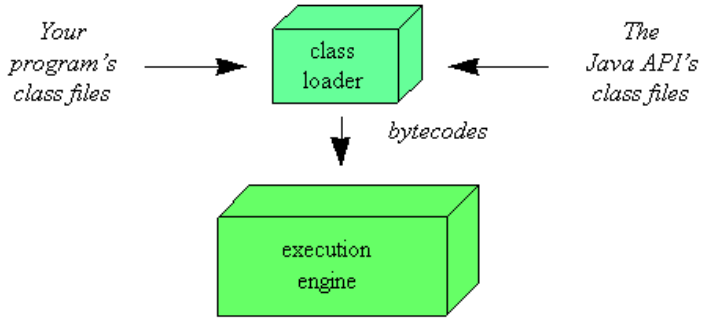

Рис 3.

Песочница представляет собой настраиваемую среду для запуска ненадежных программ и кода, в которой действуют определенные правила:

1. Считывать или записывать данные на локальный диск.
2. Устанавливать любые соединения.
3. Создавать новые процессы.
4. Загружать динамические библиотеки и напрямую вызывать native-методы.

Перечень литературы:

1. http://www.roseindia.net/java/master-java/StrongTyping.shtml
2. http://forum.vingrad.ru/forum/topic-323185.html
3. http://programmersgoodies.com/what-can-you-throw-in-java
4. http://www.javaworld.com/javaworld/jw-09-1997/jw-09-hood.html
5. http://en.wikipedia.org/wiki/Java_version_history