International Journal of Scientific & Engineering Research, Volume 3, Issue 11, November-2012
ISSN 2229-5518

Overhead of FTPS and FTP over IPsec in IPv6 networks

Eng. N Pradeep Ruwan Nawarathne

Abstract— FTP is client-server architecture based protocol for transferring files over TCP/IP networks. It is used with either user-based password authentication or anonymous user access. FTP has no encryption support. All transmissions are in clear text format. User names, passwords, FTP commands and transferred files can be read by anyone who is sniffing the network. FTPS and FTP over IPsec are cost effective methods of securing FTP. These methods provide much better authentication and encryption functionalities for file transfer communication. Besides this, these methods introduce processing overhead and packet overhead for file transferring to different extends. This research paper compares and contrasts the overhead of secure file transfer methods - FTPS and FTP over IPsec - in IPv6 networks.

Index Terms— Cryptographic protocols, FTP, FTPS, IPsec, IPv6, SFTP, TCP/IP networks.

1 Introduction

Authentication, Integrity and Confidentiality are three basic requirements which must be satisfied when securing file transfer communications [1]. But each requirement is not always necessary to fulfill in every context. In some cases the integrity requirement is the most important and in other cases confidentiality. FTPS [2] and FTP over IPsec [3] can be configured to match different security requirements of secure file transfer communications. When choosing either FTPS or FTP over IPsec, it is needed to consider processing overhead, packet overhead and key management delays in these protocols. This research paper would be helpful to identify the potential overheads in FTPS and FTP over IPsec in IPv6 networks.

1.1 Security Issues Associated With FTP

FTP [1] is used with usernames and passwords that are sent to server in clear text format to authenticate clients via the USER and PASS commands except for services such as "anonymous" FTP archives [4]. Using a sniffer to monitor FTP traffic on local or wide-area networks, and then reading user names, passwords, FTP commands and transferred files could be done by a potential attacker. Bounce attacks, spoof attacks, brute force attacks, sniffing and port stealing are security problems associated with FTP [5]. HTTP, SMTP and Telnet also have this security problem because of these protocol specifications written prior to the creation of SSL [6]. Existing solutions for these problems are Secure Copy (SCP) [7], SSH file transfer protocol (SFTP) [8], FTP with the SSL/TLS (FTPS) [2] and FTP over IPsec.

1.2 FTP Secure (FTPS)

FTPS is used to implement security and authentication for FTP clients and servers using the TLS protocol defined by RFC 2246 [2]. FTPS is an security extension to the FTP which was introduced in RFC 2228.It is intended to provide TLS support for FTP in a similar way to that provided for SMTP in RFC 2487 (SMTP Service Extension for Secure SMTP over Transport Layer Security) and HTTP in RFC 2817 (Upgrading to TLS Within HTTP/1.1.) [2]. It communicates across a network in a private and secure fashion discouraging eavesdropping, tampering and message forgery. It includes full support for the TLS and SSL cryptographic protocols, including the use of server-side public key authentication certificates and client-side authorization certificates. It also supports compatible ciphers including AES [9], Triple DES [10], DES [11] and hash functions such as SHA [12] and MD5 [13]. In this research SHA-1 and AES-CBC with 128 bit key was used in FTPS as authentication and encryption algorithms respectively. Explicit mode (FTPES) and implicit mode (FTPS) are two separate methods were developed to invoke FTP client security. In Implicit Mode, the entire FTPS session is encrypted. In Explicit Mode client has full control over what areas of the connection are to be encrypted [14]. FTPS was used in explicit mode in this research.

