PaaS (Platform as a service), such as Google App Engine, has two characteristics: one is separation between platform ownership and application ownership, another one is separation among different application ownerships. Next, I will introduce some technique challenges to implements security isolation in PaaS and our solution.

(1) PaaS simplifies the application development. Meanwhile, it expands the scope of the definition of sensitive operations (sensitive operation means the operation which may lead to security risks). For example, the method java.net.Inet4Address.getByName(host) will not be allowed to invoke by tenants because such method may lead to PaaS structure information leakage. As shown in Table 1, there are only 1313 classes in Google App Engine's white list.

Table 1 Classes are allowed in PaaS

JDK Version	Google App Engine White list [1]	Total public classes in JDK	Total classes in JDK
Sun JDK 1.6.0	1313	5206	7069

ONCE-AccessController provides two mechanisms to solve this problem: a) Blacklist mechanism. Classes in the blacklist will be filtered during the web application deployment; b) Redirection mechanism. During the web application deployment, security insensitive classes, such as java.net.Inet4Address, will be redirect to auto-generated proxy classes, which are security sensitive.

(2) In the Java security model, some privileges are coarse-grained. For example, Google App Engine does not support reflection [2] [3], because tenant can access the resources owned by the other tenants by reflection without security constraints.

Reflection in the Java security model is coarse-grained. If one tenant has the reflection privilege, one can accesses any resources by reflection.

ONCE-AccessController provides fine-grained mechanism to tackle this problem. When a tenant accesses resources by reflection, our mechanism will check whether the resource is owned by this tenant.

NOTE : Google App Engine has support reflection from version 1.2.1

(3) It is difficult to know security requirements of tenant application accurately because of the openness of PaaS. For example, Google App Engine does not support Struts2 completely [4], this is because Google App Engine does not support file operations [2], but some methods in Struts2 will create some temporary file during web application initialization [5].

ONCE-AccessController provides context-awareness mechanism to solve this problem. When a tenant creates some temporary file by using Struts2, our mechanism is path-awareness, and will activate the file operation privileges in the run-time for the tenant. In the other file operation situation, our mechanism will deactivate the file operation privileges in the run-time for the tenant.

(4) In the Java security model, the Security Manager is based on singleton pattern, which shared by all tenants. This design of Security Manager cannot meet the demand of separation between different application ownership. For example, one tenant may modify the other tenant's security constraints.

ONCE-AccessController provides isolation mechanism for this problem; each tenant has an isolated security manager instance. In this design, a tenant has no chance to get the security manager instance of another one's, and then the tenant cannot modify another one's security constraints.

[1] http://code.google.com/appengine/docs/java/jrewhitelist.html

[2] http://code.google.com/appengine/docs/java/runtime.html#The_Sandbox

[3] http://googleappengine.blogspot.com/2009/04/many-languages-and-in-runtime-bind-them.html

[4] http://groups.google.com/group/google-appengine-java/web/will-it-play-in-app-engine

[5]

http://whyjava.wordpress.com/2009/08/30/creating-struts2-application-on-google-app-engine-gae/