Network and Complex Systems
ISSN 2224-610X (Paper) ISSN 2225-0603 (Online)
Vol.3, No.7, 2013

Performance Evaluation of Network Security Protocols on Open Source and Microsoft Windows Platforms

Oluwaranti A.I. (Corresponding Author)
Department of Computer Science and Engineering,Obafemi Awolowo University, Ile-Ife.
E-mail: aranti@oauife.edu.ng
Adejumo, E.O.
Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife.
E-mail: yinkaadejumo@gmail.com

Abstract
Internet is increasingly being used to support collaborative applications such as voice and video-conferencing, replicated servers and databases of different types. Since most communication over the Internet involves the traversal of insecure open networks, basic security services such as data privacy, integrity and authentication are necessary. One of the levels of computer security is operating system security. This paper analyzes the limitations and behavioral patterns of security protocols across different platform. It compared the performance of security protocols in terms of authentication, encryption algorithm, cryptographic methods etc.; in order to determine which platform provides better support for security protocols.
Network simulator tool was used to simulate different scenarios to show the performance of security protocols across two Operating System Platforms (Linux and Windows). Analysis of the simulation values of selected performance metrics of the security protocols, across both platforms, were evaluated.
Results obtained showed comparable differences in the values of the performance parameters considered. For instance, IP processing delay of the Windows Client node was initially high (about 0.0125 milliseconds), but later decreases to about 0.0115 milliseconds, while the Linux Client node is constant at about 0.0115 milliseconds. Variations in the values of the performance parameters for both platforms, in both network scenarios are not significant enough to reflect a noticeable difference in the impacts of the network security protocols on the performance of the operating system platforms.
Keywords: Open Source, IP Security, SSL, OPNET, Security Protocol, Operating Systems

1. Introduction

The Internet consists of an enormous number of heterogeneous, independently managed computer networks. It interconnects mutually distrustful organizations and people with no central management. Internet Users has come to depend on it for reliability inspite of its security issues. More reliance on the Internet is predictable in the coming years, along with increased concern over its security. Security and privacy are growing concerns in the Internet community, due to the Internet's rapid growth and the desire to conduct business over it safely. Basically, the security of a system builds on the combination of its ability to maintain confidentiality, integrity, and availability. This desire has led to the advent of several proposals for security standards such as secure Internet Protocol (IPSec), and the Secure Socket Layer (SSL) (Erich et al., 1996).

Most network security protocols on the Internet run on open source and/or windows platforms. They are prone to some limitations in their operations, which vary across platforms. These limitations occur in areas such as authentication, encryption algorithms etc. Maintaining a secure operating environment on a computer network requires familiarity with key security capabilities that meet the need for functionality, reduce risk and ensure compliance. This paper investigates-the operating system platform on which network security protocols would have the best performance.

This work is targeted at analyzing the limitations and behavioral patterns of security protocols across different platforms; comparing the performance of security protocols in terms of authentication, encryption algorithm, packet header, mode of key exchange, cryptographic methods etc.; and determining which platform provides better support for security protocols.

1.1 Open Source Operating System Platforms

Open Source refers to an approach to design, development, and distribution offering practical accessibility to a product's source (goods and knowledge). Open Source projects are generally proposed by a single developer or group of core developers who make the software application codes available to the public and use a system of peer review to test and refine the application.

Linux, a clone of the Unix operating system that has been popular in academia and many business environments for years, is the flagship of open source model. It consists of a kernel, which is the core control software, and many libraries and utilities that rely upon the kernel to provide features with which users interact. Most Linux software are available in open source form, and can be compiled on any Unix machine. Linux allows

customization of configuration files in ways that a Graphic User Interface (GUI) does not allow. Linux can be easily administered remotely, using common remote login tools such as Secure Shell (SSH), which allows running of text-based Linux programs from another system (Nash and Nash, 2001). Linux is not the only open source operating system in existence. Most competing open source operating systems are, like Linux, clones of Unix. The main competing family: FreeBSD, NetBSD, and OpenBSD are derived directly from mainstream Unix.

1.2 Microsoft Windows Operating Systems

Microsoft Windows refers to a series of operating system software and Graphical User Interfaces (GUIs) produced by Microsoft. The first version of Windows was introduced as an add-on to MS-DOS in response to the growing interest in graphical user interfaces (GUIs). Microsoft Windows dominates the world's personal computer market, with approximately 90% of the client operating system market as of 2004 (Hitslink, 2009).

Microsoft markets Windows as the competition to Unix and Linux. This operating system branch uses a kernel with support for features such as file-system security and multitasking. One of the main differences between Windows and Linux is that the former is much more integrated with its graphical user interface (GUI). This makes Windows easier to learn, but at the same time, reduces its flexibility. However, a major advantage of any Microsoft or Microsoft-related operating system is the application base for desktop use.

