Security Vulnerability Notice

SE-2012-01-IBM-5

[Security vulnerabilities in Java SE, Issue 70#2]

DISCLAIMER

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 that Issue 70 [1] reported to IBM in Oct 2013 was improperly fixed. According to the company, the vulnerability was addressed in a release of IBM Java from Nov 2013. Below, technical details of the flawed fix implementation are provided.

Issue 70 had its origin in `com.ibm.rmi.io.SunSerializableFactory` class. It was caused by insecure implementation of a deserialization process of arbitrary classes. More specifically, during this process, a constructor of the first non-serializable superclass was called inside `AccessController`'s `doPrivileged` block. This condition could be successfully exploited to create custom and fully functional `java.lang.ClassLoader` objects [2]. As a result, a complete Java security sandbox escape could be gained.

IBM addressed Issue 70 (CVE-2013-5456) by restricting access to classes from the `com.ibm.rmi.io` package[1]. As a result, the exploit scenario illustrated by our Proof of Concept Code published in Nov 2013[2] was closed. The vulnerable functionality could be however still accessed through some other code paths as illustrated on Fig. 1.

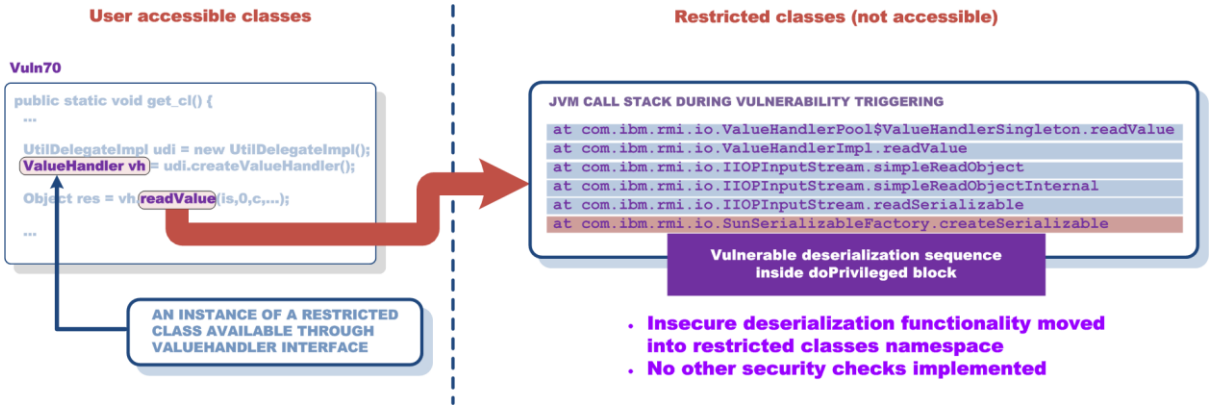

Fig. 1 A code path illustrating Issue 70 patch bypass.

This in particular includes the code path originating in a restricted `com.ibm.rmi.io.ValueHandlerPool.ValueHandlerSingleton` class, which implements `javax.rmi.CORBA.ValueHandler` interface. The instance of this class could be obtained by calling `createValueHandler` method of `com.ibm.CORBA.iiop.UtilDelegateImpl` class. A call to `readValue` method invoked on the acquired `ValueHandler` interface instance can directly lead to the invocation of a vulnerable code sequence in `com.ibm.rmi.io.SunSerializableFactory` class (Issue 70 patch bypass).

Similarly to the broken fix for Issue 67 [3], the actual root cause of Issue 70 hasn't been addressed at all. The constructor of the first non-serializable superclass was still called inside `AccessController`'s `doPrivileged` block. There were no security checks introduced anywhere in the code. The patch primarily addressed the scenario illustrated by the Proof of Concept code. It didn't take into account all code paths that could be used to reach the vulnerable code sequence.

[1] this was accomplished by adding a `com.ibm.rmi.io.` string to the `package.access` definition in a `java.policy` file.
[2] the scenario relying on a `com.ibm.rmi.io.FastPathForCollocated` class.

We implemented a Proof of Concept code that illustrates the impact of the broken fix described above. It has been successfully tested in a 32-bit Linux OS environment and with the following versions of IBM SDK:

- IBM SDK, Java Technology Edition, Version 7.1 for Linux (32-bit x86) released on 2016-01-26 (build pxi3270_27sr3fp30-20160112_01(SR3 FP30))
- IBM SDK, Java Technology Edition, Version 8.0 for Linux (32-bit x86) released on 2016-01-26 (build pxi3280sr2fp10-20160108_01(SR2 FP10))

We verified that, a complete Java security sandbox escape could be achieved with it.

REFERENCES

[1] SE-2012-01-IBM-3, Issues 70-71
http://www.security-explorations.com/materials/SE-2012-01-IBM-3.pdf
[2] Calendar Bug, (Slightly) Random Broken Thoughts, Sami Koivu
http://slightlyrandombrokenthoughts.blogspot.com/2008/12/calendar-bug.html
[3] SE-2012-01-IBM-4, Issue 67#2
http://www.security-explorations.com/materials/SE-2012-01-IBM-4.pdf

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.