第5章

软件安全测试

5.1 软件安全测试的内容

软件安全测试的目标是识别被测软件中的安全威胁和漏洞，以帮助开发团队降低软件系统的安全风险，使系统在遇到威胁时不会停止运行或被利用。软件安全测试基于软件的安全需求进行测试，其测试内容通常涉及数据的保密性、完整性和可用性，通信过程中的身份认证、授权和访问控制，通信方的不可抵赖性，隐私保护和安全管理，以及软件中的安全漏洞等。

依据测试内容，软件安全测试包括两种类型，即软件安全功能测试和软件安全漏洞测试。

1. 软件安全功能测试

软件安全功能测试用于确认软件的安全属性和安全机制等安全需求是否得到满足。依据 GB/T 25000.10—2016《系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第 10 部分：系统与软件质量模型》，软件安全属性包括以下内容。

(1) 保密性。软件确保数据只有在被授权时才能被访问的程度。

(2) 完整性。软件或组件防止未授权访问、防止篡改计算机程序或数据的程度。

(3) 抗抵赖性。活动发生后可以被证实且不可否认的程度。

(4) 可核查性。实体的活动可以被唯一地追溯到该实体的程度。

(5) 信息安全性的依从性。产品或系统遵循与信息安全性相关的标准、约定或法规以及类似规定的程度。

软件安全功能测试采用黑盒测试方法，检测与安全相关的软件功能是否有效，相应的软件功能模块通常包括用户管理模块、权限管理模块、加密系统和认证系统等。根据 GB/T 25000.51—2016《系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第 51 部分：就绪可用软件产品(RUSP)的质量要求和测试细则》，软件安全性方面的质量要求包括以下内容。

(1) 软件应按照用户文档集中定义的信息安全性特征来运行。

(2) 软件应能防止对程序和数据的未授权访问(不管是无意的还是故意的)。

(3) 软件应能识别出对结构数据库或文件完整性产生损害的事件,而且能阻止该事件并通报给授权用户。

(4) 软件应能按照信息安全要求对访问权限进行管理。

(5) 软件应能对保密数据进行保护,只允许授权用户访问。

2. 软件安全漏洞测试

软件安全漏洞测试旨在从攻击者的角度发现系统的安全风险,通常采用模拟攻击和渗透测试等手段扮演攻击者,试图攻击系统、利用其漏洞来破坏系统的安全性。安全漏洞测试通过识别系统中的风险并创建由这些风险驱动的测试,专注于可能被成功攻击的系统脆弱点。

漏洞扫描是一种重要的软件安全测试技术,通过匹配已知的漏洞模式来发现被测软件中的漏洞,这些漏洞模式类似于病毒扫描程序所匹配的病毒签名。漏洞扫描工具不仅可用于扫描应用软件,也可用于扫描 Web 服务系统、数据库管理系统和操作系统等系统软件。常见的漏洞扫描工具可发现与各个已知漏洞模式相匹配的漏洞,但是无法识别未知模式的漏洞,也难以发现与漏洞聚合相关的风险。

软件安全测试应该遵循一些基本原则,例如 OWASP 组织列出了安全性测试的如下原则。

1) 没有万能方案

虽然安全扫描或应用防火墙可帮助识别软件的安全问题或提供针对攻击的防御,但实际上并没有彻底解决全部安全隐患。安全评估软件作为漏洞发现的第一步很有用,但在深入评估或提供足够的测试覆盖率方面通常不成熟且效率低下。

2) 尽早测试、经常测试

如果在软件开发生命周期中能尽早检测到软件安全缺陷,则能以尽量低的成本尽快解决安全问题。为此,需要对开发团队进行培训,使团队成员熟悉常见的安全问题,掌握预防和检测这些问题的方法,学会从攻击者的角度测试软件,并使得安全性测试活动成为常态。

