PROTOCOL INDEPENDENT LIGHTWEIGHT SECURE COMMUNICATION

M. Amaç Güvensan, A. Gökhan Yavuz
Computer Engineering Department, Yıldız Technical University, Yıldız, İstanbul, Turkey

Keywords: Protocol-independent, easily configurable structure, kernel-level implementation, fast secure communication.

Abstract: This paper introduces a new protocol independent security mechanism, called PILSC (Protocol Independent Lightweight Secure Communication). PILSC utilizes the security feature of IPv4, defined but not used yet, in order to have standardization in secure communication. We aim to increase the efficiency of the secure data transfer by means of examining the shortages of different security protocols. Although IPSec is the only protocol independent protocol, the redundant overhead and its hardly configurable structure encourages us to design a more fast and easy configurable mechanism, whose architecture is presented in detail in this paper. The implementation of PILSC on the kernel-level brings %75-%90 performance enhancement on cryptographic process time in comparison to the implementation of cryptographic processes in the user-space. Moreover, secure data transfer rate of PILSC is %20-25 faster than IPSec and SSL.

1 INTRODUCTION

Nowadays, by the agency of Internet, information sharing is very much to happen in an easy and rapid way. Every day, important and secure information is travelling through LAN's and WAN's from one computer to another around the world. The rapid growth of data transfer increases the importance of secure communication.

There are many researches to achieve secure data transfer mainly to protect against the attacks, which are growing rapidly. A malicious person can have three intentions on data : *monitor, interception and construction* whereas the goal of network security is to provide confidentiality, integrity and authenticity (Alshamsi and Saito, 2005)(Schneier,1996). In recent years, many security protocols and mechanisms have been proposed to accomplish this goal.

Researchers developed many different applications communicating securily. However, these application-dependent architectures did not have any standard. Therefore, new mechanisms (SSL, TLS) were needed to be designed which would work at the lower layers. After construction of such transport layer mechanisms, secure communications can be handled for any application that runs over TCP. However, these tools can not encrypt TCP header of transmitted packets. Moreover, they do not support other protocols like UDP. As a consequence of these shortages, new approaches were taken out. The most famous, secure and widely deployed IPSec framework is designed to work at the Network Layer(Alshamsi and Saito, 2005).

In this paper we provide a new security mechanism that aims protocol independent, fast and configurable secure data transfer to increase the efficiency of the communication. This mechanism mainly focuses on the confidentiality. It does not directly address the authenticity and the integrity, although it can be extended this way very easily.

2 COMMONLY USED SECURE COMMUNICATION METHODS

There are different types of protocols for secure communication, each of them running at one of the layers listed below.

- Application Layer
- Transport Layer
- Network Layer

Amaç Güvensan M. and Gökhan Yavuz A. (2006).
PROTOCOL INDEPENDENT LIGHTWEIGHT SECURE COMMUNICATION.
In *Proceedings of the International Conference on Security and Cryptography*, pages 211-218
DOI: 10.5220/0002101002110218

In the next subsections we will examine them briefly.

2.1 Application Layer Protocols

Many different protocols, such as SSH, Secure FTP, were developed to accomplish secure data transfer. As an example SSH, the Secure Shell(Saito, et.al., 2002), is widely used as a secure remote terminal software. The SSH can make one login to a remote computer over insecure networks, execute commands, and transfer files between a remote computer and a local computer(Saito, et.al., 2002). It is very clear that SSH can only satisfy remote login procedures securily, likewise SecureFTP can only handle secure transmission of FTP protocol commands and data. Other secure applications can command only for secure data created by them. All these applications have individual solutions for secure communication that prevents standardization and centric management.

Secure Socket Layer (SSL) is another application layer protocol and compatible with applications running only over TCP, but some modifications are required for the applications to run over SSL. It can not handle UDP, ICMP, etc. packets. SSL protocol needs some negotiation data to be exchanged between client/server applications. Although it ensures message integrity and packet authentication, the created overhead and hash algorithms (MD5, SHA1) slow down the data transfer(Alshamsi and Saito, 2005) (see Table 1).

