高速网络环境下内容审计系统的研究与实现*)

万国根　秦志光　刘锦德
（电子科技大学计算机科学与工程学院　成都610064）

The Study and Implement of Content Audit System on High Speed Network

WAN Guo-Gen　QIN Zhi-Guang　LIU Jin-De
(College of Computer Science and Engineering, UEST of China, Chengdu 610054)

Abstract The paper presents a content audit system on high speed network. A new algorithm is provided for load balancing and data distribution which is based on application protocol and session states. A fast text searching approach is also introduced for string matching which includes chars and chinese characters.

Keywords Content audit, Load balancing, Protocol analysis, String matching

网络内容审计就是实时检测网络上正在发生的事情，记录并分析网络上相关数据包，以发现可疑的内容和目标，并对这些内容和目标进行记录、报警和阻断等。

网络内容审计实现方法与入侵检测系统有相似之处，都需要从网络数据包中提取需要的内容。然而，在实现技术和采用方法方面，网络内容审计比入侵检测系统提出了更高的要求。

一是网络内容审计必须使用更高效的数据采集方法。入侵检测系统一般基于单包处理（部分系统除外），丢失部分数据包只会造成单个入侵的漏报，其它截获的数据包仍然可以被分析检测，而入侵行为一般具有重复、多次的特点，因此，丢失部分数据包对系统不会造成太大影响。然而，对于内容审计系统来说，一般基于连接和会话进行分析，应用连接的一个包被丢失，则整个连接的数据都会失去分析的意义。随着网络的发展，网络带宽正由10M/100M 迅速向千兆发展，要从高速、形似海量的网络数据包中提取需要的内容并实现内容的分析与监控，需要采用更高效的算法。

二是内容审计必须采用更高效的串匹配算法。内容审计系统必须基于多个数据包甚至整个会话连接进行内容匹配，从中找到需要的内容和目标，因此采用高效、快速的串匹配算法是内容审计系统的关键。

三是各种协议数据的分析、还原和基于会话过程的重放。内容审计主要工作在应用层，而基于应用层的协议很多，许多新的应用协议还在不断产生。在同一个会话中，往往存在多个协议同时工作的情况。

实际上，目前市场上还没有一个既高速，又有效的网络信息内容分析与审计平台。为此，我们基于分布式思想和负载均衡与数据分流技术，采用快速串匹配算法，设计了一个新型的高速网络内容审计系统（High-speed Network Audit System），简称 HNAS。

1　系统模型及设计

图1是 HNAS 系统模型。系统由分流器、分析还原系统（主机）、规则库、数据仓库、用户认证管理服务器、查询/管理终端组成。整个系统的连接应采用分布式结构，基于 Intranet 技术连接。

系统按地理位置分成三部分，一是前台部分，包括分流器、分析还原系统等；二是中间部分，包括规则库、数据仓库、用户认证管理服务器等；三是后台部分，主要是各类查询/管理终端。前台部分的每个探测点的设备组成一个内部网，中间和后台部分可以组成一个内部网，也可以单独组网。

前台部分又称探测点，HNAS 系统可以有多个探测点，每个探测点接入一个 ISP 或一个 Internet 子网。探测点的分路器100M/1000M 网卡接入公用信息网交换机的镜像端口，分路器获得的数据按一定的负载均衡算法通过100M 网卡传递到分析还原系统。分析还原系统完成对数据的分析、还原及多协议的协同，然后把处理结果发送到节点控制器，节点控制器对数据进行协调，然后发送到数据中心。