3) 测试自动化

现代的软件开发方法,例如敏捷、DevOps/DevSecOps 或快速应用开发方法,将安全测试持续集成于开发的工作流,以维护基线安全并识别未处理的弱点。具体的实现措施包括在标准的工作流平台中引入自动化的软件安全测试工具,例如静态或动态的应用程序安全测试工具、软件依赖项跟踪工具等。

4) 了解安全的范围

了解特定项目需要多少安全性非常重要,应该对要保护的资产进行分类,并说明以何种安全级别(例如秘密、机密、绝密)来进行保护。根据《中华人民共和国网络安全法》,我国实行网络安全等级保护制度。网络运营者应当按照网络安全等级保护制度的要求,保障网络免受干扰、破坏或者未经授权的访问,防止网络数据泄露或者被窃取、篡改。

5) 尽量涉及源代码

黑盒渗透测试虽然有助于证明漏洞是如何暴露在生产环境中的,但无法发现在软件

需求中未提及而在代码中隐藏实现的很多漏洞，因此并非保护软件的最有效方法。如果被测软件的源代码可用，则应将其交给安全工程师进行代码审查和白盒测试，从而揭示出代码中隐藏的安全漏洞。

5.2　软件安全测试的方法

和通用的软件测试方法一样，软件安全测试的方法可以从是否关心软件内部结构、是否执行程序、测试阶段和程序执行方式等多角度进行划分，包括以下 4 种常见类型：基于模型的安全测试、基于代码的静态安全测试、动态安全测试和安全回归测试。

1. 基于模型的安全测试

基于模型的安全测试以降低软件系统的安全风险作为测试过程各个阶段活动的指南，这些安全测试活动涵盖测试的计划、设计、实施、执行和评估等。基于模型的安全测试用于验证与软件安全属性相关的软件需求，基于被测软件的模型验证软件的保密性、完整性、抗抵赖性和可核查性等安全属性。以下 3 种软件安全测试模型分别关注软件系统不同方面的安全属性。

（1）架构和功能模型。关注被测软件的安全需求及其实现，聚焦于预期的软件行为。

（2）威胁、故障和风险模型。关注软件可能出错的地方，关注系统威胁、故障和风险的原因及后果。

（3）弱点和漏洞模型。用于描述软件的弱点或漏洞本身的特性。

2. 基于代码的静态安全测试

基于代码的静态测试是软件安全开发过程的重要组成部分，有助于在软件开发的早期阶段发现漏洞，降低测试成本。静态代码审查可采用人工或自动的手段来完成，基于静态分析的安全测试则通常需要借助测试工具，被分析的对象包括被测程序的源代码或者编译后的代码（例如二进制码或字节码）。其中，基于源代码的静态安全测试更为精确，更有可能提供在源代码上修复漏洞的详细建议，因此更多地应用于软件的安全开发过程中。

与大多数动态安全测试工具相比，静态安全测试工具可以分析程序的所有控制流，实现更高的被测程序覆盖率。值得注意的是，静态安全测试工具报告的是潜在安全漏洞的列表。对于每个被报告的漏洞，安全专家需要进一步评估其是否为真实的漏洞，是否能被攻击者利用，从而决定是否需要被修复。

3. 动态安全测试

渗透测试是一种典型的动态安全测试，模拟来自恶意第三方的攻击进行测试，这意味着在大多数情况下，测试者只有有限的被测系统信息，并且只能与系统的公共接口交互。渗透测试通常针对的是生产环境中功能完整的软件，测试者可利用软件的充足数据来执行各种已实现的工作流。安全专家在进行渗透测试时通常会利用黑盒的漏洞扫描工具。这些工具使用一组预定义的攻击数据来查询被测程序的接口，分析程序的响应以判断攻

击是否成功，并在攻击未成功时提示如何在后续尝试中改变攻击方式。污点分析是安全测试的重要技术，通过信息流分析跟踪带有污点信息的执行路径来找到不受信任的数据等对象，从而发现代码漏洞，例如 SQL 注入和跨站脚本等漏洞。

