**Weblogic RCE 仅通过一个 GET 请求 — CVE-2020–14882 分析**



就在那时，我花了大约一年时间，去年正和盖吉特·因斯佩克托一起工作，找到CVE-2020-2555，并报告给ZDI。

*散漫咔嚓咔嚓，谁想读就往下拉也行*

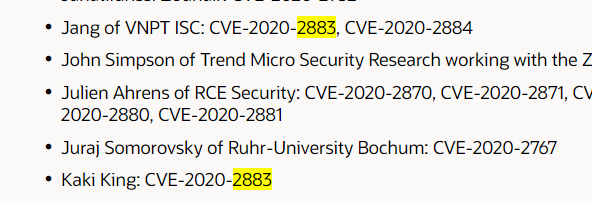
那一天，甲骨文CPU的更新是期待，看看自己的bug是否修补，多么公共的PoC...最后，6个月后，甲骨文终于重新发布了自己报告的bug补丁。

当时也有点青涩，没想到找到一个新的代币瓶也算新bug，并被编号为9.8/10...

在 CVE-2020-2883 和 2884（2555 的通行证）之后，我厌倦了，不想再从事搜索工作，并在 Weblogic 上重复了 1 个 T3 点。

那段时间也是项目季节，所以我把所有工作都交给我的同事，是谁，想必很多人都知道=））。

在CVE-2020-2555之后，许多人也意识到，钡钴有很多东西可以利用。从那时起，每次甲骨文CPU重新发布时，都会添加一堆新的网络博客T3增强剂，有时还有与某人重合的CVE：



事实上，找到加奇奇奇是有趣的，但只有在内在点很有趣。

阻止很简单，只要T3/IIOP的移动，就停止游戏̄\\_(ツ)\_/ ̄。

.

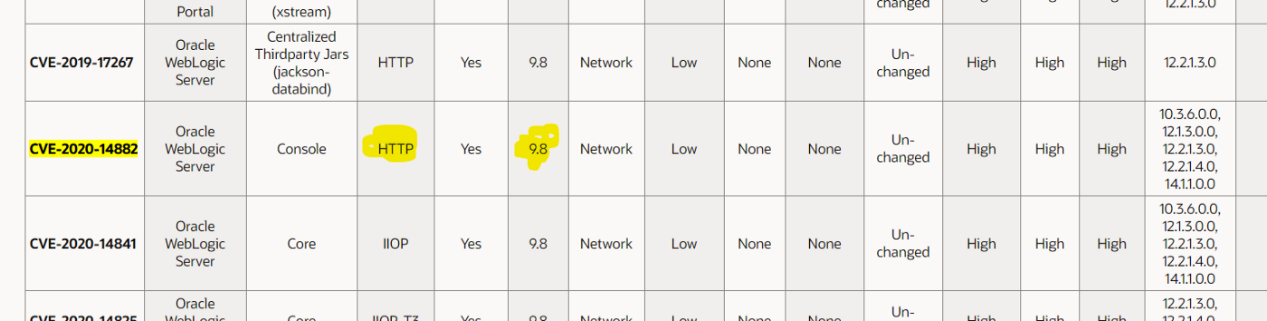
.

.

直到去年10月21日，甲骨文CPU Oct带来了一些巨大的惊喜。

看一眼网络，就像往常一样：T3，IIOP...正常，

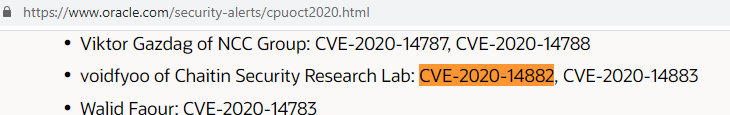
然而，这里有一个新的点：



**CVE-2020–14882**：

* CVSS： 9.8/10
* 协议： **HTTP**
* 影响所有版本

这个bug是由一个年轻人报告的：

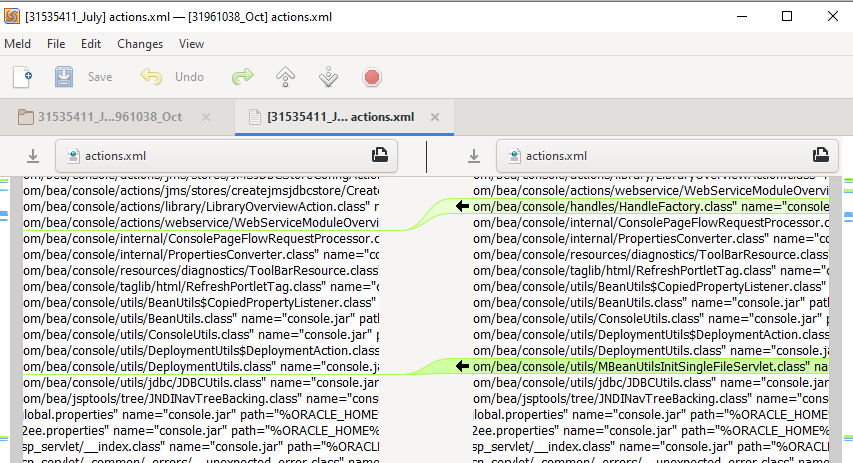


一直以来，网络博客在T3沙漠中几乎只有bug，一次又一次地修补，然后穿，再打补丁...（想想也够累的）。

咔嚓咔嚓，又要拉帕奇回去看看吧...