2. Literature Review

Security has become an increasingly important issue in modern distributed systems. The Internet is increasingly being used to support collaborative applications such as voice and video-conferencing, white-boards, distributed simulations, replicated servers and databases of different types. Since most communication over the Internet involves the traversal of insecure open networks, basic security services such as data privacy, integrity and authentication are necessary. A well-guarded enterprise deploys different security technologies. A computer network can be secured at many levels. One of these levels of computer security is operating system security.
It is easy to misunderstand assumptions on the environment in which security protocols are to be used and what their secure functioning may rely on. Security violations often occur at the boundaries between security mechanisms and the general system (Cole et al., 2005).

2.1 Network and Security Protocols

A protocol is a standard that controls or enables the connection, communication, and data transfer between two computing endpoints. A protocol can be defined as the rules governing the syntax, semantics, and synchronization of communication. Network security mechanisms are essential in other to prevent security threats. Protocols prevent security threats by providing the following; confidentiality (concealing the quantity or destination of data), data integrity (detecting and preventing tampering), originality (detecting replays), timeliness (detecting delaying tactics), authentication (ensuring that the communication is between the supposed parties), availability, non repudiation (detects bogus denial of transactions) and non forge-ability (detects claims of bogus transactions) (Joshi et al., 2008).

2.1.1 IP Security (IPSec)

IP Security (IPSec) is the leading standard for cryptographically based authentication, integrity, and confidentiality services at the IP datagram layer. Support for the IPSec architecture is mandatory in IPv6 but optional in IPv4. IPSec is a framework for providing a number of security services, as opposed to a single protocol or system.

When viewed from a high level, IPSec consists of two parts. The first part is a pair of protocols that implement the available security services. They are the Authentication Header (AH), which provides access control, connectionless message integrity, authentication, and antireplay protection, and the Encapsulating Security Payload (ESP), which supports these same services, plus confidentiality. The second part of IPSec is support for key management, which fits under an umbrella protocol known as Internet Security Association and Key Management Protocol (ISAKMP). The abstraction that binds these two parts together is the security association (SA). A security association is a simplex (one-way) connection with one or more of the available security properties.

IPSec supports a tunnel mode as well as the more straightforward transport mode. Each SA operates in one or the other mode. In a transport mode SA, ESP's payload data is simply a message for a higher layer such as UDP or TCP. In this mode, IPSec acts as an intermediate protocol layer, much like SSL/TLS does between TCP and a higher layer (Joshi et al., 2008).
In a tunnel mode SA, however, ESP's payload data is itself an IP packet. The source and destination of this inner IP packet may be different from those of the outer IP packet. The most common way to use the ESP is to build an IPSec tunnel between two routers, typically firewalls. According to Anderson (2001), the tunnel may be configured to use ESP with confidentiality and authentication, thus preventing unauthorized access to the data that passes through the link and ensuring that no spurious data is received at the far end of the tunnel. A network

of such tunnels can be used to implement an entire virtual private network (VPN).

2.1.2 Web security (Secure Sockets Layer and Transport Layer Security)

The design goals and requirements for the Transport Layer Security (TLS) standard and the Secure Socket Layer (SSL) on which TLS is based, are based on solving problems that emerged from the growth of the Internet and digital data transmission. As commercial enterprises began to take an interest in the World Wide Web, the need for some level of security for transactions on the Web became obvious; confidentiality, integrity, and authentication. SSL was the first widely used solution to this problem. It was originally developed by Netscape and subsequently the basis for the IETF's TLS standard. TLS is the latest enhancement of SSL.

SSL was not designed exclusively for Web transactions (i.e., those using HTTP). Rather, it was built as a general-purpose protocol that sits between an application protocol such as HTTP and a transport protocol such as TCP. From the application's perspective, this protocol layer looks just like a normal transport protocol except for the fact that it is secure. That is, the sender can open connections and deliver bytes for transmission, and the secure transport layer will get them to the receiver with the necessary confidentiality, integrity, and authentication.

2.2 Related Work

Series of research work have been carried out and are still being carried out on performance of network security protocols. In a work of Miltchev et al., (2001) the authors investigated the performance of IPSec by considering the type of encryption algorithm used by IPSec, the network topology, and the effects that the added security has on the performance of the system. IPSec was compared with SSL as used by HTTPS. The OpenBSD operating system was used as the experimental platform.

Also, in the work of Argyroudis et al., (2004), the performance analysis of three commonly used security protocols, SSL, IPSec and S/MIME was presented. This work compares the performance of a mobile platform with and without security protocols, to prove that the complexity of sophisticated cryptographic protocols do not prevent them from being used on a mobile platform. In contrast, this work investigates the difference in the performance of security protocols on two different operating system platforms.

3. Methodology

In order to study and exploit the properties of security protocols, a computer tool is needed, by which computer networks can be modeled, simulated and evaluated. This work evaluates the performance of security protocols using a network simulator, which uses packet level analysis to measure network performance. The following sub section discusses the performance metrics, the network simulation tool, and network models used in evaluating the performance of the security protocols.

