



# 网络安全

## 第七章 WEB应用漏洞攻防

黄 瑋



基于Web应用程序服务模型

# WEB应用程序漏洞原理



# Web应用程序服务模型





# 输入有关的安全问题





# 一切罪恶都是源于恶意输入数据

- 不要相信任何来自客户端的提交数据
  - 客户端的任何数据校验都是纸老虎
    - 客户端的数据校验机制防君子不防黑客
- 数据和指令/代码必须严格区分
  - 缓冲区溢出时的机器指令运行在可执行堆栈上
  - SQL注入时执行任意SQL语句
  - XSS时执行任何客户端脚本代码 (JS/Flash AS)
  - 文件上传时上传服务端脚本代码在服务器端执行任意代码



## 1. 未验证的用户输入-示例

- 设计输入
  - ../ViewServlet?url=http://backendhost/images/bg.gif
- 恶意输入
  - ../ViewServlet?url=http://weblogic/console
  - ../ViewServlet?url=file:///etc/passwd
  - ../ViewServlet?url=../../../../etc/passwd
- 通过这个简单的应用程序可以间接实现文件枚举和后台程序扫描



## 1. 未验证的用户输入-描述(1/2)

- 攻击者可以篡改HTTP request的任何一个部分
  - url
  - 请求字符串
  - HTTP头
  - Cookies
  - 表单域
  - 隐藏域



## 1. 未验证的用户输入-描述(2/2)

- 常见的输入篡改攻击包括：

- 强制浏览
- 命令注入
- 跨站点脚本攻击
- 缓冲区溢出攻击
- 格式化字符串攻击
- SQL注入
- Cookie毒化
- 隐藏域控制



# 输入篡改攻击的成因

- 可能的原因
  - 只在客户端进行输入验证
  - 过滤时未进行规范化
    - 过滤后引入新漏洞
- 导致其他的漏洞



# 1. 未验证的用户输入-工具演示

The screenshot displays two windows from the Live HTTP Headers and Live HTTP Replay tools, illustrating the capture and replay of HTTP requests.

**Live HTTP Headers (Left Window):**

- HTTP Headers:**  
http://localhost/WebGoat/images/menu\_images/1x1\_open.gif
- Request:**  
GET /WebGoat/images/menu\_images/1x1\_open.gif HTTP/1.1
- Headers:**  
Host: localhost  
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.8.1.4) Gecko/20070515 Firefox/2.0.0.4  
Accept: text/xml, application/xml, application/xhtml+xml, text/html;q=0.9, text/plain;q=0.8, image/png,\*/\*;q=0.5  
Accept-Language: zh-cn, zh;q=0.5  
Accept-Encoding: gzip, deflate  
Accept-Charset: gb2312, utf-8;q=0.7, \*;q=0.7  
Keep-Alive: 300  
Connection: keep-alive  
Referer: http://localhost/WebGoat/attack?menu=710  
Cookie: JSESSIONID=8AB92307A9A3013E11D859AA31C28DA9  
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
- HTTP/1.x:**  
HTTP/1.0 404 /WebGoat/images/menu\_images/1x1\_open.gif  
Server: Apache-Coyote/1.1  
Pragma: No-cache  
Cache-Control: no-cache
- Buttons:** Save All..., Replay..., Capture

**Live HTTP Replay (Right Window):**

- HTTP Headers:**  
GET http://localhost/WebGoat/attack?Screen=106&menu=110 HTTP/1.1
- Headers:**  
Host: localhost  
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.8.1.4) Gecko/20070515 Firefox/2.0.0.4  
Accept: text/xml, application/xml, application/xhtml+xml, text/html;q=0.9, text/plain;q=0.8, image/png,\*/\*;q=0.5  
Accept-Language: zh-cn, zh;q=0.5  
Accept-Encoding: gzip, deflate  
Accept-Charset: gb2312, utf-8;q=0.7, \*;q=0.7  
Keep-Alive: 300  
Connection: keep-alive  
Referer: http://localhost/WebGoat/attack?Screen=106&menu=110  
Cookie: JSESSIONID=8AB92307A9A3013E11D859AA31C28DA9  
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
- Buttons:** Send POST Content?, Replay, Close



## 1. 未验证的用户输入-解决方案(1/3)

- 所有的用户输入需要在服务器端进行集中的统一验证
  - 请求参数
  - Cookies
  - HTTP请求头
- 代码复查
- 不要“滥用”隐藏域
  - 存储在Session中或从每次请求中获取参数值



## 1. 未验证的用户输入-解决方案(2/3)

- 请求参数需要严格的验证其类型
  - 数据类型 (string, integer, real, etc…)
  - 最小和最大长度
  - 是否允许null
  - 参数是否是必需的
  - 数字的取值范围
  - 特定模式 (正则表达式)
    - 白名单机制



## 1. 未验证的用户输入-解决方案(3/3)

- 服务器返回给客户端的重要参数、赋值使用 HMAC 进行参数签名
- 千万不要使用 MD5、SHA-XXX 之类的摘要算法对参数进行摘要计算，也不要使用基于“秘密盐值”的 MD5、SHA-XXX 之类的摘要算法对参数进行摘要计算 **小心 Hash 长度扩展攻击**
- 对客户端提交的请求校验关键逻辑代码中的参数，一旦消息完整性签名校验失败，说明客户端尝试篡改请求参数攻击，代码逻辑直接跳过后续业务逻辑代码，给客户端返回统一的错误信息



## 2. 缓冲区溢出-描述(1/2)

- 应用程序的缓冲区中存在过量的输入数据，溢出的数据中包含恶意指令且恶意指令被精确填充到可执行堆/栈（内存）中进而导致恶意代码被执行
- 一般情况下，Web应用程序不存在缓冲区溢出漏洞
- Java Web应用程序不存在缓冲区溢出风险?
  - OutOfMemoryError
  - [CVE-2011-0311](#)
  - [CVE-2009-1099](#)



## 2. 缓冲区溢出-描述(2/2)

- PHP Web应用程序不存在缓冲区溢出风险?
  - [CVE-2011-3268](#)
  - [CVE-2008-5557](#)
  - [CVE-2008-2050](#)
  - [CVE-2007-1399](#)
  - [CVE-2007-1381](#)
  - PHP的缓冲区溢出相关漏洞历史: 1997~2011.8 共计**76**个
- 其他语言编写的Web应用程序呢?
  - 后台应用系统
  - 本地代码



## 2. 缓冲区溢出-解决措施

- 避免使用本地代码
- 避免直接调用本地应用程序
- 及时更新应用运行环境
  - Java虚拟机的安全更新补丁
  - PHP语言的安全更新补丁
- 限制Web应用程序的运行权限
  - 沙盒技术



# 后台相关的安全问题





### 3. 注入缺陷-示例

- 登录模块使用了如下的SQL查询语句
  - "select \* from users where user=' + username + '' and password=' + hashedPassword + '''"
- 很容易被以下的用户输入破解登录功能
  - username:<任何存在的用户名>' or '1'='1
  - password:任何可以通过验证规则的密码
- 拼接后形成的SQL查询语句
  - select \* from users where user='admin' or '1'=1' and password='secret'



### 3. 注入缺陷-描述

- 在任何存在解释器的地方都可能存在
  - 脚本语言，例如Perl, Python和JavaScript
  - Shell脚本语言（执行系统应用程序）
  - 通过系统调用访问操作系统
  - 数据库系统：SQL注入
  - 目录遍历（e.g. ../../etc/passwd）
- 典型缺陷
  - Runtime.exec() / system() / exec()
  - 拼接字符串的SQL
  - 文件输入和输出流操作



### 3. 注入缺陷-操作提示

- SQL注入提示
  - 必须先确定目标数据库类型/版本
  - 手边准备好相应数据库的“手册”
    - 通常cheatsheet足以
- 操作系统命令注入提示
  - 必须先确定目标操作系统类型/版本/应用软件配置信息
  - 手边准备好相应操作系统的“手册”
    - 操作系统命令
    - 敏感文件路径大全



## 创意无所不在的SQL注入（1/3）

- 史上最牛车牌





## 创意无所不在的SQL注入（2/3）

- 一维条形码



' or 1=1 --

- 二维条形码





## 创意无所不在的SQL注入 (3/3)

HI, THIS IS  
YOUR SON'S SCHOOL.  
WE'RE HAVING SOME  
COMPUTER TROUBLE.



OH, DEAR - DID HE  
BREAK SOMETHING?  
IN A WAY - )



DID YOU REALLY  
NAME YOUR SON  
Robert'); DROP  
TABLE Students;-- ?

OH, YES. LITTLE  
BOBBY TABLES,  
WE CALL HIM.

