Proprietary Cipher on TCP/IP Performance Analysis for IPsec

Yousuf Al Boloushi yousuf.al-boloushi@ensimag.imag.fr
Amal Al Dhaheri amal.al-dhaheri@ensimag.imag.fr

June 17, 2009

Contents

Abstract

In this document we will illustrate the transmission performance using TCP/IP with and without IPsec between two hosts that are connected via a direct Ethernet cable. First, we will give a general overview about the project principles and ideas. Then, we will talk about different kinds of tools that help us to measure the performance. Then, we will show the different tests and results we got for each tests. Finally, we will compare the different results and we will analyze them to give the final conclusion for the overall work.

1 Introduction

Nowadays, the security becomes very important and rises as a major issue in many fields. In every transactions of different data between many hosts where sensitives data are exchange. Many fields required to get these data to transfer in a safe way. With the fast development of telecommunication, security protocols and theory has been add to secure transfer data using TCP/IP and one of these way is IPSec protocol (see Figure 1). This project goal is to show

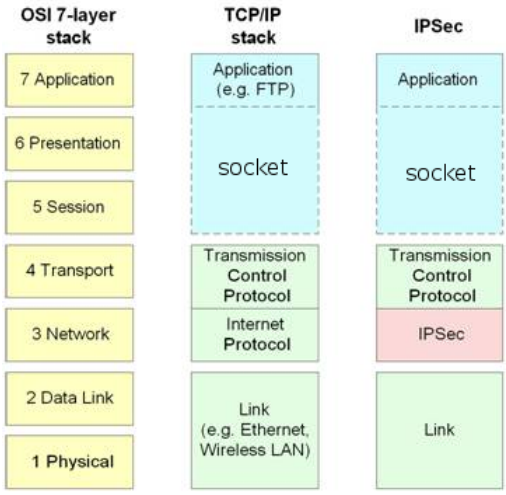

Figure 1: IPsec in OSI Layers

the impact of using secure communications through transmission the data between hosts and analysis the the performance before and after enabling the secure communications. We will use TCP/IP protocol as it is known that it is a communication protocol for communication between computers and we will enable IPsec protocol to secure the communication between hosts. The IPsec is an extension to the IP protocol which provides security to the IP and the upper-layer protocols. It uses two different protocols AH and ESP (explained in

section 3.1.2) to ensure the authentication, integrity and confidentiality of the communication [1].

IPsec has two modes called tunnel mode and transport mode. We used the transport mode in this project which is the default mode for IPSec. It is used in particular for end-to-end communications (for example, for communications between a client and a server). This mode encrypts only the IP payload. Transport mode provides the protection of an IP payload through an AH or ESP header (see Figure 2).

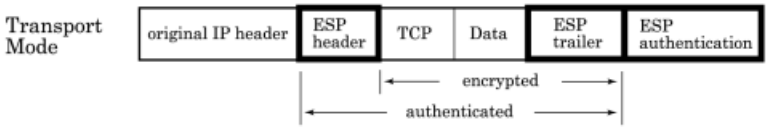

Figure 2: Transport Mode

After that, we will measure the bandwidth and RTT (Round Trip Time) of the transmission before and after securing the communications. Then, analyze these measurements results and explain what we conclude from these results.

2 Different Tools

We used different tools and transmission configuration that help us to do the measurements when we benchmark the performance under different scenario. Here we will illustrate different kind of tools that we used.

2.1 Performance measurement and Sniffer tools

There are two main categories of tools that support different functionality which help us to achieve the measurements:

- First category are packet sniffers that runs under the application layer. It allows the user to intercept and display TCP/IP and other packets being transmitted or received over a network to which the computer is attached such that Wireshark or Tcpdump. We used both to see the transmission behavior in physical link..
- Second category are network testing tools that can create TCP and UDP data streams and measure the throughput, latency and bandwidth. They allow the user to set various options that can be used for testing a network, or alternately for optimizing or tuning a network such as massage size, socket buffer, numbers of parallel flows of data, and destination address. They support a client and server functionality and some example of these tools are: iperf, nttcp and Lmbench. We used the Lmbench tool that satisfy the requirements to do the measurements and we will explain more about it in section 2.3.