3.1 Performance Metrics

According to Agarwal and Wang (2005), the performance impact of security policies on a system's Quality of Service (QoS) can be measured with the following metrics:

(a) **Authentication Time** (*AT*) is defined as the time involved in an authentication phase of a security protocol. Here, the steps to calculate the authentication time (*AT*) are described as follows:

i. Assume that security policy P_ϕ is configured in the network. Now, through experiments the time involved in processing k_{th} packet by P_ϕ during its authentication phase is determined. Let, it be denoted as $t_k(P_\phi)$.

ii. Assume N packets are exchanged during authentication phase. Let total time in processing N packets be represented by $TN(P_\phi)$, which can be calculated as follows:

$$TN(P_\phi) = \sum_{k=1}^{N} t_k(P_\phi) \quad (1)$$

iii. Let AT denote authentication time. As it depends on mobility scenarios N, R and security policies P as defined above, therefore AT can be represented as $AT(N,R,P)$ and can be calculated using the equation for AT above, as follows:

$$AT(N, R, P) = \sum_{k=1}^{N} t_k(P_\phi) \quad (2)$$

(b) **Number of Authentication Messages** (*AM*) is concerned about the messages exchanged during an authentication phase. Ethereal snapshots have been taken to obtain messages exchanged for different security protocols. This parameter is related to overhead signaling of authentication.

(c) **Policy Overhead (Bytes/Second)** $O(P_\phi)$ refers to the overhead associated in encrypting and decrypting. Once data transfer phase is initiated after initial protocol negotiation, encryption and decryption is the only operation on data. So their cost affects total overhead of security policies. It is assumed in the experiments that security policies do not renegotiate security parameters during a session, thus eliminating the overhead introduced by renegotiation of security policies.

(d) **Traffic Streams** (*Tr*) is considered with regards to TCP and UDP traffic streams in the experiments. Since most of the applications run over TCP or UDP, the experimental data is applicable to many applications in wireless LANs.

(e) **Response Time (End-to-End)** (*RS*) is a measure of the delay in transmission of data between a sender and a receiver, usually in seconds.

(f) **Throughput (Bytes/Second)** (*Th*) is a measure of the data transfer during per unit time between participating nodes. The throughput is obtained according to following steps:

- Determine time $t_f(P_\phi)$ when first data packet is sent from a sender to a receiver with security policy P_ϕ.
- Determine time $t_l(P_\phi)$ when last data packet is delivered to a receiver j from a sender i with security policy P_ϕ.
- Calculate total time, denoted as *tt*, by subtracting $t_f(P_\phi)$ from $t_l(P_\phi)$ which can be given as follows:

$$tt = t_l(P_\phi) - t_f(P_\phi) \quad (3)$$

- Assume that total data exchanged between users *i* and *j* are denoted as *D* in bytes. Since data rate, denoted as *dt*, is defined as data sent per unit time, therefore *dt* can be represented using the equation for *tt* above, as follows:

$$dt = \frac{D}{t_l(P_\phi) - t_f(P_\phi)} \quad (4)$$

- Since throughput *Th* depends on factors such as *N, R, P, Tr* and *DS*, where *Tr* represents traffic types such as TCP or UDP, *DS* denotes total data sent between a sender *i* and receiver *j* and other denotations are the same as defined above. Therefore, throughput can be represented as *Th*(*N,R,P, Tr, DS*), which can be obtained by using the equation for *dt* above, as follows:

$$Th(N, R, P, Tr, DS) = \frac{D}{t_l(P_\phi) - t_f(P_\phi)} \quad (5)$$

3.2 Simulation

Performance of the network security protocols were evaluated by measuring the values of performance metrics using OPNET. OPNET provides a GUI for network topology design, which allows for realistic simulation of networks, and has a performance data collection and display module. It has been used extensively and there is wide confidence in the validity of the results it produces (Guo et al., 2007).

OPNET IT Guru Academic Edition (ITGAE) is a free simulation tool, offered from the manufacturer of OPNET, and is intended for educational University programs. It is useful within education process concerned with communication technologies through practical simulation examples. Within the widely-supported components library can be found computer workstations and servers, routers, switches, bridges, stars, access points, links, firewalls, gateways, servers etc. The software is user-friendly, because the whole application can be constructed in a graphical project editor.

4. System Design and Implementation

Events in the modelled system are scheduled to occur at discrete points in time. The design of a network topology model to analyze the performance of security protocols in the operating systems described earlier is required.

4.1 Network Models

Network models were simulated, the first network model was used to evaluate IPSec and the second to evaluate SSL.

4.1.1 IPSec Network Model

This network model consist two Point-to-Point Protocol (PPP) workstations connected to the Internet via a router each; and a PPP server connected to the Internet through a router and a firewall. All devices are connected with PPP DS1 (Digital Signal 1) lines as shown in Figure 1. PPP DS1 connects two nodes running IP; its data rate is 1.544 Mbps.

