

**中国矿业大学**

**面向对象程序设计**

**课 题： 扫描器**

**学 院： 计算机科学与技术学院**

**班 级： 信息安全12-2班**

**组 长： 刘 铭**

**组 员： 起 文 屈云轩**

**指导 教师： 鲍 宇**

**2015年 10 月 16 日**

目录

[一、问题定义 2](#_Toc432755184)

[二、项目可行性 2](#_Toc432755185)

[1. 经济可行性 2](#_Toc432755186)

[2. 技术可行性 2](#_Toc432755187)

[3. 运行可行性 2](#_Toc432755188)

[4. 法律可行性 2](#_Toc432755189)

[三、需求分析 2](#_Toc432755190)

[1. 系统介绍 2](#_Toc432755191)

[2. 功能需求 2](#_Toc432755192)

[3. 性能需求 2](#_Toc432755193)

[4. 运行需求 2](#_Toc432755194)

[5. 未来可能出现的问题 2](#_Toc432755195)

[6. 系统用例图 2](#_Toc432755196)

[7. 分析用例图 2](#_Toc432755197)

[7.1 图形界面 2](#_Toc432755198)

[7.2 扫描器 2](#_Toc432755199)

[8 系统活动图 2](#_Toc432755200)

[9 开发平台 2](#_Toc432755201)

[9.1网络部分 2](#_Toc432755202)

[9.2 编程语言 2](#_Toc432755203)

[9.2 图形界面 2](#_Toc432755204)

[9.4 开发平台总结 2](#_Toc432755205)

[四、概要设计 2](#_Toc432755206)

[1. 软件结构 2](#_Toc432755207)

[2 层次关系 2](#_Toc432755208)

[3 图形界面 2](#_Toc432755209)

[4扫描服务 2](#_Toc432755210)

[五、详细设计 2](#_Toc432755211)

[1 CMFCApplication4Dlg类 2](#_Toc432755212)

[1.1类图 2](#_Toc432755213)

[1.2类说明 2](#_Toc432755214)

[2. TCPScanner类 2](#_Toc432755215)

[2.1 类图 2](#_Toc432755216)

[2.2 类说明 2](#_Toc432755217)

[3 序列图 2](#_Toc432755218)

[4.序列图分析 2](#_Toc432755219)

[六、编码 2](#_Toc432755220)

[七、测试 2](#_Toc432755221)

[八、总结 2](#_Toc432755222)

# 一、问题定义

扫描器是一种自动扫描本地或远程计算机端口并给出开放端口的程序。有些扫描器可以比较准确的确定端口运行的服务类型和软件及版本、操作系统类型及版本，还有一些强大的扫描器可以扫描系统和软件漏洞，甚至进行简单的网络攻击。系统管理员和黑客可以利用扫描器确定系统开放端口和运行的服务，以便对系统安全性进行配置或收集足够的信息发动攻击。

本次要做的扫描器为简单的扫描器，可以扫描指定ip计算机端口，并给出指定范围开放的端口号，具有一定的操作系统识别能力。

# 二、项目可行性

## 1. 经济可行性

该软件代码编写测试在正常的教学时间内，除了必要的电费、网费并不产生其他费用，而且该软件也不以营利为目的，所以在经济上可行。

## 2. 技术可行性

该软件需要输入目标主机的IP地址和要扫描的端口范围，扫描器尝试连接目标主机指定范围内的每一个端口，根据目标系统返回的数据包判断目标系统端口是否开放，生成扫描报告，反馈给用户。

该软件需要处理用户输入，与目标主机建立网络连接或向目标主机发送非标准的数据包，接收目标主机返回的数据包并解析数据包以此判断目标主机相应端口是否开放，根据端口开放情况生成扫描报告。该软件可以构造数据包，利用不同操作系统协议栈的差别识别部分操作系统。