Table 1: SSL Handshake Time (Alshamsi and Saito, 2005).

Mode	Establishing
Servet Authentication	41.7 msec
Client Authentication	74.8 msec

2.2 Transport Layer Protocols

Internet Engineering Task Force (IETF) has standardized SSL under the name Transport Layer Security (TLS)(Yasinsac and Childs, 2001). Any application that runs over TCP can also run over TLS. There are many examples of applications such as TELNET and FTP running transparently over TLS. However, TLS is most widely used secure transport layer below HTTP(http://searchsecurity.techtarget.com)(RFC 2402). TLS is still insufficient to solve problems of SSL.

TLS uses a handshake mechanism to exchange public keys. However, data items exchanged during TLS handshake increase the latency of HTTP transactions(Apostoloupos, et.al., 1999).

2.3 Network Layer Protocols

IPSec (Internet Protocol Security) is a framework for a set of protocols for security at the network or packet processing layer of network communication(http://searchsecurity.techtarget.com) . IPSec provides security at network layer between two applications independent of the protocol being used. We can say that it is the only protocol independent solution for secure communication.

IPsec provides two choices of security service: AH (Authentication Header), which essentially allows authentication of the sender of data, and ESP (Encapsulating Security Payload), which supports both authentication of the sender and encryption of data as well. The specific information associated with each of these services is inserted into the packet in a header that follows the IP packet header (http://searchsecurity.techtarget.com)(RFC2402)(RFC 2406).

IPSec uses extra header information during the secure data transfer. Its goal is to support authenticity. However, extra data causes extra time for the transmitted secure information as in SSL/TLS. IPSec protocols must cope with reliability and fragmentation issues, adding their complexity and processing overhead. SSL/TLS, in contrast, rely on a higher level layer TCP (OSI Layer 4) to manage reliability and fragmentation (http://en.wikipedia.org).

A main disadvantage of IPSec is its hardly configurable structure. Although IPSec supports encryption for all IP protocols, its handshake mechanism is slower than SSL handshake mechanism (see Table 1, 2). In most cases IPSec does not interoperate well, so both sides of connection are required to have the same vendor's devices(Alshamsi and Saito, 2005).

Table 2: IPSec Handshake Time.

Mode	Establishing
Main Mode (PSK)	97 msec
Aggresive Mode (PSK)	56 msec
Main Mode (RSA)	170 msec

2.4 Comparison of the Security Features of three Different Layer Protocols

In the above sections we discussed frequently used security protocols. Table 3 illustrates main features of these protocols comparatively.

Table 3: Different Layer Protocols.

Features	SSH/SecureFTP	SSL/TLS	IPSec
Application Dependent	Yes	No	No
Protocol Dependent	Yes	Yes	No
Handshake Time	Fast	Fast	Slow
Full TCP Support	No	Partial	All
UDP Support	No	No	Yes
Configurability	Easy	Easy	Hard

3 WHY PILSC?

Cryptographic processes are time consuming. Therefore, the balance between secure data transfer and fast data transfer needs to be adjusted according to the importance of data. From time to time, security can be discarded if there is nothing to hide. Some data, like anonymous pictures, mails etc. do not need to be secured. On the other hand, many other data transfers have to be done securily.

Security is not the only criteria for some data transfers. For an example, if an application using protocols like FTP, HTTP etc, needs fast communication, IPSec does not completely satisfy this necessity. IPSec, using hash algorithms (MD5,SHA1), is designed to accomplish the confidentiality, integrity and authenticity. However, due to the hash algorithms and handshake mechanisms time penalty is inevitable when transfering secure data.

An increasing number of applications – especially in real-time and multicast communications – are based on the connectionless User Datagram Protocol (UDP) that is generally hard to secure at the transport layer. However, the main disadvantage of security at the Internet Layer is that IP stacks must either be changed or extended(Oppliger,1998). PILSC (Protocol Independent Lightweight Secure Communication) proposes a faster mechanism than IPSec to decrease the effects of this disadvantage.

