tshark 增加保存http图片、音乐功能

github: https://github.com/KevinsBobo/wireshark-modify

tshark 捕获、解析逻辑

```
tshark.c
main()
    解析命令行参数
    初始化环境
    -->
    caption()
        初始化信息
        while(…)
            --> capchild\capture_sync.c -> syn_pipe_input_cp(…) // 回调的方式
                从管道(文件)中读取数据头部信息
```

```
                如果读到的是一个数据包的头部则开始处理
                --> capture_input_new_packets(capture_session …)
                    根据传入的capture_session获得一个 capture_file 类型的数据指针
                    如需解码:
                        获得一个 epan_dissect_t 类型的空间
                        并根据 capture_file 结构体初始化数据
                        // edt 中将包含一个具有链表结构的数据包
                        --> process_packet(capture_file *cf, epan_dissect_t *edt …)
                            解析数据
                            --> epan_dissect_run_with_taps(…)
                                获得 edt 中的 tvbuff_t 类型的数据指针 // 具有链表结构的数据包
                            --> print_packet(…)
                                打印数据包
                                --> print_columns()
                                    以列的形式打印数据
                                    通过 cf 获得数据包的列结构数据
                                    // 编号、时间 IP TCP/UDP 端口 类型 数据类型
                                    // 我是在这里通过字符串比较判断是否为 HTTP 图片 jpg/png/gif
                                    如果通过参数设置需要打印数据包16进制数据包
                                        则继续打印16进制数据
                                        这里打印的16进制数据是数据包的原格式
                                        后面我借鉴了这里面对 tvbuff_t 类型数据的分解的代码
                                        从而得到图片数据
```

调试、分析过程

抓包时调试、分析

> `tshark src tcp 80`

1. 单步找到 `caption() capture_input_new_packets() print_packet() print_columns()` 等函数
2. 通过vs2013下 `fprintf()` API断点并通过栈回溯分析函数调用情况(因为tshark是通过 `fprintf` 输出数据的)
3. 通过API断电发现了 `print_columns()` 的输出 `HTTP` 状态的位置

分析数据包时调试

> `tshark -r out.pcap -Y http -x`

1. 在 `printf_columns()` 中发现了输出16进制数据的 `print.c -> print_hex_data()` 函数
2. 通过条件断点发现这里输出的数据是抓取到的整个数据包
3. 于是找到了 `print_hex_data()` 中使用 `tvbuff_t` 的代码

保存图片新增代码

> 注：在tshark原代码中新增的代码都以 `/* 保存图片新增 start */ ... /* 保存图片新增 stop */` 的形式标注
> 调试新增代码都以 `/* 调试新增 start */ ... /* 调试新增 stop */` 的形式标注
>
> 为方便管理，新增代码的实现和声明放在了单独的 `addsave-....c` 和 `addsave-....h` 文件中，位于wireshark源码目录的 `addsave` 目录中
> 所有函数和全局变量均以 `addsave-...` 开头

addsave-file.h

> 这里的函数是文件相关的，一个实创建目录，一个是初始化文件，没什么说的

```
#define ADDSAVE_MAXPATHLEN 255

void addsave_init_floder(char* szFloder);

int addsave_init_file(char* szFile);
```

addsave-save.h

> 这是保存图片的主要代码

```
/* 照抄 tsark.h 包含的头文件 start */
… // 省略
/* 照抄 tsark.h 包含的头文件 stop */

/* 新增的头文件 start */
#include <epan/tvbuff-int.h>
#include <addsave/addsave-file.h>
/* 新增的头文件 stop */

// 图片根目录
#define ADDSAVE_FILE_FLODER_NAME "..\\addsave_save\\"
// 图片名前缀
#define ADDSAVE_FILE_NAME "save_file"

// 全局变量 标记数据包是否为图片
extern int addsave_g_is_savefile;
// 全局变量 标记数据包来源IP
extern char addsave_g_src_ip[ ADDSAVE_MAXPATHLEN ];

// 图片类型
#define ADDSAVE_PIC_JPG 1
#define ADDSAVE_PIC_PNG 2
#define ADDSAVE_PIC_GIF 3
```

```
#define ADDSAVE_FILE_AUDIO 4

// 主要功能函数
void addsave_save_pic(epan_dissect_t * edt);
```

