**期 末 作 业 实 验 报 告**

1. **实验名称：**端口扫描编程（安装SSL并测试443端口）
2. **实验内容：**

IP地址端口扫描；在Windows环境下，使用多线程进行端口扫描，并用图形化界面进行用户交互

Windows Server 2016安装SSL并测试其443端口是否开放

1. **实验原理：**

使用socket创建套接字，利用三次握手协议来判断目标端口是否存活。如果connect连接成功，说明该端口开启了TCP监听(LISTENING或者LINSTEN)，否则说明没有开启。

1、扫描端向目标端发送SYN请求建立连接

2、目标端收到请求后，回复ACK同意连接并同意发送SYN请求建立连接

3、扫描端收到后，发送ACK同意，此时三次握手完成，以此来判断端口是否存活



多线程扫描，将需要扫描的端口号，放到队列中，扫描的线程每次从队列中取出一个端口，进行connect操作，判断是否端口开放。单个线程执行该过程，后面的端口需要等待前面的端口扫描结束，造成时间的浪费；使用多线程执行对单个端口号的connect，缩短程序运行时间。

软件图形化界面，每次多线程端口扫描执行结束后，将开放端口和执行时间写入图形化界面，同时图形化界面准备进行下一次执行端口扫描。

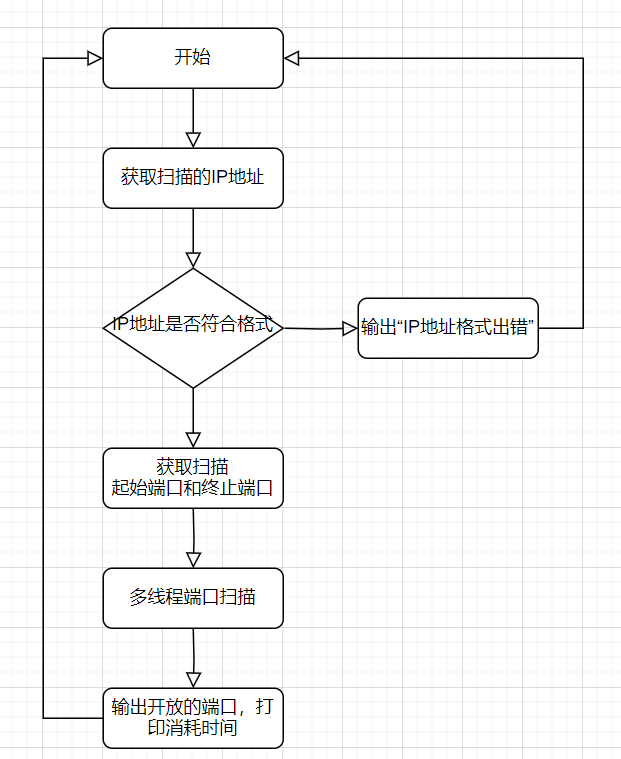
1. **设计思路**

端口扫描流程：

获取扫描IP地址，起始端口号和终止端口号

执行多线程端口扫描；每个线程执行connect操作，如果连接成功，说明该端口开放

输出开放的端口，重新回到初始状态，等待下一次扫描

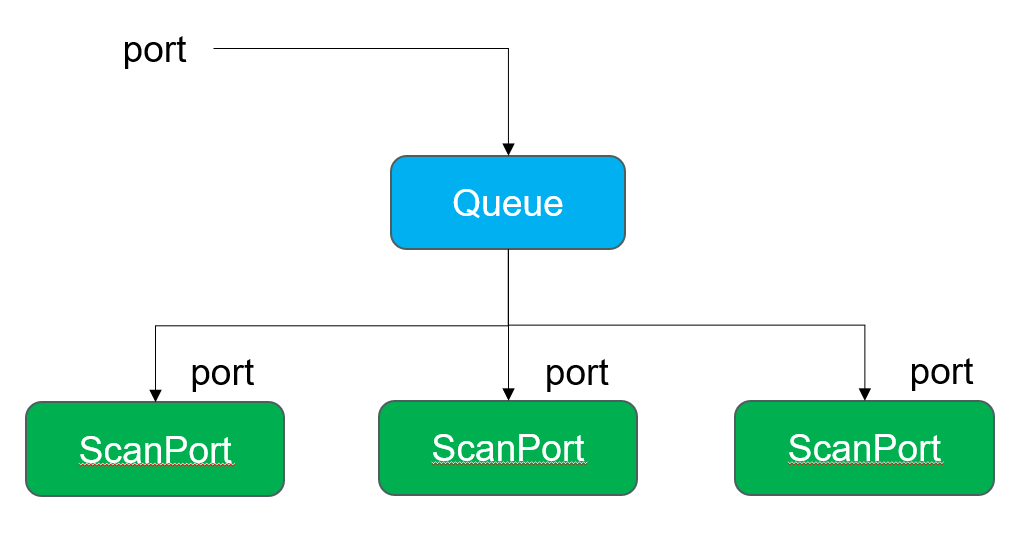


多线程扫描流程：

将所有的端口都放到队列中

每个线程从队列中取出一个端口号，进行connect连接；

如果端口开放，将端口信息输出到屏幕



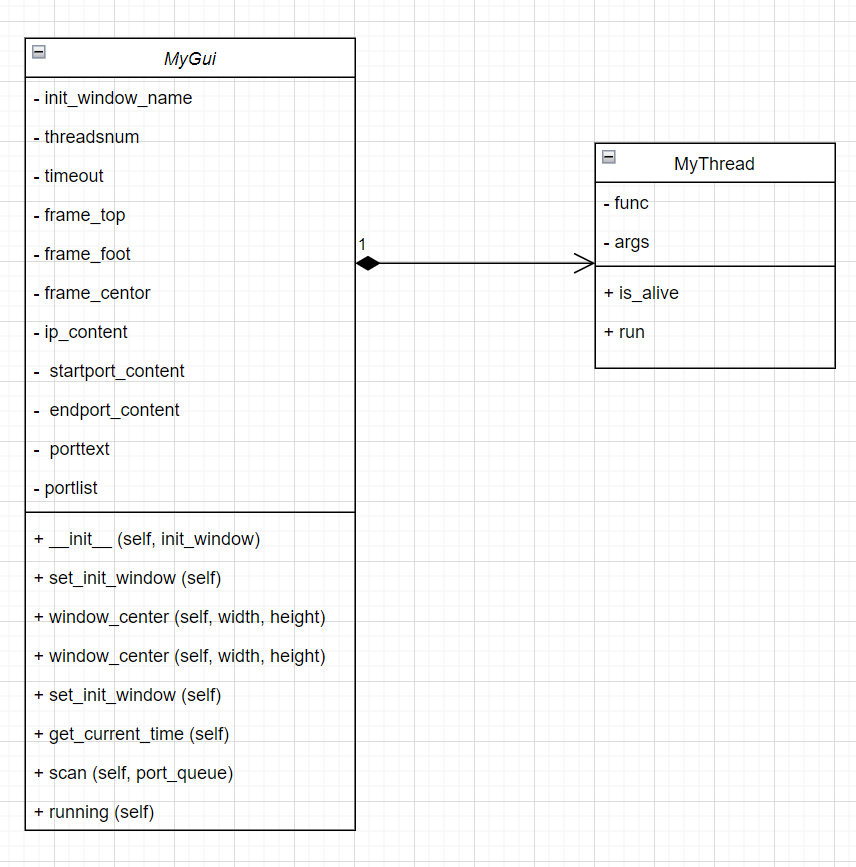
图形化界面类

在开始按钮绑定多线程端口扫描程序running；running中执行获取参数，使用MyThread进行多线程扫描，并打印输出结果；

MyThread类中is\_alive方法，检测当前线程是否已经全部执行完，用于统计执行时间；

scan方法是真正执行的端口扫描方法，每次从队列中取出需要检测的端口，使用参数中的IP地址和端口号进行connect，判断端口是否开启。

MyGui类中porttext是打印结果的位置；threadsum是开启的线程数，portlist是从输入的起始端口和终止端口计算的需要检测端口的列表，需要将其插入队列。



