DKBA

华为技术有限公司内部技术规范

DKBA 2355-2009.7

Web应用安全测试规范V1.4



2009年7月5日发布 2009年7月5日实施

华为技术有限公司

Huawei Technologies Co., Ltd.

版权所有 侵权必究

All rights reserved

**修订声明Revision declaration**

本规范拟制与解释部门：

安全解决方案部电信网络与业务安全实验室、软件公司安全TMG、软件公司测试业务管理部

本规范的相关系列规范或文件：

《Web应用安全开发规范》

相关国际规范或文件一致性：

《OWASP Testing Guide v3》

《信息安全技术信息安全风险评估指南》

《Information technology Security techniques Management of information and communications technology security》－ISO 13335

替代或作废的其它规范或文件：

无

相关规范或文件的相互关系：

本规范以《Web应用安全开发规范》为基础、结合Web应用的特点而制定。

| 版本号 | 主要起草部门专家 | 主要评审部门专家 | 修订情况 |
| --- | --- | --- | --- |
| V1.1 | 安全解决方案：赵武、吕晓雨56987、王欢00104062  业务与软件测试部系统部：孟咏喜00137435 | 安全解决方案：刘海军、吴宇翔、吴海翔、杨光磊、宋柳松、梁业金、姜凡  业务与软件：何伟祥、刘高峰、龚连阳、许汝波、王娟娟、龚小华、王向辉、李磊、杨晓峰  网络解决方案验证部：张喆  核心网：车广芳、苏三、刘京、冻良 |  |
| V1.2 | 何伟祥33428 | 刘高峰、吴宇翔、杨光磊、王欢、胡坤红、张伟、孟咏喜、成庆华、黎迎斌、周鹏、张喆、文桂林、张理、姜永章、苏燕鲁 | 增加Web Service、上传、下载、控制台等方面的测试规范，更正V1.1一些描述不准确的测试项 |
| V1.3 | 何伟祥00162822 | 刘高峰、胡坤红、张伟 | 增加“会话固定”、“审计日志”、“DWR”的安全测试规范，完善验证码安全测试的方法。 |
| V1.4 | 刘振南 00264924 | 胡坤红、张伟、宋飞、夏成林、王欢、沈海涛、钱育波、石功 | 新增以下内容：  3.3.5 SEC\_Web\_ DIR\_05\_02 版本控制软件SVN备份文件测试  3.5.8 会话标识随机性测试  3.6.3 跨站伪造请求测试  3.9.2 LDAP注入测试  3.9.5 回车换行符（CRLF）注入测试  3.9.6 XML注入测试  3.13 SEC\_Web\_SERVICE\_02 Web Service测试  3.15 HTML5安全测试  3.16 FLASH安全配置测试  3.17.3 WEB部署与管理测试  3.17.4 Struts2框架测试  5.2 HTML5新增标签 |

**目 录 Table of Contents**

[1 概述 9](#_Toc402773657)

[1.1 背景简介 9](#_Toc402773658)

[1.2 适用读者 9](#_Toc402773659)

[1.3 适用范围 9](#_Toc402773660)

[1.4 安全测试在IPD流程中所处的位置 10](#_Toc402773661)

[1.5 安全测试与安全风险评估的关系说明 10](#_Toc402773662)

[1.6 注意事项 11](#_Toc402773663)

[1.7 测试用例级别说明 11](#_Toc402773664)

[2 测试过程示意图 12](#_Toc402773665)

[3 Web安全测试规范 13](#_Toc402773666)

[3.1 自动化Web漏洞扫描工具测试 13](#_Toc402773667)

[3.1.1 AppScan application扫描测试 14](#_Toc402773668)

[3.1.2 AppScan Web Service 扫描测试 15](#_Toc402773669)

[3.2 服务器信息收集 15](#_Toc402773670)

[3.2.1 运行帐号权限测试 15](#_Toc402773671)

[3.2.2 Web服务器端口扫描 16](#_Toc402773672)

[3.2.3 HTTP方法测试 16](#_Toc402773673)

[3.2.4 HTTP PUT方法测试 17](#_Toc402773674)

[3.2.5 HTTP DELETE方法测试 18](#_Toc402773675)

[3.2.6 HTTP TRACE方法测试 19](#_Toc402773676)

[3.2.7 HTTP MOVE方法测试 20](#_Toc402773677)

[3.2.8 HTTP COPY方法测试 20](#_Toc402773678)

[3.2.9 Web服务器版本信息收集 21](#_Toc402773679)

[3.3 文件、目录测试 22](#_Toc402773680)

[3.3.1 工具方式的敏感接口遍历 22](#_Toc402773681)

[3.3.2 Robots方式的敏感接口查找 24](#_Toc402773682)

[3.3.3 Web服务器的控制台 25](#_Toc402773683)

[3.3.4 目录列表测试 27](#_Toc402773684)

[3.3.5 文件归档测试 29](#_Toc402773685)

[3.4 认证测试 30](#_Toc402773686)

[3.4.1 验证码测试 30](#_Toc402773687)

[3.4.2 认证错误提示 31](#_Toc402773688)

[3.4.3 锁定策略测试 32](#_Toc402773689)

[3.4.4 认证绕过测试 33](#_Toc402773690)

[3.4.5 找回密码测试 33](#_Toc402773691)

[3.4.6 修改密码测试 34](#_Toc402773692)

[3.4.7 不安全的数据传输 35](#_Toc402773693)

[3.4.8 强口令策略测试 36](#_Toc402773694)

[3.5 会话管理测试 37](#_Toc402773695)

[3.5.1 身份信息维护方式测试 37](#_Toc402773696)

[3.5.2 Cookie存储方式测试 38](#_Toc402773697)

[3.5.3 用户注销登陆的方式测试 38](#_Toc402773698)

[3.5.4 注销时会话信息是否清除测试 39](#_Toc402773699)

[3.5.5 会话超时时间测试 40](#_Toc402773700)

[3.5.6 会话定置测试 40](#_Toc402773701)

[3.5.7 会话标识携带 41](#_Toc402773702)

[3.5.8 会话标识随机性测试 42](#_Toc402773703)

[3.6 权限管理测试 46](#_Toc402773704)

[3.6.1 横向测试 47](#_Toc402773705)

[3.6.2 纵向测试 49](#_Toc402773706)

[3.6.3 跨站伪造请求测试 54](#_Toc402773707)

[3.7 文件上传下载测试 56](#_Toc402773708)

[3.7.1 文件上传测试 56](#_Toc402773709)

[3.7.2 文件下载测试 57](#_Toc402773710)

[3.8 信息泄漏测试 59](#_Toc402773711)

[3.8.1 连接数据库的帐号密码加密测试 59](#_Toc402773712)

[3.8.2 客户端源代码敏感信息测试 59](#_Toc402773713)

[3.8.3 客户端源代码注释测试 60](#_Toc402773714)

[3.8.4 异常处理 60](#_Toc402773715)

[3.8.5 HappyAxis.jsp页面测试 62](#_Toc402773716)

[3.8.6 Web服务器状态信息测试 63](#_Toc402773717)

[3.8.7 不安全的存储 63](#_Toc402773718)

[3.9 输入数据测试 64](#_Toc402773719)

[3.9.1 SQL注入测试 64](#_Toc402773720)

[3.9.2 LDAP注入测试 66](#_Toc402773721)

[3.9.3 MML语法注入 66](#_Toc402773722)

[3.9.4 命令执行测试 67](#_Toc402773723)

[3.9.5 回车换行符（CRLF）注入测试 68](#_Toc402773724)

[3.9.6 XML注入测试 70](#_Toc402773725)

[3.10 跨站脚本测试 73](#_Toc402773726)

[3.10.1 GET方式跨站脚本测试 73](#_Toc402773727)

[3.10.2 POST方式跨站脚本测试 74](#_Toc402773728)

[3.10.3 URL跨站脚本测试 75](#_Toc402773729)

[3.11 逻辑测试 76](#_Toc402773730)

[3.12 搜索引擎信息收集 76](#_Toc402773731)

[3.13 Web Service测试 77](#_Toc402773732)

[3.14 DWR（Direct Web Remoting）测试 81](#_Toc402773733)

[3.15 HTML5安全测试 83](#_Toc402773734)

[3.15.1 跨域资源共享测试 83](#_Toc402773735)

[3.15.2 Web客户端存储安全测试 86](#_Toc402773736)

[3.15.3 WebWorker安全测试 89](#_Toc402773737)

[3.16 FLASH安全配置测试 90](#_Toc402773738)

[3.17 其他 91](#_Toc402773739)

[3.17.1 日志审计 91](#_Toc402773740)

[3.17.2 class文件反编译测试 92](#_Toc402773741)

[3.17.3 WEB部署与管理测试 92](#_Toc402773742)

[3.17.4 Struts2框架测试 93](#_Toc402773743)

[4 AppScan测试覆盖项说明 95](#_Toc402773744)

[5 附件 96](#_Toc402773745)

[5.1 本规范所涉及的测试工具 96](#_Toc402773746)

[5.2 HTML5新增标签 96](#_Toc402773747)

[5.2.1 结构标签 96](#_Toc402773748)

[5.2.2 多媒体标签 97](#_Toc402773749)

[5.2.3 Web应用标签 97](#_Toc402773750)

[5.2.4 其他标签 97](#_Toc402773751)

Web安全测试规范

缩略语清单

|  |  |
| --- | --- |
| **缩略语** | **全称** |
| CRLF | \r\n回车换行 |
| LDAP | Lightweight Directory Access Protocol 轻量级目录访问协议 |
| MML | man-machine language 人机交互语言 |
| SessionID | 标志会话的ID |
| Web Service | Web服务是一种面向服务的架构的技术，通过标准的Web协议提供服务，目的是保证不同平台的应用服务可以互操作。 |
| SOAP | Simple Object Access Protocol 简单对象访问协议 |
| XSS | Cross Site Scripting 跨站脚本 |
| CSRF | Cross Site Request Forgery 跨站请求伪造 |

# 概述

## 背景简介

在Internet大众化、Web技术飞速演变、黑客工具日益普及的今天，针对Web的攻击和破坏不断增多，在线安全面临日益严峻的挑战，安全风险达到了前所未有的高度。

为了规避Web安全风险、规范Web安全开发，公司已经制定并发布了《Web 应用安全开发规范》；但如何系统的进行Web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《Web 安全测试规范》，本规范可让测试人员针对Web常见安全漏洞或攻击方式，系统的对被测Web系统进行快速安全性测试。

## 适用读者

本规范的读者及使用对象主要为Web相关的测试人员、开发人员、专职的安全测试评估人员等。

## 适用范围

本规范主要针对基于通用服务器的Web应用系统，其他Web系统也可以参考，如基于嵌入式系统的Web维护接口等。

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：

**Web应用**

**应用服务器**

* Uniportal
* JBoss
* ……

**客户端**

**Web服务器**

* IIS
* Apache
* ……



**数据库服务器**

* Oracle
* DB2
* ……



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，涵盖Web安全开发规范的内容。

## 安全测试在IPD流程中所处的位置

一般建议在TR4前根据产品实现架构及安全需求，完成Web安全性测试需求分析和测试设计，准备好Web安全测试用例。

在TR4版本正式转测试后，即可进行Web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Web安全测试执行，但最迟不得超过TR5。

