java.net

# Java程序设计

## 第05-02讲 安全（security）

Java课程组

# 本讲教学目标


- 理解JavaWeb应用中认证与安全的概念
- 掌握JavaWeb安全的技术



# 本讲内容

- 基本概念
- 认证机制
- 安全声明
- 安全编程

# 安全问题的产生


- 互联网作为商业交易的工具快速发展，越来越多的公司提供网络交易服务。很多商业活动在网上进行
- 当前，成千上万的网民在网上进行各种活动的同时，也在网上传递其个人信息
- 每天在网上发生各种各样的商业活动，如银行交易、股票交易等
- 为了支持这些应用，我们需要一个健壮、安全的互联网保障机制。电子商务没有安全的保障是不可能的

# 基本概念


- 随着公司、个人对其资源和隐私的重视度的增加，网络安全的重要性也日益突出
- Servlet规范提供了方法和途径来实现Web应用程序的安全，在讨论实现安全特性之前，先了解以下与安全相关的基本概念
  - 认证
  - 授权
  - 数据完整性
  - 数据私密性
  - 审核
  - 恶意代码
  - 网站攻击

# 认证(authentication)


- 安全的第一个基本要求就是用户认证
- 认证是一个鉴别用户、确认身份的过程，这意味着校验用户是否是其所宣称的身份
- 例如：用户名和密码
  - 是否存在该用户
  - 提供的用户名是否与其密码相符



# 授权(authorize)


- 用户认证通过，必须被授权
- 授权是一个决定用户是否允许访问特定资源的过程
- 例如：普通用户不可以进行后台管理功能
- 授权通常被维护在一个访问控制列表(ACL)中，这个列表指定了用户及其可以访问的资源



# 数据完整性


- 数据完整性是一个确保数据从发送端到接收端不受到损害的过程
  - 例如：如果用户发送一个从其账户转出1000元的请求，银行系统应该确保转出的金额是1000元而不是10000元
- 数据完整性通常由伴随数据一起发送的一个数字签名来保证。在接收端，数据签名获得校验



# 数据私密性


- 数据私密性是确保只有数据合法访问者可以访问敏感信息的过程
  - 如：用户发送用户名和密码登录网站时，如果这些信息以原始格式发送，在互联网上传输时，网络黑客完全可能通过监听HTTP数据包窃取这些敏感信息。在这种情况下，数据无法保证其机密性



# 数据私密性


- 数据的私密性一般是通过数据加密来实现的，这样只有合法用户才能解密获取的信息。大多数网站使用HTTPS协议来加密数据
- 认证和数据私密性之间的不同在于信息是受保护的。认证是防止访问未授权的信息，而数据私密性是确保即使信息落入到非法用户手中也无法获取信息



# 审核


- 审核是记录系统中与安全相关的事件，确保对每个用户的行为有据可查
- 审核能追踪到系统发生安全问题的原因，通常是由应用程序产生的日志文件来完成



# 恶意代码


- 将引起计算机系统损害的一段代码称作恶意代码
  - 典型的恶意代码包括病毒、蠕虫和木马等
  - 有时系统开发者在编写程序时会留下一个后门漏洞，这是一个潜在的误用机会，也称为恶意代码
- 尽管我们不能防止不知名程序员的恶意代码，但对于内部程序员的一对一审查则可以杜绝系统后门漏洞



# 网站攻击


- 任何认为有价值的事物均是潜在受攻击目标，应受到保护。网站是最易遭到攻击的目标。网站的价值在于其所包含的信息以及其给合法用户提供的服务
- 一般存在3种类型的网站攻击
  - 安全攻击
  - 伪装攻击
  - 服务攻击



# 网站攻击


- 一般存在3种类型的网站攻击
  - 安全攻击：通过监听两台机器间的通信来窃取机密信息，通过加密传输数据可防止此类攻击，如使用HTTPS协议进行信道加密
  - 伪装攻击：通过改变传输的信息来达到恶意企图。如伪装IP地址，使用使服务器IP地址发消息给服务器。通过健全的认证机制可以防止此种攻击
  - 服务攻击：发送大量假请求使得系统无法有效处理合法的请求，这种伪造的大量假数据包可以使网络堵塞。通过使用防火墙限制端口和控制网络通信量，可以防止此类攻击



