Performance Implications of IPsec Deployment

John Ronan, Steven Davy, Paul Malone, Mícheál Ó Foghlu

Telecommunications Software & Systems Group (TSSG), Waterford Institute of Technology, Ireland.
E-mail: jronan,sdavy,pmalone,mofoghlu@tssg.org

Abstract

Virtual Private Networks (VPNs) use the Internet or other data network service as a backbone to provide a secure connection across a potentially hostile WAN. Such security guarantees provide the motivation for VPN deployment. This security does, however, come at a performance cost brought about by the increased processing overhead. This paper presents an investigation into these overheads. In particular, this investigation will consider different user resource availability in addition to router type and encryption algorithms.

1. Introduction

This work comes from the simulation of several broad categories of users. Researchers with high speed networking capabilities (GEANT) and SMEs, with more constrained bandwidth links have very different resource availability and as such suffer from different consequences of IPsec performance overhead. The results will indicate the realistic performance of IPsec under a range of conditions. These conditions will be representative of various software and hardware platforms commonly available, corresponding to different user categories.

In total, five user categories are represented in terms of bandwidth; specifically 100Mbps, 34Mbps, 4Mbps, 2Mbps and 1Mbps links. These were tested using three different routers (AMD Opteron, Compaq DL380 G2 and Pentium III 550 10/100 ports). Two algorithms are being used (AES for encryption and MD5 for authentication) and tests are being done for both TCP and UDP transport protocols.

This work has evolved from an investigation within the TORRENT IST project where there is a requirement to encipher a communications link between one server (LAP) and many clients. In time, this has led to an evaluation of the performance implications of using IPsec to achieve this goal. This, in turn, has brought about a more detailed investigation as it became apparent that there were scalability issues involved.

This work is partly funded by TORRENT[1], an IST FP5 project and by SEINIT[2] an IST FP6 Integrated Project. The results of this work will feed into WP4 of the SEINIT project.

2 IPsec Protocol Suite

IPsec is the security architecture for the Internet Protocol (IP). This protocol is applicable to both IPv4 and IPv6. The architecture is defined in [1] and addresses the following 4 elements:

- Security Protocols: Authentication Header (AH) [2] and Encapsulating Security Payload (ESP)[3].
- Security Associations: Definition, management and processing.[6]
- Key Management: The Internet Key Exchange (IKE) [6],[7],[8],[9].
- Algorithms: Requirements of the authentication and encryption algorithms.

2.1 Security Protocols

Traffic Security is provided by two security protocols:

- The Authentication Header protocol [2] provides connectionless integrity and data origin authentication. There is also an optional anti-replay service available.
- The Encapsulating Security Payload protocol [3] [4] [5] potentially provides two types of security service. The first is confidentiality via encryption and limited traffic flow confidentiality. The second type is connectionless integrity, data origin authentication and an anti-replay service.

Either of these protocols can be applied alone or in combination, thus providing the desired level of security. The IPsec security protocols are represented by headers that appear before the IP header in the IP packet.

[1] www.torrent-innovations.org
[2] www.seinit.org

2.2 Security Associations

The security protocol headers do not contain information pertaining to the cryptographic algorithms and the associated parameters. These representations are achieved through the transmission of a *Security Parameter Index* (SPI). This index combined with the destination IP addresses and the type of protocol header (AH or ESP) determines the parameters of the IPsec processing.

These parameters of a unidirectional security service are represented by a Security Association (SA). There are two types of SAs:

- *Transport Mode SA*: This is a security association between two hosts, generally used to secure the traffic of the upper layer protocols.
- *Tunnel Mode SA*: This is a security association in an IP-in-IP tunnel, generally used in connecting to security gateways.

2.3 Key Management

IPsec mandates support for two separate methods of cryptographic key and SA management: manual and automatic.

- *Manual Key Management*: This is the simplest form of key management and involves each IPsec connection to be configured manually on both hosts. While this is suitable in small static situations, it is unsuitable in larger deployment scenarios due to scalability problems.
- *Automatic Key and SA Management*: Larger deployment scenarios call for an Internet-standard, scalable and automated SA and key management protocol. This is provided by *Internet Key Exchange* (IKE). IKE is required to allow for use of anti-replay features of AH and ESP and to facilitate on-demand creation of SAs.

2.4 Algorithms

The IPsec protocol suite does not define the authentication and encryption algorithms used in implementations. These are defined in individual RFCs per algorithm. Algorithms used in these tests were:

- AES [10]
- HMAC-MD5 [11] [12]

Our reasons for choosing these algorithms are higher performance and are detailed in [**?**].

