SNMP跨平台移植和交换机端口IP探测

吴 昊,杨 凯,胡 术,李娜娜
(四川大学 计算机学院,四川 成都 610064)

摘 要:在复杂的网络环境中,获取交换机端口IP对于网络监控和硬件维护是很有意义的。介绍了支持SNMPv3的Net-Snmp在Linux主机上的移植配置过程和SNMP++的跨平台移植方法。在此基础上,分析提出了一种基于SNMP和ARP的交换机端口IP通用探测方案,并给出了多平台的详细算法描述。该方案已经被应用到空管系统的网络中。经实践检验,该方案快速准确,具有令人满意的效果。

关键词:简单网络管理协议v3;跨平台;交换机;端口;探测
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2008)11-0018-04

Cross-Platform Transportation of SNMPv3 and Detect IP of Switch's Ports

WU Hao, YANG Kai, HU Shu, LI Na-na
(Computer College, Sichuan University, Chengdu 610064, China)

Abstract: In a complex network environment, to get switch ports' IP is meaningful for network monitoring and hardware maintenance. Introduces the process of transplanting and configuring Net-Snmp on Linux host, and the way about SNMP++'s cross-platform transportation step by step. Both of them support SNMPv3. A generic method based on SNMP and ARP is proposed to detect switch ports' IP, and its algorithm on multi platform is given in detail. It was applied in a network of ATC(Air Traffic Control) system, and has been fully proved through practice that the method's effect and efficiency is satisfied.

Key words: SNMPv3; cross-platform; switch; port; detect

0 引 言

SNMP(Simple Network Management Protocol,简单网络管理协议)是目前TCP/IP网络中应用最为广泛的网络管理协议,迄今有三个版本v1、v2C和v3。v1和v2C的安全机制比较简单;而在v3结构中引入了基于用户的安全模型用于保证消息安全,以及基于视图的访问控制模型用于访问控制(USM),提高了安全性。

作者在Linux主机上成功安装和配置了Net-Snmp5.4.1作为SNMPv3的代理端;借助安装Net-Snmp的主机,在Linux, SCO, Tru64, AIX, Solaris和Windows等多个平台上成功移植了SNMP管理站端的开发包SNMP++v3.2.23;并使用SNMP++实现了对交换机各端口上IP地址的发现,满足了监控的需要。

收稿日期:2008-02-28
基金项目:国家863计划资助项目(2006AA12A104)
作者简介:吴 昊(1981-),男,四川眉山人,硕士研究生,研究方向为实时软件工程。

1 Net-Snmp和SNMP++简介

Net-Snmp是著名的开放源码SNMP代理软件包,其前身是Ucd-Snmp。Ucd-Snmp源自于卡耐基·梅隆大学的SNMP软件包CMU Snmp2.1.2.1。2000年11月Ucd-Snmp项目转由著名开源网站www.sourceforge.net维护至今,并更名为Net-Snmp[1],当前最新版本是5.4.1。Net-Snmp包括了以下内容:

①MD5SHA1的认证支持(Authentication support)和DES AES的加密支持(Encryption support);

②一个功能强大、可扩展的SNMP代理;

③一套完整的C和Perl API用于支持IPv4和IPv6的SNMP应用程序开发;

④一个图形化的MIB浏览器;

⑤产生(generate)和处理(handle)Trap的工具。

SNMP++是一套为网络管理应用程序开发者提供SNMP服务的C++类集合,其早期版本(2.8以前)是由HP公司推出的开放源码SNMP开发包,支持v1、v2C。目前该软件由www.agentpp.com网站维护。

这里使用的是当前最新版 3.2.23,这个版本的 SNMP + + 具有以下特点:

①支持基于 IPv4 和 IPv6 的 SNMPv1,v2C 和 v3;

②提供对于 SNMP 简单易用的接口,并保证了传输的可靠性,使得开发者专注于上层业务[2];

③为实现对 v3 安全性能的支持,提供了三个加密库:libdes、OpenSSL 和 libtomcrypt。libdes 使用了 MD5 和 SHA1 算法。后两者使用了 SHA1, MD5, DES 和 AES。libdes 由 SNMP + + 默认使用,也是工作中所用;而若在 config_snmp_pp.h 定义_USE_OPENSSL 或_USE_LIBTOMCRYPT,则可以使用后两者。三个加密库都可以独立编译,以满足开发者的不同需求。

