第 34 卷 第 11 期 计 算 机 工 程 2008 年 6 月
Vol.34 No.11 Computer Engineering June 2008
·安全技术· 文章编号：1000—3428(2008)11—0160—03 文献标识码：A 中图分类号：TP393.08

基于 NetChannel 技术的 UTM 架构设计

薛弘晔[1,2]，张 科[1]
(1. 西北工业大学航天学院，西安 710072；2. 西安科技大学计算机系，西安 710054)

摘 要：针对目前 UTM 产品应用层数据处理所导致的性能下降的问题，分析两种流行的 UTM 解决方案，使用 NetChannel 技术构建 UTM 体系架构，解决基于多核/多处理器下、UTM 体系架构中的网络性能瓶颈以及延迟增大的问题。对于解决应用层安全问题，提出一种新的架构设计。

关键词：防火墙；统一威胁管理；NetChannel 技术；多核/多处理器

UTM Architecture Design Based on NetChannel Technology

XUE Hong-ye[1,2], ZHANG Ke[1]
(1. College of Astronautics, Northwestern Polytechnical University, Xi'an 710072;
2. Dept. of Computer, Xi'an University of Science and Technology, Xi'an 710054)

【Abstract】 United Threat Management(UTM) uses many security technologies, however deep inspection for application layer leads to lost performance heavily and big latency, and it becomes a bottleneck for the whole network. This article analyzes two popular solutions, and gives a new architecture for UTM based on the NetChannel technology. The solution resolves the performance and latency issues in MultiCore/MultiProcessor architecture.

【Key words】 firewall; United Threat Management(UTM); NetChannel technology; multicore/multiprocessor

1 统一威胁管理技术

1.1 统一威胁管理的提出

目前，网络安全成为困扰网络发展的重要问题。防火墙技术解决了以太层和网络层等的安全问题，主要对网络层的信息进行检测，如 IP 地址、协议、端口进行规则设置，允许或者禁止数据包的通过，但不能过滤包含病毒、木马等存在潜在威胁的应用层数据。用户为了保护其 IT 资源必须购买防病毒、虚拟专用网络(VPN)、入侵监测、网络审计设备，同时由于大量设备存在，因此增加了网络管理的复杂性，降低了网络的可靠性与稳定性。然而，即使用户部署了安全设备，也不能做到万无一失。诸如不能避免类似尼姆达、冲击波等病毒的危害和攻击事件的发生，不同厂商缺乏沟通，不能有效地互动，从而降低了安全的保护力度。即使某些产品之间有互动机制，但由于各方面利益关系制约，产品之间耦合程度很低。因此，将众多安全技术集成在一个产品中，不但可以解决以上问题，也能适应用户的后继需求，而且减少了网络维护的复杂度，达到降低用户成本目的。

因此，IDC的Charles Kolodgy于 2004 年最早提出统一威胁管理概念。按照IDC的定义，UTM设备至少包含 3 种功能：防火墙，入侵防御，防病毒功能[1-5]。

1.2 UTM 发展现状

目前基于 TCP/IP 协议的网络所面临的各种安全威胁和安全需求有两方面：(1)网络层：针对协议缺陷的攻击行为，未授权访问；(2)应用层：用户应用数据安全，未授权访问。因此，IDC 的 UTM 概念，迅速被市场接受，并被扩展到以下方面：(1)防火墙；(2)入侵监测；(3)防病毒；(4)反垃圾邮件；(5)防窃听；(6)防网络钓鱼；(7)VPN。

UTM集多项安全技术于一身，2006 年由赛迪顾问发布的2005 年~2006 年中国内容网络安全网关报告显示，2005 年较 2004 年中国安全市场增长率 27.7%，达 44.61 亿元，防病毒、防火墙、IPS/IDS占整个市场 80%的份额，达 35.47 亿元[5]。对UTM产品来说，这是机遇也是挑战，其面临一个严峻的问题：对数据包的深度检测，即大量的加密/解密操作、实时内容分析与处理、特征匹配和数据包扫描、流量整形等过程处理，导致其性能急剧下降，使得UTM成为整个网络的瓶颈。据 2006 年 8 月份IDG的NetWorld与《网络世界》联合举行的UTM产品测试表明：当全部的UTM功能打开时，性能降低50%～60%，因此，性能问题成为制约UTM大量部署的重要原因之一。如何突破UTM性能问题是业界面临的一个重要课题。目前业界出现了 2 种不同的解决方案：(1)基于ASIC+X86架构+加速卡；(2)基于X86 架构的多核/多CPU+加速卡。在方案(1)中，安全厂商可以专门开发一种专用的硬件，加强特定的安全功能，但是，随着越来越多的安全应用的出现，专用硬件缺乏足够的灵活性和适应性，因此，ASIC架构特点是：高性能，大投入，不能及时适应新的业务需求；方案(2)的特点为高性能、易于升级，可以快速地适应各种新的安全需求。

