1、什么是XXE漏洞

XXE(XML External Entity)是指xml外部实体攻击漏洞。XML外部实体攻击是针对解析XML输入的应用程序的一种攻击。当包含对外部实体的引用的XML输入被弱配置XML解析器处理时，就会发生这种攻击。这种攻击通过构造恶意内容，可导致读取任意文件、执行系统命令、探测内网端口、攻击内网网站等危害。

2 XML结构介绍

要了解XXE漏洞，那么一定要先学习一下有关XML的基础知识。

XML文件的结构性内容，包括节点关系以及属性内容等等。
元素是组成XML的最基本的单位，它由开始标记，属性和结束标记组成。**
就是一个元素的例子，每个元素必须有一个元素名，元素可以若干个属性以及属性值。
xml文件和html文件一样，实际上是一个文本文件。显然大家立刻就会明白，创建xml文件最普通的工具和html一样，就是“记事本”了。
一个xml文件的例子
现在我们暂且使用“记事本”来创建我们的xml文件吧。先看一个xml文件：
例1
〈?xml version="1.0" encoding="gb2312" ?〉
〈参考资料〉
〈书籍〉
〈名称〉xml入门精解〈/名称〉
〈作者〉张三〈/作者〉
〈价格 货币单位="人民币"〉20.00〈/价格〉
〈/书籍〉
〈书籍〉
〈名称〉xml语法〈/名称〉
〈!--此书即将出版--〉
〈作者〉李四〈/作者〉
〈价格 货币单位="人民币"〉18.00〈/价格〉
〈/书籍〉
〈/参考资料〉
————————————————————————

** 这是一个典型的xml文件，编辑好后保存为一个以.xml为后缀的文件。** 我 们可以将此文件分为文件序言（prolog）和文件主体两个大的部分。在此文件中的第一行即是文件序言。该行是一个xml文件必须要声明的东西，而且也必 须位于xml文件的第一行，它主要是告诉xml解析器如何工作。其中，version是标明此xml文件所用的标准的版本号，必须要有；encoding 指明了此xml文件中所使用的字符类型，可以省略，在你省略此声明的时候，后面的字符码必须是unicode字符码（建议不要省略）。
文件的其余部分都是属于文件主体，xml文件的内容信息存放在此。我们可以看到，文件主体是由开始的〈参考资料〉和结束的〈/参考资料〉控制标记组成，这 个称为xml文件的“根元素”；〈书籍〉是作为直属于根元素下的“子元素”；在〈书籍〉下又有〈名称〉、〈作者〉、〈价格〉这些子元素。货币单位是〈价 格〉元素中的一个“属性”，“人民币”则是“属性值”。元素与属性的关系如图1。
〈!--此书即将出版--〉这一句同html一样，是注释，在xml文件里，注释部分是放在“〈!--”与“--〉”标记之间的部分。
大家可以看到，xml文件是相当简单的。同html一样，xml文件也是由一系列的标记组成，不过，xml文件中的标记是我们自定义的标记，具有明确的含义，我们可以对标记中的内容的含义作出说明。
xml文件的语法
对xml文件有了初步的印象之后，我们就来详细地谈一谈xml文件的语法。**在讲语法之前，我们必须要了解一个重要的概念，就是xml解析器（xml parse）。**

1. xml解析器
   解析器的主要功能就是检查xml文件是否有结构上的错误， 剥离xml文件中的标记，读出正确的内容，以交给下一步的应用程序处理。xml是一种用来结构化文件信息的标记语言，xml规范中对于如何标记文件的结构性有一个详细的法则，解析器就是根据这些法则写出来的软件（多用java写成）。
2. well-formed的xml文件
   我们知道，xml必须是well-formed的，才能够被解析器正确地解析出来，显示在浏览器中。那么什么是well-formed的xml文件呢？主要有下面几个准则，我们在创建xml文件的时候，必须满足它们。