## 安全测试与安全风险评估的关系说明

安全风险是指威胁利用脆弱性对目标系统造成安全影响的可能性及严重程度。其中威胁（Threat）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。脆弱性(Vulnerability)也称弱点，是应用系统本身存在的，包括系统实现中的缺陷、配置中的弱点等。外部威胁利用系统的脆弱性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脆弱性识别部分，特别是技术性的脆弱性识别。

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》。

## 注意事项

* Web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），建议只在测试环境中进行；如果一定要在现网运行环境中执行，那么务必配置专门的测试数据，测试执行是应该非常慎重，只能修改或删除这些测试数据，禁止修改、删除现网其他数据。
* 本规范最主要目的是为了发现安全弱点，至于发现一个弱点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申请和使用，请遵循公司信息安全相关规定。
* 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。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。

## 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** |
| **1** | 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。 |
| **2** | 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。 |
| **3** | 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。 |
| **4** | 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 测试级别说明表

# 测试过程示意图

本测试主要包括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。

**造成的影响**

**主动**

**模式**

**被动模式**

初始化

完成

Web结构获取

权限测试

归档测试

参数分析

异常处理

注入测试

命令执行

文件包含

跨站脚本

信息窃取

备份文件

后台查找

目录列表

会话管理

信息泄漏

数据破坏

拒绝服务

信息获取

熟悉业务逻辑

Google Hacking

接口测试

认证测试

暴力破解

认证绕过

逻辑处理

越权操作

身份仿冒

日志检查

拒绝服务

上传下载

# Web安全测试规范

## 自动化Web漏洞扫描工具测试

自动化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application和Web service两部分。Application指通常意义上的Web应用，而Web service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。

### AppScan application扫描测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_TOOL\_01 |
| 测试用例名称 | AppScan application 扫描测试 |
| 测试目的 | 利用自动化的Web安全扫描工具AppScan进行扫描，以发现Web应用中存在的常见漏洞 |
| 用例级别 | 1 |
| 测试条件 | 1. 已知Web服务器域名或IP地址 2. Web业务运行正常 3. 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No Login，点击next 5. 不需修改任何参数，点击next 6. 不需修改参数，选择Start a full automatic scan，点击finish完成配置，开始扫描 7. 扫描完成，保存扫描结果，并对结果进行分析 |
| 预期结果 | 经过对扫描结果分析，确认不存在“中等等级”及以上级别的漏洞。 |
| 备注 | 注意：该用例的执行对被测系统的性能影响比较大，而且可能导致一些垃圾数据，建议只在测试环境执行。  由于自动化工具在很多情况下只是提示一种漏洞存在的可能，因此需要对所有的结果进行人工的分析判断。分析过程参考以下章节的测试项，使用辅助工具或者是手动验证。  业界常用的自动化扫描工具还有WebInspcet，NStalker，Acunetix Web Vulnerability Scanner。在有条件的情况下，可以综合使用。 |
| 测试结果 |  |

### AppScan Web Service 扫描测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ TOOL\_02 |
| 测试用例名称 | AppScan Web Service 扫描测试 |
| 测试目的 | 利用自动化的Web安全扫描工具AppScan进行扫描，以发现Web Service中存在的漏洞 |
| 用例级别 | 1 |
| 测试条件 | 1. 已知Web服务器域名或IP地址 2. Web业务运行正常 3. 目标系统使用了Web Service服务 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 Service 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. 不需修改任何参数，点击next 5. 不需修改任何参数，点击Finish完成配置，开始扫描 6. 扫描完成，保存扫描结果，并对结果进行分析 |
| 预期结果 | 经过分析确认以后的扫描结果中不存在信息提示以上等级的漏洞。 |
| 备注 | 注意：该用例的执行对被测系统的性能影响比较大，而且可能导致一些垃圾数据，建议只在测试环境执行。  由于自动化工具在很多情况下只是提示一种漏洞存在的可能，因此需要对所有的结果进行人工的分析判断。 |
| 测试结果 |  |

## 服务器信息收集

### 运行帐号权限测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ SERVERINFO\_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 | 1 |
| 测试条件 | 1. 已知Web网站IP地址和OS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。    * 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；    * unix：使用“ps –ef|grep java”命令，返回结果第一列的操作系统用户就是运行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 |  |

### Web服务器端口扫描

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_SERVERINFO\_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（如Jboss开放的8083），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 | 1 |
| 测试条件 | 1. 已知Web服务器域名或IP地址，假设IP地址为192.168.1.1 2. 测试用机安装了nmap，假设路径为d:\nmap |
| 执行步骤 | 1. 打开命令提示符，切换到nmap路径下,输入cd /d d:\nmap 2. 运行nmap –n –P0 –sS –sV –p1-65535 –oX scan\_report.xml 192.168.1.1 3. 使用浏览器打开scan\_report.xml 4. 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 |  |

### HTTP方法测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ SERVERINFO\_03 |
| 测试用例名称 | 开放HTTP方法测试 |
| 测试目的 | 有些Web服务器默认情况下开放了一些不必要的HTTP方法（如DELETE、PUT、TRACE、MOVE、COPY），这样就增加了受攻击面。 |
| 用例级别 | 1 |
| 测试条件 | 1. 已知Web网站IP地址及Web访问端口 2. Web业务正常运行 |
| 执行步骤 | 1. 点击“开始”－“运行”，输入cmd并回车，运行cmd.exe 2. 输入telnet *IP* *端口 （其中IP和端口按实际情况填写，用空格隔开）* 3. 回车 4. 在新行中输入如下一行，并回车   OPTIONS / HTTP/1.0   1. 观察返回结果中的Allow的方法列表   返回结果样例：  HTTP/1.1 200 OK  Server: Apache-Coyote/1.1  X-Powered-By: Servlet 2.4; JBoss-4.0.5.GA (build: CVSTag=Branch\_4\_0 date=200610162339)/Tomcat-5.5  Allow: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS  Content-Length: 0  Date: Mon, 29 Jun 2009 08:02:47 GMT  Connection: close  如果返回结果中包含不安全的HTTP方法（DELETE、PUT、TRACE、MOVE、COPY），则验证这些防范是否可用（参考3.2.4 / 3.2.5 / 3.2.6 / 3.2.7 / 3.2.8）   1. 如果方法可用则说明存在漏洞，测试不通过。 |
| 预期结果 | 不包含不安全的HTTP方法（如DELETE、PUT、TRACE、MOVE、COPY） |
| 备注 | 由于不同的Web服务器支持的HTTP协议版本不同，如果系统不支持HTTP/1.0，那么步骤4返回 “HTTP/1.0 400 Bad Request”；这种情况下，应该更改步骤4的输入行为OPTIONS / HTTP/1.1 |
| 测试结果 |  |

### HTTP PUT方法测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ SERVERINFO\_04 |
| 测试用例名称 | HTTP PUT方法测试 |
| 测试目的 | 有些Web服务器开放了PUT方法，攻击者能够通过该方法上传任意文件到Web服务器的一些目录中去。包括Web木马程序。 |
| 用例级别 | 1 |
| 测试条件 | 1. 已知Web网站IP地址 2. Web业务正常运行 |
| 执行步骤 | 1. 点击“开始”－“运行”，输入cmd并回车，运行cmd.exe 2. 输入telnet *IP* *端口 （其中IP和端口按实际情况填写，用空格隔开，比如：telnet 10.70.100.171 8080）* 3. 回车 4. 在新行中拷贝并粘贴以下蓝色内容，然后回车   PUT /alert.txt HTTP/1.0  Content-Length: 16  Hacker accessed!  备注：其中“/alert.txt”可以根据实际应用路径构造，比如“/rbt/alert.txt”   1. 打开IE访问http:// 10.70.100.171:8080/alert.txt 2. 如果响应的页面，显示内容是“Hacker accessed!”，则说明文件已创建，Web服务器开放了PUT方法，存在严重安全漏洞。 |
| 预期结果 | 访问不到内容为“Hacker accessed!”的alert.txt文件 |
| 备注 | 1、由于不同的Web服务器支持的HTTP协议版本不同，如果系统不支持HTTP/1.0，那么步骤4返回 “HTTP/1.0 400 Bad Request”；这种情况下，应该更改步骤4的输入行为PUT /alert.txt HTTP/1.1  2、如果Web服务器上运行着多个Web应用，比如：  http:// 10.70.100.171:8080/application1/  http:// 10.70.100.171:8080/application2/  http:// 10.70.100.171:8080/application3/  那么，必须这些应用分别进行测试，只需要在步骤4中修改路径部分，例如：  PUT /application3/alert.txt HTTP/1.0  Content-Length: 16  Hacker accessed! |
| 测试结果 |  |

### HTTP DELETE方法测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ SERVERINFO\_05 |
| 测试用例名称 | HTTP DELETE方法测试 |
| 测试目的 | 有些Web服务器开放了DELETE方法，攻击者能够通过该方法删除Web服务器上的文件。 |
| 用例级别 | 1 |
| 测试条件 | 1. 已知Web网站IP地址 2. Web业务正常运行 3. Web网站存在/alert.txt文件（如果没有，手工创建） |
| 执行步骤 | 1. 点击“开始”－“运行”，输入cmd并回车，运行cmd.exe 2. 输入telnet *IP* *端口 （其中IP和端口按实际情况填写，用空格隔开，比如：telnet 10.70.100.171 8080）* 3. 回车 4. 在新行中输入如下一行，并回车   DELETE /alert.txt HTTP/1.0   1. 查看Web服务器上alert.txt是否被删除 |
| 预期结果 | Web服务器上alert.txt文件依然存在 |
| 备注 | 1、由于不同的Web服务器支持的HTTP协议版本不同，如果系统不支持HTTP/1.0，那么步骤4返回 “HTTP/1.0 400 Bad Request”；这种情况下，应该更改步骤4的输入行为DELETE /alert.txt HTTP/1.1  2、如果Web服务器上运行着多个Web应用，比如：  http:// 10.70.100.171:8080/application1/  http:// 10.70.100.171:8080/application2/  http:// 10.70.100.171:8080/application3/  那么，必须这些应用分别进行测试，只需要在步骤4中修改路径部分，例如：  DELETE /application3/alert.txt HTTP/1.1 |
| 测试结果 |  |

