系统安全实验六

19210204 尹畅

2024 年 6 月 26 日

1 实验目的

熟悉原始套接字编程
了解网络传输底层协议

2 实验内容

在 WINDOWS 环境下实现基本的 PING 程序（回送测试功能）

Ping 程序是用来探测主机到主机之间是否可通信，如果不能 Ping 到某台主机，表明不能和这台主机建立连接。Ping 使用的是 ICMP 协议，它发送 ICMP 回送请求报文给目的主机。ICMP 协议规定：目的主机必须返回 ICMP 回送应答报文给源主机。如果源主机在一定时间内收到应答，则认为主机可达。

3 实验步骤

定义 IP 报头数据结构和 ICMP 报头数据结构。
定义回送请求数据包和回送应答数据包的数据结构。
使用 WSAStartup 函数初始化 Windows 协议栈。
使用 socket 函数创建原始套接口。
使用 gethostbyname 函数根据主机名查询主机 IP 地址。
填充回送请求信息，计算校验和。
调用 sendto 函数发送 ICMP 回送请求报文。

使用 select 函数查询套接口的状态。

当目标主机应答后，调用 recvfrom 接收 ICMP 回送应答报文。

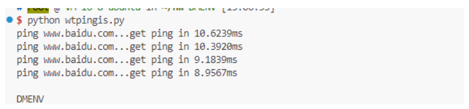
比较 ICMP 回送请求报文和 ICMP 回送应答报文判断到目标主机的连通性，给出提示信息。

使用 closesocket 关闭原始套接口。

4 代码实现

```python
import os, sys, socket, struct, select, time

if sys.platform == "win32":
    # On Windows, the best timer is time.clock()
    default_timer = time.clock
else:
    # On most other platforms the best timer is time.time()
    default_timer = time.time

# From /usr/include/linux/icmp.h; your milage may vary.
ICMP_ECHO_REQUEST = 8


def checksum(source_string):
    """
    I'm not too confident that this is right but testing seems
    to suggest that it gives the same answers as in_cksum in
        ↪ ping.c
    """
    countTo = (int(len(source_string) / 2)) * 2
    sum = 0
    count = 0

    # Handle bytes in pairs (decoding as short ints)
    loByte = 0
    hiByte = 0
    while count < countTo:
        if (sys.byteorder == "little"):
            loByte = source_string[count]
            hiByte = source_string[count + 1]
        else:
            loByte = source_string[count + 1]
            hiByte = source_string[count]
        sum = sum + (hiByte * 256 + loByte)
        count += 2

    # Handle last byte if applicable (odd-number of bytes)
    # Endianness should be irrelevant in this case
    if countTo < len(source_string): # Check for odd length
        loByte = source_string[len(source_string) - 1]
        sum += loByte

    sum &= 0xffffffff # Truncate sum to 32 bits (a variance from ping.c, which
                      # uses signed ints, but overflow is unlikely in ping)

    sum = (sum >> 16) + (sum & 0xffff) # Add high 16 bits to low 16 bits
    sum += (sum >> 16) # Add carry from above (if any)
    answer = ~sum & 0xffff # Invert and truncate to 16 bits
    answer = socket.htons(answer)

    return answer


def receive_one_ping(my_socket, ID, timeout):
    """
    receive the ping from the socket.
    """
    timeLeft = timeout
    while True:
        startedSelect = default_timer()
        whatReady = select.select([my_socket], [], [], timeLeft)
        howLongInSelect = (default_timer() - startedSelect)
        if whatReady[0] == []: # Timeout
            return

        timeReceived = default_timer()
        recPacket, addr = my_socket.recvfrom(1024)
        icmpHeader = recPacket[20:28]
        type, code, checksum, packetID, sequence = struct.unpack(
            "bbHHh", icmpHeader
        )
        # Filters out the echo request itself.
        # This can be tested by pinging 127.0.0.1
        # You'll see your own request
        if type != 8 and packetID == ID:
            bytesInDouble = struct.calcsize("d")
            timeSent = struct.unpack("d", recPacket[28:28 + bytesInDouble])[0]
            return timeReceived - timeSent

        timeLeft = timeLeft - howLongInSelect
        if timeLeft <= 0:
            return


def send_one_ping(my_socket, dest_addr, ID):
    """
    Send one ping to the given >dest_addr<.
    """
    dest_addr = socket.gethostbyname(dest_addr)

    # Header is type (8), code (8), checksum (16), id (16),
        ↪ sequence (16)
    my_checksum = 0

    # Make a dummy heder with a 0 checksum.
    header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0,
        ↪ my_checksum, ID, 1)
    bytesInDouble = struct.calcsize("d")
    data = (192 - bytesInDouble) * "Q"
    data = struct.pack("d", default_timer()) + data.encode()

    # Calculate the checksum on the data and the dummy header.
    my_checksum = checksum(header + data)

    # Now that we have the right checksum, we put that in. It'
        ↪ s just easier
    # to make up a new header than to stuff it into the dummy.
    header = struct.pack(
        "bbHHh", ICMP_ECHO_REQUEST, 0, socket.htons(my_checksum
            ↪ ), ID, 1
    )
    packet = header + data
    my_socket.sendto(packet, (dest_addr, 1)) # Don't know
        ↪ about the 1


def do_one(dest_addr, timeout):
    """
    Returns either the delay (in seconds) or none on timeout.
    """
    icmp = socket.getprotobyname("icmp")
    try:
        my_socket = socket.socket(socket.AF_INET, socket.
            ↪ SOCK_RAW, icmp)
    except socket.error as e:
        errno, msg = e.args
        if errno == 1:
        # Operation not permitted
            msg = msg + (
                "␣-␣Note␣that␣ICMP␣messages␣can␣only␣be␣sent␣
                    ↪ from␣processes"
                "␣running␣as␣root."
            )
            raise socket.error(msg)
        raise # raise the original error

    my_ID = os.getpid() & 0xFFFF

    send_one_ping(my_socket, dest_addr, my_ID)
    delay = receive_one_ping(my_socket, my_ID, timeout)

    my_socket.close()
    return delay


def verbose_ping(dest_addr, timeout = 2, count = 4):
    """
    Send >count< ping to >dest_addr< with the given >timeout<
        ↪ and display
    the result.
    """
    for i in range(count):
        print("ping␣%s..." % dest_addr, end="")
        try:
            delay = do_one(dest_addr, timeout)
        except socket.gaierror as e:
            print("failed.␣(socket␣error:␣'%s')" % e[1])
            break

        if delay == None:
            print("failed.␣(timeout␣within␣%ssec.)" % timeout)
        else:
            delay = delay * 1000
            print("get␣ping␣in␣%0.4fms" % delay)
    print()


if __name__ == '__main__':
    verbose_ping("www.baidu.com␣")
    verbose_ping("google.com")
    verbose_ping("a-test-url-taht-is-not-available.com")
    verbose_ping("192.168.1.1")
```

5 实验结果



```
$ python wtpingis.py
ping www.baidu.com...get ping in 10.6239ms
ping www.baidu.com...get ping in 10.3920ms
ping www.baidu.com...get ping in 9.1839ms
ping www.baidu.com...get ping in 8.9567ms

DMENV
```

Enter Caption