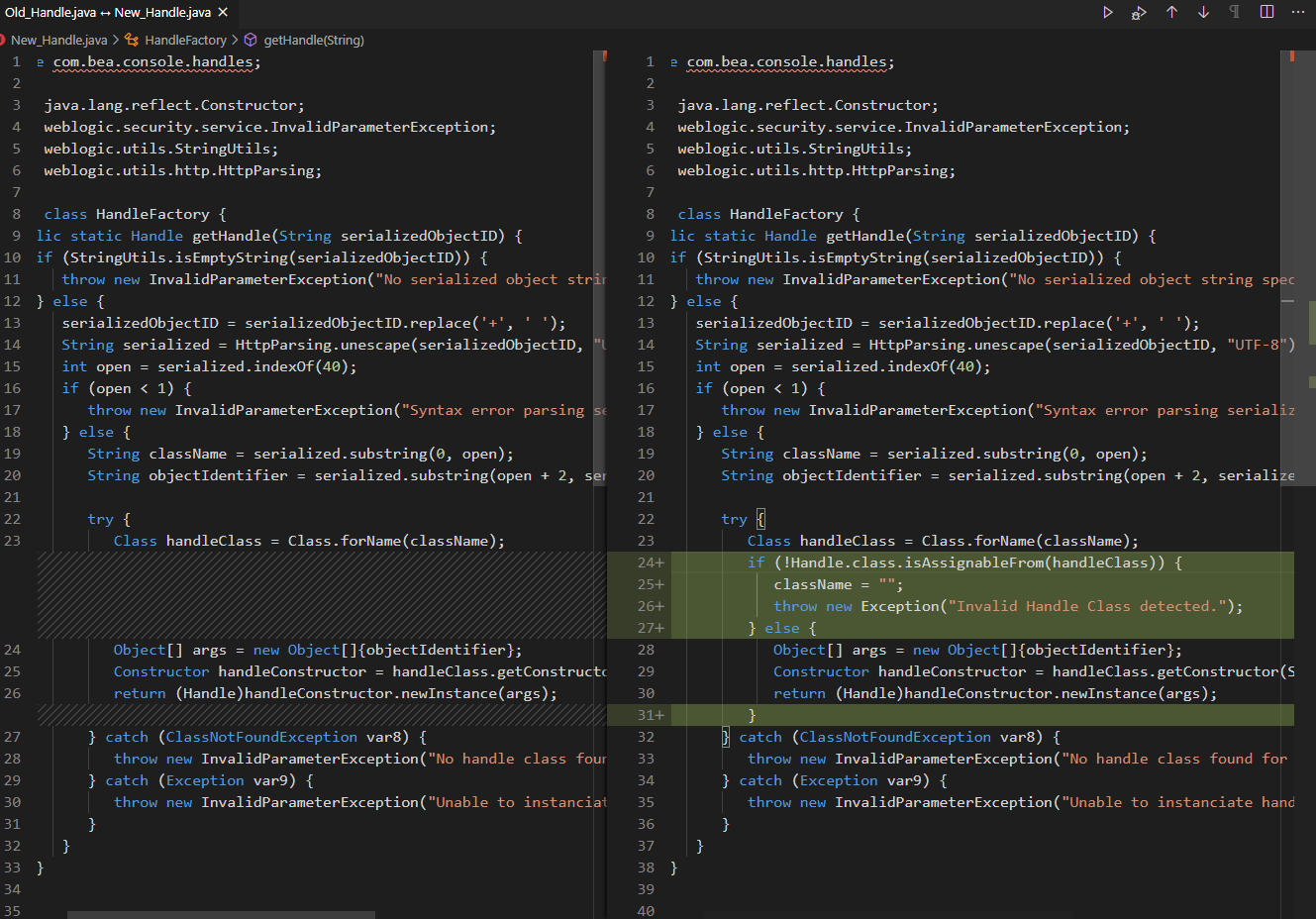
我选择的版本来分析是 Weblogic 12.2.1.3， Oct 修补程序有3191038， 朱莉是31535411。

Diff 2 补丁，在组件控制台中发现，一些类文件在新修补程序中被更改：**手部工厂.class**和 **MBeanUtilsInitSingle 服务.class**

