第28卷 第2期 计 算 机 工 程 2002年2月
Vol.28 № 2 Computer Engineering February 2002
· 安全技术 · 文章编号：1000—3428(2002)02—0165—03 文献标识码：A 中图分类号： TP393.08

分布式入侵检测系统的数据采集技术

张铭来，金成飚，赵文耘
(复旦大学计算机科学系，上海 200433)

摘 要：讨论了分布式入侵检测系统中的数据采集子系统，通过对几种不同的数据采集技术的优势和局限的分析，提出了分布式入侵检测系统的数据采集子系统的实现方法。

关键词：入侵检测；分布式入侵检测系统；数据采集

Data Collection Technology for Distributed Intrusion Detection System

ZHANG Minglai，JIN Chenbiao，ZHAO Wenyun
(Computer Science Department of Fudan University, Shanghai 200433)

【Abstract】 This paper discusses the data collection sub-system of distributed IDS, analyzes different kinds of data collection technology, and claims a method to implement it in distributed IDS.

【Key words】 Intrusion detection；Distributed IDS；Data collection

1 背景

1.1 入侵监测系统概念的提出

入侵检测，顾名思义，便是对入侵行为的发觉。它通过对计算机网络或计算机系统中的若干关键点收集信息并对其进行分析，从中发现网络或系统中是否有违反安全策略的行为和被攻击的迹象。

一个良好的IDS系统应该具备以下特征：

(1)必须在管理人员很少干预的时候也能持续稳定运行。

(2)必须在系统崩溃后能保存当时的状态并自动恢复，继续对系统或网络进行监控。

(3)必须能发现自己是否被入侵者攻击或已经被入侵者修改。

(4)必须在运行的计算机上消耗尽量少的资源，不影响正常操作。

(5)能适应运行环境的变化，在不同的系统中都能正常运行。

(6)随着网络的扩大和被监测系统的增加，能进行相应的扩展。

(7)如果IDS的一部分意外停止工作，对其它部分的影响要达到最小。

(8)应该允许动态设置，如果IDS在监测许多系统，修改IDS的设置不应该要求被监测的系统全部重新启动。

1.2 入侵检测系统的分类

1.2.1 集中式IDS和分布式IDS

IDS按照数据处理的方式不同分为集中式IDS和分布式IDS。分布式IDS的数据采集和分析在多个主机上完成，而集中式IDS的数据采集可以是分布式的，但数据处理是在一台主机上进行的。集中式IDS存在以下问题：

(1)IDS所在的计算机很可能是攻击的目标，如果入侵者能使这台计算机瘫痪或运行得非常缓慢，那么整个网络就会失去保护。

(2)可伸缩性有限。由于一台计算机的处理能力有限，一个IDS要负责整个网络的安全，网络不能过大，否则IDS来不及处理。如果采用分布式数据采集还会造成更大的网络负担。

(3)重新设置IDS或增加新的功能比较困难。重新设置IDS通常要修改配置文件、在数据表中增加一项内容或添加一个新的模块，这常常要求IDS重新启动才能生效。

分布式IDS的结构使它具有更好的伸缩性和扩展性，分布在不同计算机上的IDS模块相互独立且能相互通信，其中一部分出故障不会造成所有的模块全部瘫痪。

1.2.2 基于网络和基于主机的数据采集

入侵监测系统(IDS)按照数据采集的方式不同可以分为基于网络的IDS和基于主机的IDS。基于网络的IDS的数据源通常是侦听网络上的数据包，而基于主机的IDS的数据源通常是系统日志或系统调用。

现有的IDS发现的入侵中大部分是在主机上发生的：执行一个命令，向一个服务提供不合适的数据，等等。虽然攻击可能是通过网络进行的，但攻击最终发生在主机上。纯粹针对网络的攻击是那些将网络带宽占满的洪流攻击，但其中大部分也能被基于主机的数据所发现。例如，要发现Ping flood攻击，只要在主机的ICMP层出现大量的Echo Request包就能作出判断。有一种情况使基于网络的数据采集优于基于主机的数据采集，即对网络的攻击不会有任何主机响应(例如将数据包发至所有主机都关闭的端口)。然而即使在这种情况下，在主机的底层网络协议中仍能发现攻击。

