WEB安全基础

1 WEB基础

HTTP协议：超文本传输协议

统一资源标示符：URL

HTTP协议采用了**请求/响应**模型。

HTML：超文本标记语言

- Head：头 关于网页的信息
- body：主体 关于网页的具体内容

标记一般都是成对的出现的，<html>为我们的开头，然后</html>就是我们的结尾

（部分标记除外例如：
）

```
<head></head>
```

这2个标记符分别表示头部信息的开始和结尾。**头部中包含的标记是页面的标题、序言、说明等内容，它本身不作为内容来显示，但影响网页显示的效果**。头部中最常用的标记符是标题标记符和meta标记符，其中**标题标记符用于定义网页的标题，它的内容显示在网页窗口的标题栏中**，网页标题可被浏览器用作书签和收藏清单。

JavaScript：是一种嵌入在我们的html语言中的脚本语言，可以提供动态功能

```
<html>
        <head>
              <title>Javascript简单示例</title>
        </head>
        <body>
              <script language="javascript">
                   alert("第一个javascript");
             </script>
        </body>
</html>
```

有它自身的**基本数据类型，表达式和算术运算符及程序的基本程序框架**

2 WEB编程环境

WEB静态语言和WEB动态语言

- WEB静态语言就是通常所见到的超文本标记语言（标准通用标记语言下的一个应用）
- WEB动态语言主要是ASP、PHP、JAVASCRIPT、JAVA、CGI等计算机脚本语言编写出来的执行灵活的互联网网页程序。

3 PHP语言

PHP:

- 是一种**解释性语言**。PHP的语法和C/C++，Java，Perl，ASP，JSP有相通之处并且加上了自己的语法。
- 由于PHP是一种**面向HTML的解析语言**，所以，PHP语句被包含在PHP标记里面，PHP标记外的语句都被直接输出。包括在PHP标记中的语句被解析，在其外的语句原样输出并且接受PHP语句的控制

标记	解释	示例
<?php ?>	标准php标记	<?php echo $variable; ?>
<script language="php"> </script>	长标记	<script language="php"> echo $vaiable; </script>
<? ?>	短标记	<? echo $variable; ?>
<% %>	仿ASP	<%=$variable;%>

变量解析当遇到符号（$）时产生，**解析器会尽可能多地取得后面的字符以组成一个合法的变量名，然后将变量值替换他们**，如果$后面没有有效的变量名，则输出"$"。**如果想明确的变量名可以用花括号把变量名括起来**。

4 HTTP会话管理

HTTP协议属于**无状态的通信协议**

当浏览器发送请求给服务器的时候，服务器响应，但是当同一个浏览器再发送请求给服务器的时候，他不知道你就是刚才那个浏览器。简单地说，就是服务器不会去记得你

HTTP是短连接的

为了识别不同的请求是否来自同一客户，需要引用HTTP会话机制：

多次HTTP连接间维护用户与同一用户发出的不同请求之间关联的情况称为维护一个会话（session）。

Cookie与session是与HTTP会话相关的两个内容，其中**Cookie存在在浏览器，session存储在服务器中**

Cookies是服务器在本地机器上存储的小段文本并随每一个请求发送至同一个服务器

- 正统的cookie分发是通过扩展HTTP协议来实现的，服务器通过在HTTP的响应头中加上一行特殊的指示以提示浏览器按照指示生成相应的cookie。
- 浏览器检查所有存储的cookie，如果某个cookie所声明的作用范围大于等于将要请求的资源所在的位置，则把该cookie附在请求资源的HTTP请求头上发送给服务器。

cookies的内容：**名字，值，过期时间，路径和域**

若不设置过期时间，则表示**这个cookie的生命期为浏览器会话期间**，关闭浏览器窗口，cookie就消失。这种生命期为浏览器会话期的cookie被称为**会话cookie**。