关键代码：

多线程端口扫描：

1. *# 扫描函数；connect判断TCP是否开放此端口*
2. *# 多线程从端口队列中取出端口，进行扫描*
3. def scan(self, port\_queue):
4. *# 不断循环直到port队列为空*
5. while True:
6. if port\_queue.empty():
7. break
8. ip = self.ip
9. port = port\_queue.get()
10. timeout = self.timeout
11. *# 创建socket对象，connect连接端口*
12. try:
13. s = socket.socket()
14. s.settimeout(timeout)
15. s.connect((ip, port))
16. string = "Port " + str(port) + " is OPEN\n"
17. self.porttext.insert(END, string)
18. self.porttext.see(END)
19. except Exception as e:
20. pass
21. finally:
22. s.close()
23. *# 多线程类，将执行函数作为一个对象；将端口列表传入参数*
24. class MyThread(threading.Thread):
25. def \_\_init\_\_(self, func, \*args):
26. super().\_\_init\_\_()
27. self.func = func
28. self.args = args
29. self.setDaemon(True)
30. self.start()  *# 在这里开始*
31. def is\_alive(self) -> bool:
32. return super().is\_alive()
33. def run(self):
34. self.func(\*self.args)
35. *# 开始扫描按钮的事件*
36. def runing(self):
37. self.porttext.delete(1.0, END)
38. *# 判断ip格式是否正确*
39. if re.match(r"^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$", self.ip\_content.get()):
40. self.ip = self.ip\_content.get()
41. else:
42. msgbox.showerror(title="Error", message="ip格式错误，格式为：x.x.x.x")
43. return
45. *# 获取端口范围*
46. startport = self.startport\_content.get()
47. endport = self.endport\_content.get()
48. self.porttext.insert(END, "当前时间为： "+self.get\_current\_time()+"\nIP地址："+self.ip+"\n端口列表："\
49. +self.startport\_content.get()+"-"+self.endport\_content.get()+"\n开始扫描\n")
51. self.portslist = list(range(int(startport), int(endport) + 1))
52. *# 将端口列表放入队列中*
53. port\_queue = queue.Queue()
54. for port in self.portslist:
55. port\_queue.put(port)
56. threads = []
57. *# 多线程执行扫描函数，设置线程数为800*
58. start = time.time()
59. for \_ in range(0, self.threadsnum):
60. thread = self.MyThread(self.scan, port\_queue)
61. threads.append(thread)
62. *# 等待线程结束*
63. while 1:
64. for thread in threads:
65. if thread.is\_alive():
66. time.sleep(0.0001)
67. break
68. end = time.time()
69. spent = str(end-start)
70. spent = spent[:spent.find('.')+3]
71. self.porttext.insert(END, "扫描结束\n")
72. *# 打印扫描时间*
73. self.porttext.insert(END, "\nTime Cost: "+str(spent)+"\n\n")
75. **实验器材（设备、元器件）**

Python3.8.4；

Windows 11主机；

Windows Server 2016虚拟机；

1. **实验步骤**

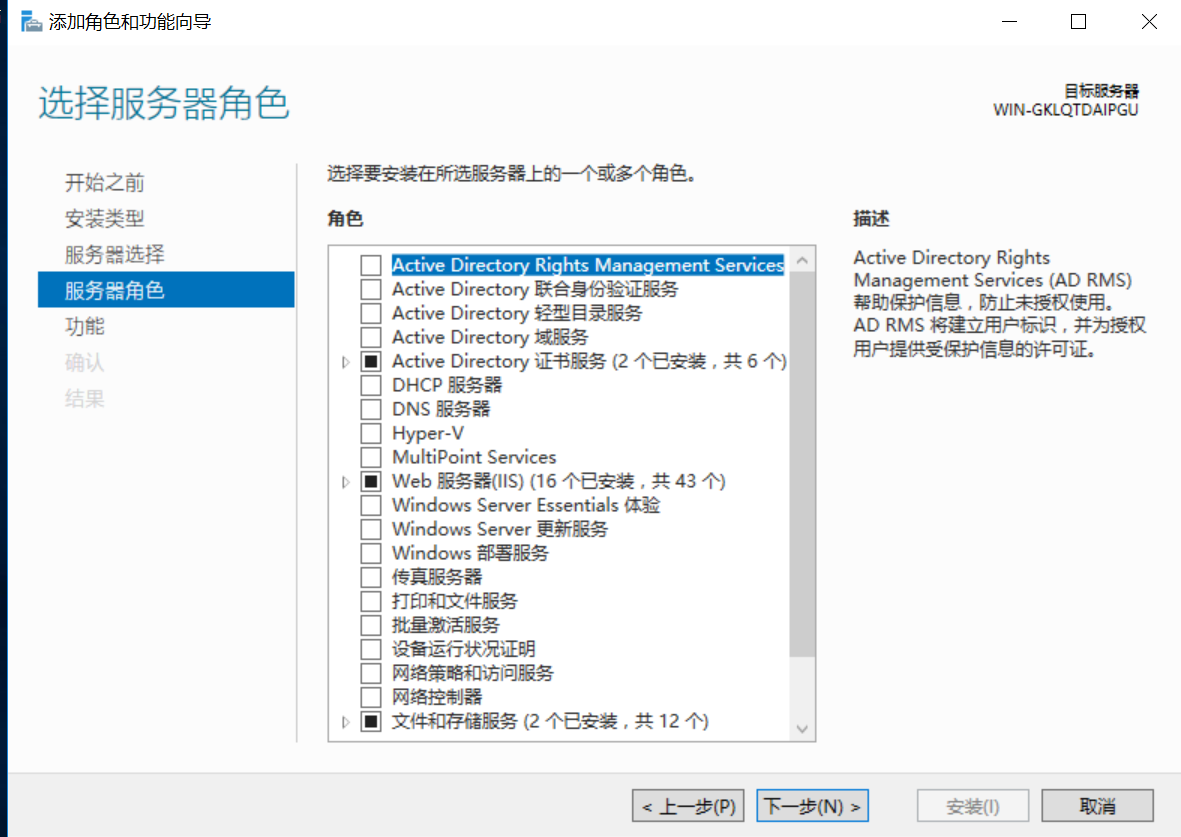
虚拟机：

1. 安装IIS服务和证书服务
2. 启动IIS服务和证书服务
3. 服务器证书申请、颁发和安装
4. 在服务器上配置SSL
5. 客户端通过SSL与服务器建立连接

