Improving Unified Threat Management Architecture Based on Net Channel Technology

Hu Hao
Jianghan University
Wuhan, China
E-mail: huhaohust@163.com

***Abstract*—In view of the performance reduction as a result of data processing of application layer of present Unified Threat Management (UTM) product, this paper analyses two popular UTM solutions. As a new architecture based on Multicore/Multiprocessor, NetChannel solves the network performance bottlenecks under the UTM system architecture. It is difficult to develop application layer protocol stack of UTM architecture. Therefore, this paper puts forward an improved NetChannel architecture—NetChannel plus. The new architecture makes full use of present new technologies of UTM software and hardware design to combine the advancement of NetChannel architecture with maturity of Linux protocol stack. The results of the experiments show that NetChannel plus significantly improves UTM performance and is a realizable NetChannel architecture.**

Keyword-UTM; netchannel technology; multicore processor

I. INTRODUCTION

In consideration of the performance, the traditional firewall protects the link layer, network layer and gives up the protection of application layer. With the development of network technology, in view of emergence of the network attacks of application layer, it is necessary to purchase firewall along with Anti-Virus (AV), Intrusion Prevention System (IPS), Content Filtering (CF) and other equipment at the same time to control the viruses, Trojans and other application layer attacks. However, coexistence of various safety equipments is bound to cause the complexity of network topology and network management. Due to lack of coordination and interoperability of equipments from different manufacturers, it often fails to provide strong and effective protection for the safety of network and data. In view of the increasingly serious network security problems, IDC puts forward the concept of UTM which not only protects the link layer and network layer but also protects the application layer. UTM fused many network security technologies on a device and simplifies the network structure and management of network security equipment.

Most of the Traditional firewalls adopt ASIC+X86CPU architecture. ASIC integrates many functions such as session, route & policy route, bridge and VPN. Comparing to x86 architecture, the packet can easily achieve the line speed. However, it is quite difficult for ASIC to integrate many complex functions such as AV and IPS and the risks are unable to control. Obviously it is more convenient to process CF, IPS and AV with CPU. With the popularity of Intel multi-core processor and PCI-E bus, more and more UTM adopt multi-core X86CPU architecture.

Multi-core architecture is extensible and easy to update and transplant. However, under the multi-core architecture, the sharing and mutual exclusion of resources, resource scheduling and such problems are becoming more and more complicated. UTM multi-core x 86 hardware platforms need better system architecture.

II. NETCHANNEL

Jacobson proposes the NetChannel architecture in 2006 annual meeting of the Linux and attributes poor performance to Linux kernel stack which arouses fierce argument in Linux online community.

A.NetChannel Architecture

NetChannel system framework (see Figure 1)

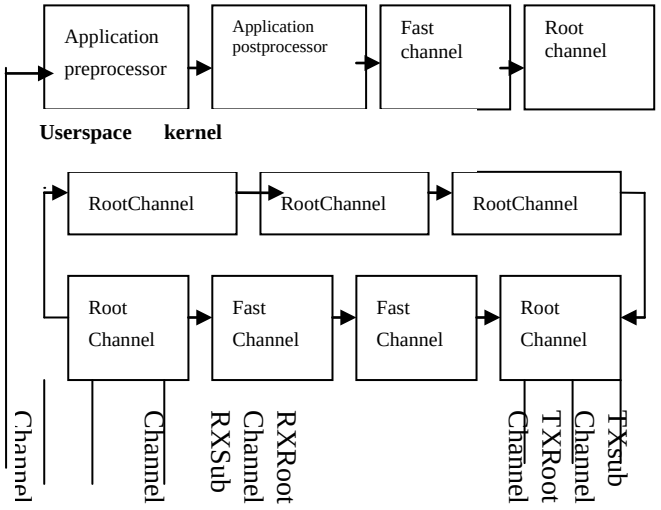

Figure 1 NetChannel system framework

NetChannel system mainly includes RootChannel and SubChannel. RootChannel processes Ethernet/Internet layer data while SubChannel processes

application layer. Application preprocessor and application postprocessor constitute the user-space protocol stack.

B. Advantages of NetChannel

Comparing to two communication rings of Linux, NetChannel applies point-to-point communication. Application data directly reaches application programs through SubChannel instead of Linux kernel stack. It has following advantages:

a. Without passing through Linux kernel stack it avoids copies from packet to skb and copies from skb to application layer which is the most radical zero copy technique.
b. Without passing through Linux kernel stack it will not process large numbers of shared data in Linux protocol stack thereby reducing the locks which benefits localization of cache.
c. If the packet is application data, raise software interrupt is not required. It reduces the number of system softirqs and context switches of system which benefits localization of cache.