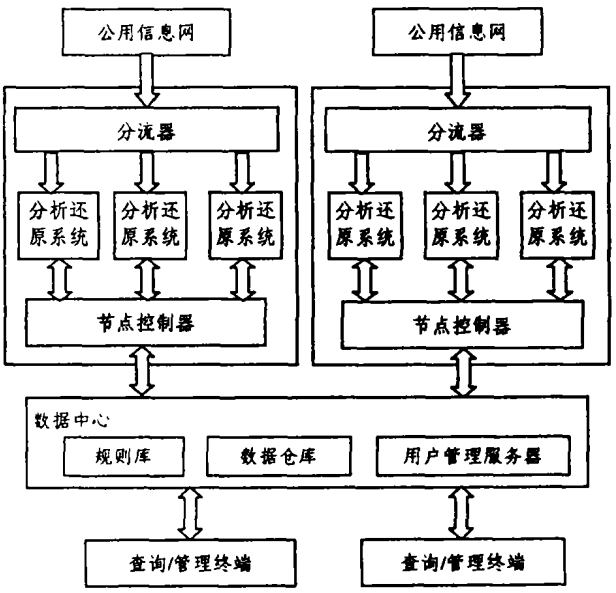

图1　HNAS 系统组成

中间部分又称数据中心。数据中心的规则库保存需要匹配的规则；数据仓库存储符合规则的会话信息，数据仓库可有多台，还可根据需要提供数据备份仓库；用户认证管理服务器完成查询/管理终端用户的身份识别和前台节点接入的认证。

*)国家信息产业部项目。**万国根**　博士研究生，研究方向：网络安全理论与技术，中文信息处理；**秦志光**　教授，博士生导师，研究方向：网络安全理论与技术；**刘锦德**　教授，博士生导师，研究方向：开放系统技术，多媒体技术和虚拟现实应用。

后台部分主要是指查询/管理终端。查询/管理终端可以有多台。其中查询终端根据规则查询数据仓库内容，并实现会话重放，管理终端对规则库进行管理，并对数据仓库内容进行维护，如备份、删除等。

系统运行时，三部分设备保持动态、高速的连接。一方面，分析还原系统动态从规则库获取规则，并将分析还原结果动态保存到数据仓库中；另一方面，数据仓库、用户认证管理服务器随时接收查询/管理终端的请求，提供数据或修改规则。

2 关键技术

HNAS 系统使用的技术很多，这里，根据高速网络环境下的特点，重点介绍数据分流技术，包分析还原技术、字符串匹配技术等。

2.1 高速网络数据分流

HNAS 系统的数据分流器工作在网络的第三层（路由层），其主要功能是将公网数据包（100/1000M）分配给多个能接收100M 数据的分析还原主机。为了保证其性能，同时便于采用硬件实现，在其上不做内容分析，而只做简单的数据包过滤及负载均衡。

1. 数据包过滤　数据包过滤的主要目的是为了达到缩减数据和保护自身安全，去除不关心的网络包并屏蔽一些针对本系统内网的攻击。为了提高吞吐力，包过滤只做简单的基于包头内容的过滤。如 IP 地址、TCP/UDP 端口、TCP 标志位等。由于内容审计主要针对 TCP 和 UDP 协议，数据分流器只接收 TCP 和 UDP 协议的数据，其它协议一律丢弃。这样，经过包过滤之后的网络包数据将大大减少，从而减少分析还原系统的工作负担。

2. 负载均衡算法　为了将网络流量较平均地分配给每个分析还原主机，一般有两种办法。一种是静态分配，如采用循环分配（Round Robin）或加权循环分配（Weighted Round Robin）等，另一种方法是根据端口号，即根据数据包的应用端口号将包发给指定的分析还原主机。然而，这两种算法都不能适应网络内容审计的需要。主要原因是，这些方法有可能将同一个连接的不同数据包分配给不同的分析还原主机。例如，用户使用 FTP 传递数据时，使用的网络端口有控制端口和数据端口。设客户 IP 为 CIP，服务器 IP 为 SIP，需要传递的数据包为1000个，如果采用循环分配算法，则可能使 CIP 与 SIP 传递的数据被分配到不同的分析还原主机；如果采用按端口号分配算法，则可能使控制端口和数据端口的数据被分配到不同的分析还原主机，这两种方法都将导致每个分析还原主机接收的数据不完整，从而得不到完整的连接信息。

为此，在 HNAS 中，设计了一个根据 CIP 和 SIP 进行数据分流的算法。即

输出端口 H＝(CIP＋SIP)mod M ，M 是分流器的输出端口个数。

设有数据包1的源、目标的 IP 为 CIP1、SIP1，输出端口 H1；数据包2的源、目标的 IP 为 CIP2，SIP2，输出端口 H2。如果 CIP1＝ CIP2、SIP1＝ SIP2，或 CIP1＝ SIP2、SIP1＝ CIP2 则：