## 3. 运行可行性

该软件在Windows桌面版的环境下运行，不需要很多计算资源，普通PC即可运行。从这一点上运行完全可行。

## 4. 法律可行性

该软件用于查看指定主机开放端口，不具有攻击性，禁止非法使用，法律上完全可行。

# 三、需求分析

## 1. 系统介绍

该系统是为了使管理员了解网络中的主机开放端口的情况帮助使管理员做出决策。

## 2. 功能需求

该系统是一个简易的网络扫描器，可以扫描指定IP地址范围和指定端口范围内主机开放的端口。

为了避免触发防火墙警告，对扫描的端口进行乱序处理。

为了避免占用过多的计算资源，对同时扫描的线程数进行限制。

系统需要向使用者提供扫描报告，报告目标系统端口开放情况。

## 3. 性能需求

该系统为简单的网络扫描器，需要提供比较强的可靠性，在速度上为避免触发对方系统防火墙和IDS系统的警告可以不提供过于高速的扫描速度。

## 4. 运行需求

该系统运行于Windows系统之上，兼容win7系统。需要使用操作系统的网络资源，可以建立套接字，访问原始数据包。提供比较友好的使用界面。

## 5. 未来可能出现的问题

应该保留相关的接口，未来可以方便的添加基于IPv6的扫描功能，提供完善的服务指纹识别、系统指纹识别和漏洞扫描功能。

## 6. 系统用例图



图1 系统用例图

## 7. 分析用例图

系统主要由两个用例组成，一个是负责显示的图形界面，一个是负责扫描的扫描器。管理员与图形界面进行交互，输入必须的信息，获取输出的结果。图形界面可以向扫描器传递必要的信息和获取扫描结果。

### 7.1 图形界面

用例名称：图形界面

参与者：黑阔/系统管理员

前置条件： 无

后置条件： 在图形界面中参与者可以输入与扫描相关的参数，包括开始ip地址，结束ip地址，开始端口号和结束端口号。系统检查用户输入的ip地址和端口号的合法性，通过检查则将扫描参数传递给扫描模块。

基本事件流：启动软件时用例启动

(1) 启动软件

异常事件流：系统检查输入的ip地址范围和端口号，如果不能通过检查则弹出对话框提示参与者输入条件错误后等待参与者重新输入相关参数。

### 7.2 扫描器

用例名称：扫描器

参与者：无

前置条件：此用例开始前，必须通过图形界面设置相关参数并且通过检查。

后置条件：此用例将扫描结果反馈给图形界面。扫描结束后通知图形界面用例扫描结束。

基本事件流：开始扫描后该用例被启动。

(1) 启动软件

(2) 设置扫描参数

(3) 检查扫描参数

(4) 开始扫描

异常事件流：无

## 8 系统活动图



图2 系统活动图

系统活动图如图2所示，系统主要由两部分组成，图形界面负责与用户进行交互，接受用户输入参数，并对用户输入的参数进行检查，检查通过后讲参数传递给扫描器。图形界面还要根据扫描结果生成扫描报告，以友好的方式将扫描结果反馈给用户。扫描器是扫描任务的实际执行者，扫描器使用图形界面传递过来的参数初始化扫描器，然后在图形界面的控制下开始扫描任务，图形界面可以控制扫描器，主要表现为可以开始和中止扫描器的扫描任务。扫描任务结束后，扫描器要通知图形界面完成扫描。图形界面接收到扫描完成的通知后要通知用户扫描结束。

## 9 开发平台

可供选择的开发平台有多种选择，对操作系统而言主要包括Linux和Windows，图形界面部分也有多种可以选择的开发环境。下面将对操作系统和图形界面的开发环境做出抉择。

### 9.1网络部分

单就网络编程的难度而言，Linux网络编程的难度要小于Windows网络编程的难度。由于Linux操作系统本身与网络联系非常密切，使得Linux下的扫描器非常多，编写、修改容易，运行效率高。但Linux操作系统本身不提供图形界面，Linux操作系统的图形界面由GNOME、KDE等提供，图形界面的运行效率不高，编程较为复杂。Windows系统普及程度高，使用方便，但网络编程难度较高，扫描器编写、移植较Linux下的困难一些。但Windows下图形界面效率较高，编程比较方便。综上所述，加上以前编写过基于Linux的扫描器，本系统选择Windows作为开发平台。