### HTTP TRACE方法测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ SERVERINFO\_06 |
| 测试用例名称 | HTTP TRACE方法测试 |
| 测试目的 | 有些Web服务器开放了TRACE方法（主要是用于客户端通过向Web服务器提交TRACE请求来进行测试或获得诊断信息），攻击者能够通过该方法构造跨站攻击。 |
| 用例级别 | 1 |
| 测试条件 | 1. 已知Web网站IP地址 2. Web业务正常运行 |
| 执行步骤 | 1. 点击“开始”－“运行”，输入cmd并回车，运行cmd.exe 2. 输入telnet *IP* *端口 （其中IP和端口按实际情况填写，用空格隔开，比如：telnet 10.70.100.1 80）* 3. 回车 4. 在新行中输入如下一行，并回车   TRACE / HTTP/1.0   1. 观察返回结果 |
| 预期结果 | Web服务器的返回信息提示Trace方法“not allowed” |
| 备注 | 由于不同的Web服务器支持的HTTP协议版本不同，如果系统不支持HTTP/1.0，那么步骤4返回 “HTTP/1.0 400 Bad Request”；这种情况下，应该更改步骤4的输入行为TRACE / HTTP/1.1 |
| 测试结果 |  |

### HTTP MOVE方法测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ SERVERINFO\_07 |
| 测试用例名称 | HTTP MOVE方法测试 |
| 测试目的 | 有些Web服务器开放了MOVE方法，用于请求服务器将指定的页面移到另一个网络地址，该方法不安全，容易被利用。 |
| 用例级别 | 1 |
| 测试条件 | 1. 已知Web网站IP地址 2. Web业务正常运行 |
| 执行步骤 | 1. 点击“开始”－“运行”，输入cmd并回车，运行cmd.exe 2. 输入telnet *IP* *端口 （其中IP和端口按实际情况填写，用空格隔开，比如：telnet 10.70.100.1 80）* 3. 回车 4. 在新行中输入如下一行，并回车   MOVE /test/a.html /a.html HTTP/1.0   1. 观察返回结果 |
| 预期结果 | Web服务器的返回信息提示MOVE方法“not supported” |
| 备注 | 由于不同的Web服务器支持的HTTP协议版本不同，如果系统不支持HTTP/1.0，那么步骤4返回 “HTTP/1.0 400 Bad Request”；这种情况下，应该更改步骤4的输入行为MOVE /test/a.html /a.html HTTP/1.1 |
| 测试结果 |  |

### HTTP COPY方法测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ SERVERINFO\_08 |
| 测试用例名称 | HTTP COPY方法测试 |
| 测试目的 | 有些Web服务器开放了COPY方法，用于请求服务器将指定的页面拷贝到另一个网络地址，该方法不安全，容易被利用。 |
| 用例级别 | 1 |
| 测试条件 | 1. 已知Web网站IP地址 2. Web业务正常运行 |
| 执行步骤 | 1. 点击“开始”－“运行”，输入cmd并回车，运行cmd.exe 2. 输入telnet *IP* *端口 （其中IP和端口按实际情况填写，用空格隔开，比如：telnet 10.70.100.1 80）* 3. 回车 4. 在新行中输入如下一行，并回车   COPY /test/a.html /a.html HTTP/1.0   1. 观察返回结果 |
| 预期结果 | Web服务器的返回信息提示COPY方法“not supported” |
| 备注 | 由于不同的Web服务器支持的HTTP协议版本不同，如果系统不支持HTTP/1.0，那么步骤4返回 “HTTP/1.0 400 Bad Request”；这种情况下，应该更改步骤4的输入行为COPY /test/a.html /a.html HTTP/1.1 |
| 测试结果 |  |

### Web服务器版本信息收集

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ SERVERINFO\_09 |
| 测试用例名称 | Web服务器版本信息收集 |
| 测试目的 | 一些情况下，Web服务器能通过隐藏或者修改banner信息的方式防止黑客攻击。这时候我们需要使用不依靠服务器Server标题头的扫描方式进行服务器类型、版本判断。 |
| 用例级别 | 4 |
| 测试条件 | 1. Web业务运行正常 2. 已知Web服务器域名或IP地址 3. 测试用机安装了httprint（Windows环境） |
| 执行步骤 | 1. 运行Httprint\_gui.exe 2. 在Host列中输入主机域名（如果没有域名则输入IP地址），在端口列中输入端口号。如果为HTTPS则要选择锁图标下面的选择框。      1. 点击程序下方的运行按钮      1. 观察程序输出的结果 |
| 预期结果 | 不能够得到Web服务器准确的版本信息 |
| 备注 |  |
| 测试结果 |  |

## 文件、目录测试

### 工具方式的敏感接口遍历

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_DIR\_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 | 2 |
| 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 |
| 执行步骤 | 1. 双击运行DirBuster.jar 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      1. 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 2. 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 3. 其他选项不变，点击右下角的start，启动目录查找      1. 观察返回结果，可点击右下角的report，生成目录报告 |
| 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 | 举一个测试不通过的例子：   |  |  |  | | --- | --- | --- | | Type | Found | Response | | File | /admin/adduser.jsp | 200 | |
| 测试结果 |  |

### Robots方式的敏感接口查找

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ DIR\_02 |
| 测试用例名称 | Robots方式的敏感接口查找 |
| 测试目的 | 为了防止搜索引擎的爬虫访问敏感的目录接口，服务器上可能会编辑一个robots.txt文件，内容为需要保护的文件或目录名称。直接访问robots.txt文件能够获取一些可能的敏感接口 |
| 用例级别 | 2 |
| 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 |
| 执行步骤 | 1. 尝试访问http://www.example.com/robots.txt 例如可能返回如下结果： 2. 观察返回结果 |
| 预期结果 | 通过robots.txt文件不能获取敏感的目录或文件信息。 |
| 备注 | 敏感目录举例：/employee/salary\_files/  敏感文件举例：/sys\_manager/setup.jsp |
| 测试结果 |  |

### Web服务器的控制台

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_DIR\_03 |
| 测试用例名称 | Web服务器的控制台 |
| 测试目的 | 检查是否部署了Web服务器的控制台，控制台是否存在默认帐号、口令，是否存在弱口令。 |
| 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 3. 已知Web服务器版本信息 |
| 执行步骤 | 1. 根据Web服务器的版本信息，通过google查询其对应的控制台URL地址和默认的帐号、口令。以下列出一些常见的Web服务器控制台URL地址和默认帐号、口令：   Tomcat控制台URL：http://www.example.com/manager/html  Tomcat控制台默认帐号admin，默认密码tomcat或admin或空  Tomcat控制台默认帐号tomcat，默认密码tomcat  Jboss控制台URL：http://www.example.com/jmx-console/  Jboss控制台URL：http://www.example.com/web-console/  Jboss控制台默认无须登陆，或者admin/admin  WebSphere控制台URL：http://www.example.com/ibm/console/logon.jsp  WebSphere默认帐号admin，默认密码admin  WebLogic控制台URL：http://www.example.com/console/ （一般console的端口为7001）  WebLogic默认帐号weblogic，默认密码weblogic  Apache控制台URL：http://www.example.com/server-status  Axis2控制台URL：http://www.example.com/axis2-admin/  Axis2控制台默认口令帐户：admin/axis2  iSAP控制台URL：http://www.example.com/admin/login.jsp  iSAP控制台默认的帐号和密码：admin/admin  Virgo控制台URL：http://www.example.com/admin  Virgo控制台默认的帐号和密码：admin/springsource  uniportal控制台URL：http://www.example.com/mc/zh/login.jsp  uniportal控制台默认的帐号和密码：admin/uniportal  Resin3.0控制台URL：http://www.example.com/admin\_manage/  Resin3.1及更新版本的控制台URL：http://www.example.com/resin-admin/  （Resin的控制台应该限制只能本机访问，也就是只能从安装本resin服务器的机器上访问对应resin的控制台）  “普元”管理控制台URL：http://www.example.com/eosmgr/  “普元”管理控制台默认的帐号和密码：sysadmin/000000   1. 在浏览器地址栏中输入Web服务器对应的URL。由于不同的应用可能目录有所差异，如果访问不到，可以登陆后台服务器以find命令查找，比如find / -name “\*console\*”，find / -name “\*admin\*”，看看实际的URL应该是什么，再次尝试。 2. 检查是否存在Web服务器控制台 3. 如果存在控制台，使用默认帐号、口令登录，弱口令登录，查看是否登录成功，如果可以则存在漏洞。 |
| 预期结果 | 不存在Web服务器控制台，或者存在控制台但有强口令。 |
| 备注 | 弱口令例子：123、123456、abc、huawei、admin、tellin、isap、scp等；另外，和用户名相同或非常接近的口令也属于弱口令，比如用户名为tomcat，口令为tomcat1。 |
| 测试结果 |  |

### 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非常容易进行的。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 3. 测试用机上需安装JRE 4. 测试用机上有DirBuster软件 |
| 执行步骤 | 1. 双击运行DirBuster-0.9.8.jar 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      1. 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： 2. 去除Burte Force Files选项 3. 其他选项不变，点击右下角的start，启动目录查找      1. 依次右击Response值为200的行，在出现的菜单中点击Open In Browser      1. 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 |  |

### 文件归档测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ DIR\_05\_01 |
| 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） |
| 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 |  |

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ DIR\_05\_02 |
| 测试用例名称 | 版本控制软件SVN备份文件测试 |
| 测试目的 | 在开发环境时常使用版本控制软件管理代码，常用的有SVN，SVN在管理代码的过程中会自动生成一个名为.svn的隐藏文件夹，svn目录下还包含了以.svn-base结尾的源代码文件副本。但一些网站管理员在发布代码时，不愿意使用‘导出’功能，而是直接复制代码文件夹到WEB服务器上，这就使.svn隐藏文件夹被暴露于外网环境，黑客可以借助其中包含的用于版本信息追踪的‘entries’文件，逐步摸清站点结构。” |
| 用例级别 | 1 |
| 测试条件 | 1、Web业务运行正常并用SVN进行版本管理  2、在生产环境中未删除相关SVN的文件 |
| 执行步骤 | 1、 假设存在这样的目录，如http://www.example.com/xxx。  2、 打开浏览器，直接在URL后面加上“.svn/entries”，查看是否出现目录结构、文件等敏感信息  3、 假设存在这样的文件，如http://www.example.com/xxx.jsp。  打开浏览器，直接在URL后面加上“.svn/text-base/xxx.jsp.text-base”，查看是否出现该jsp文件的源码。 |
| 预期结果 | 不能访问以下链接获得目录结构和页面源代码等敏感信息 |
| 备注 | 发现有SVN漏洞后可以用Seay-SVN这个软件把整个Web站点下载下来或在线管理，如图：  无标题6.png |
| 测试结果 |  |

## 认证测试

暴力破解是目前最直接有效的攻击方式，特别对于电信业务来说，很多情况下口令都为6位纯数字，很容易被攻击。本测试项在于检查认证系统对暴力破解的防护性。

在以下的一些测试中，围绕能否满足暴力破解进行的设计，未设计直接进行暴力破解的攻击用例。如果需要，测试人员可以使用hydra和AppScan中集成的Authentication Tester工具进行。