• N. Pradeep Ruwan Nawarathne, Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka. PH-(+94)777461027. E-mail: prnawa@gmail.com

1.3 Internet Protocol Security (IPsec)

IPsec [15] is the Security Architecture for internet protocol defined by the IETF. It provides four security properties such as authentication, data integrity, confidentiality and replay protection. IPsec support is mandatory in IPv6 but not in IPv4, where it is optional. IPsec, however, is not widely used at present except for securing traffic between IPv6 Border Gateway Protocol routers [16].Upper layer protocols such as FTP, HTTP and IP communication can be secured from the security features of IPsec. Moreover, security provided by the IPsec is transparent to upper layer protocols. It operates on top of IPv4/IPv6. Security can either be end-to-end or take some intermediate security gateways as end-points of the secured channel.

IPsec contains two security protocols: the Authentication Header (AH) [17] and the Encapsulated Security Payload (ESP) [18]. The ESP protocol can be combined with the AH. The IPsec possesses two different protocol modes: Tunnel Mode and Transport Mode. In this research AH and ESP was used in transport mode. IPsec in transport mode provides end-to-end protection of the payload. AH and ESP can each be used equally in Tunnel and Transport mode [16].

2 IPSEC SECURITY MECHANISMS

2.1 Authentication Header (AH)

AH provides connectionless integrity, data origin authentication, and replay protection. But it doesn't provide confidentiality [17]. Its scope is the whole static part of the IP packet including non-volatile IP header fields, the payload and the AH specific header information. The Authentication Header protocol can utilize different authentication algorithms. Message Authentication Codes (MAC) protects point-to-point connections. One way hash functions such as MD5, SHA-1 or symmetric encryption algorithms such as 3DES, Blowfish [19], AES can be used. In this research SHA-1 was used as the authentication algorithm. The length of the Authentication header is 12 bytes plus the length of the authentication data [16].

2.2 Encapsulated Security Payload (ESP)

ESP protocol may provide confidentiality. It also may provide connectionless integrity, data origin authentication, and an anti-replay service [15]. ESP operates on the payload of the packet and provides additional protection for ESP header fields. It does not cover static fields of the IP header. The ESP header is inserted after the IP header and before transport layer data or encapsulated IP packets in the Tunnel Mode. Confidentiality can be specified alone, whether in this circumstance other means for authentication like the AH protocol are required for secure operation. Traffic flow confidentiality can be achieved in conjunction with the Tunnel Mode. The optional authentication uses the same functions for MAC generation as the AH. The already stated encryption algorithms serve for the confidentiality of the payload. In this research AES-CBC with 128 bit key was used as the encryption algorithm. The length of the Encapsulated Security Payload header is 10 bytes plus the length of the padding, optionally, the length of the authentication data and the length of an Initialization Vector (usually 8 bytes) if required by the encryption algorithm [16].

2.3 Security Association (SA)

In RFC 2401, SA is defined as a simplex "connection" that affords security services to the traffic carried by it. Security services are afforded to a SA by the use of AH, or ESP, but not both. If both AH and ESP protection is applied to a traffic stream, then two (or more) SAs are created to afford protection to the traffic stream. To secure typical, bi-directional communication between two hosts, or between two security gateways, two Security Associations (one in each direction) are required [15]. SA is uniquely identified by a triple consisting of a Security Parameter Index (SPI), an IP Destination Address and a security protocol (AH or ESP) identifier.

2.4 SAD and SPD

In each IPsec implementation there is a nominal Security Association Database (SAD), in which each entry defines the parameters associated with one SA. Each SA has an entry in the SAD .The Security Policy Database (SPD) stores policies that define which inbound and outbound traffic must be protected by what security services. The SPD defines how SAs must be established and what parameters are necessary [15].

2.5 Key Management

SPD is usually managed manually by system administrators. The Internet Key Exchange (IKE) is used to establish authenticated keying material and maintaining SAs. It runs on top of the ISAKMP [20] framework and utilizes the Diffie-Hellman [21] algorithm to set up a shared session key. It can use pre-shared secrets or X.509 certificates to authenticate the parties. The IETF is currently standardizing the Internet Key Exchange (IKEv2) Protocol [21] which integrates previously independent standards such as ISAKMP and introduces new functionalities such as NAT traversal, Legacy Authentication, Remote Address Acquisition. However, IKEv2 is not interoperable with IKE. The overhead of key negotiation decreases considerably as multiple transport protocols and applications can share the key management infrastructure provided by the network layer.

