行政院國家科學委員會補助專題研究計畫成果報告

網路安全與頻寬控制閘道器之實作與研究

Implementation and Research of Security and Bandwidth Management Gateways

計畫類別： 個別型計畫 整合型計畫

計畫編號：NSC 90-2213-E-009-161-

執行期間： 2001 年 08 月 01 日至 2002 年 7 月 31 日

計畫主持人： 林盈達

共同主持人：

計畫參與人員： 魏煥雲、梁元彪、張智晴

本成果報告包括以下應繳交之附件：

赴國外出差或研習心得報告一份

赴大陸地區出差或研習心得報告一份

出席國際學術會議心得報告及發表之論文各一份

國際合作研究計畫國外研究報告書一份

執行單位： 國立交通大學資訊科學系

1. Abstract

中文摘要：
網路安全和頻寬管理對企業而言已經成為一個重要的議題。本研究著重在整合與安全及頻寬管理相關的開放性原始碼套件在單一個閘道器上。各套件彼此的衝突已被消弭以確保其互通性。其次，這項研究並從內部及外部量測各功能元件的效能，以發現更進一步的研究議題。我們也提出了一個 PostACK 的方式來管理 TCP 流量，在一個會損失封包的廣域網路下較 TCR 的方式節省了 96%的緩衝區空間，及提高 10%的實質輸送量。最後，我們並提供幾種提升各元件效能方法的建議。

English abstract:
Network security and bandwidth management have become a critical issue for enterprises. The research focuses on integrating components of open source packages for security and bandwidth management into a single gateway. Conflicts among the packages are resolved to ensure interoperability. Next, this study internally and externally evaluates the performance of each component to identify the problems for future research directions. We also propose a PostACK to shape TCP flows, which saves 96% buffer space but also has 10% goodput improvement against TCP rate control (TCR) [1] under a lossy WAN in our implementation. Finally, several approaches to scale up these software components are suggested to improve the performance.

2. Motivation and introduction

To guarantee the secure operations of an enterprise network, firewalls, virtual private networks (VPN) and intrusion detection systems (IDS), respectively, address the requirement. Originally they were separate devices, but they have been integrated with other services such a network address translation (NAT) as a single security gateway. Besides, bandwidth management is an increasing demand. It is expected to be integrated into the gateway as well. We integrate several open source packages for these purposes into a single gateway. These packages are listed in Table 1 [2-7].

Package Name	User-space Program	Kernel-space Program	Package Size	Version
ipchains	Command-line management tool	Kernel built-in packet filtering firewall and NAT	63KB	1.3.9
Squid	Daemon (Cache server, transparent proxy**, and URL filter)	No	1104KB	2.3
FWTK	Daemon (Application proxies for Web content filter)	No	476KB	2.1
FreeS/WAN	Pluto Daemon (Internet key exchange, IKE)	KLIPS kernel patch (Encryption and authentication)	1252KB	1.5
Snort	Daemon (Intrusion detection)	No	644KB	1.7
ALTQ	Management tools	QoS components	507 KB	3.1

Table 1 Software packages information

Figure 1 illustrates the complete packet processing flows of the security part of the integrated gateway.

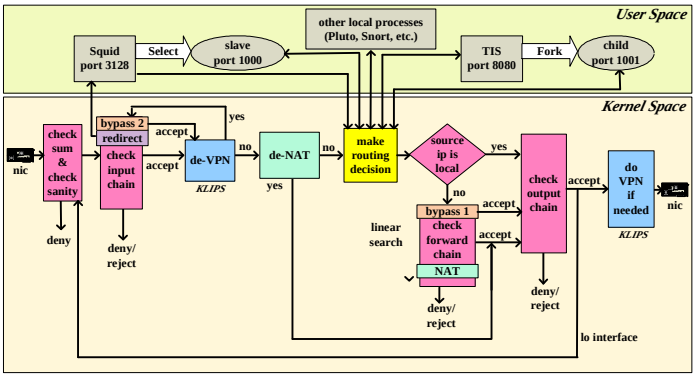

Figure 1: Complete Packet Processing Flows of the security Integration

In general, the packet flow from the private interface to the public interface is as follows: (1) perform checksum calculations and sanity checks of the packet; (2) check the packet through the input chain; (3) routes the packet; (4) judge whether the packet is generated by the gateway itself; (5) check the packet through the forward chain; (6) NAT the packet if needed; (7) check the packet through the output chain, and (8) perform VPN processing if needed. In addition, if the packet's source and destination IP addresses both belong to the gateway, i.e., inter-daemon communication from Squid to FWTK, the gateway passes the packet back to the loopback (lo) interface after checking the output chain.

