虽然，JavaEE 内置了一些非常优秀的安全机制，但是它不能全面应对应用程序面临的各种威胁，尤其许多最常见的攻击：跨站攻击（XSS），SQL 注入，Cross-Site Request Forgery (CSRF), 与 XML eXternal Entities (XXE) 等。如果你不对系统做大量的安全测试、漏洞修补以及购买应用级安全防护工具，应用程序就完全暴露在这些攻击之下。

幸运的是，开源 Web 应用安全组织（OWASP）已经将这下面10个 Web 问题列为最重要的安全攻击，详情请参见：「Ten Most Critical Web Application Security Risks」 报告。 希望引起大家这些安全问题的足够重视。

下面就详细解释一下这些最著名的安全攻击在 JavaEE 的 Web 应用程序和 Web 服务上是如何工作的：

1、注入攻击

在编写程序时，任何可疑的信息输入都可能是注入攻击，比如 request.getParameter(), request.getCookie() 以及request.getHeader()，甚至在用户命令行接口也存在注入风险。如果开发人员以数据和 SQL 命令拼接的方式形成 SQL 语句就存在 SQL 注入风险，比如： “SELECT * FROM users WHERE username=‘“ + request.getParameter(“user”) + “‘ AND password=‘“ + request.getParameter(“pass”) = “‘“; 开发者正确的写法应该是用 PreparedStatement 方式避免黑客有机会改变 SQL 语句的原意进而控制数据库。除了 SQL 注入还有很多注入攻击的方式，包括：Command Injection, LDAP Injection, 与 Expression Language (EL) Injection，所有的这些注入都非常非常危险，在编写接受数据的模块时一定要非常非常小心。

2、失效的身份和回话管理

JavaEE 对身份校验和会话管理都能够支持，但是安全方面做得很不够，有很多种方法可以破坏。程序员不得不确保每个身份校验都通过 SSL 安全通道，并且还要确保没有异常发生。如果不幸暴露了一个 JSESSIONID，黑客只要掌握了该

JSESSIONID 就可以劫持会话，很多时候为了防止会话固定攻击还不得不对 JSESSIONID 进行混淆。使用 response.encodeURL() 将 JSESSIONID 加到 URL 里面是非常危险的，JSESSIONID 很容易被偷窃，这种行为一定要避免。

3、Cross-Site Scripting (XSS)

若应用程序收到不可信的数据，在没有进行适当的验证和转义的情况下，就将它发送给一个网页浏览器，就会产生跨站脚本攻击（简称 XSS）。XSS 允许攻击者在受害者的浏览器上执行脚本，从而劫持用户会话、危害网站、或者将用户转向恶意网站。

4、不安全的直接对象引用

当开发人员暴露一个对内部实现对象的引用时，例如，一个文件、目录或者数据库密匙，就会产生一个不安全的直接对象引用。在没有访问控制检测或其他保护时，攻击者会操控这些引用去访问未授权数据。

5、安全配置错误

现代 JavaEE 应用程序和框架如 Struts，Spring 都有很多的安全配置，当使用这些框架一定要确保这些配置是正确的。比如在开发 Web 应用程序时一定要当心 里的 标签，该标签的意思是 security-constraint 只作用于标签里面列出的方法，黑客可以利用这个使用列表以外的方法如：HEAD 和 PUT 进行攻击，从而越过安全限制。大多数情况下开发者应该删掉 web.xml 里面的 标签。

6、敏感信息泄露

Java 使用扩展库的方式实现加解密，Java 提供通用的接口，任何用户，只需要简单的配置，都可以根据接口来实现加密，这样的好处是扩展性很强，弊端是如何正确使用密码库是非常不容易事情：第一步，找一个基于 JCE 的顶级加密库，提供简单、安全的加密方法，Jasypt 与 ESAPI 就非常不错。第二步，应该使用强加密算法如：加密用 AES，哈希用 SHA256，像密码这种敏感信息，广哈希是不够的，黑客可以通过 Rainbow 表来破译，因此需使用自适应安全算法如 bcrypt 或 PBKDF2。任何使

用不当都可能造成敏感信息泄露。

7、功能级访问控制缺失

JavaEE 同时支持声明式和编程两种访问控制方式，像 Spring 等框架也支持基于注解的访问控制，但是很多应用程序还是选择创建自己的访问控制流程，其实这是非常危险的行为。更重要的是要确保每一个暴露出去的接口和 Web 服务要有正确的访问控制，千万不要想当然的假设客户应该可以控制一切，这样黑客就可以直接访问程序了。

8、 跨站请求伪造（CSRF ）

每一个状态改变，应用程序都应该校验该请求是否伪造，开发者在每一个用户会话里面放置一个随机令牌，然后每次请求进来都进行校验，否则攻击者可能会创建一些包含有害标签，例如：IMG, SCRIPT, FRAME 或者 FORM，这些标签可能会指向没有保护的应用程序，当受害者访问这样的页面，浏览器就会自动产生一个伪造的 HTTP 请求到标签里指定的 URL，这个 URL 通常会包含受害者的凭证。

9、使用含有已知漏洞的组件

现代 JavaEE 应用程序通常会包括数百种库，尤其像依赖管理工具问世5年来，这个数目更是爆炸式的增长。广泛应用的Java 库都包含了很多已知漏洞，这些漏洞非常危险。对这些漏洞没有其他办法，只能等库的提供商修复漏洞，及时更新到最新版本。

10、未验证的重定向和转发

Web 应用程序经常将用户重定向或转发到其他网页和网站，并且利用不可信的数据去判定目的页面。如果没有得到适当验证，攻击者可以重定向受害用户到钓鱼软件或恶意网站，或者使用转发去访问未授权的页面， 在 JavaEE Web 程序里当调用 response.sendRedirect() 在使用 request.getParameter() 或 request.getCookie() 去获取不信任的数据时，经常会发生这种情况。

每个 JavaEE 程序员一定会经常遇到这十个安全问题，同时新的攻击和漏洞不断地被发现，我们现在能做的就是在开发、测试和部署的过程中不断地用安全代码检查工具对项目进行扫描，检查不修复漏洞。

大家可以尝试用 Eclipse 的一些免费对比插件来检查这些漏洞，这些不仅是静态分析工具，C4E 是一个非常有代表意义的工具，它利用 Java Instrumentation API 去见监控应用中所有和安全相关的内容。 它还可以实时分析整个数据流，在一个复杂的应用里从请求开始跟踪数据。 比如 JavaEE Web 应用里常见的数据流如下：代码从 request 取得参数，用 base64 解码，把数据存到 Map 里，再将 Map 存到一个 Bean 里面，然后将这个Bean 放到 Session 里作为 attribute，最后从 JSP 里取出，使用 EL 语言将这个 Bean 值填入页面。Eclipse 对比工具就可以跟踪这个数据流并报告是否存在 XSS 漏洞。这个工具非常方便，甚至对使用了非常复杂的框架和库的应用程序也管用，较现有的很多分析工具在速度，准确性和易用性上都有明显优势。

当然，还有现在非常流行的 RASP（Runtime Aplication Security Protector），也是对付这十大安全问题的利器，它将代码实时检查和实时拦截相结合，将安全保护代码和应用程序结合在一起，像疫苗一样使应用程序具备自我免疫的能力。这是 Gartner 极力推荐的应用程序安全保护方案，它使用非常方便，保护实时彻底，容易使用，不需要修改任何应用程序代码就可以轻松实现安全保护。