SecureGuard-Solaris 环境下 VPN 的设计与实现*)

SecureGuard-The Design and Implementation of A VPN Software based on Solaris

李方敏[1,2] 叶澄清[2] 蒋晓宁[2]

(湘潭工学院计算机科学系 湘潭411201)[1](浙江大学计算机科学系 杭州310027)[2]

Abstract VPN can be implemnted based on hardware or software, but these products are very expensive produced by some famous foreign country. Sun workstations are the main platform in China. So, developing security software based on Solaris is profound. First, according to the stream mechanisms, the architecture and key technology of implementing SecureGuard are analyzed. Then, how to use SecureGuard to build applications is presented.

Keywords VPN(virtual private network), Solaris, Stream mechanism, Network security

1 引言

VPN(Virtual Private Network)具体实现是采用隧道技术,将企业网的数据封装在隧道中进行传输。因此,VPN 技术的复杂性首先建立在隧道协议复杂性的基础之上。隧道协议中最为典型的有 CRE、IPSec、L2TP、PPTP、L2F 等。其中 GRE、IPSEC 属于第三层隧道协议,L2TP、PPTP、L2F 属于第二层隧道协议。第二层隧道和第三层隧道的本质区别在于用户的 IP 数据包是被封装在何种数据包中在隧道中传输的。

VPN 产品可以基于硬件和软件方式实现,如 Cisco、3Com、Nortel 等国际著名网络公司都有基于硬件实现的产品,基于软件的产品,如 Sun 公司的 SunScreen Secure Net,它们的价格昂贵,并且将整个网络的安全托付给国外的公司不能令人放心。

因此,浙江大学计算机系系统工程研究所联合公司正在开发基于 NT、Solaris、Linux 环境下的 VPN 产品。本文主要论述我们在 Solaris 环境下开发的 VPN 产品—SecureGuard 的总体结构,以及基于 Solaris 的流机制[1]实现安全模块嵌入的关键技术,关于安全协议标准请参见 IETF 的 RFC[2]和文[3]。

2 SecureGuard 的总体结构

在 Solaris 环境下实现的 SecureGuard 从功能总体结构来讲,类似我们已经实现的 NetGuard[3],但 NetGuard 是基于 Windos NT 平台。

SecureGuard 主要由安全引擎模块,驱动程序,安全策略 SP 和安全协定 SA 管理模块(含 SPDB 和 SADB),算法模块,SP 配置模块和密钥管理模块组成。如图1所示。

·安全引擎模块:该模块作为一个伪设备驱动程序实现,其作用是截断 IP 层与数据链路层(Data Link Layer),也称网络接口层(Network Interface)的直接联系,并完成安全协议模块的功能。修改相关数据结构,抓获由 IP 层下传的 IP 包,经过安全协议模块处理后再传给网络接口层。抓获由网络接口层上传的 IP 包经过安全协议模块处理后再交给 IP 层处理。并解决分段与 PMTU 之类的网络层问题。

·安全策略 SP 和安全协议 SA 管理模块:SP 管理模块,决定对数据包进行保护的安全策略(包括访问控制),这个模块管理着 SPDB 数据库。外出包和进入包的处理都要查阅这个 SPDB。对外出包而言,查询 SPDB,决定这个包需要什么样的安全保护。对进入包而言,查询 SPDB,检验为这个包提供的安全保护是否和策略配置的安全保护相符。安全协定 SA 管理模块,决定对数据包进行保护的安全协定(包括相应的密钥材料)。它维护着一个活动的 SADB。所谓活动,是指 SADB 中的条目可能被自动更新,比如,某一条目的生存期已到,SA 管理模块将申请 IKE 密钥管理模块与对方协商,重建 SA。外出 SA 用来保障外出包的安全,进入 SA 用来处理带有安全头的进入包。

·算法模块:实现各种算法,如国际上常用的 DES-CBC,IDEA-CBC,HMAC-MD5-96,HMAC-SHA-96等等。也可以是用户自己设计的算法。这部份可按需要用硬件或软件实现。