2.6 FTP over IPsec

FTP over IPsec is a well-known method of securing FTP traffic. In transport mode, it enables server to client and client to server security so that every piece of file transfer communication can be secured. Security provided by IPsec is transparent to FTP applications. Although FTP over IPsec has more features than FTPS it is more often difficult to implement and require special support in routers. IPsec with AH and ESP in transport mode was used in this research to secure the FTP traffic in an IPv6 Local Area Network.

3 METHODOLOGY

3.1 Overview of the Methodology

Investigating the overhead of FTPS and FTP over IPsec was decomposed into six steps: (i) setting up an IPv6 test bed, (ii) setting up client and server applications for FTP, FTPS and FTP over IPsec on that test bed, (iii) transferring variety of different sizes of files between server and client using FTP, FTPS and FTP over IPsec, (iv) measure the file transfer time for each case (transfer time is used to calculate overhead), (v) aggregating and comparing the data obtain from test bed, and calculating the overhead of FTPS and FTP over IPsec relative to the FTP and (vi) obtaining statistically best fitted models for overhead of FTPS and FTP over IPsec from statistical analysis using calculated values of overheads.

3.2 IPv6 Test bed

Fig. 1 illustrates the IPv6 test bed which was used in this research for measure the overhead of FTPS and FTP over IPsec. Host operating systems have fully IPv6 support. Each Ethernet Network Interface was only configured work with IPv6 protocol. Each host was given a global IPv6 address. A 10-Mbps Ethernet local area network is used to connect the server, client and Sniffer. Sniffer was used to monitor the network traffic.

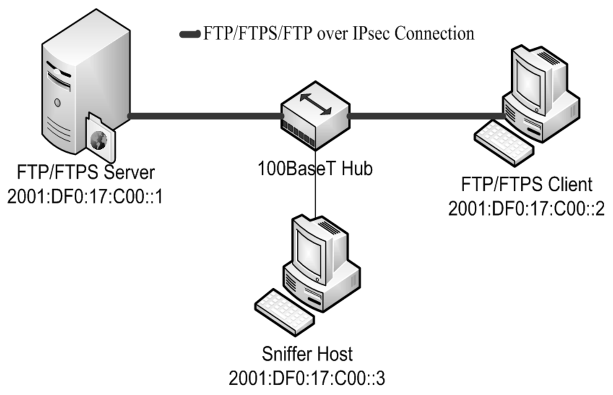

Fig. 1. IPv6 test bed used to measure the overhead of FTPS and FTP over IPsec.

3.3 Hardware

Each computer on the test bed had Intel Pentium 4 Processor (1.8 GHz), 512MB RAM and 10/100Mbps Realtek RTL8139 family PCI Fast Ethernet NIC. Other hardware configuration parameters are same for both server and client.

3.4 Software

Ubuntu 9.10 server edition was used as server operating system and Ubuntu 9.10 desktop edition was used as client operating system with Linux kernel version 2.6.31.

Pure-FTPd is a free (BSD), secure, production-quality and standard-conformant FTP server [22]. It supports both FTP and FTPS protocols. In this research Pure-FTPd was used as FTP and FTPS server. Pure-FTPd was used as a FTP server with FTP relevant configurations and as a FTPS server with a self-signed certificate and FTPS relevant configurations. FTPS server was operated only in Explicit Mode.

FileZilla is the file transfer client used in this research. It Supports FTP, FTP over SSL/TLS (FTPS) and SSH File Transfer Protocol (SFTP) [23]. Pure-FTPd and FileZilla have fully IPv6 support. Same file transfer server (Pure-FTPd) and client (FileZilla) applications were used in order to neglect the overhead due to the implementation differences of applications.

