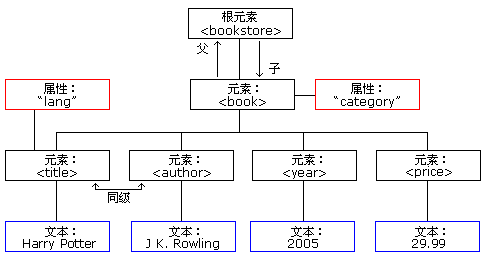
**XXE基础**

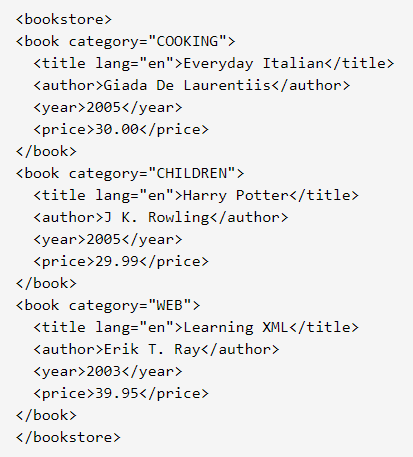
XXE(XML External Entity Injection, XML外部实体注入)，网页解析XML时没有禁止外部实体加载，通过注入恶意外部文件，造成文件读取、命令执行、内网端口扫描、攻击内网网站、拒绝服务攻击等危害。

1. 背景知识
   1. **XML**

XML(Extensible Markup Language, 可扩展标记语言)，是一种标记电子文件，使其具有结构性的标记语言，可以用来标记数据、定义数据类型，同时允许用户定义自己的标签和文档结构等。

XML为树结构，由一个根元素以及多个子元素构成。具体结构与语法可以参考w3school中的这个例子：





XML的结构包括XML声明、DTD文档类型定义、文档元素。

XML由元素、属性、实体、PCDATA、CDATA五个构建模块组成。

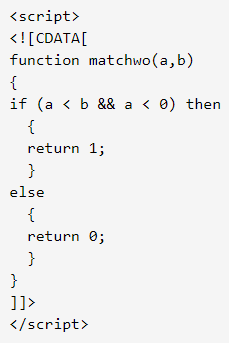
* 1. **PCDATA与CDATA**

PCDATA(Parsed Character DATA, 解析字符数据)是会被解析器解析的文本，文本中的标签会作为标记处理，实体会被展开。文本中不能包含任何单独的 & < > 字符，需要使用实体来替代。

CDATA(Unparsed Character DATA, 字符数据)不会被解析器解析。某些文本，例如JavaScript代码，包含大量 & < > 等字符，为了避免错误，可以定义为CDATA。

定义CDATA: <![CDATA[ Content ]]>

例：w3school



* 1. **DTD**

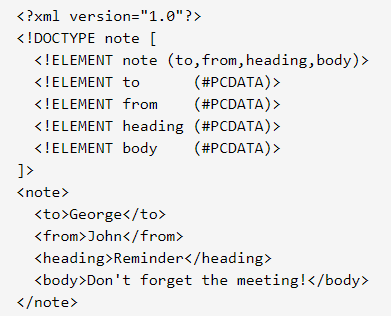
DTD(Document Type Definition, 文档类型定义)，使用一系列合法的元素定义文档结构，它可以在XML文档内部定义，也可以作为外部引用。

* + - **内部声明**

DOCTYPE声明：<!DOCTYPE root [element]>

Element声明：<!ELEMENT name (content)>

例子参考：w3school



第一行为XML声明。

第二行开始，定义DTD，使用DOCTYPE声明，定义此文档根元素为note。

第三行到第七行，每一行定义了一个元素：

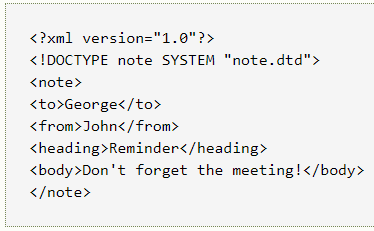
note元素有to, from, heading, body四个子元素；

这四个子元素均为 #PCDATA 类型。

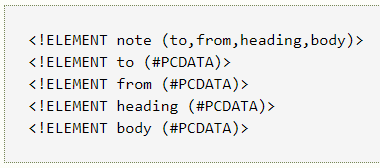
* + - **外部文档引用**

DOCTYPE声明：<!DOCTYPE root SYSTEM "file">

例子参考：w3school



其中note.dtd:



DTD是对文档自身的描述，使用DTD便于文档的传输、共享与验证。

更加细致的DTD具体内容可以参考：

[XML——XML文件约束之DTD详解\_Gavin-CSDN博客](https://blog.csdn.net/gavin_john/article/details/51532756)

* 1. **DTD实体**

DTD实体是用于定义引用普通文本或特殊字符的快捷方式的变量，可以在内部或外部声明。在XML中，通过 &实体名称;的方式引用实体。

* + - **内部实体声明**

ENTITY声明：<!ENTITY name "value">

例：



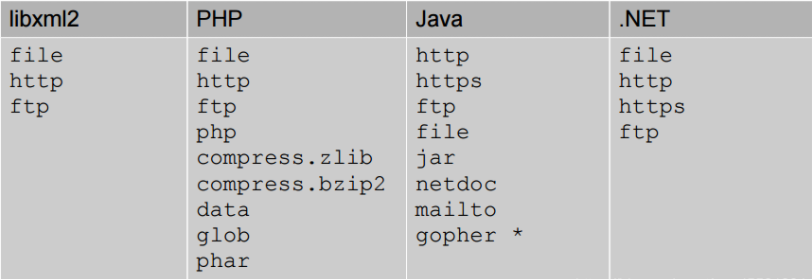
* + - **外部实体声明**

ENTITY声明：<!ENTITY name SYSTEM "URI/URL">

例：



外部引用支持的协议参考：



* + - **参数实体**

参数实体是只能用于DTD内部的实体，该类型实体用%声明，用%实体名称;的方式引用。

内部声明：<!ENTITY % name "value">

外部声明：<!ENTITY % name SYSTEM "URI/URL">

* 1. **XML外部实体攻击**

SYSTEM关键字会令XML解析器从URI中读取内容，并允许它在XML文档中被替换，因此攻击者可以通过实体将自定义的值发送给应用程序，并让应用程序呈现，这就能够通过XML解析器访问指定的资源内容。

* 1. **Blind XXE**

Blind XXE是在网页无回显的情况下进行XXE攻击的一种方法。这种方法构建了一条带外信道来提取数据，利用外部实体中的URL发出访问，从而使目标服务器与攻击者的服务器产生联系。

Blind XXE主要使用DTD中的参数实体和内部实体。参数实体上面已经提到过，而这里的内部实体指的是在一个实体中嵌套定义的另一个实体，内层的参数实体的%需要进行HTML转义。

常见的Blind XXE使用三层嵌套来从目标服务器上获取数据。

更加具体的内容可以参考：

[Blind XXE详解与Google CTF一道题分析 - FreeBuf网络安全行业门户](https://www.freebuf.com/vuls/207639.html)

[Finding and exploiting XXE - XML external entities injection - Infosec Resources (infosecinstitute.com)](https://resources.infosecinstitute.com/topic/finding-and-exploiting-xxe-xml-external-entities-injection/)

* 1. **XXE的防御方法**

php提供了禁用外部实体的方法：

libxml\_disable\_entity\_loader(true);

也可以对用户提交的XML数据进行过滤，主要是SYSTEM、PUBLIC这两个关键词，也可直接过滤<!DOCTYPE。

各种语言下关闭外部实体的方法可以参考：

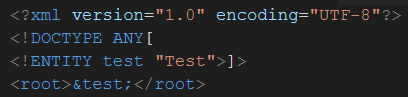
[应用安全系列之八：XXE\_jimmyleeee的专栏-CSDN博客](https://blog.csdn.net/jimmyleeee/article/details/114104185)