H1＝(CIP1＋SIP1)mod M ＝(CIP1)mod M ＋(SIP1) mod M＝(CIP2)mod M ＋(SIP2)mod M＝H2

因此，可以确保数据包1和数据包2从同一端口发送。

下面是该负载均衡算法描述如下：

(1)接收到一个 IP 包，利用协议分析取出该数据包的 CIP 和 SIP。

(2)计算 H＝(CIP＋SIP)mod M。

(3)将该 IP 包发给输出端口号 H 对应的分析还原主机。

算法分析：此算法实现简单，计算量小，且易于硬件实现。可能产生的问题是导致数据分流的不均匀，例如，设 M＝8，则如果 CIP＋SIP 都等于 2、10、18、26时，H＝2，然而实际测试表明，在一个中大型交换网络中，其通过的数据包 IP 地址具有均匀分布的特点，因此采用该算法，可以保证系统在中大型网络中的负载均衡。

2.2 高效并行的还原分析技术

对于基于内容的审计系统来说，仅分析零碎的数据包是没有任何价值的，而必需将一个会话中双向传输的所有数据包进行拼接，并排除协商、应答、重传、包头等网络附加信息。HNAS 在进行数据截获时，首先分析数据包的会话特征，然后对属于同一个会话的数据包进行重组、拼接，并去除协商、应答、重传、包头等网络信息，以获取一条基于会话的完整记录。在会话结束后，会话的完整内容信息被传递到数据仓库保存，用户可以根据需要实现会话重现。对于 UDP 上的应用，HNAS 也采用了模拟面向会话的方式。

因特网数据具有阵发性的特点，因此，在 HNAS 系统中，数据接收、分析处理、数据发送均采取缓存技术，以提高系统的整体吞吐率。

典型的缓存技术是分析还原系统的数据处理部分（见图2）。从中看到，我们用多个缓冲区构成缓冲池以便提高并发度。数据接收进程（线程）首先将接收的数据包按协议类型分别存放到相关的缓冲队列中。这里，接收队列1到 n 分别对应不同协议的数据队列，如 HTTP 协议接收队列、SMTP 协议接收队列等。数据处理1到 n 分别对应不同协议的数据分析还原进程。这些分析还原进程从接收队列获取数据包，经还原分析后，存放到对应的发送队列中。发送进程从发送队列获取数据包，然后使用 socket 方法将数据发送到数据仓库服务器。

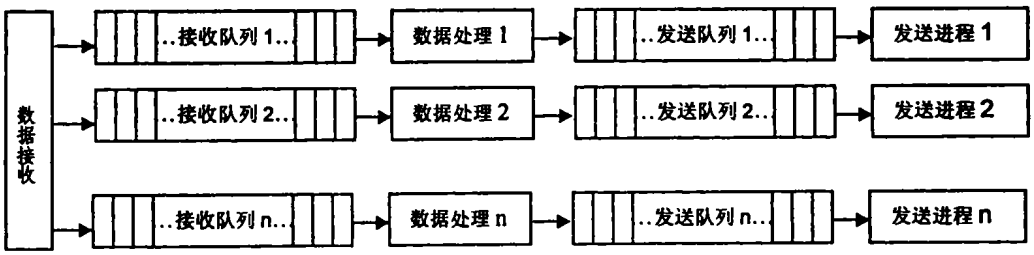

图2　分析还原系统数据流程图

缓冲区是一种临界资源，因此需要协调进程（线程）间的同步与互斥。不难观察，这是较常见的多个生产者和一个消费

者的关系。数据接收进程(线程)是数据包的生产者,但在生产前先要获得接收队列的使用权;数据处理进程(线程)是接收队列的消费者,同时又是发送队列的生产者,在消费前要获得接收队列资源的使用权,消费完毕后必须释放接收队列资源的使用权,然后申请发送队列资源的使用权;发送进程(线程)是发送队列资源的消费者,消费前要获得发送队列资源的使用权,消费完毕后必须释放发送队列资源的使用权。

2.3 快速的字符串匹配算法

字符串匹配是指在文本 $Text=t_1t_2\cdots t_n$ 中检索子串 $Pat=p_1p_2\cdots p_m$ 的所有出现。文[1]记载,在入侵检测系统中(SNORT 版),字符串匹配占到整个 CPU 运行时间的31%,文[2]记载,在 Snort 2.0中,由于采用比以前版本更高效的匹配算法,SNORT 系统的处理效率提高了约500%。由此可见,选择一个好的字符串匹配算法是内容审计系统成败的关键。

