Design, Implementation and Performance Evaluation of IP-VPN

Jin-Cherng Lin , Ching-Tien Chang and Wei-Tao Chung
Dept. of Computer Science and Engineering Tatung University
Email:jclin@cse.ttu.edu.tw

ABSTRACT

Network security has always been a significant issue, but a recognized priority today due to the popular of internet. The issue is not if security should be implemented on a network; rather, the question to ask is if security has been implemented properly and the interoperability with today's network architecture. Although there are various ways to perform a secure network environment, but the most popular and the most progressive network security mechanism is Security Architecture for IP (IPSec), offered by IETF (Internet Engineering Task Force).

In this paper, we will discuss the problems when combine IPSec into current TCP/IP module by porting an IPSec shareware (FreeS/WAN) into a router. Finally, in order to understand the impact on router's performance when using various services and hash/encryption algorithms provided by IPSec, we testing the throughput of the router before and after applying IPSec.

1. Introduction

Two major elements are necessary to construct a VPN: a tunneling protocol and a means to authenticate that tunnel origin. Tunneling is a method for sending data packets securely over the Internet or other public network [Youn00]. The most popular VPN protocols is IPSec (Internet Protocol Security) currently. IPSec is a collection of protocols, authentication and encryption mechanisms. It is an extension to the standard IP protocols. In addition, the IPSec packet may also have an authentication header, which authenticates the validity of the entire IPSec packet. This enables the receiver to verify that the packet has not been modified en route [Youn00].

IPSec is a Layer 3 protocol standard designed as an end-to-end mechanism for ensuring data security in IP based communications. IPSec allows IP payloads to be encrypted and encapsulated in an IP header for secure transfer across the Internet (or a corporate IP inter-network) [Youn00].

This work was supported in part by NSC under Grant NSC90-2623-7-036-002- and by Tatung University under Grant B90-1600-04

2. IPSec Implementation

2.1 The Architecture of IPSec Software Module

Our IPSec module interfaces to the host's IP protocol stack are on IP packet basis. It provides a set of APIs to interface with key management protocols such as Internet Key Exchange (IKE) mechanism. It also provides a set of APIs to configure and manage security policies and system preferences. Generally, as Figure 1 shows, the IPSec modules and the modules of IKE protocol are dependent on each other. However, it is possible to apply IPSec modules independently. IPSec module can also be used with another key management protocol.

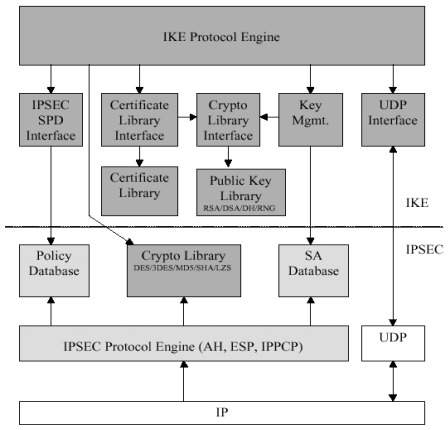

Figure 1: IPSec and IKE Architecture

2.2 Implementing IPSec

The file ip_output.c contains many subroutines for processing outgoing packets. There are three subroutines which are directly called by upper layer to send out the packets:

- ip_queue_xmit(): Queues a packet to be sent, and starts the transmitter if necessary. This routine also put the total length and computes the checksum.
- ip_build_xmit(): This subroutine is a faster way to send ICMP and UDP packets while the packet does not need to perform fragmentation.
- ip_build_xmit_slow(): This function is only called by ip_build_xmit() when the packet need to be fragmented.

In vLinux, the main receive routine in IP layer is

Proceedings of the17 th International Conference on Advanced Information Networking and Applications (AINA'03)
0-7695-1906-7/03 $17.00 © 2003 IEEE

ip_rcv() in ip_input.c. While receiving a packet this routine:

1. Check the packet if it is for this host.
2. Check the packet length at least greater than the size of IP header.
3. Check the version field in IP header.
4. Check the checksum.
5. De-fragment the packets if necessary.
6. Verify against the firewall rules (if any).
7. Process optional fields in IP header.
8. Deliver the packet to upper layer.

2.2.1 vLinux Firewall Process for Input, Output and Forward Packets

There are three types of firewall checkings in vLinux (kernel 2.2.14) IP layer:

- Input firewall checking,
- Forward firewall checking, and
- Output firewall checking.