WELL, WE'VE LOST THIS  
YEAR'S STUDENT RECORDS.  
I HOPE YOU'RE HAPPY.



AND I HOPE  
YOU'VE LEARNED  
TO SANITIZE YOUR  
DATABASE INPUTS.



### 3. 注入缺陷-解决方案(1/2)

- 在任何时候避免直接使用外部的解释器，而使用编程语言提供的API库
  - 避免使用Runtime.exec()，通过JavaMail API来发邮件
- 在将数据发送给后台程序时对数据进行编码
  - SQL语句中的单引号 / 注释符
  - LDAP语句中的逗号，括号等



### 3. 注入缺陷-解决方案(2/2)

- 更好的解决办法
  - Java: 使用JDBC驱动的PreparedStatements
  - PHP: 使用预编译SQL语句
- 以受限制的系统权限运行Web应用程序
  - 沙盒技术
- 所有的外部调用的输出、返回代码值和错误代码值都需要检查



# 输出相关的安全问题





## 4.跨站点脚本(XSS)-示例

- Web应用程序直接将请求中的参数“回显”在用户的浏览器中
- URL中请求参数的回显
  - 正常的URL: <http://victim.org/victim.jsp?kind=simple>
  - 被注入JS代码后的URL:  
`http://victim.org/victim.jsp?kind=simple"  
name="kind"><script>alert("test")</script><input type="hidden"`

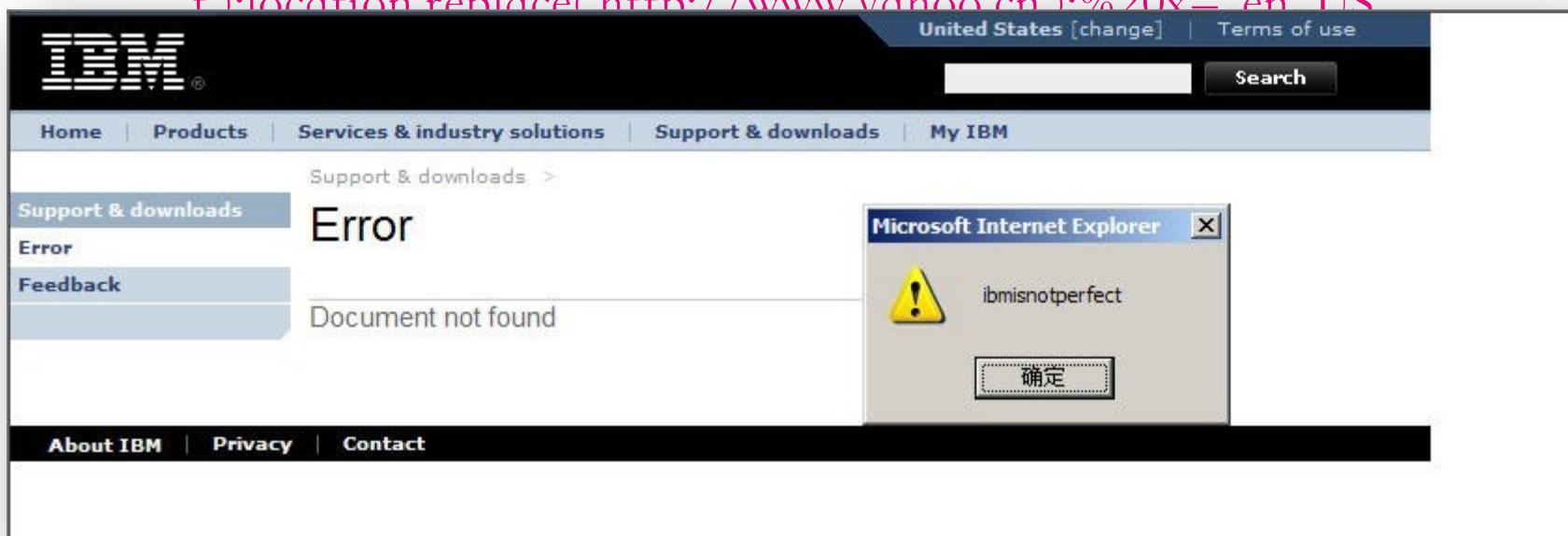


## 4. 著名公司的XSS漏洞

- IBM

— XSS演示代码（2007年6月首次发现，目前IBM已修复该漏洞）

- [http://www-1.ibm.com/support/docview.wss?uid=swg21233077&loc=%3E%20onload=alert\('IbmIsNotPerfect'\)%3E%20location.replace\('http://www.yahoo.cn'\)%20x='en\\_US](http://www-1.ibm.com/support/docview.wss?uid=swg21233077&loc=%3E%20onload=alert('IbmIsNotPerfect')%3E%20location.replace('http://www.yahoo.cn')%20x='en_US)





## 4. 著名公司的XSS漏洞

- TOM

### — XSS演示代码

- [http://search.tom.com/m.php?w=%3Cscript%3Eeval\(String.fromCharCode\(97,108,101,114,116,40,34,84,79,77,25628,32034,28431,27934,65292,50,31186,21518,33258,21160,36339,36716,21040,89,97,104,111,112,25628,32034,34,41,59,115,101,116,84,105,109,101,111,117,116,40,102,117,110,99,116,105,111,110,40,41,123,108,111,99,97,116,105,111,110,46,114,101,112,108,97,99,101,40,34,104,116,116,112,58,47,47,119,119,119,46,121,97,104,111,111,46,99,110,34,41,125,41\)\)%3C/script%3E&mimetype=wma](http://search.tom.com/m.php?w=%3Cscript%3Eeval(String.fromCharCode(97,108,101,114,116,40,34,84,79,77,25628,32034,28431,27934,65292,50,31186,21518,33258,21160,36339,36716,21040,89,97,104,111,112,25628,32034,34,41,59,115,101,116,84,105,109,101,111,117,116,40,102,117,110,99,116,105,111,110,40,41,123,108,111,99,97,116,105,111,110,46,114,101,112,108,97,99,101,40,34,104,116,116,112,58,47,47,119,119,119,46,121,97,104,111,111,46,99,110,34,41,125,41))%3C/script%3E&mimetype=wma)

```
w=%3Cscript%3Eeval(String.fromCharCode(97,108,101,114,116,40,34,84,79,77,25628,32034,28431,27934,65292,50,31186,21518,33258,21160,36339,36716,21040,89,97,104,111,112,25628,32034,34,41,59,115,101,116,84,105,109,101,111,117,116,40,102,117,110,99,116,105,111,110,40,41,123,108,111,99,97,116,105,111,110,46,114,101,112,108,97,99,101,40,34,104,116,116,112,58,47,47,119,119,119,46,121,97,104,111,111,46,99,110,34,41,125,41))%3C/script%3E&mimetype=wma
```





## 4. 著名公司的XSS漏洞

- Sogou

### — XSS演示代码

- [http://www.sogou.com/web?  
query=%3CIFRAME+WIDTH%3D400+HEIGHT%3D400+SRC%3D%22HTTP%3A%2F%2FWWW.YAHOO.CN%22%22%3E%3C%2F](http://www.sogou.com/web?query=%3CIFRAME+WIDTH%3D400+HEIGHT%3D400+SRC%3D%22HTTP%3A%2F%2FWWW.YAHOO.CN%22%22%3E%3C%2F)





## 4. 跨站点脚本(XSS)-描述

- 攻击者将恶意脚本代码发送到终端用户的浏览器
  - Web应用程序的输出直接回显到用户的浏览器而未经过检查
  - 浏览器信任Web应用程序的代码
- 恶意脚本可以
  - 访问cookie, 会话令牌, 或其他通过用户浏览器获得的敏感信息
  - 重写HTML页面



## 4. 跨站点脚本(XSS)-描述

- 2种基本策略
  - 持久化的（可自动触发），例如恶意代码存储到数据库中，通过论坛发帖，访客留言等
  - 反射型（诱骗点击型），例如错误消息，搜索引擎
- 危害示例
  - 会话劫持
  - 钓鱼攻击
  - DDoS攻击
  - 远程信息获取，如端口扫描、用户浏览历史信息枚举



## 4.跨站点脚本(XSS)-操作提示

- 确定好攻击向量的目标浏览器
  - 很多XSS攻击只能在特定浏览器平台上触发
- 手边准备好
  - Javascript语法手册
  - HTML手册
  - CSS手册
  - Flash ActionScript手册
  - 浏览器技术文档
    - 浏览器相关特性速查



# 创意无所不在的XSS

- 二维码





## 4. 跨站点脚本(XSS)-解决方案

- 输入校验
- 编码所有的展现层输出 (HTMLEncode或JSTL的c:out、Struts的<bean:write>标签等)