IPsec tools [24] were used to implement native IPsec support in both client and server. Wireshark packet analyzer [25] was used to analyze the packets of file transfer communications.

3.5 Implementing IPsec

IPsec tools were used to implement native IPsec support in Linux kernel version 2.6.31. setkey is an IPsec tool which was used to store and modify all parameters in the SAD and the SPD in this research. IPsec connection in transport mode with AH and ESP was used to provide authentication and encryption for IPv6 packets respectively. Fig. 2 demonstrates a structure of IPv6 packets which were transferred through the created IPsec connection.

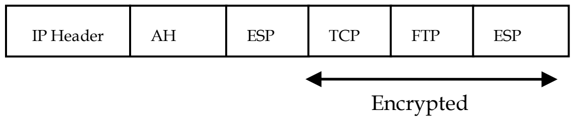

Fig. 2. Structure of an IPv6 packet with native IPsec support.

3.6 Data Acquisition

File transfer scenarios, files and protocols which were used for the data acquisition will be described here. There are three major file transfer scenarios in this research: (I) FTP file transfer, (II) FTPS in Explicit mode file transfer (III) FTP over IPsec in transport mode file transfer. Each key scenario can be divided in to another two research scenarios: (D) Server to client file transfer (download) and (U) Client to server file transfer (upload). All together there were six research scenarios such as download a file using FTP (I, D), uploading a file Using FTP over IPsec (III, U), etc.

In order to get five test data samples for each scenario, each one was conducted five times. All together there were thirty test data samples. The metric that was used in each scenario was the transfer time between the server and the client in seconds. The files, which were used to transfer in between client and server, were 1KB, 10KB, 100KB, 1MB, 10MB, 100MB and 1GB in size. Furthermore, those were in RAR compressed format. In scenarios (II) and (III) AES-CBC with 128 bit key and SHA-1 cryptographic algorithms were used for encryption and authentication respectively. The Internet Protocol version 6 was used in all scenarios.

3.7 Calculating the Overhead

File transfer time was used to calculate the overhead of FTPS and FTP over IPsec. Following equation was used to calculate the overhead in scenario (N, n) with respect to scenario (I, n).

$$\text{Overhead}_{(N,n)} = f\left(t_{(N,n)}, t_{(I,n)}\right) \qquad (1)$$

$$f\left(t_{(N,n)}, t_{(I,n)}\right) = \left[\frac{\left(t_{(N,n)} - t_{(I,n)}\right)}{t_{(I,n)}}\right] \text{x } 100\% \qquad (2)$$

D = Download a file; U = Upload a file; I = FTP, II = FTPS; III = FTP over IPsec
N = {I, II, III}; n = {D, U}
(N, n) = {(I, D), (I, U), (II, D), (II, U), (III, D), (III, U)}
(I, D): Downloading a file using FTP scenario
(II, U): Uploading a file using FTPS scenario
$t_{(N,n)}$: Mean file transfer time for scenario (N, n)
$\text{Overhead}_{(N,n)}$: Calculated overhead of scenario (N, n)

Outputs of the (2) were used to plot graphs and obtain statistical models for each research scenario. So that it was helpful for understanding and forecasting the potential overheads of FTPS and FTP over IPsec with respect to FTP.

4 Results and Discussion

4.1 Mean File Transfer Time ($t_{(N,n)}$)

Table 1 illustrates the mean file transfer time for each scenario with relevant file sizes. Mean values were calculated from acquired test data samples. File sizes and mean file transfer time were measured in KB and seconds respectively.

4.2 Calculated Overhead

Table 2 shows the calculated overhead using (2) for FTPS and FTP over IPsec with respect to each file size. File sizes was measured in KB.

