服务器知识手册（三十）

狄登峰 主编

目　　录

Linux 系列

用 Linux 防火墙伪装抵住黑客攻击

防火墙可分为几种不同的安全等级。在 Linux 中，由于有许多不同的防火墙软件可供选择，安全性可低可高，最复杂的软件可提供几乎无法渗透的保护能力。不过，Linux 核心本身内建了一种称作“伪装”的简单机制，除了最专门的黑客攻击外，可以抵挡住绝大部分的攻击行动。

当我们拨号连接 Internet 后，我们的计算机会被赋给一个 IP 地址，可让网上的其他人回传资料到我们的计算机。黑客就是用你的 IP 来存取你计算机上的资料。Linux 所用的“IP 伪装”法，就是把你的 IP 藏起来，不让网络上的其他人看到。有几组 IP 地址是特别保留给本地网络使用的，Internet 骨干路由器并不能识别。像作者计算机的 IP 是 192.168.127.1，但如果你把这个地址输入到你的浏览器中，相信什么也收不到，这是因为 Internet 骨干是不认得 192.168.X.X 这组 IP 的。在其他 Internet 上有数不清的计算机，也是用同样的 IP，由于你根本不能存取，当然不能侵入或破解了。

那么，解决 Internet 上的安全问题，看来似乎是一件简单的事，只要为你的计算机选一个别人无法存取的 IP 地址，就什么都解决了。错！因为当你浏览 Internet 时，同样也需要服务器将资料回传给你，否则你在屏幕

上什么也看不到，而服务器只能将资料回传给 Internet 骨干上登记的合法 IP 地址。

“IP 伪装”，就是用来解决此两难困境的技术。当你有一部安装 Linux 的计算机，设定要使用“IP 伪装”时，它会将内部与两网络桥接起来，并自动解译由内往外或由外至内的 IP 地址，通常这个动作称为网络地址转换。

实际上的“IP 伪装”要比上述的还要复杂一些。基本上，“IP 伪装”服务器架设在两个网络之间。如果你用模拟的拨号调制解调器来存取 Internet 上的资料，这便是其中一个网络；你的内部网络通常会对应到一张以太网卡，这就是第二个网络。若你使用的是 DSL 调制解调器或缆线调制解调器（CableModem），那么系统中将会有第二张以太网卡，代替了模拟调制解调器。而 Linux 可以管理这些网络的每一个 IP 地址，因此，如果你有一部安装 Windows 的计算机（IP 为 192.168.1.25）位于第二个网络上（Etherneteth1）的话，要存取位于 Internet(Etherneteth0) 上的缆线调制解调器（207.176.253.15）时，Linux 的“IP 伪装”就会拦截从你的浏览器所发出的所有 TCP/IP 封包，抽出原本的本地地十（192.168.1.25），再以真实地址（207.176.253.15）取代。接着，当服务器回传资料到 207.176.253.15 时，Linux 也会自动拦截回传封包，并填回正解的本地地址（192.168.1.25）。

Linux 可管理数台本地计算机并处理每一个封包，而不致发生混淆。作者有一部安装 SlackWareLinux 的老 486 计算机，可同时处理由四部计算机送往缆线调制解调器的封包，而且速度不减少。

在第二版核心前，“IP 伪装”是以 IP 发送管理模块

（IPFWADM，Ipfwadm）来管理。第二版核心虽然提供了更快、也更复杂的 IPCHAINS，但仍旧提供了 IPFWADMwrapper 来保持向下兼容性，因此，作者在本文中会以 IPFWADM 为例，来解说如何设定“IP 伪装（您可至

http://metalab.unc.edu/mdw/HOWTO/IPCHAINS-HOWTO.html 查询使用 IPCHAINS 的方法，该页并有“IP 伪装”更详尽的说明）。

另外，某些应用程序如 RealAudio 与 CU-SeeME 所用的非标准封包，则需要特殊的模块，您同样可从上述网站得到相关信息。

作者的服务器有两张以太网卡，在核心激活过程中，分别被设定在 eth0 与 eth1。这两张卡均为 SN2000 式无跳脚的 ISA 适配卡，而且绝大多数的 Linux 都认得这两张卡。作者的以太网络初始化步骤在 rc.inetl 中设定，指令如下：

IPADDR=“207.175.253.15”

#换成您缆线调制解调器的 IP 地址。

NETMASK=“255.255.255.0""

#换成您的网络屏蔽。

NETWORK=“207.175.253.0""

#换成您的网络地址。

BROADCAST=“207.175.253.255""

#换成你的广播地址.

