
Web 安全测试

文件状态	<input type="checkbox"/> 草稿 <input checked="" type="checkbox"/> 正式发布 <input type="checkbox"/> 正在修改		
当前版本	V1.0		
拟 制		日期	
审 核		日期	
批 准		日期	

目 录

1	概述.....	4
1.1	背景简介.....	4
1.2	适用读者.....	4
1.3	适用范围.....	4
1.4	安全测试在项目整体流程中所处的位置.....	5
1.5	安全测试在安全风险评估的关系说明.....	5
1.6	注意事项.....	5
1.7	测试用例级别说明.....	6
2	Web 安全测试方法.....	7
2.1	安全功能验证.....	7
2.2	漏洞扫描.....	7
2.3	模拟攻击实验.....	7
2.4	侦听技术.....	7
3	Appscan 工具介绍.....	8
3.1	AppScan 工作原理.....	8
3.2	AppScan 扫描阶段.....	9
3.3	AppScan 工具使用.....	10
4	测试用例和规范标准.....	18
4.1	输入数据测试.....	19
4.1.1	SQL 注入测试.....	19
4.1.2	命令执行测试.....	24
4.2	跨站脚本攻击测试.....	25
4.2.1	GET 方式跨站脚本测试.....	26
4.2.2	POST 方式跨站脚本测试.....	27
4.2.3	跨站脚本工具实例解析.....	29
4.3	权限管理测试.....	31
4.3.1	横向测试.....	31
4.3.2	纵向测试.....	32
4.4	服务器信息收集.....	36
4.4.1	运行账号权限测试.....	36
4.4.2	Web 服务器端口扫描.....	36
4.5	文件、目录测试.....	37
4.5.1	工具方式的敏感接口遍历.....	37
4.5.2	目录列表测试.....	39
4.5.3	文件归档测试.....	42
4.6	认证测试.....	43
4.6.1	验证码测试.....	43
4.6.2	认证错误提示.....	44
4.6.3	锁定策略测试.....	44
4.6.4	认证绕过测试.....	45

4.6.5	修复密码测试.....	46
4.6.6	不安全的数据传输.....	47
4.6.7	强口令策略测试.....	47
4.7	会话管理测试.....	49
4.7.1	身份信息维护方式测试.....	49
4.7.2	Cookie 存储方式测试.....	50
4.7.3	用户注销登陆的方式测试.....	50
4.7.4	注销时会话信息是否清除测试.....	51
4.7.5	会话超时时间测试.....	52
4.7.6	会话定置测试.....	52
4.8	文件上传下载测试.....	54
4.8.1	文件上传测试.....	54
4.8.2	文件下载测试.....	55
4.9	信息泄漏测试.....	56
4.9.1	连接数据库的账号密码加密测试.....	56
4.9.2	客户端源代码敏感信息测试.....	56
4.9.3	客户端源代码注释测试.....	57
4.9.4	异常处理.....	57
4.9.5	不安全的存储.....	59
4.10	逻辑测试.....	60
4.11	其他.....	60
4.11.1	Class 文件反编译测试.....	60
5	本规范所涉及的测试工具.....	61

1 概述

1.1 背景简介

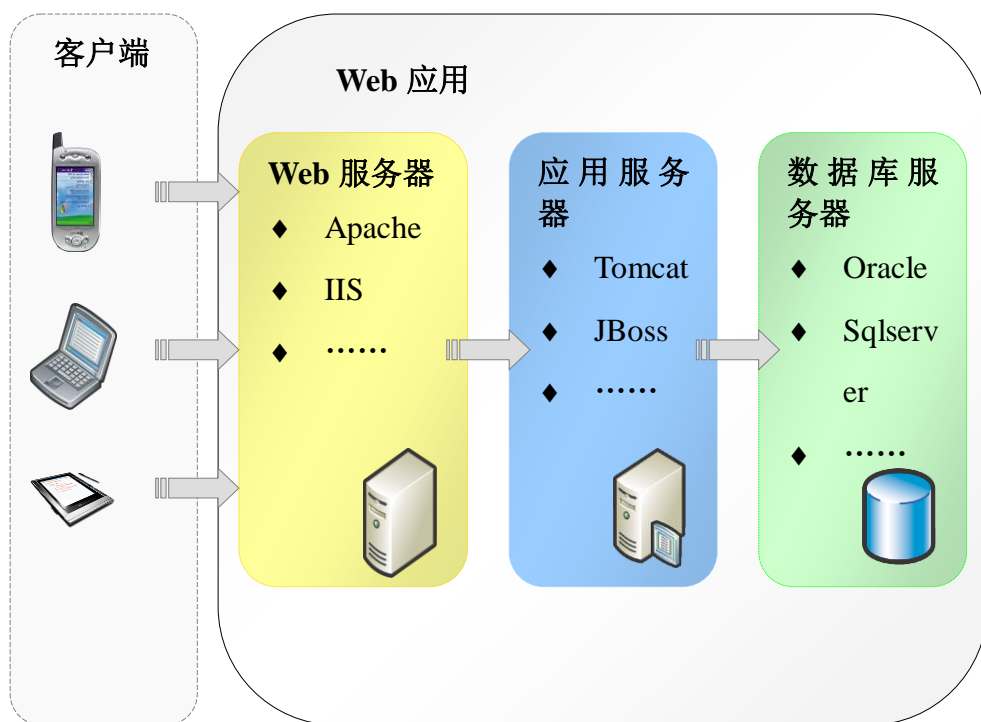
为了规避安全风险、规范代码的安全开发,以及如何系统的进行安全性测试,目前缺少相应的理论和方法支撑。为此,我们制定《安全测试规范》,本规范可以让测试人员针对常见安全漏洞或攻击方式,系统的对被测系统进行快速安全性测试。

1.2 适用读者

本规范的读者及使用对象主要为测试人员、开发人员等。

1.3 适用范围

本规范主要针对基于通用服务器的Web应用系统为例,其他系统也可以参考。如下图例说明了一种典型的基于通用服务器的Web应用系统:



本规范中的方法以攻击性测试为主。除了覆盖业界常见的 Web 安全测试方法以外，也借鉴了一些业界最佳安全实践。

1.4 安全测试在项目整体流程中所处的位置

一般建议在集成测试前根据产品实现架构及安全需求，完成安全性测试需求分析和测试设计，准备好安全测试用例。

在集成版本正式转测试后，即可进行安全测试。如果产品质量不稳定，前期功能性问题较多，则可适当推后安全测试执行。

1.5 安全测试在安全风险评估的关系说明

安全风险是指威胁利用漏洞对目标系统造成安全影响的可能性及严重程度。其中威胁是指可能对目标系统造成损害的潜在原因，包括物理环境威胁、人为威胁等。漏洞也称弱点，是应用系统本身存在的，包括系统实现中的缺陷、配置中的漏洞等。外部威胁利用系统的漏洞达到破坏系统安全运行的目的。

本规范所描述的安全测试仅是安全风险评估中的一个活动，对应于安全风险评估过程中的漏洞识别部分，特别是技术性的漏洞识别。

完整的安全风险评估过程、概念见 GB/T 20984-2007《信息安全技术 信息安全风险评估规范》。

1.6 注意事项

- 安全测试的执行，对于被测系统，或多或少都会存在一些影响（比如性能、垃圾数据），只能在测试环境中进行；**不允许在正式环境运行，避免系统存在漏洞导致丢失数据和数据库损坏。**
- 本规范最主要目的是为了发现安全漏洞，至于发现一个漏洞以后更深一步的渗透测试在这里不会涉及。例如针对暴力破解测试，我们只给出验证存在暴力破解漏洞的可能，但不会提供暴力破解的方法。
- 本文档中所有提及的测试工具的使用，请遵循公司相关规定。
- 如果是内部试验环境进行测试，可以考虑去除一些防护措施或设备（如

防火墙)，这样能保证发现问题的全面性。

- 本文档根据最严格的方式对目标进行测试，如果项目系统对安全的要求不高且有自身的安全策略规定时，可以视情况对测试项进行部分测试。例如密码策略测试项中，测试人员可以根据测试目标的密码位数要求进行测试，而不需要完全依照测试项里面规定的位数进行测试。

1.7 测试用例级别说明

一个安全漏洞的风险程度受危害程度和概率的影响，我们定义了如下所示的关系：

危害程度 \ 发生概率	高	中	低
高	高	高	中
中	高	中	低
低	中	低	低

表 1 风险等级界定表

本测试规范用例根据上面的定义分为四个测试级别：

测试用例级别	说明
一级	基本：该类用例涉及可能导致风险程度为高的安全漏洞，在任何情况下都必须进行测试。
二级	重要：该类用例涉及可能导致风险程度为中的安全漏洞，在条件允许（时间、人力充沛）情况下必须进行测试。
三级	一般：该类用例涉及可能导致风险程度为低的安全漏洞，测试结果可能对其他测试有帮助。测试与否根据业务系统的重要性来判断。
四级	生僻：该类用例涉及可能导致风险程度为极低的安全漏洞，攻击者只能收集到很少且无关紧要的信息。一般情况下不建议进行测试。

表 2 测试用例级别说明表

2 Web 安全测试方法

2.1 安全功能验证

功能验证是采用软件测试当中的黑盒测试方法，对涉及安全的软件功能，如：用户管理模块，权限管理模块，加密系统，认证系统等进行测试，主要验证上述功能是否有效，不存在安全漏洞，具体方法可使用黑盒测试方法。

2.2 漏洞扫描

漏洞扫描是指基于漏洞数据库，通过扫描等手段对指定的远程或者本地计算机系统的安全脆弱性进行检测，发现可利用的漏洞的一种安全检测（渗透攻击）行为。

漏洞扫描技术是一类重要的网络安全技术。它和防火墙、入侵检测系统互相配合，能够有效提高网络的安全性。通过对网络的扫描，网络管理员能了解网络的安全设置和运行的应用服务，及时发现安全漏洞，客观评估网络风险等级。网络管理员能根据扫描的结果更正网络安全漏洞和系统中的错误设置，在黑客攻击前进行防范。如果说防火墙和网络监视系统是被动的防御手段，那么安全扫描就是一种主动的防范措施，能有效避免黑客攻击行为，做到防患于未然。

2.3 模拟攻击实验

模拟攻击测试是一组特殊的黑盒测试案例，以模拟攻击来验证软件或信息系统的安全防护能力，可使用冒充、重演、消息篡改、服务拒绝等方法来实现。

2.4 侦听技术

在数据通信或数据交互过程，对数据进行截取分析的过程。目前最为流行的是网络数据包的捕获技术。

3 Appscan 工具介绍

Appscan 扫描工具只能检测到部分常见的漏洞（如跨站脚本、SQL 注入等），不是针对用户代码的，也就是说不能理解业务逻辑，无法对这些漏洞做进一步业务上的判断。往往最严重的安全问题并不是常见的漏洞，而是通过这些漏洞针对业务逻辑和应用的攻击。

Web 目前分为“应用”和“Web 服务”两部分。应用指通常意义上的 Web 应用，而 Web 服务是一种面向服务的架构的技术，通过标准的 Web 协议（如 HTTP、XML、SOAP、WSDL）提供服务。

3.1 AppScan 工作原理

AppScan 工作原理：

- 1、通过搜索(爬行)发现整个 Web 应用结构。
- 2、根据分析，发送修改的 HTTP Request 进行攻击尝试(扫描规则库)。
- 3、通过对于 Respond 的分析验证是否存在安全漏洞。

所以，AppScan 的核心是提供一个扫描规则库，然后利用自动化的“探索”技术得到众多的页面和页面参数，进而对这些页面和页面参数进行安全性测试。

“扫描规则库”，“探索”，“测试”就构成了 AppScan 的核心三要素。



如上图，单击“扫描”下面的小三角，可以出现如下的三个选型“完全扫描”、“仅探索”、“仅测试”三个名词，该三个类型为 AppScan 三个核心要素；

AppScan 是对网站等 Web 应用进行安全攻击来检查网站是否存在安全漏洞；对网站来说，一个网站存在的页面，可能成千上万。每个页面也都可能存在

多个字段（参数），比如一个登陆界面，至少要输入用户名和密码，就是说一个页面存在两个字段，你提交了用户名密码等登陆信息，网站要有地方接受并且检查是否正确，这就可能存在一个新的检查页面。这里的每个页面的每个参数都可能存在安全漏洞，都是被攻击对象，所以都需要检查。

这就存在一个问题，我们来负责来检查一个网站的安全性，这个网站有多少个页面，有多少个参数，页面之间如何跳转，我们可能并不明确，如何知道这些信息？

访问一个网站的时候，我们需要知道的最重要的信息是哪个？网站主页地址？从网站地址开始，其他页面都可以链接过去，那么可不可以有种技术，告诉了它网站的入口地址，然后它“顺藤摸瓜”，找出其他的网页和页面参数？所以这就是“爬虫”技术，具体说，是“网站爬虫”，其利用了网页的请求都是用 http 协议发送的，发送和返回的内容都是统一的语言 HTML，那么对 HTML 语言进行分析，找到里面的参数和链接，纪录并继续发送，最终，找到了这个网站的众多的页面和目录。这个 AppScan 就提供，这里的术语叫“探索”。

在使用 AppScan 的时候，要配置的第一个就是要检查的网站地址，配置了以后，AppScan 就会利用“探索”技术去发现这个网站存在多少个目录，多少个页面，页面中有哪些参数等，简单说，了解网站的结构。

“探索”了解了，测试的目标和范围就大致确定了，然后利用规则库（AppScan 的扫描规则库，其类似杀毒软件的病毒库），发送各种发送请求，进行安全攻击，这个过程就是“测试”；具体可以检查的安全攻击类型都在里面做好了，我们去使用即可。

完全测试就是把上面的两个步骤整合起来，“探索”+“测试”；在安全测试过程中，可以先只进行探索，不进行测试，目的是了解被测的网站结构，评估范围；然后选择“仅测试”，只对前面探索过的页面进行测试，不对新发现的页面进行测试。“完全测试”就是把两个步骤结合在一起，一边探索，一边测试。

3.2 AppScan 扫描阶段

Appscan 全面扫描包括两个步骤：探索和测试；

1、探索阶段：appscan 会通过模仿成 web 用户单击链接并填写表单字段来探索

站点（web 应用程序或 web server）这就是探索阶段。探索阶段可以遍历每个 URL 路径，并分析后创建测试点。

既工具会模仿一个用户对被访问的 Web 应用或 Web 服务站点进行探测访问，通过发送请求对站点内的链接与表单域进行访问或填写，以获取相应的站点信息。Appscan 分析器将会对自己发送的每一个请求后的响应做出判断，查找出任何可能潜在风险的地方，并针对这些可能会隐含风险的响应，确定将要自动生成的测试用例。但是在探测阶段完成后，这些高危区域是否真的隐含着安全缺陷或应做更好的改良，以及这些隐含的风险是处于什么层度的，是需要测试执行完成后，才能最终得出结论。

- 2、测试阶段：“测试”期间，appscan 会发送它在“探索”阶段创建的成千上万个定制的测试请求，通过你定制好的测试策略分析每个测试的响应，最后根据规则识别应用程序中的安全问题，并排列这些安全问题的风险级别。

既 Appscan 是通过测试策略库中对相应安全隐患的检测规则而生成对应的足够全面而且复杂的测试输入（测试用例）。

- 3、扫描阶段：Appscan 才是真正的工作起来，他将会把上个阶段的测试用例产生的服务请求陆续的发送出去。然后再检测分析服务的响应结果，从而判断该测试用例的输入，是否造成了安全隐患或安全问题。然后再通过测试用例生成的策略，找出该安全问题的描述，以及该问题的解决方案，同时还报告相关参数的请求发送以及响应结果。

