**安全科学与工程学院**

**2021－2022 学年第 2 学期**

**计算机网络课程设计**

**题目： 基于scapy的报文分析平台**

**班级： 190342B**

**学号： 190146841**

**姓名： 邹嘉旭**

**教师： 张礼哲、雷缙**

**成绩：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 项目 | 网络协议分析(50%) | | 网络应用编程(50%) | |
| 程序(25%) | 报告(25%) | 程序(25%) | 报告(25%) |
| 分数 |  |  |  |  |

目 录

1 网络协议分析 1

1.1 系统概述 1

1.2 开发环境 1

1.3 系统设计 1

1.4 程序流程 1

1.5 主要数据结构 1

1.6 主要函数说明 1

1.7 系统使用说明 1

1.8 项目分析总结 2

2 网络应用编程（注明具体题目） 2

2.1 系统概述 2

2.2 开发环境 2

2.3 系统设计 2

2.4 程序流程 2

2.5 主要数据结构 2

2.6 主要函数说明 2

2.7 系统使用说明 2

2.8 项目分析总结 3

3 课程设计总结 3

参考文献 3

# 网络协议分析

本模块要求设计实现协议分析器或协议编辑器。协议分析器实现捕获、分析数据包功能，协议编辑器实现生成、发送数据包功能。要求至少支持以太网MAC协议、IP和TCP/UDP协议。

## 系统概述

网络由协议构成，网络安全需要从最基础的网络协议入手，对报文的分析至关重要。

Scapy是一个功能强大的交互式数据包操作程序，可以探测，扫描或攻击网络，使用户能够发送，嗅探和剖析网络数据包。它也能够伪造或解码大量协议的数据包，通过线路发送，捕获它们，匹配请求和回复等等。Scapy可以轻松处理大多数经典任务，如扫描，跟踪路由，探测，单元测试，攻击或网络发现。

利用Scapy强大的功能，可以使用Python语言实现一个基本的网络报文发送解析平台。

## 开发环境

KUbuntu21.10(Linux 5.13)+python3.9.7+PyQt5 5.15.6+Visual Studio Code

## 系统设计

程序共包含6个主要模块，分别为：MAC帧的发送与嗅探、ARP报文的发送、IP报文的发送与接收、UDP报文的发送与接收、TCP报文的发送与接收及全部报文的嗅探，如图1。

每个模块设置一个stackedWidget，模块中每一个功能均设置于一个子标签页。发送、接收的报文的详细信息均在程序的Terminal中输出。程序的Terminal可随时清空。

各接收功能均由一个新线程完成，保证GUI不被阻塞。



图1 整体架构

为了在Terminal中对报文加以区分区，各模块均设有标志色。其中，MAC帧为#FF9933，ARP报文为#FA8027，IP报文为#0000FF，UDP报文为#00FF00，TCP报文为#FF00FF。效果如图2、图3所示。