<	&lt;	>	&gt;
(	&#40;	)	&#41;
#	&#35;	&	&#38;

- 对输入进行长度限制或截短



## 4. 跨站点脚本(XSS)-解决方案

- 如果你的应用程序需要显示用户提交HTML内容，你应该过滤<script>标签，。。。，要确保用户不能提交恶意脚本代码

以上解决方案是远远不够的，可以参考MySpace的Samy蠕虫的攻击代码



## 5.不恰当的错误处理-示例

- 错误的用户名

请输入用户名和密码

User name:  

Password:

不存在该用户，请重试

- 错误的用户口令

请输入用户名和密码

User name:

Password:  

用户口令错误，请重试



## 5.不恰当的错误处理-描述

- 程序的错误消息会暴露程序的一些实现细节
- 示例
  - 堆栈调试信息，数据库错误消息，错误代码
  - JSP编译错误信息包含物理路径信息
  - 不一致的错误消息（例如拒绝访问或没有找到）
  - 错误导致的服务器宕机（DoS）
- 用户错误输入回显到页面时没有进行过滤或转义导致的XSS攻击



## 5.不恰当的错误处理-解决方案

- 定义一套清晰和一致的错误处理机制
  - 简明扼要的易于用户理解的错误消息（例如，不同的错误消息对应一个错误代码id）
  - 为系统管理员记录重要信息（关联错误代码id）
  - 不要暴露出任何对攻击者有用的信息（程序的调试信息和异常时堆栈信息等）
- 当需要显示用户的错误输入时，一定要编码（过滤或转义）用户的错误输入
- 修改默认的错误页面（404, 401等）
- 执行代码复查



# Web应用程序设计实现的安全问题





## 6.脆弱的访问控制-示例

- 文档/软件的下载链接地址保护
  - <http://victim.org/docs/1.doc>
  - <http://victim.org/docs/download.do?id=1>
- Web应用程序的后台管理入口地址
  - <http://victim.org/admin>
  - <http://victim.org/console/login>
- 后台操作未执行用户身份认证
  - <http://victim.org/users/deleteUser.do?userId=001>
  - <http://victim.org/users/addUser.do?userId=001>



## 6.脆弱的访问控制-描述

- 内容或程序功能未能有效的保护以限制只允许合法用户的访问
- 典型案例
  - 不安全的id
  - 强制浏览（直接在浏览器的地址栏中输入URL）
  - 目录遍历
  - 文件访问权限
  - 客户端缓存



## 6. 脆弱的访问控制-描述

- 可能的原因
  - 认证只发生在用户登录时
  - 仅对URL进行鉴权，而不是对内容进行鉴权
  - 未采取集中式的授权管理，而是分散授权管理



## 6. 脆弱的访问控制-解决方案

- 对每个需要保护的请求进行检查，不仅是在用户第一次请求时进行检查
- 避免使用自己开发的访问控制，而是使用J2EE提供的CMS或者其他的一些安全框架，如Acegi
  - 采用声明式而非硬编码的访问控制
  - 集中化访问控制而非分散访问控制



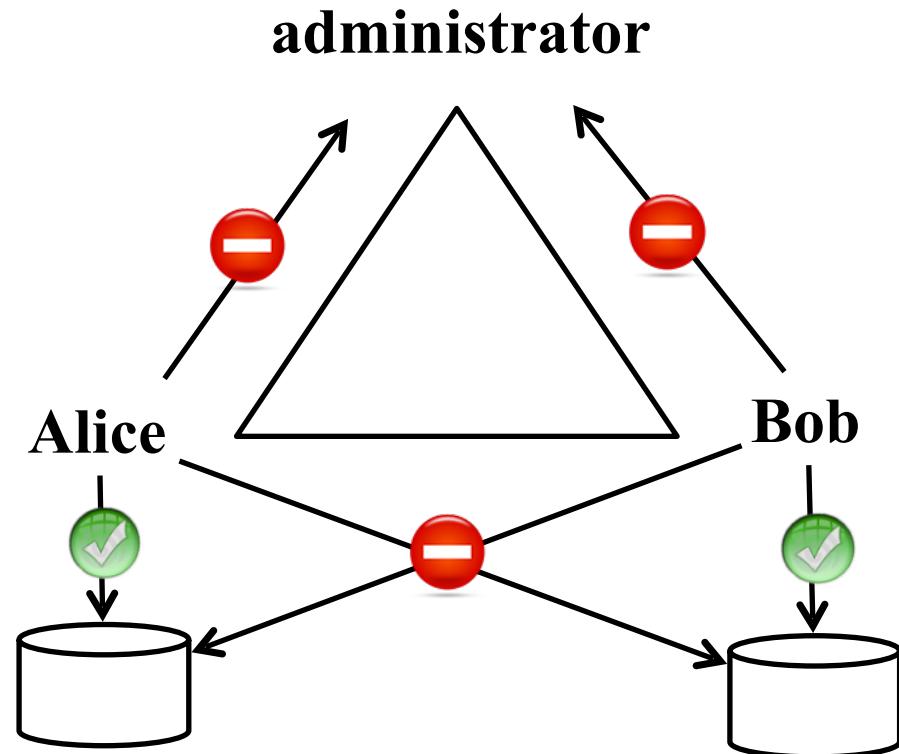
## 6. 脆弱的访问控制-解决方案 (1/3)

- 注意：J2EE容器默认允许所有URL的访问
- （可选）扩展基于实例的访问控制
- 防止客户端缓存重要内容：设置HTTP请求头和meta标签
- 在服务器端使用操作系统提供的访问控制保护文件的未经授权的访问



## 6.脆弱的访问控制-解决方案 (2/3)

- 业务模型的访问控制授权建模
  - 访问控制权限划分的三角形基本法则
- 平行权限访问
  - 属主权限检查
- 提升权限访问
  - 使用ACL





## 6. 脆弱的访问控制-解决方案 (3/3)

- 属主权限检查

主体	客体
alice	/srv/www/upload/1.doc
bob	/srv/www/upload/2.doc

- 使用ACL

主体	客体
alice	/user/alice/view.php /user/alice/add.php
bob	/user/bob/view.php



## 7.脆弱认证和会话管理-示例

- 未采用Session cookie，而是在URL中编码已通过认证的用户名和密码
  - [https://host/admin/list.jsp?  
password=0c6ccf51b817885e&username=11335984ea80882d](https://host/admin/list.jsp?password=0c6ccf51b817885e&username=11335984ea80882d)
- 上面的这个URL很容易被一次XSS攻击截获到



## 7.脆弱认证和会话管理-描述

- 脆弱的认证和会话管理
- 典型案例
  - 简单易猜解的用户名和用户口令
  - 存在缺陷的身份管理功能，例如密码修改功能，忘记密码和账户更新功能
  - 主动会话劫持，假冒已通过身份认证的合法用户
- HTTP协议的会话管理依赖于应用程序的实现
  - 使用jsessionid的URL重写
  - (Session) cookies



## 7.脆弱认证和会话管理-解决方案

- 使用强认证机制
  - 密码策略（密码强度，使用/更改/存储控制）
  - 安全传输（SSL）
  - 小心实现“找回密码”功能
  - 移除默认用户
- Session机制需要注意的问题
  - cookie必须是“安全”的（例如readonly机制）
  - Session id必须是“不可预测”的
- 尽量使用应用程序服务器提供的安全机制，而不是实现自己的安全机制



# 基础平台的安全问题





## 8. 不安全的存储-示例

- 日常备份策略
  - 程序的备份采用可擦写的设备，如移动硬盘、U 盘等
  - 使用UltraEdit等编辑器编辑文件后未删除编辑器自动生成的.bak备份文件，导致源代码泄露
- 数据未加密存储
  - 用户口令等机密数据以明文形式存储在数据库中



## 8. 不安全的存储-描述

- 敏感/重要的数据应该采取安全的存储方式
- 典型案例
  - 没有加密存储关键数据
  - 密钥、证书和密码等采用了不安全的存储介质
  - 弱随机数字发生器
  - 弱加密算法
  - 未采用安全的密钥交换机制



## 8. 不安全的存储-解决方案

- 仅存储那些必须存储的数据
  - 要求用户每次重新输入
  - 存储Hash值，而不是加密值
- 不允许对后台应用程序的直接访问
  - 数据库访问
  - 文件系统访问
- 不要在Web应用程序服务器所在的根目录存储重要文件
- 不要使用自己的加密算法，使用原生库



