蚌埠和能信息科技有限公司

常见Web应用程序攻击和漏洞测试

V0.1 日期: 2020/4/13/ 11:45

修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 作者 | 日期 | 修改说明 |
|  |  |  |  |

目录

1 背景介绍 1

2 常见的安全漏洞 1

2.1 SQL注入 1

2.2 XSS跨站点脚本攻击 2

2.3 CSRF跨站伪造请求攻击 2

2.4 文件上传漏洞 3

2.5 URL跳转漏洞 3

3 Web安全开发自检清单 3

4 常见Web安全工具 9

5 总结 9

# 背景介绍

Web应用程序通常是指通过HTTP / HTTPS协议共同提供服务的B / S体系结构。 随着Internet的发展，Web应用程序已集成到我们日常生活的各个方面。 在当前的Web应用程序中，大多数应用程序不是静态Web浏览，而是涉及服务器的动态处理。 如果开发人员的安全意识不强，将导致无休止的Web应用程序安全问题。

我们通常指的Web应用程序攻击是指攻击者通过浏览器或其他攻击工具向URL或其他输入区域（例如表单）中的Web服务器发送特殊请求，以发现Web应用程序的存在。 反过来，通过操作和控制网站来达到入侵者的目的。

# 常见的安全漏洞

## SQL注入

SQL注入（SQL Injection）是最常见的漏洞，具有多种影响。 攻击者将SQL命令插入Web表单以提交或输入域名或页面请求的查询字符串，并最终诱使服务器执行恶意SQL命令，从而入侵数据库以执行任意查询。

SQL注入可能造成的危害是：篡改了网页和数据，窃取了核心数据，攻击了数据库所在的服务器，并使之成为a主机。

例如，某些网站不使用预编译的SQL，并且用户在界面上输入的某些字段将添加到SQL。 这些字段可能包含一些恶意SQL命令。 例如：password =“ 1'OR'1'='1”; 即使您不知道用户密码，也可以正常登录。

* 测试方法：

在需要查询的页面上，输入简单的SQL语句，例如正确的查询条件和1 = 1，然后检查响应结果。如果结果与正确的查询条件相符，则表明该应用程序尚未筛选用户输入，并且可以初步判断它存在。 SQL注入漏洞

## XSS跨站点脚本攻击

SS（跨站点脚本）类似于SQL注入，XSS通过网页插入恶意脚本。使用的主要技术是前端HTML和JavaScript脚本。当用户浏览网页时，将实施一种控制用户浏览器行为的攻击方法。

成功的XSS可以获取用户的cookie，并使用该cookie窃取用户在网站上的操作权限。它还可以获取用户的联系人列表，并使用攻击者的身份将大量垃圾邮件发送到特定的目标组。 ，还有很多。

XSS分为三类：存储（持久XSS），反射（非持久XSS）和DOM。

* 测试方法：

在数据输入界面上，输入：保存成功后，弹出对话框，提示存在XSS漏洞。

或更改url请求中的参数。如果页面上弹出对话框，则表明存在XSS漏洞。

## CSRF跨站伪造请求攻击

CSRF(Cross Site Request Forgery)，利用已登录的用户身份，以用户的名义发送恶意请求，完成非法操作。

例如，如果用户浏览并信任具有CSRF漏洞的网站A，则浏览器会生成相应的cookie，并且用户访问危险的网站B而不退出网站。

危险网站B要求访问网站A并提出要求。 浏览器使用用户的cookie信息访问网站A。 由于网站A不知道是用户自身发出的请求还是危险网站B发出的请求，因此将处理危险网站B的请求，从而完成了用户操作目的的模拟。 这是CSRF攻击的基本思路。

* 测试方法：

同个浏览器打开两个页面，一个页面权限失效后，另一个页面是否可操作成功，如果仍然能操作成功即存在风险。2.使用工具发送请求，在http请求头中不加入referer字段，检验返回消息的应答，应该重新定位到错误界面或者登录界面。