我们认为基于主机的数据采集优于基于网络的数据采集，理由如下：

(1)基于主机的数据采集能准确地知道主机上发生什么，而基于网络的数据采集是在"估计"主机对收到某一个数据包的反应，因此容易受到"插入"(Insertion)攻击和"逃避"(Evasion)攻击，Ptacek和Newsham在文献[1]中详细分析了这两种攻击的过程。

(2)在高速网络环境中，一个基于网络的侦听器(Sniffer)很可能因为来不及处理而丢掉一些数据包，而且Sniffer不能收到跨网段的数据包。

(3)基于网络的数据采集对加密的数据无能为力。当浏览器和一个使用安全套接字(SSL)的Web服务器相连时，数据包直到到达Web服务器应用本身时才被解密，这种应用级加密会使网络监视器对许多攻击视而不见。

1.2.3 主机型IDS的数据采集可以分为直接监视和间接监视

作者简介：张铭来(1977～)，硕士生，研究方向：软件工程、网络管理；金成飚，硕士生；赵文耘，教授、中国计算机学会软件工程专委会副主任

收稿日期：2001-05-16

直接监视获取的数据是从产生数据的对象那里获取数据，例如我们要直接获取CPU负荷的数据，就要从内核结构中获得数据。如果要监视inetd后台进程提供的网络服务，我们就可以直接从inetd获取信息。

间接监视获取的数据通常来自于日志文件。间接监视CPU负荷的信息需要查找相关的日志文件，间接监视网络服务需要查阅inetd产生的日志文件。

在实现IDS的时候，我们认为直接监视优于间接监视，理由如下：

(1)日志文件作为IDS的数据源时可能已经被入侵者改动过。

(2)有些事件没有记录在日志文件中。例如，并非inetd的每一个动作都记在日志文件中。

(3)由于日志文件在产生的时候并不知道IDS会使用它，所以日志文件的容量通常很大，其中很多是IDS不关心的，如果全部由IDS处理将会降低其效率。对于采用直接监视的方法来获取数据的IDS而言，它们只获取需要的数据，这样产生的数据量就小得多。

(4)间接监视所用的数据源的产生和IDS访问这些数据之间存在一个延迟，而直接监视时IDS能立即对收到的数据作出反应。

如果IDS要用直接监视来采集数据，必须有相关操作系统的支持。一种方法是在内核里加入一些代码来实现对系统的监视；另一种方法是写一些程序与现成的应用进行通信，通过观察应用的输入和输出来监视。因此分布式IDS中代理的实现可以有两种选择：内部代理和外部代理。内部代理是指实现代理的程序已经成为被监视的应用的一部分；外部代理是用独立的程序来监视某个应用。

因为内部代理是被监视的应用的一部分，所以不容易被攻击者关闭或修改，并且能够最快地得到应用程序的有关信息。如果能成功地实现内部代理，它占用的系统资源比外部代理要少，因为它的运行不需要产生新的进程。内部代理能够比外部代理获取更多的应用程序的信息，因为它能访问应用内部的数据结构和变量。而外部代理相对而言比较容易修改、增添或删除。

从软件工程的角度看，内部代理与外部代理各有特点：

(1)错误的引入：内部代理的使用更容易引入错误，因为要修改原来应用程序的代码，外部代理也可能引入错误(例如消耗过多的系统资源)。然而，如果内部代理的实现只需修改很少的应用程序代码，就能控制错误的出现，也更容易进行检查。在OpenBSD系统上实现的内部代理能发现15种攻击[2]，包括Land，TearDrop，Ping of Death等等，这个代理只修改了内核中的73行代码，另外附加了两个文件共354行代码。因此采用内部代理是能够控制错误的。