# 本讲内容

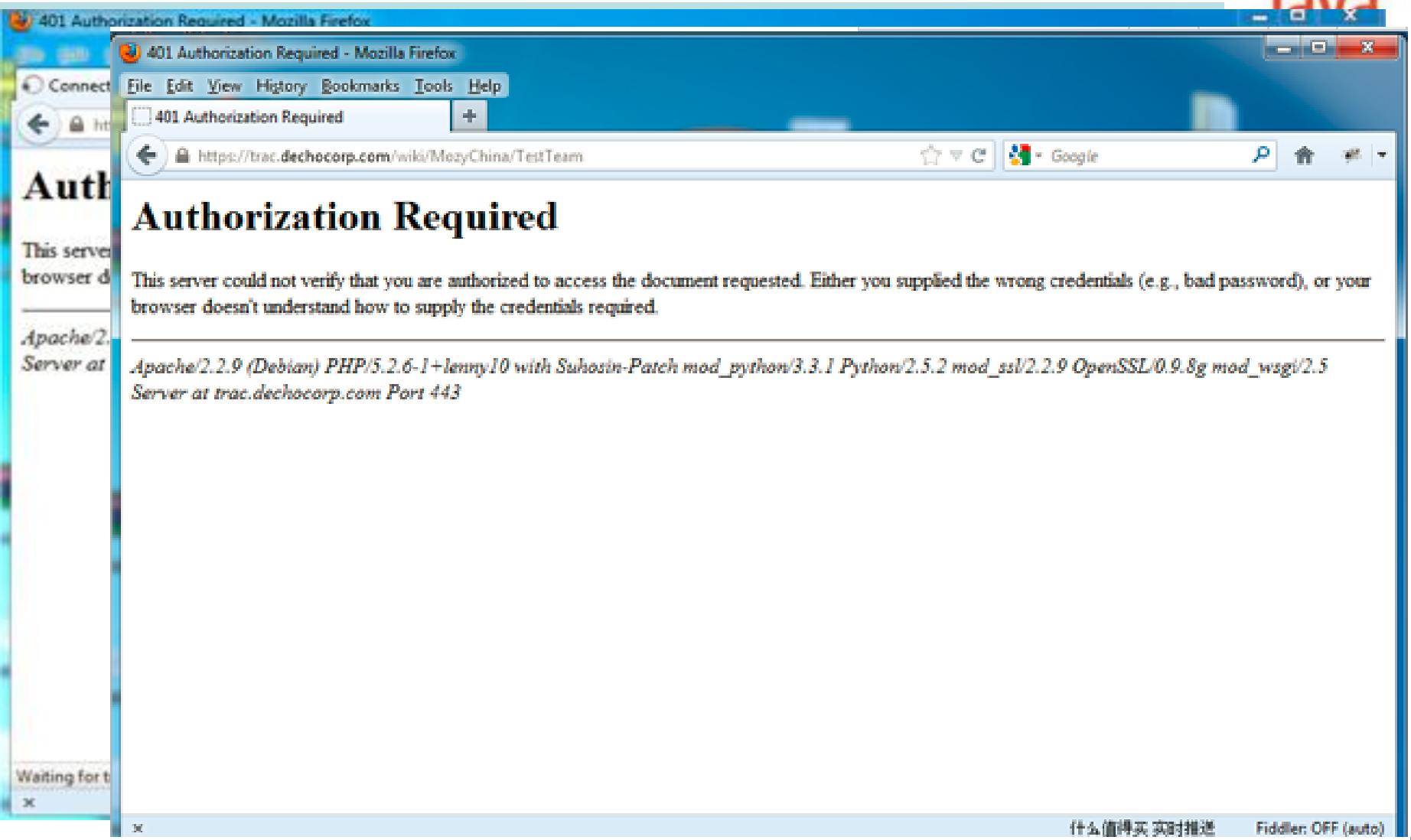
- 基本概念
- 认证机制
- 安全声明
- 安全编程

# Servlet规范的认证机制


- Servlet规范定义了4种认证机制：
  - HTTP基本认证
  - HTTP摘要认证
  - HTTPS客户端认证
  - HTTP表单认证
- 以上4种认证都是基于用户名/密码机制，由服务器维护一个所有用户名和密码的列表，并且保护该列表资源



