DOI:10.16208/j.issn1000-7024.2011.03.068

Wireshark 环境下的网络协议解析与验证方法

罗青林[1]， 徐克付[2]， 臧文羽[2]， 刘金刚[1]

(1. 首都师范大学 计算机科学联合研究院，北京 100048；
2. 中国科学院计算技术研究所 信息内容安全技术国家工程实验室，北京 100190)

摘 要：网络协议解析通过程序分析网络数据包的协议头及其负载，是一系列网络功能的基础。分析了 Wireshark 的功能、作用、体系结构以及开发环境，给出了 Wireshark 在 Windows 系统下对网络协议解析的两种方法，总结了两种方法的特性。实验结果表明，在两种方式下添加的协议解析器都能正确解析网络数据包相应协议并分析数据包的负载内容，对网络协议的解析与验证及网络数据包的内容分析具有借鉴意义。

关键词：Wireshark； 网络协议； 协议树； 协议提交； 协议解析

中图法分类号：TP393 **文献标识码**：A **文章编号**：1000-7024 (2011) 03-0770-04

Network protocol parser and verification method based on Wireshark

LUO Qing-lin[1], XU Ke-fu[2], ZANG Wen-yi[2], LIU Jin-gang[1]

(1. Join Faculty of Computer Scientific Research, Capital Normal University, Beijing 100048, China;
2. National Engineering Laboratory for Information Security Technologies, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: The parsing of network protocol is the basis of a series function of the network, which dependent on parsing the head and payload of the packet. The Wireshar's function, architecture, components and development environment are analyzed, and the source code is also studied. How to add network parsing protocol in two different ways in Windows is researched, and the two methods with their own characteristics are summarized too. Finally, the validity and efficiency of the above presented method is demonstrated by the experiments. The reference is provided for network protocol parsing and verification, as well as lay the foundation for follow-up designing and developing more complicated network application.

Key words: wireshark; network protocol; protocol tree; protocol registration; protocol parser

0 引 言

网络协议解析是指通过程序分析网络数据包的协议头及其负载，从而了解网络数据包在产生和传输过程中的行为。包含该解析程序的软件或设备通称为协议解析器。网络协议解析有着广泛而深入的应用，如监视网络流量、分析数据包、监视网络资源利用、执行网络安全操作规则，鉴定分析网络数据以及诊断并修复网络问题等等，是一系列网络功能的基础。由于网络协议种类繁多，而且各种新的协议层出不穷，因此需要解析各种各样新的网络协议，同时解析器必须具有良好的可扩展性，以方便地为系统添加新的协议解析程序。Wireshark (前称 Ethereal) 是一款免费开源的协议解析器，是目前世界范围内应用最广泛的网络协议解析软件之一。在 GNUGPL 通用许可证的保障范围内，使用者可以免费取得该软件及其源代码，并可以根据自身的需要对 Wireshark 进行定制或扩展。

本文介绍了 WireShark 的功能、作用、构成、开发环境等；研究了 WireShark 协议解析器的两种设计方法及其流程、网络数据包的内容检测方法，并给出了试验验证，为网络协议的解析与验证、数据包的内容检测提供了借鉴意义，并为后续设计与开发更复杂的网络应用系统打下了基础。

1 WireShark 的系统结构

WireShark 能够在网卡接口处捕捉数据包、并实时显示包的详细协议信息；能够打开/保存捕捉的包、导入导出其它捕捉程序支持的包数据格式；能够在网卡处有选择性捕捉数据包、在捕捉到的包中也可以有选择性的显示不同条件的数据包；还可以显示多种统计分析结果(比如 TCP、UDP 流、各个协议层统计信息等)，同时扩展了大量的 Ethereal(Wireshark 前身)

收稿日期：2010-03-12；修订日期：2010-05-16。
基金项目：国家 973 重点基础研究发展计划基金项目 (2007CB311100)。
作者简介：罗青林 (1985－)，男，江西九江人，硕士研究生，研究方向为计算机网络、信息内容安全； 徐克付 (1977－)，男，湖北随州人，博士后，研究方向为信息内容安全、分布式系统； 臧文羽 (1988－)，女，湖南湘潭人，硕士研究生，研究方向为信息内容安全； 刘金刚 (1962－)，男，教授，研究员，博士生导师，研究方向为智能接口技术、操作系统。E-mail：luoqinglin@software.ict.ac.cn

