Security Vulnerability Notice

SE-2014-02-GOOGLE

[Google App Engine Java security sandbox bypasses, Issues 32-34]

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 three additional security vulnerabilities in Google App Engine for Java. A table below, presents their technical summary:

ISSUE #	TECHNICAL DETAILS	
32	origin	`com.google.apphosting.runtime.security.shared.SafeClassDefiner` class
	cause	missing `safeDefineClass(ClassLoader,String,ByteBuffer,CodeSource)` method in `SafeClassDefiner` implementation
	impact	access to security sensitive `defineClass` method handle of `java.security.SecureClassLoader` class
	type	partial GAE security bypass vulnerability
33	origin	`com.google.apphosting.runtime.security.shared.SafeClassDefiner` class
	cause	missing `safeDefineClass(ClassLoader,String,byte[],int,int,CodeSource)` method in `SafeClassDefiner` implementation
	impact	access to security sensitive `defineClass` method handle of `java.security.SecureClassLoader` class
	type	partial GAE security bypass vulnerability
34	origin	`com.google.apphosting.runtime.security.shared.intercept.java.lang.invoke.MethodHandles.Lookup_` class
	cause	missing security check in a `bind` method
	impact	obtaining protected method handles from a non-user class loader namespace
	type	partial GAE security bypass vulnerability

The first two weaknesses are similar to Issues 12 and 14 reported to Google in Dec 2014. Again, `SafeClassDefiner` class does not implement all security relevant methods corresponding to the `defineClass` of `java.lang.ClassLoader` class. However, this time the missing methods have its origin in `java.security.SecureClassLoader` class, which is a subclass of `java.lang.ClassLoader` class.

The implementation of the following two methods is missing from `SafeClassDefiner` class:

```
public static Class safeDefineClass(ClassLoader,String,ByteBuffer,CodeSource)
public static Class safeDefineClass(ClassLoader,String,byte[],int,int,CodeSource)
```

The above methods lie at the core of Issues 32 and 33. As explained in our technical report [1], if a search for a safe replacement method handle cannot find it in the `SafeClassDefiner` class, the method handle lookup operation proceeds against the original class. As a result, arbitrary access to a security sensitive `defineClass` method could be obtained.

Additionally, due to the security patches introduced in GAE around Feb 2015, method handle lookup operations have been limited to public members of system classes (non-user defined classes). That's primarily due to the following security check added to several method lookup operations (`findVirtual`, `findSpecial`, `findStatic`, etc.):

```
if (!RuntimeVerifier.wasClassHierarchyLoadedByUserClassLoader(refc))
    lookup = java.lang.invoke.MethodHandles.publicLookup();
```

The above check enforces the use of a public `Lookup` object for any method handle lookup operation against a non-user defined class. This check is however missing from the implementation of a `bind` method of `MethodHandles.Lookup` mirror class (Issue 34). As a result, method handles to protected members of system classes (such as `java.security.SecureClassLoader`) can be obtained by the means of a `bind` operation. Such method handles can be further invoked without any restrictions. This can lead to the complete escape of a GAE security sandbox.

In our Proof of Concept code, the following exploitation scenario is implemented to achieve that by combining either Issue 32 or 33 with Issue 34:

- an instance of a custom Class Loader that is a subclass of `java.net.URLClassLoader` class is created (`MyCL`),
- an intermediate Class Loader class (`PrivLoader`) is defined in `MyCL` namespace and outside of a GAE Class Sweeper sandbox with the use of a method handle corresponding to one of a `defineClass` methods of `java.security.SecureClassLoader` class,
- an instance of a `PrivLoader` class is created and used to define a privileged `HelperClass` class,
- `HelperClass` class is instantiated and a Security Manager is turned off.

Attached to this report, there are two Proof of Concept codes that illustrates the impact of the vulnerabilities described above. They have been successfully tested in a production GAE environment patched against security issues we reported to Google in Dec 2014 / Jan 2015.

REFERENCES

[1] "Google App Engine Java security sandbox bypasses", technical report `http://www.security-explorations.com/materials/se-2014-02-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.