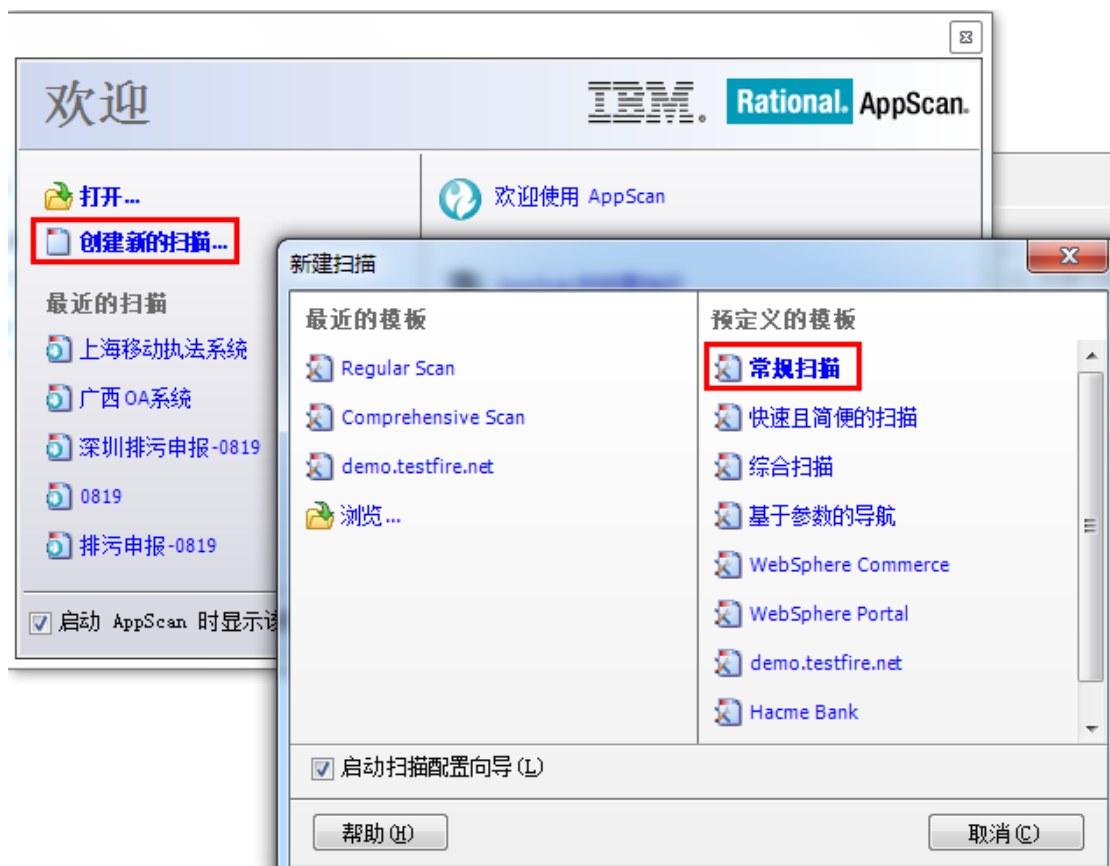
3.3 AppScan 工具使用

➤ 扫描

如果是第一次启动，屏幕中央将会出现一个“欢迎”对话框。在此对话中，可以点击“入门”链接，查看 IBM Rational AppScan 的“新手入门帮助文档”，如下图：



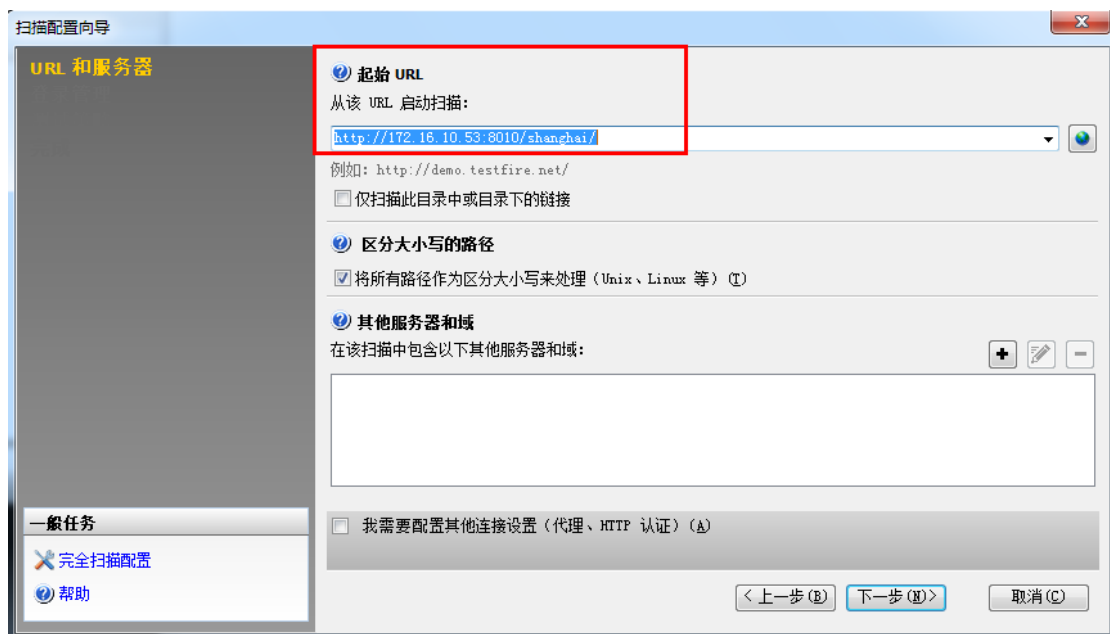
点击“创建新的扫描”，就可以开始扫描任务，选择“常规扫描”为例，如下图：



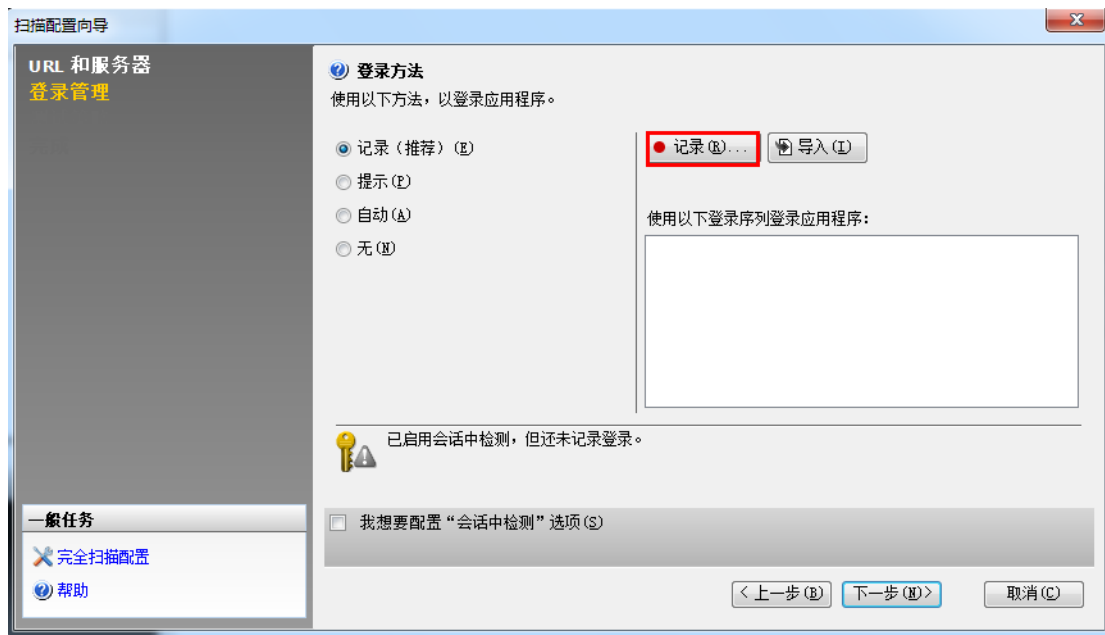
选择扫描类型为：Web 应用程序扫描，如下图：



输入起始 URL，如下图：

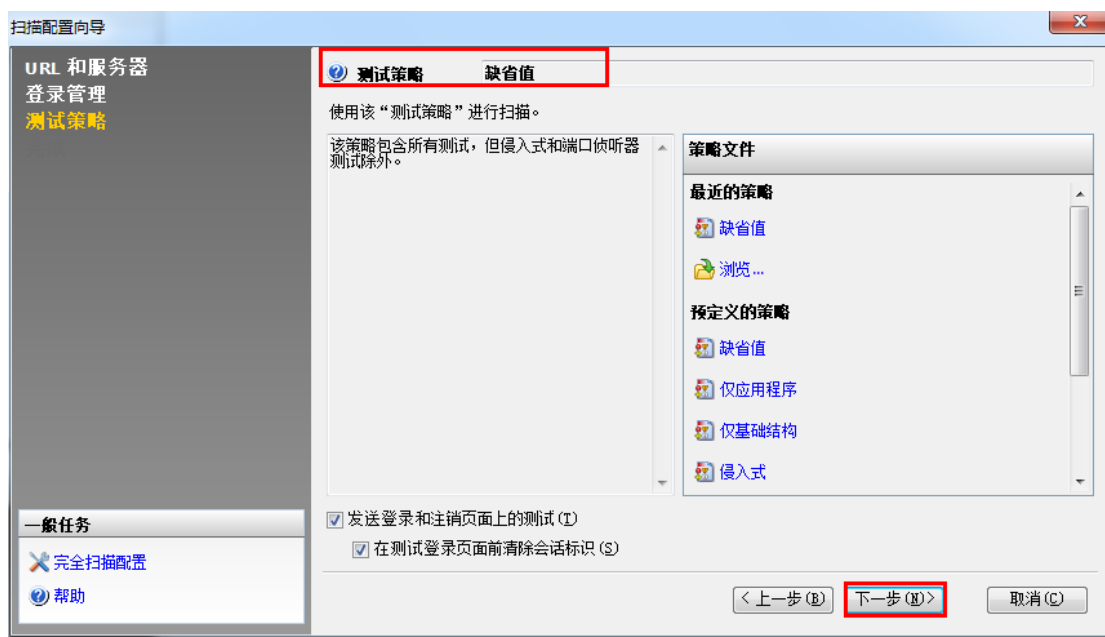


点击“记录”，如下图：

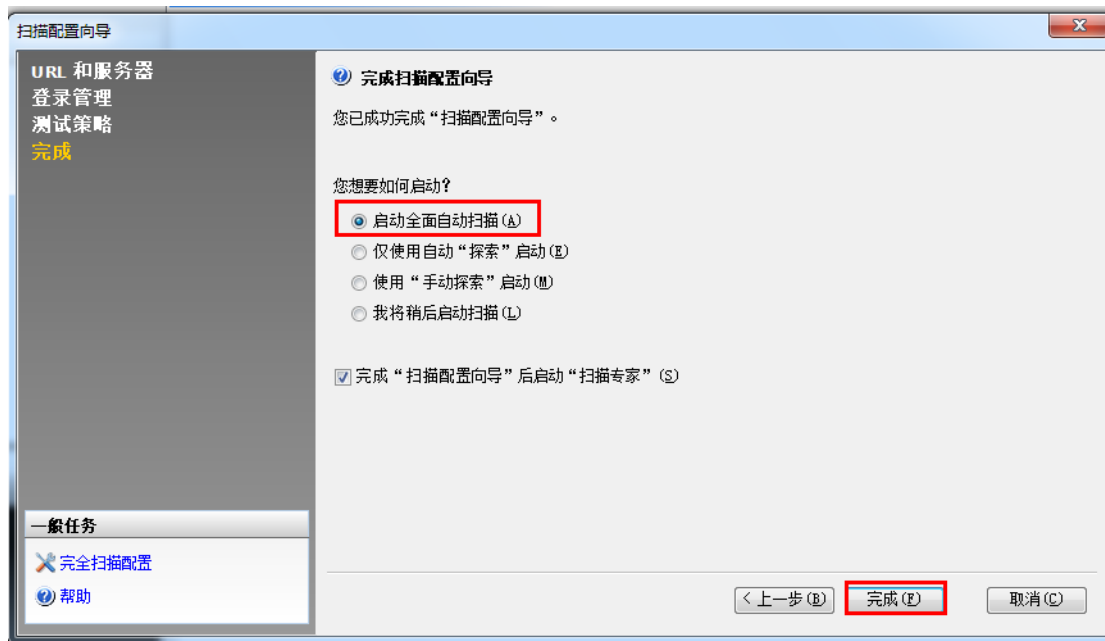


选择“测试策略”为缺省值，如下图：

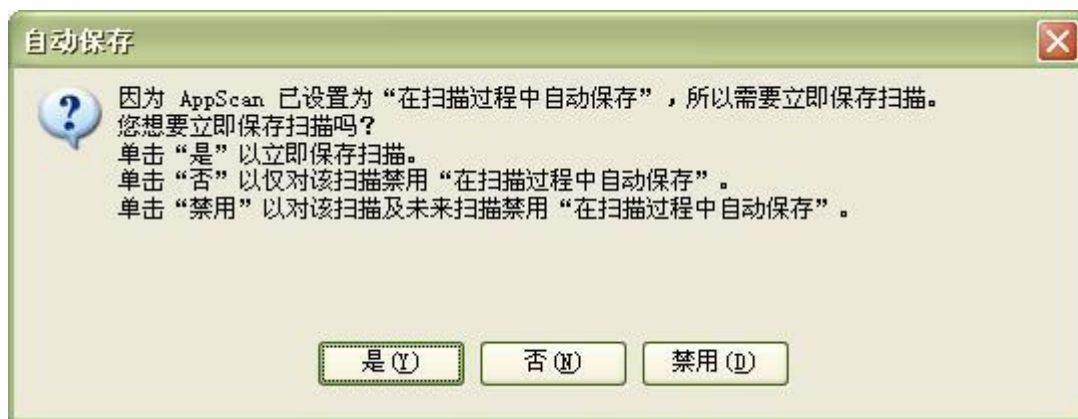
策略名称	描述
缺省值	包含所有测试，但侵入式和端口侦听器测试除外。
仅限应用程序	包含所有应用程序级别的测试，但侵入式和端口侦听器测试除外。
仅限基础结构	包含所有基础结构级别的测试，但侵入式和端口侦听器测试除外。
侵入式	包含所有侵入式测试（可能影响服务器稳定性的测试）。
完成	包含所有 AppScan 测试。
Web Service	包含所有 SOAP 相关测试，但侵入式和端口侦听器测试除外。
关键的少数	包含一些成功可能性极高的测试精选。在时间有限时对站点评估可能有用。
开发者精要	包含一些成功可能性极高的应用程序测试的精选。在时间有限时对站点评估可能有用。