## 9. 不安全的配置管理-描述

- Web应用程序的运行环境包括
  - 应用程序服务器（如Tomcat、WebSphere和WebLogic等）和Web服务器（如IIS、Apache等）
  - 后台系统（数据库服务器、目录服务器、邮件服务器等）
  - 操作系统和基础网络架构
- 最常见的配置漏洞
  - 未及时更新安全补丁（操作系统、应用程序等）
  - 不必要的默认、备份、示例文件
  - 开放具有管理权限的服务
  - 默认的用户账户和默认口令
  - 配置不当的SSL证书
- 开发者和管理者（部署人员）之间缺乏沟通



## 9. 不安全的配置管理-解决方案

- 为每一个服务器配置创建一个安全基准
  - 配置所有的安全机制（补丁更新策略、访问控制策略、密码策略等）
  - 关闭所有不使用的服务
  - 创建用户角色、权限和账户，包括禁用所有默认账户或修改默认口令
  - 日志和警告系统
- (半) 自动化配置过程
  - 使用项目批量构建工具（如Ant）和Ghost
- 保持更新
  - 保持Web应用程序的运行环境的安全补丁的更新
  - 更新安全配置基准
  - 定期执行内部和外部的漏洞扫描工具



## 10. 拒绝服务攻击-示例

- 应用程序从后台的内容管理系统获取了大量的信息
- 一次前台的请求导致了对后台数据库的多次操作请求



## 10. 拒绝服务攻击-描述

- Web应用程序非常容易遭受拒绝服务攻击，这是由于Web应用程序本身无法区分正常的请求通信和恶意的通信数据
- 容易产生大量的攻击负载



## 10. 拒绝服务攻击-典型案例

- 有限的资源特别容易成为DoS攻击的目标
  - 带宽
  - 数据库连接
  - 磁盘容量
  - CPU资源
  - 内存
  - 最大并发线程数 / 可用文件句柄
- 特定于用户的DoS
  - 用户并发访问控制锁
  - 用户密码更改
- 未处理的程序异常



## 10. 拒绝服务攻击-解决方案

- 避免可能会导致资源密集性消耗的请求
  - CPU: 频繁的请求, JDBC连接
  - 内存或磁盘容量: 大数据量的POST和过量的HttpSession数据
  - 匿名用户访问的限制
- 在大负载的情况下测试应用程序的性能
- 利用缓存服务器或限制数据库连接
- 小心使用“锁”机制



## 又是输入相关的问题





## 11. 跨站点请求伪造-示例(1/2)

- 利用站点已验证通过的用户会话（无需获取用户的登录凭证） 教育人博客CSRF漏洞演示
  - [http://victim.org/addFriend.do?  
friend=attacker@gmail.com](http://victim.org/addFriend.do?friend=attacker@gmail.com)
- 当一个已经登录victim.org的用户打开一个包含有XSS攻击代码的页面（或者通过一个隐藏的iframe），并且该XSS代码执行上述的URL请求，则该用户就会执行addFriend这个操作
- 结果：用户在不知情的情况下添加了攻击者作为自己的好友



## 11.跨站点请求伪造-示例(2/2)

- 新浪微博2011年6月28日晚间的大规模XSS+CSRF蠕虫事件
  - 事件时间线：16分钟
    - 20:14，开始有大量带V的认证用户中招转发蠕虫
    - 20:30，2kt.cn中的病毒页面无法访问
    - 20:32，新浪微博中hellosamy用户无法访问
    - 21:02，新浪漏洞修补完毕
  - 感染人数：32961人！
- 病毒作者使用的用户名是为了向世界上第一个XSS+CSRF蠕虫作者samy致敬



## 11.跨站点请求伪造-描述

- 从名称上来看类似跨站点攻击，但实质上完全不同：
  - XSS是滥用用户对Web站点的信任
  - CSRF是滥用Web站点对其授权用户的信任
- 伪装成来自受信任站点的合法用户
  - 有时也被称为会话劫持攻击
- 典型案例
  - 诱骗用户访问一个图片源标记为恶意请求链接的页面，从而触发一个异步的恶意远程调用
  - 接受受信任并且通过验证的用户的输入但并不检查数据的来源地址



## 11.跨站点请求伪造-与XSS的联系

- 跨站点请求伪造通常伴随XSS漏洞利用过程
- 先有XSS，再有CSRF
  - 借助XSS漏洞获得在用户浏览器执行脚本的机会
- 没有XSS，一样可以有CSRF
  - 借助已通过网站认证和获得授权的用户浏览器会话
    - 假借用户的合法cookie
    - 一个URL即可触发一次CSRF
    - <http://victim.org/deluser.php?id=admin>



## 11.跨站点请求伪造-解决方案

- 使用GET方法进行查询操作
  - 方便用户加入收藏夹
  - 可以通过电子邮件的方式发送链接地址给其他用户
- 使用POST方法进行更新操作
  - 不能被用户加入收藏夹或通过电子邮件告知
  - 不能被随意的重新提交
  - 增加XSS攻击的难度



## 11.跨站点请求伪造-解决方案

- 在链接中使用时间戳和加密（防止简单重放）
- 在关键应用处使用CAPTCHA机制
  - 一次性口令
  - 图片验证码
- 构造不可预测性



## 12. PHP的文件包含漏洞(1/4)

- 漏洞代码实例一

```
include($server_root . '/myapp_header.php');
```

- 当php.ini中**register\_globals =on**时

- 攻击者通过构造**\$server\_root**作为请求参数，动态控制文件包含指令

- 漏洞代码实例二

```
include($_GET['filename']);
```

- 攻击者通过构造URL请求参数中的**filename**字段，动态控制文件包含指令

- <http://test.com/test.php?filename=../../../../etc/passwd>



## 12. PHP的文件包含漏洞(2/4)

- 原理
  - PHP代码的文件包含指令中含有动态变量参数，该参数会被攻击者恶意控制实现动态包含任意文件
  - 当动态包含服务器的本地文件时，导致本地文件包含漏洞
  - 当动态包含远程第三方文件时，导致远程文件包含漏洞



## 12. PHP的文件包含漏洞(3/4)

The screenshot shows a browser window with the URL `qun.qq.com/air/?w=n&c=../../../../../../../../etc/passwd%00.html&a=dismiss&g=`. A red box highlights the URL bar. The page content displays a long list of system users and their details from the /etc/passwd file, including root, daemon, bin, sys, adm, etc.

```
t:x:25:25:Batch jobs daemon:/var/spool/atjobs:/bin/bash bin:x:1:1:bin:/bin/bash daemon:x:2:2:Daemon:/sbin/nologin lp:x:4:7:Printing daemon:/var/spool/lpd:/bin/bash mail:x:8:12:Mailer:/var/spool/clientmqueue:/bin/false messagebus:x:100:101:User for D-BUS:/var/run/dbus:/bin/false nobody:x:65534:65533:nobody:/var/lib/nobody:/bin/bash ntp:x:74:103:NTP daemon:/var/lib/ntp:/bin/false postfix:x:51:51:Postfix Daemon:/var/spool/postfix:/bin/false postgres:x:26:26:PostgreSQL Server:/var/lib/pgsql/root:x:0:0:root:/root:/bin/bash sshd:x:71:65:SSH daemon:/var/lib/sshd:/bin/false suse-ncc:x:102:104:Novell Customer User:/var/lib/YaST2/suse-ncc-fakehome:/bin/bash wwwrun:x:30:8:WWW daemon apache:/var/lib/wwwrun:/bin/false www:x:30:8:WWW :/var/lib/www:/bin/false man:x:13:62:Manual pages viewer:/var/cache/man:/bin/bash news:x:9:13:News system:/etc/news:/bin/bash uucp:x:10:14:Unix-to-Unix CoPy system:/etc/uucp:/bin/bash oicq:x:1000:100::/home/oicq:/bin/false memcached:x:103:107:user for memcached:/var/lib/memcached:/bin/false webdep:x:1001:100::/home/webdep:/bin/bash
```

现在网速有一些不稳哦，刷新一下试试看

[刷新重试](#)



## 12. PHP的文件包含漏洞(4/4)

- 解决方案建议
  - 严格检查变量是否已经初始化
  - 输入参数过滤
    - 服务器端验证
  - 严格检查include类的文件包含函数中的参数是否外界可控
    - 源代码检查



## 13. 文件上传漏洞

- 允许用户上传文件可能会让黑客
  - 在网页中嵌入恶意代码
    - 网页木马：控制客户端（网站用户）
    - XSS漏洞 / CSRF漏洞 / 构造钓鱼页面…
  - 上传webshell
    - 控制服务器
- 文件上传漏洞原理
  - 接下来会通过PHP代码实例进行讲解