PILSC uses security option field of IPv4. The value “130” which is set in the option field represents the security(Tanenbaum, 1999) (http://www.networksorcery.com). IPv6 has a standart feature, which can be set by a user, to accomplish secure data transfer. PILSC implements this built-in security feature of IPv6 for IPv4.

PILSC is an assertive mechanism to be configured very easily. IP addresses, protocols, port numbers and encryption algorithm can be configured to determine an effective communication scheme for all protocols and applications. Thus, we can talk about a protocol and application independent, user transparent mechanism.

A new approach of PILSC is that a user can choose different encryption algorithms for each protocol. Users will decide only about the level of security for each application-level communication. According to this decision, the system determines the type of encryption algorithm. PILSC main goal is to achieve computationally secure data transfer.

PILSC main characteristics are summarized in Table 4.

Table 4: PILSC Characteristics.

1	Application Independence
2	Protocol Independence
3	Easy Configuration
4	User Transparency
5	Deterministic Approach
6	Support of various Encryption Algorithms
7	Use of Security Option of IPv4 (Tanenbaum, 1999)
8	Computationally Secure Data Transfer

4 DESIGN OF PILSC

PILSC works at the Network Layer as shown in Figure 1. It can handle all IP datagrams including TCP, UDP, ICMP, etc. protocols. This mechanism fullfills its function within the Network Layer managing the secure IP traffic.

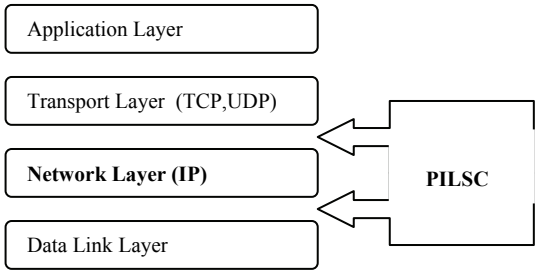

Figure 1: The place of PILSC on TCP Layer.

4.1 PILSC Rule Database

The rule database is the key of the PILSC architecture. It is assumed that packet traffic flows normally unsecure. With the help of this database system manages secure data transfer. It is a flat file which can be edited easily. The information; including rule number, security option, IP adresses, protocol type, port number, encryption type, application mode and key, saved in this file has to be protected against security attacks. The operating system, where PILSC is running on, is in charge of the protection of the rule database. The rule database contains special tags to be interpreted quickly as shown in Figure 2. As an example, Figure 3 shows one real record written in the rule database. The design of the rule database allows to establish many different individual secure connections. It is loaded into the memory during run-time to fasten the search and matching process.

Rule	<1..n>
Security	<true|false>
IP	<00:00:00:00 .. ff:ff:ff:ff>
Protocol	<TCP|UDP|ICMP>
Port	<0.. ffff>
Encryption	<0.. n>
Applmode	<Server|Client>
Key	<0..n>
#	Comment

Figure 2: Tag specification of the Rule Database.

Rule = 1;
Security = true;
IP = 193.140.1.1
Protocol = TCP;
Port = 20 – 21;
Encryption = 1;
Applmode = Server;
Key = 0xcda8;
Secure Data Transfer with the
computer having above mentioned
information. Encryption algoritm is AES

Figure 3: Example Record from Rule Database.

4.2 PILSC Main Steps

PILSC mechanism has four main steps (see Figure 2). The first step, called initialization, is processed only once to create necessary data structures in memory. All IP packets go through the second step. However, depending to the result of the third step only some IP packets are processed by the cryptographic step.

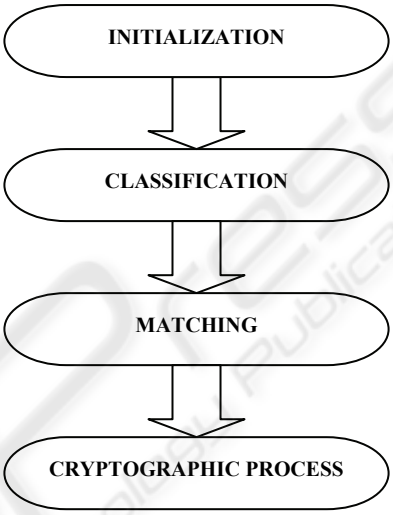

Figure 4: PILSC Main Steps.

