Custom Protocol Analysis Based on Wireshark

ZHANG Peng [1], *XIA Ke-jian* [2]
1. University of Science&Technology Beijing, USTB, Beijing, China
2. University of Science&Technology Beijing, USTB, Beijing, China
1. zhang_peng18@163.com, 2. bjxkj@vip.163.com

Abstract: Wireshark (formerly Ethereal) is currently the more popular form of computer network debugging and packet sniffing software, in order to better resolve the custom protocol, the paper achieved a custom protocol analysis system under the win32 platform using Wireshark source code. The paper discusses the main modules and scalable system architecture, analysis the tree protocol analysis strategy of wireshark, describes principles and processes of Wireshark with GTEL as an example, give part of the code. Finally system showed results of the analysis for GTEL protocol perfectly. System can run with good performance.

Keywords: Ethereal; Wireshark; GTEL; Protocol; Protocol Parser

基于 wireshark 的自定义协议分析

张鹏 [1]，**夏克俭** [2]
1. 北京科技大学，北京，中国，100083
2. 1. 北京科技大学，北京，中国，100083
1. zhang_peng18@163.com, 2. bjxkj@vip.163.com

【摘要】Wireshark（前称 Ethereal）是当前较为流行的一种计算机网络调试和数据包嗅探软件论文系统介绍了 Wireshark 的主要功能模块和可扩展的系统结构，分析 Wireshark 树形结构的数据包协议分析策略，阐述了如何在 Win32 平台下利用 Wireshark 开源代码实现自定义协议解析。并以 GTEL 协议为例重点介绍 Wireshark 协议解析的原理和流程，并给出部分代码，最后显示完整的 GTEL 协议分析结果。

【关键词】Ethereal；Wireshark；GTEL；协议；解析器

1 引言

随着网络应用的不断普及和网络技术的不断发展，新的协议层出不穷，这就需要一个具有良好可扩展性的协议解析分析工具。Wireshark 以其良好的系统结构及开源性被越来越多的人所推崇，本文系统介绍了 Wireshark 可扩展性的系统结构，并结合自定义协议 GTEL 阐述如何利用 Wireshark 源代码实现自定义协议分析。

2. Wireshark 系统结构

Wireshark（前称 Ethereal）是当前较为流行的一种计算机网络调试和数据包嗅探软件，Wireshark 具有设计完美、方便用户的 GUI 和过滤选项功能，尤其 Wireshark 以其跨平台性及开源性被越来越多的人所偏爱。

Wireshark 起初是由 Gerald Combs 开发，目前 Wireshark 已经可以支持七百多种协议的解析，其功能可以和商业的网络协议分析系统所媲美。Wireshark 具有良好的可扩展性[1]，能够方便地为系统添加新的协议解析器。开发人员不需要具体了解 Wireshark 系统实现的细节就可以根据系统留出的接口添加自己的协议解析器。

Wireshark 是开源软件项目，发布遵循GNU General Public Licence（GPL协议），Wireshark代码结构清晰，各层报文解析引擎相互独立，通过函数指针实现报文从底层到高层的依次解析。Wireshark解析采用树形结构，更加清晰易懂。

2.1 代码结构

如图 1 所示 wireshark 包括 6 个功能模块[1], 主要模块功能介绍如下：

Core：核心模块，通过函数调用将其他模块连接在一起，源码在根目录。

Epan：wireshark 包分析引擎，源码在 epan 目录。Epan 目录下有几个重要的组成部分如下：Protocol-Tree：保存数据包的协议信息；Dissectors：在 epan/dissector 目录下，各种协议解码器，对于每种协议，解码器都能识别出协议字段，并显示出字段值由于网络协议种类很多，为了使协议和协议间层次关系明显，对数据流里的各个层次的协议能够逐层处理，wireshark 系统采用了协议树的方式；Plugins：一些协议解码器以插件形式实现，源码在 plugins 目录。