所不支持的协议，支持在 UNIX 和 Windows 平台下进行开发。其系统结构[1]如图 1 所示。

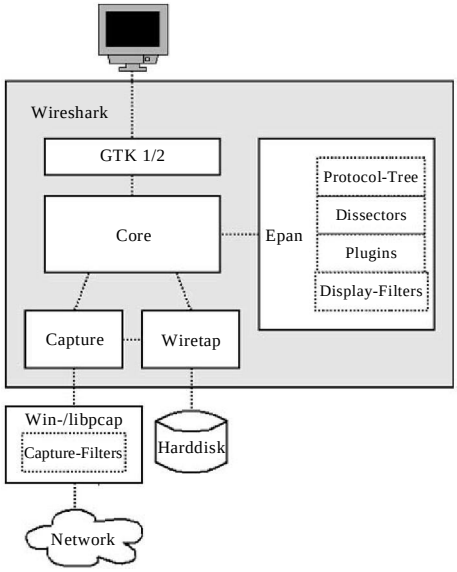

图 1 Wireshark 的系统结构

Wireshark 系统结构的 6 个功能模块如下：

GTK 1/2：图形窗口编程工具。

Core：将其它模块连接起来，起到综合调度的作用。

Epan：Wireshark 对协议的具体解析，其中协议解析器的开发包括内置(built-in)和插件(在 plugin)的方式，后面将详细介绍这两种方式的开发。

Win-/libpcap：底层抓包工具，提供了一整套的抓包函数库。

Capture：抓包引擎。

Wiretap：向磁盘读写包文件，包括 libpcap 和其它格式的包文件。

2 开发环境介绍

下面以 Windows 平台为例，详细说明 Wireshark 的工作原理与具体实现机制。首先介绍开发平台的搭建方法与过程，包括所需要的工具和详细步骤[2]。为了叙述方便，本文假设源代码存放 F:\。

2.1 所需要的工具

Cygwin：一个在 Windows 平台上运行的 Unix 模拟环境。

Compiler：本文采用 vs2008。

Python：使 cygwin 的包在 win32 下准确畅通地执行。

Subversion：从 Wireshark 的源代码池中获得最新源代码。

NSIS：把 Wireshar 编译成可安装的 exe 文件。

2.2 详细步骤

步骤 1：下载并安装 vs2008，包括用于设置环境变量 vcvars32.bat。

步骤 2：下载并安装 cygwin，这一步要另外多选择几个包，分别是：Archive/unzip、Devel/bison、Devel/flex、Interpreters/perl、Utils/patch、Web/wget、Archive/mt；因为这些包在 cygwin 默认安装的时候并不包括。

步骤 3：下载并安装 python，保持默认即可。

步骤 4：下安装 Subersion Client。

假设以上几步都成功安装，并且从 Wireshark 源代码池中获取了最新源代码，接下来就可以编译源文件。

步骤 5：修改 c:\wireshark\config.nmake 中对应的编译器选项、程序安装路径等。

步骤 6：检查所需要的工具是否全部安装了，在 cmd.exe (而不是刚刚安装的 cygwin 的 bash)设置好了环境变量和路径后键入：

nmake -f Makefile.nmake verify_tools

如果出现和图 2 一样的内容，说明编译所需要的工具全部准确安装了。

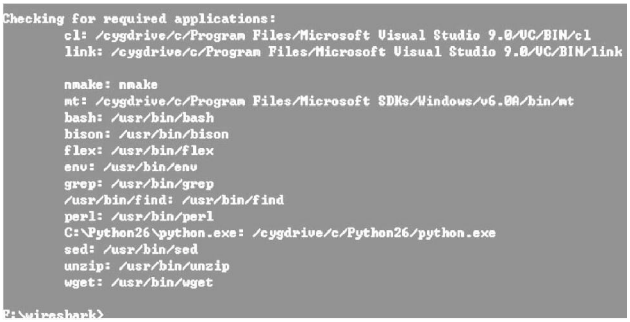

图 2 检查工具安装的输出

步骤 7：在 cmd.exe 键入(设置要环境变量和路径)nmake -f Makefile.nmake setup，自动下载生成编译所需的库文件。

步骤 8：在 cmd.exe 键入(设置要环境变量和路径)nmake -f Makefile.nmake disclean，这一步同样非常重要，因为 Wireshark 的源文件中包含一些在 Unix 编译的文件，因此需要把它删除，以免在 Windows 下编译 Wireshark 时存在一些文件冲突而造成莫名其妙的错误。