# HTTP基本认证


401 Authorization Required - Mozilla Firefox
File Edit View History Bookmarks Tools Help
401 Authorization Required
https://trac.dechocorp.com/wiki/MozyChina/TestTeam
Google
Authorization Required
This server could not verify that you are authorized to access the document requested. Either you supplied the wrong credentials (e.g., bad password), or your browser doesn't understand how to supply the credentials required.
Apache/2.2.9 (Debian) PHP/5.2.6-1+lenny10 with Suhosin-Patch mod_python/3.3.1 Python/2.5.2 mod_ssl/2.2.9 OpenSSL/0.9.8g mod_wsgi/2.5
Server at trac.dechocorp.com Port 443
Fiddler: OFF (auto)

# HTTP基本认证过程


- 浏览器发送一个访问受保护资源的请求
- 服务器观察到请求的资源是受保护的，于是服务器发送一个代表未授权的401代码到客户端
- 在接收到上述响应后，浏览器会打开一个对话框，提示用户输入用户名和密码
- 用户输入了用户名和密码，浏览器再次发送请求，并在名为Authorization的头信息中传递值(Base64编码)
- 服务器接收该请求，检验用户名和密码，如果合法则发送请求资源给客户端。否则再次发送代表未授权的401代码到客户端



# HTTP基本认证示例


- Tomcat的Manager站点使用的就是HTTP基本认证的方式
- 在web.xml中配置安全相关信息

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>protected Resource</web-resource-name>
    <url-pattern>/BasicVerify/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>role1</role-name>
  </auth-constraint>
</security-constraint>
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Default</realm-name>
</login-config>
<security-role>
  <description>this is a user</description>
  <role-name>role1</role-name>
</security-role>
```



# HTTP基本认证的优缺点


- HTTP基本认证的优点
  - 非常容易构建
  - 所有浏览器均支持
- HTTP基本认证的缺点
  - 安全性不能保证，因为用户名和密码未加密(注意：Base64编码不是加密方法。sun.misc.Base64和sun.misc.Base64Decoder类可对字符串编码和解码)
  - 无法定制与应用程序相匹配的用户名和密码对话框外观



# HTTP摘要认证


- HTTP摘要认证是在基本认证的基础上密码是加密发送
- HTTP摘要认证的优点
  - 比HTTP基本认证安全
- HTTP摘要认证的缺点
  - 仅由微软的IE浏览器支持。因为规范没有对HTTP摘要认证强制，所以许多Servlet容器不提供支持



# HTTPS客户认证


- HTTPS就是构建在安全套接字(Secure Socket Layer,SSL)之上的HTTP
  - SSL是网景公司开发的，用于确保互联网上传输敏感数据私密性的协议。所有被传输的数据采用公开密钥的加密方式
- HTTPS客户认证的优点
  - 是4种认证中最安全的
  - 被所有浏览器支持
- HTTPS客户认证的缺点
  - 需要认证中心颁发的认证书
  - 实现、维护该认证具有较高成本



# HTTP表单认证


- HTTP表单认证与基本认证很相似。但未使用浏览器弹出对话框，而是使用一个HTML的表单来获取用户名和密码。
- 开发都必须创建一个包含表单的HTML页面，并可以定制表单外观。Form表单的唯一要求就是其action属性必须取值为j_security_check，并提供两个文本域用于输入用户名和密码(j_username, j_password)，除此强制要求外，其他事务均可由开发者自定义
- HTTP表单认证的优点
  - 非常容易建立
  - 所有浏览器均支持
  - 可以定制登录窗体的外观
- HTTP表单认证的缺点
  - 不安全，因为用户名和密码未加密
  - 只有在使用Cookie维持会话时，才可以使用



# 定制认证机制


- 在web.xml中定义认证机制
- 在指定具体认证用户前，要配置用户名和密码，这个步骤取决于具体的Servlet容器
- Tomcat服务器是在下面的文件中：<tomcat-root>/conf/tomcat-users.xml

```
<tomcat-users>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="role1" password="tomcat" roles="role1"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
</tomcat-users>
```

# 定制认证机制


- 权限分配给角色而不是实际用户