Win-/libpcap：这是 Wireshark 的抓包平台，也是协议解析系统的基础。

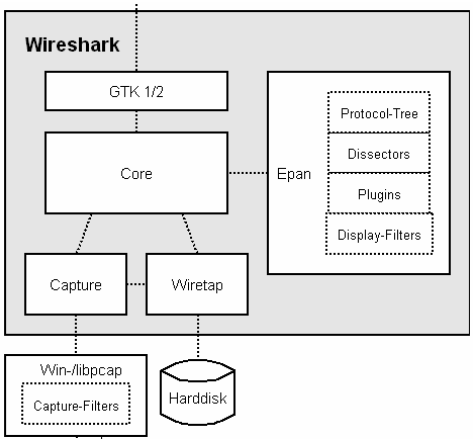

Figure 1. Function module of wireshark
图 1. wireshark 功能模块

2.2 树形结构的数据包协议分析策略

Wireshark采用树形结构[3]的数据包协议分析策略，很直观地表达了协议的层次结构。协议分析从OSI七层协议模型的数据链路层到应用层依次进行。

将最底层的数据流作为根节点，具有相同父节点的协议成为兄弟节点，如图2，IP和ARP是Frame的两个儿子节点，他们互为兄弟节点，IP和ARP又各自拥有自己的子节点，这就构成了一个协议树，在解析新协议时，首先要找到新协议是封装在哪个已有的协议数据中，即新协议的父节点，然后在父节点下继续添加新的协议分支。

Wireshark采用协议的特征字来区别同样父节点的兄弟节点，新协议在添加时都要注册自己的特征字，这些特征字给自己的子节点协议提供可以区分的标识，例如Ethernet协议的ethertype字段注册后ethertype=0x0800就可以认定为是IP协议。

Wireshark首先调用最底层的协议解析函数进行报文解析，处理完成后通过调用函数dissector_try_port(dissector_table_t sub_dissectors, const guint32 port, tvbuff_t *tvb, packet_info *pinfo, proto _ tree *tree)查找协议树找到自己协议下面的子协议，其中参数port就是协议注册的特征字的值，判断应由哪个子协议来执行， 找到正确的子协议后就转交给子协议注册的解析模块处理，特征字加协议树的设计使得wireshark在协议解析上有更强的扩展性，需要增加一个协议解析器时只需要添加一个新的解析函数，然后将这个协议函数挂到协议树的响应父节点上即可。

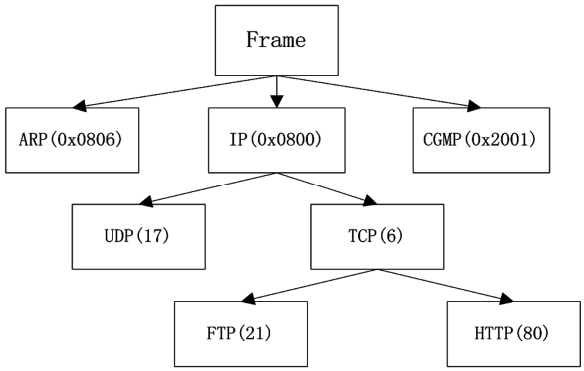

Figure 2. protocol tree of wireshark
图 2. wireshark 协议树

2.3 Wireshark 内置插件

Wireshark 创建协议解析器有两种方式：一种是内置解码器；另一种方法是通过插件技术添加协议解析插件[4]，所谓插件技术，就是在程序的设计开发过程中，把整个应用程序分成宿主程序和插件两个部分，宿主程序与插件能够相互通信，并且，在宿主程序不变的情况下，可以通过增减插件或修改插件来调整应用程序的功能。运用插件技术可以开发出伸缩性良好、便于维护的应用程序。

在 Wireshark 中新增加一个协议插件一般需要插件安装或者注册，插件初始化，插件处理 3 个步骤，具体流程和实现方式将在下文中详细阐述。

3. GTEL 协议解析器

如图所示，GTEL 是北京易通公司自定义的一种协议格式，用于封装 GSM(Global System for Mobile Communications)协议报文，该格式用于电信招标项

978-1-935068-23-5 © 2010 SciRes.