会话cookie一般不存储在硬盘上而是保存在内存里

- 若设置了过期时间，浏览器就会把cookie保存到硬盘上
- 存储在硬盘上的cookie可以在不同的浏览器进程间共享，比如两个IE窗口

session id的值应该是一个既不会重复，又不容易被找到规律以仿造的字符串

5 HTTP请求

method属性指定了与服务器进行信息交互的方法为POST。

交互的四种方法：

- GET：GET一般用于获取/查询资源信息
- POST：而POST一般用于更新资源信息
- DELETE
- PUT

以?分割URL和传输数据，参数之间以&相连，如：`login.action?name=sean&password=123`

- GET请求的数据会附在URL之后
- POST把提交的数据则放置在是HTTP包的包体中

POST的安全性要比GET的安全性高：

- GET模式下，通过URL就可以作数据修改
- GET模式下，用户名和密码将明文出现在URL上，因为登录页面有可能被浏览器缓存、其他人查看浏览器的历史纪录，那么别人就可以拿到你的账号和密码了
- GET模式下，提交数据还可能会造成跨站请求伪造攻击

6 PHP连接数据库

表userinfo：有两个字段

- username
- pwd

数据库应用开发三步骤：

- 连接数据库
  - `$conn=mysql_connect("localhost", "root", "123456");`
- 执行SQL操作
  - `$result=mysql_db_query("MyDB", $SQLStr, $conn);`

- 关闭连接
  - `mysql_free_result($result); mysql_close($conn);`

7 Cookie实战

cookie和session的关系：

- 共性：都可以暂时保存在多个页面中使用的变量。
- 区别：cookie存放在客户端浏览器，session保存在服务器。它们之间的联系是session ID一般保存在cookie中，来实现HTTP会话管理

工作原理：

- 生成Cookie：当客户访问某网站时，PHP可以使用setcookie函数告诉浏览器生成一个cookie，并把这个cookie保存在c:\Documents and Settings\用户名\Cookies目录下。
- 使用Cookie：Cookie是HTTP标头的一部分，当客户再次访问该网站时，浏览器会自动把与该站点对应的cookie发送到服务器，服务器则把从客户端传来的cookie将自动地转化成一个PHP变量。

setcookie格式

- setcookie(name, value, expire, path, domain, secure)
- name 必需。规定 cookie 的名称。
- value 必需。规定 cookie 的值。
- expire 可选。规定 cookie 的有效期。

8 十大WEB安全威胁

- 注入
- 跨站脚本
- 遭破坏的身份认证和会话管理
  - 攻击者窃听了用户访问HTTP时的用户名和密码，或者是用户的会话，从而得到sessionID或用户身份信息，进而冒充用户进行HTTP访问的过程
  - 检测是否使用HTTPS的最简单方法就是使用网络嗅探工具
  - **会话劫持就是一种窃取用户SessionID后，使用该SessionID登录进入目标账户的攻击方法，此时攻击者实际上是利用了目标账户的有效Session。如果SessionID是被保存在Cookie中，则这种攻击被称为Cookie劫持。**
  - 如果攻击者窃取了用户的Session，并一直保持一个有效的Session，攻击者就能通过此有效Session一直使用用户的账户，即成为一个永久的“后门”，**这就是会话保持攻击**
- 不安全的直接对象引用
  - **直接对象引用：是指WEB应用程序的开发人员将一些不应公开的对象引用直接暴露给用户，使得用户可以通过更改URL等操作直接引用对象。**

  - **不安全的直接对象引用：是指一个用户通过更改URL等操作可以成功访问到未被授权的内容。比如一个网站上的用户通过更改URL可以访问到其他用户的私密信息和数据等。**

- 伪造跨站请求
- 安全配置错误
  - **默认的用户名密码没有及时修改**
- 不安全的加密存储
  - Web应用系统**没有对敏感性资料进行加密**
- 没有限制的URL访问
- 传输层保护不足和未验证的重定向和转发
  - **中间人攻击**(Man-in-the-middle attack)，即MITM。HTTP连接的目标是Web服务器，如果传输层保护不足，攻击者可以担任中间人的角色，在**用户和Web服务器之间截获数据并在两者之间进行转发**，使用户和服务器之间的整个通信过程暴露在攻击者面前。

