EFFICIENT ASYMMETRIC IPSEC FOR SECURE ISCSI

Murthy S. Andukuri and C. Edward Chow
Department of Computer Science, University of Colorado at Colorado Springs
1420 Austin Bluffs Parkway, Colorado Springs, CO 80918, U.S.A.

Keywords: IPSec, Asymmetric Secure Protocol, iSCSI, Online Data Backup.

Abstract: In this paper we propose a new asymmetric IPsec scheme to enhance the security of data at the remote end, while simultaneously improving the overall performance. The idea is to apply IPsec encryption/decryption in a segmented manner on the iSCSI traffic, such that the user data remains encrypted after leaving the sender, and is decrypted only when it is retrieved by the sender. A dual key cryptographic scheme is proposed where the private key is used to encrypt the iSCSI payload at the sender and traditional IPsec is modified to encrypt/decrypt only on the TCP/iSCSI headers. A development test bed was built using User-Mode-Linux virtual machines for developing and debugging the asymmetric IPsec software and running as the sender and receiver to verify the functionality and security features of the proposed design. A benchmark test bed was built with two real PCs where the asymmetric IPsec modules can be dynamically loaded. The performance results show that the existing implementation of the proposed asymmetric IPsec scheme reduces the IPsec processing time by about 25%.

1 INTRODUCTION

Remote backup of data for security has become a subject of rapidly growing interest in the recent times (Kirk, 2006). The importance of backups, and remote storage for security in today's networked world can hardly be overstated. Of the various options available, iSCSI seemed the most worthy of study because its design smartly makes full use of the universally proven strengths of existing protocols like TCP, IP and IPsec, thereby reducing the cost, effort and time of learning, setup and deployment. The various mechanisms that can be used are FCIP, iFCP, iSCSI (Clark, 2002) (Shurtleff, 2004). Among these, iSCSI has been getting a lot of attention of late because it can be run on commonly available, relatively inexpensive IP networking infrastructure already in place. iSCSI is an application layer protocol that uses the available IP network to make a remote storage disk accessible as a simulated local SCSI disk.

This locally accessible remote disk can be written to, or read from, like any local disk. An iSCSI setup has two parts - The iSCSI initiator is the 'client' program located on the source machine and writes to / read from the remote machine. The iSCSI target is the software on the destination machine that helps store the data and return it on demand. iSCSI restricts itself to handling the user-level data and leaves the actual details of transmission to the TCP and IP layers. By default, the data is transmitted in plain text between the initiator and the target. This vulnerability can be remedied by using IPsec to secure the data in transit.

IPsec is designed to provide interoperable, high quality, cryptographically-based security for IPv4 and IPv6. The set of security services offered includes access control, connectionless integrity, data origin authentication, protection against replays (a form of partial sequence integrity), confidentiality (encryption), and limited traffic flow confidentiality. These services are provided at the IP layer, offering protection for IP and/or upper layer protocols (RFC2401).

IPsec encrypts the data leaving the network layer on the sender and, at the receiving end, decrypts the data before it leaves the network layer. This secures the data in transit but does not help secure the data AFTER it has reached its destination. This makes data very vulnerable to theft when the target site gets break-in. This vulnerability can be alleviated by reencrypting the received data using a third party software – and redecrypt, so that the IPsec layer can encrypt it in preparation for transmission back to the sender. Figure 1 shows such a scenario.

S. Andukuri M. and Edward Chow C. (2010).
EFFICIENT ASYMMETRIC IPSEC FOR SECURE ISCSI.
In *Proceedings of the International Conference on Security and Cryptography*, pages 314-317
DOI: 10.5220/0002996903140317

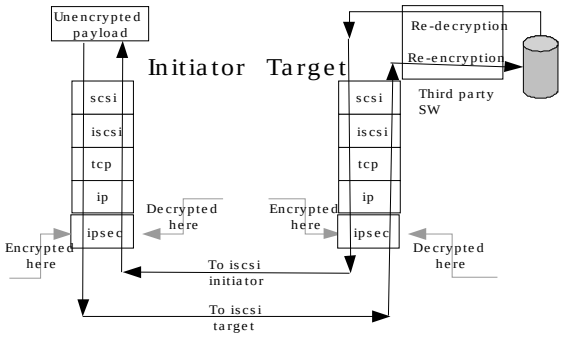

Figure 1: Re-encryption/decryption at target site.

This situation presents the following issues:

- The data is in an un-secured form on a remote disk just after being received, and just before being transmitted.
- This scheme involves three encryptions and three decryptions that increase the computational and operational costs.
- The third party software involves extra cost.

Another solution (Shurtleff, 2004) is to use application layer software to encrypt the data on the sender, store it in the encrypted state on the receiver and decrypt it only on retrieval. This scheme also involves three encryptions and three decryptions. However, this is better than that of Figure 1 described above, because the data is never in an un-encrypted state outside of the Initiator. This presents two choices, both of which have issues of their own.

Scenario 1: Use an application layer software to encrypt user data, and transmit it without IPsec. Figure 3 shows this scenario. This leaves the iSCSI, TCP and IP headers exposed during transit. While the data is encrypted, the headers remain vulnerable.