模糊测试是另一种重要的动态安全测试技术，由 Wisconsin 大学的 Barton Miller 在 1990 年开创，其核心思想是向被测程序输入随机生成的数据，直至监测到程序异常（例如崩溃或断言失败），从而发现程序漏洞。最早的模糊测试方法基于随机生成的测试数据，但更先进的模糊测试方法结合符号执行和人工智能等技术，能自动化或半自动化地生成更有效的测试数据来指导被测程序的执行以暴露漏洞。

4. 安全回归测试

软件在使用过程中面临着不断变化的环境、新的业务需求、新的法规和新技术。软件系统本身或其环境的变化可能会引入新的安全威胁和漏洞，这使得保持软件系统的持续安全变得非常具有挑战性。为此，必须对变更了的软件系统进行安全回归测试，以确认对软件系统所做的更改是否损害其安全性。回归测试不仅要检测对软件所做的更改是否对未更改的部分产生意外影响，而且要检测软件的更改部分是否能按预期运行。

5.3 静态的软件安全测试

5.3.1 代码安全审查

源代码安全审查是人工检查程序源代码是否存在安全问题的过程。源代码安全审查所能发现的安全问题包括访问控制问题、密码弱点、后门、木马和逻辑炸弹等形式的恶意代码，有安全隐患的业务逻辑，未执行输入验证的代码以及故意开放控制程序的代码。许多软件安全问题难以通过其他形式的分析或测试来发现，这使源代码分析和审查成为重要的软件安全测试方法。通过源代码审查，测试人员可以准确地确定漏洞所在的代码位置，消除风险猜测的不确定性。

软件的代码安全审查应该发现常见的安全漏洞以及软件特定业务逻辑的问题。因此，审查者必须了解软件的业务目的和关键业务影响，还应了解攻击面，识别不同的威胁代理及其动机和攻击方式。在理想情况下，审查者应参与软件的设计阶段，熟悉设计的架构和文档。代码安全审查者可从以下几方面着手审查的准备工作。

（1）功能和业务规则。了解软件当前提供的所有功能，获取与其相关的所有业务限制或规则。注意从软件需求和设计中可推出的软件潜在功能，从而做出面向未来的安全决策。

（2）运行环境。运行环境是安全代码检查和风险评估必须要考虑的应用场景，所用的安全机制应当与此场景相匹配。例如，对手机应用商店的应用程序采用军用标准的安全机制就大可不必。

（3）敏感数据。对具有安全敏感性的数据实体，例如账号和密码，进行分类审查。

（4）用户角色和访问权限。了解软件的用户角色及其部署环境，分清不同角色用户

的访问权限和越权访问所带来的安全威胁。

（5）软件类型。分清软件是基于浏览器的应用、网络服务软件、移动应用还是混合应用软件，熟悉不同类型软件所面临的不同类型安全威胁和漏洞。

（6）代码。了解所使用的开发语言及其常见的安全问题特点。

（7）设计。分清软件设计是采用通用框架还是自己定制的结构，了解通用设计框架的常见安全问题特点。

（8）标准和规范。熟悉公司和行业所制定的安全开发标准和规范（例如安全编码规范），了解公司管理层如何控制软件的安全级别。

代码的安全审查者在做好上述准备工作之后，还需要搜集有关软件的更多信息以更有效地开展工作，通常可通过研究软件的需求、设计和测试等的文档来获得这些信息。为了尽快熟悉被测软件，审查者可与系统架构师和开发人员进行交谈，并实际运行软件以更好地了解软件。软件安全代码审查的对象不仅包括代码，而且包括数据：审查代码是为了检测代码是否能充分保护其信息和资产，待处理数据的上下文对于确定潜在风险非常重要。审查者在分析对软件设计的威胁时，需要从攻击者的角度观察设计并发现其中存在的后门和不安全因素，针对代码设计的常见安全审查问题如表 5-1 所示。