2　Net-Snmp 安装和配置

安装所用 Linux 主机是 Red Hat9,内核版本为 2.4.20-8,并自带 Net-Snmp5.1.2。安装 Net-Snmp 官方网站上下载的 5.4.1 版本,关键步骤如下:

①二进制方式将 net-snmp-5.4.1.tar.gz 上传到主机;

②命令:tar xzf net-snmp-5.4.1.tar.gz,解压安装包;

③进入/net-snmp-5.4.1 输入命令:./configure,完成安装前的配置。提示和输入依次为:Default version of SNMP to use(3):3(可分别输入:1:表示使用 v1;2:表示使用 v2C;3:表示使用 v3)

System Contact Information (xyz@):(此处输入系统子树中的联系信息,可以忽略,下同)

System Location (Unknown):

Location to write logfile (/var/log/snmpd.log):

Location to write persistent information (/var/net-snmp):

④输入命令:make,编译 Net-Snmp。这一步需要确认系统时间是否合理,否则不能正常编译;

⑤在超级用户下输入命令:make install,安装 SNMP 服务到系统;

⑥进入/usr/local/lib,输入命令:cp * /usr/lib,拷贝所有文件到/usr/lib 目录;

⑦输入 ntsysv 命令,选中 snmpd, snmptrapd 两项服务,配置为自动运行;

⑧重启后,确认"ethereal-网络交通分析器"、"ethereal-gnome"、"net-snmp-util"已经安装。

安装完成后,对 snmpd.conf 文件进行修改以配置主机,步骤如下[3]:

①定义一个 community,及可以访问这个 public 的 sec name:

```
# sec.name source community
com2sec notConfigUser 192.168.14.0/24 public
```

②定义一个 groupName,以及该组的 securityModel 和 securityName:

```
# groupName      sec.model   sec.name
group notConfigGroup v1    notConfigUser
group notConfigGroup v2c   notConfigUser
group notConfigGroup usm   notConfigUser
```

③定义一个可操作的 view, name 是 all, subtree 是.1:

```
# name incl/excl subtree mask(optional)
view all included      .1          80
```

④定义 notConfigGroup 在 all 这个 view 范围内可做的操作。这里定义 notConfigGroup 组成员可对.1 这个范围做只读操作(sec.level 可选项为:noauth、authNoPriv、authPriv;当 sec.level 配置为 noauth 时,普通用户需要 authentication 密码;sec.level 配置为 authNoPriv 时,普通用户至少需要 authentica-tion 密码;当 sec.level 配置为 authPriv 时,普通用户必须使用 authentication 密码和 privacy 密码):

```
# group context sec.model sec.level prefix read write notify
access notConfigGroup "" any noauth exact all none none
```

⑤创建用户 notConfiguser,密码 12345678(密码长度不低于 8,认证协议包括:MD5,SHA):

```
createUser notConfiguser MD5 "12345678" DES "87654321"
```

⑥赋予用户权限,可读写需要密码验证(需要通顺):

```
rwuser mtuser auth
```

⑦将/etc/snmp/snmpd.conf 拷贝到/usr/local/share/snmp/目录下,然后重新启动 snmpd 进程。

此外如果在/usr/local/share/snmp/snmpd.conf 中未添加以下两行设置,会阻止 v1 或 v2C 版本的请求,此时只有 v3 能够得到响应。如果添加,那么 v1 和 v2C 版本的请求同样能够得到响应:

```
rocommnity public
rwcommnity private
```

3　SNMP + + 的跨平台移植

3.1　Windows 上的编译

Windows 平台下需要将 SNMP + + 与 libdes 加密库一起编译。SNMP + + 需要在"预处理器定义"中定义三个宏:SNMP_PP_DLL, SNMP_PP_EXPORTS,

MSDOS,并在 config_snmp_pp.h 文件添加合适语句以生成 dll 文件。加密库中需要进行编译的源文件包括:cbc_enc.c,des_enc.c,ecb_enc.c,ede_enc.c,fcrypt.c,ncbc_enc.c,set_key.c。

3.2 Linux 上的编译

SNMP++为 Linux 平台提供了可以直接使用的 make 文件:Makefile.linux 和 common.mk。使用命令:make-f Makefile.linux,就能同时产生纯粹的 SNMP++静态库和动态库。而编译含有 libdes 加密库的 SNMP++库,则需要手动修改 make 文件:

①生成静态库时,编译选项为 -D_XPG4_EXTENDE D-D_unix -Wall-g,并需要增加加密库文件:cbc_enc.c,des_enc.c,ecb_enc.c,ede_enc.c,fcrypt.c,ncbc_enc.c,set_key.c;

