Security Vulnerability Notice

SE-2012-01-IBM

[Security vulnerabilities in Java SE, Issues 33-49]

DISLAIMER

INFORMATION PROVIDED IN THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW NEITHER SECURITY EXPLORATIONS, ITS LICENSORS OR AFFILIATES, NOR THE COPYRIGHT HOLDERS MAKE ANY REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR THAT THE INFORMATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS, OR OTHER RIGHTS. THERE IS NO WARRANTY BY SECURITY EXPLORATIONS OR BY ANY OTHER PARTY THAT THE INFORMATION CONTAINED IN THE THIS DOCUMENT WILL MEET YOUR REQUIREMENTS OR THAT IT WILL BE ERROR-FREE.

YOU ASSUME ALL RESPONSIBILITY AND RISK FOR THE SELECTION AND USE OF THE INFORMATION TO ACHIEVE YOUR INTENDED RESULTS AND FOR THE INSTALLATION, USE, AND RESULTS OBTAINED FROM IT.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL SECURITY EXPLORATIONS, ITS EMPLOYEES OR LICENSORS OR AFFILIATES BE LIABLE FOR ANY LOST PROFITS, REVENUE, SALES, DATA, OR COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, PROPERTY DAMAGE, PERSONAL INJURY, INTERRUPTION OF BUSINESS, LOSS OF BUSINESS INFORMATION, OR FOR ANY SPECIAL, DIRECT, INDIRECT, INCIDENTAL, ECONOMIC, COVER, PUNITIVE, SPECIAL, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND WHETHER ARISING UNDER CONTRACT, TORT, NEGLIGENCE, OR OTHER THEORY OF LIABILITY ARISING OUT OF THE USE OF OR INABILITY TO USE THE INFORMATION CONTAINED IN THIS DOCUMENT, EVEN IF SECURITY EXPLORATIONS OR ITS LICENSORS OR AFFILIATES ARE ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS.

Security Explorations discovered 17 security issues in the latest version of IBM SDK, Java Technology Edition software [1]. Most of them are caused by the unsafe use of a Reflection API. Since, security checks in use by the aforementioned API rely on a caller's class, proper delegation of the calls from untrusted code may lead to the successful bypass of these checks. This may further lead to the creation of arbitrary class instances from restricted packages as well as to the invocation of arbitrary methods on such objects. As a result, complete Java security sandbox compromise can be usually obtained.

A table below, presents a technical summary of all identified issues:

ISSUE #	TECHNICAL DETAILS	
33	origin	`com.ibm.rmi.util.ProxyUtil` class
	cause	insecure use of `invoke` method of `java.lang.reflect.Method` class
	impact	arbitrary method invocation inside `AccessController`'s `doPrivileged` block
	type	complete security bypass vulnerability
34	origin	`com.ibm.rmi.util.ProxyUtil` class
	cause	insecure use of `invoke` method of `java.lang.reflect.Method` class
	impact	arbitrary method invocation inside `AccessController`'s `doPrivileged` block
	type	complete security bypass vulnerability
35	origin	`com.ibm.xtq.xslt.runtime.extensions.JavaExtensionUtils` class
	cause	insecure use of `invoke` method of `java.lang.reflect.Method` class
	impact	restricted package bypass via arbitrary method invocation
	type	complete security bypass vulnerability
36	origin	`com.ibm.xylem.instructions.StaticMethodInvocationInstruction` class
	cause	insecure use of `invoke` method of `java.lang.reflect.Method` class
	impact	restricted package bypass via arbitrary method invocation
	type	complete security bypass vulnerability
37	origin	`com.ibm.xylem.instructions.JavaMethodInvocationInstruction` class
	cause	insecure use of `invoke` method of `java.lang.reflect.Method` class
	impact	restricted package bypass via arbitrary method invocation
	type	complete security bypass vulnerability
38	origin	`com.ibm.rmi.io.ObjectStreamClass` class
	cause	insecure use of `getDeclaredMethods` method of `java.lang.Class` class
	impact	access to declared methods of arbitrary classes
	type	partial security bypass vulnerability
39	origin	`com.ibm.rmi.io.ObjectStreamClass` class
	cause	insecure use of `setAccessible` method of `java.lang.reflect.AccessibleObject` class
	impact	overriding standard access permissions of Reflection API object instances
	type	partial security bypass vulnerability
40	origin	`com.ibm.lang.management.ManagementUtils` class
	cause	insecure use of `forName` method of `java.lang.Class` class
	impact	access to restricted classes
	type	partial security bypass vulnerability
41	origin	`com.ibm.xylem.interpreter.InterpreterUtilities` class
	cause	insecure use of `getMethods` method of `java.lang.Class` class
	impact	access to methods of restricted classes
	type	partial security bypass vulnerability

42	origin	`com.ibm.xylem.interpreter.InterpreterUtilities` class
	cause	insecure use of `getConstructors` method of `java.lang.Class` class
	impact	access to constructors of restricted classes
	type	partial security bypass vulnerability
43	origin	`com.ibm.rmi.corba.DynamicAny.DynValueCommonImpl` class
	cause	insecure use of `forName` method of `java.lang.Class` class
	impact	access to restricted classes
	type	partial security bypass vulnerability
44	origin	`com.ibm.xtq.xslt.runtime.JavaMethodResolver` class
	cause	insecure use of `getMethods` method of `java.lang.Class` class
	impact	access to methods of restricted classes
	type	partial security bypass vulnerability
45	origin	`com.ibm.xtq.xslt.runtime.JavaMethodResolver` class
	cause	insecure use of `getConstructors` method of `java.lang.Class` class
	impact	access to constructors of restricted classes
	type	partial security bypass vulnerability
46	origin	`com.ibm.rmi.util.ClassCache` class
	cause	insecure use of `forName` method of `java.lang.Class` class
	impact	access to restricted classes
	type	partial security bypass vulnerability
47	origin	`com.ibm.xtq.xslt.translator.XSLTCHelper` class
	cause	insecure use of `getMethods` method of `java.lang.Class` class
	impact	access to methods of restricted classes
	type	partial security bypass vulnerability
48	origin	`com.ibm.xtq.xslt.translator.XSLTCHelper` class
	cause	insecure use of `getConstructors` method of `java.lang.Class` class
	impact	access to constructors of restricted classes
	type	partial security bypass vulnerability
49	origin	`com.ibm.xtq.xslt.jaxp.compiler.TransformerFactoryImpl` class
	cause	insecure use of `defineClass` method of `java.lang.ClassLoder` class
	impact	arbitrary class definition in a privileged classloader namespace
	type	complete security bypass vulnerability