### 9.2 编程语言

面向对象程序设计，当然要用面向对象的编程语言，常见的面向对象的编程语言有C++、Java、C#和VB等。VB封装最为严重，屏蔽了众多细节，提供了丰富的功能，编程非常简单，但由于屏蔽了太多细节导致不够灵活，且效率不高。C#和Java都源于C/C++，语法与C++相似，但抛弃了C++的面向过程部分，为纯面向对象的语言，提供了丰富的类库。C++支持面向对象和面向过程两种方式，没有过多的屏蔽编程细节。由于个人偏好的原因，开发语言选择C++。

### 9.2 图形界面

Windows下的图形界面编程方法繁多，有基于Windows API的、MFC的和众多第三方类库。Windows API编程较为复杂，需要对Windows运行原理比较了解，但Windows API提供了非常灵活的方式用于图形界面的处理。MFC是微软开发的微软基础类，对API进行了封装，提供了相对友好的接口，简化了程序实现。第三方类库则各有各的特点。本系统使用MFC实现图形界面。

### 9.4 开发平台总结

本系统运行于Windows上，使用C++语言编写，使用MFC实现图形界面和网络编程。

# 四、概要设计

## 1. 软件结构

系统由两部分组成，分别是负责和使用者交互的图形界面和实际用于完成扫描任务的扫描器。这两部分的功能由两个类承担，一个是和图形界面相关的类CMFCApplication4Dlg，一个是核心扫描类TCPScanner。



图3 软件结构

## 2 层次关系

从层次上看，直接和用户进行交互的是图形界面，扫描类为图形界面提供相应的接口供图形界面调用，从而完成扫描任务。图形界面负责沟通用户和扫描类，屏蔽扫描类的细节，以友好的方式获取扫描参数并将这些参数转换成扫描类可以理解的数据结构。同时图形界面也屏蔽了扫描类不友好的输出结果，将扫描结果转换成较为友好的方式，清晰的展示给用户。

用户的一切输入都要通过图形界面，扫描的结果也是通过图形界面反馈给用户，图形界面在软件运行的过程中一直存活着，而扫描类只有在开始扫描开始被创建，在扫描结束后即被销毁。

## 3 图形界面

图形界面应该可以提供尽量友好的界面，要清晰、明确的告诉使用者应该提供的参数，应该方便用户输入相关参数。图形界面应该允许用户开始扫描和中止扫描。图形界面应该清晰、明确的显示扫描结果，方便用户获取相关的信息。

图形界面需要接受用户的输入，在开始扫描之前对用户的输入进行检查，拒绝不合法的输入，并要求用户重新输入正确的参数后开始扫描。如果输入的参数合法则初始化一个扫描器类然后调用相关的函数进行扫描。

开始扫描宜采用新建线程的方法以保证扫描线程和图形界面分离，保证图形界面可以相应用户的操作，不至于卡死。

开始扫描后，新建的扫描线程在后台运行，主线程则返回等待和用户进行交互，这时主界面应该向用户提供中止扫描的功能，用户选择中止扫描后通知扫描服务中止扫描。

图形界面设计如图4所示：



图4 UI

从图中可以看出用户界面由两个Group-box Control组成，第一个Group-box Control的标签是扫描选项，这个Group-box Control中包括四个Static Text控件、四个Edit Control、一个Check-box Control和连个Button Control组成。这个Group-box Control控件中的空间允许用户输入包括起始IP、结束IP、起始端口和结束端口，允许用户选择是否按顺序扫描，同时允许用户控制开始扫描和结束扫描。第二个Group-box Control的标签是扫描结果，这个控件包含一个Tree-view Control，用于显示扫描结果。