addsave-save.c

> 实现代码

```
#include <addsave/addsave-save.h>

int addsave_g_is_savefile = 0;
char addsave_g_src_ip[ ADDSAVE_MAXPATHLEN ];

void addsave_save_pic(epan_dissect_t * edt)
{
  if(edt == NULL)
  {
    return ;
  }

  char szFilePath[ MAXPATHLEN ] = { 0 };
  char* pszFileType = NULL;
  tvbuff_t * tvb = NULL;
  u_char* pData = NULL;
  unsigned long long *pVerify = NULL;
  FILE* fpPic = NULL;
  time_t save_time = 0;
  time(&save_time);
  static int nTime = 0;
  const guchar *cp = NULL;
  guint         length = 0;

  // gboolean      multiple_sources;
  GSList       *src_le = NULL;
  struct data_source *src = NULL;


  // 将 tvb 指针移动到合适的位置
  for( tvb = edt->tvb ; tvb != NULL; tvb = tvb->next)
  {
    // 可以确定 jgp/gif 图片肯定在最后一个数据包
    // PNG 图片在倒数第六个数据包
    // 所以直接将 jpg/gif 的指针移到最后一个数据包的位置
    if(((addsave_g_is_savefile == ADDSAVE_PIC_JPG
      || addsave_g_is_savefile == ADDSAVE_PIC_GIF
      || addsave_g_is_savefile == ADDSAVE_FILE_AUDIO)
         && tvb->next != NULL))
    {
```

```
      continue;
    }

    if(tvb->real_data == NULL)
    {
      return ;
    }

    // jpg 数据首地址在最后一个数据包地址的前两个字节的位置，png 和 gif 则正常
    pData = (unsigned char*)(tvb->real_data);
    pVerify = (unsigned long long*)tvb->real_data;
    // 再次判断，匹配则跳出循环，按照逻辑只有png才会多次判断
    if((*(unsigned short*)(pData - 2) == 0xD8FF && *pVerify ==
0x4649464A1000E0FF)       // jpg
       || (*(unsigned long long*)(pData) == 0x0A1A0A0D474E5089)
         // png
       || (((*(unsigned long long*)(pData)) & 0x0000FFFFFFFFFFFF) ==
0x0000613938464947) // gif
       || addsave_g_is_savefile == ADDSAVE_FILE_AUDIO   // audio 类型直接跳出
       )
    {
      break;
    }
  }


  /* 参考自 print.c -> print_hex_data 函数 */
  // 获取数据长度 和 http 数据包首部指针
  for(src_le = edt->pi.data_src; src_le != NULL;
      src_le = src_le->next)
  {
    if(src_le->next != NULL)
    {
      continue;
    }

    src = (struct data_source *)src_le->data;
    tvb = get_data_source_tvb(src);
    length = tvb_captured_length(tvb);
    if(length == 0)
      return ;
    // 获取http数据首部指针
    cp = tvb_get_ptr(tvb , 0 , length);

    if(cp == NULL)
    {
      return ;
    }

    break;
  }
```

```
89
90  if(addsave_g_is_savefile == ADDSAVE_PIC_JPG)
91  {
92    // 偏移指针
93    pData -= 2;
94    pszFileType = "jpg";
95  }
96  else if(addsave_g_is_savefile == ADDSAVE_PIC_PNG)
97  {
98    pszFileType = "png";
99  }
100 else if(addsave_g_is_savefile == ADDSAVE_PIC_GIF)
101 {
102   pszFileType = "gif";
103 }
104 else if(addsave_g_is_savefile == ADDSAVE_FILE_AUDIO)
105 {
106   pszFileType = "mp3";
107 }
108
109 if(pszFileType == NULL)
110 {
111   return ;
112 }
113
114 sprintf_s(szFilePath , MAXPATHLEN , "%s%s\\%s\\" ,
115           ADDSAVE_FILE_FLODER_NAME , addsave_g_src_ip, pszFileType);
116 // 创建文件夹
117 addsave_init_floder(szFilePath);
118
119 sprintf_s(szFilePath ,
120           MAXPATHLEN ,
121           "%s%s\\%s\\%s_%lld.%s" ,
122           ADDSAVE_FILE_FLODER_NAME ,
123           addsave_g_src_ip,
124           pszFileType ,
125           ADDSAVE_FILE_NAME,
126           save_time + nTime++,
127           pszFileType);
128
129 // 创建文件
130 if(addsave_init_file(szFilePath))
131 {
132   return ;
133 }
134
135 // 打开文件
136 fopen_s(&fpPic , szFilePath , "wb");
137 if(fpPic == NULL)
138 {
139   return ;
```

```
  }

  // 获取文件长度
  u_int pic_length = length - (pData - cp);

  // 写文件
  fwrite(pData , pic_length , 1 , fpPic);

  // 关闭文件
  fclose(fpPic);
}
```

