Security Vulnerability Notice

SE-2014-02-ORACLE

[Google App Engine Java security sandbox bypasses, Issue 42]

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 a security vulnerability in Oracle Java SE 7 code. A table below presents its technical summary:

ISSUE #	TECHNICAL DETAILS	
42	origin	HotSpot VM
	cause	improper initialization of non-public interface method slots
	impact	invocation of protected interface methods
	type	partial security bypass vulnerability (Java SE 7) complete security sandbox bypass vulnerability (GAE)

Issue 42 has its origin in `klassItable::initialize_itable_for_interface` method's implementation of Java SE 7 HotSpot VM [1]:

```
methodOop target = klass->uncached_lookup_method(
                                      method_name, method_signature);
...
if (target == NULL || !target->is_public() || target->is_abstract()) {

    // Entry do not resolve. Leave it empty  <---- EMPTY CODE SEQUENCE

} else {
  // Entry did resolve, check loader constraints before initializing
  // if checkconstraints requested
  methodHandle  target_h (THREAD, target); // preserve across gc

  if (checkconstraints) {
   ..,
  }
}

// ime may have moved during GC so recalculate address

                                          <---- INITIALIZATION OF A METHOD SLOT
itableOffsetEntry::method_entry(_klass(), method_table_offset)
                                      [ime_num].initialize(target_h());
```

In the above code, a method slot corresponding to an interface method is always successfully initialized. This happens regardless of the fact that an instance method implementing the target interface method might not be public. As a result, protected instance methods can be successfully used as interface methods. This violates the Java Virtual Machine Language Specification [2]. The description of the *invokeinterface* bytecode instruction states that "if the selected method is not public, *invokeinterface* should throw an *IllegalAccessError*".

The empty code sequence handling invalid (such as non-public) interface methods was changed in Java SE 8, so that a slot of a non-public interface method is filled with a pointer to the method throwing an *IllegalAccessError*.

```
if (target == NULL || !target->is_public() || target->is_abstract()) {
  // Entry does not resolve. Leave it empty for AbstractMethodError.
  if (!(target == NULL) && !target->is_public()) {
    // Stuff an IllegalAccessError throwing method in there instead.
    itableOffsetEntry::method_entry(
```

```
                    _klass(), method_table_offset)[m->itable_index()].
                    initialize(Universe::throw_illegal_access_error());

        }
    }
```

As a result of the above code change, arbitrary invocation of a non-public interface method always triggers an exception.

Attached to this report, there is a Proof of Concept code that illustrates the reported issue in Oracle Java SE environment. It has been successfully tested under JDK 7 Update 80 (build 1.7.0_80-b15).

Issue 42 was also verified to affect a production Google App Engine for Java [3][4] environment patched against all security issues reported to the company from Dec 2014 to Apr 2015 [5] (a complete Java security sandbox escape could be achieved in it).

REFERENCES

[1] jdk7u80-b05 source code
http://hg.openjdk.java.net/jdk7u/jdk7u-dev/hotspot/file/2cd3690f644c/src/share/vm/oops/klassVtable.cpp
[2] The Java Virtual Machine Specification, Java SE 7 Edition
http://docs.oracle.com/javase/specs/jvms/se7/html/
[3] Google App Engine: Platform as a Service
https://cloud.google.com/appengine/docs
[4] Java Runtime Environment
https://cloud.google.com/appengine/docs/java/
[5] SE-2014-02 Details
http://www.security-explorations.com/en/SE-2014-02-details.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.