到此为止所有准备工作都成功结束了，现在可以编译 Wireshark 了。

步骤 9：在 cmd.exe 键入(设置要环境变量和路径)nmake -f Makefile.nmake all 这个过程大概需要十几分钟左右，具体差距因机器的配置而不同。

成功编译后，将会在 Wireshark 的根目录下面看到诸如 exe、sln、vcproj 等文件，可以运行 Wireshark 抓取和解析数据包了。

3 Wireshark 解析协议的开发与应用

3.1 数据包协议解析工作原理

应用层数据在经网络传输之前，先由上往下层层封装的[3-4]，而 Wireshark 的解析正好相反，由下往上层层解封装的，Wireshark 以一棵协议树的形式对数据包分析，解析到某一层的时候由某一层具体的解析协议来解析。图 3 描述了一个 http 数据包的详细解析过程，首先在以太网卡处获得数据帧，由 ARP/RARP 等解析数据包的源/目的 MAC 地址等，到了 IP 层时由 IP 解析协议来解析，它将详细解析 IP 报头的各个字段信息，里面的载荷(payload)数据则根据不同的数据包类型交给不同的子解析协议来出来，这里是 TCP 解析协议来处

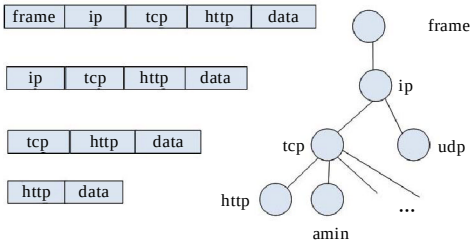

图 3 数据包的协议解析过程

理，TCP 再将它里面封装的载荷类型交给 HTTP 解析协议模块来处理。

3.2 解析协议的开发

Wireshark 的协议解析开发有两种方式：内置型和插件型。插件类型开发相对容易编写，因为它不需要知道系统的整个框架结构以及整个程序的详细流程；而只需要根据接口的相关说明直接调用就可以了。因此本文先从插件型协议解析开发开始介绍。

3.2.1 插件型(plugin)

Wireshark 提供了一种通用的接口通过插件[5]的方式开发解析协议，即在源代码的指定目录下（…\wireshark\wireshark-gtk2\plugins\）增加编译好的库文件(比如 amin.dll)。

为了编译解析器并创建插件，需要在解析器代码文件所在的目录下创建一些提供支持的文件，分别是：

- Makefile.am：UNIX/Linux 的 makefile 模板
- Makefile.common：包含了插件文件的名称
- Makefile.nmake：包含了在 Windows 平台下 Wireshark 插件的 makefile
- moduleinfo.h：包含了插件版本信息
- moduleinfo.nmake：包含了针对 Windows 平台动态链接库(DLL)版本信息
- amin.c：要开发的解析器的源代码文件
- plugin.rc.in：包含了针对 Windows 平台的 DLL 资源模板

上述支持文件具体编写，可以参照 Wireshark 源文件在(plugin\)提供的一些经典实例(比如 profinet 文件)。

3.2.2 内置型(built-in)

内置型的协议解析开发对比比较插件方式而言：需要了解 Wireshark 的组织架构，哪些协议解析本身就是以内置型的方式开发的，源文件放在什么位置，是如何被编译和运行的等。

由于内置型[7]的协议解析源代码都在目录(...\wireshark\epan\dissectors)下，因此首先将要添加的解析器的源文件和头文件必须放在此目录下；然后同样在此目录下找到 Make-file.common 文件，分别将源文件和头文件名添加到对应的'DISSECTOR_SRC'和'DISSECTOR_INCLUDE'宏目录下面。