Scenario 2: Use an application layer software to encrypt user data and decrypt it after retrieval. Transmit using IPsec.

This secures the TCP and iSCSI headers (and optionally the IP header as well). However, this also involves

- RE-encryption of the encrypted payload on the sending side,
- Decryption of the same on the receiving side to undo the above encryptions
- RE-encryption of the encrypted payload on the receiving side for retrieval by sender
- Decryption of the same on the sender (after retrieval) to undo the above, second encryption.

As such, it is obvious that this scheme only partially addresses the shortcomings of the previous approach.

The proposed efficient asymmetric IPsec scheme hopes to address the above concerns as follows. It is proposed that the process of encrypting/decrypting the transmitted data be divided into two parts:

• The encryption of the TCP and iSCSI headers is performed per the normal IPsec procedures – using the keys generated and managed by Internet Key exchange (IKE) between cthe source and destination.

• The core IPsec encryption functionality, i.e., the algorithm implementation excluding the IKE, is still used to encrypt the user data. However, the key for the encryption is generated on the source machine independent of the IKE mechanism. This key will NOT be shared with the destination.

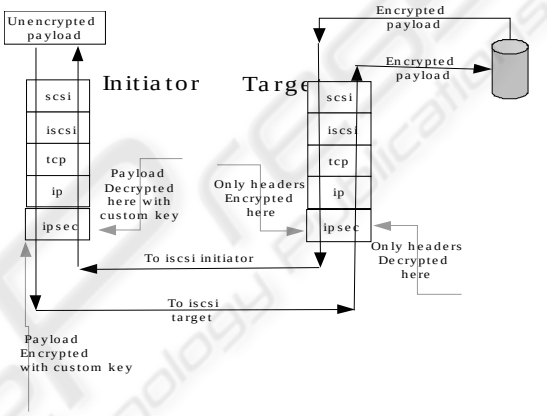

Figure 2: Proposed AsymmetricIPsec Scheme.

At the destination only the TCP, iSCSI header are decrypted per normal IPsec process to extract the iSCSI details and to write the user data, which continues to be in encrypted form, to the remote disk. During retrieval by the sender, the user data is returned in the same encrypted form and accompanied by headers that are now appropriately encrypted by IPsec on the target. Upon arrival, the headers are decrypted per the normal IPsec scheme, i.e. using the keys mutually agreed upon through the IKE mechanism. The user data is now decrypted using the core IPSec decryption functionality but with the customized, locally generated key that was used originally to encrypt the data. Figure 2 shows the proposed scheme.

2 ASYMMETRIC IPSEC FOR ISCSI PROTOCOL DESIGN

The essentials of the Asymmetric IPsec for iSCSI scheme are detailed as follows:

2.1 When the Initiator is Sending iSCSI Date to the Target

- In the 'sending side' code of the IPsec layer on the initiator, identify and isolate the user data in the network traffic going to the target .
- Encrypt the rest of the traffic (i.e. All traffic other than the user data) using the standard IPsec mechanism, using keys generated and managed by the IKE.
- Use a custom key, generated independently of the IKE mechanism, to encrypt the user data. Do not share this key with the target. Save this key for future use to decrypt the same user-payload when it is returned.
- At the target, decrypt the headers using standard IPsec procedure, but do not attempt to decrypt the user payload. Pass it in the encrypted form to the upper layers so that the SCSI layer can write it as is (in the encrypted form) to disk.

2.2 When the Initiator is Trying to Read the iSCSI Data from the Target

- On the target, encrypt the headers using standard IPsec mechanism. Do not attempt to encrypt the user payload.
- On the initiator, decrypt the headers using the keys generated and managed by IKE. Use the second, custom key originally used to encrypt the user data, to decrypt the data.
- In order to come up with an implementation of this scheme, the pattern of the flow of packets between the initiator and the target, WITHOUT IPsec was studied to understand the exact sequence of packets – both when writing to the target and when reading from the target. The study threw a surprise. When the initiator is writing, the user payload is carried as a part of Data-out PDUs. When the initiator is reading, the user-payload is carried in a plain-vanilla TCP packet. A packet with a Data-in PDU precedes this packet. Even more surprisingly, the *DataSegmentLength* field of the Data-In PDU reflects the length of the user payload, even though the payload is actually carried by a separate packet. The author could not find an answer for this behavior nor a way to change it so that a Data-in PDU contains the user payload. Hence the solution implemented was designed accordingly.

The proposed scheme entails changes to the IPsec-specific code in the Linux 2.6 network stack. To understand how the actual code-modification scheme was arrived at, it helps to recap how data would be handled by the IPsec code in its native form.