## 4扫描服务

扫描类的初始化参数有图形界面传入在传入前图形界面已经对输入参数的有效性进行了检查，所以此处不需要再进行检查，可以直接使用构造函数初始换相关的参数。

基本的扫描原理是在指定端口范围内尝试连接每个端口，若能建立连接说明该端口开放，如果无法建立连接说明端口关闭。但实际上由于防火墙或IDS/IPS的存在，实际扫描结果可能会不准确。

扫描器初始化完毕后由图形界面调用扫描类的相关成员函数开始扫描。开始扫描后根据一定的规则创建实际执行扫描任务的线程，同时可以根据用户的要求对端口扫描提供一定的乱序功能以欺骗对方的防火墙或IDS/IPS。同时要对同时扫描的最大线程数进行限制，防止过多的线程占用过多的资源，导致程序或系统卡死。

如果扫到开放的端口要将开放端口的端口号添加到开放端口集合中，同时要像主界面树相应IP地址的根节点下添加叶节点。并且要在最后一个线程结束时通知图形界面扫描完成。

扫描类还需要提供中止扫描的功能，接受图形界面发来的中止扫描请求，控制扫描线程停止产生新的扫描线程。

# 五、详细设计

## 1 CMFCApplication4Dlg类

用户界面主要功能是与用户进行交互，接收用户输入并对用户输入的参数进行检查，并根据用户的选择开始或中止扫描，扫描结束后向用户显示扫描结果。

图形界面由CMFCApplication4Dlg类实现。

对话框初始化时完成绘制对话框的任务，完成变量和控件的动态绑定，并对对话框内的文本进行设定，并将数据流动方向设置为从变量到控件完成相关控件文本的初始化工作。然后进入消息循环，等待用户输入相关参数开始扫描。

当用户单击开始扫描按钮后，图形界面调用相关函数。该函数首先将数据流动方向设置为从控件流向变量，获取控件文本，即用户的输入。获取用户的输入后检查是否正在进行扫描，如果正在进行扫描则结束正在进行的扫描。然后检查用户的输入参数是否合法，若合法则清空树的内容即清除上一次的扫描结果，创建新的扫描线程，将数据流动方向设置为从变量流向控件。

当用户单击结束扫描时首先检测是否有正在运行中的扫描任务，若有则调用相关函数结束扫描，如果没有正在运行中的扫描则不理用户。

扫描函数可以新建一个扫描类，然后开始扫描过程。

扫描过程结束后由扫描类调用回调函数通知图形界面。

由于系统提供的IP地址转换函数可能出现错误而且不提供错误的具体说明，所以该类提供了IP地址在点分十进制和整型变量之间转换的函数，并保证合法的IP地址都能成功转换。

变量和控件的动态绑定有DoDataExchange函数完成，在该函数中调用DDX\_Control、DDX\_TEXT和DDX\_Check函数完成动态绑定。

BEGIN\_MESSAGE\_MAP宏定义了相关的消息映射。

OnPaint函数用于客户区域的绘制。

### 1.1类图



图5 CMFApplication4Dlg的UML类图

### 1.2类说明

表1 CMFCApplication4Dlg类说明

|  |  |  |
| --- | --- | --- |
| CMFCApplication4Dlg类 | | |
| 父类 | CDialogEx | |
| 成员函数 | 参数 | 功能 |
| CMFCApplication4Dlg（） | CWnd\* pParent = NULL | 标准构造函数 |
| GetPortName（） | UINT uPort, BOOL bIsTCP | 获取知名端口运行服务 |
| IPIntToStr（） | UINT | 整型IP地址转换成点分十进制 |
| IPDwordToStr（） | UINT | DWORD类型IP地址转换成点分十进制 |
| IPStrToInt | CString | 点分十进制转换成整型 |
| ScanThread（） | LPVOID | 扫描线程 |
| OnCancel（） | 无 | 退出程序 |
| FreeScanner（） | int, void\* | 删除扫描类 |
| DoDataExchange（） | CDataExchange\* | 完成动态绑定 |
| OnInitDialog（） | 无 | 初始化对话框 |
| OnSysCommand（） | UINT, LPARAM | 系统响应 |
| OnPaint() | 无 | 重绘界面 |
| OnQueryDragIcon | 无 | 显示图标 |
| DECLARE\_MESSAGE\_MAP（） | 无 | 消息映射（宏） |
| OnBnClickedButtonScan() | 无 | 单击开始扫描按钮，开始扫描 |
| OnBnClickedButtonScanstop() | 无 | 单击结束扫描按钮，中止扫描 |
| 成员变量 | 类型 | 含义 |
| m\_ctlTreeResult | CTreeCtrl | 控制树的显示 |
| m\_strIPEnd | CString | 开始IP |
| m\_strIPBegin | CString | 结束IP |
| m\_strComment | CString | 系统命令 |
| m\_uPortBegin | UINT | 开始端口 |
| m\_uPortEnd | UINT | 结束端口 |
| m\_bNoOrder | BOOL | 是否按顺序扫描 |
| test | TCPScanner\* | 指向扫描类的指针 |
| m\_hIcon | HICON | 图标句柄 |