WEB渗透实战

1　文件上传漏洞

指网络攻击者上传了一个可执行的文件到服务器并执行。这里上传的文件可以是木马，病毒，恶意脚本或者WebShell等

WebShell就是以asp、php、jsp或者cgi等网页文件形式存在的一种命令执行环境，也可以将其称之为一种网页后门

- 攻击者在入侵了一个网站后，通常会将这些asp或php后门文件与网站服务器web目录下正常的网页文件混在一起，然后使用浏览器来访问这些后门，**得到一个命令执行环境，以达到控制网站服务器的目的**（可以上传下载或者修改文件，操作数据库，执行任意命令等）。
- WebShell后门隐蔽较性高，可以轻松穿越防火墙，访问WebShell时不会留下系统日志，只会在网站的web日志中留下一些数据提交记录，没有经验的管理员不容易发现入侵痕迹。
- 攻击者可以将WebShell隐藏在正常文件中并修改文件时间增强隐蔽性，也可以采用一些函数对WebShell进行编码或者拼接以规避检测。

原理：

- 一些文件上传功能实现代码没有严格限制用户上传的文件后缀以及文件类型，导致允许攻击者向某个可通过Web访问的目录上传任意PHP文件，并能够将这些文件传递给PHP解释器，就可以在远程服务器上执行任意PHP脚本。
- 当系统存在文件上传漏洞时攻击者可以将病毒，木马，WebShell，其他恶意脚本或者是包含了脚本的图片上传到服务器，这些文件将对攻击者后续攻击提供便利。根据具体漏洞的差异，此处上传的脚本可以是正常后缀的PHP，ASP以及JSP脚本，也可以是篡改后缀后的这几类脚本。

2 跨站脚本攻击

跨站脚本攻击与SQL注入攻击区别在于：**XSS主要影响的是Web应用程序的用户，而SQL注入则主要影响Web应用程序自身。**

启用并使用脚本并不是XSS漏洞存在的原因。只有当Web应用程序开发人员犯错误时才会变得危险

分为两类攻击方式：

- 反射式跨站脚本：也称作非持久型、参数型跨站脚本。**主要用于将恶意脚本附加到URL地址的参数中**
- 持久式跨站脚本：比反射式跨站脚本更具有威胁性，并且可能影响到Web服务器自身的安全

两者的异同：

- 存储式XSS与反射式XSS类似的地方在于，会在Web应用程序的网页中显示未经编码的攻击者脚本。
- 它们的区别在于，存储式XSS中的脚本并非来自于Web应用程序请求；相反，脚本是由Web应用程序进行存储的，并且会将其其作为内容显示给浏览用户。

一般来说，存储式XSS的风险会高于反射式XSS。因为存储式XSS会保存在服务器上，有可能会跨页面存在。

从攻击过程来说，反射式XSS一般要求攻击者诱使用户点击一个包含XSS代码的URL链接；而存储式XSS则只需让用户查看一个正常的URL链接

3 SQL注入漏洞

常用的特殊字符：

```
’ ’   字符串指示器(‘string’)
;          语句终结符
||         对于Oracle、PostgreSQL而言为连接（合并）
--        注释（单行）
#         注释（单行）
/**/     注释（多行）
```

SQL注入是一种**将SQL代码插入或添加到应用（用户）的输入参数中的攻击**，之后再将这些参数传递给后台SQL服务器加以解析并执行

联系过程：

当用户通过浏览器向表单提交了用户名“bob”，密码“abc123”时，那么下面的HTTP查询将被发送给Web服务器：

```
http://xxxx.com/xxx.php?user=bob&passwd=abc123
```

当Web服务器收到这个请求时，将构建并执行一条（发送给数据库服务器的）SQL查询。在这个示例中，该SQL请求如下所示：

```
SELECT * FROM table WHERE user=’bob’ and password=’abc123’
```

我们可以选择把我们的后面给注释掉，只需要加上一定的特殊字符就可以了

```
http://xxxx.com/xxx.php?user=bob’--&passwd=xxxxxx
```

