

软件与系统安全

2020年3月24日 20:22

软件安全共包含三方面的问题

1、有什么安全问题，安全问题产生的原因

①二进制方面：主要是内存相关问题和逻辑错误问题

内存相关问题有明确的机制，比如缓冲区溢出、空指针、格式化字符串
逻辑错误多种多样

②Web方面：有典型问题和非典型问题

典型问题有XSS、SQL注入等

非典型问题多种多样

2、如何去发现问题（安全漏洞）

①白盒分析：通过分析软件的源代码去寻找问题。这里包括手工代码分析（代码review、软件测试技术）
但是软件的源代码通常比较庞大，手工分析起来很费劲

缺点：数据量大，需要投入很多人工。

②黑盒分析：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。

对软件来说，就是看输入和输出，由此猜测内部机制。

一种典型的黑盒分析法：Fuzzing技术，中文叫模糊测试。

3、如果有人利用这些安全问题，如何防御

①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就是好样的，经过多年的发展，大家终于形成了一套行之有效的办法，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。

②寻找记录攻击的痕迹，然后分析这些数据。

数据的来源分为主机层面和网络层面，这就是我们入侵防御系统所做的事情。

当然攻击者会进行伪装，本课程主要研究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。

这方面，典型的技术就是加壳脱壳技术和Rootkit技术。

所以，就有人研究如何自动化的分析代码，典型的技术就是符号执行。

但是难免会有人利用这些漏洞，所以在其他方面也需要防御机制。

一、缓冲区溢出和xss

- 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，C和C++这类编程语言开发的软件有一个过程叫编译链接，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。所有要访问的数据，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。地址也是数字，如果计算错误，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。

使用C语言和C++就不可避免直接操作内存，也就是使用指针。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，因此就发明了脚本语言。

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。但是软件的可定制性比CPU就高多了，可以想定义什么指令就定义什么指令。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。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。

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。

所有大家看到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，又引出了其他的问题。比如XSS的问题，就是web程序，存在一种高交互性。web是互联网时代的软件的基本框架，所以一定会有用户提交数据。当初为了网页动态的需求，开发了网页的前端脚本，比如js。直接把脚本嵌入到网页中。浏览器只要发现了script标签，就去当脚本来执行。把网页按照程序员的定制，变的丰富多彩，变得富于变化。但是，恰恰另外一种需求，就是UGC软件，所谓用户产生内容。也就是网页的内容来自于用户提交的内容，这种软件已经非常常见了。比如BBS、博客、微博，电商视频网站的用户评论，都会涉及到用户提交的内容在页面上呈现。

这两种机制，放在一起就产生了神奇的效果。当用户提交的内容里含有脚本呢？如果直接将用户提交的内容放在页面上，那么用户提交的内容中的脚本会不会被浏览

器解析执行呢？
那么一个用户提交了一个脚本就可以在这个页面的所有用户主机上执行呢？用户能提交程序执行了，怎么才能不保证这个程序不是恶意的呢？

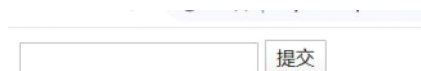
要知道，前端脚本，除了渲染页面元素这样的功能，还有获得用户的输入跳转页面到其他地址等等丰富的功能。只要能执行，就能干很多事情。

二、搭建一个Web服务器，重现XSS

<html>
<body>

```
<form method="post">
  <input type="text">
  <button>提交</button>
</form>
</body>
</html>
```

这个保存为.html文件，head标签里通过meta指定，要注意网页的编码方式。然后用浏览器打开就是这个样子。



下面搭建一个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，点击提交，浏览器就会把编辑框中的数据封装为一个POST请求，发给服务器。服务器会把这个数据发给后端脚本来处理。

你可以通过定义 from 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，就会看到刚才那个表单用户提交的内容。

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。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。

最简单的实验环境，就是在vscode中，安装一个php插件，然后编写一个简单的php脚本，调试运行这个脚本。F5 vscode会自动选择脚本运行的方式，把用户的表单输入写入到html文件。再通过浏览器访问这个html文件，这就是一个最简单的xss运行环境了。

实际的xss漏洞可能很复杂，比如还会有数据库，登录等等，但是万变不离其宗，基本原理都是这样。