②生成动态库时,编译选项再添加 -fPIC -shared,并且将提前生成的加密库添加到命令中。使用动态库时,需要系统库-lm -ldl -lresolv -lcurses -lnsl-pthread。

此外,如果在程序中包含了头文件<curses.h>,由于该头文件中的 # define timeout(delay) wtimeout(stdscr,delay)会与 SNMP++库的使用产生冲突,故<curses.h>头文件的包含,必须在 snmp_pp.h 之后。

3.3 Tru64 上的编译

在 Tru64 平台上,编译选项中不能使用 -Wall;添加-DPERL5 宏定义,如果未添加,crypt 函数会与系统函数冲突;添加-Dalpha,该宏定义仅用于 Tru64,在 Tru64 下 sizeof(long)等于 8,而在其他平台下的 sizeof(long)等于 4,此宏定义就是为了防止歧义的发生。最后的编译选项为 -g -pt -pthread -D_unix -D_XPG4_EXTENDED -DPERL5 -Dalpha-D_REENTRANT,编译动态库还需要添加-KPIC-shared。同时,需要系统库-lpthreads -lnsl -ldl-lc 和编译出的加密库。

3.4 SCO UNIXWARE 上的编译

在 SCO 平台上,编译选项为 -g -pt -Kthread -D_XPG4_EXTENDED -D_unix -DPERL5DSCOUW,编译动态库还需要有 -KPIC-G。同时,需要系统库-lnsl -lsocket -lm -ldl -lgen -lresolv -lw -Kthread-lcurses 和编译出的加密库。

为了编译的通过,还需要对一些源文件进行修改:

①/snmp++/include/snmp_pp/address.h 第 96 行,添加! defined SCOUW 的判断,因为之后声明了一个在 Windows 系统中使用的变量 int h_errno;

②/snmp++/include/snmp_pp/config_snmp_pp.h 第 60 行,添加 # ifndef SCOUW,以取消 SNMP_PP_IPv6 的宏定义;第 162,174,186 行添加 # ifndef SCOUW,以去掉 warning;

③/snmp++/src/uxsnmp.cpp 第 302 和 428 行添加 defined SCOUW,以定义 socklen_t fromlen。

3.5 AIX 上的编译

在 AIX 平台上,编译选项为 -D_XPG4_EXTENDED -D_unix -D_AIX-D_REENTRANT;编译动态库还需要添加 -G。同时,需要系统库-lpthreads -lnsl -ldl-lc 和编译出来的加密库。

3.6 Solaris 上的编译

在 Solaris 平台上,编译选项为 -DPERL5 -Wall -D_unix -DSunOS -D_XPG4_EXTENDED;编译动态库还需要添加-dy -G。同时,需要系统库-ldes -lnsl -lintl -lsocket -lm -ldl -lgen -lresolv -lw -lcurses -lpthread -lc-lrt 和编译出来的加密库。

为了编译的通过,还需要对源文件进行修改:文件./snmp++/include/snmp_pp/config_snmp_pp.h 第 60 行,添加 # ifndef SunOS,以取消宏定义 SNMP_PP_IPv6。

3.7 各平台上的回归测试

每一个平台的移植完成后,都分别使用编译出的 SNMP++动态库和静态库,对安装 Net-Snmp 的 Linux 主机进行 v3 上的回归测试,包括 get_bulk 和 walk。通过测试,确认了上述的移植方法是可行的。

4 对交换机端口 IP 地址的探测

交换机产品的不同,造成了没有现成方法来确认交换机每个端口的 IP,而获得这一信息对于网络的监控、管理和维护都有很现实的意义。

这里提出一种方法,可以快速获得这一信息。通常,在多网段且有交换机级联的环境中,查找具体一台交换机上各端口的 IP 地址,Windows 版本程序只需要 2 到 3 秒时间。该方法的关键思想是:使用 SNMP 探测交换机,获取其转发地址表,找到每个端口对应的 MAC 地址;再读取主机的 ARP 高速缓存[4],找到每个在线 MAC 地址对应的 IP 地址;然后进行配对。这一方法具有较好的通用性,只要交换机提供转发地址表的 SNMP 实现即可[5]。

具体方法由于 Windows 和类 Unix 系统有较大不同,分为以下两种。

4.1 Windows 操作系统

生成一个 UDP Socket,向所要探测的网段的所有 IP 依次发送一个数据报(点对点),这样就在本地主机的 ARP 高速缓存中保存了当前所有在线主机的 MAC 地址[6]。

