SECURITY THROUGH SSL IN CLOUD COMPUTING: A REVIEW

Angel Makhaik, Pankaj Vaidya

Abstract– Cloud Computing provides hardware and software resources as a service to client. In it, to gain access to these services one need to have Internet access. As we all know Internet is everywhere and in every home, so the number of users has increased. Due to the increase in the number of internet users, the number of online transactions are also increasing. Like user is asked to give his sensitive data, say credit card details, for some transactions. Each information which is transmitted over the web has to be traversed to many computer servers before it reach its destination. It is up to the user which webpage is secure and which is not. For such scenario SSL is a solution. This paper will give u the insight of how the data is securely sent over the internet through SSL.

Index Term: - Cloud Computing, Diffie-Hellman Encryption, Key Exchange Protocol, RSA, SSL, Services.

1. INTRODUCTION

Cloud computing is an expression used to describe a variety of computing concepts that involve a large number of computers connected through a real-time communication network such as the Internet.[1] It allows to access applications that actually resides at a location other than your computer or other internet connected device. Cloud computing get its name as a metaphor for the internet. Typically, the internet is represented in network diagram as a cloud. Cloud computing allows consumers and businesses to use applications without installation and access their personal files at any computer with internet access.

1.1 Services:

Services of cloud computing is divided in following three categories:-
1. Software as a Service. (SaaS)
2. Platform as a Service. (PaaS)
3. Infrastructure as a Service. (IaaS)

Software as a Service

Software as a Service (SaaS) is a model in which an application is hosted as a service to customers who access it via Internet. When the software is hosted off-site, the customer doesn't have to maintain it. On the other hand, it is out of customer's hands when the hosting service decides to change it. Cost can be an important factor in this computing environment for accessing any software, rather than pay for it once and be done with it, the more you use it, the more you will have to pay ("pay-for-use").

Platform as a Service

Platform as a Service (PaaS) is another application delivery model that provides all the resources required to build application and services completely from the Internet, without downloading or installing software. PaaS services include application design, development, testing, deployment and hosting.

Infrastructure as a Service:

Infrastructure as a service is the next form of services available in cloud computing. Where SaaS, PaaS are providing applications to customers, IaaS doesn't. It simply offers the hardware so that your organization can put whatever they want onto it. Rather than purchase servers, software, racks, and having to pay for the datacenter space for them, the service provider rents those resources.

2. SECUIRTY CONCEPTS

Security over the Internet has become a major concern for the modern communication sector. Whether it is financial, personal or business everyone wants to know whom they are communicating with, ensuring that their data can be sent securely, and whether it has reached the destination correctly. Communicating data over a network always implies a possible loss of confidentiality, message integrity or endpoint authentication. These are the three major aspects one has to consider when speaking of data security [2]:

• Confidentiality: the user wants its data to be kept secret from the listeners on the network.

• Message Integrity: The user wants integrity of its data, what is received is what is sent.
• Endpoint Authentication: The user wants to be sure that he is communicating to the right partner.

SSL assures secure data transmission through the use of several security concepts. These concepts consist of one or more cryptographic algorithms, used to provide security.

2.1 Encryption

Encryption technique is used to protect the confidential data from unauthorized user. Encryption is the most effective way to achieve data security. The process of Encryption hides the contents of a message in a way that the original information is recovered only through a decryption process [3]. The purpose of Encryption is to prevent unauthorized parties from viewing or modifying the data [4].

Key-based algorithms use an Encryption key to encrypt the message. There are two general categories for key-based Encryption: Symmetric Encryption which uses a single key to encrypt and decrypt the message and Asymmetric Encryption which uses two different keys – a public key to encrypt the message, and a private key to decrypt it. Currently, there are several types of key based Encryption algorithms such as: DES, RSA, PGP, Elliptic curve, and others but all of these algorithms depend on high mathematical manipulations [5, 6].

The scheme of using the same key for encryption and decryption is often referred to as Secret Key Cryptography (SKC). Some of the most popular ciphers on the market are the Data Encryption Standard (DES) [7], Triple-DES (DES repeated three times) [8], RC2 [9] and RC4 [10]

2.1.1 Message Digests:

A message digest is a function that takes a message of arbitrary length as an input and generates a string of fix length as an output. This string is simply a characteristic representation of the initial message's content. Message digests have two important properties:

1. Irreversibility: Computing a message given its digest should be extremely difficult.

2. Collision-Resistance**:** It should be nearly impossible to produce two messages with the same digest.

The main purpose of message digests is the computation of Digital Signatures and Message Authentication Codes (MACs). Message digests generally allow to prove possession of a secret without actually revealing it. Currently, Message Digest 5 (MD5) [11] and Secure Hash Algorithm (SHA-1) [12] are the most widely used message digest algorithms.