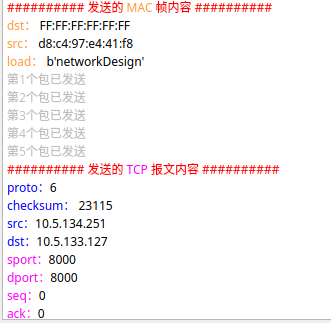


图2 颜色区分

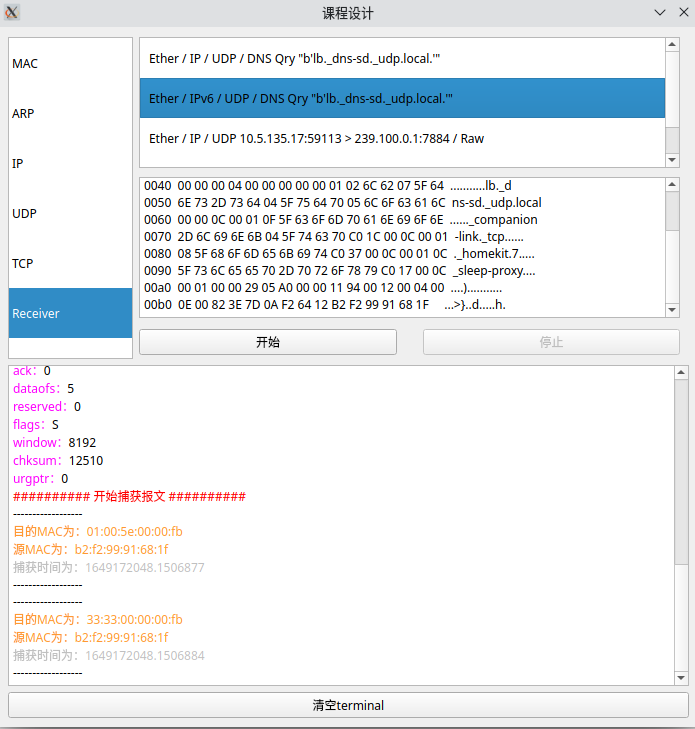


图3 颜色区分

## 程序流程

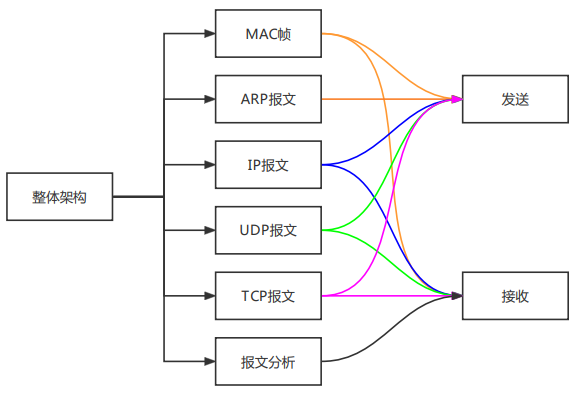


图4 程序流程

## 主要数据结构

绑定组件过程按下不表，只展示主要业务代码并以MAC帧发送类为例：发送按钮按下后，触发sendMac()方法，并从组件中获取必要信息组成并发送报文；开始接收按钮按下后，开启新线程以嗅探MAC帧，根据外部flag对sniff进行启停控制，同时，失效“开始接收”按钮，启用“停止接收”按钮。

其他功能均类似于此。

1. #MAC标签页
2. **class** Mac(QObject):
3. #由于不能跨线程操作qt组件，故设置一信号，通过该信号触发主线程的槽以更改qt组件
4. sniffcallback = QtCore.pyqtSignal(str)
6. **def** \_\_init\_\_(self, total) -> None:
7. super(Mac, self).\_\_init\_\_()
8. self.isSniff = True
9. self.sniffCount = 0
10. self.total = total
11. self.ui = total.ui
12. self.targetLine = self.ui.findChild(QLineEdit, "targetMacLine")
13. self.payloadLine = self.ui.findChild(QLineEdit, "payloadMacLine")
14. self.timesLine = self.ui.findChild(QLineEdit, "timesMacLine")
15. self.sendBtn = self.ui.findChild(QPushButton, "sendMacBtn")
16. self.startSniffBtn = self.ui.findChild(QPushButton, "sniffStartMacBtn")
17. self.stopSniffBtn = self.ui.findChild(QPushButton, "sniffStopMacBtn")
18. self.sniffTimesLine = self.ui.findChild(QLineEdit, "sniffTimesMacLine")
19. self.sendBtn.clicked.connect(**lambda**: self.sendMac())
20. self.startSniffBtn.clicked.connect(**lambda**: self.startSniff())
21. self.stopSniffBtn.clicked.connect(**lambda**: self.stopSniff())
22. self.sniffcallback.connect(self.sniffCallBack)
24. **def** sendMac(self):
25. dst = self.targetLine.text()
26. payload = self.payloadLine.text()
27. times = self.timesLine.text()
29. packet = Ether(dst=dst)/payload
31. self.total.terminal.append(
32. "<font color=#FF0000>##########  发送的  <font color=#FF9933>MAC</font>  帧内容  ##########</font>")
33. self.total.terminal.append(
34. "<font color=#FF9933>dst：</font>  %s" % packet[Ether].dst)
35. self.total.terminal.append(
36. "<font color=#FF9933>src：</font>  %s" % packet[Ether].src)
37. self.total.terminal.append(
38. "<font color=#FF9933>load：</font>  %s" % packet.load)
39. packet.show()
40. **for** i **in** range(int(times)):
41. sendp(packet)
42. self.total.terminal.append(
43. "<font color=#C0C0C0>第%s个包已发送</font>" % (str(i+1)))
45. **def** startSniff(self):
46. **def** sniffThread():
47. sniffNum = int(self.sniffTimesLine.text())
48. **if** sniffNum > 0:
49. sniff(prn=self.ether\_callback,
50. stop\_filter=**lambda** x: self.isSniff, count=sniffNum)
51. self.stopSniff()
52. **elif** sniffNum == 0:
53. sniff(prn=self.ether\_callback,
54. stop\_filter=**lambda** x: self.isSniff)
55. **else**:
56. QMessageBox.warning(
57. self.ui, "警告", "捕获报文数量大于等于0", QMessageBox.Yes)
58. self.stopSniff()
60. **if** self.isSniff **is** True:
61. self.sniffcallback.emit(
62. "<font color=#FF0000>##########  开始捕获  <font color=#FF9933>MAC</font>  报文  ##########</font>")
63. self.isSniff = False
64. t = threading.Thread(target=sniffThread, name='LoopThread')
65. t.start()
66. self.startSniffBtn.setEnabled(False)
67. self.stopSniffBtn.setEnabled(True)
69. **def** ether\_callback(self, pkt):
70. self.sniffCount += 1
71. self.sniffcallback.emit(
72. '<font color=#000000>------------------</font>')
73. self.sniffcallback.emit(
74. '<font color=#C0C0C0>捕获到第%d个数据帧 </font>' % self.sniffCount)
75. self.sniffcallback.emit(
76. '<font color=#FF9933>目的MAC为：%s </font>' % pkt[Ether].dst)
77. self.sniffcallback.emit(
78. '<font color=#FF9933>源MAC为：%s </font>' % pkt[Ether].src)
79. self.sniffcallback.emit(
80. '<font color=#C0C0C0>捕获时间为：%s </font>' % pkt.time)
81. self.sniffcallback.emit(
82. '<font color=#000000>------------------</font>')
84. time.sleep(1)
86. **def** stopSniff(self):
87. self.isSniff = True
88. self.sniffCount = 0
89. self.startSniffBtn.setEnabled(True)
90. self.stopSniffBtn.setEnabled(False)
92. **def** sniffCallBack(self, str):
93. self.total.terminal.append(str)