## 文件上传漏洞

文件上传攻击是指攻击者将可执行文件上传到服务器并执行该文件时。

这种攻击方法是最直接，最有效的。 上载的文件可以是病毒，特洛伊木马，恶意脚本或Webshell。

Webshell是Web文件（例如asp，php，jsp或cgi）形式的命令执行环境。 也可以说是Web后门。 攻击者阻止或在受影响的系统上插入Web Shell之后，他可以轻松地通过Web Shell访问系统以控制Web服务器。

* 测试方法：

严格检查上传文件的类型和大小，禁止上传带有恶意代码的文件。

检查相关目录的执行权限。 您可以通过浏览器访问Web服务器上的所有目录，并检查是否返回了目录结构。 如果显示目录结构，则可能存在安全问题。

## URL跳转漏洞

URL跳转漏洞，即未经验证的重定向漏洞，是指Web程序直接跳转到参数中的URL，或者在页面中引入了任意开发者的URL，将程序引导到不安全的第三方区域，从而导致安全问题。

* 测试方法：

1.使用数据包捕获工具捕获请求。

2.抓住302 URL，修改目标地址，然后查看它是否可以跳转。

ps：但是现在很多跳转都添加了引荐来源验证，这导致攻击者无法跳转。

# [Web安全开发自检清单](https://www.cnblogs.com/tangqingsong/p/9995842.html)

| **检查类型** | **说明** | **检查项** |
| --- | --- | --- |
| 输入验证 | 概述 | 任何来自客户端的数据,如URL和参数、HTTP头部、 Javascript戓其他嵌入代码提交的信息,都属于不可信数据。在应用外部边界或内部每个组件或功能边界,都将其当做潜在的恶意输入来校验 |
| 白名单 | 不可信数据可以设定白名单校验的,应接受所有和白名单匹配的数据,并阻止其他数据 |
| 黑名单 | 不可信数据中包含不良输入字符时,如空字节(%00)、换行符(%0d,%0a,\r, \n)、路径字符(../ 或 ..)等,建议直接阻止该数据,若需要接受该数据,则应做不同方式的净化处理 |
| 规范化 | 不可信数据的净化和校验前翯进行规范化,如将目录遍历(./或)等相对路径转化成绝对路径URL解码等。 |
| 净化 | 不可信数据需实施各种净化处理时,应彻底删除恶意字符,只留下已知安全的字符,或者在处理前对它们进行适当编码或"转义",如数据输出到应用页面时对其进行HTML编码可防止脚本攻击 |
| 合法性校验 | 不可信数据的合法性校验包括:数据类型如字符.数字、日期等特征;数据范國;数据长度等 |
| 防范SQL注入 | 不可信数据进入后端数据库操作前,建议使用正角的参数化查询来处理,避免出现SQL注入 |
| 文件校验 | 不可信数据为解压缩的文件时,如果文件位于服务目录外或文件大小超过限制,应拒绝处理 |
| 访问控制 | 不可信数据通过上述校验后,还应确认所提交的内容是否与用户的身份匹配,避免越权访问 |
| 输出验证 | 概述 | 考虑目标编译器的安全性，对所有输出字符进行正确编码 |
| 编码场景 | 不可信数据输出到前后端页面时,根据输出场景对其进行相关编码,如HTML实体编码、UR编码 |
| 净化场景 | 针对操作系统命令、SQL和LDAP查询,净化所有输出的敏感信息,如银行卡、手机号、系统信息等 |
| 身份验证 | 概述 | 所有对非公开的网页和资源的访问,必须在后端服务上执行标准的、通用的身份验证过程 |
| 提交凭证 | 用户凭据必须经过加密且以POST方式提交,建议用HTPS协议来加密通道、认证服务端 |
| 错误提示 | 安全地处理失败的身份校验,如使用"用户名或密码错误"来提示失败,防止泄露过多信息 |
| 异常处理 | 登录入口应具有防止暴力或撞库猜解(利用已泄露的密码字典进行批量登录尝试)的措施,超过1次验证失败自动启用图灵测试,超过多次验证失败自动启用账户锁定机制限制其访问 |
