Security Vulnerability Notice

SE-2013-01-ORACLE-2

[Security vulnerabilities in Oracle Java Cloud Service, Issues 29-30]

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 two additional security issues in the latest version of Oracle Java Cloud Service. Their technical details are provided below. Both vulnerabilities have been successfully tested in the environment of Oracle Java Cloud ver. 13.2 (US1 Commercial Data Center).

[Issue 29] Java API whitelisting rules bypass for privileged class definition

As a result of Java API whitelisting implementation, in ver.13.2 of Java Cloud Software certain methods are intercepted and checked at runtime in order to prevent arbitrary definition of privileged classes. This in particular concerns `defineClass` method of `java.lang.ClassLoader` class and its subclasses.

Runtime checks imposed in a transformed code make sure that any non-null `ProtectionDomain` argument of a `defineClass` method called from user code is replaced by a value representing an unprivileged domain object. This is implemented by `oracle.cloud.jcs.scanning.impl.extension.LoadingClassValidator` class:

```
        boolean pdomainExists = false;
        int protection_domain_index = findIndexOnArg(method,
java/security/ProtectionDomain);

        if (protection_domain_index > 0) {
            pdomainExists = true;
            ProtectionDomain domain =
(ProtectionDomain)objects[protection_domain_index];
            if (domain != null)
                objects[protection_domain_index] = getUserProtectionDomain();
        }
```

The above implementation was the reason for blocking our original Proof of Concept code for Issue 15 in the environment of Java Cloud Software ver. 13.2.

This vulnerability can be however still exploited and whitelisting rules imposed on Class definition bypassed. What one needs is a custom instance of `java.security.SecureClassLoader` class:

```
  public class MyCL extends SecureClassLoader {
    public PermissionCollection getPermissions(CodeSource cs) {
       Permissions perms=new Permissions();
       perms.add(new AllPermission());
       return perms;
    }

    public Class define_class(String s,byte body[],int i,int j) {
       return super.defineClass(s,body,i,j,new
                          CodeSource(null,(java.security.cert.Certificate[])null));
    }
  }
```

Invocation of `defineClass` method used in a code above will not enforce any changes to the `ProtectionDomain` argument by runtime checks added in a transformed code. The reason is the lack of a domain argument in the method invocation (whitelisting rules pattern match `defineClass` methods containing `ProtectionDomain` argument).

This domain value, although missing from the original call to `defineClass` method can be still provided by an attacker. This can occur with the help of a `SecureClassLoader` class implementation:

```
    protected final Class defineClass(String s, byte abyte0[], int i, int j,
CodeSource codesource) {

        return defineClass(s, abyte0, i, j, getProtectionDomain(codesource));

    }
```

Its `defineClass` method makes use of a `ProtectionDomain` object for a given `CodeSource` filled with the permissions returned by a `getPermissions` method:

```
    private ProtectionDomain getProtectionDomain(CodeSource codesource) {
        if (codesource == null)
           return null;
        }
         ...
        PermissionCollection permissioncollection = getPermissions(codesource);
         protectiondomain = new ProtectionDomain(codesource, permissioncollection,
this, null);
         ...
        }

        return protectiondomain;

    }
```

Custom Class Loader objects that are subclasses of the `SecureClassLoader` class can override `getPermissions` method. Thus, fully privileged `ProtectionDomain` argument can be provided to the `defineClass` method call of `java.lang.ClassLoader` class. As a result, user provided classes can be defined with full privileges and corresponding Java API whitelisting rules bypassed.

The above scenario is illustrated by a Proof of Concept code exploiting Issue 15 that is attached to this report. It was successfully verified in the environment of Java Cloud Software ver. 13.2.

[Issue 30] Java API whitelisting rules bypass through XMLDecoder

As explained in our report from 2012 [1], specially crafted XML file fed as the input to `java.beans.XMLDecoder` object instance can be used to call arbitrary Java methods. This technique can be also successfully used to bypass Java API whitelisting rules of Oracle Java Cloud Software ver. 13.2.

Calling prohibited `setSecurityManager` of `java.lang.System` class can be simply accomplished with the use of the following XML file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<java version="1.4.0" class="java.beans.XMLDecoder">
 <void id="setsmres" class="java.lang.System" method="setSecurityManager"
  <null></null>
 </void>
```

```
 <var idref="setsmres">
 </var>
</java>
```

We verified that Issue 30 can be successfully used to bypass Java API whitelisting rules of Oracle Java Cloud Software ver. 13.2. Our Proof of Concept code illustrating this is attached to this report. It makes use of two XML files: the first one to associate all permissions with user's class Protection Domain, the second one to set Security Manager object to the `NULL` value.

REFERENCES

[1] Security Vulnerabilities in Java SE, technical report, http://www.security-explorations.com/materials/se-2012-01-report.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.