(2)维护：外部代理更容易维护，因为它们与被监视的应用程序是分开的。

(3)软件大小：内部代理更小，因为它们是现有程序的一部分，在运行时避免了产生新的进程。

(4)完整性：由于内部代理能获取被监视的应用的所有信息，而外部代理只是从外部获取信息，因此内部代理能更全面地了解被监视的应用所发生的一切。

(5)正确性：内部代理能获取更多的信息，它得出的结论比外部代理更能反映真实情况。

通过对内部代理和外部代理的综合比较，我们发现外部代理在易用性和维护方面比较好，而内部代理在监视能力以及对主机的影响方面要更好一些。两种代理都可以应用于分布式IDS中，在不同场合可以发挥各自的优点。

2 分布式IDS系统的结构

分布式IDS中包括以下几种元素：代理、转发器、监视器、过滤器和用户界面，其结构如图1所示。

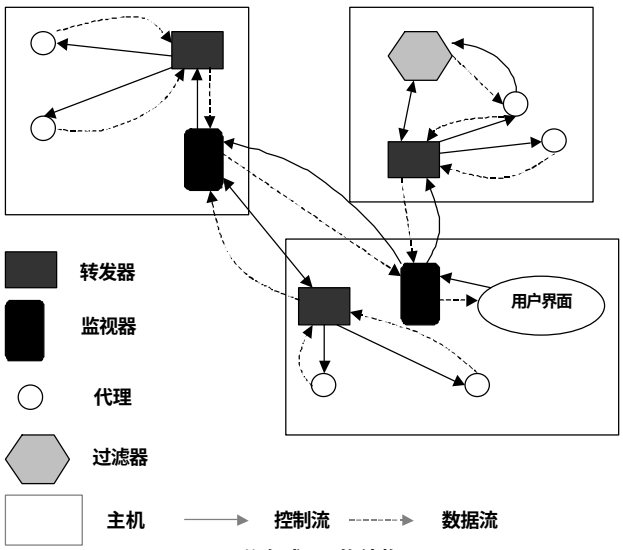

图1 分布式IDS的结构图

代理是负责监视主机的某个方面的独立实体，它将收集的信息向转发器报告。例如一个代理可以监视一个主机的许多Telnet连接，收集可疑的信息。代理向指定的转发器报告它发现的情况，但它无权直接产生报警。通常是转发器或监视器根据代理送来的信息产生报警。转发器通过不同代理送来的报告能够对主机的状态有一个全面的了解，监视器通过不同转发器发来的报告对监视的网络有一个全面的了解。

过滤器可以看作是代理的一个数据选择和抽象的层次。代理在收集主机信息时，遇到两个问题：(1)一个系统中可能有多个代理使用同一个数据源，例如在Unix中使用同一个日志文件，每个代理都对数据源处理一遍意味着重复劳动；(2)可能有一种代理在不同的Unix版本中都要用到，但不同的Unix版本日志文件的存放位置可能不同，文件格式可能不同，这意味着要写几个代理来适应不同的版本。而过滤器的提出正是为了解决这两个问题。代理向过滤器提出条件，过滤器返回符合条件的纪录。过滤器还将处理与系统版本相关的问题，使得实现统一功能的代理能用于不同的操作系统。

转发器负责收集主机上各个代理提供的信息，以及和外界的通信。每个被分布式IDS监视的主机上都有一个转发器，它既要控制相应代理的工作，又要对代理送来的数据进行处理，还要对监视器发来的请求进行响应。

监视器控制转发器的工作，并对转发器送来的数据进行处理。监视器和转发器最大的区别是：监视器控制不同主机上的转发器，而转发器控制一个主机上的代理。监视器与用户界面通信，从而为整个分布式IDS提供了管理的入口。

用户界面为用户提供可视化的图形来进行管理，并将用户的指令传达给监视器。

由于代理是独立运行的实体，它可以从系统中增加或删除而不影响其他元素，不需要重新启动IDS。使用代理来进行IDS的数据采集有利于我们测试，有利于在不同的系统中进行移植。如果一个代理停止工作，有两种情况：(1)这个代理是完全独立的，能够自己产生结果，那么将仅仅丢失这