tshrk.c新增代码

头文件

> 将 `addsave-....c` 文件加入到tshark项目工程中并包含新增代码的头文件

```
/* 保存图片新增 start */
#include <addsave/addsave-save.h>
#include <addsave/addsave-file.h>
/* 保存图片新增 stop */
```

print_columns()函数中

> 这里主要判别是否为图片，并设置标志位

```
static gboolean
print_columns(capture_file *cf)
{
  …
  switch (col_item->col_fmt) {
    …
    case COL_UNRES_NET_SRC:
      column_len = col_len = strlen(col_item->col_data);
      if (column_len < 12)
        column_len = 12;
      line_bufp = get_line_buf(buf_offset + column_len);
      put_spaces_string(line_bufp + buf_offset, col_item->col_data, col_len,
column_len);
      /* 保存图片新增 start */
      strcpy_s(addsave_g_src_ip , ADDSAVE_MAXPATHLEN , col_item->col_data);
      /* 保存图片新增 stop */
      break;
    …
    default:
      column_len = strlen(col_item->col_data);
```

```
      line_bufp = get_line_buf(buf_offset + column_len);
      put_string(line_bufp + buf_offset, col_item->col_data, column_len);
      /* 保存图片新增 start */
      if(!strcmp(col_item->col_data , "HTTP/1.1 200 OK  (JPEG JFIF image)"))
      {
        addsave_g_is_savefile = ADDSAVE_PIC_JPG;
      }
      else if(!strcmp(col_item->col_data , "HTTP/1.1 200 OK  (PNG)"))
      {
        addsave_g_is_savefile = ADDSAVE_PIC_PNG;
      }
      else if(!strcmp(col_item->col_data , "HTTP/1.1 200 OK  (GIF89a)")
              || !strcmp(col_item->col_data , "HTTP/1.1 200 OK  (GIF89a)
(image/jpeg)"))
      {
        addsave_g_is_savefile = ADDSAVE_PIC_GIF;
      }
      else if(!strcmp(col_item->col_data , "HTTP/1.1 206 Partial Content
(audio/mpeg)")
              || !strcmp(col_item->col_data , "HTTP/1.0 206 Partial Content
(audio/mpeg)"))
      {
        addsave_g_is_savefile = ADDSAVE_FILE_AUDIO;
      }
      else
      {
        // 保险措施
        addsave_g_is_savefile = 0;
      }
      /* 保存图片新增 stop */
```