TABLE 1
Mean File Transfer Time for each file transfer scenarios for FTP (I), FTPS (II) and FTP over IPsec (III)

File Size (KB)	Mean File Transfer Time (s)					
	$t_{(I, D)}$	$t_{(I, U)}$	$t_{(II, D)}$	$t_{(II, U)}$	$t_{(III, D)}$	$t_{(III, U)}$
1	0.0148	0.0134	0.0182	0.0148	0.0208	0.0148
10	0.0184	0.0228	0.02	0.0236	0.0216	0.0252
100	0.0192	0.0324	0.022	0.034	0.0236	0.0358
1024	0.1196	0.1182	0.1246	0.127	0.1522	0.1354
10240	0.9842	0.9358	1.0416	0.9506	1.1894	1.0044
102400	9.328	9.1374	9.9742	9.4798	10.8784	9.8484
1048576	96.1596	97.8614	198.5954	104.677	107.822	99.9078

D = Download file; U = Upload file; I = FTP, II = FTPS; III = FTP over IPsec (I, D) = Downloading a file using FTP scenario; (II, U) = Uploading a file using FTPS scenario.

4.3 Data Analysis

TABLE 2
Calculated Overhead of FTPS (II) and FTP over IPsec (III)

File Size (KB)	$\text{Overhead}_{(N,n)}$			
	(II,D)	(III,D)	(II,U)	(III,U)
1	22.973	40.5405	10.4478	16.4179
10	8.6957	17.3913	3.5088	10.5263
100	14.583	22.9167	4.9383	10.4938
1024	4.1806	27.2575	7.445	14.5516
10240	5.8321	20.8494	1.5815	7.3306
102400	6.9275	16.6166	3.7472	7.7812
1048576	106.53	12.1256	6.968	2.0917

D = Download file; U = Upload file; I = FTP, II = FTPS; III = FTP over IPsec (II, D) = Downloading a file using FTPS scenario; (II, U) = Uploading a file using FTPS scenario; (II, D) = Overhead of FTPS file download compare to FTP file download; (III, U) = Overhead of FTP over IPsec file upload compare to FTP file upload.

Fig. 3 visualizes the graphical representation of calculated overhead for scenario (II, D) and (III, D). According to the Fig. 3, FTPS has a lower overhead compared to the FTP over IPsec when downloading 1KB, 10KB, 100KB, 1MB, 10MB and 100MB size files from server to client. But FTPS has very much high overhead compared to FTP over IPsec when downloading 1GB size files.

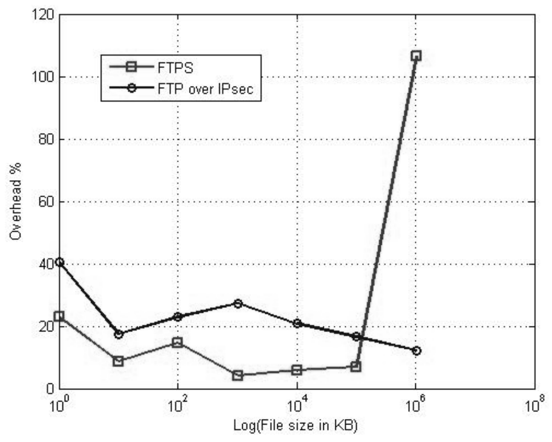

Fig. 3. Calculated overhead for scenario (II, D) and (III, D).

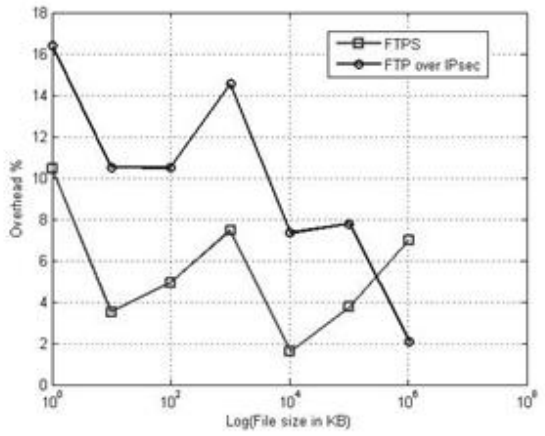

Fig. 4. Calculated overhead for scenario (II, U) and (III, U)

Fig. 4 visualizes the graphical representation of calculated overhead for scenario (II, U) and (III, U). According to the Fig. 4, FTPS has a lower overhead compared to the FTP over IPsec when upload 1KB, 10KB, 100KB, 1MB, 10MB and 100MB size files. But FTPS has higher overhead compared to FTP over IPsec when uploading 1GB size files from client to server.