### 验证码测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_AUTHEN\_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 | 1 |
| 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：    * 请求登陆页面，得到生成的验证码    * 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    * 填入错误的用户名和口令，填入正确的验证码，提交表单    * 从WebScarab拦截数据中复制对应登陆请求的POST或GET消息（文本格式），将其中的口令更改一个字符    * 在命令行中输入telnet <服务器域名或IP> <端口>，回车    * 将修改的内容粘贴到命令行窗口中，回车    * 判断返回的页面中是否包含“验证码错误”（或类似）的提示，如果没有，说明存在漏洞，完成测试 |
| 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 |  |

### 认证错误提示

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ AUTHEN\_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 | 1 |
| 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noexists，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noexists 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 |  |

### 锁定策略测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ AUTHEN\_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 | 1 |
| 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 |  |

### 认证绕过测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ AUTHEN\_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 | 1 |
| 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 |  |

### 找回密码测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ AUTHEN\_05 |
| 测试用例名称 | 找回密码测试 |
| 测试目的 | 网站在密码重设和密码找回功能上如果存在着缺陷，攻击者可以通过此功能找到指定用户的密码，更改指定用户的密码让其不能登陆，造成业务数据修改等。 |
| 用例级别 | 1 |
| 测试条件 | 1. 已知Web网站地址 2. Web业务正常运行 3. 网站提供密码重设或密码找回的功能 4. 已知某正确的用户账号 |
| 执行步骤 | 1. 打开密码找回功能的页面 2. 如果没有对用户的合法身份进行任何验证就发送密码，则说明存在漏洞。 3. 系统正常需要用户输入一些能够证明其合法身份的信息（比如回答一些原来注册用户时填写的一些信息，比如小学教师姓名等），如果是回答问题的方式，则判断问题是否类似于“我的生日”或“我的姓氏”这种答案限定范围非常小的问题。如果是说明存在漏洞，完成测试，否则进行下一步。 4. 如果为输入密码的方式，则查看该页的html源代码 5. 在html源代码中查看是否存在着可能为密码的数据，如果是，说明存在漏洞 6. 用户的密码一般通过用户邮箱或者手机短信的方式来通知用户，如果在返回的Web页面直接显示明文口令，则说明存在漏洞。 |
| 预期结果 | 密码找回不存在漏洞 |
| 备注 |  |
| 测试结果 |  |

### 修改密码测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ AUTHEN\_06 |
| 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 | 1 |
| 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 |  |

### 不安全的数据传输

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ AUTHEN\_07 |
| 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 | 1 |
| 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 | 参考Web应用安全开发规范章节3.5.3 敏感数据传输 |
| 测试结果 |  |

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ AUTHEN\_08 |
| 测试用例名称 | 修改密码信息机密性保护 |
| 测试目的 | 测试Web程序在修改密码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 | 2 |
| 测试条件 | 1. 已知Web网站地址 2. Web业务正常运行 3. Web业务存在修改密码的功能 |
| 执行步骤 | 1. 登陆Web网站 2. 找到修改密码的页面 3. 输入旧密码和新密码并提交 4. 并观察数据传输是否采用HTTPS方式 |
| 预期结果 | 用户名和密码信息采用HTTPS传输。 |
| 备注 | 创建用户时如果包含密码也应该HTTPS传输。  参考Web应用安全开发规范章节3.5.3 敏感数据传输 |
| 测试结果 |  |

### 强口令策略测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ AUTHEN\_08 |
| 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 | 2 |
| 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 | 1. 使用正确的用户、口令登陆Web业务系统 2. 打开口令修改页面 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。 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。 5. 观察结果 |
| 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 | 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。  对于一些Web应用（如移动网上客服系统）密码只能是数字组成，则不强制要求强口令。  **Web应用安全开发规范中的强口令策略：**  规则3.2.1.1：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。  规则3.2.1.2：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。  规则3.2.1.3：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。  规则3.2.1.4：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。  规则3.2.1.5：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。  规则3.2.1.6：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。  规则3.2.1.7：口令历史记录数可配置，取值范围为：0-30；建议默认：3个。  规则3.2.1.8：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。  规则3.2.1.9：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。  规则3.2.1.10：口令不能以明文的形式在界面上显示。  规则3.2.1.11：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。  规则3.2.1.12：只有当用户通过认证之后才可以修改口令。  规则3.2.1.13：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。  建议3.2.1.1：实现弱口令词典功能。 |
| 测试结果 |  |

## 会话管理测试

### 身份信息维护方式测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_SESSION\_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 | 1 |
| 测试条件 | 1. 已知Web网站地址 2. Web业务正常运行 3. Web业务存在不同级别的权限（角色） |
| 执行步骤 | 1. 登录系统 2. 配置WebScarab进行http get和post请求拦截 3. 请求Web页面 4. 在WebScarab中查看请求报文 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。 |
| 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 |  |

### Cookie存储方式测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ SESSION\_02 |
| 测试用例名称 | Cookie存储方式测试 |
| 测试目的 | 某些Web应用将SesssionId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 | 1 |
| 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 |  |

### 用户注销登陆的方式测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ SESSION\_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 | 1 |
| 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 |  |

### 注销时会话信息是否清除测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ SESSION\_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 | 1 |
| 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 |  |

### 会话超时时间测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ SESSION\_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 | 1 |
| 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web.xml文件中的session-timeout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 |  |

### 会话定置测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ SESSION\_06 |
| 测试用例名称 | 会话定置（session fixation）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）为受害着确定一个会话标识，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 | 1 |
| 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 | 。 |
| 测试结果 |  |

### 会话标识携带

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ SESSION\_07 |
| 测试用例名称 | 会话标识携带 |
| 测试目的 | 登录成功后，检查URL参数中是否携带了会话标识，URL的参数和值容易泄漏，如果URL携带了会话标识，那么会话标识很容易泄漏，会话容易被盗用。 |
| 用例级别 | 1 |
| 测试条件 | 1. 已知Web网站地址 2. Web业务运行正常 3. 已知正确的用户名、口令 |
| 执行步骤 | 1. 访问登录页面，并成功登陆。 2. 检查浏览器地址栏中的URL及参数，是否包含会话标识（例如：jsessionid或sid等）；如果包含，则存在安全漏洞。 |
| 预期结果 | URL参数中没有携带会话标识。 |
| 备注 |  |
| 测试结果 |  |

### 会话标识随机性测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ SESSION\_08 |
| 测试用例名称 | 会话标识随机性测试 |
| 测试目的 | 会话标识是用户身份的唯一标识，如果该标识被盗用、窃取、猜测则导致非授权者盗用用户身份登录系统，因此会话标识必须具备强随机性，防止非授权者猜测 |
| 用例级别 | 2 |
| 测试条件 | 1、Web业务系统登录运行正常  2、已知正确的用户名、口令 |
| 执行步骤 | * 1. 配置burpsuite为浏览器代理，如图：     2、配置完成后，访问登录页面，输入正确的用户名和密码。点击提交后burpsuite将会拦截请求，例如：  3、在拦截的请求中右键点击弹出菜单，选择“Send to Sequencer”，这时看到“Sequencer”标签高亮，点击“Sequencer”标签，如图：  4、这时可看到请求已自动带进“Sequencer”,只需在“cookie”这栏选择请求中的会话标识即可，基于JavaEE架构的会话标识符一般都以JSESSIONID命名，最后点击“Start live capture”开始发送大量请求获取足够数量的会话标识符，如图：  无标题4.png  5、等待进度条完成后，点击“Analyze now”即可在下方看到报告，这时只看红色圈部分显示“excellent”即可。 |
| 预期结果 | 测试结果显示为“excellent”。 |
| 备注 | 点击“Save tokens”可保存扫描过的会话标识，可供其他软件使用。 |
| 测试结果 |  |

## 权限管理测试

目前存在着两种越权操作类型：横向越权操作和纵向越权操作。前者指的是攻击者尝试访问与他拥有相同权限的用户的资源；而后者指的是一个低级别攻击者尝试访问高级别用户的资源。下图能够比较清晰的描绘出它们之间的关系：

横向越权

**普通管理员C**

**普通用户A**

**普通用户B**

**普通管理员D**

**超级管理员E**

纵向越权

权限管理测试更多的是进行人工分析，自动化工具无法了解页面的具体应用场景以及逻辑判断过程。因此这里的测试需要首先测试人员理解测试业务系统的逻辑处理流程，并在此基础上进行如下测试。

这里有几个测试的参考依据：

* 页面是否进行权限判断；
* 页面提交的资源标志是否与已登陆的用户身份进行匹配比对；
* 用户登陆后，服务器端不应再以客户端提交的用户身份信息为依据，而应以会话中保存的已登陆的用户身份信息为准；
* 必须在服务器端对每个请求URL进行鉴权，而不能仅仅通过客户端的菜单屏蔽或者按钮Disable来限制。

### 横向测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_PRIVI\_01\_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 | 1 |
| 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的身份登陆到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 |  |

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ PRIVI\_01\_02 |
| 测试用例名称 | 基于资源ID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 | 1 |
| 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. Web业务存在相同级别的两个用户，假设分别为userA和userB 4. 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资源的标志（假设为resource\_id） 5. 测试用机安装了WebScarab软件（本测试基于20070504版本来描述） |
| 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的身份登陆到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resource\_id参数的值为userB所属的资源id，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。 |
| 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 |  |

### 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 | SEC\_Web\_ PRIVI\_02\_01 |
| 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 | 1 |
| 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 | SEC\_Web\_ PRIVI\_02\_02 |
| 测试用例名称 | 基于源代码的测试 |
| 测试目的 | 发现页面中存在的纵向越权操作。 |
| 用例级别 | 3 |
| 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Web应用的源代码，假设存放目录为测试用机的c:\webapp目录，且假设页面为jsp 4. 已知本目录对应的Web路径，假设为http://192.168.1.1/webapp/ |
| 执行步骤 | 1. 使用无权限的用户身份访问Web应用，保持浏览器窗口。 2. cd /d c:\webapp 3. dir /B \*.jsp >list.html 4. 使用ultraedit打开list.html 5. 点击Search->Replace菜单 6. 在弹出的对话框中使用 **">test</a>^p<a href="http://192.168.1.1/webapp/** 替换 **^p** 7. 用测试第一步中保持的浏览器打开list.html文件 8. 依次点击test连接 9. 观察结果，如果能够访问，则进行记录 10. 使用的低级别的用户身份登陆系统，保持浏览器窗口 11. 重复8，9步操作 12. 观察结果 |
| 预期结果 | 不能够访问页面（如提示“拒绝访问”等信息）。 |
| 备注 |  |
| 测试结果 |  |

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ PRIVI\_02\_03 |
| 测试用例名称 | 基于crawling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 | 1 |
| 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 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      1. 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      1. 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          1. 然后点击ok返回扫描配置对话框，如第7步的图示，点击next 2. 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 3. 扫描完成，保存结果 4. 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 |  |