2 NetChannl 技术

2.1 网络数据包处理流程

图 1 显示了 X86 架构，Linux 系统网络数据包的处理流程。在这个过程中，数据包至少经过 4 次拷贝才能被发送出，目前无论方案(1)或者方案(2)，具有类似的架构，不能减少数

基金项目：航空基础科学基金资助项目(04I53067)
作者简介：薛弘晔(1960－)，男，副教授、博士研究生，主研方向：实时计算机控制；张 科，教授、博士
收稿日期：2007-09-15 **E-mail：**xuehy60@163.com

据包拷贝次数。

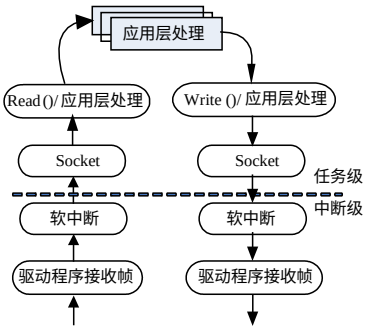

图 1　数据处理流程

2.2　TCP/IP 通信模型分析

在 2006 年澳大利亚国家Linux会议上，Van Jacobson结合TCP/IP设计思想和当前计算机的发展，分析了以Linux为代表的TCP/IP协议栈实现方式的优缺点[1]，提出了新的TCP/IP架构思想。

TCP/IP 协议是基于端到端的设计思想，因此，其通信过程可抽象为图 2。Linux 系统 TCP/IP 实现见图 3。

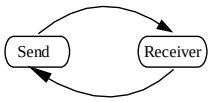

图 2　TCP/IP 通信模型

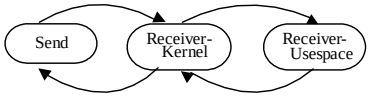

图 3　Linux 系统 TCP/IP 实现

在这个过程中，二端点构成一个通信环[3]，有较好的稳定性和可靠性。目前，存在以Linux系统为代表的TCP/IP协议栈的实现，整个通信系统可抽象为图 3 所示的结构。由 3 个端点构成二个通信环，接收端的系统内核协议栈首先接收到数据，然后经过处理后，再发送到上层应用程序。其优点是：基于TCP/IP协议栈的上层编程的人员不用关心TCP/IP协议本身，只需要使用Socket机制就可以完成通信功能。数据包首先送到接收端的内核空间，由内核空间完成各种解包/验证后再交给应用层。

这种协议栈架构模式屏蔽了协议细节，使得程序员可以把精力放在应用层数据的处理，但是引入的弊端[1]是：(1)一个通信环变为二个通信环，系统稳定性降低；(2)基于内核的实现机制，存在导致接收端RTT估算问题而引起假重传操作可能性；(3)基于内核的实现常常引来了额外的数据拷贝(驱动->skb_buff ->用户)、边界跨越(从硬件中断->软件中断->上下文交换->系统调用返回)；(4)基于内核的实现存在锁竞争和热点等问题。这些问题的存在将消耗大量资源，导致系统性能损失。

目前大多数TCP/IP的实现类似于Linux的协议栈实现方式，UTM在这种架构下，数据包经过如图 1 的各种流程后才能被送出，特别是多核技术下，由于大量的中断/锁处理、数据拷贝处理，系统开销很大，因此，Van Jacobson提出了NetChannel机制，旨在修正很久以前形成的错误概念，降低数据包从驱动到用户空间过程开销，从而达到较高的性能，同时降低协议栈实现中潜在的不稳定因素。NetChannel技术的思想就是将数据包从驱动高速的送到用户层，将TCP/IP协议栈移植为用户态的一个库函数调用，这样各个系统能并行处理，整个系统将具有非常高的效率，特别是在MultiCore/MultiProcessor的架构下，由于避免大量中断/锁的使用，可以大大提高网络的效率。实验数据显示，使用NetChannel机制要比传统的Socket机制在数据包性能方面快 2 倍左右，CPU 利用率下降了 2～3 倍，尤其在小包处理方面性能表现更明显[1]。