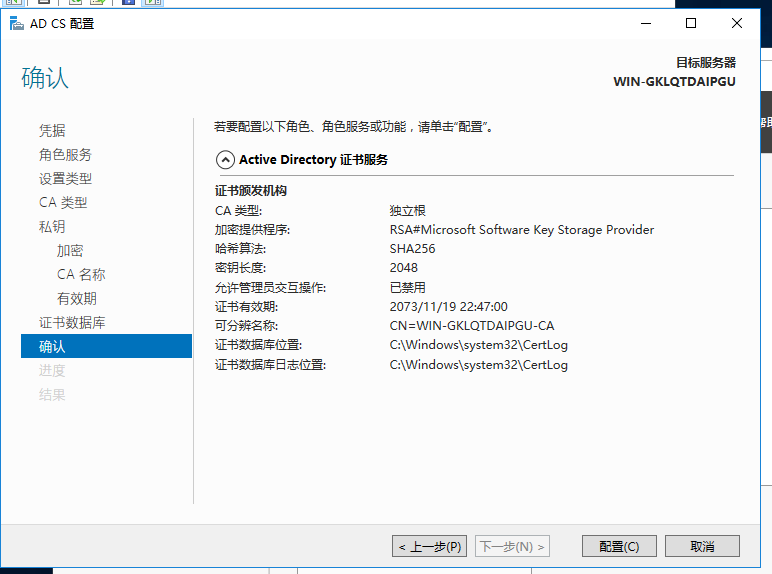
主机：

运行多线程端口扫描程序；将IP地址设置为虚拟机IP：192.168.149.130；起始端口为1，终止端口为500；查看屏幕是否显示443端口开放。

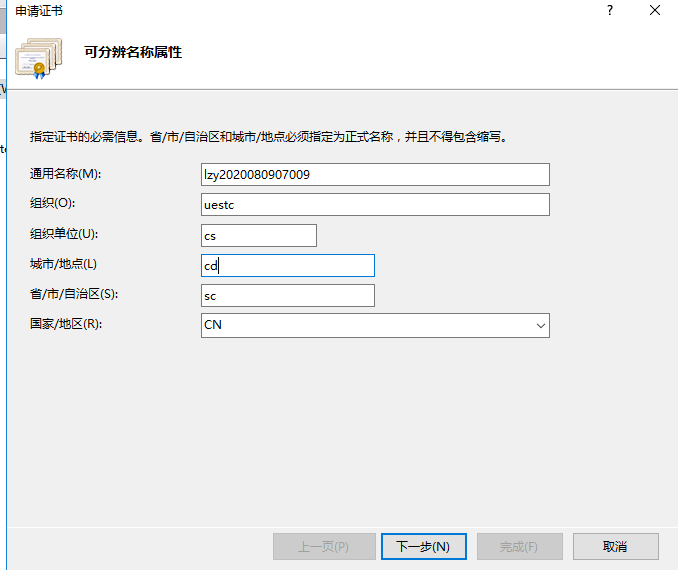
1. **实验数据及结果分析**