个代理产生的数据，其他的代理仍会正常工作；(2)这个代理产生的数据是其他几个代理的输入，那么这一组代理都将不能正常工作。无论哪一种情况，损失都限制于一组代理，其他代理会继续工作，这比起采用集中式的数据采集损失要小得多。用代理来进行数据采集和分析还有以下特点：

·在实现代理的时候我们既可以从系统日志中获得数据，也可以从网络中获取数据包，所以采用代理来进行数据采集的IDS能够吸取基于网络的IDS和基于主机的IDS两者的优势。

·由于代理的启动和关闭不会干扰IDS的其他部分，代理的升级很方便，只要代理的接口保持不变，IDS的其他元素甚至不知道代理已经升级。

·如果代理是主机上的不同进程，它们可以根据用途的不同采用不同的编程语言来实现。

3 分布式IDS数据采集子系统的实现

最初考虑把每个代理用独立的进程来实现，但要收集和分析的数据源通常来自内核，这意味着要记录内核产生的数据必须将它从核心空间传到用户空间，从而造成上下文切换，这必将增加被监测主机的负载。随着代理数量的增加，过大的负载将影响用户对该主机的正常使用，当代理用脚本语言如Perl，TCL/Tk 来实现的时候将会消耗很大的资源。

一个改进方法是使用编译语言如C语言，这将会减小对内存和CPU的消耗，但仍然不能解决上下文切换的问题，以及许多进程之间切换带来负载。我们又考虑到使用支持多线程的编程语言，让每个代理成为一个独立的线程而不是一个独立的进程，这当然会进一步减小每个代理的负载，但依旧没有解决上下文切换的问题。最后我们决定把代理变成Unix内核的一部分，例如一个监视网络连接的代理可以直接从内核中读取相关的数据结构，而不用重复地执行netstat命令。将代理写进内核可以解决上面的一些问题：

·避免了上下文切换，因为代理本身就在内核里面。

·信息收集和处理的地方和信息产生的地方非常接近，所以入侵者要修改代理输出的信息就不很容易。

·入侵者要屏蔽掉代理变得更加困难，因为必须修改内核。

转发器也可以成为内核的一部分，这样只有当转发器将数据处理的结果发送到监视器时才和内核以外的部分通信。

代理、转发器、监视器的设计和实现有两个方向：一个是尽量减少代理和转发器的工作，这样它们实现起来就比较容易而且消耗的资源可以很小，监视器设计得比较复杂，主要的工作由它来完成，这个方向发展到极端就是采用分布式数据采集的集中式IDS，这种系统存在的问题是代理和转发器进行的处理很少，于是网络上传输的数据量就会很大。另一个方向是为了减小网络的负载，让代理和转发器做更多的数据处理，这种方向的极端就是每个被监视的主机上的代理和转发器都相当于一个独立的IDS，如果代理和转发器设计得复杂，要把它们写进内核就会难以实现。

以下是针对几种常见的网络攻击的代理在OpenBSD2.6上的实现：

⑴ Land攻击

这种攻击是向一台机器开放的端口发送TCP SYN包，并且该包的目的地址和端口地址与源地址和源端口相同，这将导致被攻击的机器不停地向自身发ACK包，最终造成当机。

OpenBSD2.6一接受到这些数据包就会将其过滤，此时套接字还处于Listen状态，我们只要在丢弃这些数据包前截取这个信息，就能实现代理。

```
Case TCPS_LISTEN:   {
…
if (ti->ti_dst.s_addr = = ti->ti_src.s_addr {
        向转发器发送一条Land攻击的消息；
        Goto drop；              /* 丢弃该数据包 */
        }
}
```

⑵ Teardrop攻击

这种攻击由两个重叠的IP分片组成。第1个是大的IP分片，第2个IP分片的结束偏移量(offset)在开始偏移量之前，有些系统会给第2个分片分配一个大小为负数的内存，这将使系统崩溃。

OpenBSD2.6对接受的IP分片的偏移量要进行检查，我们实现的代理就可以放在里面：

```
I = p->ipqe_ip->ip_off + p->ipqe_ip->ip_len - ipqe->ipqe_ip->ip_off ；
If (I >0) {
        If (I >= ipqe->ipqe_ip->ip_len) {
        向转发器发送一条Teardrop攻击的消息；
        goto dropfrag                    /* 丢弃该分片 */
        }
}
```

