20200311课堂笔记

**概述**

教学资源网站：<http://programtrace.com/>

一．有什么安全问题，安全问题产生的原因

分为两个方面：

·**二进制方面**，主要是内存相关问题和逻辑错误问题。有明确的机制，比如缓冲区溢出，空指针，格式化字符串。而逻辑错误多种多样。

·**Web方面**，典型的安全问题，有XSS、SQL注入等。非典型的问题就很多了。主要是一些代码的逻辑错误。

**重点学习缓冲区溢出和XSS**

1. 如何去发现问题

分为白盒分析和黑盒分析两大点。

·**白盒分析**，就是通过分析软件的源代码去寻找问题。这里面，有手工代码分析，就是我们的软件测试技术，和代码review。但是软件的源代码通常比较庞大，手工分析起来很费劲。 所以，又有人研究如何自动化的分析代码。典型的技术呢，就是符号执行。 白盒分析，数据量很大，需要投入很多人工。就会有人想有没有简单的方法。于是，和白盒分析向对立的一种技术，黑盒分析发展起来。

·**黑盒分析**，顾名思义，就是完全不管软件的内部机理，不去看源码，把需要分析的软件当做一个黑盒子。看不见，或者根本不去看内部。不看内部怎么分析呢？看表面现象。对软件来说，就是看输入和输出。给它什么样的输入，得到什么样的输出，再去猜测他内部的机制。黑盒分析法，典型的就是Fuzzing技术，中文叫模糊测试。

1. 如果有人利用这这些安全问题，如何防御
2. **补丁**

软件由于复杂性，不可能没有问题，只要发现了问题及时修补即可。经过多年的发展，形成了一套行之有效的方法。就是通过漏洞数据库来披露和管理各种漏洞，厂家有义务定期发布软件补丁或者更新，软件用户应该及时升级软件。第三方人员，如果发现了安全问题，应该通报给厂家，而不是在漏洞修补以前，去利用漏洞搞破坏，或者不负责任的披露漏洞。

如何去寻找记录攻击的痕迹，然后分析这些数据。数据的来源又分为了 主机层面 网络层面，这就是我们防御系统所做的事情。

攻击的人会进行伪装，不让防御的人那么容易发现，攻击者可能会做哪些方面的伪装，常用的伪装技术有哪些，如何去对抗伪装，也是本课的学习内容。

这方面典型的技术就是加壳脱壳技术和Rootkit技术。

**第一堂课**

**缓冲区溢出和XSS**。XSS虽然比较上层，但是相对比较容易理解。这两个也代表了二进制软件和脚本软件（包括绝大多数Web软件都是脚本软件开发的）两大技术方向。

先了解**二进制软件和脚本软件的区别**。

计算机的底层，是CPU直接执行在内存中的机器指令，对于C和C++这类编程语言开发的软件，存在一个过程，即编译链接，其实就是为了把程序变成CPU可以直接执行的二进制指令。这类软件的一个特点，就是需要直接操作内存。内存是所有在运行态的软件及其数据保存的地方。内存分为细小的单元，每个单元有一个唯一的地址。要访问数据，必须知道数据的地址，要保存新的数据，就必须分配内存，获得可用的地址。但是“地址”本身也是数字，如果不小心计算错误了。那么就会访问到不该访问的数据，就会造成数据的泄露或者破坏。这就是二进制软件安全问题的根源。二进制程序的编程有很大的难度就在于CPU只能做出这样了，因为它是电路，即是物理的东西，不可能设计得机制太复杂。

使用C语言和C++就不可避免的一个东西：直接操作内存，也就是指针。然后在C和C++发展成熟以后，就有人去研究如何降低编程的难度，可不可以避免程序员编程时直接操作内存，把需要操作内存的地方，都封装起来。屏蔽在编程语言的内部。于是发明了脚本语言。

脚本语言，是干脆用C和C++这样的二进制程序开一个软件来执行一种新的程序。也就是用软件来模拟CPU工作。软件的可定制性比CPU高，可以把所有需要操作内存的东西，全部封闭在执行器内部，只给程序员接口，不给程序员操作内存的机会。这就是对象。比如把字符串封装为string对象。只能调用string.len()这样的方法来操作这个对象。这样就避免了由于编程不慎造成的内存相关问题。也降低了编程难度。例如经常看到的python java js这样的程序，都有一个二进制程序的执行器。比如python.exe java.exe Web浏览器等。这些脚本程序的执行器，都是二进制程序。但是，虽然这些脚本程序没有了内存相关问题，也引出了其他的问题。