表 5-1　针对代码设计的常见安全审查问题

设 计 区 域	考虑的问题
数据流	（1）用户输入是否用于直接引用业务逻辑？ （2）是否存在数据绑定缺陷？ （3）失败情况下的执行流程是否正确？
认证和访问控制	（1）设计是否实现了对所有资源的访问控制？ （2）会话处理是否正确？ （3）哪些功能不需要身份验证即可访问？
已有安全控制措施	（1）第三方安全控制是否存在任何已知弱点？ （2）安全控制的位置是否正确？
架构	（1）与外部服务器的连接是否安全？ （2）来自外部来源的输入是否经过验证？
配置文件和数据存储	（1）配置文件中是否有任何敏感数据？ （2）谁有权限访问配置或数据文件？

5.3.2　静态代码分析

静态代码分析通常借助自动化或半自动化的工具对代码进行语法和语义分析，无须运行代码而发现代码中存在的安全漏洞。基于模式匹配的静态扫描工具能快速发现已知模式的软件漏洞，而基于信息流分析的静态分析工具则能通过分析代码的数据流、控制流和事件流等发现未知模式的漏洞。

在进行代码安全分析时，可选用商用的、免费的或开源的静态分析工具，通常商用工具比免费工具具有更多功能而且更可靠，而免费工具则普遍具有易用性。在选择代码安全性静态分析工具时，通常需要考虑以下问题。

(1) 该工具是否支持被测软件所用的编程语言?
(2) 在商业工具或免费工具之间是否存在偏好?
(3) 需要执行静态分析还是动态分析?
(4) 被测软件的主要功能是什么? 安全需求如何?
(5) 工具的使用者具有何种水平的专业知识?
(6) 工具的使用者对漏洞误报的容忍程度如何?

5.4 渗透测试

5.4.1 渗透测试过程

渗透测试用黑盒方式测试软件以发现其安全漏洞,无须了解软件的内部结构。渗透测试模拟攻击者使用被测软件的方式,通过不断尝试攻击被测软件来评估软件的安全性。测试人员在进行渗透测试时,通常需要获得软件的一个或多个有效账户的访问权限,验证软件的防御能按照预期进行,还可以验证所部署的源代码修复了以往所发现的某些特定漏洞。渗透测试从攻击者的角度分析被测软件的弱点、技术缺陷或漏洞,报告软件中存在的安全隐患和问题。渗透测试是一个渐进的过程,需要不断尝试攻击、逐步深入分析。渗透测试一般不采用 DDoS 等破坏性的攻击手段,而是采用不影响软件系统运行的攻击方法。

渗透测试报告的内容与其所采用的测试标准有关,以下介绍五种常用的渗透测试标准:开源安全测试方法标准 OSSTMM(Open Source Security Testing Methodology Manual)、开源 Web 应用安全项目标准 OWASP(Open Web Application Security Project)、美国国家标准与技术研究院制定的标准 NIST(The National Institute of Standards and Technology)、渗透测试执行标准 PTES(Penetration Testing Execution Standard)和信息系统安全评估框架 ISSAF(Information System Security Assessment Framework)。

1. OSSTMM

作为业内最受认可的渗透标准之一,OSSTMM 为渗透测试和漏洞评估提供了科学方法论,指导测试者从各种潜在攻击的角度识别软件系统及其组件的安全漏洞。这种方法依赖于测试者的专业知识和经验来解释所识别的漏洞及其潜在影响。OSSTMM 允许测试人员自定义渗透测试的需求或技术背景,涉及测试的物理位置、工作流程、运营安全指标、信任分析、人员安全测试、物理安全测试、无线安全测试、通信安全测试、数据网络安全测试、合规性等操作安全性,最终生成安全测试审计报告。

2. OWASP

在应用程序的安全测试方面,OWASP 测试标准得到了业界的广泛认可。OWASP 分别为以下不同类型的软件提供了测试指南:Web 服务软件、云服务软件、移动应用程序