选择启动方式为“启动全面自动扫描”，如下图：



选择自动保存，点击“是”，如下图：



录入该网站的用户名和账号，登录系统后，点击“暂停”按钮，如下图：



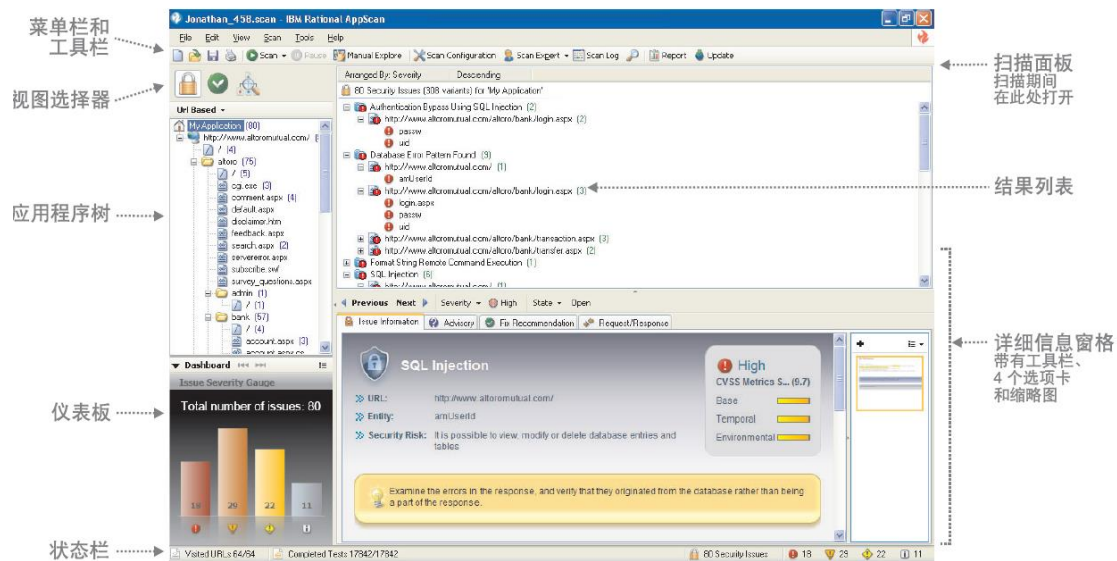
然后，点击“扫描”按钮下的“继续完全扫描”，等待扫描完成，如下图：



扫描完成后，显示结果，如下图：



工具具体分布图，如下图：



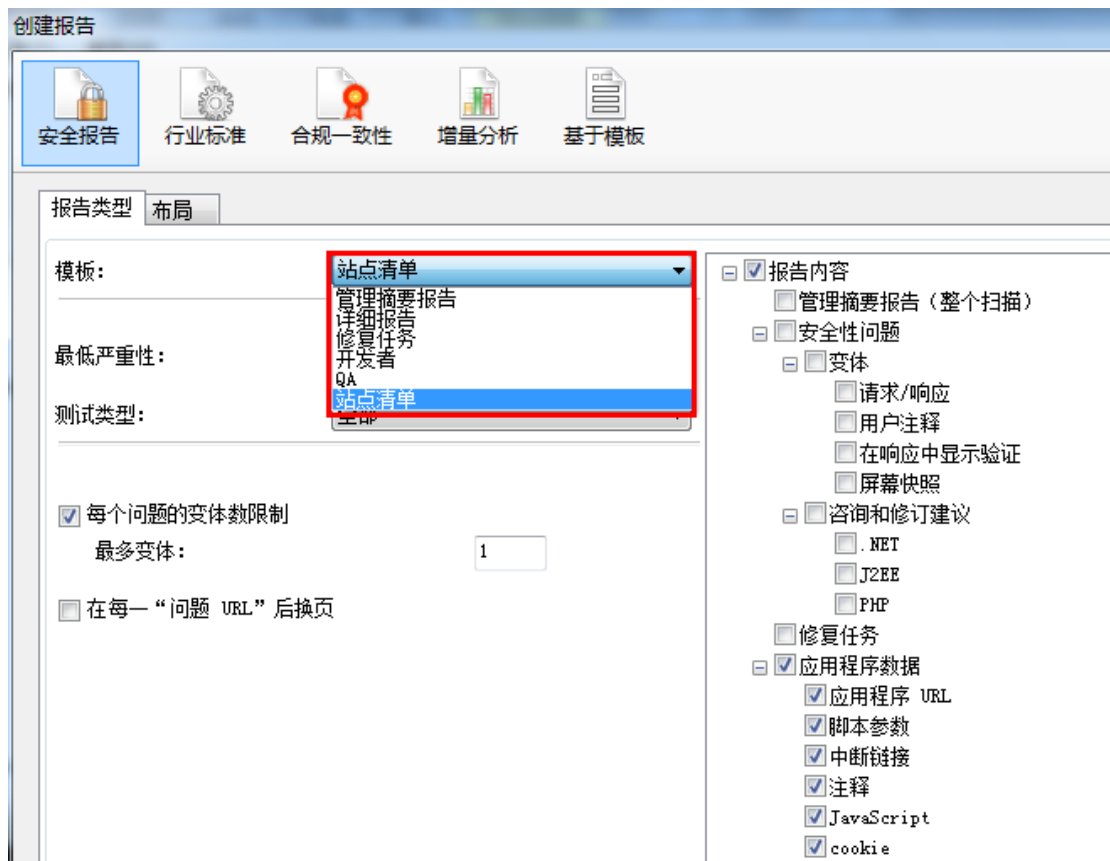
➤ 生成报告

1、在工具栏上，单击“报告”，然后选择“安全报告”。

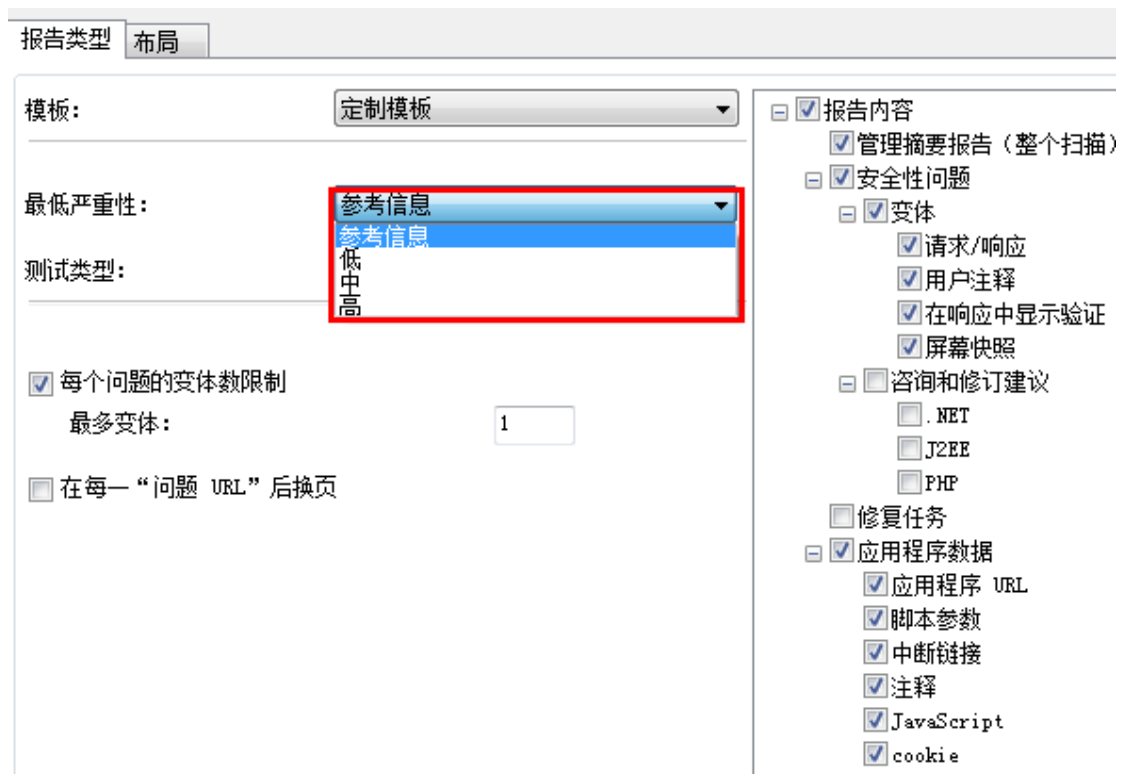


2、选择模板：管理综合报告、详细报告、修复任务、开发者、QA、站点目录。

注：可以通过你所需要的内容，在右侧树中选择你报告所有体现的内容



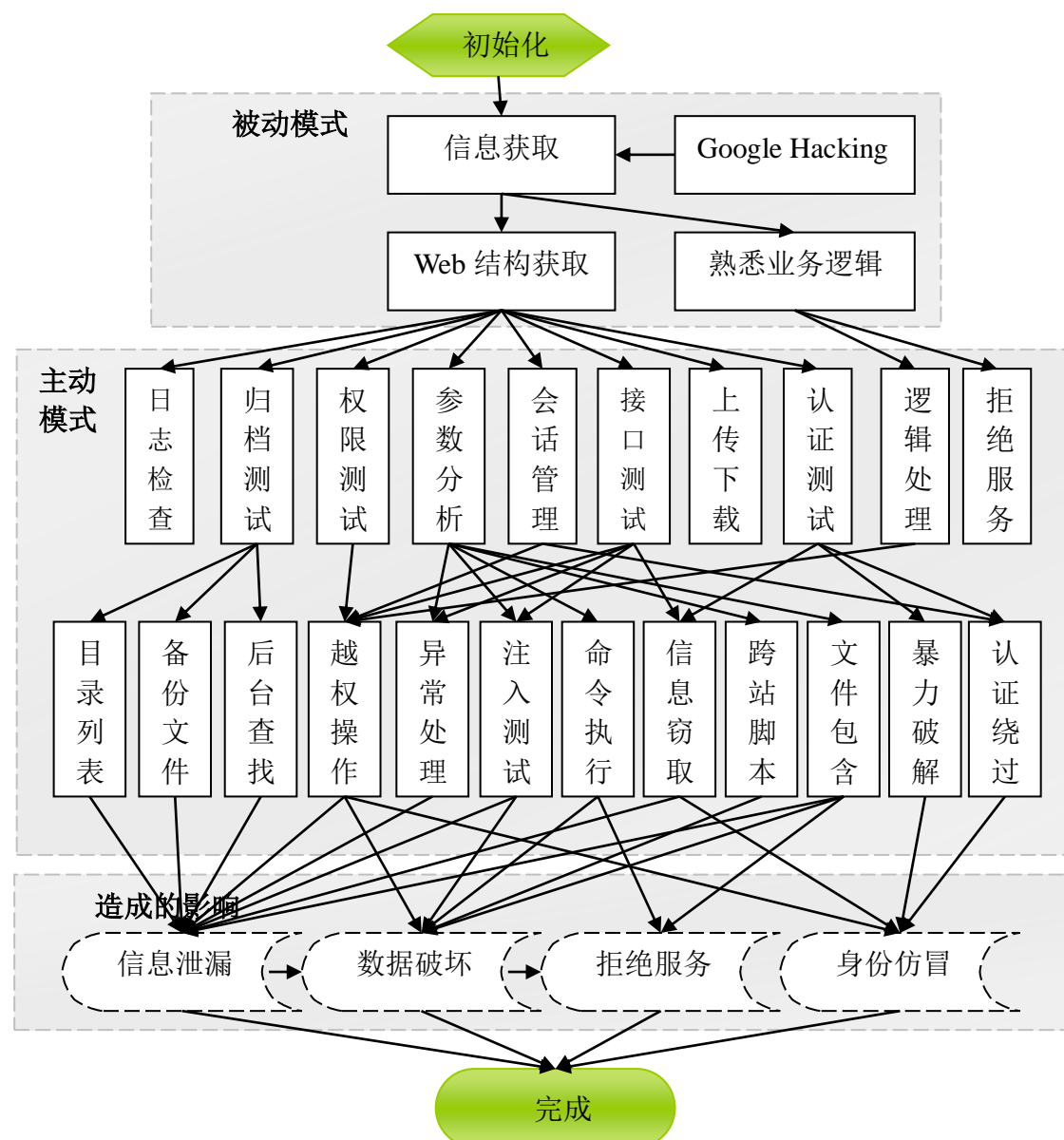
3、从最低严重性列表中，选择要包含在报告中的问题最低严重性级别。



4、点击保存报告，自动生成报告；报告提供 PDA 或 WORD 格式的保存。

4 测试用例和规范标准

测试用例和规范标准可分为主动模式和被动模式两种。在被动模式中，测试人员尽可能的了解应用逻辑：比如用工具（如 HTTPNetworkSniffer.exe, chrome 浏览器）分析所有的 HTTP 请求及响应，以便测试人员掌握应用程序所有的接入点（包括 HTTP 头，参数，cookies 等）；在主动模式中，测试人员试图以黑客的身份来对应用及其系统、后台等进行渗透测试，其可能造成的影响主要是数据破坏、拒绝服务等。一般测试人员需要先熟悉目标系统，即被动模式下的测试，然后再开展进一步的分析，即主动模式下的测试。主动测试会与被测目标进行直接的数据交互，而被动测试不需要，参考以下示意图：



4.1 输入数据测试

4.1.1 SQL 注入测试

SQL 注入是针对一种数据库而言的，而不是针对网页语言。在任何使用了数据库查询环境下都可能存在。常见的数据库包括：MSSQL、Oracle、Informix、Db2、Access、Sybase 等。

所谓 SQL 注入，就是通过把 SQL 命令插入到 Web 表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的 SQL 命令。

SQL 注入受到的威胁，（但不限于）以下几种情况：

- 数据泄漏
- 修改现有数据
- 插入新数据
- 任意的文件系统访问
- 任意的网络访问
- 系统泄漏

针对不同的数据库系统使用的一些函数会有所不同，不过从测试是否存在 SQL 注入的角度考虑，只需要进行几个最基本的判断语句就可以了。

编号	Web_ 01
测试用例名称	手工 SQL 注入测试
测试目的	由于 SQL 注入有可能造成信息泄漏，在严重情况下（根据使用的数据库而定）甚至可能造成数据修改、删除，从而导致业务中断。因此必须发现所有存在的注入点。
用例级别	一级
测试条件	1、 Web业务运行正常 2、 已知待测目标URL，假设为http://www.example.com/page.xxx 3、 待测目标存在参数输入，假设为name=value
执行步骤	1、 观察参数的值value是否为数字型。如果是数字型进行数字型测试，否则跳到第4步进行字符型测试（例如如果出现a那说明是字符型，如果出现2则将其当做数字型测试）

	<p>2、将被测参数后加上测试语句“and 1=1”，即：地址栏中填入“http://www.example.com/page.xxx?name=value and 1=1”，如果返回正确页面则进行下一步操作，否则跳到第4步。</p> <p>3、将被测参数后加上测试语句“and 1=2”（这里以第n个参数为例），其他参数保持不变，即：地址栏中填入“http://www.example.com/page.xxx? name=value and 1=2”，如果返回正确页面则进行下一步操作，否则该参数存在注入漏洞，完成测试</p> <p>4、将被测参数后加上测试语句“’ and ‘1’ =’ 1”，即：地址栏中填入“http://www.example.com/page.xxx? name=value’ and ‘1’ =’ 1”，如果返回正确页面则进行下一步操作，否则该参数存在注入漏洞，完成测试</p> <p>5、将被测参数后加上测试语句“’ and ‘1’ =’ 2”，即：地址栏中填入“http://www.example.com/page.xxx? name=value’ and ‘1’ =’ 2”，如果返回正确页面则不存在漏洞，否则该参数存在注入漏洞，完成测试</p>
预期结果	不存在注入点
备注	<p>1、页面可能接受多个参数，需对每个参数都进行测试</p> <p>2、如果客户端脚本对输入数据进行合法行校验，阻止非法数据，可以通过方法（通过WebScarab拦截并修改参数值），绕过客户端数据校验。</p> <p>3、POST、AJAX以及隐藏域提交参数也需要测试（方法是通过WebScarab拦截HTTP请求，找到提交的参数并参照上面的方法修改参数值）</p> <p>4、本测试包含了现有最常见的两种测试方法</p>
测试结果	