最后对整个工程进行编译连接，就可以把要添加的解析器，如同 Wireshark 大多数解析器一起被整合编译到 libwireshark.dll 库文件里面。

3.3 解析器的代码简要分析

一个新的协议解析器需要通过协议注册、协议提交、协议解析 3 个步骤[8]添加到 Wireshark 主程序中。事实上，Wireshark 环境下所有的协议解析器都是通过这 3 个过程来实现，比如网络传输过程使用频率较高的 tcp、udp、ftp、http 等内置型协议。

(1)协议注册

通过调用 proto_register_protocol()协议注册函数向主程序注册一个协议名(如 tcp)的解析器，返回的 proto_tcp 相当于主程序管理协议解析器的一个句柄。在这个协议注册里面同时包含两个重要的函数，即调用 proto_register_field_array()协议字段注册函数把协议字段信息注册到该协议下面，proto_register_subtree_array()函数将相应项的和值添加到协议树下面。

```
proto_tcp = proto_register_protocol(
  "Transmission Control Protocol",   /* name */
  "TCP",      /* short name */
  "tcp");   /* abbrev */
```

(2)协议提交

```
proto_reg_handoff_tcp(void){
tcp_handle = create_dissector_handle(dissect_tcp, proto_tcp);
dissector_add("ip.proto", IP_PROTO_TCP, tcp_handle);
}
```

通过将注册的协议向主程序提交，使得协议解析函数和主函数关联起来，告诉系统当 ip 数据包报头协议选项为 tcp 协议时，就要调用 dissect_tcp 这个解析模块来处理。即 tcp 协议解析器是在 IP 协议分支下注册的一个节点。

(3)协议解析

这个函数里面就是一个协议解析器具体要做的事情，比如对报头怎么解析，对载荷数据如何解析，是当前处理，还是交给相应的子节点来处理？各个协议解析器的复杂程度也就取决于这部分的具体解析[5]工作，详细解析工作可以分析 Wireshark 源文件里各个解析器的协议解析函数的源代码，同时可以将许多定制扩展功能[6]从这个解析函数里加进来。比如 tcp 协议解析函数 dissect_tcp()，它首先会将报头的各个字段信息详细给解析出来，然后对于 payload 载荷数据，如果存在有效载荷数据就交给 dissect_tcp_payload()函数去处理，或者自己处理，或者交给注册在该协议下的相应解析协议去处理。

3.4 数据包负载内容检测

在数据包通过协议树解析过程中，不仅可以对包头进行解析，实现基于报头特征的快速包分类，还可以对包负载内容通过串匹配算法来进行检测，从而对一些符合相关特性的数据包进行特殊处理。比如可以将 snort 规则集建成自动机，然后将所有的数据包内容在这个自动机上进行扫描，将匹配成功的数据包进行相应的标记以进行相应处理、报警、过滤、拦截等。

4 实例的验证

4.1 协议注册验证

假设有一个 amin 协议解析器，注册在 tcp 协议节点下且注册端口号等于 999，那么如何验证这个 amin 协议解析器成功注册了且能够进行相应的解析，可以从两个方面来验证。

(1) 查看协议和其相应的字段是否注册成功，通过运行 Wireshark 程序后，在 filter 过滤器那一栏单击 expression 按钮，如果在弹出的窗口里面能找到 amin 协议和相应的协议字段，