| 二次验证 | 在执行关键操作(如账户密码修改、资料更新、交易支付等)时,先启动图灵测试,再对用户身份进行二次验证。交易支付过程还应该形成完整的证据链,待交易数据应经过发起方数字签名 |
| 多因子验证 | 高度敏感或核心的业务系统,建议使用多因子身份验证机制,如短信验证码、软硬件 Token等。 |
| 短信验证 | 验证码生成 | 复杂度至少6位数字或字母,一次一用,建议有效期不超过180秒。 |
| 验证码限制 | 前后端设置用户获取频率为60秒一次,建议每个用户每天获取的短信最多10条 |
| 安全提示 | 增加安全提示:至少含本次操作的功能、验证码发送编号、是否是个人自己操作的风险等信息。 |
| 凭证校验 | 禁止在响应中返回验证码,服务器端同时校验密码、短信验证码等凭证信息,防止出现多阶段认证绕过的漏洞。 |
| 图灵测试 | 验证码生成 | 复杂度至少4位数字或字母,或者采用拼图等验证方式,一次一用,建议有效期不超过180秒 |
| 验证码使用 | 建议从用户体验和安全角度出发,可设计为当用户输错1次密码后自动弹出验证码输入框验证 |
| 验证码校验 | 禁止在响应中返回验证码,验证码校验应在服务端进行 |
| 密码管理 | 密码设置 | 密码设置时,应该满足8位及以上长度,含大小写字母、数字及特殊字符等的要求。用户密码设置必须经过后端验,不允许设置不满定复杂度要求的感密码。 |
| 密码存储 | 用户密码存储时,应采用哈希算法(如SHA1)计算用户密码和唯一随机盐值(Salt)的摘要值保存其摘要和Sat值,建议分开存储这两个值 |
| 密码修改 | 用户修改密码时,修改操作需要通过手机号或者邮箱地均进行一次身份验证。密码变更时,应短信或者邮件通知如用户是否是本人操作,告知其安全风险 |
| 密码找回 | 用户密码找回时,后端需要对注册手机号或邮箱进行二次验证,验证码和验证链接应发送至预先注册的地址,并设置有效期以防止暴力破解。密保问题,应当支持尽可能随机的问题提问。在多个验证操作中,要对各验证机制进行排序,以防出现跳过前面验证机制直接到最后步认证的安全风险 |
| 密码使用 | 应用开发中禁止设置万能密码、硬编码明文的密 码、使用数据库管理员账户操作、不同用户公用账 户操作或者将密码输出到日志文件或者控制台. |
| 会话安全 | 防止会话劫持 | 在应用程序进行身份验证时,建议持续使用HTTPS连接,认证站点使用HTTPS协议。如果连接是从防止会话劫持HTTP跳转到HTTPS,需要重新生成会话标识符。禁止在HTTP和HTTPS之间来回转换,这可能会导致会话被劫持 |
| 会话标识符安全 | 设置会话 Cookie时,正确设置" Httponly'属性(禁止程序加5脚本等读取 Cookie信息)" Secure'属性(禁Cookie安全设置止Cookie通过HTTP连接传递到服务器端进行验证);" Domain"属性(跨域访问时可指定的授权访问域名),"Path"属性(授权可访问的目录路径)。 |
| Cookie安全设置 | 会话标识符应放置在HTP或HTPS协议的头信息安全中,禁止以GET参数进行传递、在错误信息和日志中记录会话标识符 |
| 防止CSRF攻击 | 服务器端执行了完整的会话管理机制,保证每个会防止CSRF话请求都执行了合法的身份验证和权限控制,防止攻击发生跨站点请求伪造(CSRF)漏洞。 |
| 会话有效期 | 会话应在平衡风险和功能需求的基础上设置有效期。定期生成一个新的会话标识符并使上一个会话会话有效期标识符失效,这可以缓解那些因原会活标识符被盗而产生的会话劫持风险。 |
| 会话注销 | 注销功能应用于所有受身份验证保护的网页,用户会话注销登出后应立即清理会话相关信息,终止相关的会话连接 |