3 Scenarios

The version of IPsec used for this paper is built into the 2.6+ linux [14] kernels, for this paper the 2.6.1 kernel was used. There are a number of tools [15] that come with the kernel to aid in setting up the security associations needed for two hosts to speak IPsec, setkey command is used here. setkey can setup either tunnel or transport mode security connection.

The scenario is as follows: a set of routers will route secure traffic between two seperate networks, where the third network is assumed untrusted and simulates the Internet. After this scenario is set up, the bandwidth on the links is shaped to replicate common configurations in real world sceanrios. The configurations that are replicated represent researchers with the high speed network capabilities (GEANT) and SME's with more constrained bandwidth links. This translates into running the experiments over 100Mbps,34 Mbps, 4Mbps, 2Mbps and 1Mbps networks.

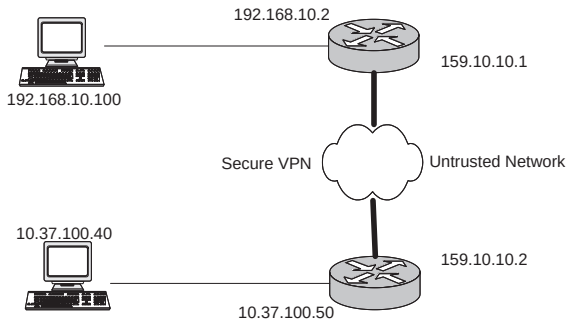

Figure 1. Illustration of Network Scenario

4 Testbed Configuration

There are three networks in this experiment, 192.168.10.0 is the Server network, 10.37.100.0 is the Client network and 159.10.10.0 is the assumed untrusted network. The Server is located at 192.168.10.100, and the Client is located at 10.37.100.40. The routers will be set up to route traffic from Server to Client and the return path. Note though that IP forwarding must first be enabled with the following command on both routers before traffic can be routed. After this is set up, any packet recieved by the router, that is not destined for one of its interface addresses, is routed through the appropriate interface. This interface is determined from the routes set up on the router.

```
echo 1 > /proc/sys/net/ipv4/ip_forwarding
```

4.1 Equipment used

All of the machines involved were using Linux Kernel version 2.6.1

- SERVER = Dual PIII 850 blade server running 100Mbps
- CLIENT = Dual PIII 850 blade client running 100Mbps
- ROUTER1 = Dual Xeon 2.4Ghz Compaq DL380-G3
- ROUTER2 = Dual PIII 850
- ROUTER3 = Dual AMD Opteron 2400

4.2 Setting up the routes

The Server machine needs to know how to route traffic to the Client network, located in the 10.37.100.0 network. This traffic is routed to the Server Router.

```
route add -net 10.37.100.0 \
netmask 255.255.255.0 gw 192.168.10.2
```

The Client machine needs to know how to route traffic to the Server network, located in the 192.168.10.0 network. This traffic is routed to the Client Router.

```
route add -net 192.168.10.0 \
netmask 255.255.255.0 gw 10.37.100.50
```

The Server router can route traffic directly to the Client router, so any traffic this router recieves destined for the Client network is sent to the Client router

```
route add -net 10.37.100.0 \
netmask 255.255.255.0 gw 159.10.10.2
```

The Client router can route traffic directly to the Server router, so any traffic this router recieves destined for the Server network is sent to the Server router.

```
route add -net 192.168.10.0 \
netmask 255.255.255.0 gw 159.10.10.1
```

4.3 Setting up IPsec Tunnel mode

We used manual keying for both authentication and encryption. As we wished to measure the performance of the routers. Tunnel Mode was employed. The security association provides confidentiality, data origin authentication, and connectionless integrity through the use of ESP. The crytographic algorithm used for ESP confidentiality is AES, and for ESP authentication is MD5. The following is applied to the setkey command (through the -f option) on the Router on the Client Side.

```
spdflush;
flush;

add 159.10.10.2 159.10.10.1 \
esp 0x750 -m tunnel -E rijndael-cbc \
0x11112222333344445555666677778888 \
-A hmac-md5 \
0x11112222333344445555666677778888;

add 159.10.10.1 159.10.10.2 \
esp 0x850 -m tunnel -E rijndael-cbc \
0x11112222333344445555666677778888 \
-A hmac-md5 \
0x11112222333344445555666677778888;

spdadd 192.168.10.100/32 10.37.100.40/32 \
any -P out IPsec \
esp/tunnel/159.10.10.2-159.10.10.1/require;

spdadd 10.37.100.40/32 192.168.10.100/32 \
any -P in IPsec \
esp/tunnel/159.10.10.1-159.10.10.2/require;
```