GATEWAY=""207.175.253.254""

#换成您的网关地址.

#用以上的宏来设定您的缆线调制解调器以太网卡

/sbin/ifconfigeth0${IPADDR}broadcast${BROADCA

ST}netmask${NETMASK}

#设定 IP 路由表

/sbin/routeadd-net${NETWORK}netmask${NETMASK}eth0

#设定 intrang 以太网络卡 eth1，不宏指令

/sbin/ifconfigeth1192.168.1.254broadcast192.168.1.255.netmask255.255.255.0

/sbin/routeadd-net192.168.1.0netmask255.255.255.0eth1

#接着设定 IPfwadm 初始化

/sbin/ipfwadm-F-pdeny

#拒绝以下位置之外的存取

#打开来自 192.168.1.X 的传送需求

/sbin/ipfwadm-F-am-S192.168.1.0./24-D0.0.0.0/0

/sbin/ipfwadm-M-s60030120

就是这样!您系统的"IP 伪装"现在应该正常工作了。如果您想得到更详细的信息，可以参考上面所提到的 HOWTO，或是至

http://albali.aquanet.com.br/howtos/Bridge+Firewall-4.html 参考 MINIHOWTO。另外关于安全性更高的防火墙技术，则可在

ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO/Firewall-HOWTO 中找到资料。

IDS 及其 Linux 下的实现

1、入侵检测系统简介

当越来越多的公司将其核心业务向互联网转移的时

候，网络安全作为一个无法回避的问题呈现在人们面前。传统上，公司一般采用防火墙作为安全的第一道防线。而随着攻击者知识的日趋成熟，攻击工具与手法的日趋复杂多样，单纯的防火墙策略已经无法满足对安全高度敏感的部门的需要，网络的防卫必须采用一种纵深的、多样的手段。与此同时，当今的网络环境也变得越来越复杂，各式各样的复杂的设备，需要不断升级、补漏的系统使得网络管理员的工作不断加重，不经意的疏忽便有可能造成安全的重大隐患。在这种环境下，入侵检测系统成为了安全市场上新的热点，不仅愈来愈多的受到人们的关注，而且已经开始在各种不同的环境中发挥其关键作用。

本文中的"入侵"（Intrusion）是个广义的概念，不仅包括被发起攻击的人（如恶意的黑客）取得超出合法范围的系统控制权，也包括收集漏洞信息，造成拒绝访问（DenialofService）等对计算机系统造成危害的行为。

入侵检测（IntrusionDetection），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得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。进行入侵检测的软件与硬件的组合便是入侵检测系统（IntrusionDetectionsystem,简称 IDS）。与其他安全产品不同的是，入侵检测系统需要更多的智能，它必须可以将得到的数据进行分析，并得出有用的结果。一个合格的入侵检测系统能大大的简化管理员的工作，保证网络安全的运行。

具体说来，入侵检测系统的主要功能有：

A)监测并分析用户和系统的活动；

B)核查系统配置和漏洞；

C)评估系统关键资源和数据文件的完整性；

D)识别已知的攻击行为；

E)统计分析异常行为；

F)操作系统日志管理，并识别违反安全策略的用户活动。

由于入侵检测系统的市场在近几年中飞速发展，许多公司投入到这一领域上来。ISS、axent、NFR、cisco等公司都推出了自己相应的产品（国内目前还没有成熟的产品出现）。但就目前而言，入侵检测系统还缺乏相应的标准。目前，试图对IDS进行标准化的工作有两个组织：IETF的IntrusionDetectionWorkingGroup(idwg)和CommonIntrusionDetectionFramework(CIDF)，但进展非常缓慢，尚没有被广泛接收的标准出台。

2、入侵检测系统模型

CommonIntrusionDetectionFramework(CIDF)（http://www.gidos.org/）阐述了一个入侵检测系统（IDS）的通用模型。它将一个入侵检测系统分为以下组件：

事件产生器（Eventgenerators）

事件分析器（Eventanalyzers

响应单元（Responseunits）

事件数据库（Eventdatabases）

CIDF将IDS需要分析的数据统称为事件（event），它可以是网络中的数据包，也可以是从系统日志等其他途径得到的信息。

事件产生器的目的是从整个计算环境中获得事件，并向系统的其他部分提供此事件。事件分析器分析得到的数据，并产生分析结果。响应单元则是对分析结果作

出作出反应的功能单元，它可以作出切断连接、改变文件属性等强烈反应，也可以只是简单的报警。事件数据库是存放各种中间和最终数据的地方的统称，它可以是复杂的数据库，也可以是简单的文本文件。