# 文件上传过程抓包截图

```
POST /--.php HTTP/1.1
Host: -
Connection: keep-alive
Referer: http://-----/l----.php
Content-Length: 69162
Cache-Control: max-age=0
Origin: http://-----
User-Agent: Mozilla/5.0 (X11; Linux i686) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/14.0.835.162 Safari/535.1
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryUBY5JVS03LX9pbUu
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,sdch
Accept-Language: zh-CN,zh;q=0.8
Accept-Charset: UTF-8,*;q=0.5
Cookie: -----
-----WebKitFormBoundaryUBY5JVS03LX9pbUu
Content-Disposition: form-data; name="name"
-----WebKitFormBoundaryUBY5JVS03LX9pbUu
Content-Disposition: form-data; name="sex"
```

```
$filename = $_FILES['userfile']['name'];
Content-Disposition: form-data; name="userfile"; filename=".....doc"
Content-Type: application/msword
$contentType = $_FILES['userfile']['type'];

.....>
~
```



# 文件上传漏洞PHP代码剖析

如果上传文件名为 `test.php.doc` 呢？如果是 `test.php%00doc` 呢？

```
4      $tuozhanming = explode ( '.', $_FILES ['userinfo'] ['name'] );
38     $_FILES ['userinfo'] ['name'] = time () . "." . $tuozhanming [1];
39     $upfile = 'uploads/' . $_FILES ['userinfo'] ['name'];
```

根据PHP官方的文档说明，该值完全可以被伪造！黑客只需修改浏览器的post请求头即可绕过这段代码检查，进而上传任意类型的文件！

```
31     //判断文件类型
32     $type = $_FILES ['userinfo'] ['type'];
33     if (($type != 'application/msword') && ($type != 'application/vnd.
openxmlformats-officedocument.wordprocessingml.document') && ($type !=
'application/pdf') && ($type != 'application/octet-stream')) {
34         echo 'Problem: file format is not permissible';
35         exit ();
36     }
```



# 有意思NULL字符截断问题

- 何为NULL字符
  - %00
  - ASCII码为0
- PHP官方在2010年12月9日PHP 5.3.4版本正式修复了该漏洞
  - CVE-2006-7243
  - 用了4年时间修补一个漏洞！
  - PHP 5.3.4之前版本仍然受此漏洞影响
- 不仅仅是PHP语言受此漏洞影响！



# 判断文件类型的安全实践(1/3)

- 读取文件头标识

- PNG(8 bytes):89 50 4E 47 0D 0A 1A 0A

- GIF(6 bytes):47 49 46 38 39 61 (GIF89a)

FF D8 FF E0 xx xx 4A 46 49 46 00	JFIF, JPE, JPEG, JPG	ÿØÿà...JF IF. <a href="#">JPEG/JFIF graphics file</a> Trailer: FF D9 (ÿÙ)
FF D8 FF E1 xx xx 45 78 69 66 00	JPG	ÿØÿá...Ex if. Digital camera JPG using <a href="#">Exchangeable Image File Format (EXIF)</a> Trailer: FF D9 (ÿÙ) See " <a href="#">Using Extended File Information (EXIF) File Headers in Digital Evidence Analysis</a> " (P. Alvarez, <i>IJDE</i> , 2(3), Winter 2004) and <a href="#">ExifTool Tag Names</a>
FF D8 FF E8 xx xx 53 50 49 46 46 00	JPG	ÿØÿè..SP IFF. <a href="#">Still Picture Interchange File Format (SPIFF)</a> Trailer: FF D9 (ÿÙ)

## JPG/JPEG

**NOTES on JPEG file headers:** It appears that one can safely say that all JPEG files start with the three hex digits 0xFF-D8-FF. The fourth digit is also indicative of JPEG content. Various options include:

- 0xFF-D8-FF-DB — Samsung D807 JPEG file.
- 0xFF-D8-FF-E0 — Shown above. [Standard JPEG/JFIF file](#).
- 0xFF-D8-FF-E1 — Shown above. Standard JPEG/Exif file.
- 0xFF-D8-FF-E2 — [Canon EOS-1D JPEG file](#).
- 0xFF-D8-FF-E3 — Samsung D500 JPEG file.
- 0xFF-D8-FF-E8 — Shown above. [Still Picture Interchange File Format \(SPIFF\)](#).



## 判断文件类型的安全实践(2/3)

- 文件头标识指纹匹配足够安全吗？

—No!

- 对于GIF图片

—补充使用getimagesize()

—限制上传的GIF图片分辨率

```
huangwei@huangwei-VirtualBox:~/ns$ cat 1.php
GIF89a
<?php
$a = "haha.jpg\0.php";
$b = "1.php\0.txt";

if(file_exists($b)) {
echo "unpatched file_exists(), vulnerable php version", PHP_EOL;
} else {
echo "safe file_exists()", PHP_EOL;
}

var_dump(strlen($a));
var_dump(basename($a));
var_dump(pathinfo($a));
huangwei@huangwei-VirtualBox:~/ns$ php 1.php
GIF89a
safe file_exists()
int(13)
string(13) "haha.jpg.php"
array(4) [
    ["dirname"]=>
        string(1) "."
    ["basename"]=>
        string(13) "haha.jpg.php"
    ["extension"]=>
        string(3) "php"
    ["filename"]=>
        string(9) "haha.jpg"
]
huangwei@huangwei-VirtualBox:~/ns$ file 1.php
1.php: GIF image data, version 89a, 15370 x 28735
huangwei@huangwei-VirtualBox:~/ns$
```



## 判断文件类型的安全实践(3/3)

- 对于其他类型文件
  - 禁用上传目录的脚本执行权限
    - 例如，apache可以使用.htaccess

```
<Directory /upload>
    AllowOverride All
</Directory>
<Location /upload>
    Options None
    Options +IncludesNoExec -ExecCGI
    RemoveHandler .php .phtml .php3 .php4 .php5
    RemoveType .php .phtml .php3 .php4 .php5
    php_flag engine off
    php_admin_flag engine off
    AddType text/plain .html .htm .shtml .php
</Location>
```



# 继续文件上传漏洞

- 即使
  - 检查是否判断了上传文件类型及后缀
  - 定义上传文件类型白名单
  - 文件上传目录禁止脚本解析
- 仍然推荐
  - 定义文件名白名单
  - 上传后文件统一重命名
    - 杜绝XSS漏洞 / 文件包含漏洞 / 字符编码漏洞 …



## 14. 字符编码漏洞——为什么要了解字符编码漏洞

- 在某些东方国家里这种攻击越来越普遍
- 很多程序员对这种攻击还不了解
- 通过分析受字符集漏洞影响的各类Web应用程序，我们可以更深入的了解和修正此类安全问题



# 字节流和数据的本质

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F		0123456789ABCDEF
00000000:	EF	BB	BF	E4	B8	AD	E6	96	87	E6	B5	8B	E8	AF	95		中 文 测 试	

UTF-8 编码

行:0 /1 列:3      字符位置:3 /15      选取长度:0      UTF-8.BOM.DOS

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F		0123456789ABCDEF
00000000:	FF	FE	2D	4E	87	65	4B	6D	D5	8B							中 文 测 试	

**BOM:Byte Order Mark——Windows记事本惹的祸**

行:0 /1 列:2      字符位置:2 /10      选取长度:0      UTF-16LE.BOM.DOS

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F		0123456789ABCDEF
00000000:	FE	FF	4E	2D	65	87	6D	4B	8B	D5							中 文 测 试	

Unicode 编码  
(little-endian)

行:0 /1 列:2      字符位置:2 /10      选取长度:0      UTF-16BE.BOM.DOS

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F		0123456789ABCDEF
00000000:	BF	5C	27	D6	D0	CE	C4	B2	E2	CA	D4	C1	5D				纟'中 文 测 试 纣	

GBK 编码

行:0 /1 列:0      字符位置:0 /13      选取长度:0      GBK.DOS

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F		0123456789ABCDEF
00000000:	5C																\\	

GBK 编码

行:0 /1 列:0      字符位置:0 /1      选取长度:0      GBK.DOS



# 常见的字符编码方式

- SBCS (Single Byte Character Set)
  - iso8859-1
- MBCS (Multi Byte Character Set)
  - variable-width encoding (GBK、GB2312、UTF-8)
  - GBK编码表
- DBCS (Double Byte Character Set)
  - fixed-width encoding (很少见)
- Unicode (Wide Character Set)
  - <http://en.wikipedia.org/wiki/Unicode>
  - UTF-8是针对Unicode的一种可变长度字符编码



## 字符集处理过程中可能出现的安全问题 (1/3)

- 上下层使用的字符集不一致，导致数据的意义出现问题
  - Web应用各层之间都需要对数据做出适当的解释
  - 理解不一致就会出现问题
- IE字符集自动识别导致XSS
  - IE会采用“智能”识别技术去识别字符集编码：
    - 根据HTTP Header头里的指定
    - 根据HTML头部数据自行选择
    - Meta头里指定的Charset
    - IE 8 Strip-meta tag 攻击



## 字符集处理过程中可能出现的安全问题 (2/3)

- 错误的进行UTF-8解码
- 对于非法数据的处理理解上不一致
  - 替换成? 或者其他的字符
  - 直接抛弃
  - 截断
  - 这几种处理方式都可能带来安全问题



## 字符集处理过程中可能出现的安全问题 (3/3)

- 某些字符集天生的缺陷

—如GB2312、BIG5等字符集低位包含某些元字符  
— \ [ | { < / > ' ' " ...