目。GTEL 是 Frame 上的协议，报文解析时首先解析 Frame 层，然后 GTEL 当做 Frame 的一个子节点添加到协议树上，GTEL 特征字是 ethertype=0x9906，因此 ethertype 字段值为 0x9906 的报文到来时，Frame 解析完成后将调用 GTEL 模块继续处理，并把解析的信息添加到 GTEL 分支。Data 是包含的以太网报文。

表 1. GTEL 协议格式
Table 1. format of GTEL protocol
表 1. GTEL 协议格式

DMAC	目的物理地址地段，长度为 6byte。
SMAC	源物理地址地段，长度为 6byte。
Type	协议类型字段，长度为 2byte
Length	报文长度字段，长度为 2byte
GtelHead	GTEL 头部字段，长度为 26byte
Data	数据字段，长度为 nbyte
FCS	校验位，长度为 4byte

3.1 环境搭建

开发环境的搭建是一个系统开发的前提与保证，为了检测 Wireshark 开发环境是否已经搭建成功，在获得 Wireshark 源代码之后，首先需要成功编译源代码[5]。Win32 平台下开发 Wireshark 协议解析插件需要按照如下几个步骤来完成：

第一步：安装 cygwin。cygwin 是一个在 Windows 平台上运行的 Unix 模拟环境。安装过程中要确保以下几个包被成功安装：Archive / unzip；Deve1 / bison： Devel / flex； Interpreters / perl；Interpreters / python；Utils / patch；Web / wget。

第二步：安装 VS2005，Win32 平台下开发需要使用 VS2005 自带的 CL 编译器和 link，nmake 工具。

第三步：path 设置和编译环境监测。将 cygwin 的路径添加到 path 变量中，使用命令 nmake -f Makefile.nmake verifytools 确认所需的工具都已经安装。

第四步:运行库下载。Wireshark网站提供所有在Win32平台下开发Wireshark协议解析器所需库文件，只需要执行nmake -f Makefile.nmake setup命令，系统将会自动下载所有的库文件。

第五步：开始编译。执行nmake -f Makefile.make all命令，成功编译后将在Wireshark源代码的根目录中生成Wireshark的可执行文件，表明开发环境已经成功搭建。

3.2 解析协议

在 Wireshark 协议分析器中新增加一个协议插件需要插件安装或者注册，插件初始化，插件处理三个步骤。

3.2.1 插件注册

```
static int proto_gtel = -1;
void proto_register_gtel( void )
{
    proto_gtel = proto_register_protocol("Gtel Protocol","Gtel", "gtel" );
    proto_register_field_array(proto_gtel,hf_r, array_length( hf_r ) );
    proto_register_subtree_array(ett_r, array_length( ett_r ) );
    register_dissector( " gtel", dissect_ gtel, proto_ gtel );
}
```

3.2.2 插件提交

```
#define ETHERNET_TYPE_GTEL 0x9906
void proto_reg_handoff_gtel (void)
{
        gtel_handle = create_dissector_handle(dissect_gtel, proto_gtel);
        dissector_add( "ethertype", ETHERNET_TYPE_GTEL, gtel_handle );
        wtap_gtel_dissector_table = find_dissector_table("wtap_encap");
}
```

在这里使得插件处理函数和主函数取得联系，这段代码告诉系统当报文的ethertype的字段值为0x9906的时候要调用dissect-gtel这个函数模块，即在协议树的Frame分支下继续添加协议分支，这里的dissect-gtel函数模块是所要编写的GTEL解析函数。

3.2.3 插件处理

```
static void dissect_gtel ( tvbuff_t * tvb,  packet_info * pinfo,  proto_tree * tree )
{
    ti_gtel = proto_tree_add_text( tree,  tvb, offset, -1, "Gtel" );
    gtel_tree = proto_item_add_subtree( ti_gtel, ett_gtel);
    proto_tree_add_uint(gtel_tree, hf_gtel_head_len, tvb, offset, 1, tvb_get_letohs( tvb, offset ) );
}
```

Wireshark将Winlap捕获的报文放置在tvb这个缓冲区中，解析工作将tvb中的报文按照字节提取出来，经过分析后将协议的信息通过系统接口函数不断地将这些信息添加到gtel -tree这个协议子树的相应节点上。其中添加分枝节点函数，根据协议信息在协议树上的显示方式不同共有如下几种：

proto_tree_add_text(proto_tree *tree, tvbuff_t *tvb, gint start, gint length, const char *format, ...)

proto_tree_add_int(proto_tree *tree, int hfindex, tvbuff_t *tvb, gint start,gint length, gint32 value)

proto_tree_add_boolean(proto_tree *tree, int hfindex, tvbuff_t *tvb, gint start,gint length, guint32 value)

proto_tree_add_string(proto_tree *tree, int hfindex, tvbuff_t *tvb, gint start,gint length, const char* value)