4.2.1 Initialization

Initialization constructs data structures, which are used in the further steps. In this step the records of the rule database are processed and loaded into the memory to fasten the search and matching process, after checking for user errors, sorting records and eliminating duplicate rules defined by user.

4.2.2 Classification

In this step all IP packets are classified according to their types, i.e. IP_BROADCAST, IP_MULTICAST, IP_OTHERHOST... . Only incoming/outgoing IP packets are selected to be forwarded to the matching process. Other IP packets are left for normal processing.

4.2.3 Matching

Matching step involves two basic steps, checking and setting the value of security option field and matching the records of rule database structures created in the initialization step. Incoming and outgoing packets follow two different paths.

IPv4 packets have a security option field which we use to classify encrypted packets. For outgoing packets this field is set to "130" to indicate that the packet is encrypted whereas incoming packets are searched for a security option field with the value "130" to determine their encryption status.

Outgoing packets are checked against the rule database. If the packet matches a rule in the database, the security option field of the packet is set. Afterwards, the packet is handed in the cryptographic process.

The security option field of incoming packets indicates whether the packet needs to be checked against the rule database or stays in the packet queue. If the packet with the value "130" in its security option field matches a rule in the rule database, the packet is sent to the cryptographic process.

In order to fasten matching process we use "binary search" algorithm, which allows us to search the entire database in just $\log_2 n$ steps in the worst case.

4.2.4 Cryptograhic Process

This step is the last step for any exactly matched incoming/outgoing packets captured by the matching process. Outgoing packets are processed by encryption function whereas incoming packets visit the decryption function. Encryption/decryption algorithms are determined by the rule database of PILSC. PILSC supports different encryption algorithms including AES, 3DES, RC5, Blowfish, etc.

After cryptographic process is completed, outgoing encrypted packets are delivered to the Data Link Layer whereas incoming decrypted packets are transferred to the Transport Layer. The block diagram of the whole mechanism is shown in Figure 3.