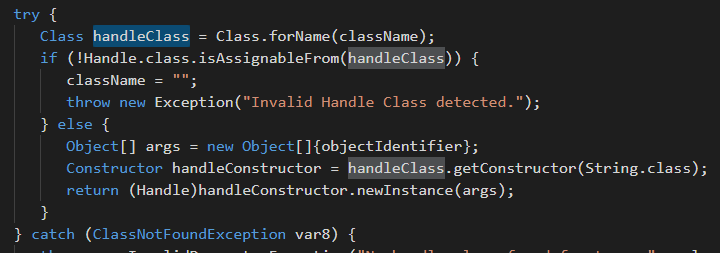


**#The水槽**

继续查看这些文件的变化，首先是 **com.bea.console.手。手部工厂.getHandle（）：**

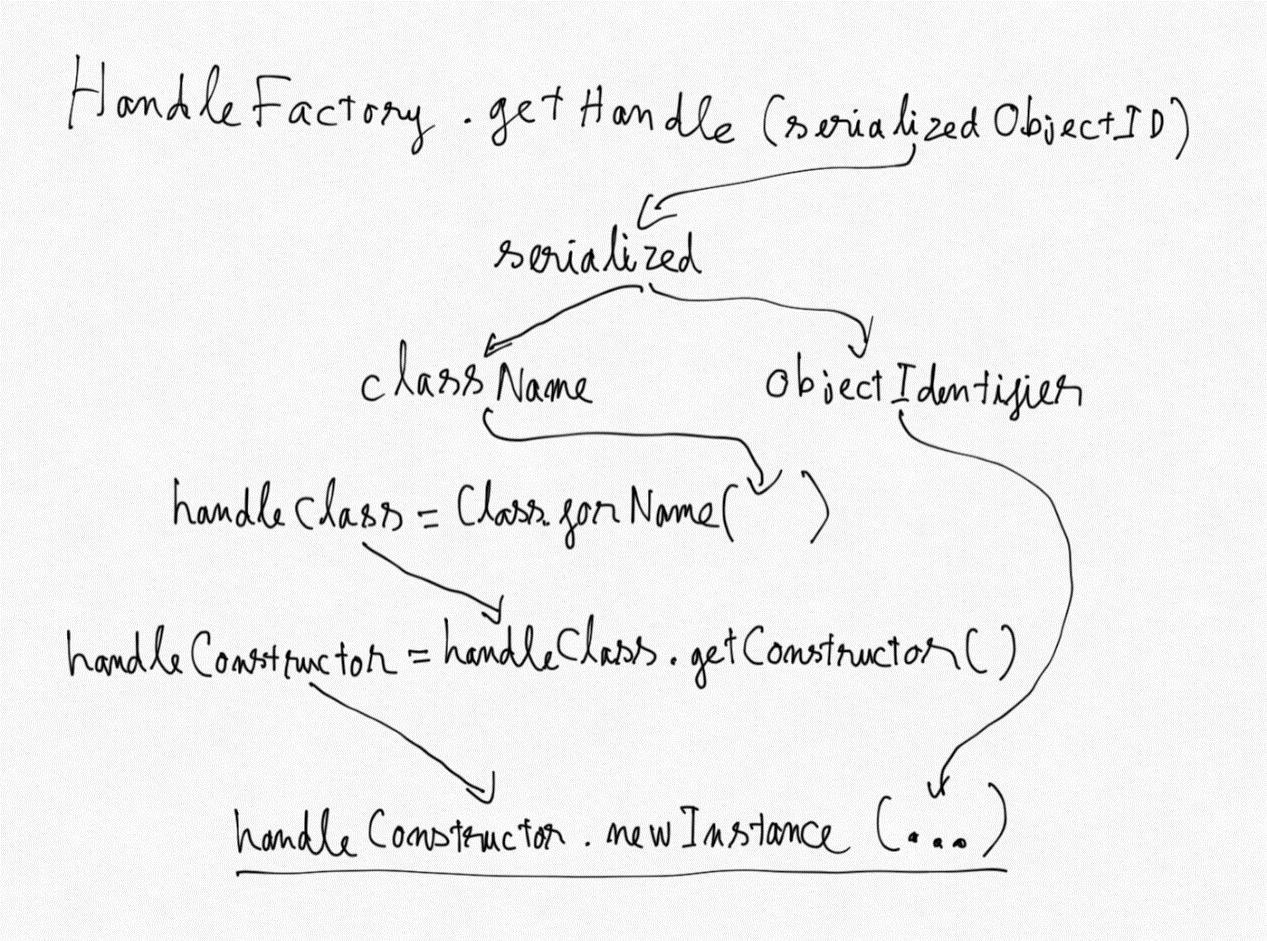


粗略一看，也可以立即注意到这一区别：**手部**的补丁这次重新测试了**手部的**样式，只允许是 **com.bea.console.手**部的内在类。



根据甲亢**手部的**流，接收的数据是1个流，经过多次处理后，将到达**一类。forName（ClassName）**来提升这个类。接下来，将搜索其中一个类中的一个建筑者与一个 arg 是斯特林，从中创建新的研究所。

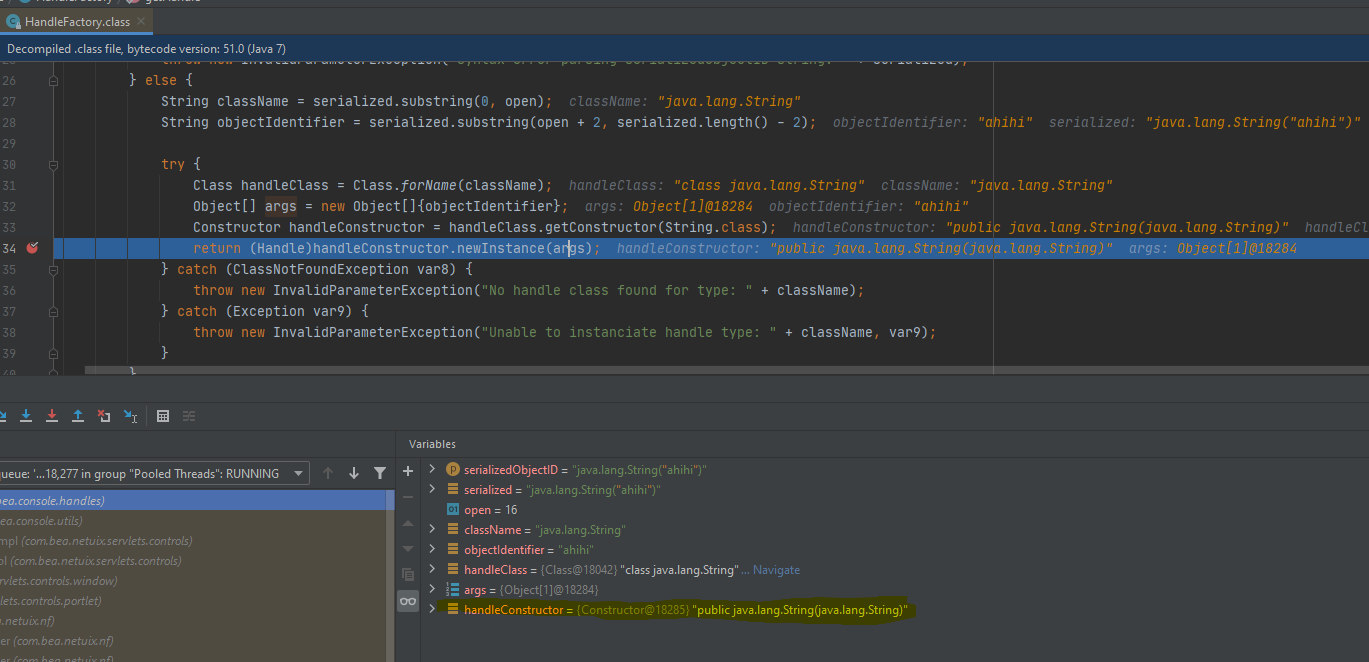
处理流程的解释如下：



可以通过以下 GET 回放此点（但此段仍然需要先进入）：

http://<target>/console/console.portal?\_nfpb=true&\_pageLabel=HomePage1&handle=**java.lang.String("ahihi")**