编号	Web _02
测试用例名称	自动化工具 SQL 注入测试
测试目的	由于 SQL 注入有可能造成信息泄漏，在严重情况下（根据使用的数据库而定）甚至可能造成数据修改、删除，从而导致业务中断。因此必须发现所有存在的注入点。
用例级别	一级
测试条件	<p>1、 Web业务运行正常</p> <p>2、 已知待测目标URL，假设为http://www.example.com/page.xxx</p>

	3、待测目标存在参数输入，假设为name=value 4、测试用机安装了pangolin测试工具
执行步骤	1、运行pangolin 2、在URL输入框中输入http://www.example.com/page.xxx?name=value 3、点击Check按钮执行扫描操作 4、观察结果
预期结果	Pangolin 工具不能得到目标服务器的注入类型和数据库类型。
备注	页面可能接受多个参数，需对每个参数都进行测试
测试结果	

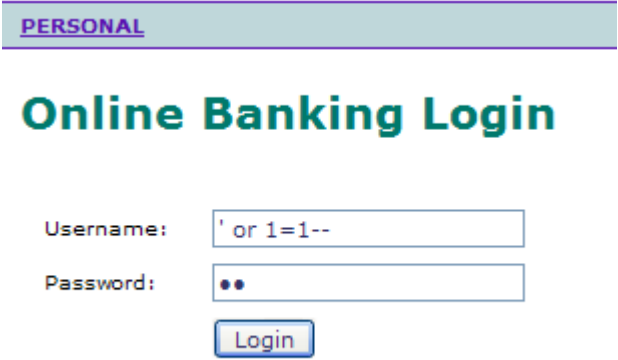
4.1.1.1 SQL 注入实例解析

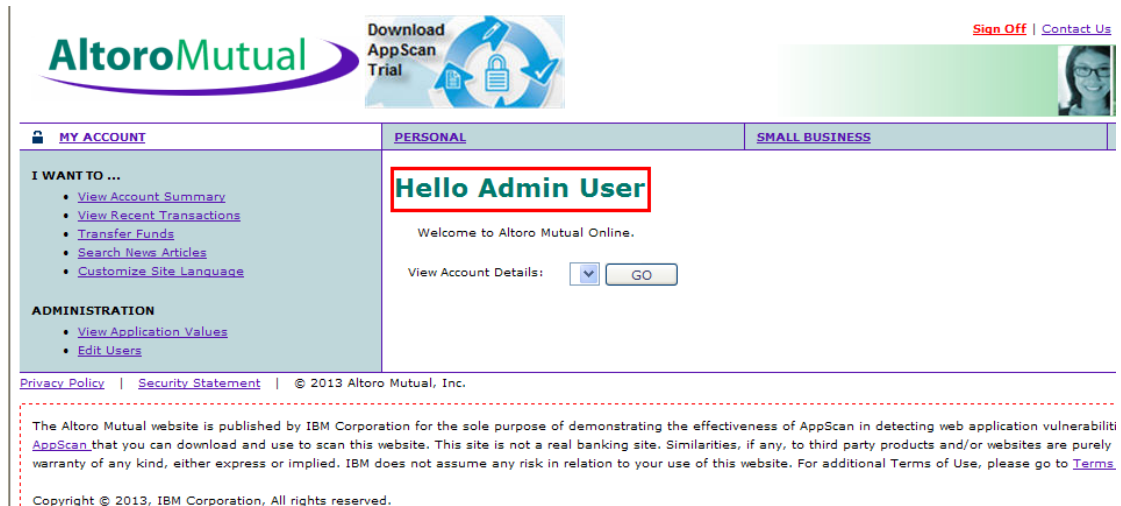
例 1:

网站地址，http://demo.testfire.net（用户名：jsmith，密码：Demo1234）

使用下面的代码，登陆系统的管理员用户

Username= ' or 1=1 -- and password=' demo1234'





例 2:

公司某系统的扫描结果如下:



下图为扫描工具判断, 我们系统存在发出去请求响应包含了 SQL Server 错误, 表明测试所插入的危险字符渗透了应用程序并到达 SQL 查询本身;



如上图, 根据对应的路径, 我们找到对应的文件, 如下图:

```

1 <%@ page language="java" pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>
2 <%@page import="com.szboanda.platform.util.IOMessageNotice"%>
3 <%@page import="com.szboanda.platform.util.permissions.UserPermissions"%>
4 <%
5 //request.setAttribute("INCLUDE_TAGLIB_ONLY",true);
6 request.setAttribute("ssid",session.getId());
7 request.setAttribute("pageTitle", "法律宣传栏");
8 request.setAttribute("SHOW_QUERY_FRAME", true);
9 %>
10 <%@ include file="/pages/ebcm/common/header.jsp"%>
11 <%
12 request.setAttribute("CURRENT_PROCESSER",com.szboanda.platform.util.helper.ActionHelper.getShareId());
13 request.setAttribute("REFRESH_TIMES","20");
14 ActionHelper hepler = ActionHelperAuditor.getInstance().getActionHelper();
15 String depbh=hepler.getUserPermissions().getUser().getDepartment();
16 request.setAttribute("depbh", depbh);
17 String lmbh=request.getParameter("lmbh");
18 StringBuffer sql = new StringBuffer();
19 sql.append("SELECT * FROM T_OA_LM_WZXX WHERE LMBH='"+lmbh+"'and SFFB='1'");
20 String queryString = sql.toString();
21 System.out.println("-----"+queryString);
22 request.setAttribute("querySql",queryString);
23 %>
24 <nbean:query var="lmDyDatainfo" sql="${requestScope.querySql}" scope="request" key="WZBH"
25 orderBy="SJ" way="DESC" paged="true" pvar="P_PAGEINFO_VAR_INDEX" length="8" pscope="request">
26 </nbean:query>
27 <style>
28 <!--
29 body{
30 background: transparent;
31 }
32 a{
33 text-decoration: none;
34 }
35 img {border-width: 0px 0px 0px 0px}
36 -->
37 </style>

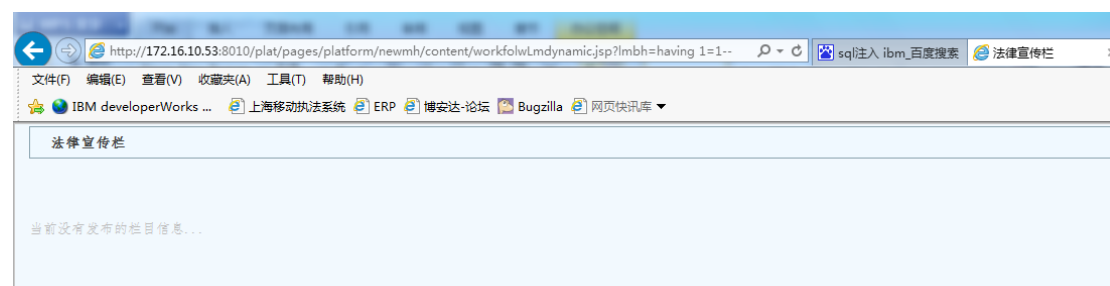
```

当前工具判断“lmbh”这个参数有存在 SQL 注入的风险，存在了 5 种变体



那么，我们可以选择其中一个变体，在浏览器上打开对应的页面，如下图：

<http://172.16.10.53:8010/plat/pages/platform/newmh/content/workfolwLmdynamic.jsp?lmbh=b844791e-ad96-4c45-a65b-4d76e857b811%a5'%20having%201=1-->



所以，在扫描工具中，会判断可能会导致 SQL 注入的可能原因，开发人员可以通过工具的提示信息，进行修改；



SQL 注入

- 严重性: 高
- 类型: 应用程序级别测试
- WASC 威胁分类: [SQL 注入](#)
- CVE 标识: 不适用
- CWE 标识: [89](#)
- 安全风险: 可能会查看、修改或删除数据库条目和表

可能原因

未对用户输入正确执行危险字符清理

4.1.2 命令执行测试

编号	Web _03
测试用例名称	命令执行测试
测试目的	某些页面可能接受类似于文件名的参数用于下载或者显示内容。
用例级别	1
测试条件	1、 Web业务运行正常 2、 已知某页面URL（假设为http://www.example.com/abc.jsp）接收参数，且参数中接收类似于系统命令的字符（假设为cmd=ls）
执行步骤	1、 更改参数的值为其他命令，可以尝试以下一些字符串： net user ipconfig cat /etc/passwd 2、 观察页面返回信息。比如使用net user命令的话，页面中可能包含类似于如下的字符串： \\host 的用户帐户 Administrator
预期结果	页面的源代码中不包含类似于系统命令的返回信息。

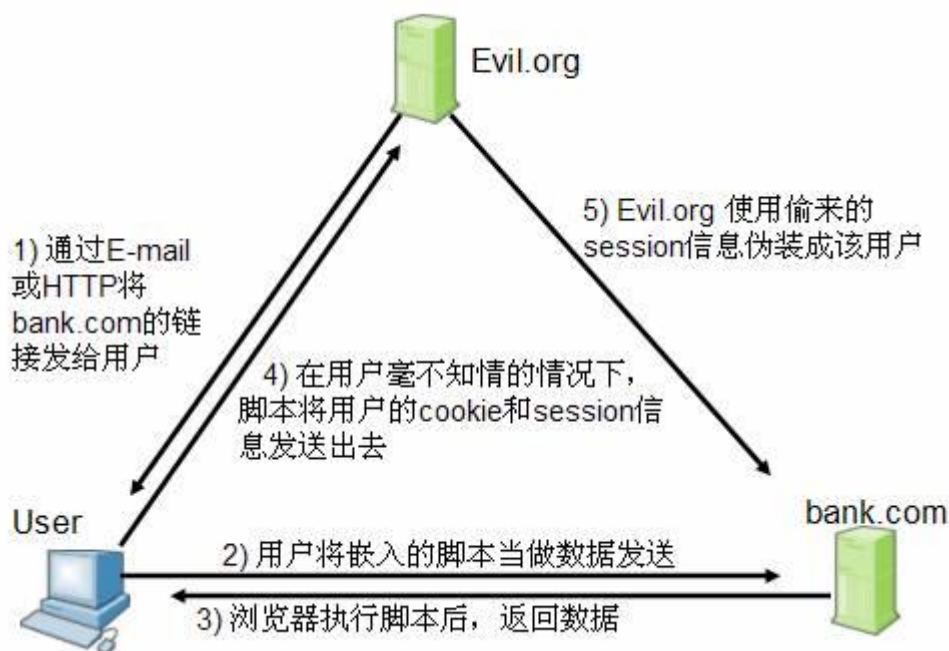
备注	
参考资料	

4.2 跨站脚本攻击测试

“跨站点脚本编制”攻击是一种隐私违例，可让攻击者获取合法用户的凭证，并在与特定 Web 站点交互时假冒这位用户。

跨站点脚本编制受到的威胁，（但不限于）以下几种情况：

- 当用户查看基于攻击者提供的内容而动态生成的页面时，他们会不知不觉地执行恶意脚本；
- 在用户的会话 cookie 失效之前，攻击者就能接管用户会话；
- 攻击者可以将用户连接到攻击者选择的恶意服务器上；
- 攻击者诱导用户访问由攻击者提供的 URL，从而导致在用户的浏览器中执行攻击者选择的脚本或 HTML。通过使用这种技术，攻击者可以使用访问过此 URL 的用户的特权来采取行动，诸如对底层 SQL 数据库发出查询并查看其结果。



- 在上图中，恶意攻击者（这里使用 Evil.org 表示）通过 E-mail 或 HTTP 将某银行的网址链接发给用户（银行用 bank.com 表示），该链接中附加了恶意的脚本（上图步骤一）；

- 用户访问发来的链接，进入银行网站，同时，嵌在链接中的脚本被用户的浏览器执行（上图步骤二、三）；
- 用户在银行网站的所有操作，包括用户的 cookie 和 session 信息，都被脚本收集到，并且在用户毫不知情的情况下发送给恶意攻击者（上图步骤四）；
- 恶意攻击者使用偷来的 session 信息，伪装成该用户，进入银行网站，进行非法活动（上图步骤五）。
- 因此，只要 Web 应用中，有可被恶意攻击者利用执行脚本的地方，都存在极大的安全隐患。黑客们如果可以让用户执行他们提供的脚本，就可以从用户正在浏览的域中偷到他的个人信息、可以完全修改用户看到的页面内容、跟踪用户在浏览器中的每一个动作，甚至利用用户浏览器的缺陷完全控制用户的机器。

4.2.1 GET 方式跨站脚本测试

编号	Web_XSS_01
测试用例名称	GET 方式跨站脚本测试
测试目的	由于跨站脚本会导致会话被劫持、敏感信息泄漏、账户被盗，严重时甚至造成数据修改、删除，从而导致业务中断，因此需检测跨站脚本是否存在
用例级别	一级
测试条件	1、 Web业务运行正常 2、 已知待测目标URL，假设为http://www.example.com/page.xxx 3、 待测目标存在参数输入，假设为name=value 4、 在某种情况下，用户输入被重新显示在网页上，包括名字、帐号、检索结果等等（说明目标网站服务器并没有对用户提交数据检测）
执行步骤	1、 在 输入 的 参 数 后 逐 条 添 加 以 下 语 句 ， 以 第 一 条 为 例 ， 输 入 http://www.example.com/page.xxx?name=<script>alert(123456)</script> 只 要 其中一条弹出显示123456的告警框，就说明存在跨站漏洞，记录漏洞，停止测试。 2、 如果没有弹出显示123456的告警框，则在返回的页面上单击鼠标右键，选择“查看

	<p>源文件”</p> <p>3、 查找网页源文件中是否包含完整的字符串<script>alert(123456)</script>, 则不管有没有弹出显示123456的告警框, 都表明存在跨站脚本漏洞。</p> <p>由于有些 HTML 元素 (比如<textarea>或”) 会影响脚本的执行, 所以不一定能够正确弹出 123456 告警框, 需要根据返回网页源文件的内容, 构造 value 的值, 比如</p> <p>多行文本输入框:</p> <pre></textarea><script>alert(123456)</script></pre> <p>文本输入框:</p> <pre></td><script>alert(123456)</script></pre> <pre>'><script>alert(123456)</script></pre> <pre>"><script>alert(123456)</script></pre> <pre></title><script>alert(123456)</script></pre> <pre>--><script>alert(123456)</script></pre> <pre>[img]javascript:alert(123456)[/img]</pre> <pre><scrip<script>t>alert(123456)</scrip</script>t></pre> <pre></div><Script>alert(123456)</script></pre>
预期结果	不存在跨站脚本漏洞
备注	<p>需要对页面上所有可以提交参数的地方进行测试。具体跨站脚本的测试语句根据实际情况的不同而不同, 这里列出了一些最常见构造语句。</p> <p>AppScan 可以找出扫描到的页面的绝大部分跨站脚本漏洞, 但对没有扫描到的网页就无能为力了。</p>
测试结果	

