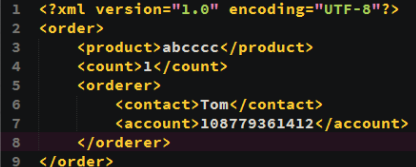
新浪微博：@讨厌被4bet的沙豆

XML的全称是eXtensible Markup Language，意思是可扩展的标记语言，它是标准通用标记语言（Standard Generalized Markup Language,SGML）的一个子集。

在80年代早期，IBM提出在各文档之间共享一些相似的属性，例如字体大小和版面。IBM设计了一种文档系统，通过在文档中添加标记，来标识文档中的各种元素，IBM把这种标识语言称作通用标记语言（Standard Generalized Markup Language,SGML），即GML。经过若干年的发展，1984年国际标准化阻止（ISO）开始对此提案进行讨论，并于1986年正式发布了为生成标准化文档而定义的标记语言标准（ISO 8879），称为新的语言SGML，即标准通用标记语言。

一个XML文件示例：



一个XML文件由XML声明、文档类型定义和文档元素组成。

那么什么是文档类型定义呢？文档类型定义，也叫DTD(Document Type Definition)，可定义合法的XML文档构建模块。它使用一系列合法的元素来定义文档的结构，DTD 可被成行地声明于 XML 文档中，也可作为一个外部引用。

DTD文档的声明及引用

内部DTD文档

<!DOCTYPE *根元素* [*定义内容*]>

外部DTD文档

<!DOCTYPE *根元素* SYSTEM "*DTD文件路径*">

内外部DTD文档结合

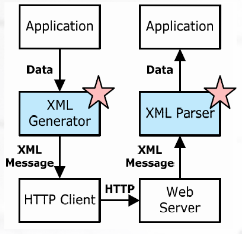
<!DOCTYPE *根元素* SYSTEM "*DTD文件路径*" [

*定义内容*

-]>

XML被设计用来存储和传输数据，任何平台上的程序都可以通过使用XML解析器来处理XML数据，XML的使用范围非常广泛，不仅仅是web应用，还包括数据库软件、浏览器等等。既然XML的覆盖面这么广，那么如果出现安全问题也是相当可怕的。而且很多软件都是使用的同一款XML解析库，如果这套解析库存在漏洞，那么无疑又将影响的范围扩大了。

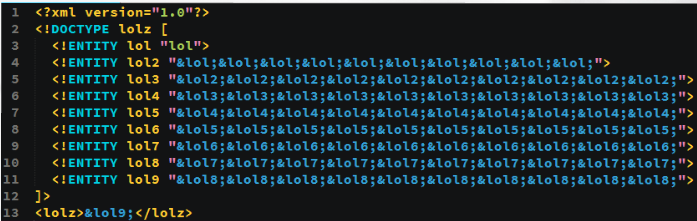
下面这张图很清晰的描述了程序在处理XML时容易出现问题的点，我们逐步来看：应用先将原始数据交给XML生成器，生成XML数据，然后将生成后的XML数据提交到web服务器，web服务器接收到XML数据后，将XML数据交给XML解析器，由XML解析器完成对XML数据的解析后把数据再返回给应用。在整个数据处理流程中，XML生成器和XML解析器这两个点是最容易出现问题的。



在XML解析的过程中，最常见的漏洞就是拒绝服务漏洞、XML注入和XML外部实体注入。

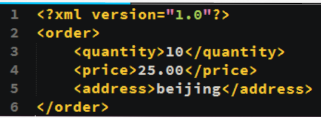
我们逐个分析，先来看拒绝服务攻击。

下图是一个典型拒绝服务攻击的payload

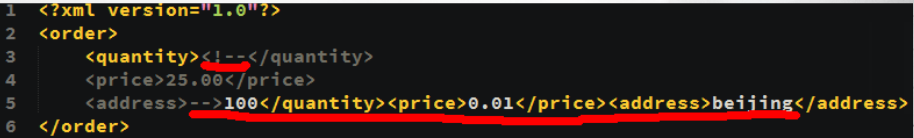


可以看到，payload中先定义了lol实体，值为“lol”字符串，然后下面又定义了lol2实体，在lol2实体中，引用10个lol实体，也就是说现在lol2的值是10个“lol”字符串，下面的lol3又引用了10个lol2实体的值，现在lol3的值是100个“lol”字符串，依此类推，到了最后在lolz元素中引用的lol9中，就会存在上亿个“lol”字符串，如果程序在解析数据时没有做特别的处理，那么极有可能对程序造成拒绝服务攻击。