The function registered for the input firewall checking will be called immediately after receiving the packet from lower layer. The ip_rcv() in the vLinux IP layer functions calls call_in_firewall(), which will call the user-defined registered input firewall checking function. We then implement our IPSec functions in the registered input firewall checking function.

The forward firewall checking and output firewall checking are processed in the same way.

2.2.2 IPSec Process for Input, Output and Forward Packets

We define three IPSec functions: ipsec_input_check(), ipsec_forward_check(), and ipsec_output_check(). We use register_firewall() in vLinux to register our IPSec processing routines (see Figure 2 and Figure 3).

vLinux stack does not expect the user-defined firewall checking routines to send out the packet directly. It simply expects to receive a YES/NO return value. However, for some kinds of reasons, we send out the packet directly in our IPSec module and do not use the vLinux TCP/IP stack process. We thus return a FW_QUEUE value from our IPSec checking routines since we do not want the vLinux TCP/IP stack to send out the packet. Besides, when we send the packet directly the sending routine frees the native buffer (skbuff) and it should not be freed again in the vLinux IP stack process after returning from our IPSec checking routines. To avoid this double freeing we change skbuff pointer to NULL before returning. This necessitates some modification in the vLinux TCP/IP stack. In the inline function kfree_skb() we use one NULL check before freeing the memory.

```
struct firewall_ops
{
 struct firewall_ops *next;
 int (*fw_forward)(struct firewall_ops *this, int pf,
      struct device *dev, void *phdr, void *arg, struct
sk_buff **pskb);
 int (*fw_input)(struct firewall_ops *this, int pf,
      struct device *dev, void *phdr, void *arg, struct
sk_buff **pskb);
 int (*fw_output)(struct firewall_ops *this, int pf,
      struct device *dev, void *phdr, void *arg, struct
sk_buff **pskb);
 int fw_pf;          /* Protocol family
      */
 int fw_priority;   /* Priority of chosen firewalls   */
};
```

Figure 2: firewall_ops structure

```
struct firewall_ops ipsec_ops =
{
 NULL,
 ipsec_forward_check,
 ipsec_input_check,
 ipsec_output_check,
 PF_INET,
 5
};
register_firewall(PF_INET,&ipsec_ops)
```

Figure 3: Register IPSec process

IPSec processing is not required for forwarding packets, since all jobs are done in ipsec_input_check() and ipsec_output_check(). So ipsec_forward_check() is a dummy function.

3. Performance Evaluation

3.1 Testing Environment

Two VPN-routers, VPN-router(A) and VPN-router(B), are used to build a security tunnel. A PC, PC-2, is with FTP and HTTP servers, while another PC, PC-1, uses FTP client and browser to download various files from PC-2 via IPSec tunnel. The hardware environment is configured as shown in Figure 4. The specification of each hardware device is shown as follows:

PC-1: CPU: PIII 850

- OS: Windows 2000 with service pack 2
- NIC: SiS 900 (NIC driver version: 1.14.1.0)

PC-2: CPU: PIII 850

- OS: Windows 2000 with service pack 2
- NIC: Intel(R) PRO/100 VE (NIC driver version: 4.3.25.0)
- FTP server: Microsoft FTP Service Version 5.0

Proceedings of the17 th International Conference on Advanced Information Networking and Applications (AINA'03)
0-7695-1906-7/03 $17.00 © 2003 IEEE

- HTTP server: Microsoft IIS 5.0
- VPN-Router(A), VPN-Router(B):
- CPU: SAMSUNG S3C4510 (ARM7TDMI)
- OS: Vitals System Inc. vLinux kernel 2.2.14 with IPSec module
- LAN port NIC: ARM7 built-in NIC 100/10Mbps (NIC driver: vLinux built-in driver)
- WAN port NIC: Realtek 8019AS (10BaseT) (NIC driver: vLinux built-in driver)

Laptop: CPU: PIII 450

- OS: Windows 2000 with service pack 2
- NIC: Billionton-LNR100B2 (NIC driver version: 5.374.303.2000)
- Packet monitor software: NAI Sniffer pro 4.5