判断字符串匹配算法优劣的指标是时间复杂度和空间复杂度。目前,在快速字符串匹配算法中,最著名的是 KMP 算法(Knuth-Morris-Pratt)和 BM 算法(Boyer-Moore)。1992年,Baeya-Yates R 与 Connect G. H[3]提出了比 BM 及 KMP 算法更好的算法(称为 KYG 算法)。然而,不论是 KMP,BM,还是 KYG 算法,它们都仅适用于西文字母串的匹配。若对中西文混合串,则由于字符集的不同而变得难以实现。例如,BM 算法需要 $O(m+\Sigma)$ 内存附加空间(Σ 为字符集的大小),这在中西文混合串中将占用巨大空间;另外,由于每个汉字占两个字节,使用这些算法还可能出现某个汉字的位码和后面汉字的区码组合恰好为另外一个汉字的现象,造成将不匹配的而被判断为匹配的错误。

文[4]根据 KYG 算法的基本思想,提出一种中西文通用的(汉字)字符串匹配算法。该算法的时间复杂性与 BYG 一样为 O(n),且与文本内容及检索串无关,但其空间复杂性比 BYG 算法还低,仅需内存附加空间 O(m+64)

HNAS 系统采用文[4]算法,并根据网络内容审计的特点进行优化和改进。其基本思想是:将检索时的状态以一个整数表示,随着从头到尾对文本字符一个个的扫描,新的检索状态由对整数的简单运算(移位、按位或)得到。当达到一个状态,其对应的整数的最低(右)位为0时,即找到了一个匹配。

设文本 $Text=t_1t_2\cdots t_n$,字符串为 $Pat=p_1p_2\cdots p_m$,字符集为 Σ,则该算法的具体实现步骤是:

(1)引入字符-模式匹配向量 MT。$MT(a)=(MT^1(a)\ MT^2(a)\cdots MT^j(a)\cdots MT^m(a))$,其中 $j=1,\cdots,m$;a 属于 Σ。

定义1 若 $a=p_i$,$MT^i(a)=0$,否则,$MT^i(a)=1$。显然,MT 可预先根据 Σ 及 Pat 求出。

(2)引入搜索状态向量 S。$S_i=(S_i^1,S_i^2,\cdots,S_i^j,\cdots,S_i^m)$ 表示在 Text 的当前位置 j,其前的字符与 Pat 的任意前缀的匹配情况,其中,$i=1,\cdots,m$,$j=1,\cdots,n$。

定义2 若 $p_1\cdots p_i=t_{j-i+1}\cdots t_j$,$S_j^i=0$,否则 $S_j^i=1$。显然,若 $S_j^m=0$,则存在精确匹配串 $t_{j-m+1}\cdots t_j$。

若已求出 S_j(约定:$S_0^i=1$),前进一个字符到 t_{j+1},则当且仅当 $S_j^{i-1}=0$,且 $t_{j+1}=p_i$ 时,$S_{j+1}^i=0$。

(3)向量的整数表示与计算。对于状态向量 $S_j=(S_j^1,S_j^2,\cdots,S_j^i,\cdots,S_j^m)$,由于 S_j^i 或为0,或为1,因此可以用一个 m 位的二进制位串来表示,即 $S_j=S_j^1S_j^2\cdots S_j^i\cdots S_j^m$。若将该位串表示为整数,则 $S_j=\sum_{i=1}^{m}(S_j^i\times 2^{m-i})$;若 S_j 的最底位为0,则找到一个匹配。

同理,对字符-模式匹配向量 MT,$MT(a)=\sum_{i=1}^{m}(MT^i(a)\times 2^{m-i})$;

S_j 到 S_{j+1} 的转换可表示为 $S_{j+1}=S_j>>1|MT(t_{j+1})$。

对于汉字,则在求出 $S_{j+1}[i]$ 之后,如果 $t_{j+1}>0xA0$,则继续前进到 t_{j+2},求出 S_{j+2}。

(4)例子。设 $\Sigma=\{a,b,c,d\}$;Pat = ababc,Text = abdababababc;则在 Text 中查找 Pat 的过程如下:

Text	a	b	d	a	b	a	b	a	b	c
$MT(T_{j+1})$	01011	10101	11111	01011	10101	01011	10101	01011	10101	11110
S_{j+1}	01111	10111	11111	01111	10111	01011	10101	01011	10101	11110

($S_0=11111$)

当到达位置 c,搜索状态向量为11110,其最低(右)位为0,找到一个匹配。

以下实现该算法的类:

```
MAXSYMBOL=256;
MAXPAT=30;
class Search
{
   private:
       unsigned long searchstate,mt[MAXPAT+1],mask;
       char inter_mt[MAXSYMBOL];
       int patlgth;
   public:
     Search(const char*pattern,long len);
     char* find(const char*TextBuf, int n);
};
Search::Search(const char*pattern,long len)
{
   int d,k;
   unsigned char*pText;
   patlgth=len;
   if(patlgth>MAXPAT) patlgth=MAXPAT;
   mask=~(~0<<patlgth);
   for(d=0;d<MAXSYMBOL;d++)inter_mt[d]=0;
   d=1; pText=(unsigned char*)pattern;
   for(k=0;k<patlgth;k++){ if(!inter_mt[*pText]) {inter_
     mt[*pText]=d;d++;};pText++;}
   for(d=0;d<MAXPAT;d++)mt[d]=~0&mask;
   pText=(unsigned char*)pattern;
   for(k=0,d=(1<<(patlgth-1));k<patlgth;d>>=1,k++)
     {mt[inter_mt[*pText]]&=~d;pText++;}
};
char* Search::find(const char*TextBuf, int n)
{
   char*pText;
   unsigned char comparechar;
   pText=(char*)TextBuf;
   searchstate=~0&mask;
   int d=0;
   while (d<n) {
       searchstate=searchstate>>1|mt[inter_mt[(unsigned
         char)*pText]];
       if((unsigned char)*pText>=0xa0){
           if(d<n){ d++; pText++;
               searchstate=searchstate>>1|mt[inter_mt
                 [(unsigned char)*pText]];
           }
       }
       if(!(searchstate&1))return(char*)(TextBuf+d);
           d++; pText++;
       }
       return NULL;
   }
```

算法分析:程序将要匹配的子串 Pattern 及其长度 len 作为参数对类进行初始化。在初始化过程中,根据 Pattern 及字母表 Σ 生成字符-模式匹配向量 MT,其处理时间为 O(m+

(下转第65页)

相同的。图7描述了一个大概的工作流程，没有涉及细节问题。在随后的部分我们以开发的一个简单实例来描述这种框架的具体应用。

5 应用实例分析

系统的构建基于以下一个场景：客户要到北京出差，他想知道各个航空公司所提供的飞机票的价格和公司名，并购买最低价格的商家的票，同时他决定预定客房也是最低价格的客房。传统的实现方式是：客户登录各个公司的商务网站，分别查询符合条件的飞机票的价格。在对各个公司的价格都查询完毕再决定自己购买哪个公司的飞机票，定房也是如此。在基于移动代理的系统中，我们可以指定各个供应商的地址（事实上，这个地址集合可以通过查询商务中心来定，在演示中我们指定），然后派出主代理到其中一个信任主机上（通过查询主机信任度表决定如图8所示），在信任主机上再产生查询价格的子代理到各个主机上（安全性的考虑），这些子代理可以是并行的即有几个子代理去并行查询各个主机，也可以是串行的即子代理在一个主机上查询完毕再到另一个主机这样最简单，我们只需一个子代理就可以完成任务了。在系统中我们选择了按功能划分的并行方式，因为这样可以直观地考查主从代理是如何工作的。相关的角色以及控制策略如表1和表2所示。

TrustTB：表

Host	Vaight
atp://192.168.0.1:3000	0
atp://192.168.0.1:4434	1
atp://192.168.0.2:4434	2
atp://192.168.0.2:3000	3
atp://192.168.0.3:4434	4
atp://192.168.0.3:3000	5
atp://192.168.0.1:6000	7
atp://192.168.0.1:5000	8

