使用Wireshark查看Userlog方法

郑鑫 2015-07-06 发表

客户局点有一台userlog服务器，无法正常解析到我司防火墙发送的userlog日志。为证明服务器是否已经正常收到我司防火墙发送的userlog日志，可以再服务器侧抓包判断。同时，一般wireshark无法正常解析userlog，需要对wireshark做一定的修改。

1、 拷贝 UserLog-new.lua到wireshark的安装目录(注意:此处拷贝到安装目录下后，建议打开文件保存一下，查看是否会在安装目录下生生一个UserLog-new.lua.bak的文件。)

```
--将FW输出的userlog信息进行解析
--by H3C TSC navy
--2014.4.8
-- 创建一个新的解析器
local userlog = Proto ("UserLog", "H3C Firewall Session Userlog")
-- 定义解析函数
function userlog.dissector( buffer, pinfo, tree )

 --向子树添加字段
 subtree  = tree:add (userlog, buffer())
 offset   = 0

 --修改columns
 pinfo.cols.protocol = "UserLog"
 pinfo.cols.info     = "Flow Version is "
 pinfo.cols.info:append(buffer(0, 1):uint())
 pinfo.cols.info:append("  Flow Count = ")
 pinfo.cols.info:append(buffer(2, 2):uint())


 --添加userlog报文头
 subtree:add("----------UserLog Head-----------by H3C TSC navy")
 subtree:add(f.version,       buffer(0, 1))
 subtree:add(f.logtype,       buffer(1, 1))
 subtree:add(f.count,         buffer(2, 2))
 subtree:add(f.timestamp,     buffer(4, 4))

 --添加userlog报文信息，根据count字段来确定需要添加多少段信息
 version = buffer(0, 1):uint()
 if version == 3 then
  --V3 版本流日志解析
  count_num = 1
  while count_num <= buffer(2, 2):uint() do
   subtree:add("----------UserLog no:" .. count_num .. "----------")
   subtree:add(f.proto,       buffer(16 + 64*(count_num - 1), 1))
   subtree:add(f.Operator,    buffer(17 + 64*(count_num - 1), 1))
   subtree:add(f.IPVerion,    buffer(18 + 64*(count_num - 1), 1))
   subtree:add(f.IPToS,       buffer(19 + 64*(count_num - 1), 1))

   subtree:add(f.SourceIP,    buffer(20 + 64*(count_num - 1), 4))
   subtree:add(f.SrcNatIP,    buffer(24 + 64*(count_num - 1), 4))
   subtree:add(f.DestIP,      buffer(28 + 64*(count_num - 1), 4))
   subtree:add(f.DestNatIP,   buffer(32 + 64*(count_num - 1), 4))
   subtree:add(f.SrcPort,     buffer(36 + 64*(count_num - 1), 2))
   subtree:add(f.SrcNatPort,  buffer(38 + 64*(count_num - 1), 2))
   subtree:add(f.DestPort,    buffer(40 + 64*(count_num - 1), 2))
   subtree:add(f.DestNatPort, buffer(42 + 64*(count_num - 1), 2))

   subtree:add(f.StartTime,   buffer(44 + 64*(count_num - 1), 4))
   subtree:add(f.EndTime,     buffer(48 + 64*(count_num - 1), 4))

   subtree:add(f.InTotalPkg,  buffer(52 + 64*(count_num - 1), 4))
   subtree:add(f.InTotalByte, buffer(56 + 64*(count_num - 1), 4))
   subtree:add(f.OutTotalPkg, buffer(60 + 64*(count_num - 1), 4))
```

```
   subtree:add(f.OutTotalByte,buffer(64 + 64*(count_num - 1), 4))

   subtree:add(f.Reserved1,   buffer(68 + 64*(count_num - 1), 4))
   subtree:add(f.Reserved2,   buffer(72 + 64*(count_num - 1), 4))
   subtree:add(f.Reserved3,   buffer(76 + 64*(count_num - 1), 4))

   count_num = count_num + 1
  end
 else
  --V1 版本流日志解析，无数据暂未验证
  count_num = 1
  while count_num <= buffer(2, 2):uint() do
   subtree:add("----------UserLog no:" .. count_num .. "----------")

   subtree:add(f.SourceIP,    buffer(16 + 64*(count_num - 1), 4))
   subtree:add(f.DestIP,      buffer(20 + 64*(count_num - 1), 4))
   subtree:add(f.SrcPort,     buffer(24 + 64*(count_num - 1), 2))
   subtree:add(f.DestPort,    buffer(26 + 64*(count_num - 1), 2))

   subtree:add(f.StartTime,   buffer(28 + 64*(count_num - 1), 4))
   subtree:add(f.EndTime,     buffer(32 + 64*(count_num - 1), 4))

   subtree:add(f.proto,       buffer(36 + 64*(count_num - 1), 1))