在这个模型中，前三者以程序的形式出现，而最后一个则往往是文件或数据流的形式。

在其他文章中，经常用数据采集部分、分析部分和控制台部分来分别代替事件产生器、事件分析器和响应单元这些术语。且常用日志来简单的指代事件数据库。如不特别指明，本文中两套术语意义相同。

3、IDS 分类

一般来说，入侵检测系统可分为主机型和网络型。

主机型入侵检测系统往往以系统日志、应用程序日志等作为数据源，当然也可以通过其他手段（如监督系统调用）从所在的主机收集信息进行分析。主机型入侵检测系统保护的一般是所在的系统。

网络型入侵检测系统的数据源则是网络上的数据包。往往将一台机子的网卡设于混杂模式（promiscmode），监听所有本网段内的数据包并进行判断。一般网络型入侵检测系统担负着保护整个网段的任务。

不难看出，网络型 IDS 的优点主要是简便：一个网段上只需安装一个或几个这样的系统，便可以监测整个网段的情况。且由于往往分出单独的计算机做这种应用，不会给运行关键业务的主机带来负载上的增加。但由于现在网络的日趋复杂和高速网络的普及，这种结构正受到越来越大的挑战。一个典型的例子便是交换式以太网。

而尽管主机型 IDS 的缺点显而易见：必须为不同平台开发不同的程序、增加系统负荷、所需安装数量众多

等，但是内在结构却没有任何束缚，同时可以利用操作系统本身提供的功能、并结合异常分析，更准确的报告攻击行为。

《NextGenerationIntrusionDetectioninHigh-SpeedNetworks》对此做了描述，感兴趣的读者可参看。

入侵检测系统的几个部件往往位于不同的主机上。一般来说会有三台机器，分别运行事件产生器、事件分析器和响应单元。HereLine 将前两者合在了一起，因此只需两台。在安装 IDS 的时候，关键是选择数据采集部分所在的位置，因为它决定了"事件"的可见度。

对于主机型 IDS，其数据采集部分当然位于其所监测的主机上。

对于网络型 IDS，其数据采集部分则有多种可能：

（1）如果网段用总线式的集线器相连，则可将其简单的接在集线器的一个端口上即可；

（2）对于交换式以太网交换机，问题则会变得复杂。由于交换机不采用共享媒质的办法，传统的采用一个 sniffer 来监听整个子网的办法不再可行。可解决的办法有：

A、交换机的核心芯片上一般有一个用于调试的端口（spanport），任何其他端口的进出信息都可从此得到。如果交换机厂商把此端口开放出来，用户可将 IDS 系统接到此端口上。

优点：无需改变 IDS 体系结构。

缺点：采用此端口会降低交换机性能。

B、把入侵检测系统放在交换机内部或防火墙内部等数据流的关键入口、出口。

优点：可得到几乎所有关键数据。

缺点：必须与其他厂商紧密合作，且会降低网络性能。

C、采用分接器（Tap），将其接在所有要监测的线路上。

优点：再不降低网络性能的前提下收集了所需的信息。

缺点：必须购买额外的设备(Tap)；若所保护的资源众多，IDS 必须配备众多网络接口。

D、可能唯一在理论上没有限制的办法就是采用主机型 IDS。

通信协议

IDS 系统组件之间需要通信，不同的厂商的 IDS 系统之间也需要通信。因此，定义统一的协议，使各部分能够根据协议所致订的的标准进行沟通是很有必要的。

IETF 目前有一个专门的小组

IntrusionDetectionWorkingGroup(idwg)负责定义这种通信格式，称作 IntrusionDetectionExchangeformat。目前只有相关的草案（internetdraft），并未形成正式的 RFC 文档。尽管如此，草案为 IDS 各部分之间甚至不同 IDS 系统之间的通信提供了一定的指引。

IAP(IntrusionAlertProtocol)是 idwg 制定的、运行于 TCP 之上的应用层协议，其设计在很大程度上参考了 HTTP，但补充了许多其他功能（如可从任意端发起连接，结合了加密、身份验证等）。对于 IAP 的具体实现，请参看《IntrusionAlertProtocol-IAP》，其中给出了非常详尽的说明。这里我们主要讨论一下设计一个入侵检测系统通信协议时应考虑的问题：