再来看XML注入的例子：

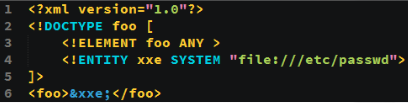


例图这是一个标准的xml文件，即将被提交到web服务器中，我们假设这是一个存有订单信息的xml文件，其中price的值不可控，quantity和address项的值可控，可以由用户自由输入，那么这时，攻击者可以实施xml注入攻击，在quantity的值中构造“<!--”，注释符，然后在address的值中先闭合前面的注释符，然后再重新根据原xml文件结构重新构造xml文件，这时候price元素的值也是可以由攻击者随意构造的，如下图：



可以看到，我们重新构造出了一个xml文件，这时候price的值由原来的25变成了0.01，也就是说，订单的价格被成功的修改了。

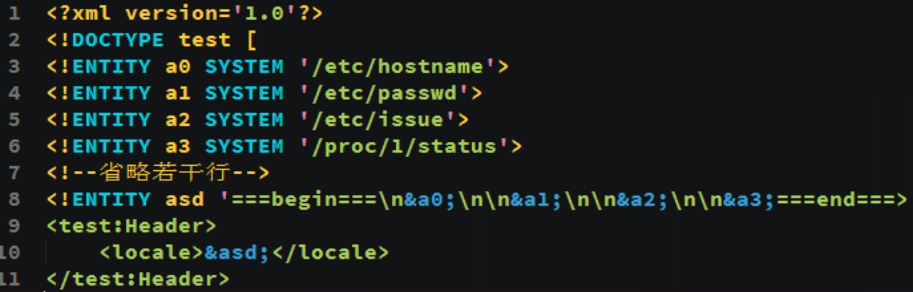
在众多xml漏洞中，最容易出现的就是XXE（XML\_External\_Entity），也就是XML外部实体注入攻击。在文档类定定义部分，可以引用外部的dtd文件，dtd文件的路径可以是URL地址，也可以使用多种协议，所以在这里会出现安全问题。



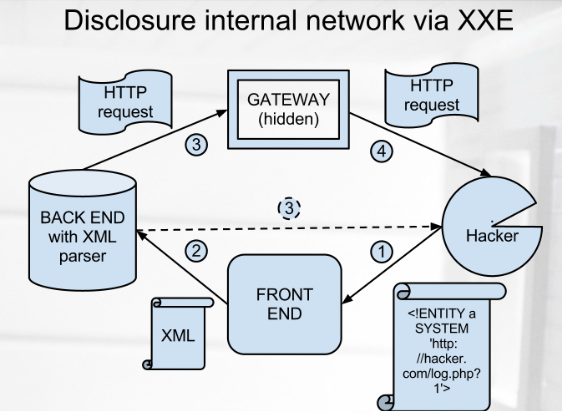
上图为一个标准的XXE Payload，可以看到，这里引用的外部实体为<file:///etc/passwd>，如果XML解析库在解析到这里时允许引用外部实体，那么程序将会请求这个地址，并将内容返回。