| 访问控制 | 控制方法 | 将访问控制的逻辑代码与应用程序其他代码分开服务端根据会话标识来进行访问控制管理。 |
| 控制管理 | 限制只有授权的用户才能访问受保护的URL、文件、服务、应用数据、配置、直接对象引用等 |
| 接口管理 | 限制只有授权的外部应用程序或接口才能访问受保护的本地程序或资源等 |
| 权限变更 | 当权限发生变更时,应记录日志,并通知用户是否是本人操作,告知存在的安全风险 |
| SQL注入 | 概述 | 用户的输入进入应用程序的SQL操作前,对输入进行合法性校验。 |
| 参数化处理 | 用参数化查询(PHP用PDO,Java用 PreparedStatement,C#用 Sqlparameter)方法对敏感字符如"进行转义,然后再进行SQL操作。 |
| 最小化授权 | 为每个应用配置最小化数据库操作权限,禁止用管理员权限进行数据库操作,限制操作连接数。 |
| 敏感数据加密 | 敏感信息都采用了加密、哈希或混淆等方式进行保密存储,降低可能漏洞带来的数据泄露风险. |
| 禁止错误回显 | 禁止系统开启 Debug模式或异常时返回包含敏感信息的提示,建议使用自定义的错误信息模板异常信息应存放在日志中用于安全审计 |
| XSS注入 | 输入校验 | 对输入的数据进行过滤和转义,包含但不限于&lt;&gt;"9%0&+\V"等危险特殊字符 |
| 输出编码 | 输入数据输出到不同场景中进行不同形式的编码,如输出到HTML标签中则进行HTML编码输出到URL中则进行URL编码,输出到JS中则行 Script编码,输出到 Stylet中则进行CSs编码 |
| XML注入 | 输入校验 | 在XML文档内部或外部引用数据时,过滤用户提交的参数,如&lt;、&gt;&等特殊字符。禁止加载外部实体,禁止报错 |
| 输出编码 | 建议对XML元素属性或者内容进行输出转义 |
| 敏感信息 | 敏感信息传输 | 敏感信息传输时,禁止在GET请求参数中包含敏感信息,如用户名、密码、卡号等。建议为所有敏感信息采用TSL加密传输。 |
| 客户端保存 | 客户端保存敏感信息时,禁止其表单中的自动填充功能、以明文形式保存敏感信息 |
| 服务端保存 | 服务端保存敏感信息时,禁止在程序中硬编码敏感信息,明文存储用户密码、身份证号、银行卡号、持卡人姓名等敏感信息,临时写入内存或文件中的敏感数据,应及时清除和释放 |
| 敏感信息维护 | 敏感信息维护时,禁止将源码或SQL库上传到开源平台或社区,如 Github、开源中国等。 |
| 敏感信息展示 | 敏感信息展示时,如果是展示在web页面上,应在后端服务器上进行敏感字段的脱敏处理。 |
| CSRF跨站请求伪造 | Token使用 | 在重要操作的表单中增加会话生成的 Token字段次一用,提交后在服务端校验该字段 |
| 二次验证 | 在关键表单提交时,要求用户进行二次身份验证如密码、图片验证码、短信验证码等 |
| Referer验证 | 检验用户请求中 Referer:字段是否存在跨域提交的情况 |
| 文件上传安全 | 身份校验 | 进行文件上传时,在服务端对用户的身份进行合法性校验 |
| 合法性校验 | 进行文件上传时,在服务端对文件属性进行合法性校验,白名单形式检查文档类型(如文件的后缓名、文件头信息校验等)和大小(图片校验长、宽和像素等)。 |
| 存储环境设置 | 进行文件保存时,保存在与应用环境独立的文档服务器中(配置独立域名),保存的目录权限应设置为不可执行 |
| 隐藏文件路径 | 进行文件保存时,成功上传的文件需要进行随机化重命名,禁止给客户端返回保存的路径信息。 |
| 文件访问设置 | 进行文件下载时,应以二进制形式下载,建议不提供直接访问(防止木马文件直接执行) |
| 接口安全 | 网络限制 | 调用方网络限制,比如通过防火墙、主机host和Nginx deny等技术措施进行校验。 |