比如XSS的问题。就是web程序，存在一种高交互性。

**实验部分**

编写一个简单的html文件，这个文件只要有一个表单，用户可以在表单中输入数据，向服务器提交。

代码：

<html>

<body>

<form method="post">

<input type="text">

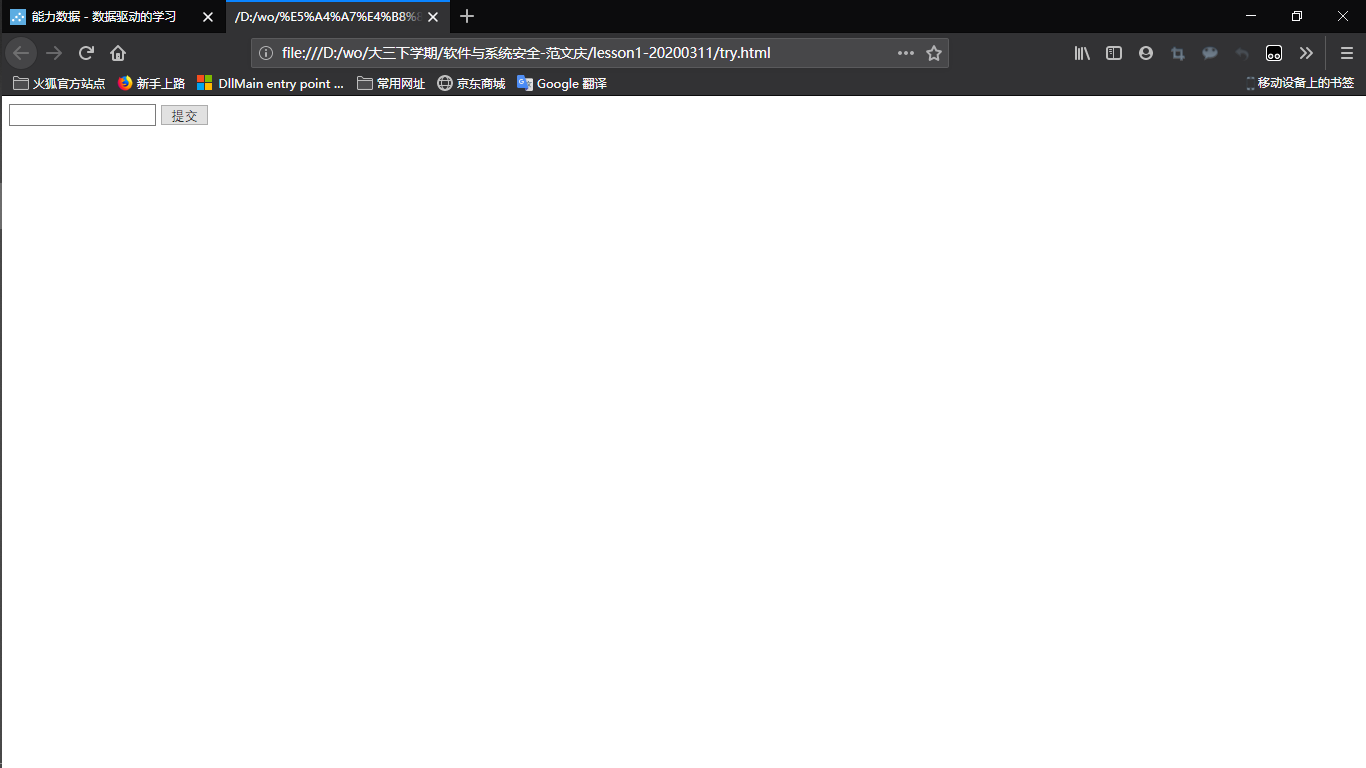
<button>提交</button>

</form>

</body>

</html>

保存为.html文件，然后用浏览器打开后：



搭建一个web服务器，再有一些简单的处理过程，就可以重现xss了。如果这个html是放在web服务器上，用户输入了数据，点击提前，浏览器就会把编辑框中的数据封装为一个POST请求，发现服务器。服务器会把这个数据发给后端脚本来处理。可以通过定义 form的属性来指明需要哪个脚本文件来处理。比如PHP程序，他有一个POST超级变量，当用户提交了数据以后，对应的php脚本的post变量就是用户提交的数据。假设服务器现在把用户提交的数据放在user\_input.html的body标签中。然后保存在服务器文件的根目录中。当有网站的用户访问 http://xxxx.com/user\_input.html的时候。就会看到刚才哪个表单用户提交的内。当然实际的情况是这两个用户可能不是同一个用户，于是A用户提交的内容B用户就访问到了。当服务器脚本是原封不动的把用户输入的数据写到html里时，如果用户提交的数据中包括<script>标签，就会被执行。然后大家需要简单的学一下js语言。比如alert函数，弹出一个消息框。既然能执行alert函数，就能执行其他功能，比如给 window.location.href赋值，让用户莫名其妙的跳转到另外一个网站。