```
<tomcat-users>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="role1" password="tomcat" roles="role1"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="john" password="jjj" roles="employee"/>
  <user username="mary" password="mmm" roles="employee"/>
  <user username="bob" password="bbb" roles="employee,supervisor"/>
</tomcat-users>
```

# 定制认证机制


```
<login-config>
        <auth-method>FORM</auth-method>
        <form-login-config>
                <form-login-page>/formlogin.html</form-login-
page>
                <form-error-page>/formerror.html</form-error-page>
        </form-login-config>
</login-config>
```

```
<login-config>
        <auth-method>BASIC</auth-method>
</login-config>
```

# 本讲内容

- 基本概念
- 认证机制
- 安全声明
- 安全编程

# 安全声明


- 开发、部署、使用Web应用程序的通常是不同的人，这就要求开发者有能力非常容易的将Web应用安全需求移交给部署者，部署者也应该有能力定制安全的各个方面而不需要修改代码
- Servlet容器允许我们在Web部署描述符中配置安全需求的细节

# JSP授权方式


- 默认情况下，Web应用程序的所有资源允许被任何人访问
- 为了对资源进行保护，可以做以下工作来限制对资源的访问
  - Web资源集合：鉴别必须受保护的资源
  - 授权限制：鉴别用户所分配的角色
  - 用户数据限制：指定发送者和接受者之间传输数据的方式

# web.xml的<security-constraint>

```
<security-constraint>
        <web-resource-collection>
                <web-resource-name>declarative security test</web-resource-name>
                <url-pattern>/secure</url-pattern>
                <http-method>POST</http-method>
        </web-resource-collection>
        <auth-constraint>
                <role-name>supervisor</role-name>
        </auth-constraint>
        <user-data-constraint>
                <transport-guarantee>NONE</transport-guarantee>
        </user-data-constraint>
</security-constraint>
<login-config>
        <auth-method>FORM</auth-method>
        <form-login-config>
                <form-login-page>/formlogin.html</form-login-page>
                <form-error-page>/formerror.html</form-error-page>
        </form-login-config>
</login-config>
<security-role>
        <role-name>supervisor</role-name>
</security-role>
```

# 本讲内容

- 基本概念
- 认证机制
- 安全声明
- 安全编程

# 安全编程


- 有时候仅采用声明的安全机制是不够的
- 例如：假设允许一个Servlet被公司所有员工访问，但是，针对管理层和普通员工该Servlet产生不同的输出。这在种情况下，Servlet规范允许Servlet拥有处理安全的代码，Servlet根据用户所扮演的角色产生相应的输出



# 安全相关API


- HttpServletRequest接口提供了3个方法用于识别用户和角色
  - String getRemoteUser()
    - 如果用户通过认证，该方法返回用户的登录名称。否则返回null值
  - Principal getUserPrincipal()
    - 该方法返回一个包含认证通过用户的java.security.Principal对象
  - Boolean isUserInRole(String rolename)
    - 该方法用于判断用户是否是指定的角色

# 小结

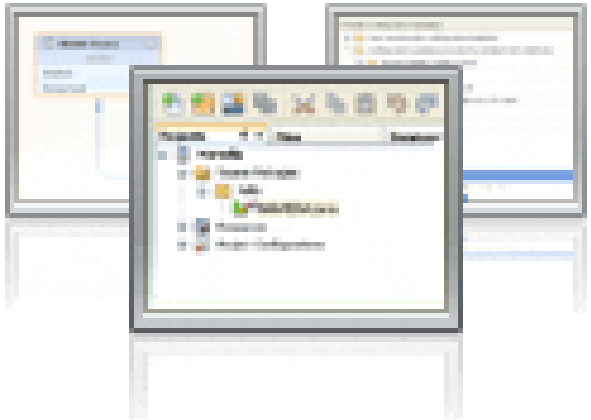

- 了解与安全相关的一些基本概念：认证就是鉴别用户，授权就是鉴别用户可以做什么，审核是记录用户的行为…
- Servlet规范定义了4种认证机制
- Servlet规范中安全声明与安全编程的概念与应用



# 实验


- 实验





# 本讲结束

- 谢谢大家