这些接口函数的参数含义如下：参数(tree)指当前要添加的分枝所在的子树节点；参数(hfindex)是节点索引标记，用来控制节点信息的显示方式；参数(tvb)指向报文缓冲区的指针，用于存放Wincap捕获的报文；参数(value)表示需要添加在协议树上的解析后的协议信息，其中参数start和length用来控制Wireshark中反白显示报文的位置和长度。

报文中GTEL中封装的是以太网的报文，所以在解析完GTEL报文以后还需要将上层报文交给以太网解析函数继续进行解析，所以在解析完GTEL层报文后应该调用函数dissector_try_port查找GTEL上层解析函数对报文继续解析：

dissector_try_port(wtap_gtel_dissector_table, pinfo->fd->lnk_t,next_tvb, pinfo, gtel_tree)

参数wtap_gtel_dissector_table指向底层协议解析引擎列表[6]，通过链路类型查找上层解析函数，至此GTEL层解析完成，继续上层协议解析。

当完成插件处理函数以后可以单独编译GTEL插件处理函数，最后只要将编译生成的.dll文件放置到系统用来存放协议插件的目录中，这时系统在运行的时候当遇到符合GTEL特征字的报文时将会自动调用此插件来执行解析工作。GTEL报文的解析如图所示。展开GTEL节点将显示GTEL协议规范定义的各个字段的详细信息。在GTEL之下就是以太网子树节点，展开此节点可以看到封装在GTEL中的以太网对应的各个字段值，解析结果如图3所示。

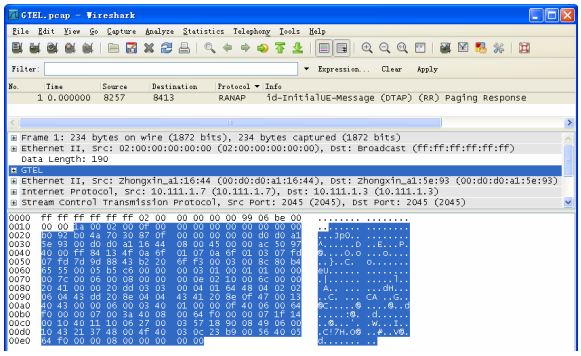

Figure 3. performance of GTEL protocol parser
图 3. 添加 GTEL 解析器后报文解析结果

4. 结论

Wireshark 的优势在于其良好的系统结构及开源性，这使得开发人员不需要详细了解 Wireshark 系统的具体实现，就能利用 Wireshark 所预留的接口函数就可以很轻松地添加一个自己的协议解析器。论文在 Wireshark 的基础上实现了自定义协议的解析，能够很好的完成既定目标，很好的实现了对 GTEL 协议的解析。

致谢

论文写作过程中遇到很多困难，在此，要特别感谢我的导师夏克俭教授，是他指导着我努力的方向；还要感谢易通科技有限公司的领导和前辈，在我的论文写作中给予无私的支持和帮助，使得论文能够与实践相结合。再次感谢他们。

References (参考文献)

[1] Lamping U. Wireshark Developer's Guide [EB / OL]. www.wireshark.com.2010.

[2] Kurose J F, Ross K W. Computer network - top-down method and Intemet characteristics [M]. Chen Ming, and other translations. Beijing: Mechanical Industry Press, 2005. 200-205.

[3] StallingsW. Internet Protocol and Technology [M]. Beijing: Electronic Industry Press, 2004.36-45.

[4] Tanenbaum A S. Computer Network [M]. 4th ed. Beijing: Tsinghua University Press, 2004, 245-250.

[5] Reed K D. Protocol analysis [M]. 7th ed. Sun Tan, Zhang Xuefeng, Yang Lin, M., Beijing: Electronic Industry Press, 2005. 123-125.
Reed K D. 协议分析[M]. 第 7 版. 孙坦，张学锋，杨琳等译. 北京：电子工业出版社，2005. P123-125

[6] Liu Wentao. Network Security Technology and programming examples [M]. Beijing: Mechanical Industry Press ,2008.55-63.
刘文涛. 网络安全编程技术与实例[M]. 北京：机械工业出版社，2008. P55-63

978-1-935068-23-5 © 2010 SciRes.