CS 425: Project Report

IPSec: Authentication and Encryption

Kritika Singh (Y9218) Pankaj Jindal (Y9399)

Nehchal Jindal (Y9366) Suresh Gugulothu (Y9610)

Contents:

Abstract

IPSec is a secure internet data exchange protocol implemented in Network layer which provides security services. It encrypts and authenticates each packet of the communication session encrypted before sending to the other host and is decrypted and authentication headers are verified at the other end.

IPsec consist of two protocols namely AH (authentication headers) and ESP(Encapsulation Security Payload). AH provides data origin authentication and connectionless integrity. ESP provides encryption services.

What we have implemented?

Our main aim is to show the simulation of the IPSec protocol by authenticating and encrypting each packet of communication session. The project includes:

1. Diffie-Hellman public key exchange protocol for negotiation of cryptographic keys to be used during the session for the encryption.
2. The AES encryption algorithm in CBC (Chain block cypher) mode.
3. The MD5 message-digest algorithm, widely used cryptographic hash function that produces 128-bit hash value.

Above algorithm are in the network layer of Internet Protocol Suite.

To show the simulation, the **application layer** consists of a chat program with GUI between client and host.

For the **link layer** BSD sockets have used.

Language and Tools used

The project has been implemented in python language.
For the GUI, Tkinter library has been used is available in python.

Implementation

The implementation is modular with the implementation of Application, Network and Link layer. Each lower layer provides services (function calls) to the above layer.

Application Layer:

We have made two Graphical user interfaces, one for the client side and the other for the server side. We have implemented the basic functionality of setting up of the connection, sending and receiving the data and showing it to the user. The application layer to do all this communicates with the lower layers.

Network Layer:

This is where we implemented the IPSec, here we have applied the following:

Diffie-Hellman Key exchange Protocol: This is one of the most popular key exchange protocols. This is implemented during the setting up of the connection where the client and the server exchange their public keys and generate a common private key to be used in encryption and decryption process.

<u>Description of the algorithm</u>: This uses a prime number (p) and a base(n) to start with. Both these are public values. Then the client on getting connected to the server takes the input of the an integer, the exponent value (expClient) of the client from the user from the terminal and generate a value publicKeyClient = (n^expClient) mod p and sends it to the server.

The server on receiving this value for the first time (checked using a boolen variable), inputs its on exponent value (expServer) from the user on the terminal, and sends the value publicKeyServer = (n^expServer) mod p and sends it back to the client. The client is kept into the blocking listen mode for this. Now, the common private key is (publicKeyClient^expServer) mod p = (publicKeyServer^expClient) mod p.

Both, the parties have now the same private key which will be used to do the encryption.

MD5: Once the secure connection is set, each packet is checked for its authenticity using MD5. MD5 algorithm is applied on the immutable data at the sender's side to generate an Integrity Check Value. This value is sent along with the data as part of the Authentication Header. On the receiver's side, MD5 is applied on the same data to generate ICV for it and it is matched with that came along with AH. If they match, we can be sure that the data is coming from an authenticated source otherwise an attack is suspected.
The Algorithm:
1) Suppose the input is of some length(in bits) b. The message is first padded to make its length is congruent to 448, modulo 512. The first padding bit is 1 and the rest zeroes.
2) A 64 bit representation of the length b of the input is then added to the padded message. The resulting message length is an exact multiple of 512, say N.
3) To compute the message digest, we use a four word buffer [A, B, C, D] each of which is 4 bytes long. Initially, these have some pre-assigned values.
4) For this step we will be using four auxiliary functions that take 3 words as input and output one word. These are:

F(X,Y,Z) = XY v not(X) Z
G(X,Y,Z) = XZ v Y not(Z)
H(X,Y,Z) = X xor Y xor Z
I(X,Y,Z) = Y xor (X v not(Z))

The input message whose length is now N(which is a multiple of 16 bytes) is divided into blocks of 16 bytes. Each block undergoes four rounds of operations on it after which the value of A, B, C, and D is updated and used for the next four rounds of operations on the next 16 byte block.
5) After all the blocks have undergone transformations; the resulting A, B, C and D give us the digest.

AES (in CBC mode): This is one of the most advanced encryption algorithms which is present today. The data is broken into chunks of 16Bytes and each chunk is then encrypted as described below. If the data is not a multiple of 16, we have taken care of that situation by using padding.

<u>Description of AES:</u>

AES requires an Initialization Vector (IV) of the same size of 16 bytes. We generated the IV from the key itself. We use AES in CBC mode to prevent generation of identical ciphertext from packets which have identical data.

The IV is XORed with the first plaintext block before it is encrypted. Then for successive blocks, the previous cipherText block is XORed with the current plaintext, before it is encrypted. AES supports three key sizes: 128 bits, 192 bits, and 256 bits. The default key size is 128 bits. AES uses a different number of rounds for each of the defined key sizes. When a 128-bit key is used, implementations MUST use 10 rounds. When a 192-bit key is used, implementations MUST use 12 rounds. When a 256-bit key is used, implementations MUST use 14 rounds.

Each data block after getting XORed with the previous cipher block or the IV goes through the following steps:

The AES Cipher operates using a varying number of rounds, based on the size of the cipher key as told above. A round of AES consists of the four operations performed in succession: AddRoundKey, SubBytes, ShiftRows, and MixColumns (MixColumns is omitted in the final round)

The similar technique is followed in the decryption part where the order of the functions inside the rounds is different and the original text is generated from it.

Link Layer

The link layer has been implemented using BSD sockets. Function calls have been implemented which provide services to network layer. On the client side, services are connecting to server, sending the data, receiving the data. On the client side, services are setting up the server, accepting the request from clients, sending data, receiving data (blocking and non-blocking).

Simulation through chat program

The complete protocol is simulated through chat program. The GUI consists of text areas and buttons. When connect on client side is clicked, connection to the server is made and public key for encryption is exchanged through Diffie Hellman protocol. For this protocol messages are sent in plain text.

Subsequently after the connection establishment and key-exchange, string is typed (to be sent) is encrypted by AES algorithm. The MD5 authentication hash function is applied to ciphertext and digest is appended to ciphertext and sent to other side. On the other side, the digest and ciphertext are separated, digest is calculated again, and if it matches, the ciphertext is passed on to the decryption algorithm. After decryption, the original text (decrypted text) is passed on to the application.

Hence, each packet is encrypted and authenticated. Also, each layer works independent of the upper layer.

Structure of Code

'client_app.py' : application chat program with GUI for client
'server_app.py': application chat program with GUI for server
'aes_client.py': network layer(ESP), for encrypting and decrypting the IP packet on client side
'aes_server.py': network layer(ESP), for encrypting and decrypting the IP packet on server side
'md5.py' and 'define.h': network layer(AH), for authentication headers(cryptographic hash)
'server.py' and 'client.py': defines basic functions in link layer

Instructions to run

cd to the appropriate folder.

```
$sudo python server.py                    //server GUI opens up
```

```
$python client.py                    //client GUI opens up
```

Click ‘Setup Server’ on server side. Connect client by clicking ‘connect’ on client side.

Then continue the chat in usual way.

Conclusion

The IPsec in network layer provides encryption and authentication. This project simulates the same through chat program between server and client.

Notes

1. In the simulation, server connects to ‘localhost’ at Port 80
2. The server connects to just on client for the purpose of simulation.

Citations and References

RFC1321 and RFC1322: detail about the IPsec protocol