   subtree:add(f.Operator,    buffer(37 + 64*(count_num - 1), 1))

   --subtree:add(f.Reserved,    buffer(38 + 64*(count_num - 1), 40))

   count_num = count_num + 1
  end
 end

end

-- 创建协议字段
f = userlog.fields
--format = {"Text", "Binary", }
--userlog 头共 16 Byte， 包括：版本version-1Byte;日志类型logtype-1Byte;包含日志数量count-2Byte;
发送时间戳timestamp-4Byte;其余未知
f.version   = ProtoField.uint8("FWVer", "FW Version", base.DEC, {[1]="V1",
               [3]="V3"})
f.logtype    = ProtoField.uint8 ("LogType", "Log Type" , base.DEC, {[1]="NAT",
                 [2]="BAS",
               [4]="Flow"})
f.count      = ProtoField.uint16 ("LogCount", "Log Count" )
f.timestamp  = ProtoField.absolute_time ("TimeStamp", "Time Stamp" )
--userlog 日志共 64 Byte， 包括：承载协议Prot-1Byte; 操作字Operator-1Byte;
--                          IP版本IPVersion-1Byte; IP ToS-1Byte;
--                          源地址SourceIP-4Byte; NAT后源地址SrcNatIP-4Byte; 目的地址DestIP-4Byte; N
AT后目的地址DestNatIP-4Byte;源端口号SrcPort- 2Byte;NAT后源端口号SrcNatPort- 2Byte; 目的端口
DestPort- 2Byte;NAT后目的端口DestNatPort 2Byte;
--        流开始时间StartTime- Byte;流结束时间EndTime- Byte;
--                          接收报文数InTotalPkg- Byte；接收报文字节InTotalByte- Byte; 发出报文数OutTo
talPkg- Byte；发出报文字节OutTotalByte- Byte
--                          Reserved1 对于0x02版本（FirewallV200R001）保留，对于0x03版本
（FirewallV200R005）第一个字节为源VPN ID，第二个字节为目的VPN ID，第三、四个字节保留
--                          Reserved2 Reserved3 保留

f.proto      = ProtoField.uint8 ("Prot", "Protocol" ,base.DEC,  {[1]="ICMP",
               [17]="UDP",
               [6]="TCP",
               [112]="VRRP",
               [89]="OSPF"})
f.Operator   = ProtoField.uint8 ("Operator", "Operator" , base.DEC, {[0]="reverse",
                [1]="normal close flow",
```

```
            [2]="timeout",
            [3]="clear flow",
            [4]="overflow",
            [5]="nat static",
            [6]="time data  threshold",
            [7]="flow delete",
            [8]="flow create"})
f.IPVerion  = ProtoField.uint8 ("IP-Version", "IP Version" )
f.IPToS     = ProtoField.uint8 ("IP-ToS", "IP ToS" )
f.SourceIP  = ProtoField.ipv4 ("Source-IP", "Source IP" )
f.SrcNatIP  = ProtoField.ipv4 ("Source-NAT-IP", "Source NAT IP" )
f.DestIP    = ProtoField.ipv4 ("Destnation-IP", "Destnation IP" )
f.DestNatIP  = ProtoField.ipv4 ("Destnation-NAT-IP", "Destnation NAT IP" )
f.SrcPort    = ProtoField.uint8 ("Source-Port", "Source Port" )
f.SrcNatPort = ProtoField.uint8 ("Source-NAT-Port", "Source NAT Port" )
f.DestPort   = ProtoField.uint8 ("Destnation-Port", "Destnation Port" )
f.DestNatPort= ProtoField.uint8 ("Destnation-NAT-Port", "Destnation NAT Port" )
f.StartTime  = ProtoField.absolute_time ("StartTime", "StartTime" )
f.EndTime    = ProtoField.absolute_time ("EndTime", "EndTime")
f.InTotalPkg  = ProtoField.uint32 ("InTotalPkg", "InTotalPkg" )
f.InTotalByte = ProtoField.uint32 ("InTotalByte", "InTotalByte" )
f.OutTotalPkg = ProtoField.uint32 ("OutTotalPkg", "OutTotalPkg" )
f.OutTotalByte= ProtoField.uint32 ("OutTotalByte", "OutTotalByte" )
f.Reserved1   = ProtoField.uint32 ("Reserved1", "Reserved1" )
f.Reserved2   = ProtoField.uint32 ("Reserved2", "Reserved2" )
f.Reserved3   = ProtoField.uint32 ("Reserved3", "Reserved3" )

--初始化
packet_counter = 0
function userlog.init( ... )
 packet_counter = 0
end
--注册解析器 UDP 端口号为40000
udp_table = DissectorTable.get("udp.port")
udp_table:add (40000, userlog)
```