使用 IP Helper API 中的 GetIpNetTable 函数获取 ARP 高速缓存的内容。该函数有三个参数:pIpNetTable 为指针,指向存储取到内容的 Buffer;pdwSize 是 pIpNetTable 指向 Buffer 的大小;bOrder 指示取得的结果是否按 IP 升序排列,TRUE 表示升序。使用时需要注意的是:如果函数获取失败,返回错误是 ERROR_INSUFFICIENT_BUFFER,则 pdwSize 的值会被函数置为实际所需大小。此时需要将函数再执行一遍。关键代码如下:

```
dwRet = GetIpNetTable((PMIB_IPNETTABLE) &tempChar,
&dwListSize, TRUE);
if (dwRet = = ERROR_INSUFFICIENT_BUFFER)
{
  PMIB_IPNETTABLE pIpNetTable =
  (PMIB_IPNETTABLE)new(char[dwListSize]);
  dwRet = GetIpNetTable(pIpNetTable, &dwListSize, TRUE);
  process content in pIpNetTable;
  delete pIpNetTable;
}
```

而获取交换机端口上的 MAC 地址,则需要针对不同产品的不同 oid,使用 SNMP + + 编程探测即可。作者所用交换机为 3COM4400,获得 MAC 地址需要对不同的 3 组 oid 依次进行探测,总共需要探测 3 次。每次探测的过程一样,伪代码如下:

```
{
  UTarget * utarget and set * utarget;
  Vb vb and set vb;
  base_oid = target_oid;
  next_oid = target_oid;
  Pdu pdu and set pdu;
  Set SNMP and start SNMP;
/* 比较 base_oid 和 next_oid 前 base_oid 长度个数
的字符 */
  While(0 = = nCompare(base_oid.len(),next_oid))
  {
    get_next(pdu, * utarget);
    if(SUCCESS)
    {
      vb.get_oid(next_oid);
      vb.get_printable_oid();
      vb.get_printable_value();
    }
    else break;
    vb.set_oid(next_oid);
    vb.set_null();
    pdu.set_vb(vb,0);
  }
}
```

4.2　类 UNIX 操作系统

在类 UNIX 操作系统下,对交换机的探测方法与 Windows 下的基本一致。获得本地主机 ARP 高速缓存的伪代码如下:

```
Create SOCK_DGRAM FD;
While(! IP 遍历完)
{
  FD.SendTO( IP );
  {
    struct arpreq req and to set rep;
    struct sockaddr_in * sin;
    sin = (struct sockaddr_in * )&(arpreq.arp_pa);
    sin - >sin_family = AF_INET;
    struct in_addr address(IP);
    memcpy(&sin - >sin_addr,&addr,sizeof(struct in_addr));
    Create SOCK_DGRAM sockfd;
    ioctl(sockfd, SIOCGARP, &req);
    保存 req 中得到的 MAC 地址;
    IP + + ;
  }
}
```

4.3　实验结果及分析

在多网段并带有级联交换机的网络环境中进行了测试。测试时,输入参数为被测交换机的 IP。实验结果见图 1。可以看出,在端口 12 和 24 上有多个 IP,这说明这些主机并非直接连接在被测交换机上,而是通过交换机级联,间接连接被探测交换机,因此它们对应了相同的端口。

对应端口	IP地址	MAC地址
24	168.192.11.65	0002b3c1d021
16	168.192.11.34	0002b3ef3e6f
24	168.192.11.72	0002b3ef8b45
21	168.192.11.40	0002b3ef8b62
20	168.192.11.38	0002b3ef8b64
19	168.192.11.36	0002b3ef8cff
24	168.192.11.52	00096b2ec417
24	168.192.11.51	00096b2ec477
10	168.192.11.41	000e0caa82e1
24	168.192.11.71	000e0caa82f3
11	168.192.11.42	000e0caa8311
24	168.192.11.63	001125c52aa2
24	168.192.11.64	001125c5308e
12	168.192.11.62	001125c531b2
24	168.192.11.61	001125c5330e
7	168.192.11.35	001125c59812
6	168.192.11.33	001125c59c08
8	168.192.11.37	001125c59c8c
9	168.192.11.39	001125c59da8
24	168.192.11.253	0016e04a5a00
12	168.192.11.251	0016e0752ac0
24	168.192.11.252	0016e0754b00
24	168.192.11.66	0050ba408830
24	168.192.11.81	0090e80a2354
12	168.192.11.82	0090e80cbe6d