The statistics selected for both the Linux Client and Windows Client nodes are:

- Client DB Response Time (seconds)
- Client HTTP Page Response Time (seconds)
- Client Remote Login Response Time (seconds)
- IP Processing Delay (seconds)

4.1.2 SSL Network Model

This network model consists of two PPP workstations connected to the Internet via a router each; a server farm consisting of a database server, an e-mail server, a FTP server, and a general server, all connected to a router which is connected to the Internet via firewall. Also connected to the Internet were two Web servers Yahoo and Amazon. All devices are connected with PPP DS1 lines as shown in Figure 2.

The statistics selected for both the Linux Client and Windows Client nodes are:

- Client Email Download Response Time (seconds)
- Client HTTP Page Response Time (seconds)

- Client FTP Download Response Time (seconds)
- IP Processing Delay (seconds)

4.3 Simulation Results

The results of the simulations of the IPSec and SSL models are presented and discussed in the following section.

4.3.1 IPSec Simulation Scenario

This is a simulation of the network and traffic models as shown in Figure 1. Database, HTTP and Remote Access traffic are transmitted through the VPN. IPSec provides confidentiality and authentication for the VPN tunnel. Figure 3 shows a graph of the average database query response time for the Windows Client and Linux Client nodes. The graph in Figure 4 shows the average HTTP page response time for the Windows Client and Linux Client nodes. Figure 5 shows the average remote access response time for the Windows Client and Linux Client nodes. Figure 6 shows the average IP processing delay for the Windows Client and Linux Client nodes.

4.3.2 SSL Simulation Scenario

This simulation model consists of the network and traffic models as shown in Figure 2. E-mail, HTTP and FTP traffic are transmitted over the network. SSL provides authentication and encryption for HTTP and FTP. Figure 7 shows a graph of the average e-mail download response time for the Windows Client and Linux Client nodes. The graph in Figure 8 shows the average HTTP page response time for the Windows Client and Linux Client nodes. Figure 9 shows the average FTP download response time for the Windows Client and Linux Client nodes. Figure 10 shows the average IP processing delay for the Windows Client and Linux Client nodes.

For both the IPSec and SSL Network scenarios, both the Linux Client node and the Windows Client node are connected simultaneously. This is done to ensure that the performance evaluation results obtained from both nodes are gotten under the same network conditions, for more accurate comparison.

4.4 Summary of Simulation Results

As discussed earlier, response time is a measure of the delay in transmission of data between a sender and a receiver, usually in seconds. The performance metrics obtained from the IPSec Network Scenario simulation include database response time, HTTP page response time and remote access response time.

The database response time is measured from the time when the database query application sends a request to the server to the time it receives a response packet. From Figure 3, the DB response time for the Windows Client node is initially about 0.190 seconds. It decreases to about 0.175 seconds quickly and remains constant. That of the Linux Client node on the other hand remains constant at about 0.180 seconds throughout.

HTTP page response time specifies the time required to retrieve an entire web page with all the contained inline objects. The HTTP response time for the Linux Client node is constantly about 0.55 seconds, while that of the Windows Client node is constant at about 0.54 seconds, as can be seen from Figure 4. Remote access response time is the time taken for the request for access to a remote resource to be granted. From Figure 5, the Remote access response time of the Linux Client node remains constant at about 0.165 seconds, while that of Windows Client node starts at 0.165 and then decreases to about 0.16 seconds.

IP processing delay is the time it takes routers to process the packet header. Additional overhead is caused by the IPSec datagram encapsulating the original IP datagram. This delay is evaluated with the following assumption. If N traffics arrive at each security router at rate V (Mbps), and if the security router has a processing capacity of rate R (Mbps), the security router processing delay, d_{sec} for data encryption or decryption is given as:

$$d_{sec} = \frac{N \times V}{R} \alpha \qquad (6)$$

where the security coefficient $\alpha = 1$ second.

From Figure 6, it can be seen that the IP processing delay of the Windows Client node is initially high (about 0.0125 milliseconds), but later decreases to about 0.0115 milliseconds. That of the Linux Client node is constant at about 0.0115 milliseconds, with a little spike initially. The IP processing delay of the Windows Client node is initially slightly higher than that of the Linux Client node, but later, both are comparably equivalent. The performance metrics obtained from the SSL Network Scenario simulation include e-mail response time, HTTP response time and FTP response time.

E-mail download response time is measured from the time an e-mail is requested to the time it is starts to download. From Figure 7, the e-mail download response time of the Windows Client node is 0.60 seconds at the start of the simulation; the graph curves slightly downward, as it becomes constant at 0.58 seconds. For the Linux Client node, the graph is slightly higher and parallel to that of the Windows Client node, starting at 0.64 seconds, and stabilizing at 0.62 seconds. HTTP response time for the Linux Client node ranges from 0.73 seconds to 0.77 seconds and is slightly higher than that of the Windows Client node, which is 0.7 seconds at the start of the simulation, spikes to 0.79 seconds, and then ranges from 0.72 seconds to 0.76 seconds as seen in

Figure 8.