⑶ TCP RST攻击

TCP RST攻击是由攻击者发出一个伪造的TCP RST包，将已经建立的TCP连接断开。这种攻击的发生是因为有些系统在处理RST包时不检查TCP序列号。在OpenBSD2.6中对RST包的TCP序列号进行了检查，如果发现TCP序列号不对，就有可能是TCP RST攻击。源代码如下：

```
If (tiflags & TH_RST) {
        If (ti->ti_seq!= tp->last_ack_sent) {
        向转发器发送一条TCP RST攻击的消息；
        goto drop；
        }
}
```

⑷ Ping of Death攻击

分片的ICMP响应请求(Echo Request)包组装以后长度大于IP包的最大值，就产生Ping of Death攻击。这是一个比较老的攻击方法，OpenBSD2.6对这种攻击进行了过滤：

```
if (( next + (ip->hl<<2)) > IP_MAXPACKET) {
        if (ip->ip_p = = IPPROTO_ICMP)
        向转发器发送一条Ping of Death攻击的消息；
        }
```

⑸ Smurf攻击

A主机冒充B主机向IP广播地址发送ICMP响应请求包，B主机将会收到大量的ICMP响应回答(echo reply)包，这时B主机就受到Smurf攻击。由此可见Smurf攻击包括两部分：反射器是ICMP响应回答包的发送者，受攻击者是ICMP响应回答包的接收者。

OpenBSD2.6可被设置为忽略IP广播响应请求，但这不是默认设置，因为向广播地址发送请求是网络管理的一个有用的工具，要防止OpenBSD2.6受攻击者利用成为反射器，可插入一个代理：

```
Case ICMP_ECHO
        If (!icmpbmcastecho && (m->m_flags & (M_MCAST|M_BCAST))!=0)
        向转发器发送一条Smurf Reflector攻击的消息；
```

Ping程序在发送ICMP响应请求时把ICMP包中的Identification域设置为进程号(pid)并且在一个原始套接字(Raw Socket)上侦听所有的ICMP包，当地址、序列号和ID域都匹配时才报告。我们可以用

(下转第194页)

显然由传输网络的性能决定，最坏的情况下可以用网络最大延迟时间来表示。这种情况下所产生的问题是：采用网络最大延迟时间来确定控制时间虽然总是能满足媒体实时同步表现的要求，但由于通信网络上的数据传输本质上具有随机性，有些数据单元可能会经过较小的延迟而达到目的端。所以必须要在目的端加上缓冲器以保留这些提前到达的数据单元且对每一个媒体流都应设置一个独立的缓冲器。

另外一个问题是，由于网络对于每一个数据单元具有不同的传输时延，这样就导致在相同的时间间隔里，到达缓冲器的数据单元并不相同。假设在某一段时间里传输网络对于其上传播的数据单元具有较小的延迟，则这段时间里到达缓冲器的数据量必然增多，但媒体数据的播放速率却是恒定的，比如每秒播放25帧，因此这段时间里滞留在缓冲器中的数据量也将增加。由于缓冲器容量的限制，随着时间的推移，当缓冲器的容量到达满载时将会导致后续部分媒体数据的丢失，从而影响播放效果。与之相反，如果某一段时间里网络具有较大的传输延迟，就可能会出现缓冲器中无数据可供播放的情形。在本框架中我们采用基于反馈信息的缓冲器的流控机制。其原理是缓冲器根据当前的容量状态，发出相应的反馈信息给远端服务器，由服务器根据该反馈信息动态调整媒体数据的发送数率。

3.3 用户接口与交互

所有媒体流的播放必须在用户的交互下由媒体播放管理器来予以组织播放。用户的交互通过用户接口向同步规范模块、同步管理模块发出指令，然后由这些模块将其转换成相应的格式来组织发起、同步、管理播放各媒体流。