2.2 IPsec-tools

IPsec-Tools is a Linux port of the user-space tools from KAME project[2]. It includes libipsec a library with a PF KEY implementation (i.e. PF KEY is a new socket protocol family used by trusted privileged key management applications to communicate with an operating system's key management internals[4]), setkey (a tool for manipulating and dumping the kernel Security Policy Database and Security Association Database), and racoon (Internet Key Exchange daemon for automatically keying IPsec connections). Instead of using racoon in this experiment we use manual keyed connection. It means all parameters needed for the setup of the connection are provided by the administrator. The administrator decides which protocol, algorithm and key to use for the creation of the security associations and populates the Security Association Database (SAD) accordingly.

Currently the tools are available at http://ipsec-tools.sourceforge.net/. When compiling the package manually we need to specify the location of the kernel headers. This package needs the kernel headers of at least kernel version 2.5.47.

In this tools there are several algorithms for Authentication and Encryption to be used in Authentication Header (AH) and Encapsulating Security Payload (ESP), Table 1 show these algorithms with the key size [1]

Algorithm	Key Length
ESP-DES	56
ESP-3DES	168 (3 x 56)
ESP-AES	128
AH-MD5	128
AH-SHA1	160

Table 1: IPsec Key length

2.3 LMbench tool

Lmbench is a serie of micro benchmarks intended to measure basic operating system and hardware system metrics. The benchmarks fall into three general classes: bandwidth, latency, and other (e.g. graph, cycle time and etc..).

Bandwidth is measured by creating two processes, a writer and a reader, which transfer specific size of data. The size was chosen so that the overhead of system calls and context switching would not dominate the benchmark time. The reader prints the timing results, which guarantees that all data has been moved before the timing is finished(i.e. while the writer stop write in the buffer and insure that all data was received successfully in the reader side the time will displayed immediately, so we guarantees the right timing)

TCP latency is measured by having a server process that waits for connections and a client process that connects to the server. The two processes then

exchange a message between them. The latency reported is one round-trip time [3].

3 Evaluation Platform Configuration

As we mentioned before the main goal of the project is to compare the different between normal TCP/IP communication and TCP/IP with IPsec communication. Therefore, we will measure the performance of the network before and after enabling IPsec and we will compare the results of these different tests. The experiment tests will go through three major parts; the first test will be basic TCP/IP test transmission, the second will be TCP/IP when enabling IPsec with Authentication Header (AH only) and the third test will be TCP/IP when enabling IPsec with Encapsulating Security Payload and Authentication Header (ESP and AH).

- In all tests we use two stations with:
- CPU product: Intel(R) Core(TM)2 Duo CPU
- Operating System: Linux v2.6.24-23-generic i686 GNU/Linux
- Network description: Ethernet interface product: 82566MM Gigabit, Network Connection vendor: Intel Corporation physical, width: 32 bits Gigabits Ethernet network cards
- cross cable connected two host

3.1 IPsec configuration setup

Here we will show the IPsec configuration setup into steps, first how to configure and check that kernel supporting IPsec. Then we will explain all the features that provided by IPsec and use it to make the secure communication between hosts.