这样输出的SQL语句就是：

```
SELECT * FROM table WHERE user='bob'--' and password='abc123'
```

寻找注入点：GET请求的最好被注入

注入方法：

- 单引号法：但是一般都会把单引号过滤掉
  - 输入

    ```
    http://localhost/test.php?id=1'
    ```

  - 输出

    ```
    select * from category where id=1‘
    ```

- 永真永假法：
  - **当与上一个永真式使逻辑不受影响时，页面应当与原页面相同。**
  - **而与上一个永假式时，则会影响原逻辑，页面可能出错或跳转（这与设计者的设计有关）。**

4 SQLMAP

就是由python开发的一款SQL注入的工具

常见语句：

- Sqlmap -u url 找到注入点
- sqlmap -u url --dbs 列出数据库
- 或者 sqlmap -u url --current-db 显示当前数据库
- sqlmap -u url –users 列出数据库用户
- 或者sqlmap -u url --current-user 当前数据库用户
- sqlmap -u url --tables -D “testdb” 列出testdb数据库的表
- sqlmap -u url --columns -T "user" -D "testdb" 列出testdb数据库user表的列
- sqlmap -u url --dump -C “id,username,password” -T “user” -D “testdb” 列出testdb数据 库user表的id,username,password这几列的数据

5 SQL盲注

SQL盲注是不能通过直接显示的途径来获取数据库数据的方法。在盲注中，攻击者根据其返回页面的不同来判断信息（可能是页面内容的不同，也可以是响应时间不同）。

一般情况下，盲注可分为三类：

- 基于布尔SQL盲注
- 基于时间的SQL盲注

- 基于报错的SQL盲注

常用的函数：

- Substr函数的用法：取得字符串中指定起始位置和长度的字符串，默认是从起始位置到结束的子串。
  - 语法为：substr(string, start_position, [length])，比如substr('目标字符串',开始位置,长度)，再如substr('This is a test', 6, 2) 将返回 'is'。
- If函数的用法：如果满足一个条件可以赋一个需要的值。
  - 语法：IF(expr1,expr2,expr3)，其中，expr1是判断条件，expr2和expr3是符合expr1的自定义的返回结果，expr1为真则返回expr2，否则返回expr3。
- Sleep函数的用法：sleep(n)让语句停留n秒时间，然后返回0，如果执行被打断，返回1。
- Ascii函数的用法：返回字符的ASCII码值。

5.1 基于布尔SQL盲注

可以利用输出的bool值来进行判断

二分法可以节省我们的猜测次数

5.2 基于时间的SQL盲注

就是看输入的数字运行时有没有明显的时间延迟，如果有的话说明是正确的语句，因为花费时间去查询了

5.3 基于报错的SQL盲注

就是通过引起报错去猜测我们的数据

6 SQL注入的防御措施

- 在服务端正式处理之前对提交数据的合法性进行检查
  - **在确认客户端的输入合法之前，服务端拒绝进行关键性的处理操作**
- 封装客户端提交信息
- 替换或删除敏感字符/字符串
  - 不一定能成功，攻击者可以输入构造的多次出现的字符串来引发攻击
- 屏蔽出错信息
  - 就是不告诉攻击者盲注的结果，这样就不好攻击了

7　文件包含漏洞

在开发web应用时，开发人员通常会将一些重复使用的代码写到单个文件中，**再通过文件包含，将这些单个文件中的代码插入到其它需要用到它们的页面中**。

- **配置文件。**用于整个web应用的配置信息，如数据库的用户名及密码，使用的数据库名，系统默认的文字编码，是否开启Debug模式等信息。右侧就是wordpress博客系统配置文件的部分内容。
- **重复使用的函数。**如连接数据库，过滤用户的输入中的危险字符等。这些函数使用的频率很高，在所有需要与数据库进行交互的地方都要用到相似的连接数据库的代码；在几乎所有涉及到获取用户输入的地方都需要对其进行过滤，以避免出现像sql注入、xss这样的安全问题。
- **重复使用的版块**。如页面的页头、页脚以及菜单文件。通过文件包含对这些文件进行引入，在某个地方需要修改时，开发人员只需要对单个文件进行更新即可，而不需要修改使用这些板块的其他文件。
- **具有相同框架的不同功能。**开发人员可以在不同的页面引入页头、页脚，也可以在定义好页头、页脚的框架中引入不同的功能。这样有新的业务需求时，开发人员只需要开发对应的功能文件，再通过文件包含引入；在有业务需要更替时，开发人员也只需要删除对应的功能文件即可。