在手部工厂.getHandle（），**手部控制器**、**ARGS**的价值都受到输入数据的控制。



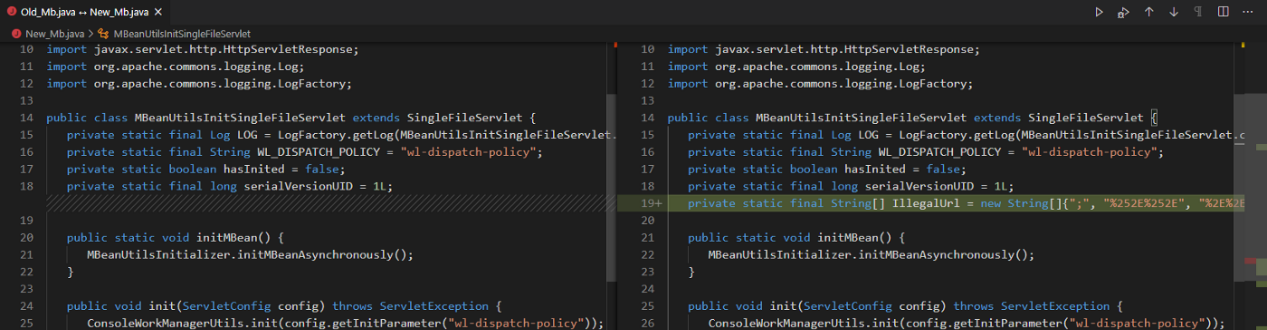
这也是 RCE 在 CVE-2020-14882 中被利用的位置，

要利用这个点，有必要找出一个包含公共建筑的类与一个镂门 = 斯特林。

当debug到达这里时，我也会放在那里，反正单人都知道，直到找到预置点，才跑tool去寻找bug的三重奏！

**#The来源**

有了手部.getHandle（）已经是这个bug的单一，那么剩下的馅饼类 —— MBeanUtilsInitSingleFileServlet 可能是bug的源！



打补丁后，将增加一个现场“伊尔加尔尔”如下：

private static final String[] **IllegalUrl** = new String[]{";", "**%252E%252E**", "**%2E%2E**", "**..**", "%3C", "%3E", "<", ">"};

这个服务员的处理代码只是检查 URL 上是否存在“**...**”字符串，如果有的话，将重新回复此！“**”**标志一定有问题， 所以它才会被重新塑造。

此外，要重新获得 M 贝安蒂尔尼特西尔服务服务（），需要再过一̄\\_(ツ)\_/ ̄步。

“*所以，在 RCE 后，怎么可能是 9.8/10...*他心想，继续摸索。

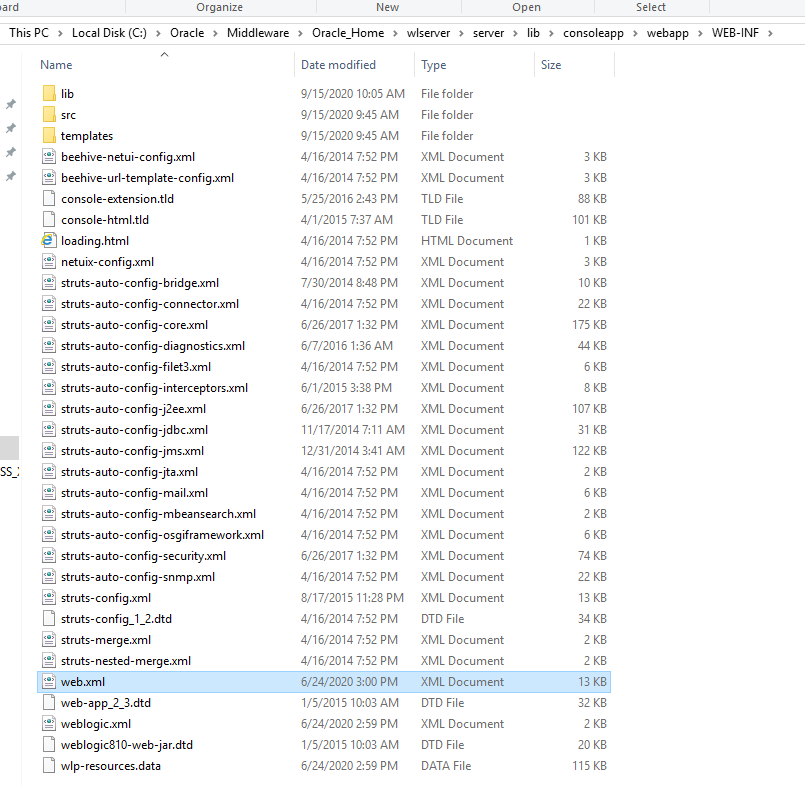
.

.

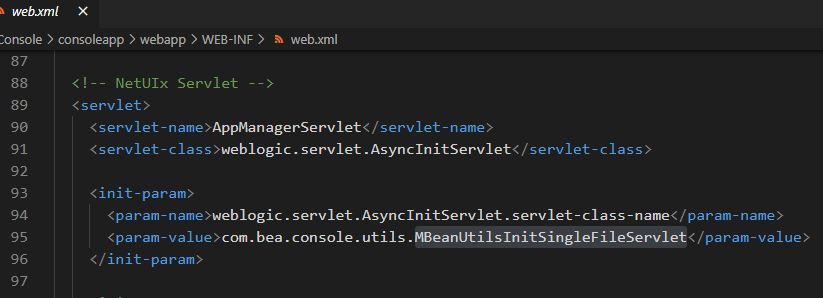
也许就像java上的一些服务器一样，网络博客也有一个网络应用程序，对于控制台来说，那就是：

“*\Middleware\Oracle\_Home\wlserver\server\lib\consoleapp\webapp*”