process_packet()函数中

> 是在获取 `edt` 中 `tvb` 指针数据后调用保存图片的函数

```
static gboolean
process_packet(capture_file *cf, epan_dissect_t *edt, gint64 offset, struct
wtap_pkthdr *whdr,
               const guchar *pd, guint tap_flags)
{
  …
  if (passed) {
    frame_data_set_after_dissect(&fdata, &cum_bytes);


    /* Process this packet. */
    if (print_packet_info) {
      /* We're printing packet information; print the information for
         this packet. */
```

```
14        print_packet(cf, edt);
15
16        /* 保存图片新增 start */
17        if(addsave_g_is_savefile)
18        {
19          addsave_save_pic(edt);
20          addsave_g_is_savefile = 0;
21        }
22        /* 保存图片新增 stop */
23    …
24
25    /* 保存图片新增 start */
26    // 保险措施，标志位置 0
27    addsave_g_is_savefile = 0;
28    /* 保存图片新增 start */
29
30    return passed;
31  }
```

使用方法

编译

1. 将 `addsave` 文件夹拷贝到 `wirshark` 源码目录中
2. 替换 `tshark.c` 文件
3. 打开 `wirshark` vs2013解决方案
4. 将 `addsave` 中的 `addsave-save.c` 和 `addsave-file.c` 添加到 `tshark` 项目 `Soure Files` 中
5. 重新编译 `tshark`

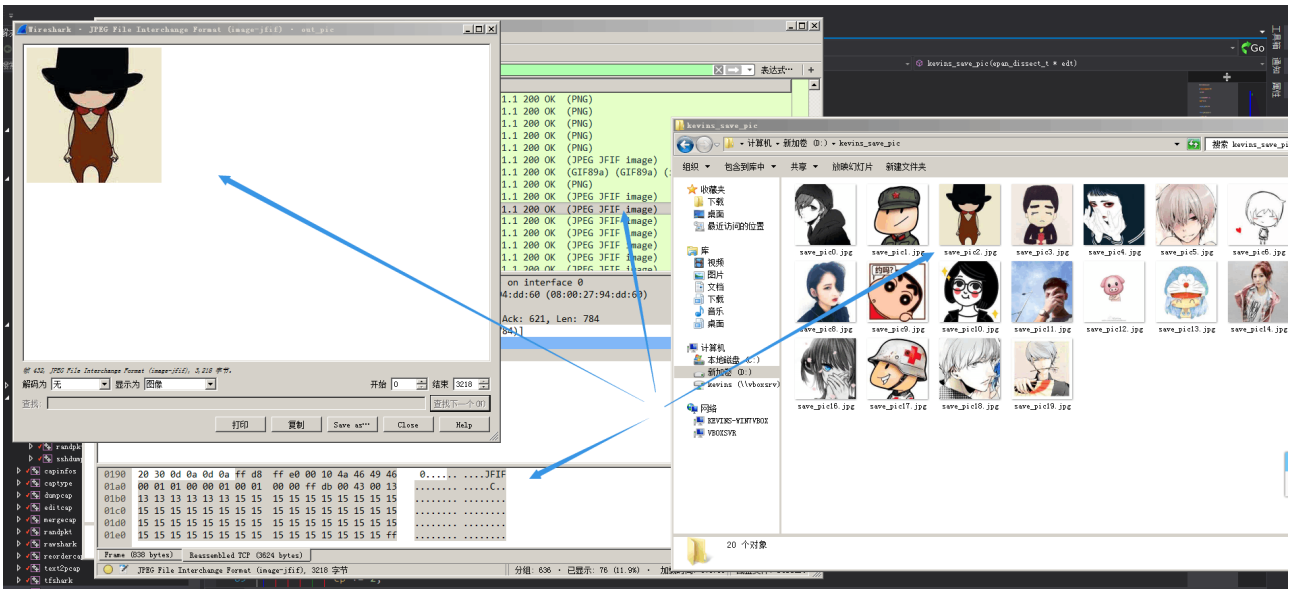
使用

1. 捕获网卡信息流时保存 `tshark src port 80`
2. 从 `.pcap` 文件中保存文件 `tshark -r out.pcap -Y http`
3. 保存的文件在执行目录的上级目录中的 `addsave_save` 目录中按照IP和文件类型分文件夹保存