下面的demo是对phpmyadmin一个xxe漏洞的利用演示。

对于XXE漏洞，我们还有更多的利用技巧，比如某些场景下，我们需要在一次请求中读取多个文件，那么可以用下面的payload来实现：

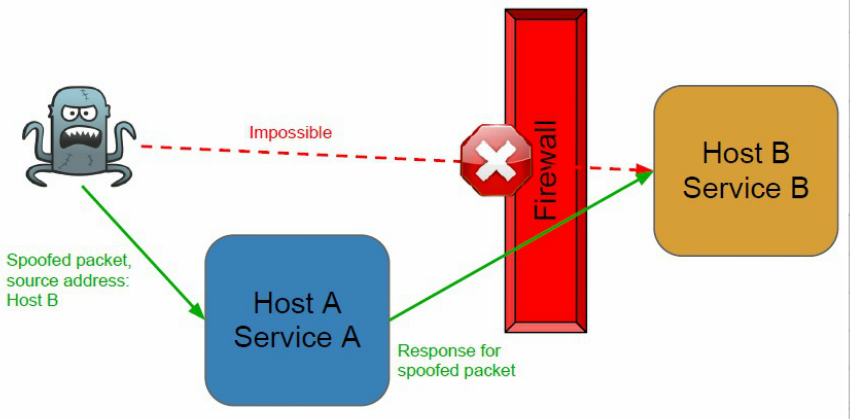


XXE不光可以读取文件，还可以通过SSRF来完成更深入的攻击。SSRF（Server Side Request Forgery）被称为服务端请求伪造攻击，与CSRF不同的是它的请求是由服务器发起的，并不是由客户端发起。



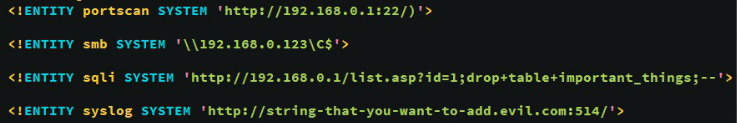
通过这张图我们可以看到，攻击者先发送包含payload的请求到前端，前端接到数据后将xml数据发送到后端的xml解析器进行解析，这时payload被执行，payload执行后，结果可能会直接被后端返回给攻击者，也可能由payload中所指向的服务器返回给攻击者。

更常见的攻击场景如下：



攻击者想要访问主机B上的服务，但是由于存在防火墙的原因无法直接访问，这是可以借助主机A来发起SSRF攻击，通过主机A向主机B发起请求，从而完成攻击。

SSRF攻击常用的payload如下：



Portscan实体是通过http协议来访问主机的某些端口，通过返回信息来判断端口是否开放

Smb实体是访问内网中的共享资源

Sqli实体是对内网中web服务器发起sql注入攻击

Syslog实体可以像内网syslog服务器添加垃圾日志

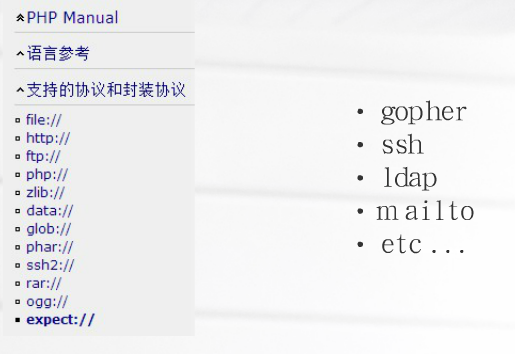
除了这些，还可以扩展思路，比如对内网中的zabbix\_agent进行攻击(在外部实体引用 [gopher://ip:10050/1system.run[ls](gopher://ip:10050/1system.run%5bls)])，即可在该主机执行系统命令。

在PHP环境下，还可以利用封装协议来直接执行系统命令，如expect://，下面的demo中演示的是利用zend freamework的xxe漏洞来执行系统命令

也可以用php://来读取文件内容



除了这些，还支持其他很多协议，可以自由发挥编写payload



刚才在SSRF攻击中提到了访问共享资源，下面这个demo演示了其所带来的危害，利用的是postgresql的xxe漏洞

随着应用越来越庞大、逻辑越来越复杂，有些需求要用到XML来实现，但很多开发人员还没有充分意识到在使用和解析XML文件时可能会导致安全问题出现，xml漏洞也是一个比较好的研究方向。