当然，也有网络.xml、流氓文件.xml像往常一样：

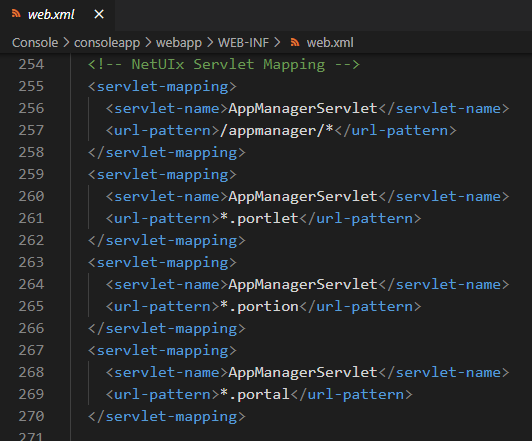


根据网络中的 config .xml， *MBeanUtilsInitSingle 服务是*App 马纳格服务器的 1 个 init-param，

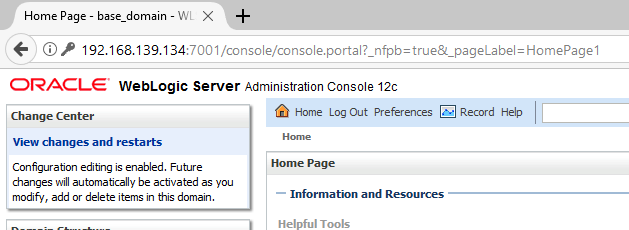


应用程序马拉格服务被地图到以下模式网址：

* /appmanager/\*
* \*.portlet
* \*.部分
* \*.门户

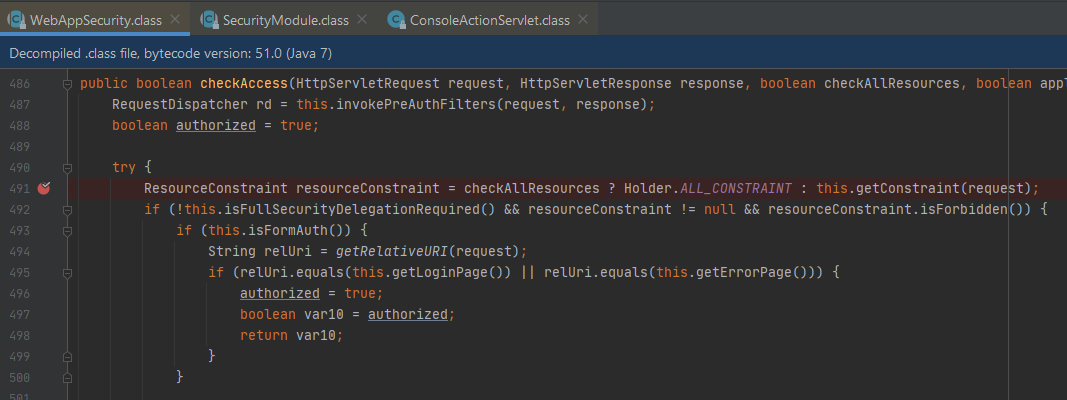


其中包括“*/控制台/控制台.portal*”，在上线后立即可以看到：



...

挖掘了一番后，也找到了逻辑处理的经典，决定了这个回溯应该是 authen or 非 authen 重新定义，在 Weblogic 12.2.1.3 中是经典 ***Weblogic.servlet.security.internal.WebAppSecurity.检查Access（）***：



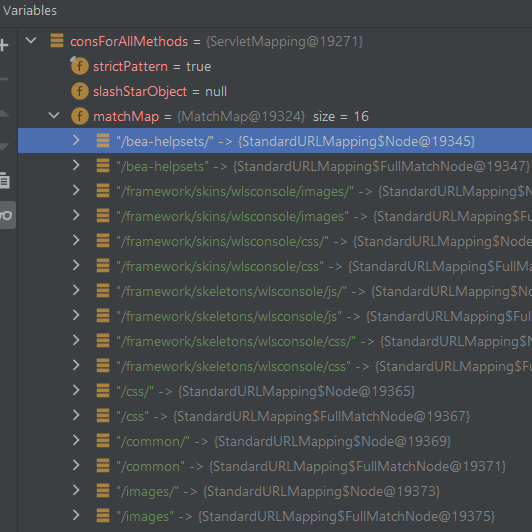
在这里，***WebAppSecurityWLS.getConstraint（回复）***将获取当前回复的对接，每个信条都包含以下信息：



其中，如果康斯特林有*未恢复的flag=true*，则该回复将决定为乌努特，否则将返回上线页面。

继续 F7 进入 ***WebAppSecurityWLS.getConstraint（）***，在这里可以获得所有转换器，根据它，哪些网址可以跳过身份验证部分，删除的模式 URL 列表如下：

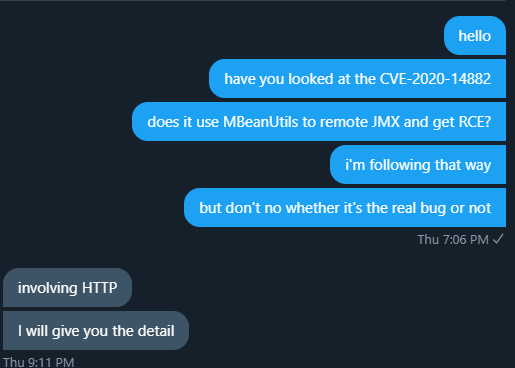
* */bea-helpsets/\**
* */framework/skins/wlsconsole/images/\**
* */framework/skins/wlsconsole/css/\**
* */framework/skeletons/wlsconsole/js/\**
* */framework/skeletons/wlsconsole/css/\**
* */css/\**
* */常见/\**
* */图片/\**



依靠这些模式列表，并尝试添加以前的“通行证UU”技巧，例如：“*..;/，/#/./*“，但仍然没有找到任何好...