2. 安装IIS服务和证书服务

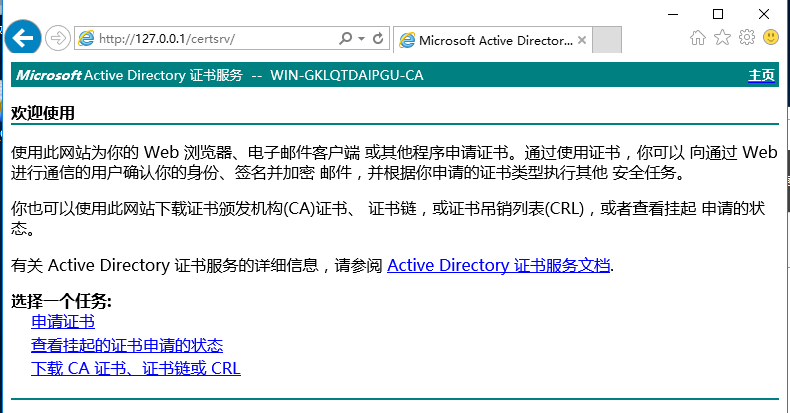


1. 配置证书服务器

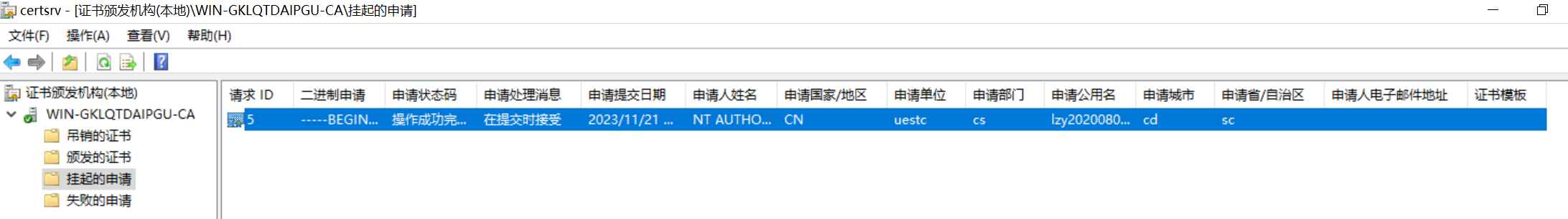


1. 为服务器创建新的证书申请

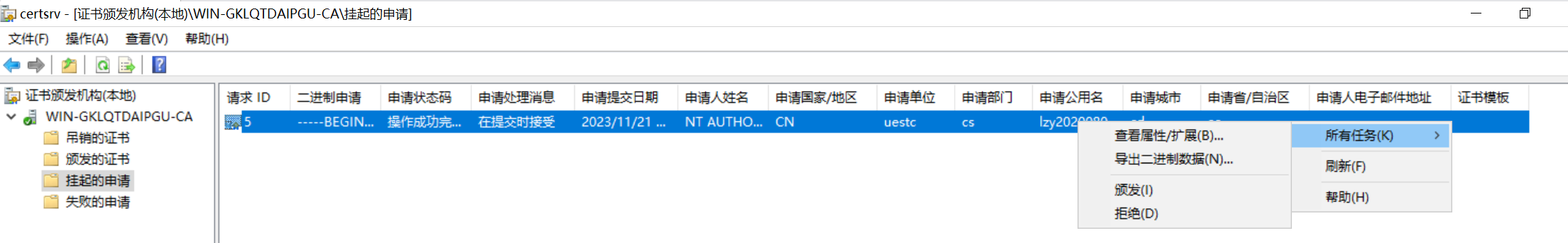