1、分析系统与控制系统之间传输的信息是非常重要

的信息，因此必须要保持数据的真实性和完整性。必须有一定的机制进行通信双方的身份验证和保密传输（同时防止主动和被动攻击）。

2、通信的双方均有可能因异常情况而导致通信中断，IDS 系统必须有额外措施保证系统正常工作。

4、入侵检测技术

对各种事件进行分析，从中发现违反安全策略的行为是入侵检测系统的核心功能。从技术上，入侵检测分为两类：一种基于标志（signature-based），另一种基于异常情况(anomaly-based)。

对于基于标识的检测技术来说，首先要定义违背安全策略的事件的特征，如网络数据包的某些头信息。检测主要判别这类特征是否在所收集到的数据中出现。此方法非常类似杀毒软件。

而基于异常的检测技术则是先定义一组系统"正常"情况的数值，如 CPU 利用率、内存利用率、文件校验和等（这类数据可以人为定义，也可以通过观察系统、并用统计的办法得出），然后将系统运行时的数值与所定义的"正常"情况比较，得出是否有被攻击的迹象。这种检测方式的核心在于如何定义所谓的"正常"情况。

两种检测技术的方法、所得出的结论有非常大的差异。基于异常的检测技术的核心是维护一个知识库。对于已知得攻击，它可以详细、准确的报告报告出攻击类型，但是对未知攻击却效果有限，而且知识库必须不断更新。基于异常的检测技术则无法准确判别出攻击的手法，但它可以（至少在理论上可以）判别更广范、甚至未发觉的攻击。

如果条件允许，两者结合的检测会达到更好的效果。

5、Linux 下的实现

5.1 系统框架

HereLine 是一个在 Linux 下运行的基于标识检测的网络 IDS。从逻辑上分为数据采集、数据分析和结果显示三部分，符合 CIDF 的规范。

从实现结构上看，HereLine 分成三个应用程序，它们分别是：

1、数据收集及分析程序(watcher)；

2、告警信息收集程序(listener)；

3、告警信息显示程序(console)。watcher 是数据采集和数据分析的合体；listener 接收 watcher 发出的告警信息，并将接到的信息存储为日志；console 为管理员提供了更友好的观察日志的图形界面。

同大多数商业 IDS 一样，HereLine 采用了分布式的结构。运行 HereLine 建议采用两台 PC 机，一台运行 watcher,另一台运行 listener 和 console.

与其他同类程序相比，HereLine 的优点主要有：

1、提供了完整的框架，可以灵活的应用于各种环境并扩充；

2、采用数据分析与告警程序分离，便于在大规模的网络环境下集中管理；

3、已经实现了对分片的处理，解决了利用分片逃避检查的问题；

4、数据分析部分不完全依赖已有攻击程序，而是分析其核心特征以察觉其变种攻击；但同时依靠已有攻击程序的细节来评估判别的准确性。以上设计增加了告警的可靠性。

各部分的实现流程及重要问题说明

5.1.1 数据采集部分

watcher 采用了 Linux2.2 内核中提供的 PF_PACKET 类型的 socket（并未采用 libpcap 提供的 API 接口）,直接从链路层获取数据帧。(直接采用操作系统提供的接口主要是考虑高效性，但为了将来的移植问题，以后仍然可能会改为使用 libpcap 库提供的函数接口。)根据 Linux 的要求,建立这样的一个 socket 需要 root 权限,即 uid=0。从 packetsocket 读到的数据是链路层格式的数据，但经过处理（socket 函数的第二个参量 SOCK_DGRAM 表示要去掉第二层的数据头，第三个参量 ETH_P_IP 表示只接收 ipv4 的数据包）后，缓冲区内的内容是个完整的 IP 包（未经任何其它处理）。分析工作交由数据分析程序去做。

数据采集部分还做了一项工作就是将网卡置于混杂模式，这样可以监听到整个网段的数据。

HereLine 的这种实现，实际是通过 socket 将数据拷贝到应用层。这种结构比较灵活与安全（应用程序崩溃不会导致系统崩溃),但频繁的应用态与核心态的转换浪费了 CPU。还有一种办法就是采用 Linux 中的模组（modular）的办法，将 IDS 作为内核的一部分。《BuildingIntoTheLinuxNetworkLayer》提供了一个内核中的 sniffer 的框架。可以利用其结构实现嵌入内核的、更加高效的入侵检测系统。HereLine 可以很容易的转移到这种方式，但考虑到其对操作系统稳定性的影响和调试的难度，目前未采用。