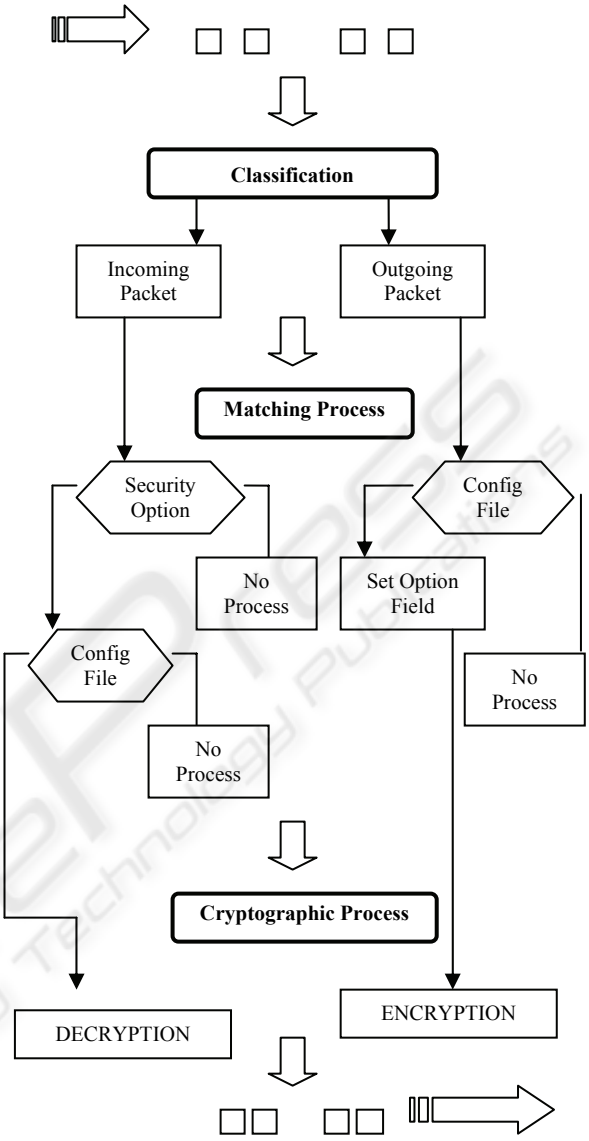

Figure 5: Block Diagram of PILSC.

5 PILSC PERFORMANCE

The experiments were conducted on two machines with the following configuration.

- Fedora Core 4 (Kernel-2.6.11)
- Intel Centrino, 1.6 GHz
- RAM 512 MB
- NIC 1000 Mbps
- Ethereal as time measuring tool

5.1 Overhead of PILSC

In order to compare the performance of PILSC and Linux Original Packet Handler we measured transfer

duration of files with different sizes transferred using FTP protocol. Table 5 shows the results of PILSC and Linux Packet Handler.

Table 5 : Transfer times (ms) of different size of files with Linux and PILSC.

Transfer Times (ms) / File Size	LINUX	PILSC
1KB	0,171	0,172
10KB	0,326	0,330
1MB	87	87,1
10MB	870,6	871

Table 6 illustrates that the overhead of PILSC is practically negligible. PILSC has an very low latency that this architecture can replace Linux Packet Handler with its new and powerful features.

Table 6 : PILSC Performance.

Delay Percentage / File Size	PILSC
1KB	%0,005
10KB	%1,3
1MB	%0,002
10MB	%0,0001

5.2 Effects of Rule Database Size on PILSC Performance

Incoming/Outgoing packets are matched with the records of the rule database. Hundreds of rules can be defined so we tested PILSC system how it reacts to an increase to the number of the rules in the rule database. The reaction is evaluated by measuring the transfer time of 1MB files. Table 7 shows how the number of rules affects the performance of PILSC. Even if with 1600 rules, transfer time increases only by % 6 - %7. Normally it is most unlikely to have even 1000 records in the rule database. Due to the design of PILSC and use of "binary search" algorithm we achieved a similar performance like the original Linux Packet Handler.

Table 7 : Measured transfer durations of 1 MB file with different number of files.

Transfer Duration (ms) / Number of Rules	PILSC
10	87,5
20	87,7
40	87,8
80	87,9
160	88
1600	93,3

5.3 Advantage of Kernel-Level Architecture

One of our design intentions was to fasten the secure data transfer as much as possible. Therefore, we designed the PILSC system to be run on the kernel-level. To see the results of running on the kernel-level , we tested two different encryption algorithms (RC5 and AES) both in the user-space and on the kernel-level. In this test scenario we measured the sum of encryption and decryption time of 1MB random data. Figure 6 shows the implementation of the same task between kernel-level and user-space. Here, PILSC shows great performance enhancement because of its implementation on the kernel-level.

5.4 Comparison of SSL, IPSec and PILSC

To compare the performances of SSL, IPSec and PILSC, we setup a test case where we used 1MB random files with 128-bits and 256-bits AES encryption, with the similar test platform used by Alshamsi and Saito. The obtained values are compared with the results of the test made by Alshamsi and Saito(2005). Figure 7 shows that PILSC is performing better than other security mechanisms except SSL with compression using AES-128 algorithm. In all other cases PILSC is not only %24-%26 faster than SSL but also %18-%22 faster than IPSec.

Besides, PILSC overcomes the encapsulation related overhead problem of IPSec. As we have mentioned in the above sections, PILSC uses the security feature of IPv4. Normally IPSec adds extra header, which varies from 32 to 48 bytes, to each transmitted datagram(Alshamsi and Saito, 2005)

(see Table 8). PILSC, on the contrary, uses the security option field of IPv4, which adds only 12 bytes. It means that each IPSec packets carries 3-4 times more extra data than packets of PILSC. Moreover, PILSC does not change the size of the encrypted IP payload so PILSC has a minimum overhead for each datagram which further fastens secure data transfer and does not waste network bandwidth.