Hub: Generic10/100 dual speed hub

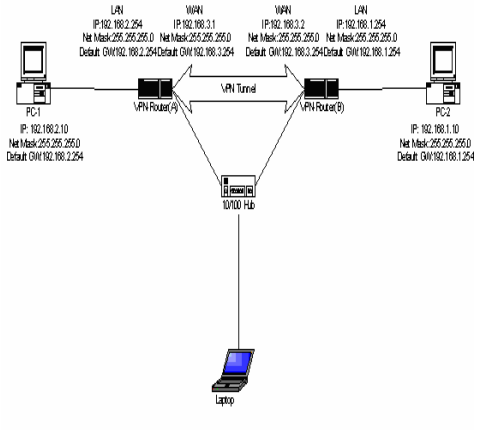

Figure 4: Testing Environment

3.2 Experimental Objectives

We use FTP and HTTP protocols to perform data download for various size files (1MB, 10MB, 100MB) from PC-2 to PC-1 passing through VPN-router(A) and VPN-router(B) using tunnel-mode and manual key management method. Various security protocols AH and ESP with various authentication and encryption algorithms listed belows are tested to verify their performance:

- ●AH-MD5
- ●AH-SHA1
- ●ESP-NULL-MD5
- ●ESP-NULL-SHA1
- ●ESP-DES-NULL
- ●ESP-3DES-NULL
- ●ESP-DES-MD5
- ●ESP-DES-SHA1
- ●ESP-3DES-MD5
- ●ESP-3DES-SHA1

3.3 Experimental Data for Performance Evaluation

We use FTP client and HTTP browser to download file 20 times in each condition and calculate the average throughput (Kbytes per second). The experimental data are listed in Table 1 and Table 2.

Size / Algorithms	1MB		10MB		100MB	
	KB/s	Efficiency(%)	KB/s	Efficiency(%)	KB/s	Efficiency(%)
None	700.50	100	658.53	100	633.32	100
AH-MD5	288.22	41.14	221.72	33.67	199.94	31.57
AH-SHA1	189.86	27.10	151.33	22.98	141.10	22.28
ESP-Null-MD5	271.93	38.82	218.58	33.19	203.71	32.17
ESP-Null-SHA1	201.34	28.74	163.28	24.79	147.62	23.31
ESP-DES-Null	107.09	15.29	97.65	14.83	96.83	15.29
ESP-3DES-Null	77.55	11.07	73.04	11.09	67.35	10.63
ESP-DES-MD5	107.50	15.35	91.35	13.87	82.63	13.05
ESP-DES-SHA1	87.33	12.47	77.98	11.84	70.56	11.14
ESP-3DES-MD5	73.88	10.55	71.01	10.78	64.10	10.12
ESP-3DES-SHA1	67.14	9.58	64.08	9.73	56.81	8.97

Table 1: The average throughput of the security gateway for various security protocol and authentication/encryption algorithm combination using FTP protocol.

Size / Algorithms	1MB		10MB		100MB	
	KB/s	Efficiency(%)	KB/s	Efficiency(%)	KB/s	Efficiency(%)
None	682.67	100	635.38	100	616.86	100
AH-MD5	291.84	42.75	219.71	34.58	193.55	31.38
AH-SHA1	188.22	27.57	146.08	23.00	133.23	21.60
ESP-Null-MD5	279.04	40.87	197.60	31.10	191.58	31.06
ESP-Null-SHA1	195.54	28.64	156.73	24.67	134.32	21.74
ESP-DES-Null	105.35	15.43	96.18	15.14	92.51	15.00
ESP-3DES-Null	78.49	11.50	72.68	11.44	64.41	10.44
ESP-DES-MD5	99.22	14.53	84.58	13.31	80.534	13.06
ESP-DES-SHA1	90.72	13.29	75.40	11.87	68.40	11.09
ESP-3DES-MD5	71.81	10.52	67.30	10.59	52.51	8.51
ESP-3DES-SHA1	65.12	9.54	62.78	9.88	52.38	8.49

Table 2: The average throughput of the security gateway for various security protocol and authentication /encryption algorithm combination using HTTP protocol.