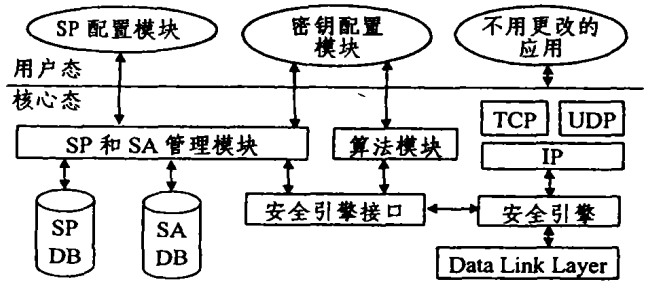

图1 SecureGuard 的总体结构

·SP 配置模块:这是一个用户进程,提供用户对安全策略的手工配置接口。由于手工方式配置的可扩展性差,对大型网络而言,该模块需提供自动策略配置功能。

·密钥配置模块:它也是一个用户进程。利用 IKE 提供自动密钥分配和自动密钥更新功能。同时也提供用户对长期预共享密钥和公钥证书的手工配置接口。当然,用于 SA 自动配置的网络流不能被安全层保护,否则,整个系统将因面临"鸡生蛋还是蛋生鸡"的问题,而不能工作。为此,SPDB 中应配置相应条目进行说明:对于 IKE 流,绕过。也由于 IKE 流不受保护,要求由 IKE 自己提供加密、鉴别等安全功能和抵抗拒绝服务攻击。我们知道,IKE 的设计已经这么做了。

*)基金项目:国家自然科学基金资助(No.69974031)。李方敏 博士研究生,副教授,研究方向为网络服务质量、新型网络协议、网络安全。叶澄清 教授,博导,研究领域为高性能体系结构、计算机网络、多媒体应用等。蒋晓宁 博士,研究方向为网络安全。

3 安全引擎实现的关键技术

Sun 工作站配置的操作系统从 Sun OS 5.0开始改称为 Solaris 2.0，如我们使用的 Ultra-10配置的 Sun OS 5.7，就是 Solaris 7，从 Sun OS 5.0开始，其 TCP/IP 协议是基于 System V 的流机制实现的，即 IP、TCP、UDP 均是一个流模块。

实现安全引擎模块的关键技术是如何将其嵌入到 IP 层和链路层之间，分析流机制，似乎可以将安全引擎作为一个流模块实现，然后用 ioctl 的 I_PUSH 将其插入。但每次插入模块时，只能将模块插入到流头下，虽然，可以首先打开网卡驱动设备，然后 PUSH 安全引擎模块，但安全引擎模块如何和 IP 流模块连接是一个问题。因此，我们通过分析 Darren Reed 设计的 IP Filter[4]的源码，进一步熟悉了 Solaris 的内部数据结构，决定将安全引擎模块作为一个伪设备驱动程序实现，方便与用户态的通信与动态配置。然后修改相关数据结构，将安全引擎插入到 IP 和 Data Link 之间。

3.1 重要数据结构

在/usr/include/inet/ip.h 头文件中定义了一个全局变量如下：extern struct ill_s * ill_g_head；该指针指向一个全局链表，每个链表项是 ill_s 类型，该类型也定义在 ip.h 中，它是一个流模块的私有数据，在下层的驱动设备打开时实例化，并对应一条消息通路，它有读写队列，通过修改它的读、写队列的 q_qinfo 指针来修改消息的处理例程，即让消息经过安全引擎模块的处理。

为了在安全引擎模块中处理消息，必须说明一个数据结构对应 ill_s，同时，也建立一个链表，与 ill_g_head 指向的链表一一对应。这样的一个结构就是 qif，定义如下：

```
typedef struct qif {
    struct qif  * qf_next; / * 指向链表的下一项 * /
    ill_t  * qf_ill; / * 指向对应的 ill_s 结构 * /
    kmutex_t qf_lock;
    void  * qf_iptr;
    void  * qf_optr;
    queue_t  * qf_in; / * 下行队列 * /
    queue_t  * qf_out; / * 上行队列 * /
    struct qinit * qf_wqinfo; / * 保存 ill_s 中写队列的 qinfo 指针 * /
    struct qinit * qf_rqinfo; / * 保存 ill_s 中读队列的 qinfo 指针 * /
    struct qinit qf_wqinit; / * 安全引擎模块的写队列的 qinit 结构
     * /
    struct qinit qf_rqinit; / * 安全引擎模块的读队列的 qinit 结构 * /
    mblk_t  * qf_m; / *  These three fields are for passing data up
    from * /
    queue_t  * qf_q; / * fr_qin and fr_qout to the packet processing.
     * /
    int qf_off;
    int qf_len; / *  this field is used for in ipfr_fastroute  * /
    char qf_name;
    int qf_hl; / * header length  * /
}qif_t;
```