图 1　在级联交换机上的探测结果

5　结束语

在网络中,保证连接的正确性,快速发现错误和丢

(下转第 25 页)

```
        s2.enter();
        }
    return 0;
ErrorExit:
        printf("Error,ec= %x\n", ec);
        return 1;
}
```

以下是运行的结果:

```
CAccountManager enter 1 times.
CAccountManager enter 2 times.
CAccountManager enter 3 times.
CAccountManager enter 4 times.
CAccountManager enter 5 times.
CAccountManager enter 6 times.
CAccountManager dtor.
```

可以看到,虽然调用的是不同的对象的 enter 方法,但是最终调用的都是同一方法,而且最后调用该类析构函数时只调用了一次。

比较一下当没有为 AccountManager 对象指定 singleton 属性时的运行结果:

```
CAccountManager enter 1 times.
CAccountManager enter 1 times.
CAccountManager enter 1 times.
CAccountManager enter 2 times.
CAccountManager enter 1 times.
CAccountManager enter 2 times.
CAccountManager enter 2 times.
CAccountManager enter 3 times.
CAccountManager enter 2 times.
CAccountManager enter 3 times.
CAccountManager enter 3 times.
CAccountManager dtor.
CAccountManager enter 3 times.
CAccountManager dtor.
```

可以看到,没有指定 Singleton 属性时是分别调用了两个不同对象实例的 enter 方法,并且最后析构时调用了两次。这个结果是符合预期的。

同时也可以看到,在具体的 CAccountManager 类的实现中,并没有包含任何 Singleton 模式的管理代码,用户只需要指定 CAccountManager 类为 Singleton 属性并重新编译后,相同的实现代码立即有了不同的运行结果。用户通过这种方式,可以很方便地用 Singleton 模式去解决其他工程问题。

5　结束语

文中基于 CAR 构件编程,设计了一种 Singleton 模式,通过编译 car 文件后就可以自动生成类的 Singleton 实现。用户不必再关心如何去实现 Singleton 模式本身,只需要把精力放在如何实现该类的其他基本功能上即可。这一设计不仅为在 CAR 构件技术中应用 Singleton 模式自动生成代码提出了方法和手段,同时也降低了用户编程的复杂程度,更利于编程。

参考文献:

[1] 陈　榕,刘艺平.基于构件、中间件的因特网操作系统及跨操作系统的构件、中间件运行平台[R].北京:[出版者不详],2003.

[2] Rogerson D. COM 技术内幕(Inside COM)[M].杨秀章译.北京:清华大学出版社,1999.

[3] Box, Don. Essential COM[M]. Menlo Park, CA: Addison-Wesley Publishing Company, 1998.

[4] 上海科泰世纪有限公司. Elastos CAR 构件与编程技术[EB/OL].2006-09. http://www.kortide.com.cn/.

[5] 上海科泰世纪有限公司.和欣资料大全[EB/OL].2006-09. http://www.kortide.com.cn/

(上接第 21 页)

失的连接,对于管理和维护网络都是非常必要的。由于交换机产品种类众多,具体功能各异,往往缺乏统一便捷的手段来进行这方面的工作。文中提出了一种基于 SNMP 和 ARP 的交换机端口 IP 探测方案,经过在多个平台上的使用验证,能快速准确地获取结果,并具有较强的通用性。方案已经在空中交通管制系统中使用,使得网络监控管理和维护的工作更为简便,取得了令人满意的效果。

参考文献:

[1] 李明江. SNMP 简单网络管理协议[M].北京:电子工业出版社,2007.

[2] 周奕利,杨红雨,覃树建,在 Tru64Unix 上使用 SNMP++获取交换机的流量信息[J].中国民航飞行学院学报,2006,17(2):55-57.

[3] 王　芬,赵梗明. 基于 SNMPv3 网络管理系统的研究和应用[J].计算机技术与发展,2006,16(4):199-202.

[4] Wright G R, Stevens W R. TCP/IP Illustrated, Volume 2: The Implementation[M]. USA: Addison Wesley, 1995.

[5] 陈　然,杜晓黎. 基于统一接口的机群中交换机监控系统的设计[J].计算机工程,2005,31(16):225-227.

[6] Stevens W R, Fenner B, Rudoff A M. Unix 网络编程(第一卷)[M].第 3 版.杨继张,译.北京:清华大学出版社,2006.