渗透测试的两大阶段

渗透测试不能测试出所有可能的安全问题，它只是一个特定环境下才合适的 WEB 应用安全测试技术。OWASP 的渗透测试方法是基于墨盒方法的，测试人员在测试前不知道或只知道很有限的关于被测试应用的信息。

渗透测试被分成两大阶段：

■ **被动模式阶段**

在这个阶段，测试人员试图去理解被测应用的逻辑，并且去使用它。可以使用工具去收集信息，例如，可以用 HTTP 代理工具去观察所有请求与响应。本阶段结束后，测试人员应该理解了应用的所有访问点（如，HTTP 报头、参数和 COOKIE）。信息收集一节将介绍如何进行被动模式的测试。

■ **主动模式阶段**

这个阶段里，测试人员将利用后述的 9 大类 66 种方法主动地去测试。

被动模式阶段

信息收集

安全评估的第一步是收集尽可能多的关于被测应用的信息。信息收集是渗透测试的必要步骤。通常使用公共工具（搜索引擎）、扫描器、发送简单或特别的 HTTP 请求等来迫使被测应用泄漏信息。

◆ 使用蜘蛛、机器人和爬虫

目标是浏览和捕获被测应用相关的资源。

◆ 搜索引擎发现与侦察

类似 **GOOGLE** 这样的搜索引擎可以用来发现被测应用中已经被公开的错误页面或 WEB 应用结构问题。

◆ 识别应用入口点

枚举被测应用及其攻击面是展开任何攻击的的一个关键性前提。

◆ 测试 WEB 应用指纹

应用指纹是信息收集的第一步。知道正在运行的 WEB 服务器的版本和类型后，测试人员可以确定已知的漏洞和测试过程中的相应攻击方法。获取 WEB 应用指纹的自动化工具 Httprint 和在线工具 Netcraft。

◆ 应用发现

应用发现是一项面向驻留在 WEB/应用服务器中的 WEB 应用识别的活动。这种分析很重要，因为没有一个链接直接连接到主要应用的后端。分析可以发现有助于揭示诸如用于管理目的的 WEB 应用程序的细节。此外，它可以揭示诸如取消删除的，过时的脚本文件，这些文件通常是在测试、开发或维护过程产生的。可能使用到的工具：

1、DNS 查询工具，如 nslookup， dig 等。

2、端口扫描器（如 nmap： http：//www.insecure.org）和漏洞扫描器（如 Nessus： http：//www.nessus.org 和 wikto： [1]）。

3、搜索引擎（如 Google）。

4、基于 WEB 的与 DNS 相关的专业服务，如 Netcraft 的 DNS 搜索服务：http：//searchdns.netcraft.com/？host。

◆ 错误代码分析

在渗透测试过程中，WEB 应用程序可能会泄露一些不应该被最终用户看到的信息。测试人员根据诸如错误代码之类的信息可以推测出应用所使用的技术和产品。在不当的异常处理设计与编码的情况下，错误代码通常不需要专门技能或工具就可以很容易地去调用它。显然，只注重于 WEB 应用不可能达到详尽的测试，它达不到通过更广泛地基础分析收集到的信息后对被测应用的理解程度。

主动模式

配置管理测试

经常分析基础结构和拓扑结构可以获取大量的 Web 应用程序的信息。如源代码，可允许的 HTTP 方法，管理功能，身份认证的方法和基础结构的配置。

◆ SSL/TLS 测试

SSL 和 TLS 是两个通过加密为传播的信息提供安全信道的协议，该安全信道具有保护，保密和身份认证的功能。考虑到这些安全工具的关键性，确保加密算法的高强度及其正确执行非常重要。

◆ 数据库监听测试

在数据库服务器配置时，许多数据库管理员没有充分考虑到数据库侦听器组件的安全。如果没有进行安全的配置而使用手动或自动的技术进行侦听，侦听器就可能泄露敏感数据以及配置信息或运行的数据库实例信息。泄露的信息对测试者来说通常是有用的，他能将此投入到后续更有影响的测试中去。可能用到的工具：

tnscmd（Perl）： http：//www.jammed.com/%7Ejwa/hacks/security/tnscmd/tnscmd-doc.html

Toad for Oracle： http：//www.quest.com/toad

◆ 基础结构配置管理测试

相互联系的混杂的 Web 服务器结构能有数以百计的 web 应用程序，这种固有的复杂性使配置管理和审查成为测试和部署每一个应用程序的一个基本步骤。事实上一个漏洞就能破坏整个基础结构的安全，甚至某些微小且（几乎）不重要的问题可能对于相同服务器上的另外一个应用程序是个严重的威胁。为了解决这些问题，对配置和已知的安全问题执行深入审查是非常重要的。