3.2 安全引擎模块的插入

我们将安全引擎模块实现为一个伪设备驱动程序，并修改相应的启动脚本，在系统启动时可以自动加载，然后通过管理进程进行动态配置和监控。为了方便讨论，我们给出安全引擎模块的说明：

```
…… / * 头文件及有关定义 * /
static struct cb_ops ipf_cb_ops={
      iplopen,
      iplclose,
      nodev, / * strategy * /
      nodev, / * print * /
      nodev, / * dump * /
      iplread,
      nodev, / * write * /
      iplioctl, / * ioctl * /
      nodev, / * devmap * /
      nodev, / * mmap * /
      nodev, / * segmap * /
      nochpoll, / * poll * /
      ddi_prop_op,
      NULL,
      D_MTSAFE,
#if SOLARIS2>=4
      CB_REV,
      nodev, / * aread * /
      nodev, / * awrite * /
#endif
};
static struct dev_ops ipf_ops={
      DEVO_REV,
      0,
      ipf_getinfo,
      ipf_identify,
      ipf_probe,
      ipf_attach,
      ipf_detach,
      nodev, / * reset * /
      &ipf_cb_ops,
      (struct bus_ops * )0
};
extern struct mod_ops mod_driverops;
static struct modldrv iplmod={
      &mod_driverops,IPL_VERSION,&ipf_ops};
static struct modlinkage modlink1 = {MODREV_1, &iplmod,
NULL};
static dev_info_t * ipf_dev_info=NULL;
int_init()
{
#ifdef IPFDEBUG
      int ipfinst=mod_install(&modlink1);
      cmn_err(CE_NOTE,"IP Filter:_init()=%d\n",ipfinst);
      return ipfinst;
#else
     return mod_install(&modlink1);
#endif
}
int_fini(void)
{
     int ipfinst;
     ipfinst=mod_remove(&modlink1);
#ifdef IPFDEBUG
      cmn_err(CE_NOTE,"IP Filter:_Fini()=%d\n",ipfinst);
#endif
      return ipfinst;
}
int_info(modinfop)
struct modinfo  * modinfop;
{
#ifdef IPFDEBUG
      int ipfinst=mod_info(&modlink1,modinfop);
      cmn_err(CE_NOTE,"IP Filter:_info(%x)=%x\n",modin-
      fop,ipfinst);
      if(fr_running)
         ipfsync();
      return ipfinst;
#else
     if(fr_running)
        ipfsync();
     return mod_info(&modlink1,modinfop);
#endif
}
……/ * 函数实现 * /
```

这段代码来自 IP Filter，它将 IP 过滤器作为一个伪设备驱动程序实现，而我们在实现 SecureGuard 时，也将安全引擎模块实现为一个伪设备驱动程序。cb_ops 是所有设备驱动程序所必需的，在非流驱动程序中，cb_ops 包含指向表驱动入口点的向量；而对流驱动程序，cb_ops 包含大部分 nodev 的入口点。另外，声明的 dev_ops 结构除指向 cb_ops 结构外，还指向各个初始化入口点。最后，声明的 struct modldrv 和 struct modlinkage 结构用来在驱动程序动态装入时供内核连接程序使用。

在加载设备驱动程度时，主要的初始化工作由 ipfattach 完成，如登记中断、创建设备节点等，因为我们实现的是一个伪设备驱动程序，没有用到中断，只是创建设备节点和初始化一些信号量，其中完成的一个最重要的工作是建立 qif 链表与 ill_s 链表对应，下面，我们专门以一节介绍在 ipfattach 中是如何完成 qif 的建立及链接的。