分类：

- 本地文件包含漏洞
  - 如果被包含文件的文件名是从用户处获得的，且没有经过恰当的检测，从而包含了预想之外的文件，导致了文件泄露甚至是恶意代码注入，这就是**文件包含漏洞**。如果被包含的文件储存在服务器上，那么对于应用来说，被包含的文件就在本地，就称之为**本地文件包含漏洞**。
    - **包含上传的合法文件**
    - **包含日志文件**
      - 先构造一条包含恶意代码的请求，这一条请求会被web服务器写入日志文件中
      - 再利用本地文件包含漏洞，将日志文件引入，使得植入的恶意代码得到执行
- 远程文件包含漏洞
  - 如果存在文件包含漏洞，且允许被包含的文件可以通过url获取，则称为远程文件包含漏洞。
    - **包含攻击者服务器上的恶意文件**
    - **通过PHP伪协议进行包含**
      - php://input可以访问请求的原始数据的只读流，也就是通过POST方式发送的内容。借助PHP伪协议，攻击者直接将想要在服务器上执行的恶意代码通过POST的方式发送给服务器就能完成攻击。

PHP带有很多内置URL风格的封装协议，可用于类似fopen()、copy()、 file_exists() 和 filesize() 的文件系统函数，可在include命令中使用。除了这些封装协议，还能注册自定义的封装协议。常见的协议有：

- file:// — 访问本地文件系统
- http:// — 访问 HTTP(s) 网址
- ftp:// — 访问 FTP(s) URLs
- php:// — 访问各个输入/输出流（I/O streams）

- zlib:// — 压缩流
- phar:// — PHP 归档
- **php://filter** 是一种元封装器，设计用于数据流打开时的筛选过滤应用。
- **phar://与zip://**可以获取压缩文件内的内容，如在hack.zip的压缩包中，有一个shell.php的文件，则可以通过phar://hack.zip/shell.php的方式访问压缩包内的文件，zip://也是类似。这两个协议不受文件后缀名的影响

8 反序列化漏洞

序列化是指将对象、数组等数据结构转化为可以储存的格式的过程

要想要将内存中的变量写入磁盘中或是通过网络传输，就需要对其进行序列化操作，序列化能将一个对象转换成一个字符串。

PHP魔术方法：PHP有一类特殊的方法，它们以__(两个下划线)开头，在特定的条件下会被调用

```
__construct()，类的构造函数，创建新的对象时会被调用
__destruct()，类的析构函数，当对象被销毁时会被调用
__call()，在对象中调用一个不可访问方法时会被调用
__callStatic()，用静态方式中调用一个不可访问方法时调用
__get()，读取一个不可访问属性的值时会被调用
__set()，给不可访问的属性赋值时会被调用
__isset()，当对不可访问属性调用isset()或empty()时调用
__unset()，当对不可访问属性调用unset()时被调用。
__sleep()，执行serialize()时，先会调用这个函数
__wakeup()，执行unserialize()时，先会调用这个函数
__toString()，类被当成字符串时的回应方法
__invoke()，调用函数的方式调用一个对象时的回应方法
__set_state()，调用var_export()导出类时，此静态方法会被调用。
__clone()，当对象复制完成时调用
__autoload()，尝试加载未定义的类
__debugInfo()，打印所需调试信息
```

PHP反序列化漏洞又叫PHP对象注入漏洞。

在一个应用中，如果传给unserialize()的参数是用户可控的，那么攻击者就可以通过**传入一个精心构造的序列化字符串**，利用PHP魔术方法来控制对象内部的变量甚至是函数。