FTP download response time is measured from the time a file is requested to the time it starts to download. The FTP download response time for the Linux Client node is constantly about 3.1 seconds, while that of the Windows Client node is 3.5 seconds at the start of the simulation, and decreases to and remains constant at 3.09 seconds. The IP processing delay of both the Linux Client and Windows Client nodes are approximately equal and constant at about 0.018 milliseconds, though at the start of the simulation that of the Linux client node is 0.0175 milliseconds, and that of the Windows Client node is about 0.017 seconds.

For the IPSec Network Scenario, the average response time for the database, HTTP and remote access applications is slightly greater for the Linux Client node than for the Windows Client mode. However, the IP Processing Delay is slightly greater for the Windows Client node than it is for the Linux Client node.

For the SSL Network Scenario, the average response time for the e-mail and HTTP applications is greater for the Linux Client node than it is for the Windows Client node. The FTP download response time is initially higher for the Windows Client node, and then later, lower. The IP Processing Delay is approximately equal for both the Linux Client and Windows Client nodes. It can be seen that the values of the performance parameters of the two operating system platforms considered for both scenarios are somewhat comparable.

5.0 Conclusion

For the IPSec Network Scenario, the average response time for the database, HTTP and remote access applications is slightly greater for the Linux Client node than for the Windows Client mode. However, the IP Processing Delay is slightly greater for the Windows Client node than it is for the Linux Client node.

For the SSL Network Scenario, the average response time for the e-mail and HTTP applications is greater for the Linux Client node than it is for the Windows Client node. The FTP download response time is initially higher for the Windows Client node, and then later, lower. The IP Processing Delay is approximately equal for both the Linux Client and Windows Client nodes. In each case, the differences in the values of the performance parameters are less than 5%. It can be seen from these results that the variations in the values of the performance parameters considered for the Linux and Windows operating system platforms, in both the IPSec and SSL Network Scenarios, are not significant enough to reflect a noticeable difference in the impacts of the network security protocols on the performance of the operating system platforms. Thus, it can be concluded that the effects of the network security protocols considered on the performances of both operating system platforms are comparable.

References

Agarwal, A. K., and Wang, W. (2005). Measuring Performance Impact of Security Protocols in Wireless Local Area Networks. Proceedings from the Second International Conference on Broadband Networks. (IEEE BROADNETS 2005).

Anderson, R. (2001). *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2nd ed. Wiley Publishing Inc. Indiana.

Argyroudis P. G., Verma R., Tewari H., O'Mahony D. (2004). Performance Analysis of Cryptographic Protocols on Handheld Devices. Proceedings of the Third IEEE International Symposium on Network Computing and Applications (NCA'04).

Cole, E., Krutz, R., and Conley, J. W. (2005). *Network Security Bible*. Wiley Publishing Inc., Indiana.

Erich, N., Yates, D. J., O'Malley, S., Orman, H., and Schroeppel, R. (1996). Parallelized Network Security Protocols. Proceedings of the 1996 IEEE Symposium on Network and Distributed Systems Security, 1-3.

Guo, J., Xiang, W., and Wang, S. (2007). Reinforce Networking Theory with OPNET Simulation. *Journal of Information Technology Education*, **6**, 215-226. [Online] http://www.usenix.org/event/usenix02/tech/freenix/fullpapers/miltchev/miltchev.ps [Accessed February, 2011]

Hitslink website, [Online] <http://www.hitslink.com > [Accessed February, 2011]

Joshi, J., Peterson, L. L., Bruce, S. D., Krishnamurthy, P. (2008). *Network Security: Know it all*, 2nd ed. Morgan Kaufmann, San Francisco.

Miltchev, S., Ioannidis, S., Keromytis, A. D. (2001). A Study of the Relative Costs of Network Security Protocols.

Nash, A. and Nash, J. (2001). *LPIC Certification Bible*. Hungry Minds Inc., New York.