3.1.1 Enable IPsec in Kernel

To provide IPsec we need to recompile the kernel or check if our version of kernel support IPsec. This can be done by enabling these options in the kernel configurations :

```
Networking support (NET) [Y/n/?] y
*
* Networking options
*
  PF_KEY sockets (NET_KEY) [Y/n/m/?] y
  IP: AH transformation (INET_AH) [Y/n/m/?] y
  IP: ESP transformation (INET_ESP) [Y/n/m/?] y
  IP: IPsec user configuration interface (XFRM_USER) [Y/n/m/?] y
Cryptographic API (CRYPTO) [Y/n/?] y
```

```
HMAC support (CRYPTO_HMAC) [Y/n/?] y
Null algorithms (CRYPTO_NULL) [Y/n/m/?] y
MD5 digest algorithm (CRYPTO_MD5) [Y/n/m/?] y
SHA1 digest algorithm (CRYPTO_SHA1) [Y/n/m/?] y
DES and Triple DES EDE cipher algorithms (CRYPTO_DES) [Y/n/m/?] y
AES cipher algorithms (CRYPTO_AES) [Y/n/m/?] y
```

Once the kernel is compiled and installed then IPsec-tools can be installed.

3.1.2 IPsec Features

We enable the IPsec to get the secure communications. It relies on:

- Authentication Header (AH): to provide connectionless integrity and data origin authentication for IP datagrams and to provide protection against replay attacks.
- Encapsulating Security Payload (ESP): to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service (a form of partial sequence integrity), and limited traffic flow confidentiality.

While we are doing experiment, we establish connection between two stations together by using TCP/IP with Authentication Header (AH) protocol. In next stage of testing we add Encapsulating Security Payload (ESP) and Authentication Header (AH) together. Initially, we started our experiments using 3DES-CBC algorithm for encryption and HMAC-MD5 algorithm for hash function be used in Authentication Header. In a second step we change these algorithms to AES-CBC and HMAC-SHA1 for more security.

3.2 RTT Evaluation Methodology

In this test, we connect two stations together and establish the connection using TCP/IP protocol. Then we measure the RTT (Round Trip Time) for this kind of test with different segment size by specify the MTU (Maximum Transmission Unit) which is the sum of MSS(Maximum Segment Size), IP header and TCP header, for IPsec MTU can be computed in same way in addition to AH Header and ESP Header(in case we use both).

Actually we prefer to measure the RTT which is sensitive measurement instead of measuring the one way latency, and the reason behind this is to avoid inaccurate time when the two hosts does not synchronize their time, to get synchronizes we lost some delay time so it is better to measure RTT as parameter in our experiment to gain accurate results.

3.3 Bandwidth Evaluation Mythology

We specify the size of the data that will be transmitted over the network. To that purpose, we carried out several experiments to identify at which point the bandwidth achieves its maximum and we kept the corresponding data size.

Therefore, there is no need to go more than this size which is 1 MB where is the test go through many size exactly from 10 KB to 80 KB. Starting from the 1 MB, the bandwidth reach it is maximum which is around 116.00 MB/sec as shown in the table 2 and figure 3.

Message size	Bandwidth MB/sec
10 KB	89.66
100 KB	102.49
300 KB	111.19
500 KB	113.94
1 MB	116.65
10 MB	116.92
20 MB	116.93
40 MB	116.95
80 MB	116.98

Table 2: Bandwidth Stability

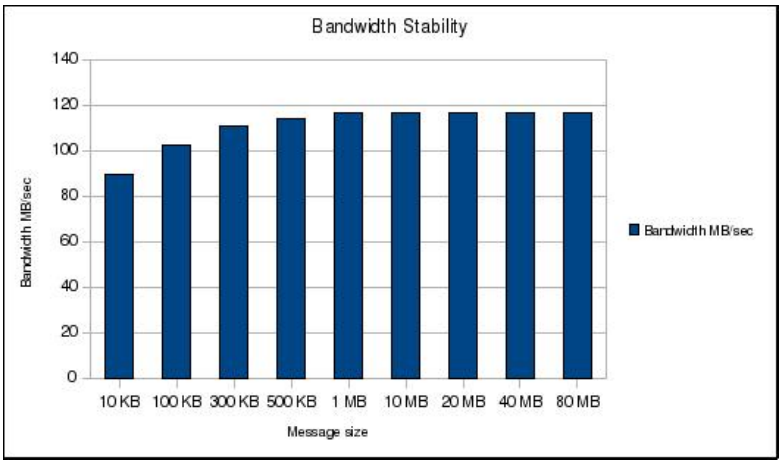

Figure 3: Bandwidth Stability Graph

After that, we measure the bandwidth with various number of flows(parallel) with the fixed data size (1MB), to see the impact of changing numbers of clients whom transmit data simultaneity to one server.