中的表格。

1. XXE简要思路

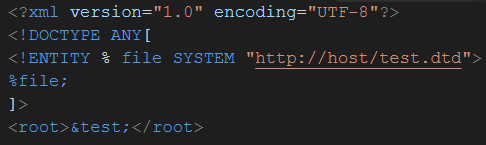
XXE作为一个注入漏洞，首先要做的是判断网站是否存在可以注入内容的点位。由于存在漏洞的服务必然存在xml传输，一个简单的办法是利用burp抓包观察http头的Content-Type字段，常见的是直接包含xml字样的，例如text/xml, application/xml等，也有一些application/json的可以通过修改为application/xml进行注入。

对于有回显的网页，首先判断传入的XML文件是否会被解析：



若传入后页面显示Test字样，那么说明这个test实体得到了解析，并返回了root节点，说明XML文件能够被解析。

再判断是否支持引用DTD外部实体：



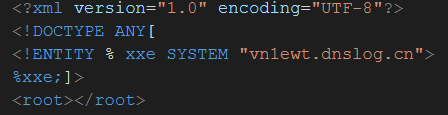
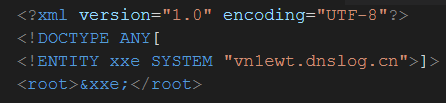


若传入后页面显示Test字样，则说明该站点可以引用外部实体。

对于没有回显的网页，可以利用DNSLog平台，或任意远程服务器进行测试：

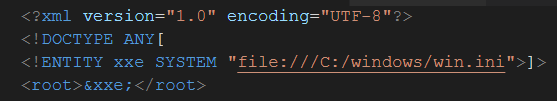


点击左侧按钮会在中间生成一个地址。将地址作为实体的值传入站点，之后查看该页面是否有访问记录即可。



1. XXE攻击应用
   1. **文件读取**

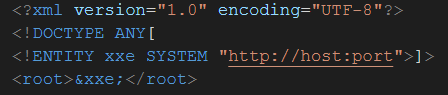
XXE攻击特别常见的一种应用。利用外部实体可以引用文件的特性，将系统文件的内容显示在页面中：



如果确定服务器为windows系统，则一定存在win.ini文件，通过这种方式可以判断能否访问系统文件。如果为linux系统，可以访问/etc/passwd。一旦成功访问，就可以将路径换为任意系统文件。

* 1. **内网端口探测**

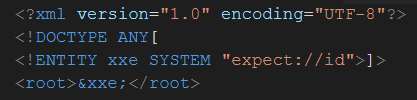
根据解析器种类的不同，可能可以利用XXE攻击进行内网端口探测。



利用这种方法进行探测，根据页面是否提示连接信息、响应的时间和长度，有时可以判断内网端口是否处于开启状态。

* 1. **远程代码执行**

一些情况下，网页存在配置不当的问题，如果PHP expect模块被加载到易受攻击的系统或处理XML的内部应用程序上，就可以执行远程代码。



* 1. **拒绝服务攻击**

一个非常经典的例子：Billion Laughs。

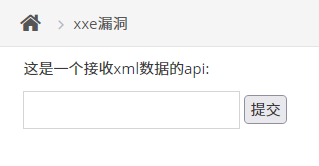


简直是lol宇宙。通过这种定义，lol9会成为一个由10^9个lol拼接而成的庞然大物，如果未作任何处理，解析这段XML会消耗巨量内存，从而造成拒绝服务攻击。

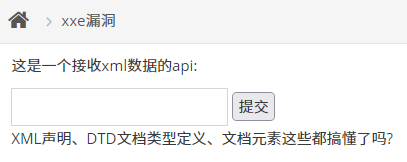
1. 靶场实例
   1. **Pikachu-XXE**

Pikachu是一个包含各种常见漏洞的靶场。[项目地址](https://github.com/zhuifengshaonianhanlu/pikachu)

