安全编程

2019年10月16日

11:24

**SQL注入**

**定义**:把SQL命令插入到Web表单中提交或者插入到页面请求的查询字符串中，最终达到欺骗服务器执行恶意的SQL命令

**目的**:使服务器执行恶意的SQL命令

**防止手段**:1->限制前端输入类型

2->采用PreparedStatement访问数据库

**跨站脚本攻击(XSS)**

**攻击形式:**在网页中嵌入脚本,比如JS代码

**目的**:常用来获取用户的cookie,导航至恶意网站

**防止手段:**1->对文本框均做了输入验证

例子:d[this.name] = HtmlUtil.htmlEncode(this.value);

2->对用户输入的内容进行了替换

**前台**:d[this.name] = HtmlUtil.htmlEncode(this.value);

var HtmlUtil = {

/\*用浏览器内部转换器实现html转码\*/

htmlEncode:function (html){

//1.首先动态创建一个容器标签元素，如DIV

var temp = document.createElement ("div");

/\*\*2.然后将要转换的字符串设置为这个元素的innerText(ie支持)或者         textContent(火狐，google支持)\*/

(temp.textContent != undefined ) ?

(temp.textContent = html) : (temp.innerText = html);

//3.最后返回这个元素的innerHTML，即得到经过HTML编码转换的字符串了

var output = temp.innerHTML;

temp = null;

return output;

}

}

**后台:**

var HtmlUtil = {

/\*用浏览器内部转换器实现html转码\*/

htmlEncode:function (html){

//1.首先动态创建一个容器标签元素，如DIV

var temp = document.createElement ("div");

/\*\*2.然后将要转换的字符串设置为这个元素的innerText(ie支持)或者         textContent(火狐，google支持)\*/

(temp.textContent != undefined ) ?

(temp.textContent = html) : (temp.innerText = html);

//3.最后返回这个元素的innerHTML，即得到经过HTML编码转换的字符串了

var output = temp.innerHTML;

temp = null;

return output;

}

}

**跨站点请求伪造(CSRF):**

**攻击形式**:伪造用户请求,向网站发起请求攻击

**攻击目的**:通过伪造用户请求篡改网站内容,数据库等

**防止手段:**1->验证码

CSRF在用户不知情的情况下构造了网络请求，使用验证码就强制用户必须和应用进行交互

2->CSRF Token

为每个session创建唯一的随机字符串并在受理请求时验证

**文件上传漏洞:**

攻击形式:用户上传了一个可执行的脚本文件，并通过此脚本文件获得了执行服务器端命令的能力

**攻击目的:**上传木马文件等实现用户自动下载

**防止手段:**1->JavaScript中判断文件类型，只允许特定的文件形式上传，尽量使用白名单形式

2->后台仍需判断文件类型，只允许特定的文件形式上传，尽量使用白名单形式

3->文件应上传至不可访问路径，利用随机数修改文件名

**用户登录验证:**

**系统信息泄露:**

<!--获取系统用户数据-->//在控制台上显示,容易泄露信息

<%//获取系统用户数据%>//不再控制台上显示,不容易泄露信息

**JS文件混淆:**

**网页被框架盗链或者点击劫持**

**防止手段:**1->在每个网页上增加如下脚本来禁止iframe嵌套

<script>

//js 判断顶层窗口跳转

if(top != self) top.location.href = location.href;

</script>

2->通过header控制

response.addHeader("x-frame-options","DENY");

  return "login";

**共享的全局变量:**采用局部变量代替全局变量

**限制变量的访问:**使用取值方法而不是 public 变量

**不允许扩展的类和方法应该声明为 final** 。这样做防止了系统外的代码扩展类并修改类的行为。

**敏感数据的表单提交**:禁止使用 HTTP-GET 方法提交带有敏感数据的表单（form）。

**安全编程的原则:**

* 程序只实现你指定的功能
* 永远不要信任用户输入，对用户输入数据做有效性检查
* 对数据的校验不能仅仅停留在前端，要尽可能的在后台加入校验
* 必须考虑意外情况并进行处理，不要试图在发现错误之后继续执行
* 小心、认真、细致地编程

**PowerDesigner:**

基于Entity-Relation的模型,分别从概念数据模型,逻辑数据模型,物理数据模型三个层次对数据库进行设计

遵循三大范式:

* 第一范式:要求属性具有原子性,不可再分解
* 要求记录有唯一标识,即实体的唯一性
* 任何字段不能由其他字段派生出来,要求字段没有冗余

注意:为提高效率,需要适当保留冗余数据,具体做法:在概念数据模型设计的时候遵循第三范式,降低范式标准在进行物理数据模型设计的时候考虑

降低范式:增加字段,允许冗余

**文件锁定**

* 当我们进行一些重要的复杂的修改时,为了避免出现冲突而无法正常提交的情况,可以采用文件锁定的方式来进行操作