(Android/iOS)和物联网固件等。OWASP 框架所提供的每种渗透测试方法都附有详细的指南,提供 60 多个可供评估的控件以帮助测试人员发现漏洞,不仅可识别 Web 或移动应用等软件中常见的安全问题,还可识别源于不安全开发实践的复杂逻辑缺陷。

3. NIST

NIST 测试指南将渗透测试过程分为以下四个阶段。

(1) 规划阶段。不进行实际测试,而是定义和记录测试的重要附加条件和边界。例如,确定作为测试对象的软件组件,确定测试的性质、范围和侵入程度。

(2) 发现阶段。首先系统地识别和枚举被测系统的所有可访问的外部接口,这组接口构成系统的初始攻击面。然后识别与接口匹配的漏洞类型,例如是 HTTP 服务的跨站点脚本漏洞或是数据库应用程序的 SQL 注入漏洞。如果被测的软件组件包含在已公开的漏洞库中,则在此阶段还可检查被测软件是否受到已公开漏洞的影响。

(3) 攻击阶段。针对识别出的接口进行一系列攻击尝试,主动发送攻击数据来破坏系统。在攻击成功的情况下,利用所发现的安全漏洞以获取更多的系统信息,扩大测试访问的权限并进一步找到暴露额外接口的更多系统组件。这个扩大的攻击面被反馈到发现阶段,以进行回溯处理。

(4) 报告阶段。此阶段与其他三个阶段同时进行,用于记录所有的安全问题并评估其严重程度。

4. PTES

PTES 标准将渗透测试定义为以下七个阶段,给出了渗透测试实践的建议,并进行测试工具的推荐。

(1) 前期交互。测试人员准备渗透测试所需的工具和环境。所需的工具因测试的类型和范围而异,由测试人员在测试活动开始时定义。

(2) 情报收集。被测软件的组织将向测试人员提供有关测试范围和目标等信息,测试人员从可公开访问的来源收集有关被测软件的更多详细信息。

(3) 威胁建模。威胁建模用于确定在何处应用修复策略以确保系统安全,关注业务资产、业务流程、威胁社区等关键要素。

(4) 漏洞分析。渗透测试人员需要识别、验证和评估漏洞所带来的安全风险,发现软件系统中可能被恶意滥用的缺陷。

(5) 漏洞攻击。此阶段利用已识别的漏洞试图破坏软件系统安全性,关注软件的薄弱点。

(6) 后渗透。在上一步的攻击完成后,渗透测试人员进一步分析、渗透受感染软件,以更深入地发现软件漏洞并尝试更进一步的攻击。

(7) 报告。最终将提供执行级别和技术级别的报告,涵盖测试内容、测试方式、发现的漏洞、渗透测试人员发现和利用这些漏洞的细节。该报告可为提升被测软件的安全性提供直接而有效的指导。

5. ISSAF

ISSAF 框架曾得到开放信息系统安全组(Open Information Systems Security Group, OISSG)的支持,目前已不再被维护,但它的优势之一是将单个渗透测试步骤与渗透测试工具相关联。ISSAF 标准包含比以往标准更加结构化和专业化的渗透测试方法,并提供进行渗透测试的综合指南。这些标准使测试人员能够精心计划和记录测试的各个步骤,并将每个步骤与特定工具联系起来。ISSAF 将渗透测试分为以下三个阶段。

(1) 规划和准备阶段。描述交换初始信息、计划和准备测试的步骤,强调在任何测试开始之前签署正式的评估协议。该协议为本次测试提供法律保护,并确定测试活动的参与团队、确切的日期和时间、升级路径和其他安排等。具体的测试活动包括确定公司和渗透测试团队之间的沟通渠道、确认测试的范围和方法、同意特定的测试用例和升级路径。

