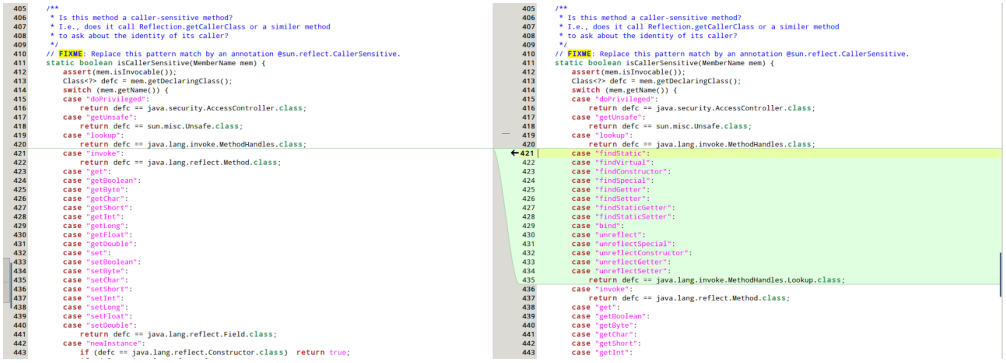
Java7u11 sebezhetség

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 sebezhetségeket is elkezdték kihasználni.

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



```
/**
 * Is this method a caller-sensitive method?
 * I.e., does it call Reflection.getCallerClass or a similer method
 * to ask about the identity of its caller?
 */
// FIXME: Replace this pattern match by an annotation @sun.reflect.CallerSensitive.
static boolean isCallerSensitive(MemberName mem) {
    assert(mem.isInvocable());
    Class<?> defc = mem.getDeclaringClass();
    switch (mem.getName()) {
    case "doPrivileged":
        return defc == java.security.AccessController.class;
    case "getUnsafe":
        return defc == sun.misc.Unsafe.class;
    case "lookup":
        return defc == java.lang.invoke.MethodHandles.class;
    case "invoke":
        return defc == java.lang.reflect.Method.class;
    case "get":
    case "getBoolean":
    case "getByte":
    case "getChar":
    case "getShort":
    case "getInt":
    case "getLong":
    case "getFloat":
    case "getDouble":
    case "set":
    case "setBoolean":
    case "setByte":
    case "setChar":
    case "setShort":
    case "setInt":
    case "setLong":
    case "setFloat":
    case "setDouble":
        return defc == java.lang.reflect.Field.class;
    case "newInstance":
        if (defc == java.lang.reflect.Constructor.class)  return true;
```

```
/**
 * Is this method a caller-sensitive method?
 * I.e., does it call Reflection.getCallerClass or a similer method
 * to ask about the identity of its caller?
 */
// FIXME: Replace this pattern match by an annotation @sun.reflect.CallerSensitive.
static boolean isCallerSensitive(MemberName mem) {
    assert(mem.isInvocable());
    Class<?> defc = mem.getDeclaringClass();
    switch (mem.getName()) {
    case "doPrivileged":
        return defc == java.security.AccessController.class;
    case "getUnsafe":
        return defc == sun.misc.Unsafe.class;
    case "lookup":
        return defc == java.lang.invoke.MethodHandles.class;
    case "findStatic":
    case "findVirtual":
    case "findConstructor":
    case "findSpecial":
    case "findGetter":
    case "findSetter":
    case "findStaticGetter":
    case "findStaticSetter":
    case "bind":
    case "unreflect":
    case "unreflectSpecial":
    case "unreflectConstructor":
    case "unreflectGetter":
    case "unreflectSetter":
        return defc == java.lang.invoke.MethodHandles.Lookup.class;
    case "invoke":
        return defc == java.lang.reflect.Method.class;
    case "get":
    case "getBoolean":
    case "getByte":
    case "getChar":
    case "getShort":
    case "getInt":
```

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

Célszer 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 részérl...