3　基于 NetChannel 的 UTM 结构设计

3.1　MultiCore/MultiProcessor 的体系架构分析

在UTM架构的 2 种方案中，业界普遍看好的是基于MultiCore/MultiProcessor的架构模式，具有可扩充、易更新、易移植等优点。在MultiCore/MultiProcessor的架构下，根据Amdahl定律：系统对某一部件采用某种更快执行方式所能获得的系统性能改进程序，取决于这种执行方式被使用的频率，或所占总执行时间的比例。系统容量遵循：$C(n)=an-bn^2$[1]，因此，随着Core/Processor数量的增加，性能增长接近线性，但系统开销则成平方增长。大量的资源共享与互斥、资源调度等问题，这些开销将导致系统的性能随着处理器的增多而失去增长的空间[4]。表 1 显示了在传统的X86 架构下，单处理器和多处理器在运行NetPerf时的资源使用情况，处理器数量的增加导致额外系统开销的增加[1]。

表 1　单处理器/多处理器资源对比

CPU	*Busy*	*Intr*	*Softint*	*Socket*	*Locks*	*Sched*	*App*
CPU1	50	7	11	16	8	5	1
CPU2	77	9	13	24	14	12	1

可以预见，当前 MultiCore/MultiProcessor 的方案并不能完全解决 UTM 性能和延迟问题。

3.2　基于 NetChannel 的 UTM 体系架构

MultiCore/MultiProcessor 架构存在一个缺陷：随着 Core/Processor 数量的增多，其并非带来系统性能的线性增长。这是因为随着处理单元的增加，大量的中断、锁机制、上下文交换等额外操作会消耗大量的资源，而且多次拷贝操作，尤其是在大量小字节数据包情况下，系统开销更明显。因此，借助 NetChannel 思想，以改善基于 MultiCore/Multi Processor 的 UTM 系统架构的性能瓶颈问题。

综合 UTM 的 7 大功能，结合 NetChannel 的特点，整个系统在数据层面分为：

(1)传统的以太层/网络层的数据处理。如 VPN/地址转换/路由等处理，这部分由于要求快速响应、用户透明的特点，可以在系统的内核空间完成。

(2)应用层数据部分。对于应用层数据，因为其要处理类似病毒检测/攻击检测/反垃圾检测等比较耗费资源且相对危险数据，所以这部分操作在用户空间完成；同时，考虑到不同用户需求，可以提供开放应用层接口，使用部分硬件完成比如模式匹配等工作。

基于内核的模块完成数据包分类、访问控制、VPN、地址转换、建立/销毁 SubChannel、建立/销毁防火墙会话表等操作。常驻系统的父进程 KernelRootThread 在每个网卡驱动程序注册 2 个 ID 为 0 的 RootChannel，分别为 Tx-RootChannel 和 Rcv-RootChannel，网卡接收到数据包首先根据数据包的五元组查找是否存在已注册 SubChannel，如果找到注册的 SubChannel，则将数据包 push 到指定的 SubChannel，由应用层处理，否则 push 到 RootChannel 中等待处理。系统框架如图 4 所示。

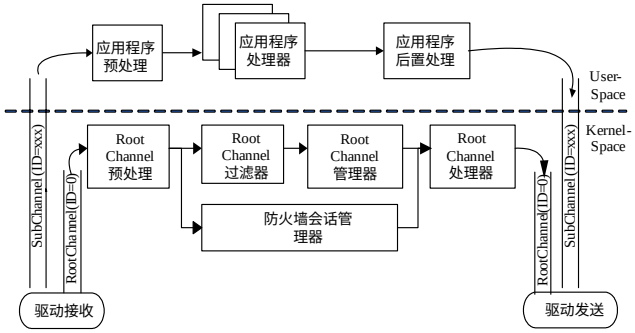

图 4 基于 NetChannel UTM 的架构

各个模块功能如下：

(1)RootChannel 预处理器。从 ID 为 0 的 Channel(即 RootChannel)中得到数据，然后进行以下处理：若此数据包是一个加密数据包，则进行解密；若是分片数据包，则进行分片重组；然后根据数据包的五元组信息(源 IP 地址、目标地址、协议、源端口、目标端口)在防火墙会话表中查找，若找到对应的会话条目，则把数据包交给防火墙会话管理器进行处理。否则数据包进入 RootChannel 过滤器。

(2)RootChannel 过滤器。根据用户的规则配置进行数据包的规则验证，对于非法数据包，将被丢弃。合法数据包将根据用户配置完成地址转化、路由查找等操作，建立防火墙会话表。

(3)RootChannel 管理器。根据 RootChannel 过滤器的输出，将建立从源接口到目的接口的 SubChannel，并把对应的防火墙会话信息加载到 SubChannel 上；如果数据包需要应用层处理，建立从驱动到应用层的 rcv-SubChannel 和 tx-SubChannel，这 2 个 Channel 共享同一个 ChannelID，Sub Channel 的 ID 为数据包的传输层信息，比如 80/TCP。