—这样程序即使对字符集完全处理正确，在处理字节流时，  
很可能吃掉上面的这些元字符，从而造成元字符过滤的  
匹配失败 (bypass filter)

offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF
00000000:	5C	00	00	5B	00	00	7C	00	00	7B	00	00	3C	00	00	2F	\ [   { < /
00000010:	00	00	3E	00	00	60	00	00	27	00	00	22					> ' ' " ]



# 几个MBCS处理的漏洞list

- Hotmail XSS <http://www.securityfocus.com/archive/1/450723>
- Yahoo webmail xss
  - Yahoo修复过在html属性里出现的字符集问题
  - 字符集问题不只出现在html属性，同样存在于css, js等地方
  - Yahoo的Filter基于语法分析，语法分析的时候为非MBCS字符集
  - Filter没有处理好在css里的非法字符，导致MBCS环境下的xss
  - Exploit

```
- <style>
#x{
background: url('http://www.google.cn{chr}');background:
url('http://;color:red;www.google.cn');
}
</style>
<div id="x">codz</div>
```



## 几个UTF-7处理的漏洞list

---

- Microsoft Internet Information Services  
UTF-7 XSS Vulnerability [MS06-053]
- Apache2 Undefined Charset UTF-7 XSS  
Vulnerability
- XSS with UTF-7 in Google



# 字符集处理过程漏洞的利用

- 钓鱼
- 逃避杀毒软件查杀
- 规避各种过滤规则
  - XSS过滤绕过
  - SQL Injection过滤绕过
  - PHP magic\_code机制绕过



## 字符集问题的安全防护

- 用正确的字符集编码理解数据
- 理解上下层之间处理数据的方式
- 尽量使用相对安全的编码：UTF-8
- 处理好非法数据



# UTF-8就安全了吗？

- Apache Tomcat Directory Traversal Vulnerability
  - 类似于2000年IIS 5.0的Unicode漏洞
  - %c0%ae%c0%ae被解析成..
  - CVE-2008-2938
  - 影响Tomcat 6.0.18之前所有版本
- 应用程序解析不当一样导致漏洞！



## 15. 第三方程序漏洞

- 2009年discuz站点批量被黑事件中的教训
  - Web内容引用/代码包含需谨慎
    - 第三方网站广告链接  
`<img src= " http://malware-script ">`
    - 第三方网站脚本/API  
`<script src= " http://malware-script"></script>`
    - 第三方网站页面嵌入  
`<iframe><div><script>`
- Java的Struts 2.x框架漏洞
  - HTTP GET参数中的同名参数处理缺陷



## 2009年discuz站点批量被黑事件(1/2)

- 事件过程还原

- 2009年1月8日11:38分 部分使用Discuz!搭建论坛的站长发现论坛首页出现" Hacked by ring04h, just for fun! "的提示
- 2009年1月8日11:54分 官方安全部门发现站点 <http://customer.discuz.net> 域名被劫持
- 2009年1月8日11:55分 被劫持域名的重新定向工作完成
- 2009年1月8日12:15分 官方论坛已经发放代码清除产品包



## 2009年discuz站点批量被黑事件(2/2)

- 事件原因分析

—站长登录Discuz!后台首页后， Discuz!系统将自动输出类似如下的代码

```
<script language="JavaScript"
src="http://customer.discuz.net/news.php?
update=dW5pcXV1aWQ9Sk1QUFdNYTEzMFR4NFRoaS22ZXJzaW9uPTYuMS4wRiZyZWx1
YXN1PTIwMDgxMTE3JnBocD01Lj1uMTAmBX1zcWw9MS4xLjM3LWNvbW11bm10eS2jaGF
yc2U0PXU0Zi04JmJibmFtZT1EaXNjdXo1MjE1MjBCb2FyZC2tYXN0ZXJtb2JpbGU9Jg
%3D%3D&md5hash=9f1c792e&timestamp=1250524671"></script>
```

—攻击者挟持Discuz!后台信息通知系统域名并在news.php里面写入恶意代码

```
<script>function init() { document.write('Hacked by ring04h, just for fun!
');}window.onload = init;</script>
```



# Java的Struts 2.x框架漏洞分析(1/4)

- HTTP Parameter Pollution 处理缺陷

- http://www.example.com/app/test!aa.action?id=ccc&id=aaa

- Action 中 定义了

- private String id;

```
public String getId() {  
    return id;  
}
```

```
public void setId(String id) {  
    this.id = id;  
}
```



## Java的Struts 2.x框架漏洞分析(2/4)

- Action会取到id的值为“ccc, aaa”注意中间是有空格的。
- 这种数据是由struts2把两个参数合并而成的，但是如果我们request.getParameter("id");拿到的值，却只是第一个（值为ccc）。



## Java的Struts 2.x框架漏洞分析(3/4)

- 以下是问题代码的片段：
- String id = request.getParameter("id"); if(id!=null&& id.indexOf("'"')>-1){  
    …//SQL注入检测代码  
}

//业务逻辑代码

```
String sql = "select book_name,book_content from  
books"; if (id != null) { sql += " where book_id like '%"  
+ id + "%'"; }
```



# Java的Struts 2.x框架漏洞分析(4/4)

- 漏洞利用代码

—`http://www.example.com/app/test!findUserById.action?id=aaaaaa&id=a' union select name,pass from user where "<>'`

- 解释

—上述代码直接绕过了拦截器的判断。因为拦截器获取的`request.getParameter("id")`，是第一个参数的值aaaaa，单引号成功逃避检测！



# Web应用程序漏洞Top 10 in 2010

OWASP Top 10 – 2007 (Previous)	OWASP Top 10 – 2010 (New)
A2 – Injection Flaws	A1 – Injection
A1 – Cross Site Scripting (XSS)	A2 – Cross-Site Scripting (XSS)
A7 – Broken Authentication and Session Management	A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object Reference	A4 – Insecure Direct Object References
A5 – Cross Site Request Forgery (CSRF)	A5 – Cross-Site Request Forgery (CSRF)
<was T10 2004 A10 – Insecure Configuration Management>	A6 – Security Misconfiguration (NEW)
A8 – Insecure Cryptographic Storage	A7 – Insecure Cryptographic Storage
A10 – Failure to Restrict URL Access	A8 – Failure to Restrict URL Access
A9 – Insecure Communications	A9 – Insufficient Transport Layer Protection
<not in T10 2007>	A10 – Unvalidated Redirects and Forwards (NEW)
A3 – Malicious File Execution	<dropped from T10 2010>
A6 – Information Leakage and Improper Error Handling	<dropped from T10 2010>



# Web应用程序漏洞Top 10 in 2013

OWASP Top 10 – 2010 (Previous)	OWASP Top 10 – 2013 (New)
A1 – Injection	A1 – Injection
A3 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References	A4 – Insecure Direct Object References
A6 – Security Misconfiguration	A5 – Security Misconfiguration
A7 – Insecure Cryptographic Storage – Merged with A9 →	A6 – Sensitive Data Exposure
A8 – Failure to Restrict URL Access – Broadened into →	A7 – Missing Function Level Access Control
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<buried in A6: Security Misconfiguration>	A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards	A10 – Unvalidated Redirects and Forwards
A9 – Insufficient Transport Layer Protection	Merged with 2010-A7 into new 2013-A6

[https://www.owasp.org/index.php/Category:OWASP\\_Top\\_Ten\\_Project](https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)



## 再谈谈平台相关的漏洞

- Tomcat 4.x泄漏文件源代码漏洞
- nginx文件类型错误解析漏洞
- 平台相关弱口令/空口令
  - FTP / SSH / 远程桌面 …
- PHP语言和解析器漏洞
- 特定浏览器上的“诡异”漏洞



# Tomcat 4.x泄漏文件源代码漏洞演示

- 源代码泄漏缺陷

```
<%!
//str_BankNull():接收传递过来的参数时，将null转换成"",同时去除前后的空格.
String str_BankNull(String s) {
    return (s == null || s.length() == 0) ? "" : s.trim();
}
//character():根据不同情况判断是否进行字符集的转换
String character(String s)
    throws IOException{
    //Tomcat在unix平台
    //s = new String(s.trim().getBytes(),"8859_1");
    //Tomcat在windows平台
    s = new String(s.trim());
    return s;
}
```

Pa) is not available.