4.2.2 POST 方式跨站脚本测试

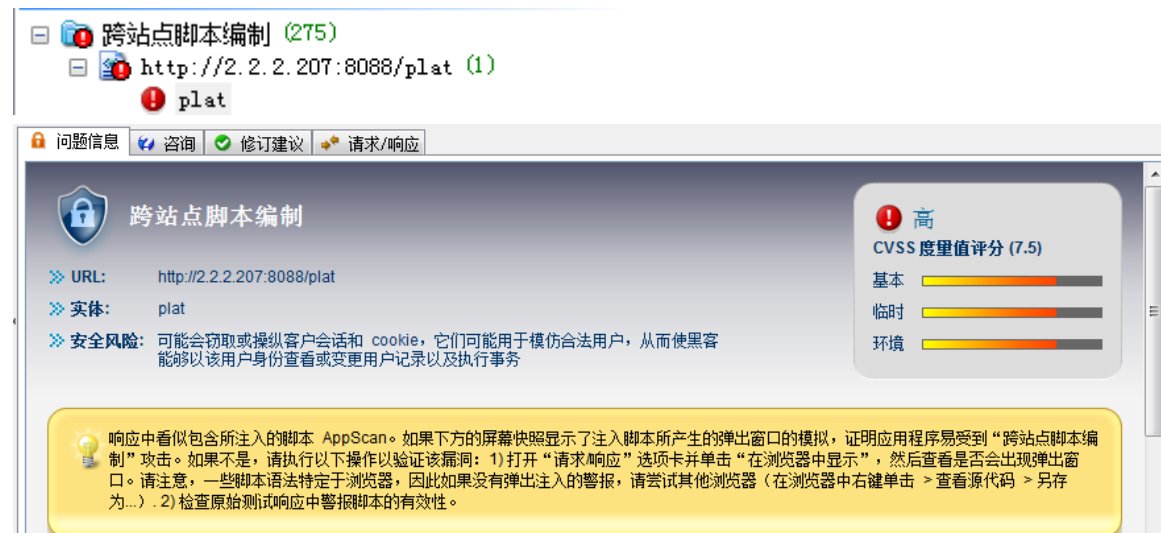
编号	Web_ XSS_02
测试用例名称	POST 方式跨站脚本测试
测试目的	由于跨站脚本会导致会话被劫持、敏感信息泄漏、账户被盗, 严重时甚至造成数据修改、

	删除，从而导致业务中断，因此需检测跨站脚本是否存在
用例级别	一级
测试条件	<p>1、 Web业务运行正常</p> <p>2、 已知待测目标URL，假设为http://www.example.com/page.xxx</p> <p>3、 待测目标以POST方式提交参数，显示为表单方式</p> <p>4、 在某种情况下，用户输入被重新显示在网页上，包括名字、帐号、检索结果等等（说明目标网站服务器并没有对用户提交数据检测）</p>
执行步骤	<p>1、 在POST表单中逐条输入以下语句，只要其中一条弹出显示123456的对话框，就说明存在跨站漏洞，记录漏洞，停止测试。</p> <pre><script>alert(123456)</script></pre> <pre>[img].javascript:alert(123456);[/img]</pre> <p>2、 如果没有弹出显示123456的告警框，则在返回的页面上单击鼠标右键，选择“查看源文件”</p> <p>3、 查找网页源文件中是否包含完整的字符串<script>alert(123456)</script>，则不管有没有弹出显示123456的告警框，都表明存在跨站脚本漏洞。</p> <p>由于有些 HTML 元素（比如<textarea>或”）会影响脚本的执行，所以不一定能够正确弹出 123456 告警框，需要根据返回网页源文件的内容，构造 value 的值，比如</p> <pre></textarea><script>alert(123456)</script></pre> <hr/> <pre>'><script>alert(123456)</script></pre> <pre>"><script>alert(123456)</script></pre> <pre></title><script>alert(123456)</script></pre> <pre>--><script>alert(123456)</script></pre> <pre>[img].javascript:alert(123456)[/img]</pre> <pre><scrip<script>t>alert(123456)</scrip</script>t></pre> <pre></div><Script>alert(123456)</script></pre>
预期结果	不会弹出显示 123456 的对话框。
备注	需要对页面上所有可以提交参数的地方进行测试。
测试结果	

4.2.3 跨站脚本工具实例解析

例：

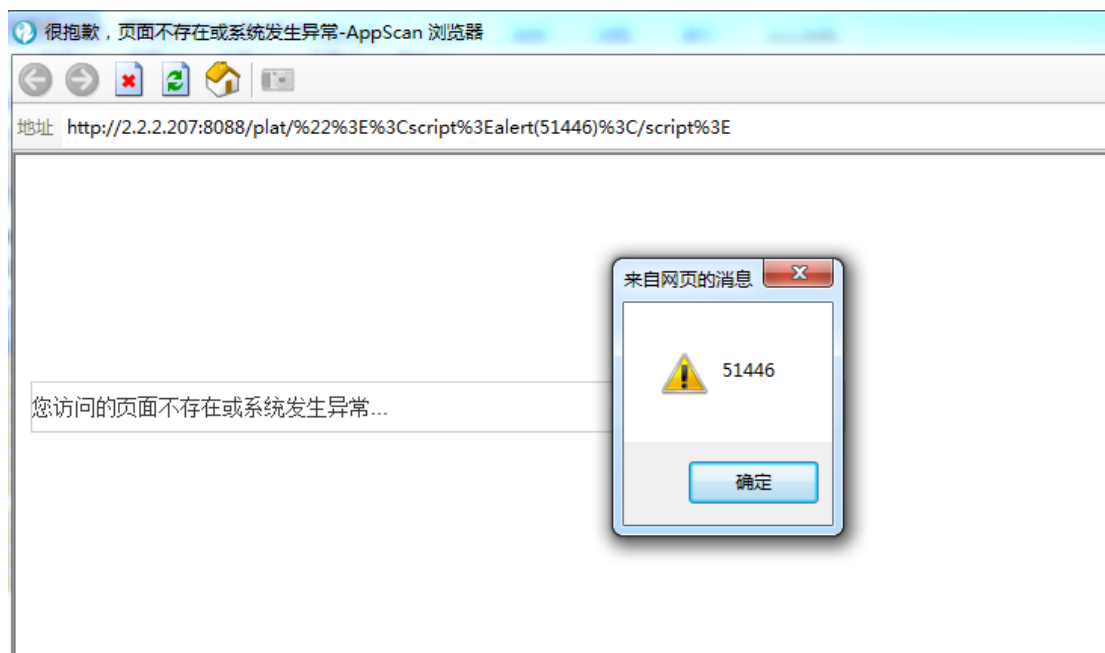
公司某系统的扫描结果如下（获取 cookie 信息）：



下图为扫描工具判断, 我们系统开发代码中可能出现下面的代码语句 (具体可以打开对应的文件查看), 这种代码可能会让黑客进行跨站点攻击:

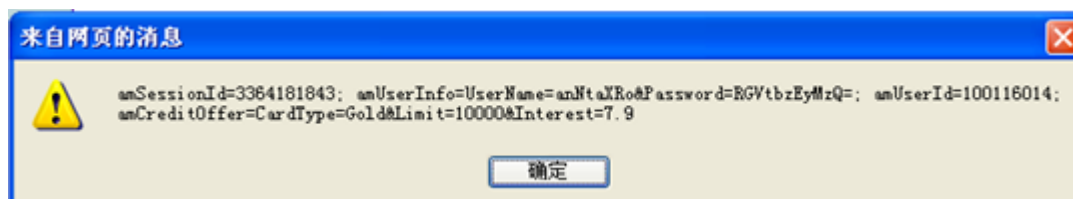


如上图, `/plat/"><script>alert(51446)</script>` 为修改后的代码内容, 如果在浏览器上展示, 如下图:



这样黑客也可以修改为:

`/plat/" ><script>alert(document.cookie)</script>`, 来获取我们系统中的 cookie 信息, 如下图:



从上图弹出的 cookie 信息中, 黑客可以通过获取 sessionID 模拟合法用户 (如果需要弹出该提示, 需要修改 IE 浏览器的设置, 关闭 IE “internet 选项-安全-脚本-自定义级别-脚本-禁用 XSS 筛选器”)。

所以, 在扫描工具中, 会判断可能会导致跨站点脚本工具的可能原因, 开发人员可以通过工具的提示信息, 进行修改;



4.3 权限管理测试

4.3.1 横向测试

编号	Web_01
测试用例名称	基于用户身份处理的横向越权操作测试
测试目的	发现页面中存在的横向越权操作。
用例级别	一级 L
测试条件	1、 Web业务运行正常 2、 Web业务存在身份级别控制 3、 已知某页面（假设为http://www.example.com/abc.jsp）.提交的参数中存在着代表用户（假设为userA）身份的标志（假设为operator） 4、 与userA同级别权限的用户userB 5、 测试用机安装了WebScarab软件
执行步骤	1、 运行WebScarab 2、 点击Proxy标签页->Manual Edit标签页 3、 选中Intercept requests 4、 打开浏览器，在代理地址中配置host为127.0.0.1，port为8008 5、 使用userA的身份登录到Web应用 6、 进入http://www.example.com/abc.jsp页面，提交数据 7、 在弹出的对话框中的URLEncoded页面中，更改operator参数的值为userB，再点击Accept Changes按钮提交 8、 观察服务器处理
预期结果	服务器返回操作失败或者以 userA 的用户身份操作。
备注	如果参数是基于 GET 方式的 URL 传递，则不需要通过 WebScarab 工具，直接在 URL 中进行修改提交即可。
参考资料	

4.3.2 纵向测试

横向测试的两个用例在本测试类别中同样使用，只需要对用户身份进行测试时使用上下级权限即可。

在下面的测试中，我们更偏向于使用白盒测试。这样对于测试人员来说节约了非常多的时间。而且也能够全面覆盖所有的页面。

编号	Web _03
测试用例名称	基于菜单 URL 的测试
测试目的	发现应用中存在的 URL 纵向越权操作。
用例级别	一级
测试条件	1、 Web业务运行正常 2、 Web业务存在身份级别控制 3、 拥有超级管理员及普通用户的帐号和密码
执行步骤	1、 以超级管理员身份登陆Web网站 2、 单击鼠标右键，选择“查看源文件” 3、 在网页“源文件”中查找重要的管理菜单（比如用户管理）的URL链接，并拷贝URL链接地址 4、 退出登陆 5、 以普通用户身份登陆Web网站 6、 在浏览器地址栏中输入“用户管理”的URL地址（如http://www.example.com/usermanage.do），然后回车 7、 观察普通用户是否能够顺利进入“用户管理”页面，并进行用户管理操作。
预期结果	普通用户不能够通过直接URL访问、使用未授权的功能。
备注	
测试结果	

编号	Web _04
测试用例名称	基于爬行的测试

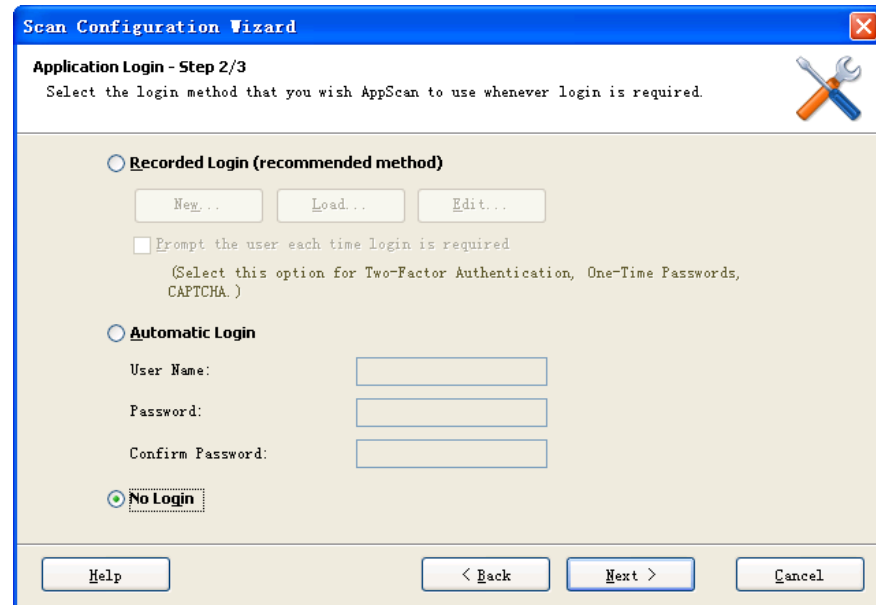
测试目的	发现页面中存在的纵向越权操作。
用例级别	一级
测试条件	1、 Web业务运行正常 2、 Web业务存在身份级别控制 3、 已知待测目标URL，假设为http://www.example.com/page.xxx 4、 测试用机上安装了AppScan
执行步骤	1、 双击运行AppScan，选择file—new新建扫描，选择扫描模板default 2、 弹出扫描配置对话框，选择扫描类型，默认为Web Application Scan，点击next 3、 在Starting URL中填入需扫描的目标服务器域名或IP地址，其他配置不需修改，点击next 4、 选择默认的Recorded Login(recommended method)，点击New 5、 在弹出的页面中用权限较高的用户身份登录，如：admin等 6、 关闭页面，弹出如下对话框，点击OK <div data-bbox="451 1019 1300 1966" data-label="Image"> </div> 7、 不需修改任何参数，点击next

