**Privilege Management: Overly Broad Access Specifier Development Mitigation SOP**

Overly broad access specifier vulnerabilities come from privileged code in public methods because they can be called from anywhere in the JVM (Java Virtual Machine). Access specifiers should be as restrictive as possible because a method with a public access specifier means that any external code can call it. When public methods perform privileged actions, vulnerabilities could occur when code is shared in libraries or in environments where code can dynamically enter the system.

**Defense Against Privilege Management: Overly Broad Access Specifier**

Methods should be private if they contain privileged code. It is also best practice to ensure that member methods that call these methods do a good job of protecting their use through the proper validation checks.

**Example**

**public** static void doPrivilegedOpenFile(final String filePath) {

final BadFileNamePrivilegedAction pa = new

BadFileNamePriviledgedAction(filePath);

FileInputStream fis = null;

…

fis = (FileInputStream)AccessController.doPrivileged(pa);

…

}

**Explanation**

The code above uses a public access specifier and also contains a privileged operation.

**Recommendation**

This vulnerability can be fixed by changing the modifier to private:

**private** static doPrivilegedOpenFile(final String filePath) {

final BadFileNamePrivilegedAction pa = new

BadFileNamePrivilegedAction(filePath);

FileInputStream fis = null;

…

fis = (FileInputStream)AccessController.doPrivileged(pa);

…

}

**References**

1. [HP Enterprise Security – Privilege Management: Overly Broad Access Specifier](http://www.hpenterprisesecurity.com/vulncat/en/vulncat/java/privilege_management_overly_broad_access_specifier.html)