We also propose PostACK to manage TCP traffic. It is designed to be more intelligent both in retaining previous TCR benefits and eliminating its deficiencies. Without measuring the WAN delay and shrinking the

RWND in TCP ACKs, PostAck can avoid the side effects of TCR. It has the following advantages: (1) low buffer requirement, (2) low data packet latency, (3) fairness among flows within a class, (4) higher goodput than TCR-applied flows, (5) more robust under various TCP implementations, (6) eliminating triple-dup ACKs. Figure 2 illustrtates theis mechanism.

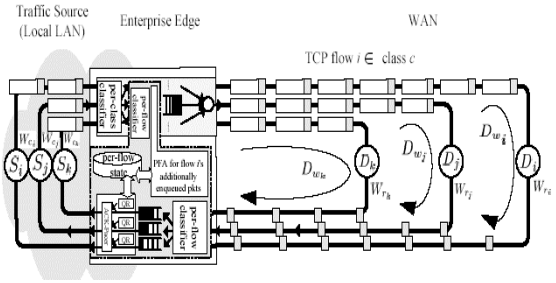

Figure 2: Efficient PostAck implementation for managing outgoing TCP traffic

3. Result and discussion

We perform both internal and external benchmarks to our implementation. Table 2 summarizes the benchmark methodology.

Category	Benchmark Tools	Settings	Benchmark Items
CPU cost	SmartBits 2000 with SmartFlows 1.2, self-written HTTP Traffic Generator	1. Enable all security functions 2. No other filters in ipchains	Who tops the processing time among all the functions
Memory and disk cost	SmartBits 2000 with SmartFlows 1.2, self-written HTTP Traffic Generator	3. Using 3DES / MD5 4. 10 URL entries are configured 5. A single HTTP connection repeatedly retrieves a 40KB Web page for 10 seconds	The disk / memory consumptions
Packet Filter	SmartBits 2000 with SmartFlows 1.2	1. Only enable packet filter 2. Various numbers of filters 3. Various packet sizes	Scalability
URL Filiter	self-written HTTP Traffic Generator	1. Various Web page sizes 2. Various URL lengths in HTTP request 3. A single HTTP connection repeatedly retrieves a 64KB Web page for 10 seconds	Scalability
Content Filter	HTTP Traffic Generator	1. Various Web page sizes 2. Various numbers of concurrent connections 3. A single HTTP connection repeatedly retrieves a 64KB Web page for 10 seconds	Scalability
IP Masquerade	SmartBits 2000 with SmartFlows 1.2	1. Security gateway equipped with 4 NICs. 1 for public interface and 3 for private interfaces 2. Various numbers of private hosts 3. Various packet sizes	Scalability
Authentication Algorithms	SmartBits 2000 with SmartFlows 1.2	Various packet sizes	Cost of MD5 and SHA1
IDS	SmartBits 2000 with SmartFlows 1.2	Various packet sizes	1. Packet loss rate 2. pattern-matching time

Table 2: Benchmark methodology

Table 3 summarizes the observations of the benchmark results. It indicates that ipchains and FreeS/WAN are more viable than commercial products, but FWTK and Snort have performance problems.

Module	Characteristics	Bottleneck	Reason	Worst-case Time Complexity
ipchains	CPU-intensive	Increasing the number of filters	Linear matching algorithm	O(l+m+n); l, m, n: number of filters in input, forward, and output chains, respectively
Squid	Memory& CPU-intensive	Increasing the number of URL regular expressions	Linear matching algorithm	O(n(l+m)); l: URL length in HTTP requests; m: average regular expression length; n: number of URL regular expressions
FWTK	CPU-intensive	Increasing the number of HTTP connections and the size of the retrieved Web page	1. Parse config file for each request 2. Read only one byte of the Web page from the socket interface at a time	O(n); n: size of the retrieved Web page
NAT	CPU-intensive	Increasing the number of private-to-public connections	Data structure of NAT table	O(n); n: number of private-to-public connections
FreeS/WAN	CPU-intensive	Using the stronger algorithms	Too many computation for encryption and authentication	O(n*m); m: key length; n: packet size
Snort	CPU-intensive	Packet loss frequently	1. Copy each packet from kernel space to user space 2. Linear matching algorithm	O(l+m*n); l: number of TCP/UDP/ICMP rule tree nodes; m: number of TCP/UDP/ICMP rule options; n: packet size

Table 3: Summary of comparisons

For bandwidth management, we compare the pros and cons of three mechanisms, Per-flow Queuing (PFQ), TCR, and PostAck, as well as their complexity. The results are summarized in Table 4 and 5.