8、不需修改参数，选择Start a full automatic scan，点击finish完成配置，开始扫描

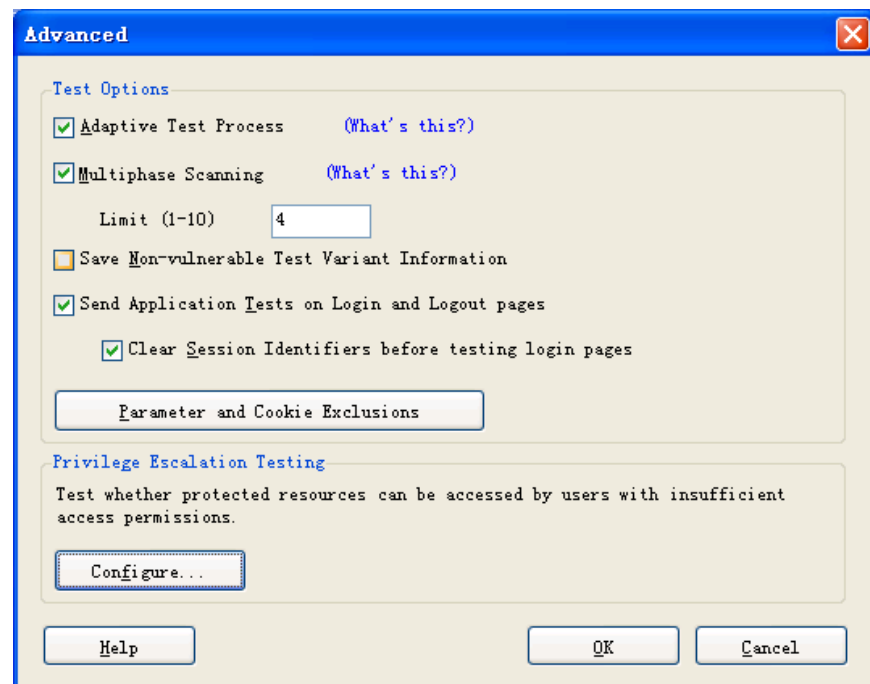
9、扫描完成，保存扫描结果，假设命名为admin. scan

10、重新开始扫描，重复步骤1、2、3

11、在选择用户身份（即：第4步）时，选择NoLogin，点击next

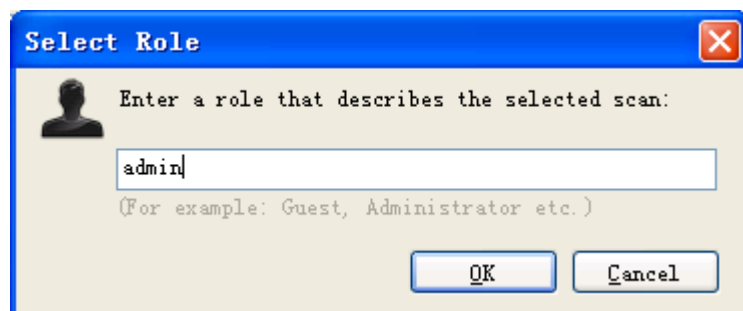
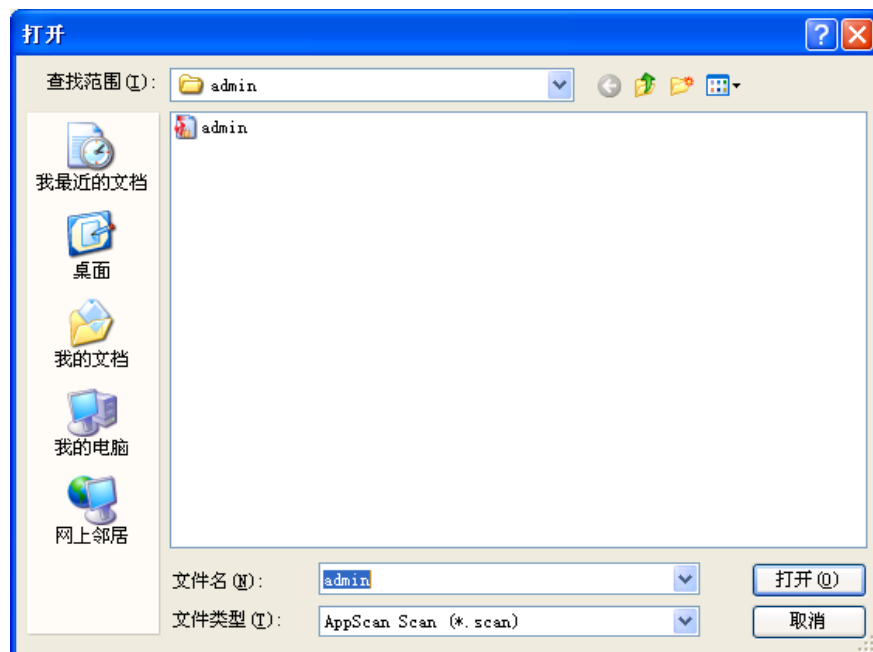
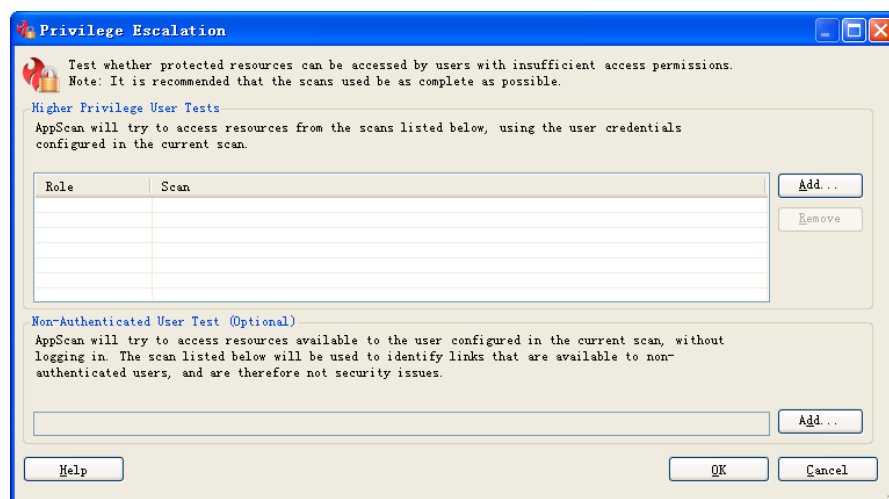


12、在弹出的配置对话框中（即：第7步的对话框）选择Advanced Test Settings，弹出下面对话框，点击Configure选项



13、在Privilege Escalation中，点击Add添加已经扫描出的结果；点击打开导入扫描

结果；并用一个名称标记它，如：admin，点击ok



14、然后点击ok返回扫描配置对话框，如第7步的图示，点击next

15、到第8步所示对话框中，不需修改参数，点击finish开始做越权扫描

16、扫描完成，保存结果

17、分析扫描结果

预期结果	扫描结果中不会提示存在漏洞。
备注	
参考资料	

4.4 服务器信息收集

4.4.1 运行账号权限测试

编号	Web_01
测试用例名称	运行帐号权限测试
测试目的	运行 Web 服务器的操作系统帐号权限越高，那么 Web 遭到攻击产生的危害就越大。因此，不应使用“root”、“administrator”、等特权帐号或高级别权限的操作系统帐号来运行 Web，应该尽可能地使用低级别权限的操作系统帐号。
用例级别	一级
测试条件	1、 已知Web网站IP地址和登陆帐号、密码
执行步骤	1、 登陆Web服务器操作系统 2、 查看运行Web服务器的操作系统帐号，不是“root”、“administrator”等特权帐号或高级别权限帐号，如果是则存在漏洞。 3、 window：打开任务管理器，选择“进程”页，勾选左下方的“显示所有用户的进程”，检查运行Web服务器的帐号；
预期结果	没有使用“root”、“administrator”等特权操作系统帐号运行 Web。
备注	
测试结果	

4.4.2 Web 服务器端口扫描

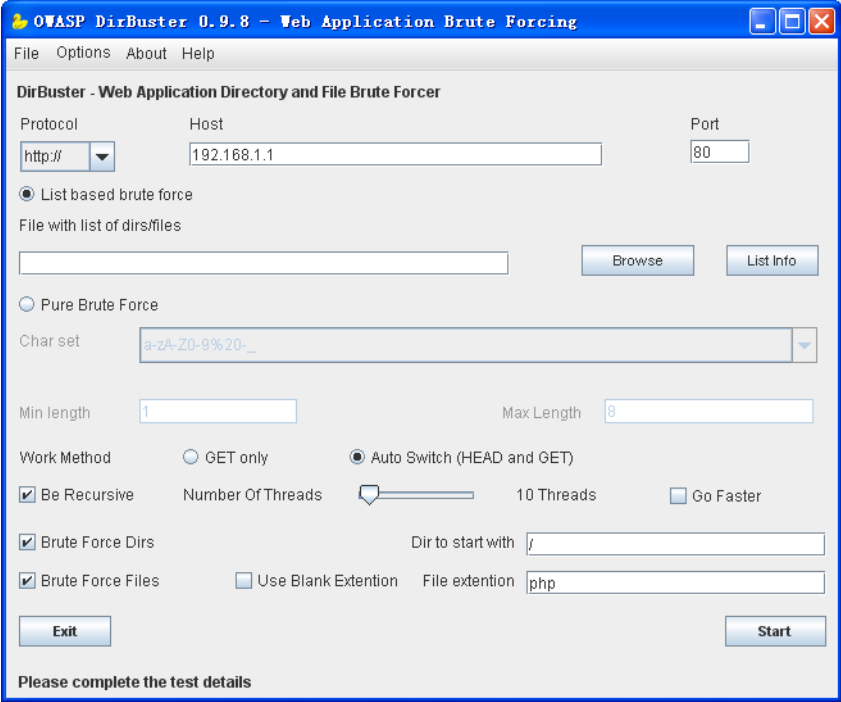
编号	Web _02
测试用例名称	Web 服务器端口扫描
测试目的	有时 Web 应用服务器除业务端口外还会开放一些默认端口，这些默认端口对最终用户

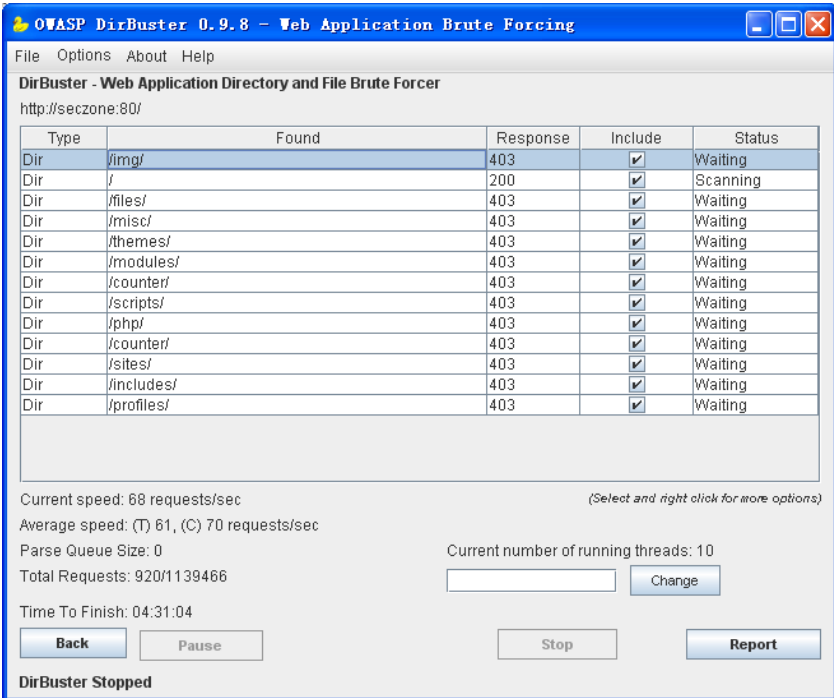
	是不需要开放的，而且也不会用于维护，容易被攻击，本测试目的在于发现服务器上未使用的 Web 端口。
用例级别	一级
测试条件	1、 已知Web服务器域名或IP地址，假设IP地址为2.2.2.1 2、 测试用机安装了nmap，假设路径为d:\nmap
执行步骤	1、 运行如下命令：Nmap -sP 2.2.2.1，来判断目标主机 Windows Server A 是否可连通 2、 使用常规扫描方式对目标主机进行 TCP 端口扫描，运行如下命令：Nmap -sT 2.2.2.1 3、 观察结果，看是否为必须开放的Web服务端口。
预期结果	系统未开放业务不需要使用的端口。
备注	各种参数扫描请参考《利用 nmap 进行端口扫描》
测试结果	

4.5 文件、目录测试

4.5.1 工具方式的敏感接口遍历

编号	Web_01
测试用例名称	工具方式的敏感接口遍历
测试目的	网站目录查找是进行攻击的必备知识，只有知道了目录信息才能确定攻击的目标，进行目录查找是测试的首要阶段，一般扫描工具进行扫描前首先要进行目录查找。其次对于某些隐藏的管理接口（目录或文件），虽然没有对外有明显的链接，但是通过一系列有特定含义的枚举是可以访问的。
用例级别	二级
测试条件	1、 Web业务运行正常 2、 已知目标网站的域名或IP地址 3、 测试用机上需安装JRE 4、 测试用机上有DirBuster软件

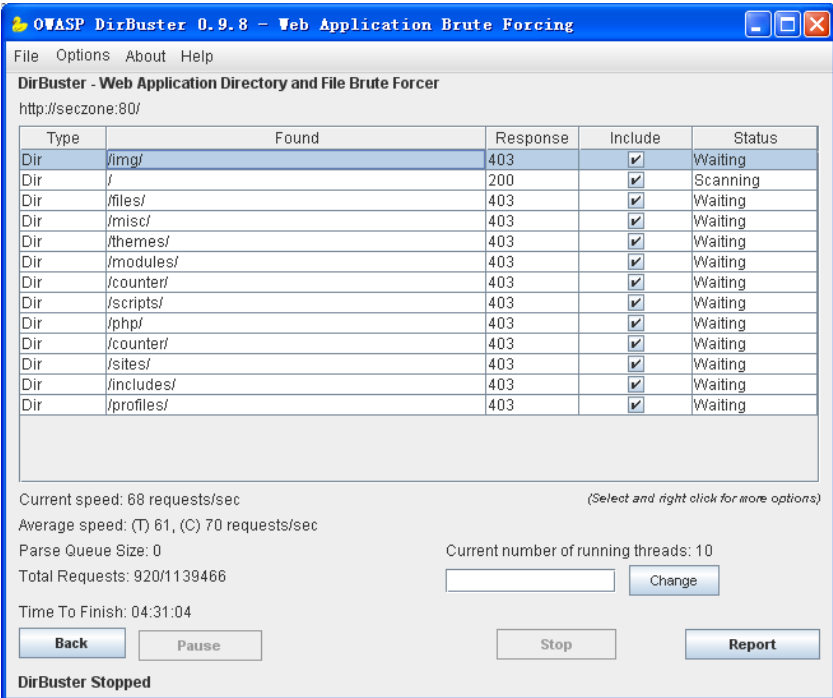
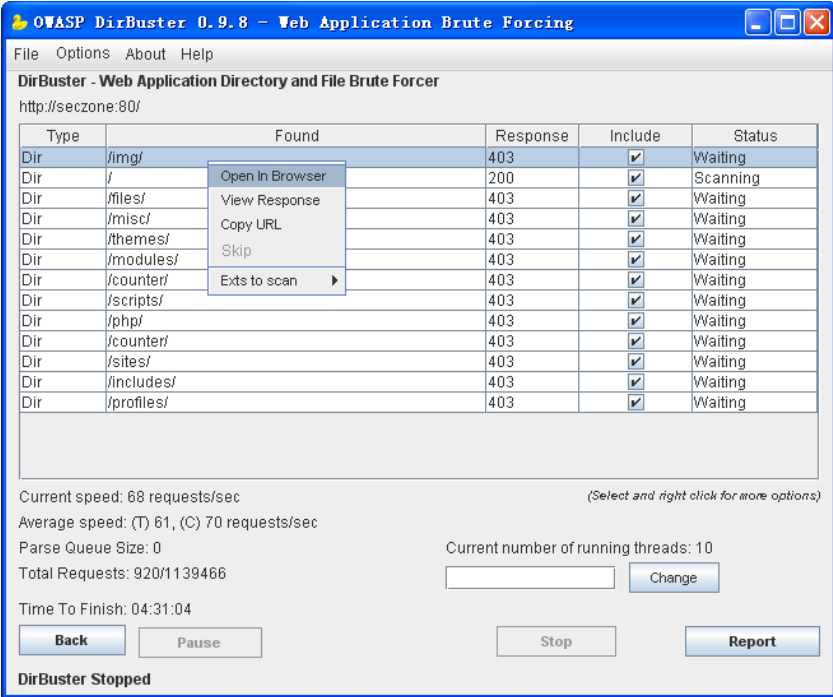
<p>执行步骤</p>	<div><div><div>1、 双击运行DirBuster. jar</div><div>2、 在host栏中填入目标IP地址或域名，在Port栏中输入服务器对应的端口；如果服务器只接受HTTPS请求，则需要选择Protocol为HTTPS</div></div><div></div><div><div>3、 在file with list of dirs/files 栏后点击browse，选择破解的字典库为 directory-list-2.3-small.txt</div><div>4、 将File extension中填入正确的文件后缀，默认为php，如果为jsp页面，需要填入jsp</div><div>5、 其他选项不变，点击右下角的start，启动目录查找</div></div></div>
-------------	--

	<div><p>6、观察返回结果，可点击右下角的report，生成目录报告</p></div>						
预期结果	经过分析以后的结果中，业务系统不存在不需要对外开放的敏感接口，或者该接口进行了完善的权限控制。						
备注	<p>举一个测试不通过的例子：</p> <table><tr><th>Type</th><th>Found</th><th>Response</th></tr><tr><td>File</td><td>/admin/adduser.jsp</td><td>200</td></tr></table>	Type	Found	Response	File	/admin/adduser.jsp	200
Type	Found	Response					
File	/admin/adduser.jsp	200					
测试结果							

