您的JAVA代码安全吗? PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/646/2021_2022__E6_82_A8_E7_9A_84JAVA_c97_646668.htm 虽然客户仍然很关心您为他们构建的应用程序的可伸缩性和可用性，但他们可能变得也很关心安全性，而且要求特别严格。应用程序可能容易受到两类安全性威胁的攻击：静态和动态。虽然开发人员不能完全控制动态威胁，但在开发应用程序时，您可以采取一些预防措施来消除静态威胁。本文概括并解释了 13 种类型的静态暴露 — 它们是系统中的缺陷，它使系统暴露在想要篡夺该系统的特权的攻击者面前。您将学会如何处理这些暴露，以及如何发现（如果不处理这些暴露）这些暴露可能造成的影响。在开发 Java Web 应用程序时，您需要确保应用程序拥有完善的安全性特征补充。这里在谈到 Java 安全性时，我们并不谈及 Java 语言提供的安全性 API，也不涉及使用 Java 代码来保护应用程序。本文将着重讨论可能潜伏在您的 Java 应用程序中的 安全性暴露。安全性暴露是系统中的缺陷，它使系统无法 — 即使系统被正常使用 — 防止攻击者篡夺对系统的特权、控制系统的运行、危及系统上的数据安全或者假冒未经授权的信任。相对于安全性暴露，许多开发人员更加关心网站的感官效果。 毫无疑问，客户现在既严格地关注性能、可伸缩性和可用性也严格地关注安全性。应用程序可能容易受到两类安全性威胁的攻击： 动态和静态。动态威胁是那些同未经授权进入系统有关的威胁，或那些同跨越网络传输的数据的完整性、隐私和机密性有关的威胁。这些威胁同应用程序的功能代码没有多大关系；使用加密、加密术和认证技术来

消除这些威胁。相比之下，静态威胁却同应用程序的功能代码有关；它们同进入系统的授权用户所做的事情有关。未知用户闯入系统是动态威胁的一个示例；授权用户以未授权方式操作系统内的代码或数据是静态威胁的示例。应用程序开发人员并不能完全控制动态威胁；但开发人员在构建应用程序时却可以采取预防措施来消除静态威胁。 在本文中，我们讨论了对付 13 种不同静态暴露的技巧。对于每种暴露，我们解释了不处理这些安全性问题所造成的影响。我们还为您推荐了一些准则，要开发不受这些静态安全性暴露威胁的、健壮且安全的 Java 应用程序，您应该遵循这些准则。一有合适的时机，我们就提供代码样本（既有暴露的代码也有无暴露的代码）。 对付高严重性暴露的技巧 请遵循下列建议以避免高严重性静态安全性暴露： 限制对变量的访问 让每个类和方法都成为 final，除非有足够的理由不这样做 不要依赖包作用域 使类不可克隆 使类不可序列化 使类不可逆序列化 避免硬编码敏感数据 查找恶意代码 限制对变量的访问 如果将变量声明为 public，那么外部代码就可以操作该变量。这可能会导致安全性暴露。 影响 如果实例变量为 public ，那么就可以在类实例上直接访问和操作该实例变量。将实例变量声明为 protected 并不一定能解决这一问题：虽然不可能直接在类实例基础上访问这样的变量，但仍然可以从派生类访问这个变量。 清单 1 演示了带有 public 变量的代码，因为变量为 public 的，所以它暴露了。 清单 1. 带有 public 变量的代码 Java代码 class Test { public int id. protected String name. Test(){ id = 1. name = "hello world". } //code } public class MyClass extends Test{ public void methodIllegalSet(String name){ this.name = name. 100Test 下