Apache Tomcat/4.1.27

functions.jspa



# nginx文件类型错误解析漏洞(1/3)

- nginx默认以CGI的方式支持PHP执行

## —配置文件中支持正则表达式

```
1 location ~ \.php$ {  
2     root html;  
3     fastcgi_pass 127.0.0.1:9000;  
4     fastcgi_index index.php;  
5     fastcgi_param SCRIPT_FILENAME /scripts$fastcgi_script_name;  
6     include fastcgi_params;  
7 }
```

- CGI方式的PHP解释执行环境核心是环境变量的正确配置

[HTTP_HOST]	[SERVER_NAME]	[GATEWAY_INTERFACE]	[SCRIPT_NAME]
[HTTP_CONNECTION]	[SERVER_ADDR]	[SERVER_PROTOCOL]	[SCRIPT_FILENAME]
[HTTP_USER_AGENT]	[SERVER_PORT]	[REQUEST_METHOD]	[PATH_INFO]
[HTTP_ACCEPT]	[REMOTE_ADDR]	[QUERY_STRING]	[PATH_TRANSLATED]
[PATH]	[DOCUMENT_ROOT]	[REQUEST_URI]	[PHP_SELF]
[SERVER_SIGNATURE]	[SERVER_ADMIN]		
[SERVER_SOFTWARE]	[REMOTE_PORT]		



## nginx文件类型错误解析漏洞(2/3)

- 假设目标站点存在以下资源
  - <http://victim.org/cnss.jpg>
- 攻击者通过以下URL进行访问
  - <http://victim.org/cnss.jpg/cnss.php>
- 如果nginx配置文件中
  - `cgi.fix_pathinfo=1` (默认配置)
- 此时的nginx CGI环境变量设置情况为  
**[SCRIPT\_FILENAME]=/scripts/cnss.jpg**  
**[PATH\_INFO]=cnss.php**



## nginx文件类型错误解析漏洞(3/3)

- **http://victim.org/cnss.jpg**

HTTP/1.1 200 OK

Server: nginx/0.6.32

Date: Thu, 20 May 2010 10:05:30 GMT

Content-Type: image/jpg

Content-Length: 18

Last-Modified: Thu, 20 May 2010 06:26:34 GMT

Connection: keep-alive

Keep-Alive: timeout=20

Accept-Ranges: bytes

- **http://victim.org/cnss.jpg/cnss.php**

HTTP/1.1 200 OK

Server: nginx/0.6.32

Date: Thu, 20 May 2010 10:06:49 GMT

Content-Type: text/html

Transfer-Encoding: chunked

Connection: keep-alive

Keep-Alive: timeout=20

X-Powered-By: PHP/5.2.6

成功实现以PHP来解析任意类型文件



# 平台相关弱口令/空口令

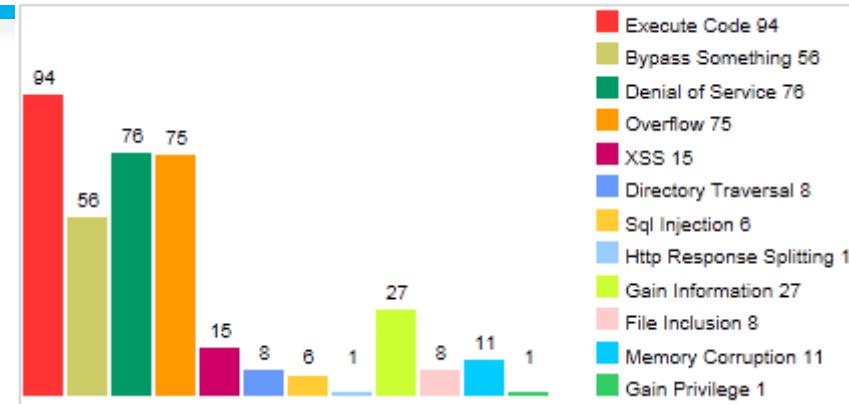
- 防御森严的网络和系统往往都是被弱口令/空口令轻易击溃！



# PHP语言和解析器漏洞

- PHP官方的bug统计
- CVE的漏洞数据统计

Vulnerabilities By Type



Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
<a href="#">2000</a>	3		<a href="#">2</a>												
<a href="#">2001</a>	4		<a href="#">1</a>							<a href="#">1</a>					
<a href="#">2002</a>	14	<a href="#">4</a>	<a href="#">2</a>	<a href="#">1</a>				<a href="#">1</a>			<a href="#">2</a>				
<a href="#">2003</a>	11	<a href="#">4</a>	<a href="#">5</a>	<a href="#">5</a>				<a href="#">1</a>			<a href="#">1</a>				
<a href="#">2004</a>	6		<a href="#">2</a>					<a href="#">1</a>			<a href="#">1</a>				
<a href="#">2005</a>	18	<a href="#">7</a>	<a href="#">4</a>	<a href="#">2</a>				<a href="#">1</a>	<a href="#">1</a>		<a href="#">3</a>				
<a href="#">2006</a>	44	<a href="#">1</a>	<a href="#">14</a>	<a href="#">8</a>		<a href="#">2</a>	<a href="#">5</a>	<a href="#">1</a>	<a href="#">1</a>	<a href="#">11</a>	<a href="#">1</a>			<a href="#">6</a>	
<a href="#">2007</a>	116	<a href="#">19</a>	<a href="#">51</a>	<a href="#">36</a>	<a href="#">2</a>		<a href="#">2</a>	<a href="#">3</a>		<a href="#">19</a>	<a href="#">2</a>	<a href="#">1</a>		<a href="#">1</a>	<a href="#">28</a>
<a href="#">2008</a>	21	<a href="#">5</a>	<a href="#">5</a>	<a href="#">6</a>				<a href="#">3</a>		<a href="#">5</a>	<a href="#">1</a>				<a href="#">1</a>
<a href="#">2009</a>	22	<a href="#">7</a>		<a href="#">1</a>		<a href="#">1</a>	<a href="#">2</a>			<a href="#">3</a>	<a href="#">1</a>				<a href="#">1</a>
<a href="#">2010</a>	35	<a href="#">9</a>	<a href="#">6</a>	<a href="#">7</a>	<a href="#">5</a>	<a href="#">2</a>	<a href="#">2</a>			<a href="#">6</a>	<a href="#">16</a>				<a href="#">2</a>
<a href="#">2011</a>	34	<a href="#">20</a>	<a href="#">2</a>	<a href="#">9</a>	<a href="#">4</a>	<a href="#">1</a>				<a href="#">4</a>	<a href="#">1</a>				<a href="#">5</a>
Total	328	<a href="#">76</a>	<a href="#">94</a>	<a href="#">75</a>	<a href="#">11</a>	<a href="#">6</a>	<a href="#">15</a>	<a href="#">8</a>	<a href="#">1</a>	<a href="#">56</a>	<a href="#">27</a>	<a href="#">1</a>		<a href="#">8</a>	<a href="#">36</a>
% Of All		23.2	28.7	22.9	3.4	1.8	4.6	2.4	0.3	17.1	8.2	0.3	0.0	2.4	



## 特定浏览器上的诡异漏洞 (1/2)

- 你能读懂下面这个URL吗？

— <http://light.lz.taobao.com/?r=%68%74%74%70%3A//%6E%35%76%31%2E%63%6F%6D%5Ct.taobao.com/a.asp?id=30760>

解码之

— <http://light.lz.taobao.com/?r=http://n5v1.com\t.taobao.com/a.asp?id=30760>

用户点击后跳转到这个网址

- <http://n5v1.com\t.taobao.com/a.asp?id=30760>



## 特定浏览器上的诡异漏洞 (2/2)

- <http://n5v1.com\ t.taobao.com/a.asp?id=30760>



- <http://n5v1.com/ t.taobao.com/a.asp?id=30760>





# 优秀的程序员



兴趣：时刻充满好奇心，要学习

中国传媒大学



# 蹩脚的程序员



不能以计算机和程序设计为乐



## 本章内容提要

- 网络与系统渗透基本原理
- 网络与系统渗透案例讲解
- 渗透测试工具
- 实验讲解



## 案例讲解

- 案例一：某大型门户网站的渗透纪实
- 案例二：某个人电脑的渗透纪实



# 案例一：某大型门户网站的渗透纪实

- 零扫描
  - Google Hacking
- 拓扑扫描
  - 域名扫描 / 主机扫描 / 服务扫描 …
- 确定渗透发力点
- 渗透过程中的细节问题
- 总结报告



## 案例二：某个人电脑的渗透纪实

- 社会工程学钓鱼案例
  - 邮箱的SMTP认证漏洞
  - 用户安全意识弱
  - 所有应用和网络服务都使用单一口令
    - 单个口令泄露整个身份认证防护体系瘫痪



## 案例启示

- 如果没有漏洞（弱点），攻击无法得手
- 安全防护的木桶原理
  - 任何一个网络与系统的安全短板都能成为渗透的跳板和入口点
- 永远不要忽视人的因素！



# 本章内容提要

- 网络与系统渗透基本原理
- 网络与系统渗透案例讲解
- 渗透测试工具
- 实验讲解



# 渗透测试工具简介

- 工具分类

- 自动化程度

- 全自动化
  - 半自动化
  - 手工测试

- 攻击性

- 漏洞扫描
  - 漏洞利用

- 测试方法

- 主动测试
  - 被动测试

- 测试对象

- 网络应用/服务
  - WEB应用/  
SMTP/IMAP/  
.....