4.5.2 目录列表测试

编号	SEC_Web_DIR_04
测试用例名称	目录列表测试
测试目的	目录列表能够造成信息泄漏，而且对于攻击者而言是很容易进行的。所以在测试过程中，我们应当找出所有的目录列表漏洞。
用例级别	1
测试条件	1、 Web业务运行正常 2、 已知目标网站的域名或IP地址

	<div>3、 测试用机上需安装JRE</div> <div>4、 测试用机上有DirBuster软件</div>
执行步骤	<div>1、 双击运行DirBuster-0.9.8.jar</div> <div>2、 在host栏中填入目标IP地址或域名，在Port栏中输入服务器对应的端口；如果服务器只接受HTTPS请求，则需要选择Protocol为HTTPS</div> <div></div> <div>3、 在file with list of dirs/files 栏后点击browse，选择破解的字典库为directory-list-2.3-small.txt:</div> <div>4、 去除Burte Force Files选项</div> <div>5、 其他选项不变，点击右下角的start，启动目录查找</div>

	<div><p>OWASP DirBuster 0.9.8 - Web Application Brute Forcing</p><p>DirBuster - Web Application Directory and File Brute Forcer</p><p>http://seczone:80/</p><table><tr><th>Type</th><th>Found</th><th>Response</th><th>Include</th><th>Status</th></tr><tr><td>Dir</td><td>/img/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/</td><td>200</td><td><input checked="" type="checkbox"/></td><td>Scanning</td></tr><tr><td>Dir</td><td>/files/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/misc/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/themes/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/modules/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/counter/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/scripts/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/php/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/counter/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/sites/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/includes/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/profiles/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr></table><p>Current speed: 68 requests/sec (Select and right click for more options)</p><p>Average speed: (T) 61, (C) 70 requests/sec</p><p>Parse Queue Size: 0</p><p>Total Requests: 920/1139466</p><p>Current number of running threads: 10</p><p>Time To Finish: 04:31:04</p><p>Buttons: Back, Pause, Stop, Report</p><p>DirBuster Stopped</p></div>	Type	Found	Response	Include	Status	Dir	/img/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/	200	<input checked="" type="checkbox"/>	Scanning	Dir	/files/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/misc/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/themes/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/modules/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/counter/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/scripts/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/php/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/counter/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/sites/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/includes/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/profiles/	403	<input checked="" type="checkbox"/>	Waiting
Type	Found	Response	Include	Status																																																																			
Dir	/img/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/	200	<input checked="" type="checkbox"/>	Scanning																																																																			
Dir	/files/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/misc/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/themes/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/modules/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/counter/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/scripts/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/php/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/counter/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/sites/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/includes/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/profiles/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
	6、依次右击Response值为200的行，在出现的菜单中点击Open In Browser																																																																						
	<div><p>OWASP DirBuster 0.9.8 - Web Application Brute Forcing</p><p>DirBuster - Web Application Directory and File Brute Forcer</p><p>http://seczone:80/</p><table><tr><th>Type</th><th>Found</th><th>Response</th><th>Include</th><th>Status</th></tr><tr><td>Dir</td><td>/img/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/</td><td>200</td><td><input checked="" type="checkbox"/></td><td>Scanning</td></tr><tr><td>Dir</td><td>/files/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/misc/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/themes/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/modules/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/counter/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/scripts/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/php/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/counter/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/sites/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/includes/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr><tr><td>Dir</td><td>/profiles/</td><td>403</td><td><input checked="" type="checkbox"/></td><td>Waiting</td></tr></table><p>Current speed: 68 requests/sec (Select and right click for more options)</p><p>Average speed: (T) 61, (C) 70 requests/sec</p><p>Parse Queue Size: 0</p><p>Total Requests: 920/1139466</p><p>Current number of running threads: 10</p><p>Time To Finish: 04:31:04</p><p>Buttons: Back, Pause, Stop, Report</p><p>DirBuster Stopped</p></div>	Type	Found	Response	Include	Status	Dir	/img/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/	200	<input checked="" type="checkbox"/>	Scanning	Dir	/files/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/misc/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/themes/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/modules/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/counter/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/scripts/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/php/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/counter/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/sites/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/includes/	403	<input checked="" type="checkbox"/>	Waiting	Dir	/profiles/	403	<input checked="" type="checkbox"/>	Waiting
Type	Found	Response	Include	Status																																																																			
Dir	/img/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/	200	<input checked="" type="checkbox"/>	Scanning																																																																			
Dir	/files/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/misc/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/themes/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/modules/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/counter/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/scripts/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/php/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/counter/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/sites/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/includes/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
Dir	/profiles/	403	<input checked="" type="checkbox"/>	Waiting																																																																			
	7、分析结果																																																																						
预期结果	所有对目录的访问均不能打印出文件列表。																																																																						
备注																																																																							
测试结果																																																																							

4.5.3 文件归档测试

编号	Web _05
测试用例名称	文件归档测试
测试目的	在网站管理员的维护过程中，很多情况下会对程序或者页面进行备份（可能是有意的或者是无意的，如 ultraedit 在修改后会生成文件名加 bak 后缀的文件）。攻击者通过直接访问这些备份的路径可以下载文件
用例级别	1
测试条件	1、 拥有运行Web服务器的操作系统帐号和口令 2、 Web业务运行正常
执行步骤	1、 登陆后台Web服务器的操作系统 2、 以cd命令进入可以通过Web方式访问的目录（比如tomcat服务器的\$home/webapps目录，jboss服务器的\$home/jboss/server/default/deploy目录） 3、 用find命令，查找是否存在以下备份文件，如果存在则测试不通过。 <hr/> "*.bak" "*.BAK" "*.old" "*.OLD" "*.zip" "*.ZIP" "*.rar" "*.tar" "*.temp" "*.save" "*.backup"
预期结果	可以通过 Web 方式访问的目录，不存在开发过程（包括现场定制）中的产生的临时文件、备份文件等。
备注	
测试结果	

4.6 认证测试

4.6.1 验证码测试

编号	Web _01
测试用例名称	验证码测试
测试目的	查看是否有验证码机制，以及验证码机制是否完善
用例级别	一级
测试条件	1、 已知Web网站地址 2、 Web业务运行正常 3、 存在登陆页面
执行步骤	1、 登陆页面是否存在验证码，不存在说明存在漏洞，完成测试 2、 验证码和用户名、密码是否一次性、同时提交给服务器验证，如果是分开提交、分开验证，则存在漏洞 3、 在服务器端，是否只有在验证码检验通过后才进行用户名和密码的检验，如果不是说明存在漏洞。（检测方法：输入错误的用户名或密码、错误的验证码。观察返回信息，是否只提示验证码错误，也就是说当验证码错误时，禁止再判断用户名和密码。） 4、 验证码是否为图片形式且在一张图片中，不为图片形式或不在一张图片中，说明存在漏洞，完成测试 5、 生成的验证码是否可以通过html源代码查看到，如果可以说明存在漏洞，完成测试 6、 生成验证码的模块是否根据提供的参数生成验证码，如果是说明存在漏洞，完成测试 7、 请求10次观察验证码是否随机生成，如果存在一定的规律（例如5次后出现同一验证码）说明存在漏洞，完成测试 8、 观察验证码图片中背景是否存在无规律的点或线条，如果背景为纯色（例如只有白色）说明存在漏洞，完成测试 9、 验证码在认证一次后是否立即失效：

预期结果	不存在上述漏洞
备注	本用例根据最严格的方式对目标进行测试，如果产品线对安全的要求不高且有自身的安全策略规定时，可以视情况对测试项进行部分测试
测试结果	

4.6.2 认证错误提示

编号	Web_02
测试用例名称	认证错误提示
测试目的	为了进行暴力破解，攻击者需要知道已存在的用户名，再对该用户名进行攻击。所以，本测试用于确认目标服务器在处理登陆操作时会提示出具体的信息。
用例级别	一级
测试条件	1、 已知Web网站地址 2、 Web业务运行正常 3、 存在登陆页面
执行步骤	1、 在用户名（或其他身份标志）的输入框中输入test，口令任意 2、 若服务器返回提示类似于“用户名不存在”，则说明存在漏洞，完成测试 3、 使用正确的用户名（或同功能的身份标志），在口令框中输入test 4、 若服务器提示类似于“密码/口令错误” “用户名不存在”之类的信息，则说明存在漏洞，完成测试。
预期结果	服务器不会针对认证错误的情况提示准确的信息。
备注	
测试结果	

4.6.3 锁定策略测试

编号	Web_03
测试用例名称	锁定策略测试
测试目的	在缺少锁定策略和验证码设计有问题的情况下，攻击者可以通过枚举的方式来进行暴

	力猜解。本测试用于发现目标系统是否缺少锁定策略。
用例级别	一级
测试条件	1、 已知Web网站地址 2、 Web业务运行正常 3、 存在登陆页面
执行步骤	1、 打开登陆页面 2、 在用户名（或同功能的身份标志）输入框中输入正确的用户名 3、 在口令（或同功能的口令标志）输入框中输入错误的口令 4、 在验证码输入框（如果有的话）中输入正确的验证码 5、 提交表单 6、 重复1~5步骤10次 7、 判断目标系统返回的信息
预期结果	目标系统提示“帐号已锁定”或者“IP 已锁定”或者类似“锁定”等之类的信息。
备注	第 6 步中重复步骤次数视各产品实际情况而定。 另外，如果系统存在一些认证接口（带认证参数的 URL，不是普通登陆页面），那么也需要对认证接口进行失败认证尝试，以测试其锁定策略。
测试结果	

4.6.4 认证绕过测试

编号	Web _04
测试用例名称	认证绕过测试
测试目的	发现目标认证系统是否存在绕过的可能
用例级别	一级
测试条件	1、 已知Web网站地址 2、 Web业务运行正常 3、 存在登陆页面
执行步骤	1、 打开登陆页面 2、 在用户名（或同功能的身份标志）输入框中输入admin' or '1' =' 1

	3、 在口令（或同功能的口令标志）输入框中输入admin' or '1' =' 1 4、 提交表单，观察返回结果
预期结果	不能够成功登陆。
备注	如果目标系统采用 javascript 限制了输入格式，可以通过 WebScarab 来进行修改。 关于 WebScarab 的使用说明，请参看相关帮助文档。
测试结果	

4.6.5 修复密码测试

编号	Web_ 05
测试用例名称	修改密码测试
测试目的	如果修改密码功能存在着缺陷，攻击者可以通过此其缺陷修改其他用户的密码。
用例级别	一级
测试条件	1、 已知Web网站地址 2、 Web业务正常运行 3、 网站提供修改密码的功能 4、 已知某正确的用户账号
执行步骤	1、 登陆网站 2、 进入密码修改页面 3、 查看是否必须提交正确旧密码，如果不需要则存在漏洞 4、 填写并提交修改密码数据，并以WebScarab拦截修改密码请求，观察新旧密码是否通过同一个HTTP请求提交到服务器，如果不是则存在漏洞； 5、 观察是否客户端是否提交用户名或用户ID，如果是则修改为其他用户名或用户ID，看看是否能够成功修改其他用户的密码，如果可以则存在漏洞。
预期结果	用户修改密码时必须提供旧密码，普通用户无法修改其他用户的密码。
备注	如果初始口令为系统提供的默认口令、或者是由管理员设定时，则在用户/操作员使用初始口令成功登录后，系统必须强制用户/操作员更改初始口令，直至更改成功，否则是漏洞。
测试结果	

4.6.6 不安全的数据传输

编号	Web_06-01
测试用例名称	登陆过程信息机密性保护
测试目的	测试 Web 程序在处理登录过程中用户名和口令的传输是否采用了加密传输的机制。
用例级别	一级
测试条件	1、 已知Web网站地址 2、 Web业务正常运行 3、 Web业务存在登陆认证模块
执行步骤	1、 已知网站登录地址为：http://www.example.com/login.xxx 2、 开启WebScarab，配置对GET和POST请求进行拦截；在浏览器中配置代理服务器IP为127.0.0.1，端口为8008 3、 在登录处输入用户名和口令、验证码登陆 4、 查看WebScarab拦截的请求中，用户名和口令是否采用HTTPS协议传输。
预期结果	用户名和密码信息采用 HTTPS 传输。
备注	
测试结果	

4.6.7 强口令策略测试

编号	Web_07
测试用例名称	强口令策略测试
测试目的	本测试为检查目标系统是否存在强口令策略。
用例级别	二级
测试条件	1、 已知Web网站地址 2、 Web业务运行正常 3、 Web业务存在帐号管理 4、 已知正常用户的用户、口令 5、 存在口令修改页面

执行步骤	<p>1、 使用正确的用户、口令登陆Web业务系统</p> <p>2、 打开口令修改页面</p> <p>3、 在新口令输入框中输入字母加数字的5位字符（如ab123）作为密码并提交，如果未提示“口令长度过短”等诸如此类的信息，说明存在弱点，完成测试。</p> <p>4、 在新口令输入框中输入6位数字（如123456）作为密码并提交，如果未提示“口令字符需要大小写”等诸如此类的信息，说明存在弱点，完成测试。</p> <p>5、 观察结果</p>
预期结果	目标系统存在满足上述步骤的较严格的口令复杂度策略。
备注	<p>上面只是举例说明口令复杂度的测试，实际上强口令策略还包括口令有效期、历史口令等，这些都要测试。</p> <p>对于一些 Web 应用密码只能是数字组成，则不强制要求强口令。</p> <p>Web 应用安全开发规范中的强口令策略：</p> <p>1. 口令长度的取值范围为：0-32 个字符；口令的最短长度和最长长度可配置；口令的最短长度建议默认为 6 个字符。</p> <p>2. 口令中至少需要包括一个大写字母（A-Z）、一个小写字母（a-z）、一个数字字符（0-9）；口令是否包含特殊字符要求可以配置。</p> <p>3. 口令中允许同一字符连续出现的最大次数可配置，取值范围：0-9，当取值为 0 时，表示无限制，建议默认为 3。</p> <p>4. 口令须设置有效期，最短有效期的取值范围：0-9999 分钟，当取值为 0 时，表示不做限制，建议默认：5 分钟；最长有效期的取值范围：0-999 天，当取值为 0 时，表示口令永久有效，建议默认：90 天。</p> <p>5. 在口令到期前，当用户登录时系统须进行提示，提前提示的天数可配置，取值范围：1-99 天，建议默认：7 天。</p> <p>6. 口令到达最长有效期后，用户再次登录成功但在进入系统前，系统强制更改口令，直至更改成功。</p> <p>7：口令历史记录数可配置，取值范围为：0-30；建议默认：3 个。</p> <p>8：管理员/操作员/最终用户修改自己的口令时，必须提供旧口令。</p> <p>9：初始口令为系统提供的默认口令、或者是由管理员设定时，则在用户/操作员使用初始口令成功登录后，要强制用户/操作员更改初始口令，直至更改成功。</p>

	<p>10: 口令不能以明文的形式在界面上显示。</p> <p>11: 口令不能以明文的形式保存，须加密保存；口令与用户名关联加密，即加密前的数据不仅包括口令，还包括用户名。</p> <p>12: 只有当用户通过认证之后才可以修改口令。</p> <p>13: 修改口令的帐号只能从服务器端的会话信息中获取，而不能由客户端指定。</p> <p>建议 3.2.1.1: 实现弱口令词典功能。</p>
测试结果	