另外，用户交互的异步性将会增加系统同步的难度和复杂度。例如，当用户要求reverse(反向)播放当前正在播放的媒体流时，将意味着必须重新向远端服务器请求那些业已播放完的媒体数据。在这个过程中，用户首先通过用户接口同时发出"reverse"消息给媒体播放管理器和同步管理模块。媒体播放管理器接到该消息后将立即停止当前的媒体播放，同时将管理器中当前正在播放的媒体数据单元的序列号返回给缓冲器，再由缓冲器将该序列号反馈给远端服务器，远端服务器在接到该反馈信息后，将停止当前的传输过程，并且重新配置其传输参数以便发出满足客户端需要的媒体数据。同步管理模块接到"reverse"消息后，将停止同步器1和同步器2中的同步操作，同时清除缓冲器中当前的及后续进来的不满足要求的媒体数据，直到服务器发回了客户端所需要的数据，由数据序列协调模块负责确定服务器是否发回了满足要求的数据。

显然，在用户发出"reverse"的操作指令到实现该操作将由于上述过程的存在而不可避免地造成一段时间延迟。为了同步该用户的交互，我们采用在客户端本地存储器中开辟一块一定容量的存储区域来存放那些最近播放完的媒体数据。这样，当用户发出"reverse"命令后，可以快速地从本地存储器中取出所需的媒体数据进行播放，而向远端服务器请求新的媒体数据的过程也在并行进行中。这样的处理大大地缩短了"reverse"发出后其实现的延迟时间，所付出的代价是需要额外占用本地的一块存储空间以及该存储空间与缓冲器进行媒体数据序列号协调所增加的复杂性。其他的用户交互还有：pause,resume,skip等，但较"reverse"而言，较为简单。

4 结束语

多媒体技术是目前计算机科学与技术领域研究最活跃的内容之一，而多媒体信息的同步作为分布式多媒体信息系统的关键问题，已经越来越受到关注。本文在媒体同步的4层参考模型的基础上提出了一个实际应用的分布式多媒体系统的同步框架，并对该框架中的一些关键内容如缓冲器及其控制时间、用户接口与用户交互等进行了讨论。该框架不要求很高的网络带宽和QoS参数，在当前广泛使用的TCP/IP网络环境中可取得较好的效果。

参考文献

1 Woo M,Qazi N U,Ghafoor A.A Synchronization Framework for Communication of Pre-orchestrated Multimedia Information.IEEE Network Magzine,1994,8(1):52-61

2 Little T,Ghafoor A.Multimedia Synchronization Protocols for Broad - band Integrated Services.IEEE Journal on Selected Areas in Commu - nication,1991-12,9:1368-1382

3 Georganas N,Steinmets R,Nakahawa T(Eds.).Synchronization Issues in Multimedia Communications.IEEE Journal on Selected Areas in Communication,1996-01,14(1)

☆☆☆

(上接第167页)

同样的方法来检查ICMP响应回答，将ICMP的ID和所有的原始套接字的ID比较(在函数check_smurf()中实现)，如果不匹配就在ICMP层将该数据包丢弃；如果匹配就将数据包发至响应的原始套接字。

```
Case ICMP_ECHOREPLY:
        …
        If (check_smurf(id_in,id_out) {
                向转发器发送一条Smurf攻击的消息；
                goto freeit；
        }
        goto RawSocket；
```

4 结束语

本文分析了分布式IDS数据采集时运用的几种技术：基于主机的和基于网络的数据采集技术，直接监视和间接监视的数据采集技术。并给出了针对几种常见的网络攻击如何实现代理的方法。

参考文献

1 Ptacek T H,Newsham T N.Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection. Secure Networks, Inc.,1998-01

2 Balasubramaniyan J S,Garcia-Fernandez J O,Spafford E,et al.An Arch - itecture for Intrusion Detection Using Autonomous Agents.COAST Laboratory, Purdue University,1998-05

3 Network- vs.Host- based Intrusion Detection.from www.iss.net

4 Escamilla T.Intrusion Detection.1999

5 Spafford E,Zamboni D.Data Collection Mechanisms for Intrusion Detection System.2000-06