2.1.2 Message Authentication Codes:

A Message Authentication Code is sort of a digest algorithm with the difference that beside the message, a key is incorporated to the computation as well. Thus a MAC always depends on both the message being treated as well as the key being used for encryption. MACs are usually constructed using digest algorithms. SSLv3 uses a variant of HMAC, while TLS uses HMAC itself. HMAC is a description of how to create MACs with provable security properties based on digest algorithms.

3. SSL OVERVIEW:-

SSL is a protocol developed by Netscape for transmitting private documents via the Internet. Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols is to provide a mechanism for secure communications between two parties over a network which neither party has end-to-end control over and thus has the potential for third parties to intercept communication.
SSL is a sophistication encryption scheme that does not require the client and the server to arrange for a secret key to be exchanged between the client and server BEFORE the transaction is started. SSL uses public/private keys to provide a flexible encryption scheme that can be setup at the time of the secure transaction.

The preconfigured secret keys are not suitable for Web based secure services that involve millions of users who have no prior secret key arrangement with the secure server.
SSL solves this problem by using asymmetric keys. These keys are defined in pairs of public and private keys. As the name suggests the public key is freely available to anybody. The private key is known only to the server. The keys have two important properties:
(1) Data encrypted by the client using the pubic key can be decrypted only by the server's private key. Due to this property of the keys, the client is able to send secure data that can be understood only by the server.
(2) Data encrypted to by the server's private key can only be decrypted using the public key. This property is useful in a client level authentication of the server. If the server sends a known message (say the name of the server), the client can be sure that it is talking to the authentic server and not an imposter if it is successfully able to decrypt the message using the public key.

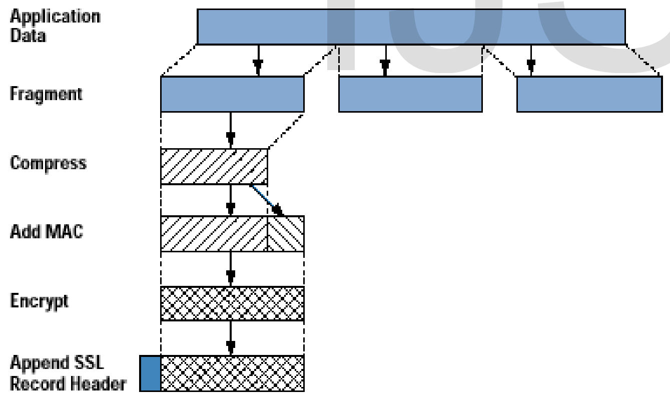
The public/private key based encryption is used only for handshaking and secret key exchange. Once the keys have been exchanged the symmetric secret keys are used. This is done for two reasons:
(1) Public/private key based encryption techniques are computationally very expensive thus their use should be minimized.
(2) The secret key mechanism is needed for server to client communication.
The SSL protocol includes two sub-protocols:

SSL record protocol: - format used to transmit data

SSL handshake protocol: - using the SSL record protocol to exchange a series of messages

3.1 SSL record protocol

The SSL Record Protocol provides two services for SSL connections: confidentiality, by encrypting application data; and message integrity, by using a *message authentication code* (MAC). The Record Protocol is a base protocol that can be utilized by some of the upper-layer protocols of SSL. One of these is the handshake protocol which, as described later, is used to exchange the encryption and authentication keys. Following Figure indicates the overall operation of SSL Record Protocol.



AS you can see in the diagram, firstly record protocol takes application message to be transmitted. Then second step is the fragmentation, each upper-layer message is fragmented into blocks of 2 14 bytes (16,384 bytes) or less. Next, compression is optionally applied. In SLLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm is null. However, specific implementations may include a compression algorithm.

The next step is to compute the Message authentication code (MAC) over the compressed data. To compute the MAC shared key is used. Next, the compressed message plus the MAC are encrypted using symmetric encryption. A variety of encryption algorithms may be used, including the Data Encryption Standard (DES) and triple DES. The final step of SSL Record Protocol processing is to prepend a header, consisting of the following fields:

- Content Type (8 bits): The higher-layer protocol used to process the enclosed fragment.
- Major Version (8 bits): Indicates major version of SSL in use. For SSLv3, the value is 3.
- Minor Version (8 bits): Indicates minor version in use. For SSLv3, the value is 0.
- Compressed Length (16 bits): The length in bytes of the plain-text fragment (or compressed fragment if compression is used).

Change CipherSpec Protocol

The Change CipherSpec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol, and it is the simplest. This protocol consists of a single message, which consists of a single byte with the value 1. The sole purpose of this message is to cause the pending state to be copied into the current state, which updates the CipherSuite to be used on this connection. This signal is used as a coordination signal. The client must send it to the server and the server must send it to the client. After each side has received it, all of the following messages are sent using the agreed-upon ciphers and keys.