C. Disadvantages of NetChannel

a. All the present mainstream protocol stacks are implemented in the kernel and the user-space protocol stack is difficult to achieve.
b. User-space protocol stack and Linux kernel stack coexist. It is hard to maintain the same function module as what user-space and kernel need to maintain such as application preprocessor and RootChannel preprocessor, application postprocessor and RootChannel postprocessor.

D. Theoretical Analysis of NetChannel

Amdahl's law is fundamental theorem of performance evaluation under MultiCore/ Multi Processor architecture. Amdahl's law points out that the system employs a certain faster execution on a component to achieve system performance improvement program which depends on the frequency of the execution, or the proportion that accounted for the total execution time. The capacity of the system follows: C(n)=a*n-b*n*n[5] in which n represents the number of Core/Processor, a*n represents that Performance growth is close to linear function, b*n*n represents the square growth of system overhead and b represents the order of severity of sharing and mutual exclusion and scheduling of resources. If b=0, then system capacity shows linear growth with the number of processors. Theoretically, without passing through Linux protocol stack NetChannel architecture avoids existing serialization of the Linux protocol stack and then reconstructs application protocol stack which may make b reach 0, but in fact application protocol stack is difficult to achieve. The NetChannel plus in this paper finds a new path. It does not blindly require that b equals to 0, but it constructs lightweight protocol stack (LWS) through Linux kernel and avoids serialization of Linux protocol stack as far as possible making b approach o. Although NetChannel plus has slight decline in performance, it greatly decreases the complexity of system. NetChannel plus combines with the new key technology of UTM hardware and software design making b constantly approach zero.

III. NETCHANNEL PLUS

Although NetChannel has many advantages, the Linux kernel stack after years of development has been very reliable and complete. Even Jacobson himself admits that the Linux kernel protocol stack is the fastest and most complete protocol stack. Separate development of user-space protocol stack has great risks so UTM designers would never adopt it. Based on the above consideration, an improved NetChannel architecture—NetChannel plus comes out.

A. NetChannel Plus Architecture

System framework (see Figure 2)

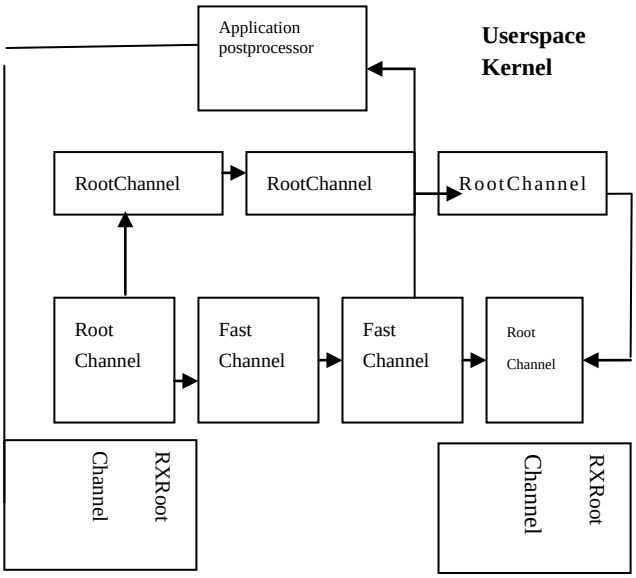

Figure 2 improved system architecture of NetChannel

NetChannel plus is divided into application layer, Ethernet/Internet layer while the latter includes slow forward layer and fast forward layer.

a. Ethernet/Internet layer, such as VPN, address resolution and routing. This part can accomplish in the kernel space because of the characteristic of quick response and users-transparency. Slow forward is mainly used to establish FastInfo which fast forward layer requires and send data while fast forward layer directly modifies packet to send data quickly mainly through FastInfo. NetChannel plus has better performance in Ethernet/Internet layer's data processing.
b. Application layer. As for data of application layer,

because it has to processes resource-consuming and relative dangerous data such as virus detection, attack detection and spam detection, this part will accomplish in users space. At the same time, considering the demands of various users, it can provide open application layer interface and use part hardware to accomplish the work such as pattern matching. Different from NetChannel, it needs to put packet into Rcv-Root Channel after application layer finishes processing data because application layer protocol stack has been canceled so as to pass through fast forward layer to accomplish data processing of Ethernet/Internet layer.

NetChannel plus replaces AppChannel with SubChannel. When slow forward layer finds application data through session table or fast forward directly finds application data through FastInfo, Ethernet/Internet layer will put data package into App Channel and then arouse application process.

B. NetChannel Plus Submodule

Modules based on the kernel accomplish the following operations such as packet classification, access control, VPN, address resolution, establishing/destroying Fast Info and firewall session table. System initialization is to register Tx-Root Channel, Rcv-Root Channel and App Channel for each network card driver.