## 2. TCPScanner类

该类是扫描器的核心，执行实际的扫描任务，该类可以根据图形界面传入的参数，在图形界面的控制下开始或停止扫描。

该类使用CSocket类进行扫描。构造函数可以接受图形界面传入的各项数据初始化相关参数并初始化socket接口。

该类提供相关的函数供图形界面调用，以实现开始扫描的功能。该函数可以再树中插入根节点，然后在事件对象和一个布尔型变量的控制下开始多个线程并发扫描。

实际连接目标主机端口的线程应该检查当前运行线程总数，当大于设定值的时候讲事件对象复位，阻止新线程的建立，然后根据是否乱序扫描决定是否随机延迟一段时间。延迟结束后尝试连接目标主机的指定端口，如果连接成功则向树中添加该端口。最后将事件对象置位，恢复新线程的建立。检查线程个数，如果等于0则通知图形界面扫描完成。

### 2.1 类图



图5 TCPScanner类的UML类图

### 2.2 类说明

表2 TCPScanner类说明

|  |  |  |
| --- | --- | --- |
| TCPScanner类 | | |
| 成员函数 | 参数 | 功能 |
| TCPScanner() | 太多了…… | 构造函数 |
| start() | 无 | 开始扫描 |
| ThreadCSocketScan() | LPVOID | 连接目标主机指定端口，实际扫描函数 |
| IPIntToStr() | UINT | 整型IP地址转换成点分十进制 |
| IPStrToInt() | CString | 点分十进制转换成整型 |
| GetOpenPort() | UINT | 获得指定IP开放端口 |
| getThread() | 无 | 获取正在运行的线程数 |
| cancle() | 无 | 结束扫描 |
| 成员变量 | 类型 | 含义 |
| inOrder | BOOL | 是否按顺序扫描 |
| BeginIP | CString | 开始IP |
| EndIP | CString | 结束IP |
| BeginPort | UINT | 开始端口 |
| EndPort | UINT | 结束端口 |
| run | BOOL | 是否继续扫描 |
| ThreadCount | int | 扫描线程计数 |
| ThreadParam | THREADPARAM | 传递给线程的参数 |
| OpenPort | map< UINT , set < UINT > > | IP地址到开放端口集合的映射 |
| m\_ctlTreeResult | CTreeCtrl\* | 指向树的指针 |
| single | HANDLE | 控制新线程的产生 |
| destory | CallBack | 通知结束扫描函数的指针 |
| caller | void\* | 调用类的指针 |

## 3 序列图



图6 UML序列图

## 4.序列图分析

（1）打开软件后显示图形界面，系统执行OnInitDialog函数，初始化对话框界面，完成控件和变量的动态绑定，然后进入消息循环。用户在主界面中输入相关的参数。

（2）用户单击开始扫描按钮，调用OnBnClickedButtonScan函数，该函数调用UpdateData(TRUE)函数，从控件获取数据。然后检查用户的输入，未通过检查提示错误，返回第一步。通过检查后将树的内容全部删除，调用AfxBeginThread函数开始新的ScanThread函数线程后调用UpdateData(FALSE)函数重新将数据流动方向设置为从变量到控件。最后返回，图形界面再次进入消息循环，等待用户指令。