(2) 评估阶段。描述要使用的渗透测试工具以及被测试评估的对象,例如应用程序、数据库和网络等。每个评估活动包括以下不同层次的内容:信息收集,即使用技术和非技术的手段查找有关目标的信息;网络映射,即识别目标网络中的所有系统和资源;漏洞识别,即检测目标软件中的漏洞;渗透,即绕过安全措施获得未经授权的访问,获得尽可能广泛的访问权限;获得访问权限和权限提升,例如在目标系统上获得管理员级别的权限;进一步枚举,例如获取有关系统进程的附加信息;利用远程用户或站点,即利用远程用户和企业网络之间的信任关系通信;保持访问,即使用隐蔽的渠道、后门和 rootkit 来隐藏黑客的存在并提供对系统的持续访问;覆盖路径,即通过隐藏文件、清除日志、破坏完整性检查和破坏防病毒软件来消除所有利用迹象。

(3) 报告、清理和销毁垃圾文件的阶段。讨论测试过程中的沟通渠道和报告方式,包括口头报告和书面报告两种方式。口头报告仅用于关键或紧急问题,即在发现需要立即关注和采取行动的问题时进行口头交流。其典型情况是在渗透测试期间发现系统易受攻击并且已经受到损害。书面报告是渗透测试的正式输出,可以为被测软件系统中不同的利益相关者提供不同的版本,还可以包括关于口头报告中已经讨论过的问题信息。ISSAF 标准中的书面报告通常包括管理总结、项目范围、使用的渗透测试工具、利用的漏洞、测试日期和时间、已识别漏洞列表和按优先级排序的缓解漏洞建议等。

ISSAF 标准还包括“删除渗透测试遗留的任何人工痕迹”,这让渗透测试人员可以自由选择如何加密、清理和销毁渗透测试期间所创建的数据。在测试系统上所创建或存储的所有信息都应从最终交付的系统中删除。如果由于某种原因无法从远程系统中执行此删除操作,则应在技术报告中提及所有这些文件及其位置,以便客户技术人员能够在收到报告后删除这些文件。

5.4.2 渗透测试工具

渗透测试是软件安全验证的重要技术,在执行渗透测试时离不开渗透测试工具的使用。合理使用工具能够显著提升渗透测试效率,有效地收集信息和识别漏洞。本节将介绍 Kali Linux 所提供的经典渗透测试工具。

Kali Linux(也称 BackTrack Linux)于 2013 年 3 月 13 日发布,是一个基于 Debian 的

开源 Linux 发行版，预装了许多渗透测试工具，例如著名的 Nmap、Wireshark 和 John the Ripper 等，可用于软件系统的高级渗透测试和安全审计。Kali Linux 包含数百个针对各种信息安全任务的工具，这些安全任务包括渗透测试、安全研究、计算机取证和逆向工程。表 5-2 列出了其中一些工具及其特点。

表 5-2　渗透测试工具

工具名称	工 具 特 点
Metasploit	是针对大量已知软件漏洞的专业级漏洞攻击工具；具有可扩展的框架，其漏洞库不断更新，包括了各种平台上常见的溢出漏洞和 shellcode
Nmap	用于扫描目标计算机网络上的计算机、服务器和硬件的类型，搜集目标主机的网络状态和设置（例如主机是否在线、所提供的网络服务、所用的操作系统等），为后续的攻击做准备
Wireshark	是免费开源的网络抓包工具，使用 WinPCAP 接口直接与网卡进行数据报文交换，可检测网络问题；支持 UNIX 和 Windows 等多个平台
John the ripper	是免费的密码破解工具，采用基于字典的暴力破解方式；支持 UNIX/Linux 和 Windows 等多个平台
Burp Suite	包括一组查找和利用 Web 应用程序漏洞的工具，提供一个集成了认证、日志、警报和 HTTP 消息处理等功能的可扩展框架
OWASP ZAP	是 OWASP 所提供的 Web 应用程序漏洞测试包，包括自动的 Web 应用程序漏洞扫描功能，还提供了手动的漏洞测试工具
SQLmap	提供专业级的 SQL 注入漏洞的检测和利用，包括自动查找 SQL 注入漏洞，以及利用这些漏洞控制数据库和服务器等