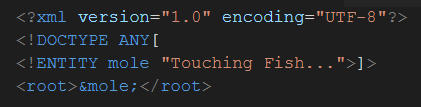
Pikachu的XXE漏洞页面非常简单粗暴，如图：



当你提交的内容为非XML时，会触发温馨提示：



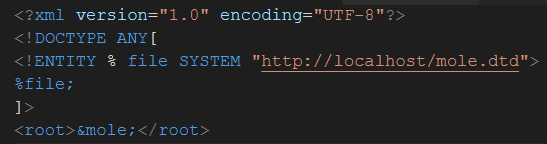
那么先尝试注入一个简单的XML文档：



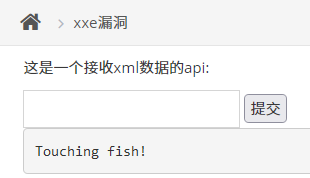


页面能够解析摸鱼信息，并且是有回显的。

接下来尝试能否引用外部实体：

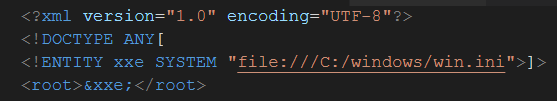






说明可以引用外部参数实体。

* + - **系统文件读取**



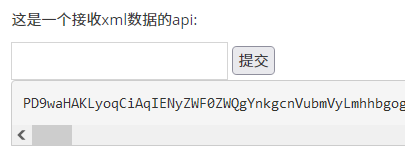


* + - **获取php源码**

利用php伪协议可以获取网页的php源码。由于源码中可能包含大量 & < > 字符，一般在获取源码的时候会进行base64编码。

例如想要获取xxe这个页面的源码，就可以将URL改为：

php://filter/convert.base64-encode/resource=D:/phpstudy\_pro/WWW/pikachu/vul/xxe/xxe\_1.php



能够获取一个很长的base64编码。解码以后即是页面源码。

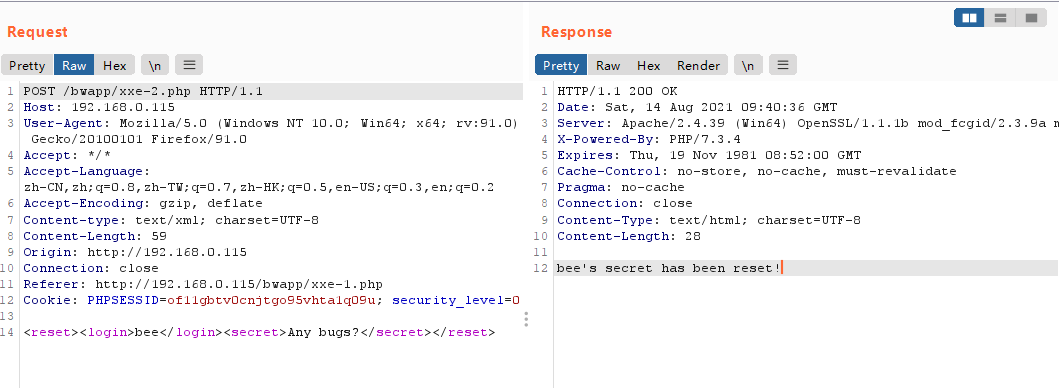
* 1. **bWapp-XXE**

bWapp是一个经典的渗透测试靶场。

XXE界面：



什么都没有。考虑先使用burpsuite抓包：



可以看到点击Any bugs按钮后，页面会发送该请求，其中包含一个XML。

如果在这里引用一个外部实体，就可能获取文件信息。

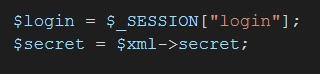
Low难度下，源码对xml没做任何过滤，且会直接将Login中的内容显示出来：



因此实际操作与上文pikachu靶场一致。

Medium&High难度：

Medium和High难度在源码中没有分别，和Low区别在于此时的login元素值从session中获取：



所以此时修改login元素值进行XXE攻击的方法不再适用。