图8 主机信息度表

表1 角色

商务代理	主代理
购票代理	从代理一
定房代理	从代理二

表2 控制策略

路由	顺序
执行/检查	无
合并/处理结果	价格最低
任务分解	按功能分

我们以在4434端口作为源宿主地并且在地址选择框中输入下列两个地址为例：atp：//192.168.0.2：3000；atp：//192.168.0.1：5000来说明系统工作的流程。当然也可以其它地址输入，但要注意输入的地址的格式。同时，要在数据表中建立相应的主机的项。基本流程是：主代理知道192.168.0.1：5000是最信任主机决定到那儿去然后我们根据功能生成一个购票代理一个定房代理，购票代理在最便宜的地方定票而定房也是在最便宜的地方实现的，为简单起见我们让主机加端口号来区分不同主机，并且每个主机承担了票和房的双重身份。实验表明这种应用是切实有效的。

结束语 解决移动代理的安全问题是使其真正实用的关键要素。本文在阐述移动代理安全问题及其对策的基础上从设计模式的角度来研究移动代理本身的安全，提出了安全的移动代理平台（SMAP），并给出了一个应用实例，下一步我们将把传统的安全技术和基于设计配置的安全技术结合起来以期建立真正安全实用的移动代理应用系统。

参 考 文 献

1 Chess D. Security Issues in Mobile Code Systems. In：Giovanni Vigna(ED.)，Mobile Agent Security，LNCS 1419，Springer，1998. 1～14

2 Karnik N M，Tripathi A R. Design Issues in Mobile Agent Programming Systems. IEEE Concurrency，1998，6(3)：52～61

3 Wilhelm U G. et al. A Pessimistic Approach to Trust in Mobile Agent Platforms. IEEE COMPUTING，September·October 2000. 40～48

4 Schaaf M，Maure F. Integrating Java and CORBA：A Programmer's Perspective. IEEE INTERNET COMPUTING，January-February 2001. 72～78

5 Wang X F，Yi X ，Lam K Y. Secure Information Agent for Internet Trading [A]. 11th Australian Joint Conference on Artificial Intelligence'98，Vol 1 544[C]. Brisban，Australia：Springer-Verlag Publishers，1998. 183～194

6 Farmer W M，Gutman J D，Swarup V. Security for Mobile Agents：Authentication and State Appraisal. In：Proc. Of the 4th European Symp. On Research in Computer Security，Rome，Italy，LNCS No. 1146，Springer，1996. 118～130

7 Eckel B. Thinking in Java，2nd Edition，Release 11. Prentice-Hall，2000

8 Pfleeger C P. Security in Computing，Second Edition. Upper Saddle River，New Jersey：Prentice Hall，1997

9 Schneier B. Applied Cryptography：Protocols，Algorithms，and Source Code in C. IS John Wiley & Sons，ISBN：0-47-111709-9

（上接第61页）

$|\Sigma|$），类进行初始化可以在匹配前完成，因此这点时间在整个分析还原过程中可以忽视不计。M 等于 len，在中西文混合串中，Σ 取值范围为0x00..0xff，因此 $|\Sigma|=256$。为了存放 MT，需 m 个字的附加空间，另外引入字符数组 inter_mt [256]，将 Pat 中的 $p_1\cdots p_m$ 映射到1…m，而将非 Pat 中的所有字符映射到 inter_mt 占的空间为256/4＝64，因此总的附加空间为64＋m。

串匹配时间仅与文本长度有关，而与模式及文本内容无关，其最好、最坏及平均匹配时间均为 o(n)。

结束语 网络内容是计算机安全领域的又一个研究方向。本文提供高速网络环境下分布式网络审计系统对此进行了大胆的探索与实践，并且取得了较好的效果。系统提出的大流量数据分流概念、缓存管理技术、多字符集字符串匹配算法等在网络审计、入侵监测等领域都有重要的参考价值。

参 考 文 献

1 Fisk M，Varghese G. Fast Content-Based Packet Handling For Intrusion Detection. http：//www. public. lanl. gov/mfisk/papers/ucsd-tr-cs2001-0670. pdf

2 Desai N. Increasing Performance in High Speed NIDS. http：// www. linuxsecurity. com/articles/ intrusion_detection_article-4617. html

3 Baeya_Yates R，Gonnel G H. A new approach to text searching. Comm. ACM，1992，35(10)：74～82

4 邹旭楷，王素琴. 一种快速(汉字)字符串匹配算法. 小型微型计算机系统，1993(11)