（3）ScanThread函数新建一个TCPScanner对象，然后调用TCPScanner::start函数开始扫描，控制权交给TCPScanner类。

（4）TCPScanner类的构造函数初始化相关参数。Start函数被调用后开始扫描。

（5）start函数首先在树中插入一个根节点，然后在成员变量run和成员变量single的控制下AfxBeginThread函数开始新的线程，尝试连接目标主机的端口。

（6）ThreadCSocketScan函数对同时运行的线程数加1，测试是否大于250个，如果过大于则复位时间对象阻止新进程的建立。然后尝试连接目标主机的目标端口，如果连接成功则将端口号插入到树中。最后，置位时间对象，如果线程数为0则通知图形界面扫描结束。

# 六、编码

编码使用Visual Studio 2015 Enterprise，使用向导创建基于对话框的MFC工程。版本控制使用github提供的git服务，通过Visual Studio提供的插件提交、推送更改。

图形界面使用可视化资源编辑器生成。

# 七、测试

测试主要从系统对资源的占用，有无BUG和结果的准确性几方面进行验证。

最初的系统由于没有对线程数进行限制导致系统占用了过多的资源，导致本系统卡死和操作系统反应迟钝，引入线程控制后得到解决。经测试，本系统占用的系统资源较少，系统运行过程中可以正常的重绘窗口，对系统性能没有太大影响。系统对内存资源的占用情况如图7所示：

