```
    public static setSecurityManager(SecurityManager sm)
    {
        if(security != null)
            throw new SecurityException
               ("SecurityManager already set");
        security = sm;
    }
}
```

The integrity of the `security` field is critical, since an applet that could set that field could control the security policy of the system. This makes `security` a likely target for attack.

15.4.3 Bypassing Java Security

The most straightforward attack would be to try to set the `security` field. The attacker's applet might be written in Java:

```
public class NastyApplet extends Applet
{
    void attack()
    {
        System.security = null;
        // The security manager is null, so everything is
        // permitted
        // Put code to wreak havoc here
    }
}
```

When this class is compiled, the Java compiler will notice that the `security` field is private to the `System` class and refuse to compile the file. This would not stop a determined attacker, who would proceed to rewrite the applet in Oolong:

```
.class public NastyApplet
.super java/applet/Applet

.method attack()V
aconst_null              ; Install null as the
                         ; security manager
putstatic java/lang/System/security Ljava/lang/SecurityManager;
;; Wreak havoc
.end method
```