来到这里，我一直叽叽喳喳的，想不出什么新东西！

在推特上，没见任何人发过PoC，只好丢下所有的廉耻去向母国兄弟们求信！（请允许匿名此人）

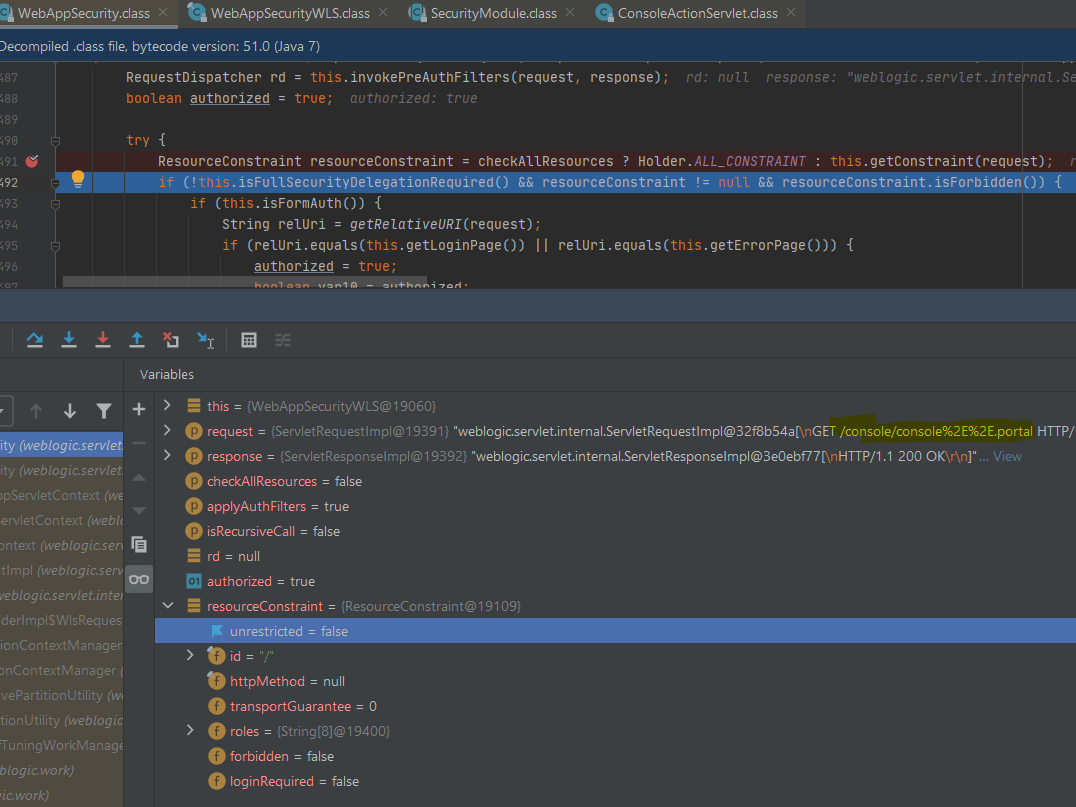


是的， 然后他给我一张 Poc 的皮， 这是善良和慷慨的！

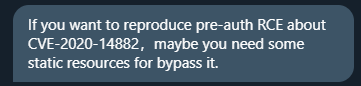


Trong đó thì phần abuse HandleFactory đã biết rồi, còn phần url kia lại là: “*/console/console%2E%2E.portal*” A.K.A “*/console/console…portal*”

Mình đem thử thì thấy ko đúng lắm, với url “*/console/console%2E%2E.portal*” thì constraint trả về vẫn là “**unrestricted=false**” nghĩa là vẫn cần authen mới qua được:



Hỏi lại thanh niên kia thì hắn mới bảo như vậy:



PoC hắn gửi ban đầu chỉ là cái hint mà thôi, để bypass thì cần 1 chút trick, abuse 1 vài resource nào đó mới có thể trở thành unauth RCE được!

Loanh quanh luẩn quẩn lại quay trở về với vị trí ban đầu,

Khi đó thì mình cũng khá là nản rồi, đành kêu thêm mấy đàn em vào ngâm cứu chung cho đỡ. Lại 1 chút flashback về ngày còn chơi CTF, ngày đó cứ mỗi khi stuck lại đem bàn giao idea cho đồng đội để san sẻ suy nghĩ, và hiệu quả lúc nào cũng cao hơn là 1 mình tự chơi …

Nhóm này thì cũng chỉ có 3 người: mình,

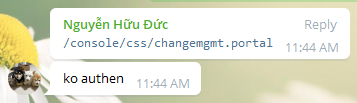
[PeterJson](https://medium.com/u/9d0bc610db35?source=post_page-----6e4b09981dbf--------------------------------)

 và @Quynh Le. Chuyên đâm chọt các loại bug xảy ra trên java

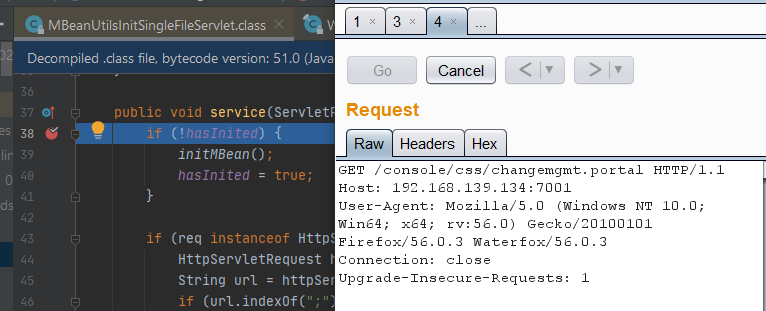


然后，德国年轻人找到了需要看到的东西=）），一个网址可以特里格格***姆贝努蒂尔斯尼特辛格尔服务，***而无需什么：

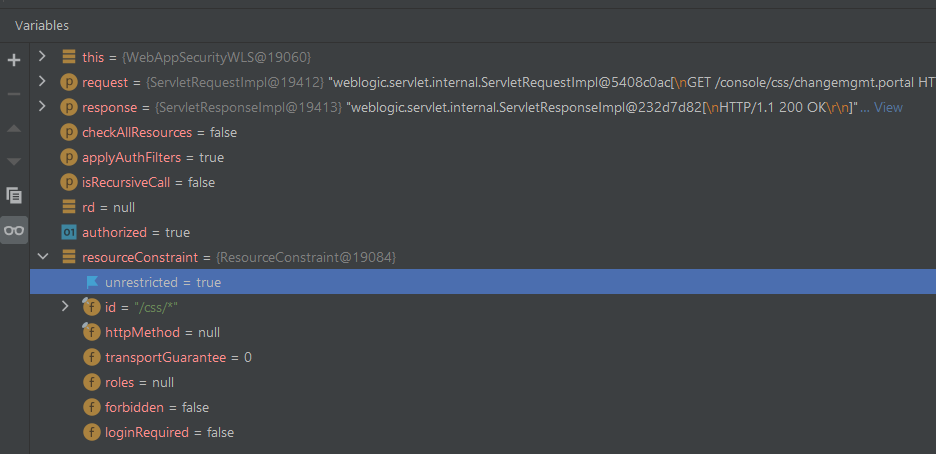
/console/css/changemgmt.portal



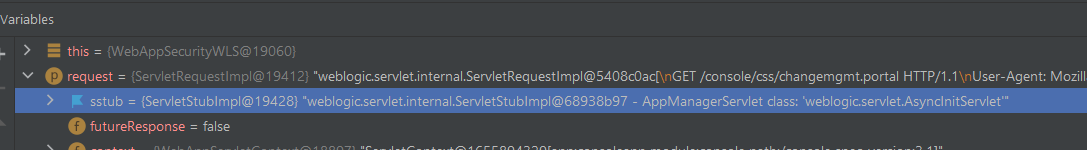
我也有点惊讶地看到这个回复， 但测试它真的可以特里格...