There is a same pattern in both Fig. 3 and 4. That is FTPS and FTP over IPsec always has a positive overhead and initially FTP over IPsec has higher overhead compared to FTPS. But gradually it decreases with file size. FTPS has lower overhead for small size file transfers. In between 100MB to 1GB it changes that behavior and introduces a higher overhead compared to FTP over IPsec.

In summing-up, both FTPS and FTP over IPsec introduces considerable amount of overhead when transferring files that are smaller than 10 Mb. Due to the fact that file downloading overhead is always greater than file uploading overhead, it is very interesting to understand behavior of download overhead in both FTPS and FTP over IPsec. Furthermore, when file size increases overhead tends to decrease in most scenarios. After a certain point (file size: 100 Mb) that behavior does not continue. There is a rapid increase of overhead in FTPS download beyond that point whereas overhead in FTP over IPsec download declines steadily.

4.4 Linear Models

Four linear models were obtained from statistical analysis - Linear regression - of calculated overhead values for scenario (II, D), (III, D), (II, U) and (III, U). So that those linear model are very useful in better understanding and predicting potential overheads of FTPS and FTP over IPsec. The parameter "size" represents the transferring file size in KB.

$$Overhead_{(II,D)} = -1.3 + 3.69 \log_{10}(size) \qquad (3)$$

$$Overhead_{(III,D)} = 32 - 1.37 \log_{10}(size) \qquad (4)$$

$$Overhead_{(II,U)} = 6.94 - 0.205 \log_{10}(size) \qquad (5)$$

$$Overhead_{(III,U)} = 15.4 - 0.798 \log_{10}(size) \qquad (6)$$

The overhead in scenario (II, D), (III, D), (II, U) and (III, U) can be model using (3), (4), (5) and (6) respectively.

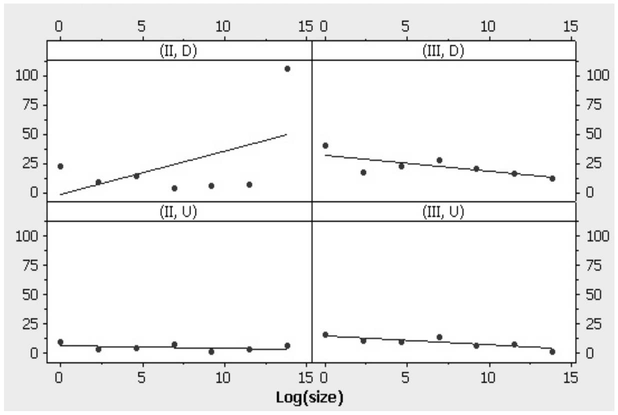

Fig. 5.Linear models of scenario (II, D), (III, D), (II, U) and (III, U)

Fig. 5 shows a graphical representation of above four models. These statistical models are not always correct and efficient but those are useful for identify potential overhead in FTPS and FTP over IPsec. Fig. 5 illustrates the same idea that is mentioned in section 4.3 data analysis summery.

5 Conclusion

According to obtained models FTP over IPsec has lower overhead compared to FTPS for larger size files. But for a smaller size file FTPS is a good solution for securing File transfer communication. It cannot be guaranteed that this is true for all environments. In this research we tried to get an idea of potential overheads of FTPS and FTP over IPsec in IPv6 LAN environment with Ethernet technologies. Router processing overhead is not involved in this environment. That overhead is not included in our calculation. But most of production environments have that processing overhead. Processing overhead may be varying with FTPS and FTP over IPsec. So selecting the right security solution for file transfer with different settings and environments can be difficult. Not all solutions fit into a specific network or host configuration. Knowing the overheads of FTPS and FTP over IPsec in IPv6 LAN environment is very important.