4 Results

From the tests we did, we got the following results and displayed in charts, to clarify and compare the output results from the different tests.

4.1 RTT Results

In Table 3, RTT results are shown for the tests which are:

- Basic TCP/IP connection without enabling IPsec
- IPsec with AH only
- IPsec with AH and ESP

In each test, we specify the MTU size and compute MSS size according to the test type. In this way we insure that the client send one packet to the server and server return the same packet so we get the result for one RTT.

We got the following results that show in table 3, RTT results with various Maximum Transmit Unit (MTU).

MTU Bytes	TCP/IP Microsecond	IPsec AH Microsecond	IPsec AH+ESP Microsecond
300	186.479	242.928	290.688
600	201.660	305.165	399.699
900	209.041	333.436	434.119
1200	215.058	341.689	463.486
1500	233.905	362.468	553.762

Table 3: RTT

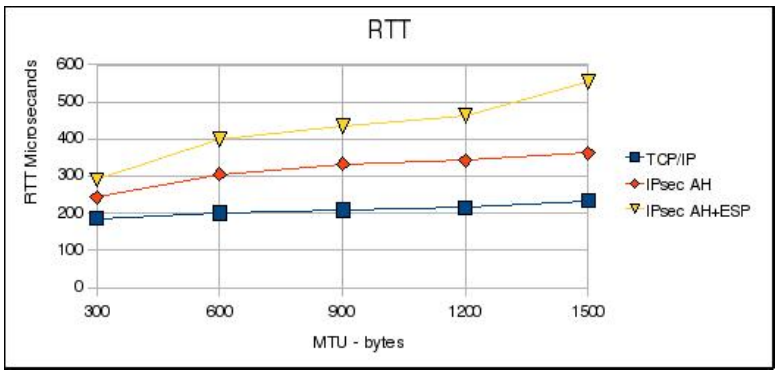

Figure 4: RTT Graph

Figure 4 show us the variation results of the three tests. In TCP/IP packet transmission the RTT was take less time than the other IPsec packet transmission. In other hand, IPsec with AH packet header with ESP packet header take more than other transmission. These delay happening when adding more overhead in IPsec layer.

4.2 Bandwidth Results

In Table 4, Bandwidth results are shown for the tests which are:

- Basic TCP/IP connection without enabling IPsec
- IPsec with AH only
- IPsec with AH and ESP

In each test, we increase numbers of flows (clients) to figure out the effect of the these changes in the bandwidth measurement with fixed data stream size to 1MB (refer section 3.3). We got the following results that show in table 4, Bandwidth results with various numbers of flows.

Flows number	TCP/IP	IPsec AH	IPsec ESP+AH
1	116.65 MB/sec	27.01 MB/sec	39.35 MB/sec
2	115.51 MB/sec	56.67 MB/sec	45.79 MB/sec
4	115.67 MB/sec	83.16 MB/sec	45.83 MB/sec
8	115.66 MB/sec	76.79 MB/sec	45.97 MB/sec
12	115.67 MB/sec	68.75 MB/sec	47.18 MB/sec
20	115.70 MB/sec	44.50 MB/sec	21.20 MB/sec
30	116.55 MB/sec	39.57 MB/sec	18.40 MB/sec
40	116.27 MB/sec	39.75 MB/sec	17.62 MB/sec
60	117.03 MB/sec	40.42 MB/sec	18.95 MB/sec
80	117.02 MB/sec	44.88 MB/sec	19.38 MB/sec