Below, we provide additional comments with respect to the issues presented in the table above:

- Issues 33 and 34 are alone sufficient to achieve a complete compromise of a Java security sandbox by the means of arbitrary methods invocation done from within a `doPrivileged` code block. In our exploit scenario, we invoke `setSecurityManager` method of `java.lang.System` class with a `null` argument. This allows for a disabling of all security checks in a target Java environment.
- Issues 35, 36 and 37 are alone sufficient to achieve a complete compromise of a Java security sandbox by the means of arbitrary methods invocation. Our exploit scenario proceeds as following:
  - First we load an instance of a restricted `com.ibm.oti.util.PriviAction` class with the use of a `forName` method call of `java.lang.Class` class.

- Then we obtain a reference to the private security field of java.lang.SecurityManager class by the means of a getDeclaredField method call of java.lang.Class class.
- Later we obtain a constructor of com.ibm.oti.util.PriviAction class by calling getDeclaredConstructor method of java.lang.Class class. The latter is used for the creation of a PriviAction object instance conducting arbitrary setAccessible method call inside a doPrivileged method block. As an argument to the constructor, a reference to the looked up security field is passed.
- Finally, we make a call to the doPrivileged method of java.security.AccessController class, which results in overriding of access permission checks for the security field. To complete a sandbox bypass, we set the value of this field to null. This has the effect of disabling of all security checks in a target Java environment.

- Issues 38 and 39 need to be combined together to achieve a complete compromise of a Java security sandbox. The exploit scenario makes use of the possibility to override access permission checks for arbitrary methods (getDeclaredFieldImpl method of java.lang.Class class) and fields (security field of java.lang.SecurityManager class).
- Issues 40, 41 and 42 need to be combined together to achieve a complete compromise of a Java security sandbox. Same for issues 43, 44, 45 and 46, 47, 48. Our exploit scenario relies again on the com.ibm.oti.util.PriviAction class, which is used to override access permission checks for the security field of java.lang.SecurityManager class.
- Issue 49 illustrates hazards related to the insecure use of a defineClass method of java.lang.ClassLoader class. In our exploit code we define a specially crafted HelperClass in a privileged ProtectionDomain. This class implements a PrivilegedAction that invokes setSecurityManager method of java.lang.System class with a null argument.

Presented security issues violate many Secure Coding Guidelines for the Java Programming Language [2]. This includes, but is not limited to:

- Guideline 4-4: Limit exposure of ClassLoader instances
- Guideline 4-5: Limit the extensibility of classes and methods
- Guideline 5-1: Validate inputs
- Guideline 9-1: Understand how permissions are checked
- Guideline 9-8: Safely invoke standard APIs that bypass SecurityManager checks depending on the immediate caller's class loader
- Guideline 9-9: Safely invoke standard APIs that perform tasks using the immediate caller's class loader instance
- Guideline 9-10: Be aware of standard APIs that perform Java language access checks against the immediate caller
- Guideline 9-11: Be aware java.lang.reflect.Method.invoke is ignored for checking the immediate caller

Attached to this report, there are 10 Proof of Concept codes that illustrate all of the reported issues. Each of them demonstrates a complete compromise of a Java security sandbox. The codes use the following convention when it comes to the class names:

- VulnX
  Code implementing a security bypass issue number X.
- Exploit
  Code implementing a complete Java security sandbox bypass exploitation.

Our Proof of Concept codes have been successfully tested in a 32-bit Linux OS environment and with the following versions of IBM SDK:

- IBM SDK, Java Technology Edition, Version 6.0 SR11 for Linux (32-bit x86) released on 2012-08-10 (build pxi3260sr11-20120806_01(SR11))
- IBM SDK, Java Technology Edition, Version 7.0 SR1 for Linux (32-bit x86) released on 2012-04-30 (build pxi3270sr1-20120330_01(SR1))

The only exception to the above statement are Issues 38 and 39, which do not seem to be present in SDK, Java Technology Edition, Version 6.0.

REFERENCES

[1] IBM developer kits , `http://www.ibm.com/developerworks/java/jdk/`

[2] Secure Coding Guidelines for the Java Programming Language, Version 4.0, `http://www.oracle.com/technetwork/java/seccodeguide-139067.html`

About Security Explorations

Security Explorations (`http://www.security-explorations.com`) is a security start-up company from Poland, providing various services in the area of security and vulnerability research. The company came to life in a result of a true passion of its founder for breaking security of things and analyzing software for security defects. Adam Gowdiak is the company's founder and its CEO. Adam is an experienced Java Virtual Machine hacker, with over 50 security issues uncovered in the Java technology over the recent years. He is also the hacking contest co-winner and the man who has put Microsoft Windows to its knees (vide MS03-026). He was also the first one to present successful and widespread attack against mobile Java platform in 2004.