### 跨站伪造请求测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_CSRF\_03 |
| 测试用例名称 | 跨站请求伪造测试（CSRF） |
| 测试目的 | 如果网站在关键的业务操作（例如转账操作、增删改用户角色操作等等）没有添加相应的令牌机制（用于识别是否授权人员本人操作），则非授权人员有可能通过伪造请求利用已授权人员的身份执行非自愿的操作。 |
| 用例级别 | 1 |
| 测试条件 | 1. Web业务运行正常 |
| 执行步骤 | 1、登陆网站，寻找网站的关键性操作（视具体的业务系统而定），这里只以通用的增加用户操作为例。由于目前的表单大多以Post方法提交为主，因此这里以Post方法为例。打开添加用户页面   1. 配置burpsuite为浏览器代理，如图：        1. 配置完成后，点击网页上添加用户的按钮提交请求，burpsuite将会拦截请求，请求中包括提交的字段以及提交的目标URL。   无标题10.png   1. 假设提交的请求中包括用户名（testUserName）和密码（testPassWord）两个字段，提交的目标URL为http://xxx.xxx.xxx.xxx/xxx/test.action，编写测试页面，代码如下：      1. 将此测试页面放入浏览器中运行，如Web业务系统中已成功添加用户名testN，则证明该Web业务系统存在跨站伪造请求的漏洞。   注意：运行测试页面时必须保持会话状态，即保持登录状态。 |
| 预期结果 | 成功添加用户名testN。 |
| 备注 | 1、如果表单是Get方式提交，则测试页面更简单，代码如下：    2、编写页面时尽量带上所有字段或是带上必填字段，这样才能增加表单提交的成功率。 |
| 测试结果 |  |

## 文件上传下载测试

### 文件上传测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_FILE\_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 | 1 |
| 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 |  |

### 文件下载测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ FILE\_02\_01 |
| 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 | 1 |
| 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下载自己的工资清单文件http://www.example.com/download/usera/salary.xls） |
| 执行步骤 | 1. 猜测并更改URL路径   [http://www.example.com/download/userb/salary.xls](http://www.exmaple.com/download/userb/salary.xls)  [http://www.example.com/download/userc/salary.xls](http://www.exmaple.com/download/userc/salary.xls)  http://www.example.com/admin/report.xls  ……   1. 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 |  |

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ FILE\_02\_02 |
| 测试用例名称 | 文件下载测试2 |
| 测试目的 | 很多网站可能接受类似于文件名的参数用于下载或者显示内容。如果未进行严格判断的话，攻击者可以通过修改这个参数值来下载、读取任意文件内容。比如、/etc/password，甚至是数据库连接配置文件以及源代码等等。 |
| 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/viewfile.do?filename=report.xls） |
| 执行步骤 | 1. 更改参数的值为其他路径和文件 2. 在浏览器地址栏中尝试以下URL   [http://www.example.com/viewfile.do?filename=../etc/passwd](http://www.exmaple.com/viewfile.do?filename=../etc/passwd)  [http://www.example.com/viewfile.do?filename=../../etc/passwd](http://www.exmaple.com/viewfile.do?filename=../etc/passwd)  [http://www.example.com/viewfile.do?filename=../../../etc/passwd](http://www.exmaple.com/viewfile.do?filename=../etc/passwd)  [http://www.example.com/viewfile.do?filename=../../../../etc/passwd](http://www.exmaple.com/viewfile.do?filename=../etc/passwd)  [http://www.example.com/viewfile.do?filename=../../../../../etc/passwd](http://www.exmaple.com/viewfile.do?filename=../../../../../etc/passwd)  ……  对于UNIX/Linux服务器可以尝试下载/etc/passwd；对于Windows服务器可以尝试下载c:\boot.ini文件   1. 观察页面返回信息，如果可以获取到passwd或boot.ini文件，则说明存在漏洞。 |
| 预期结果 | 不能获取到passwd或boot.ini或JSP源代码等文件。 |
| 备注 | 该漏洞属于OWASP 2007年十大Web安全漏洞的“不安全的直接对象引用”。 |
| 参考资料 |  |

## 信息泄漏测试

泄露出的敏感信息包括但不限于：数据库连接地址、帐号和口令等信息、服务器系统信息、Web服务器软件名称、版本、Web网站路径、除html之外的源代码、业务敏感数据等。

### 连接数据库的帐号密码加密测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_LEAKAGE\_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 | 1 |
| 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 |  |

### 客户端源代码敏感信息测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_LEAKAGE\_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 | 1 |
| 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 |  |

### 客户端源代码注释测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_LEAKAGE\_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 | 1 |
| 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 |  |

### 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

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_LEAKAGE\_04 |
| 测试用例名称 | 不存在的URL |
| 测试目的 | Web网站在处理用户提交的不存在的URL时会返回错误信息，我们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。 |
| 用例级别 | 1 |
| 测试条件 | 1、已知Web网站地址，假设为：http://www.example.com 或 IP：http://aa.bb.cc.dd  2、Web业务运行正常 |
| 执行步骤 | 1. 我们请求不存在的文件，比如：   http://www.example.com/unexist.jsp 或者 http://aa.bb.cc.dd/unexist.jsp   1. 观察返回结果 |
| 预期结果 | 返回的页面中没有敏感信息。 |
| 备注 | 在各个目录和功能模块重复此操作 |
| 测试结果 |  |

* 非法字符

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_LEAKAGE\_05 |
| 测试用例名称 | 非法字符导致信息泄漏 |
| 测试目的 | Web应用在处理用户提交的含有特殊字符的URL时，可能会返回错误的信息，通过错误信息来判断是否存在敏感信息的泄漏问题。 |
| 用例级别 | 1 |
| 测试条件 | 1、 已知Web网站地址  2、 Web业务正常运行  3、 如果是IE浏览器，在Internet选项->高级里把“显示友好HTTP信息”去掉选择状态  4、 已知待测目标URL，假设为http://www.example.com/page.xxx  待测目标存在参数输入，假设为name=value，即：  http://www.example.com/page.xxx?name= value |
| 执行步骤 | 1. 在正常的URL的参数中添加特殊字符： % \* ; ’ ?   添加特殊字符后的URL为:  http://www.example.com/page.xxx?name= value%  http://www.example.com/page.xxx?name= value\*  http://www.example.com/page.xxx?name= value;  http://www.example.com/page.xxx?name= value?  http://www.example.com/page.xxx?name= value’   1. 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 2. 观察返回结果 |
| 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 | 页面可能接受多个参数，需对每个参数都进行测试  POST情况下的参数也需要测试 |
| 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。

### HappyAxis.jsp页面测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_LEAKAGE\_05 |
| 测试用例名称 | HappyAxis.jsp页面测试 |
| 测试目的 | 如果系统采用Axis来发布Web Service，则默认情况下存在HappyAxis.jsp页面，该页面会泄漏服务器的敏感信息。 |
| 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Web服务器的操作系统 2. 用find命令查找是否存在HappyAxis.jsp页面   find ./ -name "?appy?xis.jsp"   1. 利用查询到的目录信息构造访问HappyAxis.jsp页面的URL，通过Web方式访问该页，例如   http://www.example.com/axis2/happyaxis.jsp   1. 如果能访问到happyzxis.jsp则存在漏洞 |
| 预期结果 | 不存在可以通过Web方式访问的HappyAxis.jsp文件。 |
| 备注 |  |
| 测试结果 |  |

### Web服务器状态信息测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_LEAKAGE\_06 |
| 测试用例名称 | Web服务器状态信息测试 |
| 测试目的 | Web服务器默认会提供服务器状态信息查询功能，该功能容易泄漏系统信息，从而被攻击者利用。 |
| 用例级别 | 2 |
| 测试条件 | 1、已知Web网站地址  2、Web业务正常运行 |
| 执行步骤 | 1. 访问http://www.example.com/status?full=true。 2. 观察返回结果，是否包含服务器的敏感信息。 |
| 预期结果 | 查询不到Web服务器的状态信息 |
| 备注 | 适用于Tomcat、Jboss服务器 |
| 测试结果 |  |

### 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。
* 公用文件头（如数据库连接信息、源代码头文件等）是否采用不被服务器处理的后缀（如inc，它将作为文本格式直接输出）进行存储。

## 输入数据测试

很多Web应用开发人员图简便，只是在客户端通过script对用户输入的数据进行合法性校验，在服务器端没有对输入数据的合法性进行再次校验。客户端的输入检验，根本起不到安全作用，可以通过HTTP代理轻易绕过。方法如下：

* 1. 开启WebScarab，配置对GET和POST请求进行拦截；
  2. 在浏览器中配置代理服务器IP为127.0.0.1，端口为8008；
  3. 在Web页面中输入合法数据，并提交；
  4. 此时WebScarab会弹出HTTP请求拦截窗口，修改对应的输入参数值为不合法数据（根据测试需要，构造输入数据）；
  5. 点击“Accept Changes”按钮，提交数据。这样就绕过了客户端的输入校验，向服务端提交不合法的输入数据。

### 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。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 | SEC\_Web\_INPUT\_01\_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 | 1 |
| 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） 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。 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 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 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 |
| 预期结果 | 不存在注入点 |
| 备注 | 1. 页面可能接受多个参数，需对每个参数都进行测试 2. 如果客户端script对输入数据进行合法行校验，阻止非法数据，可以通过3.9介绍的方法（通过WebScarab拦截并修改参数值），绕过客户端数据校验。 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） 4. 本测试包含了现有最常见的两种测试方法 |
| 测试结果 |  |

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ INPUT\_01\_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 | 1 |
| 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. 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 |  |

### LDAP注入测试

LDAP是轻量目录访问协议，英文全称是Lightweight Directory Access Protocol，一般都简称为LDAP。它是基于X.500标准的，但是简单多了并且可以根据需要定制。与X.500不同，LDAP支持TCP/IP，这对访问Internet是必须的。从另一个意义上 LDAP是实现了指定的数据结构的存贮，它是一种特殊的数据库。但是LDAP和一般的数据库不同，明确这一点是很重要的。LDAP对查询进行了优化，与写性能相比LDAP的读性能要优秀很多。

