Java 7 sebezhetség

A napokban a blog.fireeye.com oldalon sikerült egy Java 7 sebezhetséget találni, amelyet kihasználva egy Java applet képes kitörni a *sandbox* környezetébl. A hiba legvégén a Java 7 platformban megjelent ClassFinder osztály *resolveClass* metódusát találjuk, amely a sandbox elírásokat megkerülve képes betölteni bármilyen osztályt:

ClassFinder.java

```
public static Class<?> resolveClass(String name, ClassLoader loader) throws ClassNotFoundException {
    Class<?> type = PrimitiveTypeMap.getType(name);
    return (type == null)
            ? findClass(name, loader)
            : type;
}
```

A meghívott *findClass* metódus pedig megkeresi és visszaadja az adott osztályt:

ClassFinder.java

```
public static Class<?> findClass(String name) throws ClassNotFoundException {
    try {
        ClassLoader loader = Thread.currentThread().getContextClassLoader();
        if (loader == null) {
            // can be null in IE (see 6204697)
            loader = ClassLoader.getSystemClassLoader();
        }
        if (loader != null) {
            return Class.forName(name, false, loader);
        }

    } catch (ClassNotFoundException exception) {
        // use current class loader instead
    } catch (SecurityException exception) {
        // use current class loader instead
    }
    return Class.forName(name);
}
```

Mivel az átadott *ClassLoader* példány értéke lehet *null*, ezért ebben az esetben lekérdezésre kerül a *SystemClassLoader*, amely vissza fogja adni a kért osztályt. A hiba kihasználásában sokat nem számít, de az itt látható kód igen "szép":

- a *catch* ágakban el van nyelve a kapott *ClassNotFoundException* vagy *SecurityException* kivétel
- majd ezen hibák esetén a metódus végén a *Class.forName* meghívásra kerül

Ezt a *ClassFinder* osztályt nem tudja egy *sandbox*-ban futó applet meghívni, mivel a *com.sun.beans.finder* csomagban van, amelyet nem lehet elérni a *san dbox* környezetbl, ám az Java 1.4 verzióban megjelen java.beans.Expression osztályon keresztül igen:

Exploit.java

```
new Expression(Class.class, "forName", new Object[] {"sun.awt.SunToolkit"}).invoke();
```

Az Expression.java osztály forrásában azt láthatjuk, hogy meghívja az sosztályának a konstruktorát:

Expression.java

```
@ConstructorProperties({"target", "methodName", "arguments"})
public Expression(Object target, String methodName, Object[] arguments) {
    super(target, methodName, arguments);
}
```

A Statement.java osztályban láthatjuk az *AccessController* ellenrzése alatti invoke hívást:

Statement.java

```
try {
    return AccessController.doPrivileged(
            new PrivilegedExceptionAction<Object>() {
                public Object run() throws Exception {
                    return invokeInternal();
                }
            },
            acc
    );
}
catch (PrivilegedActionException exception) {
    throw exception.getException();
}
```

Amely tartalmaz egy szép külön ágat arra az esetre, ha a *Class.forName()* kerülne meghívásra:

Statement.java

```
// Class.forName() won't load classes outside
// of core from a class inside core. Special
// case this method.
if (target == Class.class && methodName.equals("forName")) {
    return ClassFinder.resolveClass((String)arguments[0], this.loader);
}
```

Így viszont visszaadódik a kért osztály, mindenféle ellenrzés nélkül, mert a *ClassFinder* bizony nem fog dobni *PrivilegedActionException* kivételt... ettl a ponttól már "triviális" a sandbox környezetbl való kitörés, mivel a SunToolkit és a hozzá hasonló a bels használatú osztályok elvileg nem lennének betölthetek a *sandbox* környezetben... 🙂

A Statement.java osztályban is van egy nagyon szép programozástechnikai hiba, amely közel két éve jelent meg a kódbázisban, és azóta is benne fekszik:

```
--- a/src/share/classes/java/beans/Statement.java        Thu Jan 29 15:34:50 2009 +0300
+++ b/src/share/classes/java/beans/Statement.java        Fri Jul 03 16:56:29 2009 +0400
@@ -66,6 +66,7 @@ public class Statement {
     Object target;
     String methodName;
     Object[] arguments;
+    ClassLoader loader;

     /**
      * Creates a new <code>Statement</code> object with a <code>target</code>,
@@ -157,7 +158,7 @@ public class Statement {
         // of core from a class inside core. Special
         // case this method.
         if (target == Class.class && methodName.equals("forName")) {
-            return ClassFinder.resolveClass((String)arguments[0]);
+            return ClassFinder.resolveClass((String)arguments[0], this.loader);
         }
         Class[] argClasses = new Class[arguments.length];
         for(int i = 0; i < arguments.length; i++) {
```

Noha a hibával nincs kapcsolatban, de belekerült a kódbázisba egy *loader* nev példányváltozó, amelyet átadnak paraméterként a *resolveClass* metódus hívásánál, de **sehol nem kap értéket**. Elképeszt! Bármelyik statikus kódanalizáló program hangosan ugat egy ilyen hibára, amely két éve a Java egyik fontos osztályában fekszik... 🙂

Veszélyes?

A külföldi és hazai IT portálok igen kényes sebezhetségként írták le ezt a problémát, de valószínleg nagyobb a füstje, mint a lángja... szakmailag ugyanis nagyon érdekes egy ilyen hibát megtalálni, felfedni és a sebezhetség okát kideríteni.. de a hitelesen aláírt Java appletek képesek a *sandbox* környezeten kívül is futni, és akik vissza szeretnének élni a helyzettel, azoknak nem okozhat nehézséget egy hamis, de érvényes és hiteles aláírás megszerzése... nem hiteles aláírással pedig csak egy felhasználói jóváhagyás kell, a felhasználó pedig jóvá fogja hagyni az applet futását...

...ennek ellenére tartsuk szárazon a puskaport és járjuk nyitott szemmel, de nem feltétlen szükséges a JRE/JDK eltávolítása a gépekrl, ahogy azt több helyen is olvashatjuk, a helyzet kezelését rábízhatjuk a víruskeres szoftverekre, amelyeknek illene blokkolnia egy ilyen támadást, illetve legalább egy – akár false positive – figyelmeztetést küldeni a felhasználó felé, ha egy aláírás nélküli applet szeretne elindulni és benne van egy *Expression* hívás. A legbiztosabb megoldás az, ha letiltjuk a Java plugin futását a böngészben és csak akkor engedélyezzük, ha arra kifejezetten szükségünk van.

Update

Van rá esély, hogy az OpenJDK 6 is érintett a sebezhetségben, mivel ezen Java környezet forrásban úgy tnik, hogy a fenti kódrészletek backportolásra kerültek, ugyanis mind az Oracle oldaláról letölthet OpenJDK 6 update 25 forrása, mind az online források között megtalálható a ClassFinder osztály, amely elvileg a Java 1.7 óta létezik:

(forrás: https://blogs.oracle.com/darcy/entry/openjdk_6_genealogy)

Javítócsomag

Megérkezett a patch a hibával kapcsolatban: http://www.oracle.com/technetwork/topics/security/alert-cve-2012-4681-1835715.html