And the router on the Server side

```
spdflush;
flush;

add 159.10.10.2 159.10.10.1 \
esp 0x750 -m tunnel -E rijndael-cbc \
0x11112222333344445555666677778888
-A hmac-md5 \
0x11112222333344445555666677778888;

add 159.10.10.1 159.10.10.2 \
esp 0x850 -m tunnel -E rijndael-cbc \
0x11112222333344445555666677778888
-A hmac-md5 \
0x11112222333344445555666677778888;

spdadd 192.168.10.100/32 10.37.100.40/32 \
any -P in IPsec \
esp/tunnel/159.10.10.2-159.10.10.1/require;

spdadd 10.37.100.40/32 192.168.10.100/32 \
any -P out IPsec \
esp/tunnel/159.10.10.1-159.10.10.2/require;
```

Spdflush and flush erase all IPsec associations in the kernel then, the one that are used for the experiments are added. An association is set up on the *client* router to the server router and vice-versa In summary, the above commands tell the kernel to apply ESP tunnel mode IPsec to packets originating from address 192.168.10.100 (Server) and with destination address 10.37.100.40 (Client). An association in the kernel is created for the opposite direction also.

4.4 Performance Measuring Setup

The main performance measure this paper is focused on is the throughput, and processor overhead associated with IPsec. The throughput was measured with Iperf. Iperf [16] is a freely available, multi-platform bandwidth measurement tool. Iperf can measure many network related parameters, in this case we were really only interested in bandwidth, the processor overhead generated on the router and the relationship between them. As we wished to measure lower bandwidths than 100Mbps, the linux traffic controller *tc* was used to achieve lower throughputs.

4.4.1 Setting up tc

tc is a linux program used to control and shape traffic on an interface, although this program has quite a complex set of options, only one was used for our purposes.

```
tc qdisc add dev eth0 root tbf rate 34mbit \
burst 10kb latency 70ms
```

The above command will constrain the related network interface to a throughput of 34 Mbps. A command similiar to this was issued on both routers before Iperf was started.

4.4.2 Setting up cyclesoak

Cyclesoak [17] is a tool for accurately measuring system load andused to measure CPU utilization, by soaking up all idle cycles and calculating the load on the CPU. This tool is run on both routers, as it was here that we were interested in the processor overhead.

4.5 Performance Tests

There were three routers in all, each with dual processing cababilites. Tests were performed on the routers in single processor mode and dual processor mode, to give a richer set of results.

5. Results

If we first examine Figure 3, we can see that all the routers are capable of routing 34Mbps of IPsec encrypted traffic with varying loads on the processors (Figure 5, colums labeled 34, 4, 2, 1). This is also reflected in Tables 1, 2 and 3.

Interestingly, as can be seen from Figure 5, one could deploy a Single Processor Pentium III machine to act as a VPN gateway if the bandwidth requirements were 34Mbps (Figure 5) or less, with some headroom. Whats is also interesting is the comparison of the Xeon versus the Opteron Routers. The addition of the second processor has a far