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_INPUT\_02 |
| 测试用例名称 | LDAP注入测试 |
| 测试目的 | 由于LDAP类似于数据库，必然有查询数据的功能，数据库由于未过滤用户输入而导致SQL注入，LDAP也一样，如未进行过滤也会引起LDAP注入。 |
| 用例级别 | 1 |
| 测试条件 | 1、Web业务运行正常并在某些模块使用LDAP来存储，例如用户名等  2、Web服务端接受未经过滤的用户输入对LDAP进行查询 |
| 执行步骤 | 1、由于对LDAP注入的测试并未向SQL注入一样有个大概的通用测试方法，LDAP注入的测试很大程度依靠具体的代码去测试，下面仅给出一个演示的例子，对于具体的问题具体分析：  2、假设存在一个登录框需要输入用户名和密码才能登录，用户输入提交后会传到后台的LDAP服务器进行查询，查询代码如下：(&(USER=Uname)(PASSWORD=Pwd))  3、“&”是一个过滤器，查询满足以上两个条件的数据方可登录成功。  4、由于未过滤用户输入，非法用户输入在用户名输入Uname)(&))并在密码输入Pwd（可任意），后台的查询被构造成  (&(USER= Uname)(&)(PASSWORD=Pwd))  5、由于LDAP服务器只处理第一个过滤器，即仅查询(&(USER=Uname)(&))得到了处理。这个查询永真，因而攻击者无需有效的密码就能绕过验证对系统进行访问。 |
| 预期结果 | 不存在LDAP注入点 |
| 备注 | 1、手工寻找LDAP注入点比较困难，可使用扫描工具扫描。  2、LDAP注入原理其实和SQL注入一样，只是查询的语言不一样而已，需了解LDAP查询语言。 |
| 测试结果 |  |

### MML语法注入

MML语法注入是我司特有的一种漏洞。MML语句通过分号(;)可以执行多语句。如果在缺少对输入参数进行严格控制的情况下，攻击者能够执行任意的MML语句。

由于结果判断的过程中无法给出准确的判断准则，所以这里也就没有办法给出详细的具体步骤，这里给出一下基本步骤:

1. 找出提交给MML处理的参数
2. 在参数后面加入分号(;)提交请求
3. 观察返回结果

如果结果返回正常可能就存在该漏洞。这里有一种情况：如果参数并未作为MML语句执行中的一个参数值，那么返回的结果也可能是正确的，这时候就需要测试人员自己进行判断了。

### 命令执行测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ INPUT\_04 |
| 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 GuestHelpAssistant SUPPORT\_388945a0 |
| 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 |  |

### 回车换行符（CRLF）注入测试

CRLF注入在Web系统中又称HTTP响应拆分漏洞，是指Web系统未过滤客户端传来的回车符%0d、换行符%0a导致服务器当成数据解释而形成的一种注入，常出现在Location、Cookie以及自定义header等HTTP响应头。

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_INPUT\_05 |
| 测试用例名称 | 回车换行符CRLF注入测试 |
| 测试目的 | 由于未正确过滤客户端的回车换行符，导致非法人员构造特殊的请求发送到服务端，易形成跨站脚本攻击。 |
| 用例级别 | 1 |
| 测试条件 | 1、Web业务运行正常  2、系统中客户端发送的请求内容出现在HTTP响应头中并且未过滤回车换行符 |
| 执行步骤 | 1、寻找特殊的URL，执行该URL允许客户端发送的内容出现在HTTP响应头中。  2、配置burpsuite为浏览器代理，如图：       1. 配置完成后，访问该URL。点击提交后burpsuite将会拦截请求，例如：   无标题8.png   1. 修改存在问题的参数值为%0d%0a%20SomeCustomInjectedHeader:injected\_by\_wvs（%20后面是任意的键值对），然后发送请求。 2. 观察响应头，如果在HTTP响应头中独立一行出现SomeCustomInjectedHeader:injected\_by\_wvs的自定义HTTP头，则证明存在CRLF注入漏洞，如图：   无标题9.png |
| 预期结果 | 不存在CRLF注入点 |
| 备注 | 1、手工寻找客户端发送的请求内容出现在HTTP响应头中这种URL比较烦琐并困难，可先使用扫描工具如AWVS扫描漏洞，再利用burpsuite去手工验证。 |
| 测试结果 |  |

### XML注入测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ INPUT\_06 |
| 测试用例名称 | XML注入测试 |
| 测试目的 | 可扩展标记语言 (Extensible Markup Language, XML) ，用于标记电子文件使其具有结构性的标记语言，可以用来标记数据、定义数据类型，是一种允许用户对自己的标记语言进行定义的源语言。 XML是标准通用标记语言 (SGML) 的子集，非常适合 Web 传输。XML 提供统一的方法来描述和交换独立于应用程序或供应商的结构化数据。在解析XML的过程中如果未过滤用户的输入则可能引起注入问题。 |
| 用例级别 | 1 |
| 测试条件 | 1、Web业务运行正常  2、Web系统接受用户发送XML请求，服务端解析XML数据并发送给用户。 |
| 执行步骤 | XML注入主要分为内容注入和实体注入。内容注入分为XML数据注入、样式表注入、XPATH/XQuery注入。实体注入分为实体循环DDOS、XML炸弹DDOS、外部实体注入。  1、XML数据注入：  a)、假设如下图的XML被用于从客户端提交到Web服务端。price的值不可控，quantity和address项的值可控，即可以由用户自由输入。  wKioOVIawjHQ6GyNAAA4IUckzOU462.jpg  b)、在quantity的值中构造"<!--"注释符，然后在address的值中先闭合前面的注释符，然后再重新根据原xml文件结构重新构造xml文件提交到Web服务器，这时候price元素的值也是可以由攻击者随意构造的，如下图：  wKioOVIaxA-iFRqvAAA41qmYjp0837.jpg  c)、查看订单价格是否被修改。  2、样式表注入  a)、除了注入数据到XML之外，作为XML注入的结果，使代码运行也是可能的。在XML文件中允许使用样式表，这种样式表能够把已有的XML数据转换成新的XML数据，这种新的XML文档通常以HTML的方式在Web浏览器中展现。假设XML引用了一个超链接：<link>Attacker Text</link>，Attacker Text是用户可控，服务端利用XSLT语言将上述XML转换为如下的HTML，并传送给Web浏览器：<A HREF="Attacker Text">Attacker Text</A>  b)、将Attacker Text改为"#" onclick="alert(1)" /><!-- 提交到服务器查看是否会引起XSS跨战脚本攻击。  3、XPATH/XQuery注入  a)、XPath和XQuery是能够查询XML文档的语言，类似于结构化查询语言（SQL），假设存在以下XML包含了姓名和社保号：    b)、系统通过以下语句查询XML数据，如：  //\*[contains(name，'Attacker-Data')]/name  c)、Attacker-Data是用户可控制的，修改以上语句为  //\*[contains(name，'x')] | //\*| //\*[contains(name，'y')]/name  d)、因为加了|//\*，|是或运算符，//\*查询所有，查看是否返回所有数据。  4、实体循环DDOS  a)、发送以下XML代码到服务器，如：  <!ENTITY % xx '&#x25;zz;'>  <!ENTITY % zz '&#x25;xx;'>  %xx;  b)、当执行到最后一行，%xx会变成%zz，然后%zz又会变成%xx，实体转换变成一个死循环。  c)、查看XML解析器在解析时是否被拒绝服务攻击，即Web系统是否被DDOS。  5、XML炸弹DDOS  a)、发送以下XML代码到服务器，如：  <?xml version="1.0" encoding="utf-8"?>  <!DOCTYPE something [  <!ENTITY x0 "Developers!">  <!ENTITY x1 "&x0;&x0;">  <!ENTITY x2 "&x1;&x1;">  <!ENTITY x3 "&x2;&x2;">  <!ENTITY x4 "&x3;&x3;">  ...  <!ENTITY x100 "&x99;&x99;">]>  <something>&x100;</something>  b)、当执行到最后一行，最终字符串变成了一个超大字符串。  c)、查看XML解析器在解析时是否被拒绝服务攻击，即Web系统是否被DDOS。    6、外部实体注入  a)、实体能够引用文件的内容，也可以通过指定外部URL，如发送以下代码：（windows服务器）  <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE myTest [  <!ELEMENT secTest ANY>  <!ENTITY xxe SYSTEM "c：/boot.ini">]>  <secTest>&xxe;</secTest>  b)、查看是否显示服务器C盘下的boot.ini配置内容。 |
| 预期结果 | 不存在XML注入，对用户输入进行过滤。 |
| 备注 |  |
| 参考资料 |  |

## 跨站脚本测试

### GET方式跨站脚本测试

|  |  |
| --- | --- |
| 编号 | SEC\_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 | 1 |
| 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“查看源文件” 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。 4. 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   </textarea><script>alert(123456)</script>  '><script>alert(123456)</script> "><script>alert(123456)</script> </title><script>alert(123456)</script> --><script>alert(123456)</script> [img]javascript:alert(123456)[/img] <scrip<script>t>alert(123456)</scrip</script>t> </div><Script>alert(123456)</script> |
| 预期结果 | 不存在跨站脚本漏洞 |
| 备注 | 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。  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。 |
| 测试结果 |  |

### POST方式跨站脚本测试

|  |  |
| --- | --- |
| 编号 | SEC\_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 | 1 |
| 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. 待测目标以POST方式提交参数，显示为表单方式 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。 <script>alert(123456)</script> 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” 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。 4. 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   </textarea><script>alert(123456)</script>  '><script>alert(123456)</script> "><script>alert(123456)</script> </title><script>alert(123456)</script> --><script>alert(123456)</script> [img]javascript:alert(123456)[/img] <scrip<script>t>alert(123456)</scrip</script>t> </div><Script>alert(123456)</script> |
| 预期结果 | 不存在跨站脚本漏洞 |
| 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 |  |

### URL跨站脚本测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ XSS\_03 |
| 测试用例名称 | URL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 | 1 |
| 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 |
| 执行步骤 | 1. 在浏览器地址栏中输入http://www.example.com/<script>alert(123456)</script> .xxx，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。 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” 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。 |
| 预期结果 | 不存在跨站脚本漏洞 |
| 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 |  |

## 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：某个网上用户看书流程，首先验证用户名和口令，正确后跳到下一步选择书名，然后再到下一步看书等行为。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想要看的书名对应的链接，直接在浏览器中输入链接，就能看到。
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：对于某个定购系统存在两种金额处理的流程，假设为定购与赠送。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商品价格的逻辑测试（假如将金额更改为负数）情况。

## 搜索引擎信息收集

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：

* 搜索现网已存在并被公开的漏洞，比如搜索“xxx系统 漏洞”、“入侵xxx系统”…
* 存在但是没有公开的漏洞，比如搜索目录列表漏洞：

intitle:"index of /" site:www.example.com

* 搜索目标系统隐藏的管理接口，比如：“login site:www.example.com”或者“管理登录 site:www.example.com”

## Web Service测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_SERVICE\_01 |
| 测试用例名称 | Web Service测试用例 |
| 测试目的 | 通过工具自动检查Web Service接口是否存在SQL注入、XPATH注入、跨站脚本漏洞。 |
| 用例级别 | 1 |
| 测试条件 | 1. Web Service运行正常 2. 已知某接口WSDL地址http://PortalIP:Port/Path/Interface?wsdl 3. 已知每个接口方法返回结果的含义 |
| 执行步骤 | 1. 运行WSDigger 2. 点击右下角的Next按钮进入WSDL页面 3. 在WSDL输入框中输入接口的WSDL地址http://PortalIP:Port/Path/Interface?wsdl      1. 点击Get Methods按钮，获取接口方法      1. 在左侧的接口方法树形框中选择要测试的方法      1. 选择菜单Attacks->Select Attack Type，弹出攻击类型对话框，在对话框左侧的选择列表框中选择所有的测试项  |  | | --- | |  |  1. 点击Start      1. 观察返回的结果，根据WSDL接口定义的返回事件判断是否成功 |
| 预期结果 | 均不能返回正确的结果。 |
| 备注 | 目前WSDigger支持：SQL注入、XSS、XPath注入三种攻击。 |
| 测试结果 |  |