2、 对于新版本的wireshark缺省打开对于lua的支持，无需配置，老版本需要打开以下lua的开关，开启方法：

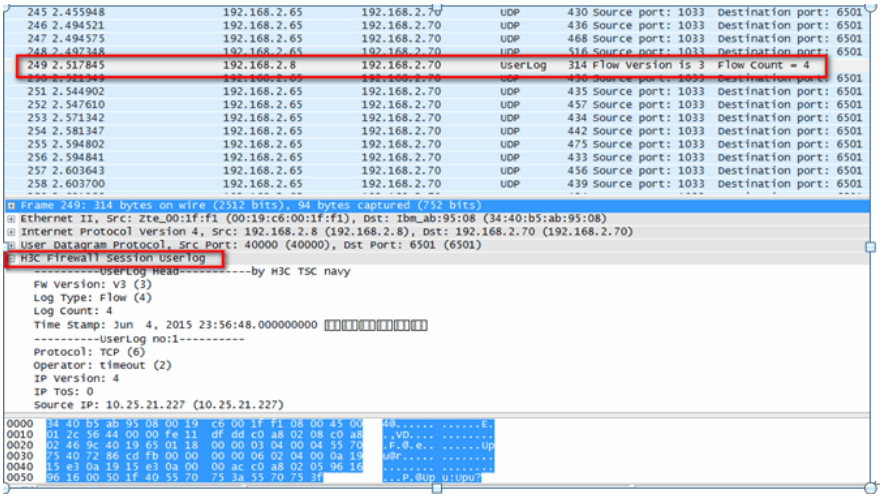
打开init.lua文件，做如下设置

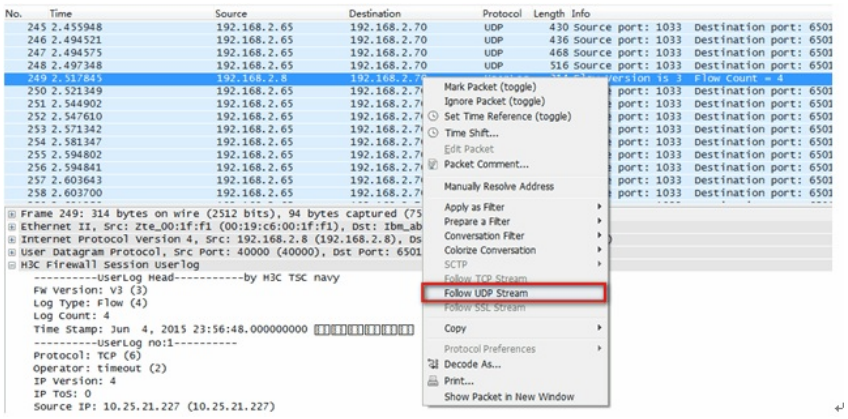
```
-- Set disable_lua to true to disable Lua support.
disable_lua = false

if disable_lua then
    return
end
```

修改同目录下的 init.lua 文件，在最后一行增加对UserLog-new.lua的调用：

```
-- deprecated function names
datafile_path = Dir.global_config_path
persconffile_path = Dir.personal_config_path


dofile(DATA_DIR.."console.lua")
dofile(DATA_DIR.."UserLog-new.lua")
--dofile(DATA_DIR.."dtd_gen.lua")
```

3、 正常打开wireshark，我们就可以看到正常的Userlog日志啦





userlog报文说明

serlog报文主要用为TCP/UDP session统计以及NAT会话统计，分为FLOW1.0和FLOW3.0，Flow1.0已经基本不用了，所以着重说说FLOW3.0。由于其是纯二进制格式的报文，所以看这个报文的时候，必须得放个大招才好使。所有的抓包，详细的TCP/UDP字段均为二进制或者十六进制格式来显示，为了方便起见，我们主要说十六进制格式的报文。

比如IP地址1.12.123.134可以用十六进制表示为：01 0C 7B 86。

其中各个字段在二进制流中的具体位置如下：

UDP首部：第0~8个B

————————————————payload部分————————————————

Version：防火墙版本，第1个B，防火墙V2R1版本为0x02，V2R5为0x03。

LogType：日志报文的类型, NAT=1 BAS(ACCESS)=2 FLOW=4。第2个B。

Second：防火墙发包时间，第4~8个B。

Operator：表示各种Session流类型，主要有流正常结束0X01、0X02流老0X08流创建。第18个B。

SourceIP：表示会话源IP地址，第20~24个B。

SrcNatIP：表示NAT转换后的源IP，第25~28个B。

DestIP：表示目的IP地址，第29~32个B。

SrcPort：表示NAT前源TCP/UDP端口号，第37~38个B。

SrcNatPort：表示NAT后源TCP/UDP端口号，第39~40个B。

DestPort：表示目的TCP/UDP端口号，第41~42个B。

StartTime：表示NAT Session创建的时间，第45~48个B。

EndTime：表示会话终止时间，第49~52个B。

PS：整个userlog部分长度为80字节，userlog首部长度为16字节，后续每64个字节则代表一条流的发起或者结束。

有了上述的解释，那么在筛选相关信息的时候就有了办法：

udp[m:n]==？表示UDP的第m个字节后n个字节的内容。

如果要查询整个userlog报文中是否包含1.12.123.134的内容，则筛选条件可以选择udp contains 01 0C 7B 86

如果要查询整个抓包中共有多少个跟源1.12.123.134相关的会话条目，则可以这样筛选udp[28+64*n 4] == 01 0C 7B 86 //28=UDP首部8+第20位置，n代表一个整数，MTU最大为1500，所以这个值最大为21，64前面有提到。

此处要注意wireshark安装目录不能有中文，否则会无法调用console.lua文件，造成无法查看userlog日志。