3.3 qif 链表的建立及链接

全局变量 ill_g_head 指向链表的每个 ill_s 项与低层的 device 一一对应，有多少个同时打开的网络设备(每个设备可以打开多次)，则有多少个 ill_s 项，也就是说有多少个流。在启动加载安全引擎模块时，调用 dev_ops 的 attach 函数完成初始化工作，在我们的实现中主要完成 qif 链表和 ill_s 链表的钩链，任一个 qif 链表项将 ill_s 对应项的模块接入点指针 q_qinfo(实际上队列定义的相关操作入口点)保存下来，然后再让它指向新的处理程序，fr_qin()和 fr_qout()，由这两个处理程序完成各种安全处理。

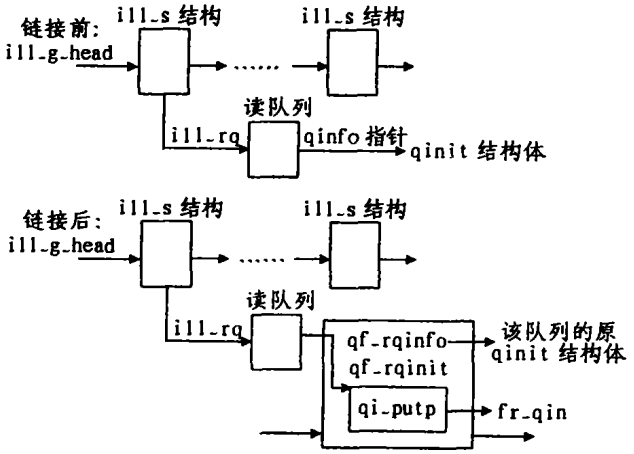

图2 ill_s 和 qif 链表的链接示意图

从图2我们看到，在打开的网络设备上加载安全引擎后，实际上将在 IP 模块中建立的 ill_s 链表与在安全引擎模块中建立的 qif 链表进行了钩链，在 qif 结构中的 qf_rqinfo 中保存 ill_s 结构中定义的读队列 ill_rq 的队列操作入口点指针 q_qinfo，然后，让其指向在 qif 中定义的新的读队列 qinit 结构 qf_rqinit，而所有安全引擎的相关操作入口都通过 qf_rqinit 进行。

对写队列也进行上述操作，只不过是修改 ill_s 结构对应的下邻接队列。

4 SecureGuard 的特点和典型应用

SecureGuard 与我们在 NT 环境实现的 NetGuard[3]功能相同，从应用的角度来看，归纳起来，系统具有如下特点：

·它是在系统内核实现网络安全，与应用程序无关，对用户操作透明。这样，不论是原来的程序还是新的程序均可照常使用，不用任何改动，程序的操作也和以前一样。但具备了对所有应用程序在私网范围内安全，不必对各个应用程序再采取特殊的安全措施。

·它与具体的通讯设备无关。凡是 Solaris 系统支持的网卡，调制解调器等物理接入设备都可使用。

·灵活性。可根据用户要求配置安全策略：譬如根据不同地址，不同服务和不同用户设置准入控制；选择加解密算法，选择保密范围，定制保密级别等。NetGuard 系统框架结构设计灵活，功能容易扩充。

·性能/价格比高。它是纯软件系统实现，比目前主要采用过滤方法的防火墙系统有更优越的性价比。

(下转第19页)

(上接第123页)

以得到一个广义的极大似然估计。在一般的数据缺失模式下，BC 估计是否就是期望的贝叶斯估计取决于所确定的无响应 ϕ 样式。

界定过程中，所有可能的估计边界表示由于缺失数据给估计过程带来的不确定性，而界定区间宽度可以度量估计时不完整样本所提供的信息的质量。通过计算边界可以将缺失数据带来的不确定性保留在分析过程中，这样由于确实缺失数据导致的抽样过程的变化和不确定性可以被独立地进行计算并表示出来。BC 方法的另一个优势就是其计算开销，对每一种条件分布，BC 方法在界定过程和塌陷过程中的凸联合针对响应变量的每一个取值进行逐步更新，减少了分析过程中的计算的复杂性。因此 BC 方法的计算复杂程度只是响应变量 Y 的可能取值个数的函数，而和缺失数据的数量无关。