◆ 应用配置管理测试

Web 应用程序隐藏了一些通常在应用程序自身开发和配置中没有考虑到的信息。这些信息可能从源代码，日志文件或 Web 服务器的默认错误代码中泄露。正确对待这一问题是安全评估中最基本的。

◆ 文件扩展名处理测试

从 Web 服务器或 Web 应用程序上的文件扩展名能够识别出目标应用程序使用的技术，例如扩展名 JSP 与 ASP。文件扩展名也可能暴露与该应用程序连接的其它系统。可能使用到的工具：Curl、漏洞扫描器、wget。

◆ 过时的、用于备份的以及未被引用的文件

Web 服务器上多余的，可读的和可下载的文件，如过时的，用于备份的和更名的文件，是信息泄漏的一个大源头。验证这些文件的存在是有必要的，因为它们可能包含应用程序和/或数据库的部分源代码，安装路径以及密码。

◆ 基础结构和应用管理接口

许多应用程序在管理接口使用一个公用路径，从而可能被用来猜测或暴力破解管理密码。此测试目的是找到管理接口，并了解是否可以利用它获取管理员权限。

◆ HTTP 方法和 XST 测试

在这个测试中，我们确保 Web 服务器没有被配置成允许使用具有潜在危险性的 HTTP 命令（方法），同时确保跨网站追踪攻击（XST）是不可能的。可能用到的工具：NetCat.

业务逻辑测试

认证测试

加密信道证书传输	在这里，测试员会试图了解用户输入 web 表单中的数据，例如，为了登录到一个网站而输入的数据，是否使用了安全协议传输，以免受到攻击。
用户枚举测试	这项测试的范围是为了验证是否有可能通过与应用的认证机制互动而收集一套有效的用户。这项测试将是有益于暴力测试。通过这种测试我们验证是否通过一个有效的用户名就可以找到相应的密码。
可猜解用户帐户猜测（遍历）测试	在这里我们测试是否有默认的用户帐户或可以猜测的用户名/密码组合（遍历测试）OWASP 测试指南 v3.0
暴力测试	当遍历攻击失败，测试者可以尝试使用暴力方法获得验证。暴力测试是不容易完成的测试，因为需要时间并且可能锁定测试者。
认证架构绕过测试	其它被动测试方法企图绕过身份的认识验证模式，因为他们认为并非所有的应用程序的资源都能得到充分的保护。测试者能够在没有认证的情况下访问这些资源。
记住密码和密码重置弱点测试	在这里，我们测试应用如何管理“忘记密码”过程。我们还检查应用是否允许用户在浏览器中存储密码（“记住密码”功能）。
注销和浏览器的缓存管理测试	在这里，我们检查注销和缓存功能得到正确实现。

CAPTCHA 测试	CAPTCHA（“全自动区分计算机和人类的图灵测试”）是一种许多 Web 的应用程序使用的挑战响应测试，以确保反应不是由计算机生成。即使产生的 captcha 是牢不可破的，其执行经常遭受各种攻击。本节将帮助您确定这些攻击的类型。
多因素身份验证测试(Testing Multiple Factors Authentication)	多因素身份验证意味着测试了以下的情况：一次性密码(OTP)所生成的验证码，加密设备，如 USB 验证码或配备了 X.509 证书的智能卡，通过 SMS 发送的随机一次性密码，只有合法用户知道的个人信息[OUTOFWALLET] 。
竞态条件测试	竞态条件是一个缺陷。当行动的时间影响了其它行动时，它会产生意想不到的结果。例如：一个多线程应用程序对同一数据进行操作时。就其性质而言，竞态条件很难测试。

授权测试

◆ 路径遍历测试

测试是否能够找到一种方法来执行路径遍历攻击并获得保留信息

◆ 绕过授权模式测试

这种测试的重点是核实如何对每一个角色/特权实施授权模式以便获得保留功能/资源。

◆ 权限提升测试

在此阶段，测试者需要确认用户不可能采用允许特权提升攻击的方式修改自己在应用程序内部的特权/角色。

会话管理测试

◆ 会话管理模式测试

分析会话管理模式，理解如何开发会话管理机制，并确定是否能打破这一机制以便绕过用户会话。测试发送给客户端浏览器的会话验证码的安全：如何对 cookie 实行反向工程，以及如何通过篡改 cookies 来劫持会话。