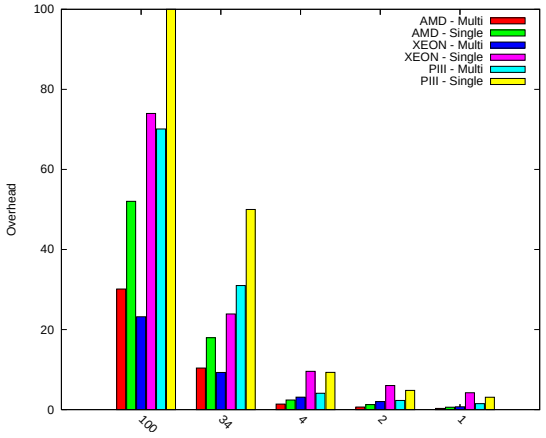

Figure 2. Overhead Analysis

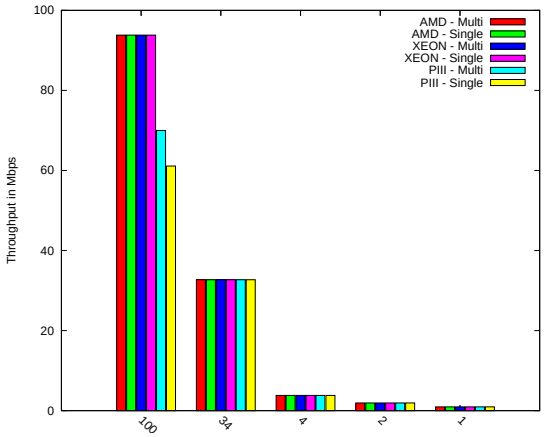

Figure 3. Throughput Analysis

Table 1. Flextel Blade Dual PIII 850 Mhz

Link Speed	Router Single		Router Dual	
	Throughput	Overhead	Throughput	Overhead
100	61.1	100%	70	70.9%
34	32.5	50.6%	31.8	32.5%
4	3.8	8.6%	3.8	4.1%
2	1.92	4.8%	1.92	2.3%
1	960	2.8%	960	1.5%

Table 2. Dual Xeon 2.4Ghz

Link Speed	Router Single		Router Dual	
	Throughput	Overhead	Throughput	Overhead
100	93.8	73.98%	93.8	23.21%
34	32.5	32.9%	32.5	9.28%
4	3.8	9.56%	3.8	3.12%
2	1.92	6.01%	1.92	2.01%
1	960	4.23%	960	.71%

Table 3. AMD Dual Opteron 2400

Bandwidth	Router Single		Router Dual	
	Throughput	Overhead	Throughput	Overhead
100	93.8	52.03%	93.8	30.13%
34	32.7	17.96%	32.7	10.38%
4	3.8	2.4%	3.8	1.38%
2	1.92	1.25%	1.92	.66%
1	.96	.6%	.96	.31%

greater effect than one would expect with the addition of a single processor. It manifests itself as the single processor Opteron showing less overhead than the Single processor Xeon, and the Dual Opteron showing more overhead than the Dual Xeon. This appears to be the effect the Xeon Hyper-Threading Technology has; we need to investigate this further.

6. Conclusion & Further Work

We need to return to the issue of the Xeon's Hyper-Threading technology. Also we would like to compare these results against a commercial offering. Also we would like to investigate how the addition of Hardware Accelerators would aid a *software* router to compete against the commercial offering.

References

[1] S. Kent, R. Atkinson, Security Architecture for the Internet Protocol. RFC 2401, November 1998.

[2] S. Kent, R. Atkinson, *IP Authentication Header*. *RFC 2402* November 1998.

[3] S. Kent, R. Atkinson, IP Encapsulating Security Payload. RFC 2406, November 1998.

[4] R. Pereira, R. Adams, The ESP CBC-Mode Cipher Algorithms. RFC 2451, November 1998.

[5] S. Frankel, R. Glenn, and S. Kelly The ESP CBC-Mode Cipher Algorithms. RFC 3602, September 2003.

[6] D. Maughan, M. Schertler, M. Schneider, and J. Turner *Internet Security Association and Key Management Protocol*. RFC 2408, November 1998.

[7] H. Orman, *The OAKLEY Key Determination Protocol*. RFC 2412, November 1998.

[8] D. Piper, *The Internet IP Security Domain of Interpretation for ISAKMP*. RFC 2407, November 1998.

[9] D. Harkins, D. Carrel *The Internet Key Exchange* (IKE) RFC 2409, November 1998.

[10] S. Franker, S. Kelly, and R. Glenn *The AES Cipher Algorithm and Its Use with IPsec* RFC 3602, September 2003.

[11] C. Madson, R. Glenn, The Use of HMAC-MD5-96 within ESP and AH. RFC 2403, November 1998.

[12] H. Krawczyk, M. Bellare, and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication*. RFC 2104 February 1997.

[13] J. Ronan, P. Malone, and M. Ó Foghlu Overhead Issues for Local Access Points in IPsec enabled VPNs. *Interdomain Performance and Simulation Workshop*, Salzburg Austria, February 2003.

[14] Linux *http://www.linux.org* Last Accessed, March 7^{th} 2004.

[15] IPsec-Tools *http://ipsec-toolos.sourceforge.net* Last Accessed, March 7^{th} 2004.

[16] Iperf *http://dast.nlanr.net/Projects/Iperf* Last Accessed, March 7^{th} 2004.

[17] Cyclesoak *http://www.zipworld.com.au/ akpm/linux/#zc* Last Accessed, March 7^{th} 2004.