渗透测试工具可用于信息收集、漏洞分析和漏洞利用等渗透测试活动。在信息收集活动中，测试人员可通过自动化工具来扫描测试目标的物理和逻辑区域，并根据漏洞分析的需要来查找有关目标的信息，包括目标网络、主机和应用程序中所存在的漏洞相关信息。在漏洞分析活动中，测试人员可以手动分析漏洞，也可以通过自动化的测试工具来辅助漏洞分析。在漏洞利用阶段，测试人员可针对所发现的软件漏洞，利用渗透测试工具对目标发起漏洞利用攻击。例如著名的渗透测试工具 Metasploit 不仅能够检测漏洞，还带有针对大量已知软件漏洞的专业级漏洞利用工具。

5.4.3　渗透测试与法律道德

渗透测试需要扫描和攻击被测软件系统，测试人员在执行渗透测试的实战和练习中，都需要小心谨慎，充分考虑法律和道德的约束，不能损坏有价值的系统，更不能触碰法律底线。渗透测试人员必须十分谨慎，莽撞操作可能会导致扫描了意外的 IP 地址或执行了意外的命令。例如，意外执行 Linux 命令“rm -rf /”将导致严重的后果，即递归删除根目录下的所有文件，而“> filename”命令则将删除文件的内容。

为坚守法律和道德底线，渗透测试人员应当遵守以下基本原则。

1. 避免接触或滥用不应拥有的系统

测试者不应当接触不属于自己的软件系统或网络，除非有法律协议允许对它们执行

某些操作。如果确实有法律协议，则仅在协议范围和参与规则内执行操作。测试者若要练习使用渗透测试工具和技术，则需要搭建自己的测试环境或者使用专用于此类练习的合法的测试环境。例如，Hack The Box 和 VulnHub 等在线资源为练习渗透测试提供了受控的测试环境。

2. 使用来源可信的工具和漏洞利用库

危害系统的恶意工具和漏洞有时会以虚假宣传来诱骗使用者，所以应使用受信任的工具（例如 Kali Linux 中包含的工具）和漏洞利用数据库（例如 Exploit DB）。恶意的工具或漏洞利用库可能导致使用者的数据损坏或被盗，为攻击者进入系统或网络打开了一扇门，从而影响系统的性能或造成安全危害。如果确实需要使用来源不明的工具，则应彻底审查该工具及其开发人员。如果测试人员无法审查来源不明的工具或漏洞利用库，则应寻找替代它们的方案。

3. 适当地隔离和分割

渗透测试人员应确保重要的数据和系统与测试环境分离并受到保护。渗透测试中的人为错误，即使是在精心执行的测试活动中，也可能导致意想不到的结果。渗透测试人员应当通过物理隔离、网络分段以及安全工具等手段来尽可能地遏制可能存在的损害，对于可以传播感染其他系统的恶意软件尤其应当进行隔离处理。

4. 在安全、受控的环境中测试漏洞和利用工具

渗透测试人员应在受控的环境中测试漏洞扫描和利用工具。操作系统、防病毒软件和防火墙等因素可能会改变这些工具和漏洞利用的行为，渗透测试人员可基于测试日志分析这些因素的影响。

5. 谨慎地进行漏洞利用

渗透测试人员在尝试漏洞利用之前应了解其工作原理和作用，在使用漏洞利用代码之前需根据使用目的对其进行审查和修改。特别要警惕未经审查来源的漏洞利用包含有危险代码，例如运行“rm -rf /”命令或其他有害功能的代码。

6. 不要恶意使用技能

如果不确定自己是否被允许做某事，很可能就不应该这样做，或者也可以事先咨询具有相关法律经验的人。无论是在进行渗透测试工作还是在练习渗透测试技巧，渗透测试者都应基于合理合法的目的进行测试操作。

5.5 模糊测试

5.5.1 模糊测试原理

模糊测试是一种通过向目标软件系统提供非预期的输入并监视异常结果来发现系统