| 身份认证 | 调用方身份认证,比如key、 secret、证书等技术措施进行校验,禁止共享凭证 |
| 完整性校验 | 调用的数据安全,对全部参数使用SHA1等摘要运算进行数字签名,识别数据被篡改 |
| 合法性校验 | 调用的参数检查,如参数是否完整,时间戳和Token是否有效,调用权限是否合法等 |
| 可用性要求 | 调用的服务要求,调用满足等幂性即保持数据一致性,对调用频率和有效期进行限制 |
| 异常处理 | 调用的异常处理,调用行为实时检测,发现异常及时阻拦 |
| I/O操作 | 共享环境文件安全 | 在多用户系统中创建文件时应指定合适的访问许可,以防止未授权的文件访问,共享目录中文件的读/写/可执行权限应该使用白名单机制,实现最小化授权。 |
| 数据访问检查 | 防止封装好的数据对象被未授权使用,设置合理的据缓存区大小以防止耗尽系统资源, |
| 应用文件处理 | 应用程序运行过程中创建的文件,需设置问权限(读、写、可执行),临时文件使及时删除 |
| 运行环境 | 最小化开放端口 | 关闭操作系统不需要的端口和服务 |
| 后台服务管理 | 后台(如数据缓存和存储、监控、业务管理等)务限内部网络访问,开放在公网的必须设置身份验证和访问控制。 |
| 环境配置 | 使用安全稳定的操作系统版本、Web股务器软件各种应用框架、数据库组件等 |
| 敏感代码处理 | 将客户端敏感代码(如软件包签名、用户名密码校验等)都放在o等软件包中防止篡改。 |
| 关闭调试通道 | 生产代码不包含任何调试代码或接口 |
| 通信安全 | 配置网站的HTTPS证书或其它加密传输措施。 |
| 异常处理 | 容错机制 | 在应用实现时应包含完整的功能异常捕获机制如try-catch块,典型位置:文件、网络、数据库、命令操作等。一旦出现异常,应该在日志中完整记录异常的发生时间、代码位置、报错详情、触发错误的可能用户等,重要系统的严重异常应该有报警的机制,及时通知系统运营者及时排查并修复题 |
| 自定义错误信息 | 在生产环境下,应用程序不应在其响应中返回任何系统生成的消息或其他调试信息,配置应用服务器使其以自定义的方式处理无法处理的应用程序错误,返回自定义错误信息 |
| 隐藏用户信息 | 禁止在系统异常时泄露用户的隐私信息,典型的有:身份信息、个人住址、电话号码、银行账号、通讯记录、定位信息等 |
| 隐藏系统信息 | 禁止在系统异常时泄露系统的敏感信息(用户账户和密码、系统开发密钥、系统源代码、应用架构、系统账户和密码、网络拓扑等)。 |
| 异常状态恢复 | 方法发生异常时要恢复到之前的对象状态,如业务操作失败时的回滚操作等,对象修改失败时要恢复对象原来的状态,维持对象状态的一致性 |
| 日志规范 | 记录原则 | 确保日志记录包含了重要的应用事件,但禁止保存敏感信息,如会话标识,账户密码、证件等 |
| 事件类型 | 记录所有的身份验证、访问操作、数据变更、关键操作、管理功能、登出记录等事件。 |
| 事件要求 | 日志一般会记录每个事件的发生时间、发出请求的IP地址和用户账户(如果已通过验证)。 |
| 日志保护 | 日志受到严格保护,避免未授权的读取或写入访问。 |

# 常见Web安全工具

1. 漏洞扫描工具 ： AppScan
2. 端口扫描工具：namp、nessus
3. 注入工具：HDSI

其他工具参考：Paros Proxy、WebScarab、Sandcat、NBSI、Domain、Nessus、NMAP、X-Scan。

# 总结

随着对网络安全性的日益重视，Web安全性测试在测试过程中的重要性日益突出。此Web应用会保存学生考卷和学生信息等保密敏感数据，并且会接入校园网，存在一定的概率会被攻击以尝试获取网站敏感数据，所以需要应用系统拥有一定的发防范和漏洞处理，能够抵御常见一些常见的Web安全漏洞。