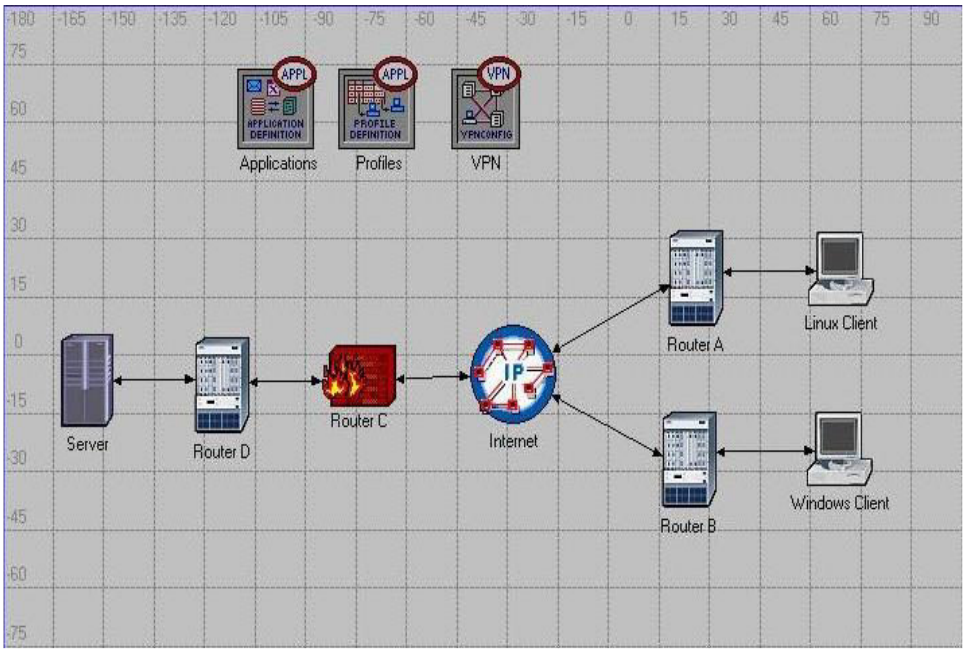

Figure 1: Network Topology Model to simulate IPSec

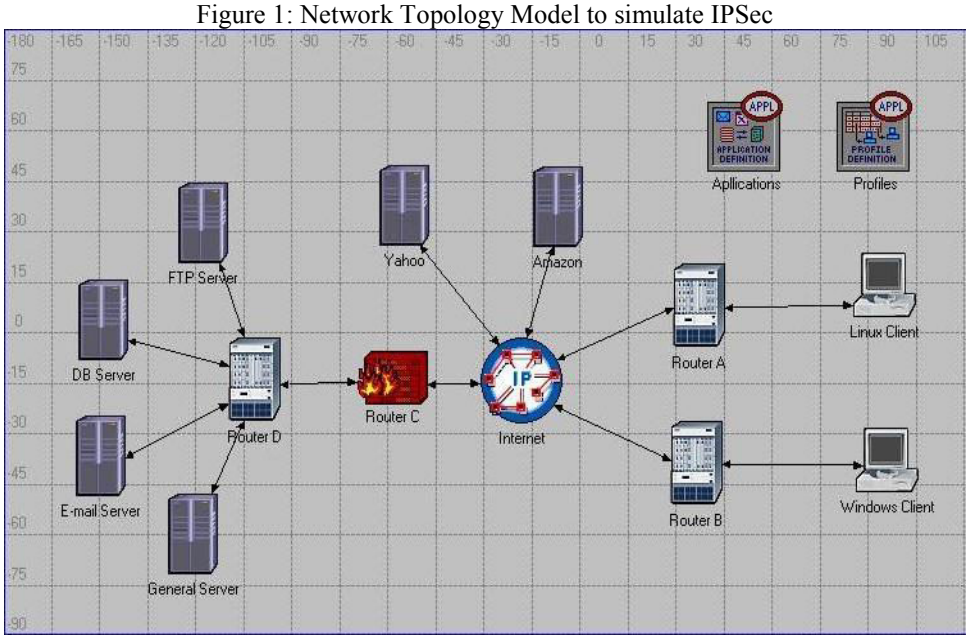

Figure 2: Network Topology Model to simulate SSL

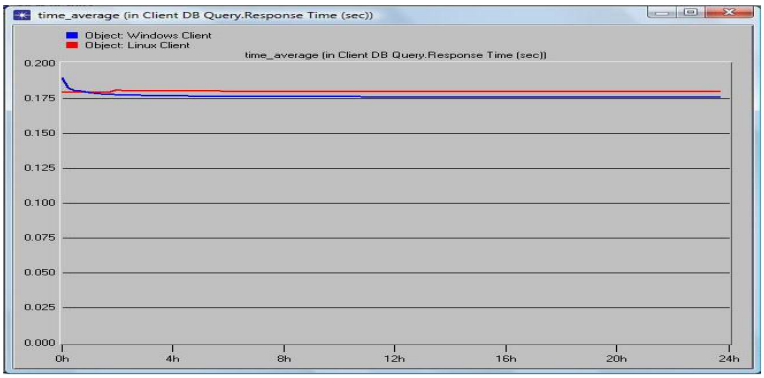

Figure 3: Comparison of Database Query Response Time between Windows and Linux nodes (IPSec).