需要指出的是，watcher 只是监听数据包，并不参与操作系统协议栈的处理。如果操作系统被攻击导致拒绝服务，watcher 也将无法运行。因此，首先要保证起所运

行的系统是安全的。有些商业入侵检测系统（如 NFR）将数据采集部分放在专门的、经过改进的、高度安全的系统之上，以保证 IDS 这一系统中的重要程序正常工作。对于 HereLine 本身来说，建议采用增加了 stackguard 功能的 gcc 编译器，减少潜在的缓冲区溢出（bufferoverflow）漏洞。

watcher 将数据读到缓冲区之后，首先将其封装为 sbuff 结构，当数据在程序中传递时，均采用此结构。其定义为：

```
structsbuff
{
union{
structtcphdr*tcph;
structudphdr*udph;
structicmphdr*icmph;
structigmphdr*igmph;
}h;
union{
structiphdr*iph;
}nh;
unsignedchar*data;
};
```

sbuff 中定义了指向协议头的指针，子程序可根据这些指针快速的定位数据头位置。

5.1.2 数据分析

数据分析部分事实上与数据采集部分作为一个进程(watcher)存在,主要是为了简化程序设计的复杂性。在得到数据帧之后，watcher 模拟操作系统的 TCP/IP 堆栈对

数据进行处理并与已知攻击行为的特征进行比较，从中发现异常行为，并向控制台告警。

在分析的过程中，首先对数据包进行协议分析、过滤，根据协议、端口等信息将数据包送到不同的检测流程（如只有 HTTP 协议的包才进行 CGI 检查），这样使检查的过程尽量缩短，提高了程序的工作效率。当进入相应的检测流程后，则按照线性的顺序工作。

数据分析部分设计为模拟 TCP/IP 堆栈的流程处理数据包，从中发现异常行为（如分片重合、异常标志等）。这样主要是为了减少误报与漏报，从攻击的核心特征发现它，而不是象许多其他系统那样，只检查攻击包的表面特征。HereLine 中包括了 IP 分片处理，包括了 TCP 状态转换。通过这种结构，可以准确的判别诸如 teardrop 等碎片攻击的各种变种。

对一些常见攻击手法的描述和处理方法见附录 1。

HereLine 目前可以处理以下攻击：

1)land

2)winnuke

3)pingofdeath

4)sourcerouting

5)若干种 CGI 攻击

6)各种 OSdetection

7)各种类型的端口扫描

8)synflood

9)smurf

10)teardrop 及其所有变种

11)jolt

若识别出有异常行为，则向控制台告警（采用类似

IAP 的协议）。HereLine 此时分出一个单独的进程处理与控制台的交互，而原进程则继续处理新的数据包。

5.1.3 控制台

控制台部分分为两个程序，listener 和 console。listener 绑定 6543 端口（目前似乎没有和其他程序冲突），接收从分析程序发出的分析结果和其他信息，并根据其类型转化到不同文件存储。此时的文件为用户可读。其实 watcher 和 listener 两者便已经形成了一个完整的 IDS。

console 为一用 GTK+图形库编制的一个窗口程序，目的主要是给用户一个更方便友好的界面来浏览警告信息。

因为 listener 已经将报警信息以明文格式存储，因此也可以采用将其转换成 HTML 文件，用浏览器查看。当然也可以采用 windows 应用程序。因 HereLine 不是一个商业软件，对界面的考虑并不多。

HereLine 的日志采用了类似 UNIX 系统日志的格式，包括 time（事件发生时间）,host（被攻击的主机名或 IP）,watcher（发现此攻击的 watcher 的名字或 IP）,event（攻击的具体描述）这些项。各项之间用空格分开，每个记录用一行，每项之中不能有空格，若有，用下划线代替（以上语法约定均是为了方便处理）。下面是一个例子：

Feb_27_20:30:31172.18.172.18watcher01Port_Scan_From_evil.com

Feb_27_20:31:07web_serverwatcher02OS_fingerprint_from_12.18.1.1

在设计日志格式时我曾经考虑过许多方案，主要是想分出更多的项，如记录两端的端口信息，加入可信度