结论 Gibbs 抽样、基于转移的方法以及高斯近似方法都有三个主要的缺点：

·都假设数据缺失模式是能够知道的，但实际情况并不总能够满足这一要求。

·由于没有完全考虑缺失数据，提供的估计结果可靠性程度的度量方法使抽样中的不确定性和无响应带来的外生不确定性混合在一起。

·这两种计算方法的计算代价都是缺失数据数量的函数。由于 Gibbs 抽样将缺失数据看作未知参数，因此该方法的收敛速度随着缺失数据的增多而下降。基于转移的方法的精度随着模拟产生的样本数量而增加，但每一次模拟的计算开销随着缺失数据的数量增加而增加。

与建立了数据缺失模式模型的现有的一些方法相比，BC 方法在塌陷阶段利用了一个无响应样式，该样式是数据缺失模式的函数，并且随后的推断都是基于这种无响应样式。由于 BC 方法计算的简单性，不同的无响应样式都可以被表示出来，从而各种不同数据缺失模式对推断结果的影响的敏感程度可以很快地计算出来。因此，BC 方法提供了一个通用的对不完整样本进行分析的框架，Kadane[1993]以及 Kadane 和 Terrin[1997]采用该方法所做的几个实际案例都证明了这一点，并且可以对不同无响应样式分别计算并求其平均来获得边际推断。

参 考 文 献

1 Bayesian H D. Networks for data mining[J]. Data Mining and Knowledge Discovery, 1997, 1: 79～119

2 Geiger D, Heckerman D. A characterization of the Dirichlet distribution with application to learning Bayesian networks[A]. In: Proc. of Eleventh Conf. on Uncertainty in Artificial Intelligency[C], Montreal, QU, 1995. 196～207

3 Dagum P, Luby M. Approximating probabilistic inference in Bayesian belief networks is NP-hard[J]. Artificial Intelligence, 1993, 60: 141～153

4 Paola S, Marco R. Baysian inference with missing data using bound and collapse. [KMI-TR-58], The open university research report, 1997

5 Gilks W R, Roberts G O. Strategies for improving MCMC. In: Gilks W R, Richardson S, Spiegelhalter D J, eds. Markov Chain Monte Carlo in Practice, pp. 89～114, Chapman and Hall, London

6 林士敏，田凤占，陆玉昌. 贝叶斯学习、贝叶斯网络与数据挖掘. 计算机科学，2000，27(10)

无线带宽状态 WB(Wireless Bandwidth):反映最近一段时间的无线通信(Wireless traffic)的拥挤程度。

同步服务器的计算负荷 CL(Computing Load):反映当前同步服务器上的计算资源的使用情况。

决策变量如下:

FUF(Fetch&Update Frequency)数据项的采集更新频度:该变量决定对所有数据项的采集更新频度,为各数据项采集更新频率的集合。

UTG(Update Transfer Granulary)数据变化部分的更新粒度:该分量决定将数据的变化控制在何种数据项类型中。

限制函数有:

SystemPerformance = Wb * WBPerformance + Wc * CPPerformance

WBPerformance=WBP(FUF,UTG,WB,CL)

WBP(FUF1,UTG)<WBP(FUF2,UTG)
当 FUF1>FUF2
WBP(FUF, UTG1)<WBP(FUF,UTG)
当 UTG1>UTG2

CPPerformance=CPP(FUF,UTG,WB,CL)

CPP(FUF ,UTG1)>CPP(FUF,UTG2)
当 UTG1>UTG2
CPP(FUF1,UTG)>CPP(FUF2,UTG)
当 FUF1>FUF2

AppiPerformance=AP(FUF,UTG)

AP(FUF1,UTG)>AP(FUF2,UTG)
当 FUF1>FUF2
AP(FUF, UTG1)>AP(FUF, UTG2)
当 UTG1>UTG2

FUF=FUF(DUF)

FUF(DUF1)>FUF(DUF2)
当 DUF1>DUF2

决策模块的功能是在当前的环境变量下,选择适当的更新频度和更新粒度,使得整体的系统性能最佳。模型中,简单地描述了各个函数随其变量变化的单调关系。在具体实现中,目前我们将各约束函数弱化为线性函数,并通过实验确定线性方程的参数。若想进一步获得更准确的决策,则还可以借助人工智能中关于决策系统的一些研究的成果。

但是,必须指出,决策模块的复杂度越高,系统的开销就越大。在我们的实现中,基本采用线性函数进行决策。

2.3 性能分析

和原型比较,性能的改进主要表现在:

第一,由于对移动端数据子集中的不同部分采用不同的更新频度,一方面减少了对非热点数据不必要的更新次数,从而减少了无线传输;另一方面使得热点数据的更新更加及时,提高了移动端应用的性能。

第二,由于采用多粒度对移动端的数据项进行更新,在系统计算资源允许的情况下,通过较小的更新粒度,显著地减少了无线数据传输量。

当然所有这些性能的改进是以系统复杂性的增加以及同步服务器计算的增加为代价的,但由于在移动计算环境中,无线通信带宽异常宝贵,通过增加固定网络上的计算和通信换取无线通信效率的提高在一定的范围中是可以接受的。

同时由于我们的更新机制能够根据实际的系统运行信息和移动端数据的使用,对各项更新参数进行自适应的调整,因此整个系统可以较好地适应不断变化的状况。

结论和展望 本文通过引入更新频度、更新粒度的动态调节以及系统的自适应能力,使得对移动端数据子集的更新占用的无线带宽分配更加合理,整个系统期望得到的性能更加良好。但在实际应用中,同步服务器由于需要处理来自各个移动设备的大量的请求,有可能会形成整个系统的瓶颈,可在我们的改进了的二层更新机制上进一步将一些集中在同步服务器上的操作,如数据变化的定位和更新信息的组织分散到相应的源数据库,采用分布式的数据收集,降低对同步服务器的负担。比如可以运用渐以成熟的移动 Agent[6]技术,通过派遣数据变化采集 Agent 至源数据库,采集并发回变化信息。

参 考 文 献

1 Barbará D. Mobile Computing and Databases-A Survey. IEEE Transactions on Knowledge and Data Engineering, 1999,11(1): 108~117
2 Pissinou N, et al. A Middleware-Based Architecture to Support Transparent Data Access by Mobile Users in Heterogeneous Environments. A research report in Center for Advanced Computer Studies & Center for Telecommunication Studies, University of Louisiana at Lafayette, 1998
3 Sandifer A, von Halle B. Business Rules: Capturing the Most Elusive. in High-Performance Web Databases Design, Development, and Deployment, Edited by Sanjiv Purba, 2001
4 Dunham M H, Helal A. Mobile Computing and Databases: anything new. SIGMOD Record, 1995, 24(4): 5~9
5 Noble B. System Support for Mobile, Adaptive Applications. IEEE Personal Computing Systems, 2000, 7(1): 44~49
6 Grabner M, et al. Agent technology: State of the Art: [Technical Report in Software Competence Center Hagenberg]. 2000

(上接第50页)

未来计算机应用是网络应用时代,特别是 Internet 这个全球范围内的公共通信通道,更是大家争相充分利用的宝贵资源。但既要借用此公共通信通道的方便性和经济性,又能保护公司私有的信息资源和商业秘密,存在相互矛盾的难点。解决这一问题是设计 SecureGuard 的初衷。这个强有力的工具,具有巨大的市场应用潜力,其中三个具体应用简列于下:

·支持远程用户通过互联网访问内部网。具备访问公司内部私网(VPN)能力的机器,即安装了 SecureGuard 的机器,可以在任何地方接入互联网,可保密安全地访问公司的内部网络,安全程度不亚于在公司内部接入的方式。这给奔走于全球各地的公司移动员工(经理、董事长)提供了特别方便安全地与公司通信的解决方案。

·通过互联网连接公司内部的网络。如果有分布于各地的不同分支机构,每个机构都有自己的局域网,只要各分支机构接入互联网的计算机安装了 SecureGuard,具有了 VPN 能力,则每个机构就可以通过 VPN 保密、安全地在互联网上通信。

·内部网的机器之间互连。利用 SeucreGuard(VPN)还可以使物理连接在同一网络上的不同小组内部进行安全保密的通信。例如机构的不同部门之间互相保密,没有 VPN 能力的计算机即使连到网络上,也无法监听网络的数据传输。

参 考 文 献

1 计算机技术开发人员宝典丛书编委会编. Sun Solaris 7流程序设计指南. 北京希望电子出版社, 2000. 6
2 IETF home page. http://www.ietf.org
3 蒋晓宁. 安全 IP 及公平交易协议的研究:[浙江大学博士学位论文]. 2000. 5
4 IP Filter home page. http://coombs.anu.edu.au/~avalon/