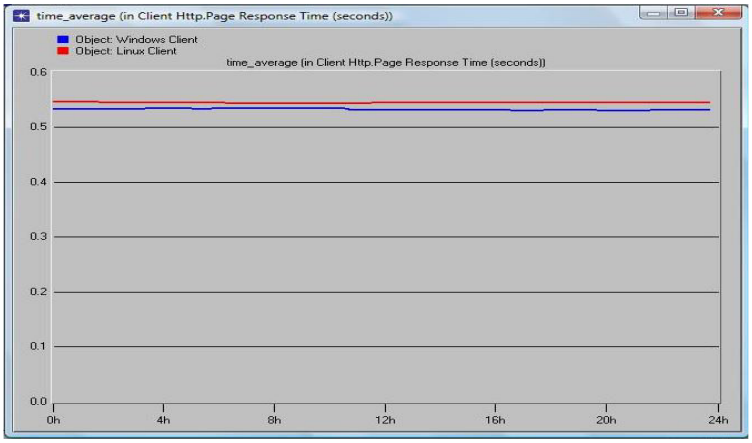

Figure 4: Comparison of HTTP Page Response Time between Windows and Linux nodes (IPSec).

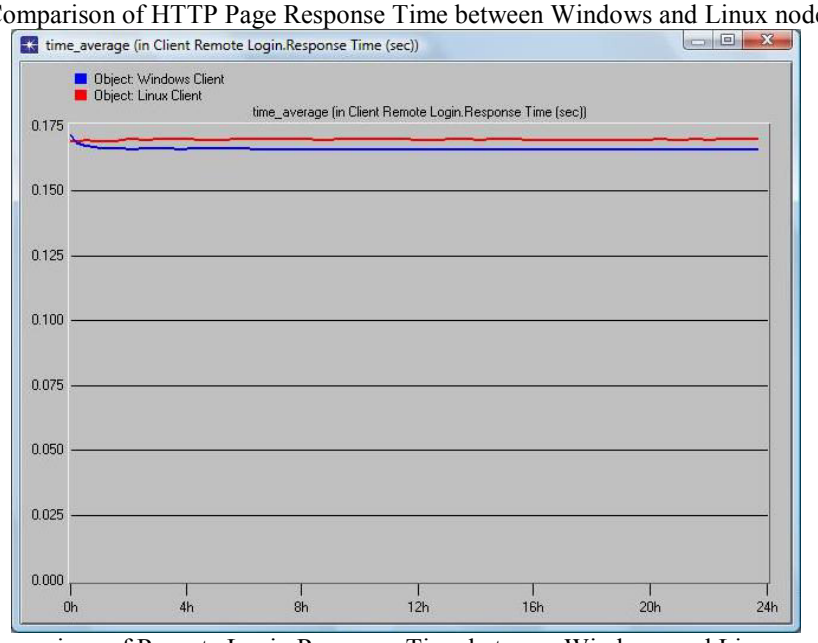

Figure 5: Comparison of Remote Login Response Time between Windows and Linux nodes (IPSec).

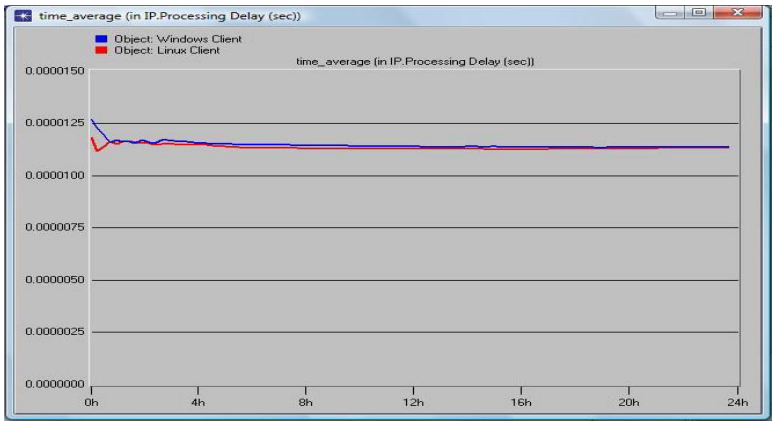

Figure 6: Comparison of IP Processing Delay between Windows and Linux nodes (IPSec).

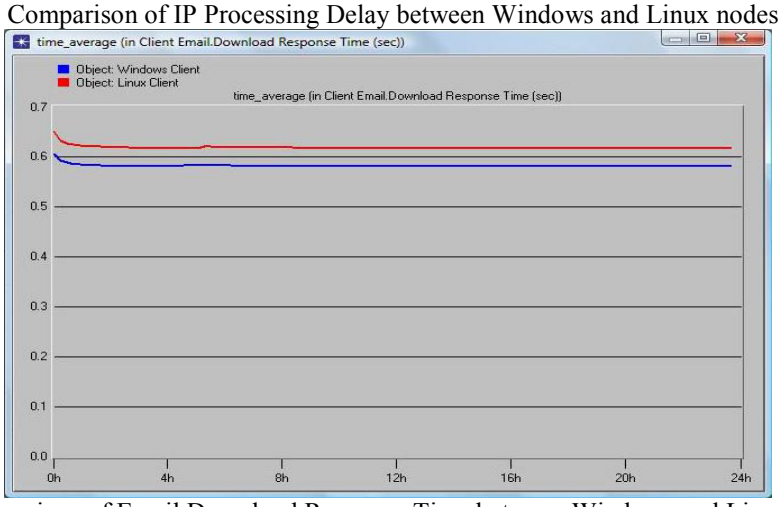

Figure 7: Comparison of Email Download Response Time between Windows and Linux nodes (SSL)