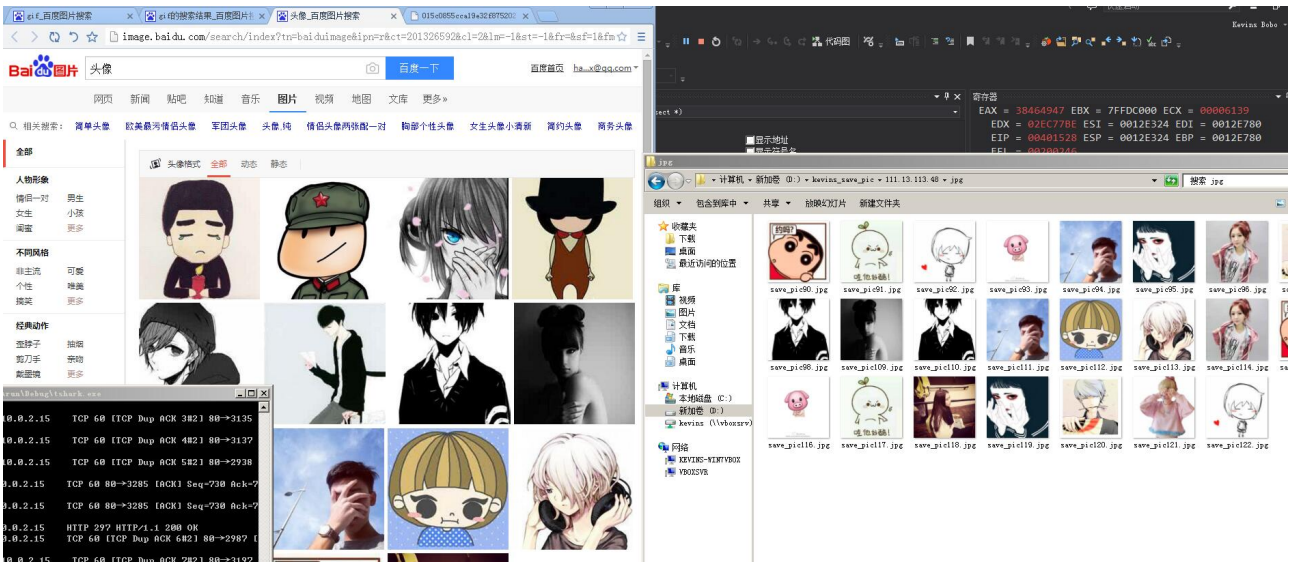
效果展示

从已经捕获的数据包中抓取图片

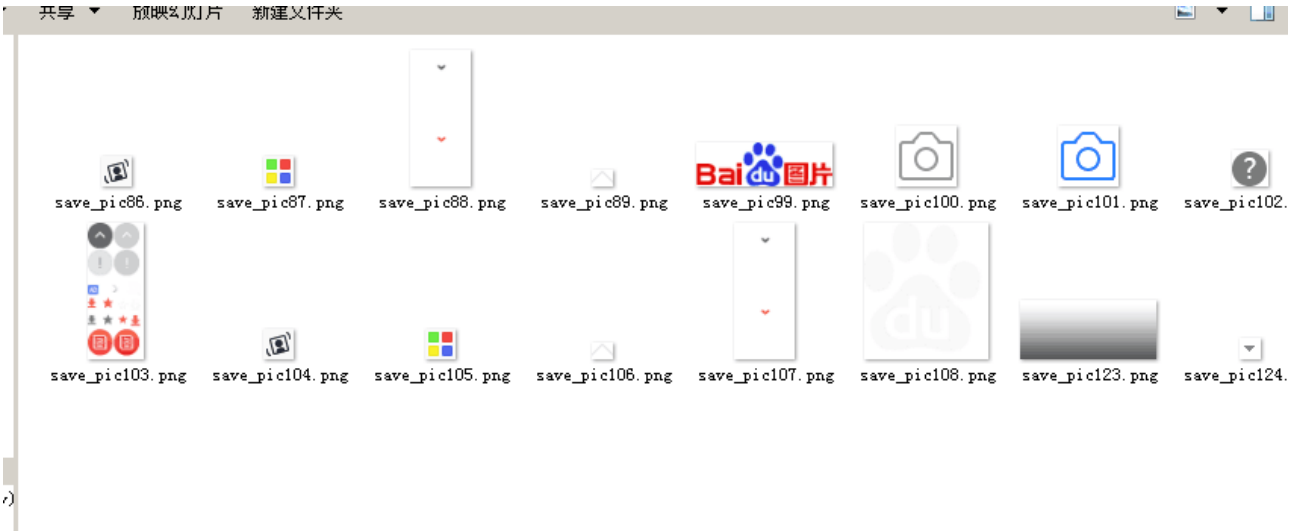
> `tshark -r out.pcap`



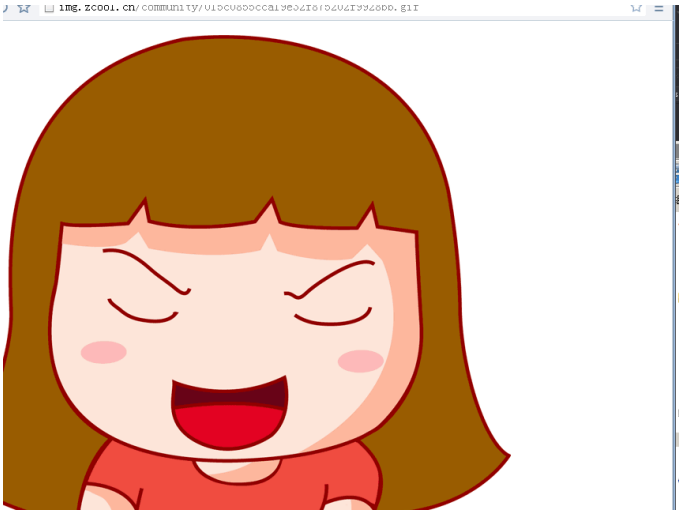
在捕获数据包同时保存图片

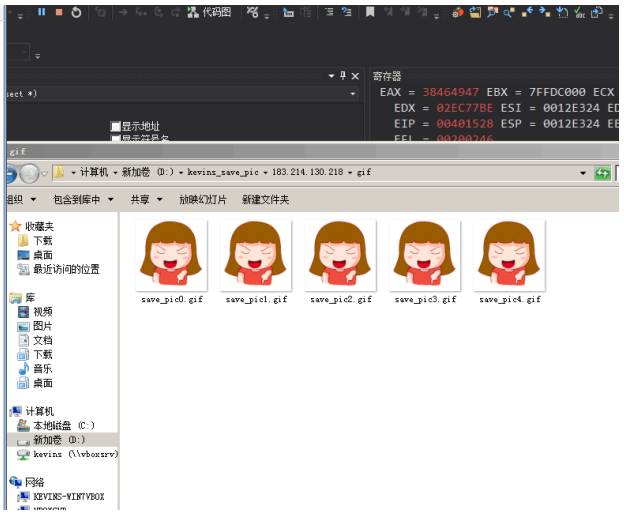


png 图片



gif 图片


img.zcool.cn/community/


代码图
寄存器
EAX = 38464947 EBX = 7FFDC000 ECX
EDX = 02EC77BE ESI = 0012E324 ED
EIP = 00401528 ESP = 0012E324 EB
显示地址
gif
计算机 ▸ 新加卷 (D:) ▸ kevins_save_pic ▸ 183.214.130.218 ▸ gif
组织
包含到库中
共享
放映幻灯片
新建文件夹
收藏夹
下载
桌面
最近访问的位置
库
视频
图片
文档
下载
音乐
桌面
计算机
本地磁盘 (C:)
新加卷 (D:)
kevins (\\vboxsrv)
网络
KEVINS-WIN7VBOX
save_pic0.gif
save_pic1.gif
save_pic2.gif
save_pic3.gif
save_pic4.gif