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_SERVICE\_02 |
| 测试用例名称 | Web Service测试用例2 |
| 测试目的 | 检查接口认证、鉴权、机密性、完整性、审计日志措施是否恰当。 |
| 用例级别 | 1 |
| 测试条件 | 1. Web Service运行正常 2. 已知某接口WSDL地址http://PortalIP:Port/Path/Interface?wsdl 3. 已知每个接口方法返回结果的含义 |
| 执行步骤 | 1. 检查Web Service接口传递SOAP消息是否使用HTTPS传输，查看WSDL地址即可。 2. 检查Web Service是否存在授权机制，主要是检查是否使用（但不限于）以下机制的其中一种：   a)、检查调用的方法是否提供额外的参数作为验证，例如getData (int id，String username，String password)方法除了正常的业务参数id之外，还提供用于验证调用者身份的参数。  b)、WebService的工程目录是否使用自带的users.lst作为验证机制，以Axis为例，检查工程目录下的WEB-INF下是否存在users.lst文件，并且server-config.wsdd文件是否配置了以下handler:    c)、如果Web Service仅需要对少数、特定的IP开放，那么建议服务器端接受到Web Service调用请求时，先判断客户端IP是否在允许范围内。检查WebService的工程目录的src目录下是否存在ip.properties之类的属性文件，该文件配置允许的客户端IP，并且server-config.wsdd文件是否配置了以下handler:    d)、检查Web Service提供方是否使用SSL服务器端证书进行身份验证，Web Service调用方是否使用SSL服务器端证书进行认证。   1. 检查Web Service服务器对提交的参数是否进行输入校验，可利用WSDigger发送非正常的请求并查看方法返回值。 2. 检查对Web Service接口调用是否进行日志记录，日志内容包括但不限于如下内容：调用时间、操作类型、调用接口名称、详细的接口参数、客户端IP、客户端机器名、调用者的用户标识、受影响的个体（数据、资源）、成功或失败标识。 3. 检查Web Service接口是否通过时间戳或流水号的机制防止对Web Service接口的重放攻击。采用时间戳机制时，必须同时确保客户端和服务器间的时间同步。 4. 检查Web Service接口是否限制SOAP消息的大小，避免导致Web Service拒绝服务。 |
| 预期结果 | Web Service接口存在完善的认证、鉴权机制，存在机密性、完整性保护措施，对接口调用有详细的调用日志记录。 |
| 备注 | 主流的web扫描工具如Appscan、WebInspect、AWVS都可以对webservice进行漏洞扫描。 |
| 测试结果 |  |

## DWR（Direct Web Remoting）测试

|  |  |
| --- | --- |
| 编号 | SEC\_DWR\_01 |
| 测试用例名称 | DWR测试用例 |
| 测试目的 | 检查网站是否使用DWR，是否存在安全漏洞。 |
| 用例级别 | 1 |
| 测试条件 | 1. Web业务运行正常 2. 测试PC安装Fiddler软件 |
| 执行步骤 | 1. 可以在portal对应的服务器上的相应目录下运行：   find ./ -name "dwr.\*"  如果存在dwr.xml和dwr.jar两个文件，则说明portal存在DWR，再执行步骤2   1. 检查对应的web.xml有没有如下配置，如果有则表明网站肯定使用了DWR，再执行步骤3   <servlet>  <servlet-name>dwr-invoker</servlet-name>  <servlet-class>org.directwebremoting.servlet.DwrServlet</servlet-class>  <init-param>  <param-name>debug</param-name>  <param-value>true</param-value>  </init-param>  <load-on-startup>11</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>dwr-invoker</servlet-name>  <url-pattern>/dwr/\*</url-pattern>  </servlet-mapping>  如果以上配置信息中，debug参数值是否为true，则表明开启了DWR的调试功能，攻击、测试起来更加容易，为了方便测试，测试时可以把该参数设置为true，但版本发布时，一定要把该参数设置为false。  *备注：如果是黑盒测试，则直接执行步骤3*   1. 访问http://www.example.com[/YOUR-WEBAPP]/dwr/，对开放的所有DWR列表逐个进行测试，看看其是否有安全保护措施（如：认证、鉴权）。   备注：如果网站关闭了DWR的调试功能（也就是web.xml文件中dwr-invoker的debug参数的值为false），那么无法访问http://www.example.com[/YOUR-WEBAPP]/dwr/；但是DWR还是可以调用到，方法参考步骤4   1. 使用DWRDigger，调用DWR方法。 2. 双击DWRDigger图标，启动DWRDigger工具。 3. 点击“导入”按钮，选择要测试DWR对应的dwr.xml文件。（可以通过产品安装包，或者直接登录web服务器，获取dwr.xml文件。） 4. 配置Web应用对应的URL（形式：http://IP:端口/[webapp]，或者http://域名/[webapp]），例如：http://10.70.170.175:8080/tellin。 5. 点击“DWR类”框中的类，自动获取DWR方法列表 6. 点击“DWR方法”框中的方法，自动获取DWR方法对应的参数列表 7. 在“输入参数和值”框中填写输入参数值，然后点击“调用”按钮 8. 在“返回结果”框中查看调用的返回结果。     备注：   1. 如果了网站开启了DWR的Debug功能（通过IE可以访问到其dwr/index.html页面），则可以在Web应用URL中输入dwr/index.html对应的URL（例如：http://10.70.170.175:8080/dwr/index.html）。这时候，就不需要导入dwr.xml文件了。 2. 如果需要测试Internet上的DWR类，在确保IE连接Internet的代理配置正确的情况下，点击选择“使用IE代理”。 3. 如果测试的DWR类有会话管理机制，则通过代理工具获取会话标识及值，并在Cookie中配置会话标识及值，例如：JSESSIONID=0599F4FC3E64C7C673D27F296AA1450D 4. 分析这些DWR提供的功能是否有安全保护措施（如：认证、鉴权）。 |
| 预期结果 | 网站没有使用DWR；或者使用了，但不存在安全漏洞。 |
| 备注 | DWR(Direct Web Remoting)是一个WEB远程调用框架，利用这个框架可以让AJAX开发变得很简单，但同时也会因为开启Debug功能、缺乏认证、鉴权机制而带来严重的安全隐患。dwr.jar是一个开源的类库，可以帮助开发人员开发包含AJAX技术的网站。它可以允许在浏览器里的代码（javascript）使用运行在WEB服务 器上的JAVA函数，就像它就在浏览器里一样。它包含两个主要的部分：允许JavaScript从WEB服务器上一个遵循了AJAX原则的Servlet(小应用程序)中获取数据。另外一方面一个 JavaScript库可以帮助网站开发人员轻松地利用获取的数据来动态改变网页的内容。 |
| 测试结果 |  |

## HTML5安全测试

HTML5作为下一代的开发规范，为开发者提供了许多新的标签（详见附件5.2）、功能（跨域资源共享、客户端存储、WebWorker、WebRTC、WebSocket等等），这些新的标签及功能增加了攻击面，存在被攻击的风险。

### 跨域资源共享测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_HTML5\_01 |
| 测试用例名称 | 跨域资源共享测试（CORS） |
| 测试目的 | 由于浏览器的同源策略给AJAX开发带来许多不便，虽然HTML5之前有许多解决跨域问题的方法，但不同的浏览器兼容性较差。自HTML5开始，允许AJAX在进行跨域操作，服务端只需要在返回的响应头中配置Access-Control-Allow-Origin属性，即可在允许的范围内进行跨域请求，但该属性配置不当会引起安全问题。 |
| 用例级别 | 1 |
| 测试条件 | 1. 存在两个Web应用A和B并且运行正常 2. A使用AJAX进行跨域请求B的资源，如请求URL为http://b.com/xxx 3. B在返回的响应头中加入Access-Control-Allow-Origin属性，如通过response.setHeader("Access-Control-Allow-Origin","\*"); |
| 执行步骤 | 1. 配置burpsuite为浏览器代理，如图：        1. 点击“Repeater”标签，并修改GET请求的URL为上面的跨域URL，，点击GO，如图：   无标题5.png  3、查看返回的响应头中有没有Access-Control-Allow-Origin属性。 |
| 预期结果 | 返回的响应头中的Access-Control-Allow-Origin不应设置为所有域“\*”，而是按需配置的域 |
| 备注 |  |
| 测试结果 |  |

### Web客户端存储安全测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ HTML5\_02 |
| 测试用例名称 | Web客户端存储安全测试 |
| 测试目的 | Web客户端存储包括WebStorage和WebSQL，在HTML5的WebStorage出现之前，客户端存储即本地存储一直都是使用cookie的方式，但cookie作为本地存储有以下缺点：数据只能最大存4k、在HTTP请求中cookie是明文传输、每次请求响应都附带cookie增加了不必要的流量损失。WebStorage是HTML新增的本地存储解决方案之一，可存达5M的数据，包括localStorage和sessionStorage两种类型API。另一种是WebSQL，实际上就是在浏览器中嵌入SQLlite数据库，在客户端存储、查询数据。 |
| 用例级别 | 1 |
| 测试条件 | 1、Web业务运行正常并且使用HTML5的WebStorage或WebSQL |
| 执行步骤 | 1、打开Chrome浏览器并在地址栏中输入待测网址，空白处点击右键选择审查元素，再点击“Resources”如图：  无标题.png  2、检查数据是否放在合适的作用域里，包括用户sessionID不要用LocalStorage存储，如有必要则放在sessionStorage里。用户数据不要储存在全局变量里，而应该放在临时变量或者局部变量里。另外，无论是LocalStorage或sessionStorage都不要存储敏感信息等重要数据。  无标题2.png  3、对于WebSQL，除了检查是否存储敏感信息之外，还需要检查是否存在SQL注入，具体测试可参考“SQL注入测试”章节。另外，由于WebSQL可通过JavaScript读取所有数据，因此需要检查是否存在XSS，具体测试可参考“跨站脚本测试”章节。  无标题3.png |
| 预期结果 | LocalStorage、sessionStorage或WebSQL均未发现敏感信息等重要数据，LocalStorage无sessionID，WebSQL无SQL注入。 |
| 备注 | 1、除利用Chrome浏览器查看WebStorage和WebSQL之外，可在页面引入html5\_attack\_framerwork.js脚本并执行方法得到LocalStorage、sessionStorage或WebSQL的数据，具体请参考https://appsec-labs.com/html5/HTML5-attack-framework |
| 测试结果 |  |