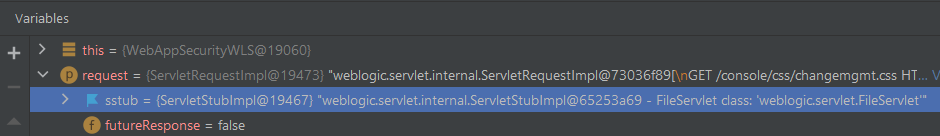
在 debugger 中再次检查， 康斯特林特是地图进入 “/css/” 应该被**真正的不复出，**



更令人惊讶的是，当再次检查这个内在点的伺服手时，发现***阿西尼特服务是***手，而不是我想的***文件服务***！



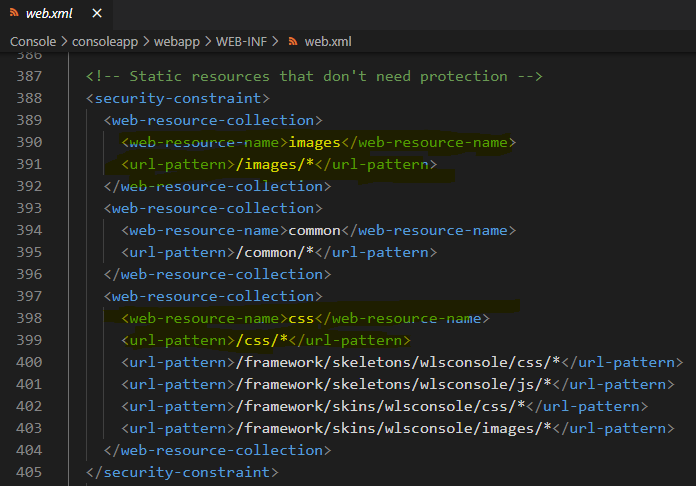
与传统的回声（有.css尾）相比，**文件Servlet**将手动回复此。



现在我以为文件Servlet有什么魔力， 所以一直埋头查找它的错误 :(...

但事实上，魔术并不在那里...它位于网络.xml！

回顾网络资源声明部分.xml：



其中：

* 网址模式 “*/common/\**” 由 *JSPCServlet*处理 （手动回信到 jsp 文件）
* “*/帧/\**”模式网址由*文件服务*处理
* ...

但是，对于*“****/图像/\*****”*和*“****/css/\*****”*的模式 URL，没有申报任何服务器将手动！

因此，当回复与 URL “***/console/css/aaasdasd.portal***”时，***AppManagerServlet***将手动回复此。

简言之：：

* “*/css/\**” 到 bypass authen
* “*\*.portal*” 到特里格

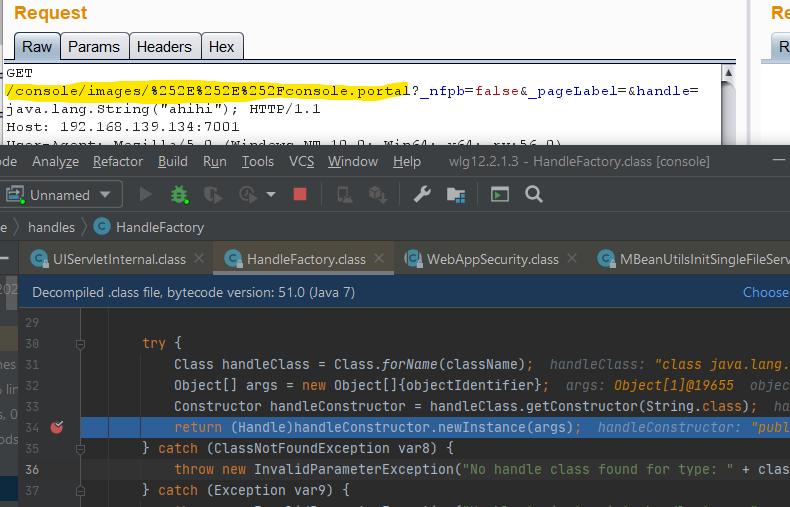
在这里，可以重新获得***MBeanUtils尼特辛格尔服务服务.服务（）***，而无需再奥森