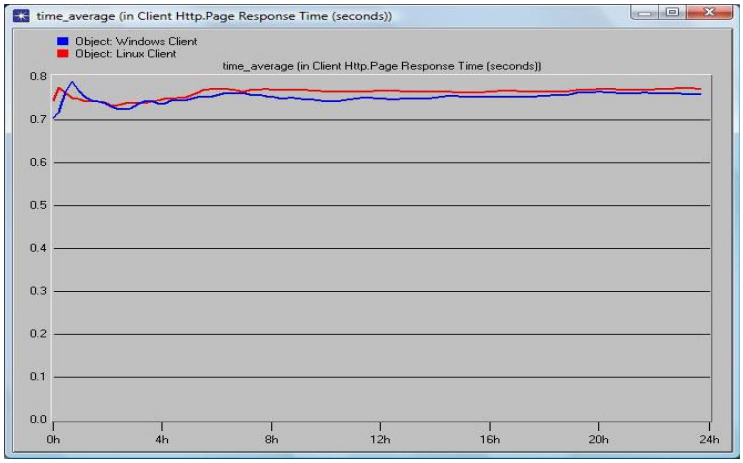

Figure 8: Comparison of HTTP Page Response Time between Windows and Linux nodes (SSL)

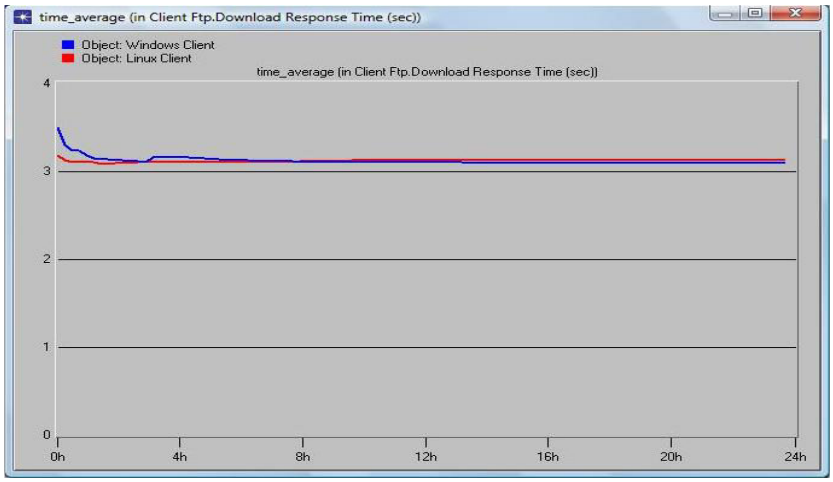

Figure 9: Comparison of FTP Download Response Time between Windows and Linux nodes (SSL)

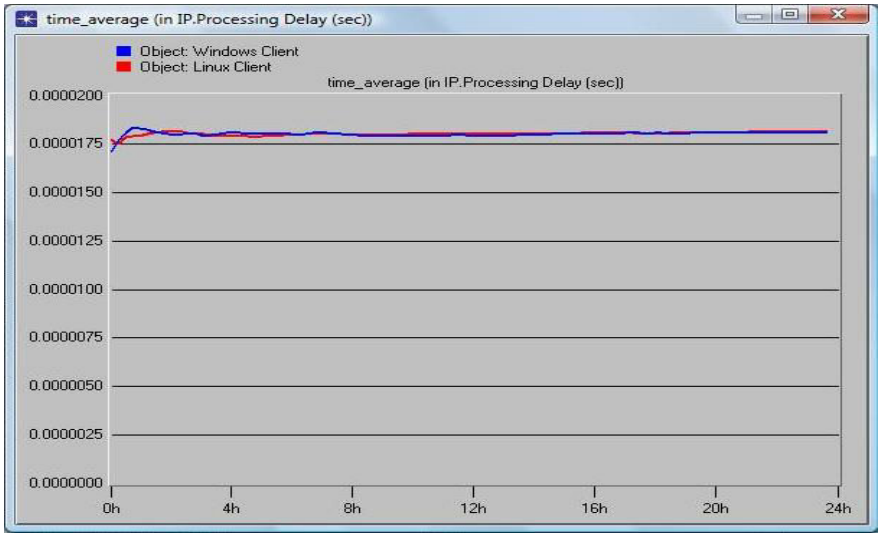

Figure 10: Comparison of IP Processing Delay between Windows and Linux nodes (SSL)