### WebWorker安全测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ HTML5\_03 |
| 测试用例名称 | WebWorker安全测试 |
| 测试目的 | 由于Javascript是单线程执行的，在执行过程中浏览器不能执行其它Javascript脚本，使用WebWorker可以将计算过程放入一个新线程里去执行，样可以同时执行多个JS任务而不会阻塞浏览器。默认情况下WebWorker无法访问DOM，只能通过postMessageAPI和主线程通信。postMessage在HTML5中被引入，用来解决跨域或者跨线程数据交互的问题。但是如果messaging可以接收任何来源的信息，此页面有可能会被攻击；另外postMessage不通过服务器，如果不经过验证和过滤，可能成为XSS注入点。 |
| 用例级别 | 1 |
| 测试条件 | 1、Web业务运行正常并在Web中使用WebWorker  2、假设存在页面A，并且引用WebWorker，如http://xxx/xxx.js，该WebWorker通过postMessage发送数据到页面A |
| 执行步骤 | 1、检查页面A接收WebWorker数据的代码（即监听事件代码）是否使用InnerHTML之类的代码。    2、检查postMessage发送到A页面的数据是否加以验证或过滤，如以上代码的e.data。 |
| 预期结果 | postMessage发送到主页面的数据应该经过验证或过滤，并且主页面上的监听事件代码不使用InnerHTML之类的API |
| 备注 |  |
| 测试结果 |  |

## FLASH安全配置测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ FLASH\_01 |
| 测试用例名称 | FLASH安全配置测试 |
| 测试目的 | FLASH作为一款浏览器的第三方插件，是对浏览器功能的延伸，已经是web必不可少的元素。但是这种延伸必然带来不安全的因素, 浏览器所贯彻的域安全策略被flash所打破，在某些情况下你甚至可以认为它等同于javascript，甚至更为危险，因此，规范flash的用法迫在眉睫。 |
| 用例级别 | 1 |
| 测试条件 | 1、Web业务运行正常并在Web中使用FLASH  2、网站根目录存在跨域策略文件crossdomain.xml |
| 执行步骤 | 1、配置文件方面在浏览器地址栏中输入http://www.example.com/crossdomain.xml，检查以下标签的配置：  （1）allow-access-from标签的domain属性检测：domain属性应根据最小化原则按需设置，仅允许可信任的来源跨域请求本域内容。禁止将该属性值设置为“\*”。  （2）allow-http-request-headers-from标签的domain属性检测：domain属性应根据最小化原则按需设置，仅允许可信任的来源向本域跨域发送内容。禁止将该属性值设置为“\*”。  （3） site-control标签的permitted-cross-domain-policies属性检测：根据业务的实际需求及可行性，对该属性做相应设置。禁止将该属性值设置为“all”。  2、前端页面检查嵌入flash文件的安全配置：  （1）禁止设置flash的allowscriptaccess为always，必须设置为never，如果设置为SameDomain，需要客户可以上传的flash文件要在单独的一个域下。  （2）设置allowNetworking选项为none。  （3）设置allowfullscreen选项为false。 |
| 预期结果 | 以下安全属性值不应设为“\*”、“all”、“always”等通用域 |
| 备注 |  |
| 测试结果 |  |

## 其他

### 日志审计

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_Log\_01 |
| 测试用例名称 | 日志审计 |
| 测试目的 | 检查安全事件及操作事件有没有相应的日志记录 |
| 用例级别 | 2 |
| 测试条件 | 1. Web业务运行正常 2. 已知日志查询URL，假设为http://www.example.com/page.xxx |
| 执行步骤 | 1. 打开日志查询页面 2. 检查安全事件及操作事件有没有相应的日志记录 3. 检查日志是否包含以下内容：事件发生的时间、客户端IP、成功或失败标识、事件的详细描述。 |
| 预期结果 | 安全事件及重要的操作日志有相应的日志记录，日志至少包含：事件发生的时间、客户端IP、成功或失败标识、事件的详细描述。 |
| 备注 | 安全事件包括登录、注销、添加、删除、修改用户、授权、取消权限、鉴权、修改用户口令等；  操作事件包括对业务系统配置参数的修改，对重要业务数据的创建、删除、修改、查询等。 |
| 测试结果 |  |

### class文件反编译测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_REVERSE\_02 |
| 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 | 4 |
| 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：加解密或涉及重要知识产权的业务核心代码） 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 |  |

### WEB部署与管理测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ DEPLOY\_03 |
| 测试用例名称 | WEB部署与管理测试 |
| 测试目的 | 除了WEB本身代码会产生安全问题之外，在生产环境中WEB系统部署不当也会引起高风险的安全问题。 |
| 用例级别 | 1 |
| 测试条件 | 1、Web业务运行正常并对Internet开放 |
| 执行步骤 | 1、查看Web部署的网络拓扑图或向管理员询问，Web服务器是否置于DMZ区，Web服务器与Internet之间、Web服务器与内网之间应当有防火墙隔离，并设置合理的策略。  2、查看Web服务器是否部署在专用的服务器上，即禁止与数据库服务器或其他核心应用部署在同一服务器上。  3、查看Web系统的根目录是否部署在非操作系统盘，如windows的C盘，单独为Web系统在一个非操作系统盘创建目录能够防止攻击者利用目录遍历攻击访问系统工具和可执行文件。  4、查看用户访问Web系统的帐号是否为特权用户，如linux下的root权限、windows下的administrator权限等等，禁止使用特殊用户。  5、查看Web系统访问后台数据库的帐号是否为为特权用户，如SQLServer的sa、Oracle的system、MySQL的root等，禁止使用特殊用户。  6、对于安全性较高的Web系统，查看是否全程采用HTTPS以保证会话的机密性和完整性。  7、查看是否将不适合公众索引或者不是所有用户可见的目录或文件列在robots.txt文件中，防止暴露Web站点的目录结构。  8、查看WEB容器是否在部署前进行安全配置，包括删除不必要的功能、禁用不需要的端口等等，如Tomcat在生产环境中应删除默认安装自带的demo等等。  9、查看Web容器及应用的文件和目录权限是否进行最小化设置，如目录权限设置为仅允许属主读、写、执行，bin目录下的可执行文件权限设置为仅允许属主读、执行，其他文件权限设置为仅允许属主读），防止其他用户获取或篡改Web文件。 |
| 预期结果 | WEB系统部署在生产环境前已做好以上安全配置。 |
| 备注 |  |
| 参考资料 |  |

### Struts2框架测试

|  |  |
| --- | --- |
| 编号 | SEC\_Web\_ FRAMEWORK\_04 |
| 测试用例名称 | Struts2框架测试 |
| 测试目的 | Struts2作为开发JavaEE的首选MVC框架为开发人员提供了便利，但这几年不断爆出的Struts漏洞成为WEB系统的恶梦，尤其是远程执行代码的漏洞再加上网上流传的自动化工具，使得无需更多专业知识的恶意人员通过工具即可入侵服务器。 |
| 用例级别 | 1 |
| 测试条件 | 1、Web业务运行正常并使用存在漏洞的Struts2作为开发框架  2、假设系统存在URL：http://www.example.com/xxx.action |
| 执行步骤 | 1、由于Struts2漏洞高危级别形影最大的主要还是远程执行代码漏洞，以下以该漏洞为例：  2、打开Struts2利用工具，输入<http://www.example.com/xxx.action>，点击“获取信息”，如果出现操作系统及JAVA环境的相关信息，证明存在漏洞，另外，Struts2远程执行代码漏洞也分为几种，可根据Struts2的版本相应选择右边的“使用漏洞”选项，如图：  无标题7.png  4、以上仅演示Struts2的远程执行漏洞检测，但要完全避免Struts2的漏洞就必须保持Struts2的最新版本，下面与Tomcat为例，演示如何测试查看Struts2是否为最新版本，进入Tomcat主目录\Webapps\你的应用\WEB-INF\lib，查看struts2-core-xxx.jar的版本，如图：    5、打开<http://struts.apache.org/> 查看struts的最新版本对比。 |
| 预期结果 | * 1. 官网上的最新版本与项目中使用的版本一致。   2. 以上Struts2利用工具不能利用 |
| 备注 |  |
| 参考资料 |  |

# AppScan测试覆盖项说明

自动化测试工具AppScan满足如下测试项：

* 参数操作
  + 跨站脚本
  + SQL注入
  + 代码执行
  + 文件包含
  + 脚本源代码检查
  + CRLF注入
  + Cross Frame Scripting (XFS)
  + PHP代码注入
  + XPath注入
  + 全路径泄漏
  + LDAP注入
  + Cookie 操作
* 文件检查
  + 检查备份的文件及目录
  + URI中的跨站脚本
  + 检查脚本错误
* 目录检查
  + 查找常见的文件
  + 发现敏感的文件和目录
  + 发现存在弱权限控制的目录
  + 路径和SESSIONID中的跨站脚本
* 文本搜索
  + 目录列表
  + 源代码泄漏
  + 注释信息搜索
  + 常见文件检查
  + Email地址收集
  + 本地路径泄漏
  + 错误信息
* Session Fixation（会话固定）
* Google Hacking

# 附件

## 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 |
| Httprint | Web服务器类型和版本的探测工具 |
| 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 |
| Burpsuite | Web攻击套件，包括代理、扫描、攻击等功能 |

## HTML5新增标签

### 结构标签

<article>  标记定义一篇文章  
 <header>  标记定义一个页面或一个区域的头部  
 <nav>   标记定义导航链接  
 <section>  标记定义一个区域  
 <aside>  标记定义页面内容部分的侧边栏  
 <hgroup>  标记定义文件中一个区块的相关信息  
 <figure>  标记定义一组媒体内容以及它们的标题   
 <figcaption>  标签定义 figure 元素的标题。  
 <footer>  标记定义一个页面或一个区域的底部  
 <dialog>  标记定义一个对话框(会话框)类似微信

### 多媒体标签

<video> 标记定义一个视频  
 <audio> 标记定义音频内容  
 <source> 标记定义媒体资源

<canvas> 标记定义图片

<embed> 标记定义外部的可交互的内容或插件，比如flash

### Web应用标签

<menu> 命令列表  
<menuitem> menu命令列表标签 FF（嵌入系统）  
<command> menu标记定义一个命令按钮

<meter> 状态标签(实时状态显示:气压、气温)C、O  
<progress> 状态标签 (任务过程:安装、加载) C、F、O

<datalist> 为input标记定义一个下拉列表,配合option F、O  
<details> 标记定义一个元素的详细内容 ，配合dt、dd   C

### 其他标签

<ruby> 标记定义 注释或音标  
<rp> 告诉那些不支持 Ruby元素的浏览器如何去显示  
 <rt> 标记定义对ruby的注释内容文本  
<keygen> 标记定义表单里一个生成的键值(加密信息传送)O、F  
<mark> 标记定义有标记的文本 (黄色选中状态)  
<output> 标记定义一些输出类型,计算表单结果配合oninput事件  
<time> 标记定义一个日期/时间 目前所有主流浏览器都不支持