然而，request 仍然无法回到***手部.getHandle（），*** 因为网址模式尚未与波特莱特匹配！

就在这时，根据那艘船青年寄出的原有的hint：

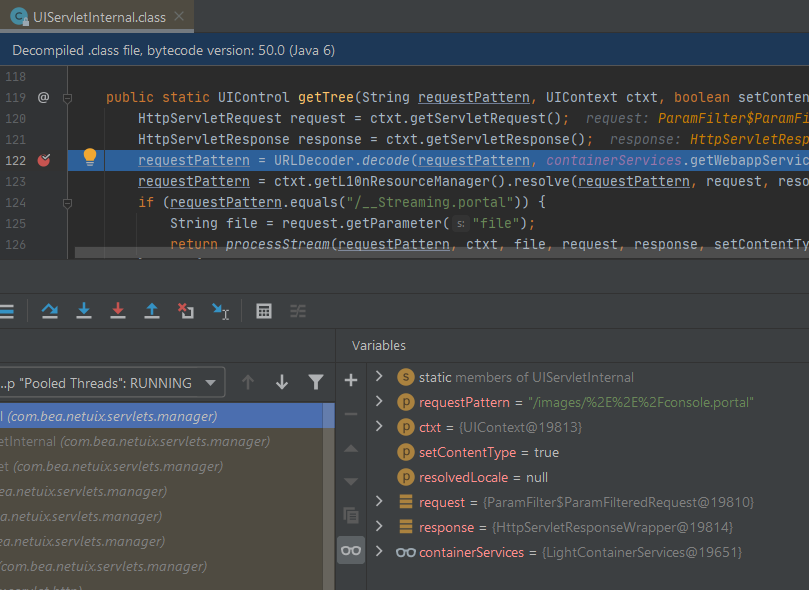


所以在这里标记“.”，是想用“.”双恩科德来发送bug，再试一次，这是真的：



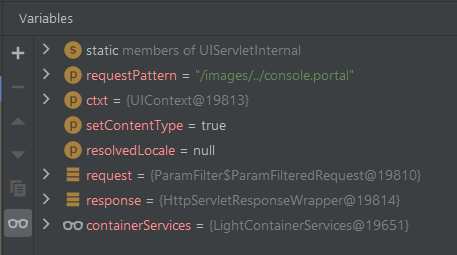
来到这里，已经可以写完POC了，但我请允许进一步分析这段，

在通过 MBeanUtilsInitSingleServlet.服务（） 和添加一些长段后，在 UIServletInternal.getTree（）中，URL 模式再次被剥离并再次执行 URL 1 分ode：



在去码之前

在去码后：



因此，它解释了为什么使用双 encoded url 可以同时通过原始 URL 的 normalized 处理，这仍然可以使以后的伺服处理段正常工作！

**#Trigger RCE**

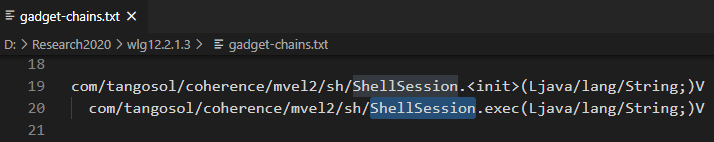
在乞丐PoC中，这个年轻人使用 ***com.bea.core.重新包装.springframework.context.支持.FileSystemXml应用程序测试（）***来发送 RCE，但这个链条需要外挂。

我是一个完美的人， 而且在实践中， 相当多的服务器阻止了外挂， 所以我决定找到一个新的瓶子， 没有外挂， RCE 在一个打击！🤣

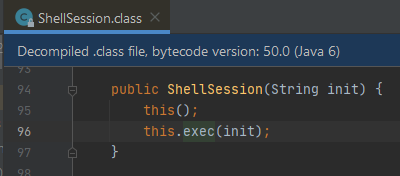
在利费雷的一天，我编辑了很多加吉特Inspector工具，用于查找代码。

到了这个时候，它又发挥了作用，把它拉出来，编辑了一些代码中的条件，以适应新的 context， 运行约 5p， 结果相当不错。

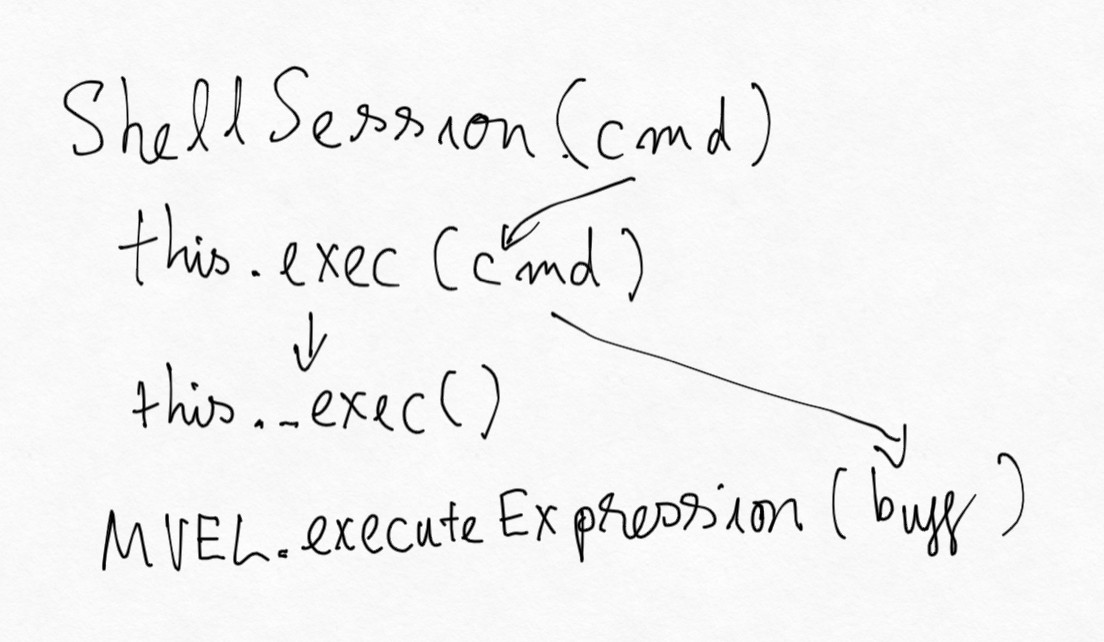
其中有一个相当有潜力的瓶子：



这个链在从建筑器收到 ARG 后，将调用*壳牌会话.exec（）* 来执行此 ARG：



*壳牌会话.exec（）*后面的链条很长，为了能够执行，我总结如下：



因此，可以任意执行 MVEL 快递（关于 MVEL，我似乎在前几篇文章中提到过）。

*事实上，直到POC成功与这个新瓶，我并没有完全惊讶，不相信有东西可以利用。我不确定这是否是代码中留下的后门 （））。*

在结合了上述所有障碍后，加入一点盐酱，我写了一个POC执行命令，然后有一个回复=））。POC最终由德国完成，而不是自己，

PoC 视频：<https://youtu.be/JFVDOIL0YtA>

=））谁需要POC，就在这里自己研究，我这里没有现成的食物。