◆ Cookies 属性测试

测试已正确配置的 cookie 的属性。Cookies 往往是恶意用户关键的攻击媒介（通常针对其他用户）。因此，应用程序应始终采取措施保护 cookie。

◆ 会话固定测试

当应用程序在成功验证用户后不更新 Cookie 时，我们就能找到会话固定漏洞并迫使用户使用攻击者已知的 cookie。

◆ 会话变量泄漏测试

因为会话验证码联系着用户身份和用户会话，所以它代表的是保密信息。我们可以测试会话验证码是否暴露在漏洞中，并试着追溯会话攻击。

◆ 跨站请求伪造（CSRF）测试

跨站请求伪造描述了在 web 应用中迫使已通过验证的未知用户执行不必要请求的方法。

数据验证测试

我们将数据验证测试划分为以下类别：

◆ 跨站点脚本的测试

在跨站脚本攻击（XSS）的测试中，我们测试能否操纵应用程序参数输入，使之产生恶意输出。当应用程序没有：验证输入数据并在我们控制下产生输出时，我们就能发现 XSS 漏洞。此漏洞会产生各种攻击。例如，窃取机密信息（如会话 cookie）或控制受害者的浏览器。一个跨站脚本攻击方式如下：Input -> Output == cross-site scripting。

- 跨站脚本反射测试
- 跨站脚本存储测试
- 跨站脚本 DOM 测试
- FLASH 跨站测试

◆ SQL 注入

SQL 注入测试检测是否有可能将数据注入到应用程序中，以便它能在后端数据库中执行用户控制的 SQL 查询。如果应用程序在没有恰当验证数据的情况下使用用户输入创建 SQL 查询，那幺说明该应用程序存在 SQL 注入漏洞。成功利用这一类别的漏洞会导致未授权用户访问或操作数据库中的数据。请注意，应用数据往往代表了公司的核心资产。SQL 注入攻击方式如下：Input -> Query SQL == SQL injection

SQL 注入测试进一步细分为：

- Oracle 测试
- MySQL 测试
- SQL Server 测试
- MS ACCESS 测试
- PostgreSQL 测试

◆ LDAP 注入

LDAP 注入测试类似于 SQL 注入测试。不同之处在于我们不是使用 SQL 而是使用 LDAP 协议，同时测试的目标是 LDAP 服务器，而不是 SQL 服务器。LDAP 注入攻击方式如下：

Input -> Query LDAP == LDAP injection

◆ ORM 注入

ORM 注入测试同样类似于 SQL 注入测试。在这种情况下，我们使用 SQL 注入攻击 ORM 产生的数据访问对象模型。从测试的角度来看，这种攻击几乎和 SQL 注入攻击相同。然而，代码中存在的注入漏洞是由 ORM 工具产生的。

◆ XML 注入

XML 注入测试检测是否有可能在应用程序中注入特定的 XML 文档。如果 XML 解析器没有验证任何数据，那么该应用程序存在 XML 注入漏洞。一个 XML 注入攻击方式如下：

Input -> XML doc == XML injection

◆ SSI 注入

Web 服务器通常让开发者在静态 HTML 网页中增加小型动态代码，而不必处理全面的服务器端或客户端语言。服务器端嵌入（SSI）注入能够体现这一特色。SSL 注入测试检测是否有可能在应用程序中注入 SSI 机制解释的数据。黑客成功利用此漏洞后能够将代码注入到 HTML 网页，甚至远程执行代码。

◆ XPath 注入

XPath 是针对部分 XML 文件而设计和开发的语言。XPath 注入测试检测是否有可能在应用程序中注入数据，以便执行用户控制的 XPath 查询。攻击者成功利用这个安全漏洞就能够绕过认证机制或未经授权获取信息。

◆ IMAP/SMTP 注入

这种威胁影响到所有与邮件服务器（IMAP/SMTP）连接的应用程序，通常是 Webmail 应用程序。由于输入没有得到验证，IMAP/SMTP 注入测试检测是否有可能在邮件服务器中注入任意的 IMAP/SMTP 命令。一个 IMAP/SMTP 注入攻击方式如下：

