

Java7u11 sebezhetőség

Adam Gowdiak jelezte pénteken a [full disclosure](#) levelez listán, hogy az Oracle által frissen kiadott Java 7u11 verzió – amely javítja az egyik aktívan kihasznált biztonsági hibát – további két biztonsági hibát tartalmaz, amelyekkel ki lehet törni a sandbox-ból, s ezeket a sebezhetéseket is elkezdték kihasználni.

További probléma, hogy az Oracle csak egy hibát javított a friss verzióban, amint erről Esteban Guillardoy ír egy friss blogbejegyzésben:

```
405 /**
406  * Is this method a caller-sensitive method?
407  * I.e., does it call Reflection.getCallerClass or a similar method
408  * to ask about the identity of its caller?
409  */
410 // FIXME: Replace this pattern match by an annotation @sun.reflect.CallerSensitive.
411 static boolean isCallerSensitive(MemberName mem) {
412     assert(mem.isInvokable());
413     Class?> defc = mem.getDeclaringClass();
414     switch (mem.getName()) {
415         case "newInstance":
416             return defc == java.security.AccessController.class;
417         case "getCaller":
418             return defc == sun.misc.Unsafe.class;
419         case "lookup":
420             return defc == java.lang.invoke.MethodHandles.class;
421     case "invoke":
422         return defc == java.lang.reflect.Method.class;
423     case "get":
424     case "getBoolean":
425     case "getBytes":
426     case "getChar":
427     case "getShort":
428     case "getInt":
429     case "getLong":
430     case "getFloat":
431     case "getDouble":
432     case "set":
433     case "setBoolean":
434     case "setByte":
435     case "setChar":
436     case "setShort":
437     case "setInt":
438     case "setLong":
439     case "setFloat":
440     case "setDouble":
441         return defc == java.lang.reflect.Field.class;
442     case "newInstance":
443         if (defc == java.lang.reflect.Constructor.class) return true;
444     }
445     return false;
446 }
```

```
405 /**
406  * Is this method a caller-sensitive method?
407  * I.e., does it call Reflection.getCallerClass or a similar method
408  * to ask about the identity of its caller?
409  */
410 // FIXME: Replace this pattern match by an annotation @sun.reflect.CallerSensitive.
411 static boolean isCallerSensitive(MemberName mem) {
412     assert(mem.isInvokable());
413     Class?> defc = mem.getDeclaringClass();
414     switch (mem.getName()) {
415         case "newInstance":
416             return defc == java.security.AccessController.class;
417         case "getCaller":
418             return defc == sun.misc.Unsafe.class;
419         case "lookup":
420             return defc == java.lang.invoke.MethodHandles.class;
421     case "invoke":
422         return defc == java.lang.invoke.MethodHandles.class;
423     case "findVirtual":
424     case "findConstructor":
425     case "findSpecial":
426     case "findMethod":
427     case "findGetter":
428     case "findStaticGetter":
429     case "findStaticSetter":
430     case "bind":
431     case "newReflect":
432     case "newReflectConstructor":
433     case "newReflectSetter":
434     case "newReflectGetter":
435         return defc == java.lang.invoke.MethodHandles.Lookup.class;
436     case "invoke":
437         return defc == java.lang.reflect.Method.class;
438     case "get":
439     case "getBoolean":
440     case "getBytes":
441     case "getChar":
442     case "getShort":
443     case "getInt":
444     case "getLong":
445     case "getFloat":
446     case "getDouble":
447         return defc == java.lang.reflect.Field.class;
448     case "set":
449     case "setBoolean":
450     case "setByte":
451     case "setChar":
452     case "setShort":
453     case "setInt":
454     case "setLong":
455     case "setFloat":
456     case "setDouble":
457         return defc == java.lang.reflect.Field.class;
458     case "newInstance":
459         if (defc == java.lang.reflect.Constructor.class) return true;
460     }
```

<http://immunityproducts.blogspot.com.ar/2013/01/confirmed-java-only-fixed-one-of-two.html>

Célszerű a Java applet futtatást csak akkor és csak addig engedélyezni, amikor és ameddig feltétlenül szükséges... a legfrissebb Java verzió nem ad elég védelmet, amíg a sandbox mechanizmus tüzetes átvizsgálása meg nem történik az Oracle részéről...