- 操作系统

- 本地应用/服务



# 自动化工具or手工测试?

- 自动化工具
  - 优势：通用性高、节省测试时间、保证测试的覆盖率
  - 不足：针对性不强、误检率通常较高、容易触发安全设备（防火墙、IDS等）的报警
- 手工测试
  - 优势：针对性强、准确性高、欺骗性更强
  - 不足：检测效率低、费时费力



# 再次重申知法守法

一. 2000 年以前.....	1.1 中华人民共和国保守国家秘密法.....
1.2	1.2 中华人民共和国计算机信息系统安全保护条例.....
1.3	三. 2001 年.....
1.4	3.1 计算机软件保护条例.....
1.	四. 2002 年.....
1.	4.1 信息产品测评办法.....
1.	五. 2003 年.....
1.	5.1 广东省电子政务信息网络安全管理规定.....
六. 2004 年.....	6.1 中华人民共和国电信条例.....
七. 2005 年.....	7.1 互联网安全保护技术措施规定.....
	2.2 计算机信息系统安全保护条例.....
	2.3 互联网信息服务管理办法.....
	2.4 中华人民共和国电信条例.....
	2.5 全国人大常委会关于维护互联网安全的决定.....
	2.6 联网单位安全员管理办法.....

7.2 商用密码产品销售管理规定.....	79
7.3 电子认证服务密码管理办法.....	82
7.4 商用密码科研管理规定.....	83
7.5 商用密码产品生产管理规定.....	85
7.6 证券期货业信息安全保障管理暂行办法.....	88
7.7 电子认证服务管理办法.....	91
八. 2006 年.....	
8.1 关于加强新技术产品使用保密管理的通知.....	96
8.2 信息网络传播权保护条例.....	96
九. 2007 年.....	
9.1 商用密码产品使用管理规定.....	97
9.2 信息安全等级保护管理办法.....	102
9.3 境外组织和个人在华使用密码产品管理办法.....	102
十. 2009 年.....	
10.1 刑法修正案（七）关于信息安全的修订与解读.....	103
10.2 深圳经济特区企业技术秘密保护条例.....	111
十一. 2010 年.....	
11.1 通信网络安全防护管理办法.....	113
11.2 中华人民共和国保守国家秘密法.....	119
11.3 中央企业商业秘密保护暂行规定.....	122



## 再次重申知法守法

刑法修正案(七)在刑法第285条中增加两款(第二款、第三款):

违反国家规定,侵入前款规定以外的计算机信息系统或者采用其他技术手段,获取该计算机信息系统中存储、处理或者传输的数据,或者对该计算机信息系统实施非法控制,情节严重的,处三年以下有期徒刑或者拘役,并处或者单处罚金;情节特别严重的,处三年以上七年以下有期徒刑,并处罚金。

提供专门用于侵入、非法控制计算机信息系统的程序、工具,或者明知他人实施侵入、非法控制计算机信息系统的违法犯罪行为而为其提供程序、工具,情节严重的,依照前款的规定处罚。



# Web渗透工具——手工测试

- Firefox的渗透测试相关扩展
  - Firebug
  - TamperData
  - FireForce
  - HackBar
  - Live HTTP Headers
  - GroundSpeed
  - XHTML Mobile Profile + wml browser + Modify Headers



# Firebug

---

- 调试工具
  - HTML / CSS / Javascript
  - 网络通信数据
- 编辑工具
  - 所见即所得编辑
  - 编辑效果即时生效
- 逆向工具
  - 反查页面元素对应代码
    - HTML/XPATH/CSS PATH



# TamperData

---

- 实时拦截HTTP请求数据
- 实时编辑HTTP请求数据
  - 手工编辑
  - 基于规则的自动替换
- 内置常用渗透测试攻击向量
  - 可以自行编辑维护



- 基于字典的Web表单认证暴力破解
  - GET
  - POST
  - 不支持CAPTCHA机制



- SQL注入
  - 内置多种编码方式支持
    - 数据库编码相关函数
    - 攻击向量混淆相关函数
- XSS
  - 内置多种编码方式支持
    - 攻击向量混淆相关函数



# Live HTTP Headers

---

- HTTP通信数据抓包
  - 支持自定义过滤规则
- HTTP通信抓包数据重放
  - 支持编辑后重放
  - GET / POST



# GroundSpeed

---

- Web表单认证傻瓜破解辅助
  - 自动显示所有表单元素
  - 快速编辑任意表单元素
    - 节点元素/属性的编辑、删除



# 移动Web应用渗透

- XHTML Mobile Profile
  - 增加桌面版Firefox的移动站点页面支持
    - 增加application/vnd.wap.xhtml+xml识别和支持
- wml browser
  - 告诉Web站点当前浏览器支持WML页面渲染
- Modify Headers
  - 自定义规则自动替换HTTP请求头
    - 篡改User-Agent，伪装任意浏览器/客户端



# 实验讲解

- 实验一：Web漏洞利用
  - 基于WebGoat
  - 基于定制开发的弱点应用程序
- 实验二：系统漏洞利用
  - 基于Metasploit（观看录像）



## 参考文献(1/3)

---

- 文件类型头标识指纹库 [http://www.garykessler.net/library/file\\_sigs.html](http://www.garykessler.net/library/file_sigs.html)
- Web漏洞知识库 by 扫啊科技 <http://www.saoatech.com/knowledge.html>
- Advanced SQL Injection [http://www.owasp.org/images/7/74/Advanced\\_SQL\\_Injection.ppt](http://www.owasp.org/images/7/74/Advanced_SQL_Injection.ppt)
- OWASP Testing Guide v3 [https://www.owasp.org/images/5/56/OWASP\\_Testing\\_Guide\\_v3.pdf](https://www.owasp.org/images/5/56/OWASP_Testing_Guide_v3.pdf)
- OWASP Top Ten Project [https://www.owasp.org/index.php/Category:OWASP\\_Top\\_Ten\\_Project](https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)



## 参考文献(2/3)

- Web安全思维导图 [http://huangwei.me/works/  
WebSecInDepth2.0.html](http://huangwei.me/works/WebSecInDepth2.0.html)
- nginx文件类型错误解析漏洞 [http://www.80sec.com/nginx-  
securit.html](http://www.80sec.com/nginx-securit.html)
- CVE漏洞统计信息 <http://www.cvedetails.com/>
- MySpace Samy蠕虫技术分析 [http://namb.la/popular/  
tech.html](http://namb.la/popular/tech.html)
- 社会工程学案例剖析 by 冷漠 [http://www.lengmo.net/post/  
404/](http://www.lengmo.net/post/404/)
- 新浪微博被攻击完整技术分析 [http://site.douban.com/widget/  
notes/2313210/note/158756925/](http://site.douban.com/widget/notes/2313210/note/158756925/)



## 参考文献(3/3)

---

- 关于 Discuz! 后台信息通知系统域名被劫持部分站点受到影响的说明 [http://  
www.discuz.net/thread-1184256-1-1.html](http://www.discuz.net/thread-1184256-1-1.html)
- 渗透测试标准PTES [http://www.pentest-  
standard.org/index.php/Main\\_Page](http://www.pentest-standard.org/index.php/Main_Page)
- How Hackers Target and Hack Your Site  
[http://resources.infosecinstitute.com/hacking-  
a-wordpress-site](http://resources.infosecinstitute.com/hacking-a-wordpress-site)



## 课后思考题

- XSS与CSRF有什么区别与联系？
- SQL注入的全面防御方案应如何设计
- 文件上传漏洞如何测试？全面防御方案应如何设计
- 调研主流WEB应用开发框架在代码层面做了哪些常见漏洞类型的防御，具体是如何实现的，历史版本中出现过哪些防御机制被突破的例子