4.7 会话管理测试

4.7.1 身份信息维护方式测试

编号	Web _01
测试用例名称	身份信息维护方式测试
测试目的	发现目标系统是否采用参数来进行身份判断。
用例级别	一级
测试条件	<p>1、 已知Web网站地址</p> <p>2、 Web业务正常运行</p> <p>3、 Web业务存在不同级别的权限（角色）</p>
执行步骤	<p>1、 登录系统</p> <p>2、 配置WebScarab进行http get和post请求拦截</p> <p>3、 请求Web页面</p> <p>4、 在WebScarab中查看请求报文</p> <p>5、 检查请求消息中是否携带了与用户身份相关的数据。如果有，修改身份信息。如果服务器端以修改后的身份进行操作，则说明存在漏洞，完成测试。</p>
预期结果	用户登陆后，身份信息不再由客户端提交，而是以服务器端会话信息中保存的身份信息为准。
备注	
测试结果	

4.7.2 Cookie 存储方式测试

编号	Web_02
测试用例名称	Cookie 存储方式测试
测试目的	某些 Web 应用将 SessionId 放到了 URL 中进行传输，攻击者能够诱使被攻击者访问特定的资源，例如图片。在被攻击者查看资源时获取该 SessionID（在 HTTP 协议中 Referer 标题头中携带了来源地址），从而导致身份盗用。
用例级别	一级
测试条件	1、 已知Web网站地址 2、 Web业务运行正常 3、 Web业务存在登陆认证模块 4、 已知正确的用户名、口令
执行步骤	1、 登录系统。 2、 请求不同的业务应用 3、 观察URL。
预期结果	URL 中没有携带 Session ID 信息（可能是 sid, JSESSIONID 等形式）。
备注	
测试结果	

4.7.3 用户注销登陆的方式测试

编号	Web_03
测试用例名称	用户注销登陆的方式测试
测试目的	查看是否提供注销登陆功能
用例级别	一级
测试条件	1、 已知Web网站地址 2、 Web业务运行正常 3、 Web业务存在登陆认证模块 4、 已知正确的用户名、口令

执行步骤	1、 使用正常的用户、口令登录系统。 2、 查找登陆后的所有页面中是否存在明确的“退出”或“注销”（或类似）的按钮或者链接
预期结果	登陆后的页面中有明确的“退出”或“注销”按钮。
备注	
测试结果	

4.7.4 注销时会话信息是否清除测试

编号	Web_04
测试用例名称	注销时（logout），会话信息是否清除
测试目的	由于网站程序在编写上考虑不周，用户注销后会话信息没有清除，导致用户在点击注销按钮之后还能继续访问注销之前（也就是登陆之后）才能访问的页面。我们需要经过测试来判断是否存在此类问题。
用例级别	一级
测试条件	1、 已知Web网站地址 2、 Web业务正常运行 3、 存在注销功能的页面
执行步骤	1、 使用合法的账户口令登陆。 2、 开启WebScarab，配置对GET和POST请求进行拦截 3、 在浏览器中配置代理服务器IP为127.0.0.1，端口为8008 4、 在Web页面中进行一些操作（比如修改个人信息），这些操作都会被WebScarab拦截，不修改，在弹出的WebScarab界面中点击“Accept Changes”按钮。这样请求就被WebScarab记录下来。 5、 然后在Web页面中点击注销/退出（logout）。 6、 点击WebScarab的“Manual Request”TAB页，在Previous Requests的下拉列表框中选择“步骤4”所产生的URL请求，然后点击“Fetch Response”，重新发送“步骤4”的URL请求。 7、 在WebScarab的Response的“Raw”Tab页中观察返回结果，如果还能够正常完成

	“步骤4”的操作，则存在安全漏洞。
预期结果	在 WebScarab 的 Response 的“Raw”Tab 页中显示“HTTP/1.1 302 Moved Temporarily”，不能够访问只有登陆才能访问的页面，不能完成只有登陆后才能完成的操作。
备注	如果存在多个注销功能的页面，要重复测试过程把所有有注销功能的页面测试完。
测试结果	

4.7.5 会话超时时间测试

编号	Web_05
测试用例名称	会话超时时间测试
测试目的	查看是否存在浏览器窗口闲置超时后需重新登录的机制
用例级别	一级
测试条件	1、 已知Web网站地址 2、 Web业务运行正常 3、 Web业务存在登陆认证模块 4、 已知正确的用户名、口令
执行步骤	1、 使用正常的用户、口令登录系统。 2、 将浏览器窗口闲置11分钟。 3、 刷新浏览器，查看是否需要重新登录。 备注：也可以登陆后台 Web 服务器，查看对应的 szboardna-config.xml 文件中的 cook.expire.time 参数值，该值表示会话的超时时间。
预期结果	会话超时时间不大于 10 分钟，刷新浏览器之后需要重新登录。
备注	
测试结果	

4.7.6 会话定置测试

编号	Web _06
测试用例名称	会话定置测试

测试目的	查看登录成功后会话标识是否变更。如果未变更，那么攻击者就可以通过一些手段（如构造 URL）为受害者确定一个会话标识，当受害者登录成功后，攻击者也可以利用这个会话标识冒充受害者访问系统。
用例级别	三级
测试条件	1、 已知Web网站地址 2、 Web业务运行正常 3、 Web业务存在登陆认证模块 4、 已知正确的用户名、口令
执行步骤	1、 访问登录页面 2、 开启WebScarab，配置对GET和POST请求进行拦截；并在浏览器中配置代理服务器IP为127.0.0.1，端口为8008 3、 填入正确的用户名和口令，填入正确的验证码，登录 4、 WebScarab跳出拦截界面，点击“RAW”TAB页，查看会话标识（例如JSP的会话标识为JSESSIONID）的值。 5、 点击“Accept Changes”，登录成功。 6、 成功登录后，点击系统提供的链接，请求任意功能页面。 7、 WebScarab再次跳出拦截界面，点击“RAW”TAB页，查看（例如JSP的会话标识为JSESSIONID）的值。 8、 如果步骤4和步骤7查看到的会话标识的值一样，说明登录前后会话标识没有变更，存在会话定置的安全漏洞。
预期结果	步骤 4 和步骤 7（登录前后）查看到的会话标识的值不一样
备注	虽然会话定置的危害比较大，但由于要事先为受害者确定会话标识，并且让受害者根据此会话标识登录系统，其发生的概率很小，所以综合风险不高，测试时根据被测系统安全要求的高低，来确定是否执行该用例。
测试结果	

4.8 文件上传下载测试

4.8.1 文件上传测试

编号	Web _01
测试用例名称	文件上传测试
测试目的	很多网站提供文件上传功能（包括图片上传），如果在服务器端没有对上传文件的类型、大小以及保存的路径及文件名进行严格限制，攻击者就很容易上传后门程序取得WebShell，从而控制服务器。
用例级别	一级
测试条件	1、 Web业务运行正常 2、 待测网站存在文件上传页面
执行步骤	1、 登陆网站，并打开文件上传页面 2、 点击“浏览”按钮，并选择本地的一个JSP文件（比如hacker.jsp），确认上传。 3、 如果客户端脚本限制了上传文件的类型（比如允许gif文件），则把hacker.jsp更名为hacker.gif；配置HTTP Proxy（WebScarab）进行http请求拦截；重新点击“浏览”按钮，并选择hacker.gif，确认上传。 4、 在WebScarab拦截的HTTP请求数据中，将hacker.gif修改为hacker.jsp，再发送请求数据。 5、 登陆后台服务器，用命令find / -name hacker.jsp查看hacker.jsp文件存放的路径。如果可以直接以Web方式访问，则构造访问的URL，并通过浏览器访问hacker.jsp，如果可以正常访问，则已经取得WebShell，测试结束。如果hacker.jsp无法通过web方式访问，例如hacker.jsp存放在/home/tmp/目录下，而/home/tomcat/webapps目录对应http://www.example.com/，则进行下一步 6、 重复1～3，在WebScarab拦截的HTTP请求数据中，将hacker.gif修改为../tomcat/webapps/hacker.jsp，再发送请求数据。 7、 在浏览器地址栏输入http://www.example.com/hacker.jsp，访问该后门程序，取得WebShell，结束测试。
预期结果	服务器端对上传文件的类型、大小以及保存的路径及文件名进行严格限制，无法上传

	后门程序。
备注	
测试结果	

4.8.2 文件下载测试

编号	Web _02
测试用例名称	文件下载测试 1
测试目的	很多网站提供文件下载功能，如果网站对下载文件的权限控制不严，攻击者很容易利用目录跨越、越权下载到本不该下载的文件（比如其他用户的私有、敏感文件）。
用例级别	一级
测试条件	1、 Web业务运行正常 2、 已知某下载页面URL（假设用户a下载自己的指导书文件 <code>http://www.example.com/download/usera/test.xls</code> ）
执行步骤	1、 猜测并更改URL路径 http://www.example.com/download/userb/test.xls http://www.example.com/download/userc/test.xls <code>http://www.example.com/admin/report.xls</code> 2、 观察页面返回信息，如果可以越权获取到其他用户的私有、敏感文件，则说明存在漏洞。
预期结果	不能越权获取到不该获取的文件
备注	
参考资料	

4.9 信息泄漏测试

4.9.1 连接数据库的账号密码加密测试

编号	Web _01
测试用例名称	连接数据库的帐号密码加密测试
测试目的	连接数据库的帐号密码在配置文件中如果明文存储，容易被恶意维护人员获取，从而直接登陆后台数据库进行数据篡改。
用例级别	一级
测试条件	1、拥有 Web 服务器操作系统的帐号和口令 2、已知连接数据库的帐号密码所在的配置文件，可以找环境搭建人员咨询。（还可以用 grep 命令查找哪些 xml 文件包含该数据库帐号，然后打开这些文件进行检查）
执行步骤	1、 登陆Web服务器的操作系统。 2、 用find命令找到对应的配置文件。 3、 查看配置文件中连接数据库的帐号密码在配置文件中是否加密。
预期结果	连接数据库的帐号密码在配置文件中加密存储
备注	
测试结果	

4.9.2 客户端源代码敏感信息测试

编号	Web _02
测试用例名称	客户端源代码敏感信息测试
测试目的	Web 页面的 html 源代码中不允许包含口令等敏感信息，特别关注修改口令、带有星号口令的 Web 页面。
用例级别	一级
测试条件	1、已知 Web 网站地址 2、Web 业务正常运行 3、待测页面能够正常访问

执行步骤	1、 登陆Web服务器。 2、 选择可能存在敏感信息的页面（比如修改口令、带有星号口令的页面）。 3、 在页面上单击鼠标右键，选择“查看源代码”。 4、 查看页面的源代码包含口令等敏感信息。
预期结果	在页面的源代码中看不到明文的口令等敏感信息
备注	
测试结果	

4.9.3 客户端源代码注释测试

编号	Web _03
测试用例名称	客户端源代码注释测试
测试目的	开发版本的 Web 程序所带有的注释在发布版本中没有被去掉，而导致一些敏感信息的泄漏。我们要查看客户端能看到的页面源代码并发现此类安全隐患。
用例级别	一级
测试条件	1、 已知 Web 网站地址 2、 Web 业务正常运行 3、 待测页面能够正常访问
执行步骤	1、 从客户端查看网页源代码。 2、 找到注释部分并查看是否有敏感信息。
预期结果	源代码里不存在敏感信息（比如：内网 IP 地址、SQL 语句、密码、物理路径等）。
备注	
测试结果	

4.9.4 异常处理

在这个部分我们通过构造各种异常的条件让 Web 程序处理，通过其返回信息来判断是否存在信息泄漏的问题。

◆ 不存在的URL

	<p>添加特殊字符后的 URL 为：</p> <p><code>http://www.example.com/page.xxx?name= value%</code></p> <p><code>http://www.example.com/page.xxx?name= value*</code></p> <p><code>http://www.example.com/page.xxx?name= value;</code></p> <p><code>http://www.example.com/page.xxx?name= value?</code></p> <p><code>http://www.example.com/page.xxx?name= value'</code></p> <p>2、 在浏览器地址栏中依次输入这些带有特殊字符的URL并回车</p> <p>3、 观察返回结果</p>
预期结果	返回的页面中没有敏感信息（比如：详细的类名、方法，SQL 语句，物理路径等）。
备注	<p>页面可能接受多个参数，需对每个参数都进行测试</p> <p>POST 情况下的参数也需要测试</p>
测试结果	

◆ 逻辑错误

Web 应用在处理一些具有逻辑错误的请求时，可能会返回错误的信息，通过返回的错误信息来确认是否会有敏感信息的泄漏问题。本测试无法给出很具体的测试指导，只能给出相应的描述说明，那就是：尽可能的尝试使用违背业务逻辑处理的参数来让 Web 应用返回异常信息。

4.9.5 不安全的存储

本测试无法给出很具体的测试指导，在测试中可以根据下列方法进行测试：

- ◆ 上传文件所在的目录（或者是临时目录）能否直接远程访问。
- ◆ 服务器配置文件目录或日志存放目录能否直接访问。
- ◆ 日志或数据库中是否能够看到明文敏感信息：Web应用一般都会对一些业务操作进行记录日志操作。在日志文件中很可能包含了一些敏感信息（例如用户在更改口令后日志文件中会记录该用户原始口令以及更改后的口令信息）。假如Web应用的日志存放的目录被攻击者发现（符合上一条测试项），那么攻击者就可以从中获取大量的敏感数据用户后续的攻击。因此，本测试需要对日志文件中的数据进行调查，检查其中是否带有敏感数据或者敏感数据进行了加密处理。同样对于数据库也述要进行审查。

4.10 逻辑测试

本测试无法给出很具体的测试指导，在测试中可以根据下列方法进行测试：

首先，测试人员应尽量理解业务系统。

其次，提炼各种业务场景和工作流程

然后，设计业务逻辑测试，设计时可参考以下原则：

- 1、对用户行为顺序处理不当，例如：某个网上申报流程，首先验证用户名和口令，正确后跳到下一步流程输入页面，然后再到下一步提交申报。一般来说如果用户名和口令验证不通过，就不能进行下一步，但如果测试人员记住了流程输入页面对应的链接，直接在浏览器中输入链接，就能跳转到对应的页面。
- 2、如果不同的业务处理流程都使用了类似的逻辑处理，则这些业务流程都需要进行相同的逻辑测试。例如：对于业务协同系统存在两种金额处理的流程，假设为支出与收入。两个流程中间都有金额参数传递，且后台进行了用户金额扣除。那么在测试中都需要考虑到金额的逻辑测试（假如将金额更改为负数）情况。

4.11 其他

4.11.1 Class 文件反编译测试

编号	Web _01
测试用例名称	class 文件反编译测试
测试目的	java 源代码经过一些低版本的编译器编译后形成的 class 类文件能够被进行反编译，完全还原成 java 源代码。在攻击者有能力获取 class 时文件时很有可能造成信息泄漏。
用例级别	三级
测试条件	1、 测试用机上安装了 jad.exe 工具，假设安装在从 c:\jad.exe
执行步骤	1、 找出一些重要的 class 文件（比如：加解密或登陆认证相关的）

	2、 进行windows命令行控制台 3、 输入命令cd /d c:\classes 4、 c:\jad.exe *.class 5、 观察在目录中输出的文件
预期结果	不能得到源文件。
备注	
参考资料	

5 本规范所涉及的测试工具

安全工具的申请和使用请遵循公司信息安全相关规定。

工具名称	简介
AppScan	IBM Rational AppScan，在 Web 安全测试中所使用的自动化扫描工具
WebScarab	Web Proxy 软件，可以对浏览器与 Web 服务器之间的通信数据进行编辑修改
DirBuster	在 Web 安全测试中用来遍历目录、文件的工具
WSDigger	Web service 安全测试工具
Jad	Java class 文件反编译软件
CAJAVA	Java class 文件反编译软件（兼容多个 JDK 版本）
Pangolin	SQL 注入测试工具
WireShark	网络协议抓包与分析工具