Apéndice B.
Algunos aspectos de seguridad en Java

Los mecanismos de seguridad de Java actúan en cuatro niveles diferentes de la arquitectura del sistema. Primero, el mismo lenguaje Java se diseño para ser seguro; el compilador Java asegura que el código fuente no le estas reglas de seguridad. Segundo, todos los bytecodes que corren a tiempo de ejecución se protege para asegurar que también obedezcan estas reglas, esta capa lo protege contra un compilador alterado que produzca código que viole estas reglas de seguridad. Tercero, el cargador de clases asegura que las clases no violen el espacio de nombres o las restricciones de acceso cuando se cargan en el sistema. Por último, la seguridad específica API previene a los applets contra la realización de algunas acciones peligrosas. Esta última capa depende de las garantías de seguridad e integridad de las otras tres capas. A continuación se examina cada una de estas capas en turno [Lemay y Perkins 1996].

El Lenguaje y el Compilador

La mayoría de los demás lenguajes estilo C cuenta con medios para controlar el acceso a ”objetos”, pero también cuentan con formas ”de crear” el acceso a objetos (o partes de objetos), por lo general al user (mar) los apuntadores. Esto introduce dos violaciones de seguridad fatales a cualquier sistema construido con estos lenguajes. Una es que ningún objeto puede protegerse contra una modificacion externa, duplicación o engaño (otros pretender ser ese objeto). Otra es que un lenguaje con apuntadores poderosos tal vez tengan errores serios que comprometan la seguridad. Java elimina estas amenazas de un golpe al suprimir por completo los apuntadores de lenguaje.

Verificación de los Bytecodes

Al tiempo de ejecución Java se obtienen la mayor parte de los bytecodes de la red; entonces nunca se diferenciara si esos bytecodes fueron generados por un compilador ”digno de confianza”. En consecuencia, deberá verificar que cumple con todos los requerimientos de seguridad.

Antes de ejecutar cualquier bytecode, el tiempo de ejecución, Java los sujeta a una serie de rigurosas pruebas que varían en complejidad, desde simples revisiones de formato hasta ejecutar un probador de teoremas, pare tener la certeza de que siguen las reglas. Estas pruebas verifican que:

los bytecodes no creen apuntadores, se violen restricciones de acceso, se de acceso a objetos como algo más de lo que son (Input Streams siempre se utilizarán como lo que son y no como otra cosa), se hagan llamadas a métodos con valves de argumentos o tipos inadecuados o se permita el sobreflujo en la pila.

Información de Tipo y Requerimientos Extra

Desde el punto de vista conceptual, antes y después de que se ejecute cada bytecode, cada casilla en la pila y cada variable local cuentan con algún tipo. Esta serie de información de tipo (todas las casillas y variables locales) se llama el *estado de tipo* del ambiente de ejecución. Un requisito importante del estado de tipo Java es que debe ser determinable en forma estática por medio de inducción, esto es, antes de que algún código de programa se ejecute. Como resultado, mientras los sistema a tiempo de ejecución leen los bytecodes, se requiere que cada uno tenga la siguiente propiedad inductiva: dado solo el estado de tipo antes de la ejecución del bytecode, este tendrá que ser determinado por completo después de la ejecución.

Otro requisito del tiempo de ejecución Java es que cuando un conjunto de byetcodes toman más de una ruta de acceso para llegar al mismo punto, sodas estas rutas deben llegar exactamente con el mismo estado de tipo. Este es un requisito estricto e implica, por ejemplo, que los compiladores no generen bytecodes que carguen todos los elementos de un arreglo en la pila.

El Verificador

Los bytecodes se revisan de acuerdo con estos requerimientos, con el uso de la información adicional de tipo en un archivo .class, por una parte del sistema a tiempo de ejecución llamada *verificador*. Este examina cada bytecode en turno con la construcción del estado de tipo completo, y verifica que todos los tipos de parámetros, argumentos y resultados sean correctos. Además, el verificador actúa como un guardián de su ambiente a tiempo de ejecución; deja entrar solo a bytecodes adecuados.

Cuando los bytecodes pasan el verificador, esta garantizado que no causarán ningún sobreflujo o bajo flujo en la pila de operandos; no se usarán tipos de parámetros, argumentos o de retorno en forma incorrecta; no se convertirán en forma ilegal datos de un tipo a otro (por ejemplo, de enteros a apuntadores); ni tampoco se tendrá acceso a algún campo de objeto de manera ilegal (este es, el verificador supervise que las reglas para public, private, package y protected se obedezcan).

Algunos otros aspectos de seguridad en Java pueden encontrarse en [Fuenfrocken 1998], [Dean et al. 1996], [Dean 1996], [Grim y Bershad

1996], [Wallach et al. 1997] y [Wetherall 1995]

Pérez Lezama, C. V. 1998. **Agentes Móviles en Bibliotecas Digitales**. Tesis Maestría. Ciencias con Especialidad en Ingeniería en Sistemas Computacionales. Departamento de Ingeniería en Sistemas Computacionales, Escuela de Ingeniería, Universidad de las Américas Puebla. Mayo.