1. 挂起的申请

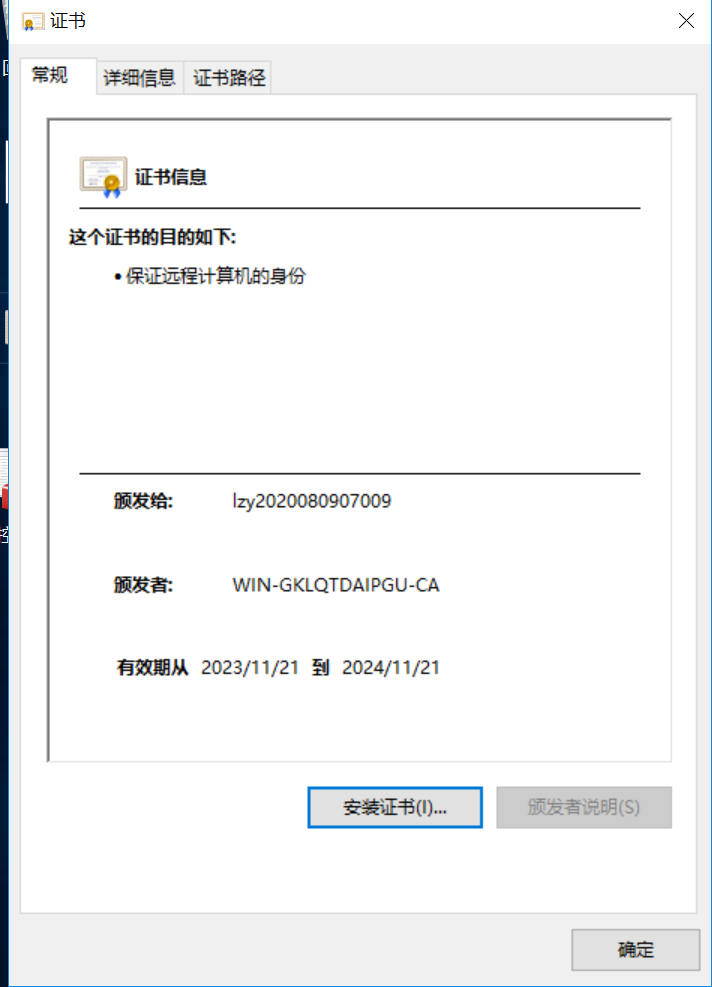


1. 颁发申请并将证书下载

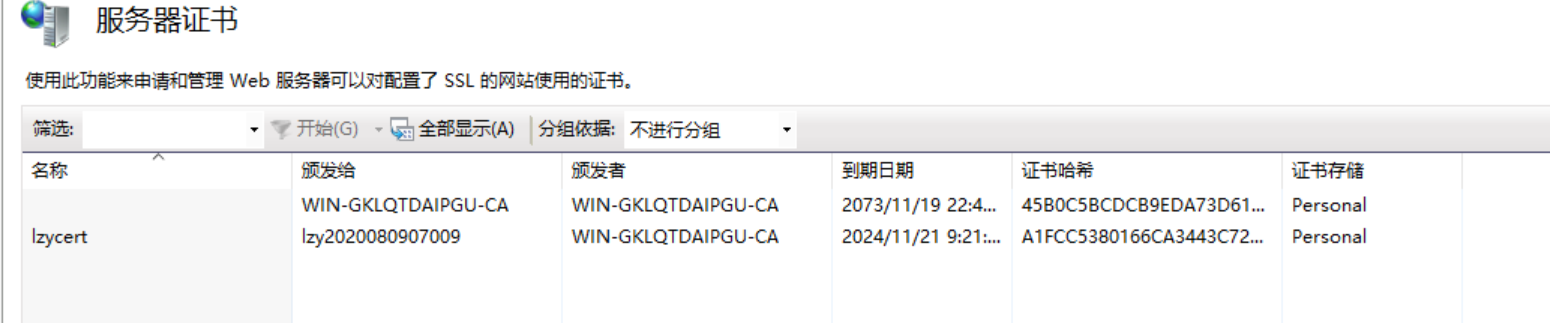




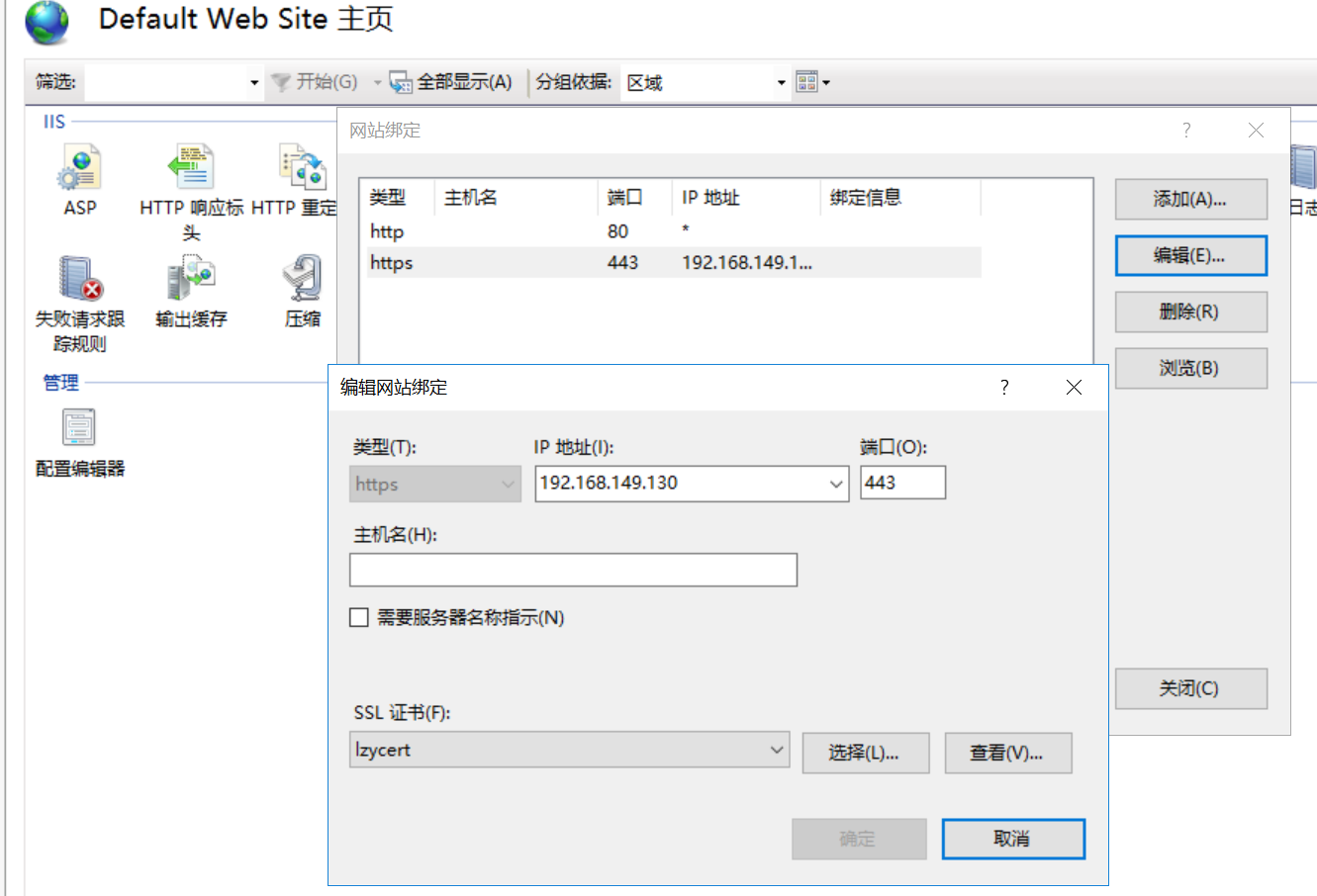
1. 根CA和服务器CA (lzycert)