Metrics	PFQ	TCR	PostACK
Goodput (under lossy WAN)	High	Slightly Lower	High
Fairness (fine-grained)	Good	Good	Slightly Lower
Fairness (under lossy WAN)	Good	Degraded	Similar to PFQ
Buffer Requirement	High	Low	Low
Data Packet Latency	Large	Little	Little
ACK Packet Latency	Little	Little	Large
Robustness (under lossy WAN)	High	Poor	High
Stability (under lossy WAN)	High	Slightly Lower	High

Table 4: Comparison of PFQ, TCR, and POSTACK: performance metrics

Complexity PFQ	TCR	PostACK
Classification	per-flow	per-flow	per-flow
Time	$O(1)$	$O(1)$	$O(1)$
Space	$O(N)$	$O(N)$	$O(N)$
RTT Measurement	No	Yes	No
Header Modification	No	Yes	No
Checksum Recalculation	No	Yes	No

Table 5: Comparison of PFQ, TCR, and PostACK: complexity

The following improvements are suggested to scale up the security packages:
(1) Improving the linear-time algorithms

For ipchains, Squid, and Snort, their linear matching algorithms can be accompanied by a flow cache so that active flows can follow a fast path. For example, signature-based IDS such as Snort, most of the signatures (545 of 763 signatures) attack Web servers. By carefully caching the valid URLs, normal URL accesses can bypass the long linear-matching phase of URL-related signatures.

(2) Proper implementation tricks

For FWTK, the configuration file should be scanned only once, and the retrieved Web page can be read multiple bytes at once from the kernel space to the user space. For the NAT module, a suitable bucket size for large enterprises can be defined to avoid hash collisions.

(3) Function relocation from daemon to kernel

For some advanced access control policy used in the application proxy, such as FTP, SMTP proxy in FWTK, only the control-plane parts are required to be directed to the user-space daemon process for checking. Other data-plane objects should pass directly through the kernel or be blocked, according to the access-control policy. Several works have focused on changing the slow kernel-daemon-kernel data path into a fast kernel data path [8,9,10]. The efforts differ primarily in the flexibility to switch between slow and fast data paths. Numerical results indicate that this pure software-based acceleration of application proxies can improve the performance by a factor of two to four.

(4) Hardware accelerators

For encryption/decryption operations in VPN processing such as that in FreeS/WAN, the 3-DES operations can be offloaded to an accelerator card or ASIC. Typical operations are: (1) formatting the data to be encrypted/decrypted; (2) feeding data to the hardware through programming I/O or DMA channels; (3) waiting for a hardware interrupt to trigger the Interrupt Service Routine (ISR) to check what is happening; (4) finding that the hardware has successfully encrypted/decrypted, and (5) continuing to process the subsequent operations.

(5) PostACK

PostACK can also achieve perfect fairness, as PFQ and TCR can, if the measuring time scale lasts for several RTTs. But if we measure the bandwidth with a very fine-grained time scale, PostACK's fairness is slightly degraded. Honestly speaking, ACK control has always been a cool hack, but not a deep solution. Our study provides a big picture of how much we can shape TCP traffic transparently, especially in the lossy WAN environments.

4. Self-evaluation of the result

Integrating many functions into a single all-in-one system or separating them as standalone devices involves security and bandwidth management issues. Many commercial security gateways choose to be an all-in-one solution. Accordingly, this study focuses only (1) building a product-like all-in-one system from numerous open-source packages and on (2) externally and internally evaluating the performance of such system. However, installing such a device does not mean secured. Other issues, such as correctly setting the administrative policy rules, increasing the security of the network architecture, and increasing the security of the encryption algorithms, are beyond the scope of this study and deserve further attention. The highly integrated system presented here, together with the self-developed Web management console, is downloadable at [11] for hands-on practice.

5. References

[1] S. Karandikar, S. Kalyanaraman, P. Bagal, and B. Packer, *TCP Rate Control*, ACM Computer Communication Review, Vol. 30, No. 1, Jan. 2000.

[2] *ipchains, http://www.netfilter.org/ipchains/.*

[3] *Squid, http://www.squid-cache.org .*
[4] *FWTK, http://www.fwtk.org .*
[5] *FreeS/WAN, http://www.freeswan.org .*
[6] *Snort, http://www.snort.org .*
[7]*ALTQ, http://www.csl.sony.co.jp/person/kjc/programs.html.*
[8] David Maltz, "TCP Splicing for Application Layer Proxy Performance," *IBM Research Report*, March 1998.
[9] Oliver Spatscheck, Jorgen S. Hansen, John H. Hartman, Larry L. Peterson, "Optimizing TCP Forwarder Performance," *IEEE/ACM Transactions on Networking*, Vol.8 No.2, April 2000.
[10] Saibal Kumar Adhya, "Asymmetric TCP Splice: A Kernel Mechanism to Increase the Flexibility of TCP Splice," *Master thesis of Dept. of C.S.*, Indian Institute of Technology, April 2001.
[11] *"Integrated security gateway," http://speed.cis.nctu.edu.tw/SG.html.*