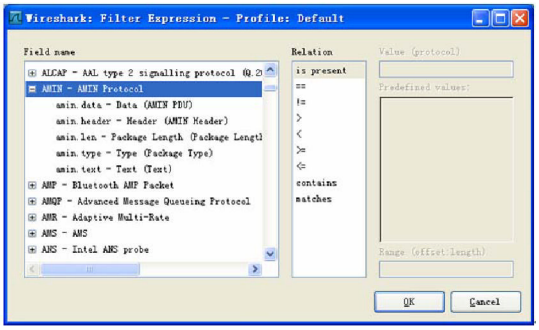

图 4　amin 协议及协议字段信息

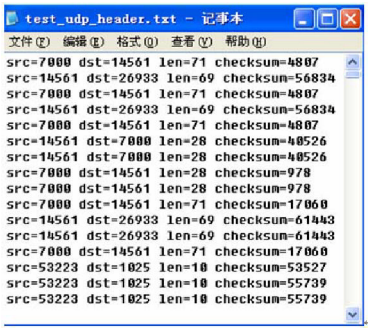

图 6　UDP 协议报头信息

则说明成功，否则协议就没有注册到主程序里来，如图 4 所示。

(2)验证协议解析器能否进行相应的解析。由于新添加的 amin 协议在真实的网络通信中还不存在，所以在真实的网络数据包中，没有符合 amin 协议数据包，因此在验证 Wireshark 系统在加入 amin 协议解析器时能不能解析进行相应的解析时，需要构造一个 amin 伪数据包。本文的实验是使用 Iris 工具构造的，并将其发送到网络上，然后打开 Wireshark 进行抓包，如果抓到一个 amin 协议类型的数据包，经过各层协议解析后，最后到 tcp 解析时，它在解析完报头时，解析载荷时就交给 amin 解析器来解析，如图 5 所示。

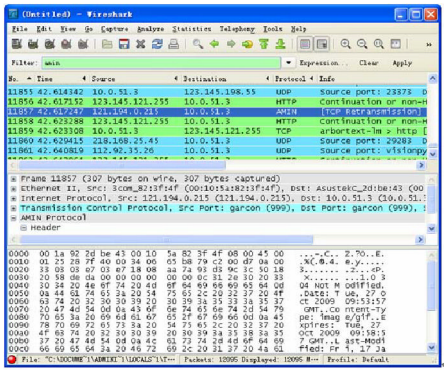

图 5　解析 amin 协议数据报文

4.2　实例应用

在 wireshark 源代码的 packet-udp.c 源文件中的 dissect_udp() 函数内插入下面一段代码，即将报头的 4 个信息源端口、目的端口、包长度、校验和等写入一个源文件。

```
fp=fopen("c:\\test_udp_header.txt","a+");
fprintf(fp,"src=%u ",tvb_get_ntohs(tvb, offset));
fprintf(fp,"dst=%u ",tvb_get_ntohs(tvb, offset+2));
fprintf(fp,"len=%u ",tvb_reported_length(tvb));
fprintf(fp,"checksum=%u",tvb_get_ntohs(tvb, offset+6));
fprintf(fp,"\n");
fclose(fp);
```

重新编译、运行 wireshark 捕捉 UDP 协议数据包，wireshark 在实时捕捉网络数据包的同时，也将相关 UDP 协议数据包报头的 4 个信息写到指定文件中，如图 6 所示。

在可以获得具体协议数据包详细信息的情况下，就可以做更为深度和具体的研究和应用。

5　结束语

随着网络的普及化，网络技术日益更新，网络协议种类繁多，新的协议也层出不穷，需要一个良好的可扩展的协议解析分析工具。Wireshark 的良好系统架构以及开源性迎合了这个需求。本文介绍了 WireShark 的功能和体系结构，在此基础上研究了 WireShark 环境下协议解析器的两种设计方法及其流程，并给出了试验验证，同时也为下一阶段对网络数据包在线应用层包分类和数据内容匹配算法研究工作打下了基础。

参考文献：

[1]　王丽萍，孙雷. 基于 Ethereal 开源代码构建协议解析器的方法研究[J]. 计算机技术与发展，2007，17(10)：27-30.

[2]　Wireshark Developer's Guide [EB/OL]. http://www.wireshark.org/docs/wsdg_html_chunked/，2009.

[3]　Kurose J F，Rose K W. 计算机网络-自顶向下方法与 Internet 特色[M]. 陈鸣，译. 3 版. 北京：机械工业出版社，2008.

[4]　Reed K D. 协议分析[M]. 孙坦，张雪峰，杨琳，等译. 7 版. 北京：电子工业出版社，2005.

[5]　懂立，赵恒永. 基于编译技术的协议解析方法[J]. 计算机工程，2007，26(11)：66-68.

[6]　石美红，延伟伟，李永刚，等. 基于 Ethereal 的 CORBA 通信协议解析功能扩展的方法与实现 [J]. 计算机工程与设计，2005，26(12)：3236-3239.

[7]　MoinMoi，Python.doc/README.developer[EB/OL]. http://wiki.wireshark.org/Development，2009.

[8]　KenThompson. Creating your own custom wireshark dissector [EB/OL]. http://www.codeproject.com/KB/IP/custom_dissector.aspx，2009.