3.4 Data Analysis

3.4.1 Comparison of MD5 and SHA-1 [Stal99]

First, we can see the obviously different performance between AH-MD5 and AH-SHA-1. Because both SHA-1 and MD5 are derived from MD4 algorithm, they are quite similar to each other. Accordingly, their strengths and other characteristics should be similar. We can compare the two algorithms as belowing aspects:

- Security against brute-force attacks: The most

Proceedings of the17 th International Conference on Advanced Information Networking and Applications (AINA'03)
0-7695-1906-7/03 $17.00 © 2003 IEEE

obvious and most important difference is that the SHA-1 digest is 32 bits longer than the MD5 digest. Using a brute-force technique, the difficulty of producing any message having a given message digest is on the order of 2^{128} operations for MD5 and 2^{160} for SHA-1. Again, using a brute-force technique, the difficulty of producing two messages having the same message digest is on the order of 2^{64} operations for MD5 and 2^{80} for SHA-1. Thus, SHA-1 is considerably stronger against brute-force attacks.

- Security against cryptanalysis: MD5 is vulnerable to cryptanalytic attacks discovered since its design [RFC1321]. SHA-1 appears not to be vulnerable to such attacks. However, little is publicly known about the design criteria for SHA-1, so its strength is more difficult to judge than would otherwise be the case.
- Speed: Because both algorithms rely heavily on addition modulo 2^{32}, both do well on a 32-bit architecture. SHA-1 involves more steps (80 versus 64) and must process a 160-bit buffer compared to MD5's 128-bit buffer. Thus, SHA-1 should execute more slowly than MD5 on the same hardware.

Because of these reasons, we can realize why using SHA-1 digest is much slower than MD5 without respect to the security protocol being AH or ESP.

3.4.2 Comparison of AH and ESP using the same authentication algorithms

Nowadays some people claim to abrogate the AH protocol, since ESP can support all the services those AH can provides. We can see the problem from the viewpoint of throughput. Although the AH header is shorter than the ESP header, AH have to calculate the digest of longer data (including new IP header field) in the tunnel mode. In Table 1 and Table 2, we can see clearly the AH and ESP (with null encryption) using the same authentication algorithms to download files to result in nearly equal throughput. It violates the generic intuition of AH is simple such that AH should have higher throughput than ESP.

3.4.3 Comparison of IPSec throughput via FTP and HTTP

IPSec is working on ISO/OSI network layer 3, this means its throughput will not be affected by upper layer protocol and data. In other words, IPSec protocol does not care about what its upper layer protocols and data are. The data type of payload data (i.e., upper layer protocols headers plus application data) will not have any influence on IPSec's throughput.

We can compare the throughput data shown in Table 1 and Table 2. The HTTP is generally slower than FTP. The throughputs between HTTP and FTP after applying IPSec are still keeping this gap although they are quite close in every experimental case.

4. Conclusion

In this article we discuss:

- how to implement IPSec module on vLinux kernel,
- the throughput after applying IPSec, and
- the difference of the throughputs between HTTP and FTP after applying IPSec.

Implementing IPSec on a gateway (e.g., router) is a good solution for existing enterprise LAN network. It is not necessary to change the original LAN architecture. It is only to replace the original gateway device by a VPN-gateway. All PC's in the enterprise LAN do not need to be changed or reconfigured. The VPN functions are only handled on the VPN-gateway. The PCs' users in the enterprise do not need to have any VPN knowledge and skill. Only one person is involved to manage the enterprise-wide VPN functions on the gateway. It is easy to be managed by the system administrator. The entire enterprise utilizes the advantage of VPN gateway, but no complex training and costly devices/package purchase are needed.

References

[Orti97] Sixto Ortiz Jr., "Virtual Private Networks: Leveraging the Internet", IEEE Computer Magazine, pp.18-20, Nov. 1997

[Cohe00] Reuven Cohen, "On the Cost of Virtual Private Networks", IEEE/ACM TRANSACTIONS ON NETWORKING, Vol. 8(6), pp.775-784, Dec. 2000

[Youn00] Roger Younglove, "Virtual Private Networks-how they work", COMPUTING & CONTROL ENGINEERING JOURNAL, pp.260-262, Dec. 2000

[Venk01] R. Venkateswaran, "Virtual Private Networks", IEEE POTENTIALS, pp.11-15, Feb. 2001

[Secu01] VPN Mailing Lists from SecurityFocus.com, moderated by Tina Bird, Counterpane Internet Security, Inc.

[Gunt99] Manuel Günter, Torsten Braun, Ibrahim khalil, "An Architecture for Managing QoS-enabled VPNs over the Internet", IEEE Conference on Local Computer Networks, pp.122-131, 1999

[Blac00] Uyless Black, "Internet Security Protocols: Protecting IP Traffic", Prentice-Hall, Inc., New Jersey, 2001