The functions of each module are as follows:

a. RootChannel preprocessor; receive data from RootChannel and then continue the following processing: if this data package is encrypted, then decrypt it; if it is shard packets, restructure the fragmentation; if the application in the package has processing marks, take out hash from the package or calculate hash based on five information of data packets (source IP address, target address, protocol, source port and target port). Look up hash item in FastInfo. If the corresponding item is found, then hand the packet to the firewall FastForard manager for processing. Otherwise the data packets will be into RootChannel filter.

b. RootChannel filter; rules verification of data packets based on rules configuration of users; discard illegal data packets; legitimate packets will accomplish address resolution, routing lookup and set up a firewall session table according to the user configuration.

c. RootChannel manager; According to output of RootChannel filter, the Fast Info is established, and the corresponding firewall session information is loaded into the Fast Info; If packets need to be processed with the application layer without application processing tag, record the hash information on the package and send the package to AppChannel to process.

d. RootChannel preprocessor; according to the firewall session information and user configuration modify packets and accomplish checksum and etc.

e. RootChannel postprocessor; if it is VPN data packets, encrypt it first; if data packets need fragmentation, accomplish it then push the finished data packets to tx-Root Channel.

f. Fast Channel manager; if Fast Info indicates that packets need to be processed with the application layer without application processing tag, record the hash information on the package and send the package to AppChannel to process.

g. Fast Channel preprocessor; according to Fast Info, modify packets directly and accomplish checksum and etc.

h. Application processor; from poll in App Channel to data packets, according to users configuration (included in firewall session information), accomplish the operations such as AV, IPS and CF; if illegal packets are detected, discard them and free the memory. This part can use external application accelerator card to speed up the processing speed. Employ application processing standard in the package, then send the package to the rx – RootChannel

i.

C. NetChannel Plus Flow Chart

NetChannel flow chart (see Figure 3)

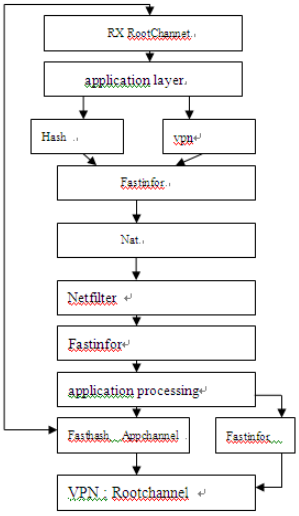

Figure 3 improved NetChannel flow chart

D. Implementation and Key Technology of NetChannel Plus

Slow forward layer is implemented with Linux netfilter while application is implemented with third-party software and hardware. The design of fast forward layer is the key to system performance. Due to FastInfo as the basis of processing package, fast forward can be designed as lightweight protocol stack (LWS), and by constructing hook point which is similar to the netfilter but highly

compact, fast foward can have open architecture for convenient integration of third-party software and hardware. Because the implementation of fast forward can make full use of and refer to the mature support function (e.g., calculate the checksum) and packet processing function that Linux protocol stack has already provided, the design and debugging of fast forward will become simple, and the application layer protocol stack of NetChannel will be struggling both in design and debugging.

The implementation of NetChannel plus is dependent on the following key technology:

a. Zero copy: Most internet cards support DMA mode to transmit package to memory [3] while the Linux memory map avoids the copy of the user space and kernel space.
b. Irq: NAPI solves the problem of multiple interrupts under high load [3] and reduces the context switches.
c. Soft irq: because the Linux ksoft_irq has lower scheduling priority [4], the context switches are reduced.
d. Lock: AppChannel using ring buffer structure does not need locks while fast forward is basically a read-only access and RCU read lock significantly improves performance.
e. Cache: most internet cards will adopt RSS and these belonging to the same flow packages will enter the same queue coupled with MSI - X and interrupt affinity technology. Flow package will be processed on the same core and the cache will implement the localization.

The above key technology shows that, although NetChannel plus still have two rings, the problems of zero copy, hardware interrupt, soft interrupt, lock and cache is not unsolvable.

IV. CONCLUSION

NetChannel plus combines the advancement of NetChannel architecture and maturity of Linux protocol stack, makes full use of the advanced technology of designs of UTM hardware and software, overcomes the defects with two communication rings, solves the problem of the implementation of NetChannel application protocol stack and provides a realizable scheme for UTM architecture design.

REFERENCE

[1] Xue Hongye, Zhang Ke. Based on the Technology of NetChannel UTM architecture design[J].Computer Engineering.2008(11).2

[2] Jacobson V. Speeding up Networking[Z]. 2006. 1-2

[3] Corbet J, Rubini A. Linux Device Driver[M]. Nanjing: Dongnan University Press, 2001.435-452

[4] Daniel P. Unstanding Linux kernel[M]. Beijing: Power Press, 2007. 154-181