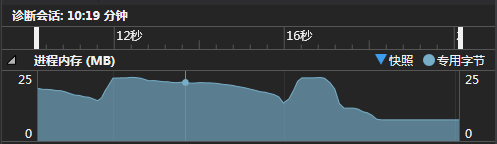


图7 内存占用

系统BUG均已排除。

对扫描准确性的测试：

尝试扫描本机小于1024的开放端口，扫描结果如图8所示：

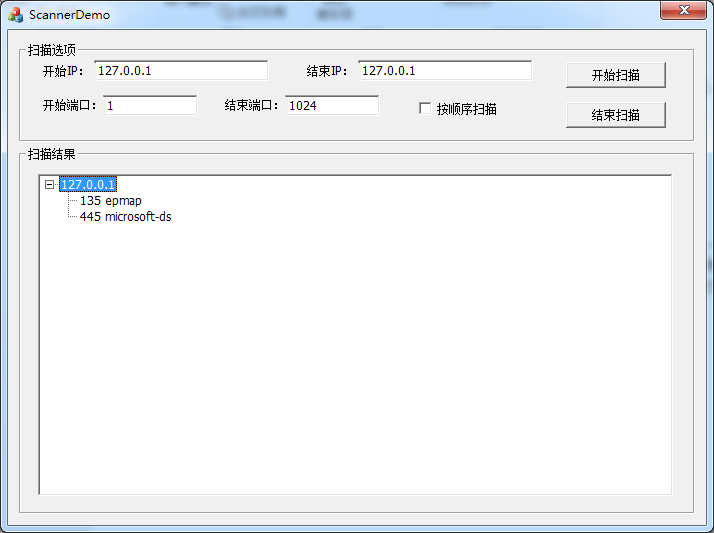


图8 扫描本机开放端口

系统中实际开放端口如图9所示：

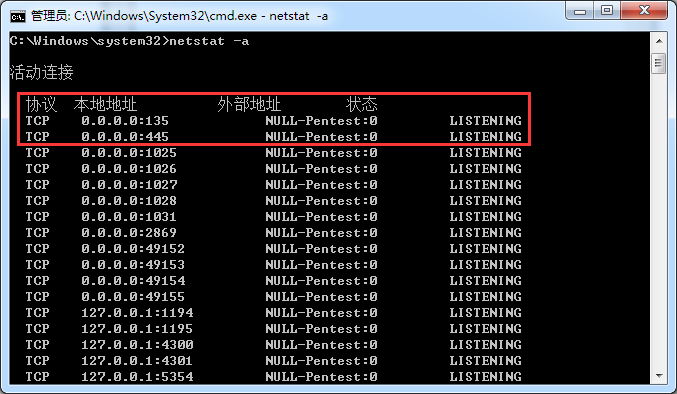


图9 系统开放端口

可以看出，扫描器对本机开放端口的识别具有一定的准确性。

测试对远程主机的扫描：选择百度为目标主机，首先解析百度的域名：



图10 解析域名

使用本系统进行扫描：

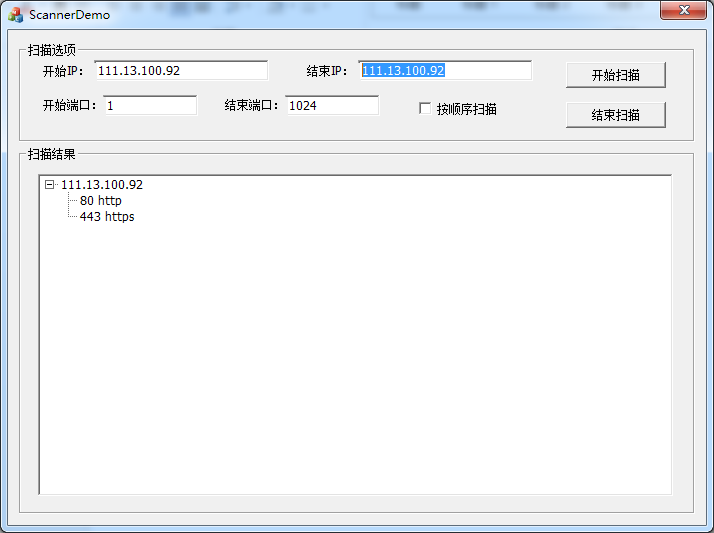


图11 本系统对百度扫描结果

使用其他黑客工具扫描结果：

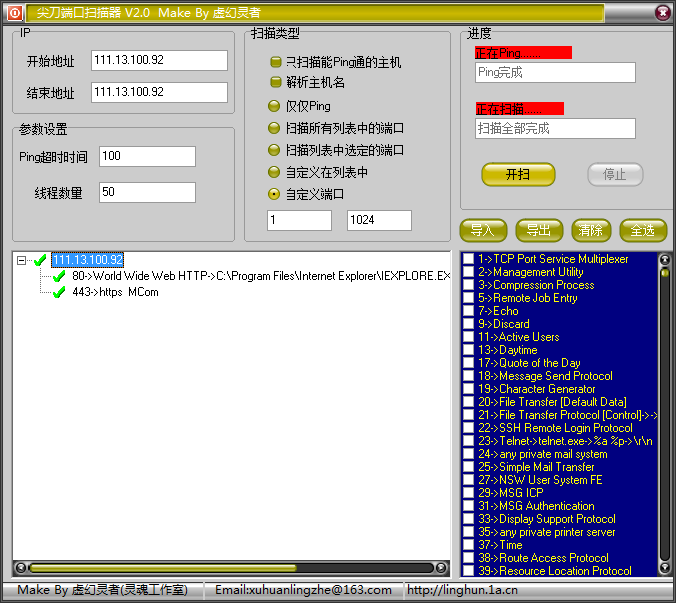


图12 其他黑客工具扫描结果

从对比中可以看出，本系统的扫描结果正确。

# 八、总结

通过这次的课程设计，我们进一步了解了软件工程的基本方法，提高了解决问题的能力。但在完成课程设计的过程中也有一些遗憾的地方，比如说，未考虑到Windows xp sp2以后对原始套接字做出了限制，WinPcap发包存在一些问题等，这些问题导致发送和接收IP层的报文出现问题，同时不能发送和接收IP层的报文也意味着无法实现操作系统识别等高级扫描技术，这些遗憾只能在以后的工程中弥补。