

Prevent reflection to access private methods and members in java classes

Contributed by Aminur Rashid
Thursday, 26 March 2009
Last Updated Thursday, 26 March 2009

Reflection is a nice evil. It can let you access private method/fields/constructor of a class.

```
import java.lang.reflect.Field;

public class UseReflection {
    public static void main(String args[]) {
        Object prey = new Prey();
        try {
            Field pf = prey.getClass().getDeclaredField("privateString");
            pf.setAccessible(true);
            pf.set(pre, "Aminur test");
            System.out.println(pf.get(pre));
        } catch (Exception e) {
            System.err.println("Caught exception " + e.toString());
        }
    }
}

class Prey {
    private String privateString = "privateValue";
}
```

Now in case you are wondering, that makes my class vulnerable to be modified, yes you are right. But then there is a way to prevent the caller from changing the modifier/changing the accessor. The easiest thing is to use the `SecurityManager`. Run the programme again using the default securitymanager provided by Java

```
java -Djava.security.manager UseReflection
```

For more on `DefaultPolicy` Implementation, you may want to read the document at [sun](#).

Another way to do is to write your own security manager, you must create a subclass of the `SecurityManager` class. Once you are done with your security manager, you can install it as the current security manager for your Java application. You do this with the `setSecurityManager()` method from the `System` class.

You can set the security manager for your application only once. In other words, your Java application can invoke `System.setSecurityManager()` only one time during its lifetime. Any subsequent attempt to install a security manager within a Java application will result in a `SecurityException`.

```
import java.lang.reflect.Field;
import java.security.Permission;

public class UseReflection {
    static{
        try {
            System.setSecurityManager(new MySecurityManager());
        } catch (SecurityException se) {
            System.out.println("SecurityManager already set!");
        }
    }
}
```

```
    }
    public static void main(String args[]) {
        Object prey = new Prey();
        try {
            Field pf = prey.getClass().getDeclaredField("privateString");
            pf.setAccessible(true);
            pf.set(pre, "Aminur test");
            System.out.println(pf.get(pre));
        } catch (Exception e) {
            System.err.println("Caught exception " + e.toString());
        }
    }
}

class Prey {
    private String privateString = "privateValue";
}

class MySecurityManager extends SecurityManager {
    public void checkPermission(Permission perm) {
        if(perm.getName().equals("suppressAccessChecks")){
            throw new SecurityException("Can not change the permission dude.!");
        }
    }
}
}
```