## 主要函数说明

开始嗅探

1. **def** startSniff(self):
2. **def** sniffThread():
3. self.sniffCount = int(self.sniffTimesLine.text())
4. #如果次数>0，通过sniffCount控制次数
5. **if** self.sniffCount > 0:
6. sniff(filter="ip", prn=self.ether\_callback,
7. stop\_filter=**lambda** x: self.isSniff, count=self.sniffCount)
8. self.stopSniff()
9. **elif** self.sniffCount == 0:
10. sniff(filter="ip", prn=self.ether\_callback,
11. stop\_filter=**lambda** x: self.isSniff)
12. **else**:
13. QMessageBox.warning(
14. self.ui, "警告", "捕获报文数量大于等于0", QMessageBox.Yes)
15. self.stopSniff()
17. **if** self.isSniff **is** True:
18. self.sniffcallback.emit(
19. "<font color=#FF0000>##########  开始捕获  <font color=#0000FF>IP</font>  报文  ##########</font>")
20. self.isSniff = False
21. #线程
22. t = threading.Thread(target=sniffThread, name='LoopThread')
23. t.start()
24. #按钮状态
25. self.startSniffBtn.setEnabled(False)
26. self.stopSniffBtn.setEnabled(True)

由于PyQt5无法在不同线程中操作组件，使用信号以实现不同线程的控制。

1. #报文嗅探窗口
2. **class** Receiver(QObject):
3. sniffcallback = QtCore.pyqtSignal(str)
4. additemcallback = QtCore.pyqtSignal(object)
6. **def** \_\_init\_\_(self, total) -> None:
7. super(Receiver, self).\_\_init\_\_()
8. ....
9. self.sniffcallback.connect(self.sniffCallBack)
10. self.additemcallback.connect(self.addItemCallBack)
12. **def** ether\_callback(self, pkt):
13. self.sniffCount += 1
14. self.additemcallback.emit(pkt)
15. time.sleep(1)
17. **def** selectItem(self):
18. self.sniffcallback.emit(
19. '<font color=#000000>------------------</font>')
20. self.sniffcallback.emit(
21. '<font color=#FF9933>目的MAC为：%s </font>' % item.packet[Ether].dst)
22. self.sniffcallback.emit(
23. '<font color=#FF9933>源MAC为：%s </font>' % item.packet[Ether].src)
24. self.sniffcallback.emit(
25. '<font color=#C0C0C0>捕获时间为：%s </font>' % item.packet.time)
26. self.sniffcallback.emit(
27. '<font color=#000000>------------------</font>')
29. **def** get\_item\_wight(self, data):
30. # 读取属性
31. packet = data
32. summary = str(packet.summary())
33. # 总Widget
34. widget = QWidget()
35. # 总体横向布局
37. nameLabel = QLabel(summary)
38. layout\_main = QHBoxLayout()
39. layout\_main.addWidget(nameLabel)  # 最左边的头像
40. widget.setLayout(layout\_main)  # 布局给wight
41. **return** widget  # 返回wight

## 系统使用说明

程序共包含6个主要模块，分别为：MAC帧的发送与嗅探、ARP报文的发送、IP报文的发送与接收、UDP报文的发送与接收、TCP报文的发送与接收及全部报文的嗅探。

各个模块可并行执行。

## 项目分析总结

本次课程设计在老师提供的代码参考下，使用PyQt5进行重构，使界面更美观，代码结构更清晰。同时，对代码进行修改的过程中对网络协议报文的结构有了更深的认识。本次课程设计存在的不足是对于ARP报文的summary过程阻塞了主线程，可修改为在子线程中进行。

# 网络应用编程（注明具体题目）

自选一个题目完成。

## 系统概述

说明选题、系统实现功能、背景知识等。

## 开发环境

## 系统设计

系统设计思路（包含系统架构、功能模块设计等）。

## 程序流程

画出各模块的程序流程图。

## 主要数据结构

列出主要数据结构并加以说明。

## 主要函数说明

给出主要函数的声明，说明其功能，不需要完整代码。

## 系统使用说明

说明系统实现的功能。

## 项目分析总结

分析项目开发过程中遇到的问题及解决办法；总结项目亮点及不足之处。

# 课程设计总结

完成课程设计之后，自己有何想法，对课程有何建议。

# 参考文献

可以是图书、期刊、网页等，借鉴过的源代码也请在这里加以说明。