2.3 The Native IPsec operation on iSCSI

The IPsec scheme used in the current thesis is called 'transport' mode. This means an ESP header is inserted between the IP header and the TCP header. The 'protocol' field in the IP header is changed by the IPsec layer to '50' to indicate the presence of an ESP header following the IP header. Prior to the 'encryption' part of the IPsec code, this 'protocol' field of the ip header was populated with ‘6’, which is 'TCP'. This information is saved in the IPsec layer, before the 'protocol' field is overwritten with '50'. The saved value will be entered later in the last byte of the padding that is going to be added at the end of the payload. The iSCSI header together with the user data forms the payload for the TCP layer. The TCP header plus the iSCSI payload, in turn forms the payload for the IPsec protocol. This IPsec payload is padded so that the total length (tcp header + IPsec payload + padding) is an exact multiple of the block size of the encryption algorithm being used. Care is taken to make sure that the padding is at least 2 bytes long. The last byte of the padding, is set to the protocol ID saved earlier. The last-but-one byte is set to the total number of padding bytes (Hence the need to make sure the padding is at least 2 bytes long). The TCP header, iSCSI header, iSCSI payload and the ESP trailer are together encrypted as one unit. The padding forms the ESP trailer. An ESP authentication trailer is inserted after the ESP-trailer. This trailer contains the cryptographic checksum of

Ipheader + esp header + tcp header + iSCSI header + iSCSI data + esp trailer.

The authentication trailer is NOT encrypted.

On the receiving end, the cryptographic checksum is recomputed on the same components as mentioned earlier. This is compared to the value stored in the ESP authentication trailer. The packet is rejected if they do not match. If they are found to be matching, the code proceeds to decrypt the tcp header + iscsi header + iscsi data + esp trailer. After decryption, the esp header placed between the IP header and the tcp header is removed. The '50' in the 'protocol' field of the IP header is replaced by the value in the last byte of the padding. The last-but-one byte of the total payload (which is the length of the padding) gives the number of padding bytes to be stripped.

3 PERFORMANCE ANALYSIS

We have set up a User Mode Linux (UML) virtual machines test bed with our Asymmetric IPsec implementation.

The available alternative involves the following computations, given in terms of 16 byte blocks – the block size for the AES encryption algorithm. Table 1 reports the respective durations taken for encryption and decryption during the round-trip of a single TCP segment of 1024 bytes. In the table, the TCP header is shown as consisting of two 16-byte blocks. This has been done for two reasons.

In the proposed scheme, the TCP header + iSCSI header unit needs to be an integer multiple of the block size. Given that the iSCSI header is fixed at 48 bytes (which happens to be an integer multiple of 16 bytes), even if the TCP header were to have the smallest possible size of 20 bytes, the TCP header still needs to be padded with of 12 bytes so that the sum of TCP header size + iSCSI header size comes to be an integer multiple of the block size. Incidentally, the TCP header on the virtual machines was indeed observed to be 32 bytes long.

For the other scheme, even if the TCP header were to be the smallest possible size of 20 bytes, the fact remains that the total of TCP header + iSCSI header + payload needs to be padded to become an integer multiple of 16 bytes. The total number of 16-byte blocks does not change. From the above numbers, it is obvious that the proposed scheme is expected to take only 36% (74/202) of the other scheme. This gain in efficiency, combined with the fact that the data never is left unencrypted outside of the initiator, makes the proposed scheme attractive.

Table 1: Number of 16-byte blocks encrypted during round-trip of 1 TCP segment.

Under Available Options (3*64 + 2 *(2+3))		**Under Proposed Scheme (64 + 2*(2+3))**	
Encrypted	**202 blocks**	**Encrypted**	**74 blocks**
Decrypted	**202 blocks**	**Decrypted**	**74 blocks**

From the experiments conducted on the test bed, we observed as the file size increases from 1M and on, the performance gains gravitate towards 25% - 30% range. However, this is less than the expected 65% gain as mentioned earlier. Running a profiler on the implementation of the proposed scheme might throw more light on whether there is room for improvement.

4 CONCLUSIONS

An efficient asymmetric IPSec protocol enhancement was proposed for reducing the processing time and improving security of secure iSCSI based online-backup systems. A development test bed was constructed using UML virtual machines to facilitate the development/debugging of IPSec kernel/networking code. A benchmark test bed with two real PCs was installed with the new modified IPSec module and a set of test runs were made to collect the performance data of the proposed system. The analysis of the data from the UML test bed does not show the expected performance gains but running the same trials on actual machines shows performance gains in the 25%-30% range.

REFERENCES

Kirk, J., 2006, “Symantec unveils remote data backup software” by Jeremy Kirk, http://www.computerworld.com/securitytopics/ security/story/0,10801,110148,00.html

Clark, T., 2002, “IP SANs: A Guide to iSCSI, iFCP, and FCIP Protocols for Storage Area Networks”. Addison Wesley Professional, 2002.

Shurtleff, J., 2004, “IP storage: A review of iSCSI, FCIP, iFCP,” 2004. http://www.iscsistorage.com/ipstorage.htm

RFC2401, 1998, “Security Architecture for IP,” Kent & Atkinson.

†: This research work was supported in part by two NISSSC AFOSR grant awards under numbers FA9550-06-1-0477 and FA9550-04-1-0239.

*: Murthy Andukuri is now with Verizon Business. This work was done as part of his master thesis.