6 Future Work

Overhead due to key management in FTPS and FTP over IPsec is not discussed in this paper, hence it will be a future work on this area. This research paper can be extended by considering overhead of alternative key management protocols.

Acknowledgment

This research was facilitated by Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka.

References

[1] J. Postel and J. Reynolds. File transfer protocol (ftp). Internet Draft (RFC 959), 1985.
[2] P. Ford-Hutchinson. Securing FTP with TLS. Internet Draft (RFC 4217), 2005.
[3] R. Atkinson. Security Architecture for the Internet Protocol. Internet Draft (RFC 1825), 1995.
[4] M. Horowitz and S. Lunt. FTP Security Extensions. Internet Draft (RFC 2228), 1997.
[5] M. Allman and S. Ostermann. FTP Security Considerations. Internet Draft (RFC 2577), 1999.
[6] M.P. Clark. Data Networks IP and the Internet. 1st ed. West Sussex, England: John Wiley & Sons Ltd. 2003.
[7] Manual Pages: scp(1). Available at: http://www.openbsd.org/cgi-bin/man.cgi?query=scp&sektion=1 [Accessed March 30, 2010].
[8] T. Ylonen and C. Lonvick, Ed.The Secure Shell (SSH) Protocol Architecture. Internet Draft (RFC 4251), 2006.
[9] Christof Paar, Jan Pelzl, "The Advanced Encryption Standard", Chapter 4 of "Understanding Cryptography, A Textbook for Students and Practitioners". Springer, 2009.
[10] Triple DES Encryption Available at: http://www.tropsoft.com/strongenc/des3.html [Accessed February 5, 2010].
[11] S. Kelly. Security Implications of Using the Data Encryption Standard (DES). Internet Draft (RFC 4772). 2006.
[12] SECURE HASH STANDARD Available at: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf [Accessed January 11, 2010]
[13] R. Rivest. The MD5 Message-Digest Algorithm. Internet Draft (RFC 1321). 1992
[14] FTP Clients - Part 2: Explicit FTPS versus Implicit FTPS : Robert McMurray : The Official Microsoft IIS Site. Available at: http://blogs.iis.net/robert_mcmurray/archive/2008/11/10/ftp-clients-part-2-explicit-ftps-versus-implicit-ftps.aspx [Accessed April

28, 2010].

[15] Stephen Kent and Randall Atkinson. Security architecture for the internet protocol. IETF, 1998.

[16] Heiko Niedermayer, Andreas Klenk, and Georg Carle. The Networking Perspective of Security Performance.

[17] Stephen Kent and Randall Atkinson. Ip authentication header. Internet Draft (RFC2402), 1998.

[18] Stephen Kent and Randall Atkinson. Ip encapsulating security payload (esp). Internet Draft (RFC2406), 1998.

[19] Blowfish. Available at: http://www.schneier.com/blowfish.html [Accessed December 28, 2009].

[20] D. Maughan, M. Schertler, M. Schneider and J. Turner. Internet Security Association and Key Management Protocol (ISAKMP). Internet Draft (RFC 2408), November 1998.

[21] P. Ford-Hutchinson. Diffie-Hellman Key Agreement Method. Internet Draft (RFC 2631), 1999.

[22] Pure-FTPd - About. Available at: http://www.pureftpd.org/project/pure-ftpd [Accessed April 3, 2010].

[23] FileZilla - Client Features. Available at: http://filezilla-project.org/client_features.php [Accessed February 20 , 2010].

[24] IPsec Tools Homepage. Available at: http://ipsec-tools.sourceforge.net/ [Accessed December 11, 2009].

[25] Wireshark. Available at: http://www.wireshark.org/ [Accessed April 18, 2010].