Table 4: Total Bandwidth

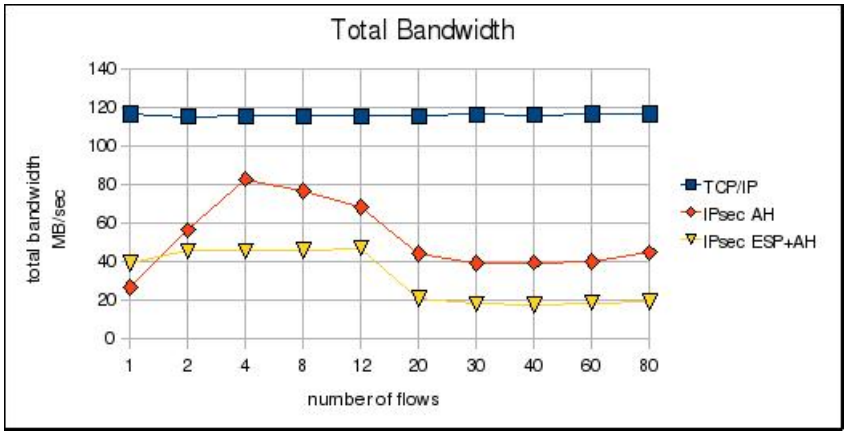

Figure 5: Bandwidth Graph

Figure 5 explain that TCP/IP have stable behavior with various numbers of flow which mean the stability of implementation of the standard type of

transmutation. While the IPsec play strange behavior, the bandwidth start to increase with the first numbers of flow until it reach 4 flows. Then it decrease until 30 flows in addition it increase slightly. We try to figure out this kind of behavior but it was mysteries while the sniffing the data is useless because ESP. So, it need more research in methodology of experiment to check if there was any bugs or start a new interesting research about this strange behavior.

5 Discussion and Conclusion

We did many tests to measure the transmission performance between two hosts connected via Ethernet cable and we test the transmission for three kind of connection which are: 1) Basic TCP/IP 2) IPsec with AH only 3) IPsec with AH and ESP. We did two kind of transmission performance measurements which are RTT and bandwidth by using Lmbench tool that support us to get accurate results for these measurements. We observe many things from the results we got.

Firstly, when we measure the RTT for the three different connection by specifying the MTU size, we notice that the basic TCP/IP take less delay time than IPsec connections. Then, the IPsec with AH header take a Little more than basic TCP/IP and less than IPsec with AH and ESP.After that,the IPsec with AH and ESP give higher RTT results than all other tests. Therefore, we conclude from the RTT test that more delay happen when more headers added to the IP packets.

Secondly, when we measure the Bandwidth for the same three tests for different number of flows and with fixing the packet size to 1 MB, we observe various behavior. In basic TCP/IP, since it is known protocol and it have been tested before, it is give us a stable and kind of constant bandwidth for different number of flows. On other hand, the IPsec gives us completely different behavior which increase little, then decrease in certain point and then increase again. Moreover, the IPsec with AH head only increase more and give results higher than IPsec with AH and ESP headers. It is gives completely not constant and strange behavior that we tried to find explanation for it but we couldn't even when we repeat the tests many times to insure the accurate of the results.

In the end, securing the communication using IPsec between hosts is important for some fields but rise some issues like it will consume more bandwidths than the basic transmission in some cases and it takes more time to reach the destination every time the MTU increase as we saw in the experiments result we got in this documents. However, even different fields these days require a specific and high expectation for transmits a large important data in high bandwidth and fasts way with less RTT, we think even with enabling to send and receive the data in a safe way more important and worth these limitations.

6 References

[1] IPsec HOWTO Ralf Spenneberg ralf (at) spenneberg.net 2003-08-18
[2] KAME project,last modified on 11 June 2009 at 12:28 , http://en.wikipedia.org/wiki/KAME_project [viewed 03/06/2009]
[3] LMbench Larry McVoy, Silicon Graphics Carl Staelin, Hewlett-Packard Laboratories
[4] RFC 2367 - PF KEY Key Management API, Version 2