(4)NetChannel 处理器。根据防火墙的会话信息、用户配置信息、NetChannel 信息对数据包进行修改，并完成校验和等操作；如果是 VPN 数据包，则先作加密处理；如果数据包需要分片，则进行分片处理，然后把处理好的数据包 push 到 tx-RootChannel；后续沿着 SubChannel 到达应用层的数据包携带了对应的防火墙会话信息，这些数据包可能是加密的、分片的、需要 sequence 调整的数据包，因此，应用层从 Sub Channel 中 poll 到数据包后，需要完整数据包的校验、解密等操作，应用层各模块流程见步骤(5)~步骤(7)所述。

(5)应用程序预处理器：对于应用层数据，数据包由驱动直接到应用层。应用程序预处理器根据数据包和 SubChannel 附带的防火墙信息进行数据包验证，如果不匹配，则丢弃此数据包。之后，如果数据包是分片包，则进行分片重组。如果数据包乱序，则在完成排序等操作后，再进入应用程序处理模块。

(6)应用程序处理器：根据用户配置(包含在防火墙会话信息中)完成 URL 拦截、防病毒、反垃圾邮件、反网络钓鱼等操作，如果检测到数据包非法，则丢弃此数据包，释放内存。这部分可以使用外部的应用程序加速卡来加快处理速度。

(7)应用程序后置处理器：根据加载在 SubChannel 上的防火墙会话表信息，修改数据包的以太层数据包头、IP 数据包头、TCP/UDP 数据包头。如果是 VPN 数据包，加密发送。如果数据包需要分片，则进行分片处理，然后将这些数据包 push 到指定的 SubChannel 中去。

这种体系架构要求在系统的用户空间有一个 TCP/IP 协议栈，这可能产生一个潜在的安全威胁，但是，由于 UTM 的特殊性，因此这种威胁可能被降低。为完成应用层处理步骤，基于 Kernel 的 UTM 结构流程见图 5。

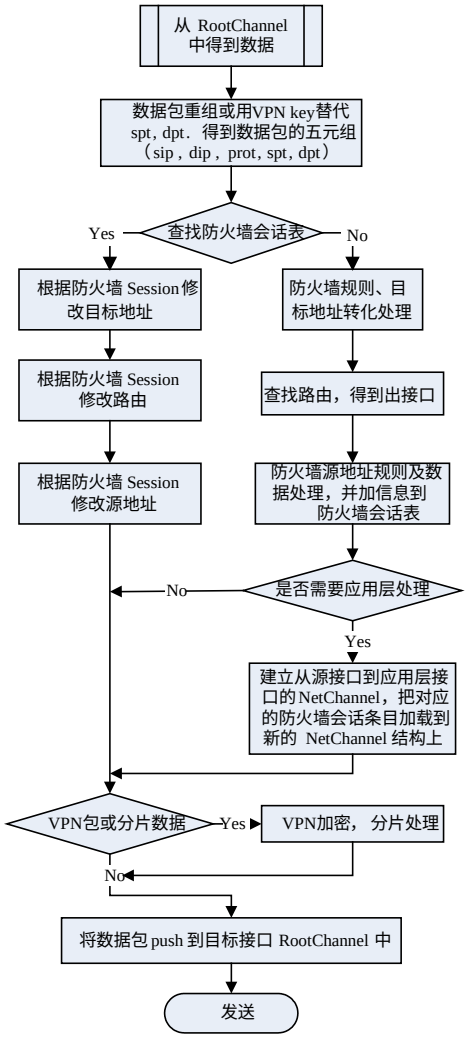

图 5 基于 Kernel 的 UTM 结构流程

4 结束语

将 NetChannel 的技术用于 UTM 体系结构设计，解决了当前 UTM 面临的性能问题的方案。在这个方案中，构建的 NetChannel 队列尽可能保证每个队列只有一个写入者，避免使用锁等耗费资源的操作，可以使用硬件加速卡来处理应用层数据的模式匹配、病毒检测等耗费资源的操作，这样可以得到一个更为高效的 UTM 系统。对于 MultiCore/MultiProcessor，可以采用优化的处理器分配算法，达到更高的性能。

参考文献

[1] Jacobson V. Speeding up Networking[Z]. 2006.

[2] Corbet J, Rubini A. Linux Device Driver[M]. Nanjing: Dongnan University Press, 2001.

[3] Richard S W. TCP/IP Illrustacted Volume 1[M]. Beijing: Machine Press, 2002: 156-180.

[4] IBM. SMP 工作负荷[EB/OL]. (2004-07-23). http://www-900.ibm.com/cn/support/viewdoc/detail? DocId= 2411084C28000.

[5] 赛迪顾问公司. 2005-2006 年中国内容安全网关市场调研报告[Z]. 2007: 4-5.