Table 8: Overhead Size.

Protocol	Mode	Byte Size
IPSec Tunnel Mode	ESP	32
IPSec Tunnel Mode	ESP and AH	44
IPSec Transport Mode	ESP	36
IPSec Transport Mode	ESP and AH	48
PILSC	-	12

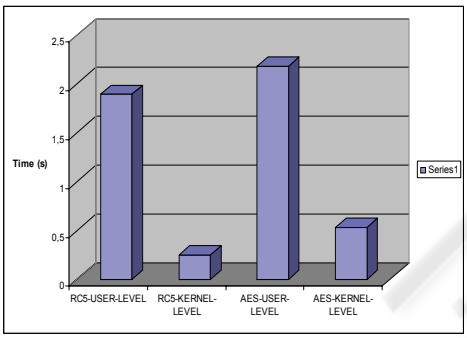

Figure 6: Performance Comparison of kernel-space and user-space.

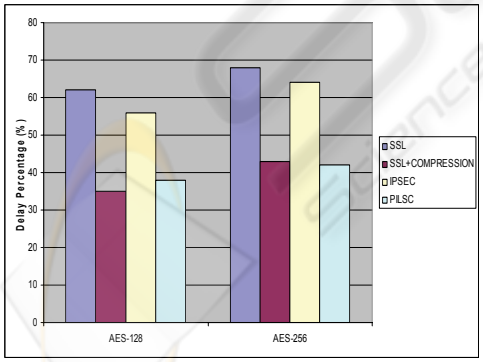

Figure 7: Performance Comparison of SSL, IPSec and PILSC.

6 CONCLUSION

The PILSC system offers an alternative secure communication model with great advantages. This system uses security option field in the IP header to achieve standardization and centric management for secure in communication in IPv4. The easily configurable structure of PILSC allows users to define various security rules with different parameters including IP address, protocol, port, encryption algorithm and encryption key.

Our main success is that PILSC performs %20-%25 faster than SSL and IPSec. One of the main reasons of this performance enhancement is that the design is completely implemented on the kernel-level instead of in the user-space. The kernel-level implementation brings %75-%90 performance gain on cryptographic process time in comparison to the implementation of cryptographic processes in the user-space.

PILSC is encryption algorithm transparent, thus, one can use any encryption algorithm, like AES, RC5, DES, Blowfish, etc.,. In conclusion, we constructed a new application and protocol independent, user transparent security model, which can be used provide appropriate security level to any application without a rewrite or compilation. Moreover, this model can be implemented in many different operating systems easily.

REFERENCES

Alshamsi, A. Saito, T.,*"A technical comparison of IPSec and SSL"*, AINA 2005

Apostolopoulos, G. Peris, V. Saha, D. ,*"Transport layer security:how much does it really cost?"*, INFOCOM 1999

Oppliger, R.,*"Security at the Internet Layer"*, Computer 1998

Saito, T.;Kito, T.; Umesawa, K.; Mizoguchi, F.; *"Architectural Defects of the Secure Shell"*,DEXA 2002

Schneier, B., "Applied Cryptography Protocols, Algorithms and Source Code in C", Second Edition, John Wiley & Sons, 1996

Tanenbaum, A., *"Computer Networks"*, Third Edition 1999

Yasinsac, A.; Childs J., (2001), "Analyzing Internet Security Protocols", HASE 2001

"TLS Protocol Version 1.0" ,RFC 2246

"IP Authentication Header(AH)", RFC 2402

"IP Encapsulating Security Payload (ESP)", RFC 2406

http://searchsecurity.techtarget.com, [24 Dec 2005]

http://en.wikipedia.org, [26 Dec 2005]

http://www.networksorcery.com, [10 Jan 2006]