Input -> IMAP/SMTP command == IMAP/SMTP Injection

◆ Code 注入

代码注入测试检测是否有可能在应用程序中注入稍后由 web 服务器执行的代码。代码注入攻击的方式如下：

Input -> malicious Code == Code Injection

◆ OS 命令

在命令注入测试中，我们设法通过 HTTP 请求在应用程序中注入 OS 命令。操作系统命令注入攻击方式如下：

Input -> OS Command == OS Command Injection

◆ 缓冲区溢出

在这些测试中，我们检查不同类型的缓冲区溢出漏洞。以下是常见的缓冲区溢出漏洞的测试方法：

- 堆溢出
- 栈溢出
- 字符串格式

一般缓冲区溢出攻击方式如下：

Input -> Fixed buffer or format string == overflow

◆ 孵育漏洞测试

孵育测试是需要多个数据验证漏洞工作的复杂的测试，需要一个以上的数据验证漏洞工作。在每一个显示的模式中，应用程序应该在信任和处理这些数据前对数据进行验证。我们的测试目的是验证应用程序实际上是否执行了验证并且不信任其输入数据。

◆ HTTP-Splitting/Smuggling 测试

讲述如何测试一个 HTTP 开发，例如： HTTP Verb， HTTP Splitting， HTTP Smuggling。

拒绝服务攻击的测试

◆ SQL 通配符攻击测试

SOL 通配符攻击是利用几个通配符强制数据库执行占用大量 CPU 处理时间的查询密集队列。这个漏洞通常存在于网络应用程序的搜索功能。成功的攻击操作会引起阻断服务攻击。

◆ 锁定用户账户

检查攻击者能否通过重复尝试用错误密码登陆，导致合法用户账户被锁定。

◆ 缓存溢出

检查能否在溢出目标应用的一个或多个数据结构的环境下引起阻断服务攻击。

◆ 分配用户指定对象

检查是否有可能使用分配大量对象的方法来耗尽服务器资源。

◆ 将用户输入作为循环计数器

检查是否可能让应用强制循环一段需要很多计算机资源的代码，目的是减弱总体性能。

◆ 将用户提供的数据写到磁盘

检查是否能通过在目标磁盘中写满日志来创造 DoS 环境。

◆ 释放资源失败

检查应用程序在使用后是否适当的释放资源（文件或内存）。

◆ 存储过多会话数据

检查是否可以通过向用户会话分配大数量的数据来耗尽内存资源。

WEB 服务测试

SOA（面向服务体系结构）/Web 服务应用程序是日渐重要的系统，它们可以使作业交互。同时它们以前所未有的速度在发展着。Web 服务“客户”不是网络前段而是后端服务器。Web 服务跟其它服务一样是暴露于网络的，但是可以在其它传输协议 HTTP，FTP，SMTP，MO 上使用。Web 服务框架在连接 XML，SOAP，WSDL 和 UDDI 技术中利用 HTTP 协议（作为标准网络应用程序）。

Web 服务中的漏洞跟别的漏洞类似，比如 SQL 注入，信息暴露和泄漏，但是这里将会讨论到 Web 服务独特的与漏洞有关的 XML/解析器。以下描述了 Web 服务测试：

- WS 信息收集
- WSDL 测试
- XML 结构测试
- XML 内容级别测试
- HTTP 获取参数/REST 测试
- 淘气的 SOAP 附件
- 回放测试

AJAX 测试

使用 AJAX 技术可以极大提高 web 对应用程序的操作性。然而从安全的观点看，AJAX 应用程序比正常的 Web 应用程序存在更大的攻击面，并且开发他们的重点在于能够做些什么，而不是应该做些什么。另外，由于程序是在客户端和服务器运行的，因此 AJAX 应用程序更复杂。使用框架掩盖这种复杂性可以帮助减少其开发的问题，但也可会导致开发者不能充分了解代码执行的具体位置。这可能导致很难正确地评估与特定应用程序或功能相关的风险。

渗透测试的基本步骤

渗透测试的基本步骤分为以下四步：

1、确定关注的安全方面

2、收集相关信息

3、使用主动攻击模式中的技术进行测试

4、评估实际风险并撰写报告

确定关注的安全方面

确定最需要关注的安全风险类型，在这个范围内进行安全测试。

收集相关信息

参见前面的“被动模式阶段”一节内容。

使用主动攻击模式中的技术进行测试

参见前面的“主动模式阶段”一节内容。

评估实际风险并撰写报告

根据风险分级标准对实际风险进行评估，并撰写安全测试报告。