   首先，xml文件的第一行必须是声明该文件是xml文件以及它所使用的xml规范版本。在文件的前面不能够有其它元素或者注释。

   第二，在xml文件中有且只能够有一个根元素。我们的第一个例子中，〈参考资料〉... 〈/参考资料〉就是此xml文件的根元素。
   xml文件中，用的大多都是自定义的标记。

   第三，在xml文件中的标记必须正确地关闭，也就是说，在xml文件中，控制标记必 须有与之对应的结束标记。如：〈名称〉标记必须有对应的〈/名称〉结束标记，不像html，某些标记的结束标记可有可无。如果在xml文件中遇到自成一个 单元的标记，就是类似于html 中的〈img src=.....〉的这些没有结束标记的时候，xml把它称为“空元素”，必须用这样的写法：〈空元素名/〉，如果元素中含有属性时写法则为：〈空元素 名 属性名=“属性值”/〉。

   第四，标记之间不得交叉。在以前的html文件中，可以这样写：
   〈b〉〈h〉xxxxxxx〈/b〉〈/h〉,〈b〉和〈h〉
   标记之间有相互重叠的区域，而在xml中，是严格禁止这样标记交错的写法，标记必须以规则性的次序来出现。

   第五，属性值必须要用“ ”号括起来。 如第一个例子中的“1.0”、“gb2312”、“人民币”。都是用“ ”号括起来了的，不能漏掉。

   第六，控制标记、指令和属性名称等英文要区分大小写。与html不同的是，在html中， 类似〈b〉和〈b〉的标记含义是一样的，而在xml中，类似〈name〉、〈name〉或〈name〉这样的标记是不同的。

   第七，我们知道，在html文件中，如果我们要浏览器原封不动地将我们所输入的东西显示出来，可以将这些东西放到〈pre〉〈/pre〉或者〈xmp〉 〈/xmp〉标记中间。这对于我们创建html教学的网页是必不可少的，因为网页中要显示html的源代码。而在xml中，要实现这样的功能，就必须使用 cdata标记。在cdata标记中的信息被解析器原封不动地传给应用程序，并且不解析该段信息中的任何控制标记。cdata区域是由：“〈! [cdata[”为开始标记，以“]]〉”为结束标记。例如：例2中的源码，除了“〈![cdata[”和“]]〉”符号，其余的内容解析器将原封不动地 交给下游的应用程序，即使cdata区域中的开始和结尾的空白以及换行字符等，都同样会被转交（注意cdata是大写的字符）。**

1 XML **的声明**

```
<?xml version=”1.0” standalone=”yes” encoding=”UTF-8”?>
```

这是一个XML处理指令。处理指令以 <? 开始，以 ?> 结束。<? 后的第一个单词是指令名，如xml，代表XML声明。
version, standalone, encoding 是三个特性，特性是由等号分开的名称-数值对，等号左边是特性名称，等号右边是特性的值，用引号引起来。

几点解释:

- version: 说明这个文档符合1.0规范
- standalone: 说明文档在这一个文件里还是需要从外部导入, standalone 的值设为yes 说明所有的文档都在这一文件里完成
- encoding: 指文档字符编码

**

2 XML **根元素定义**

XML文档的树形结构要求必须有一个根元素。根元素的起始标记要放在所有其它元素起始标记之前，根元素的结束标记根放在其它所有元素的结束标记之后，如

```
<?xml version=”1.0” standalone=”yes” encoding=”UTF-8”?>
<Settings>
<Person>Zhang San</Person>
</Settings>
```

3 **XML 元素**

元素的基本结构由 开始标记，数据内容，结束标记组成，如

```
<Person>
  <Name> Zhang San </Name>
  <Sex> Male </Sex>
</Person>
```

需要注意的是:

- 元素标记区分大小写，`<Name>` 与 `<name>` 是两个不同的标记
- 结束标记必须有反斜杠，如 `</Name>`

XML元素标记命名规则如下:

- 名字中可以包含字母，数字及其它字母
- 名字不能以数字或下划线开头
- 名字不能用xml开头
- 名字中不能包含空格和冒号

4 **XML 中的注释**

XML中注释如下:

```
<!-- this is comment -->
```

需要注意的是:

- 注释中不要出现“--”或“-”
- 注释不要放在标记中
- 注释不能嵌套

5 PI (Processing Instruction)
PI 指 Processing Instruction, 处理指令。PI以“`<?”开头，以“?>`”结束，用来给下游的文档传递信息。

```
<?xml:stylesheet href=”core.css” type=”text/css” ?>
```

例子表明这个XML文档用core.css控制显示。

6 DTD
DTD定义了XML的根元素是movies，然后元素下面还有一些子元素，其中DOCTYPE是DTD的声明；ENTITY是实体的声明，所谓实体可以理解为变量；SYSTEM、PUBLIC是外部资源的申请。那么XML到时候必须像如下这么写

示例文件(movies.dtd)

```
<?xml version="1.0" encoding="GB2312"?>
```

```
<!ELEMENT movies (id, name, brief, time)>
<!ATTLIST movies type CDATA #REQUIRED>
<!ELEMENT id (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT brief (#PCDATA)>
<!ELEMENT time (#PCDATA)>

id, name, brief, time只能包含非标记文本(不能有自己的子元素)。


XML文件如下所示(movies.xml):

<?xml version="1.0" encoding="GB2312"?>
<!DOCTYPE movies SYSTEM "movies.dtd">
<movies type="动作片">
    <id>1</id>
    <name>致命摇篮</name>
    <brief>李连杰最新力作</brief>
    <time>2003</time>
</movies>
```

8 **Entities(** 实体)
Entities(实体)是XML的存储单元，一个实体可以是字符串，文件，数据库记录等。实体的用处主要是为了避免在文档中重复输入，我们可以为一个文档定义一个实体名，然后在文档里引用实体名来代替这个文档，XML解析文档时，实体名会被替换成相应的文档。

XML为五个字符定义了实体名：

实体	字符
<	<
>	>
&	&
"	“
'	‘

定义并引用实体的示例：

```
<!DOCTYPE example [
  <!ENTITY intro "Here is some comment for entity of XML">
]>
<example>
    <hello>&intro;</hello>
</example>
```

使用&intro对上面定义的intro实体进行了引用，到时候输出的时候&&intro就会被“Here is some comment for entity of XML”替换。而内部实体是指在一个实体中定义的另一个实体，也就是嵌套定义。

漏洞产生的原因在于外部实体

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///c:/test.dtd" >]>
<creds>
    <user>&xxe;</user>
    <pass>mypass</pass>
</creds>
```

解析XML：案例

```
        String strxml ="<?xml version=\"1.0\" encoding=\"UTF-8\"?>\r\n" +
                "<!DOCTYPE foo [\r\n" +
                "<!ELEMENT foo ANY >\r\n" +
                "<!ENTITY xxe SYSTEM \"file:///c:/windows/win.ini\" >]>\r\n" +
                "<creds>\r\n" +
                "<user>&xxe;</user>\r\n" +
                "<pass>mypass</pass>\r\n" +
                "</creds>";
        // TODO Auto-generated method stub
        DocumentBuilderFactory docFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
        Document doc = docBuilder.parse(new InputSource(new
StringReader(strxml)));
        NodeList RegistrationNo = doc.getElementsByTagName("user");
        String tmp = RegistrationNo.item(0).getFirstChild().getNodeValue();
        System.out.println(tmp);
```

怎么防御：

1.使用开发语言提供的禁用外部实体的方法

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
dbf.setExpandEntityReferences(false);
```

2.过滤用户提交的XML数据

过滤关键词：!ENTITY，或者SYSTEM和PUBLI

3.使用第三方应用代码及时升级