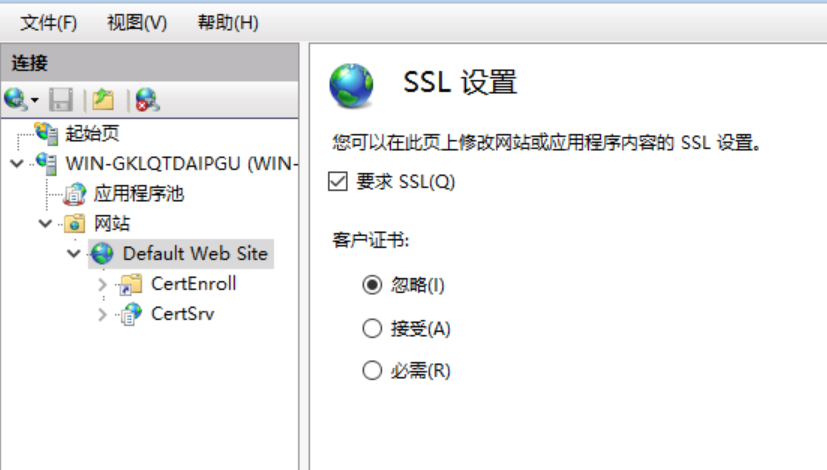
 

1. 完成服务器证书lzycert的申请



1. 默认网站绑定证书lzycert并设置SSL



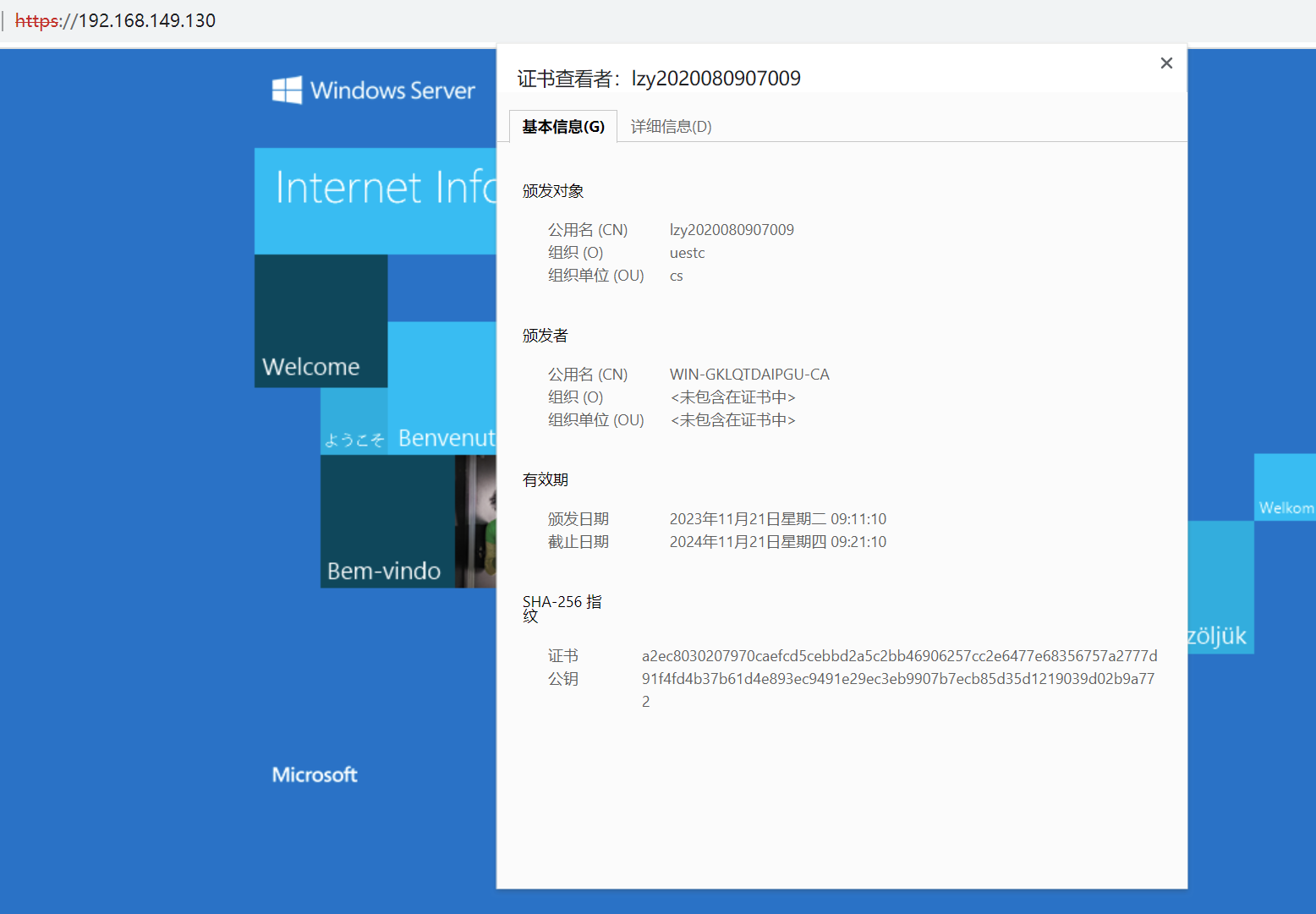


1. 默认网站绑定证书lzycert并设置SSL

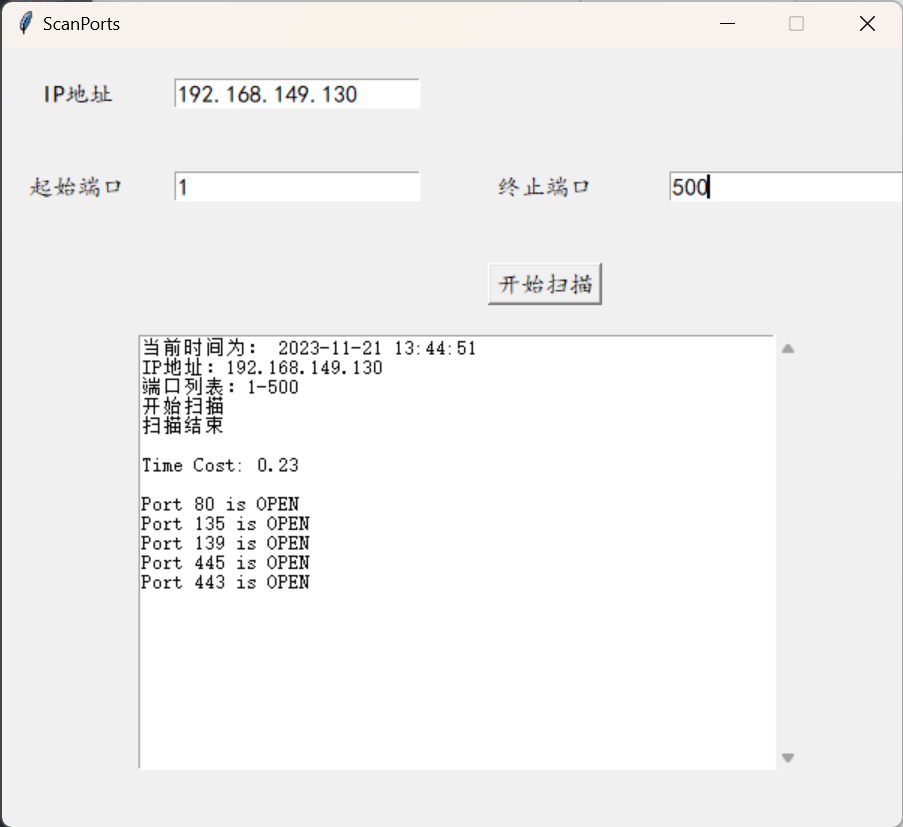
http访问拒绝



https访问接受



1. Windows server 2016安装SSL后，测试发现443端口已经开放



1. **实验结论、心得体会：**

经过测试，Windows Server 2016开启了SSL服务后，TCP端口默认打开；我们的端口扫描程序可以检测到已经打开的端口。

通过这次端口扫描实验的编程，我对课上的OpenSSL协议有了更深的认识；同时学习了socket套接字编程和多线程编程，通过socket中的connect操作来判断系统中TCP端口是否开启，同时利用多线程提高程序运行速度，提高了我的动手能力。

但是这个扫描采用conncet，容易被发现，可以考虑构造虚假IP的SYN包，通过接收的回复来确定端口是否连通；而且部分端口可能对UDP开放，也应该增加对UDP端口的扫描。