Alert Protocol

The Alert Protocol is used to convey SSL-related alerts to the peer entity. Each message in this protocol consists of two bytes. The first byte takes the value "warning" (1) or "fatal"(2) to convey the severity of the message. If the level is fatal, SSL immediately terminates the connection. Other connections on the same session may continue, but no new connections on this session may be established.
The second byte contains a code that indicates the specific alert. An example of a fatal message is illegal parameter (a field in a handshake message was out of range or inconsistent with other fields). An example of a warning message is close notify (notifies the recipient that the sender will not send any more messages on this connection; each party is required to send a close notify alert before closing the right side of a connection).

3.2 SSL Handshake Protocol

The most complex part of SSL is the Handshake Protocol. This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record. The Handshake Protocol is used before any application data is transmitted. The Handshake Protocol consists of a series of messages exchanged by the client and the server.

The Handshake Protocol is shown in figure 11.6. This consists of four phases:

1. Establish security capabilities including protocol version, session ID, cipher suite, compression method and initial random numbers. This phase consists of the client hello and server hello messages which contain the following (this is for the client however it's a little different for the server):
   - Version: The highest SSL version understood by client
   - Random: 32-bit timestamp and 28 byte nonce.
   - Session ID: A variable length session identifier.
   - CipherSuite: List of crypto algorithms supported by client in decreasing order of preference. Both key exchange and CipherSpec (this includes fields such as Cipher Algorithm, Mac Algorithm, Cipher.

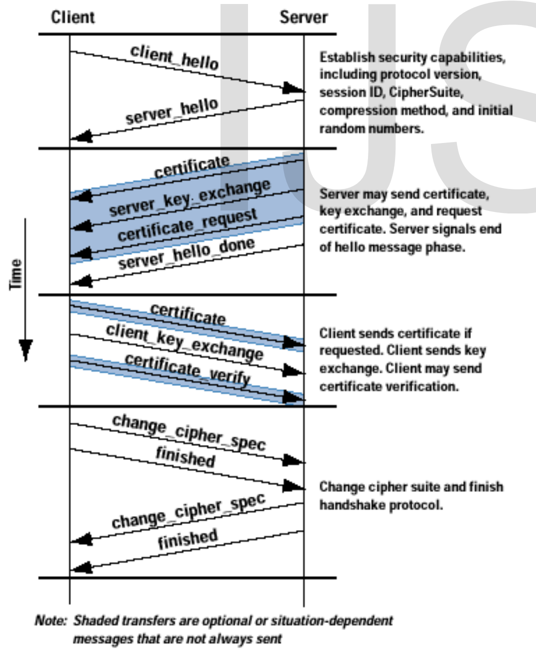

Client
Server
client_hello
server_hello
Establish security capabilities, including protocol version, session ID, CipherSuite, compression method, and initial random numbers.
certificate
server_key_exchange
certificate_request
server_hello_done
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Time
certificate
client_key_exchange
certificate_verify
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
change_cipher_spec
finished
change_cipher_spec
finished
Change cipher suite and finish handshake protocol.
Note: Shaded transfers are optional or situation-dependent messages that are not always sent

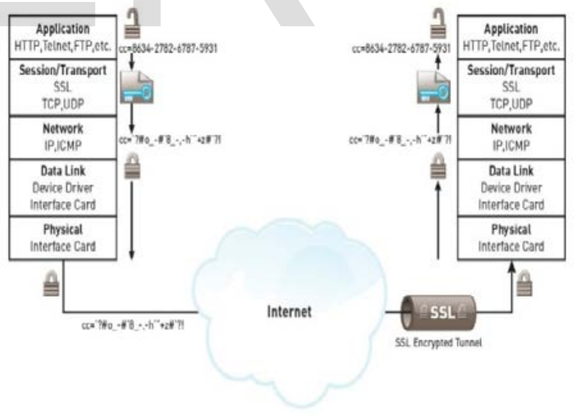
Figure 2: Handshake Protocol Action

   - Type, Hash Size, Key Material and IV Size) are defined.
   - Compression Method: List of methods supported by client.

2. Server may send certificate, key exchange, and request certificate it also signals end of hello message phase. The certificate sent is one of a chain of X.509 certificates discussed earlier in the course. The server key exchange is sent only if required. A certificate may be requested from the client if needs be by certificate request.
3. Upon receipt of the server done message, the client should verify that the server provided a valid certificate, if required, and check that the server hello parameters are acceptable. If all is satisfactory, the client sends one or more messages back to the server. The client sends certificate if requested (if none available then it sends a no certificate alert instead). Next the client sends client key exchange message. Finally, the client may send certificate verification.
4. Change cipher suite and finish handshake protocol. The secure connection is now setup and the client and server may begin to exchange application layer data.

4. SSL AND THE OSI MODEL

The SSL protocol is a security protocol that sits on top of TCP at the transport layer. In the OSI model, application layer protocols such as HTTP or IMAP, handle user application tasks such as displaying web pages or running email servers. Session layer protocols establish and maintain communications channels. Transport layer protocols such as TCP and UDP, handle the flow of data between two hosts. Network layer protocols such as IP and ICMP provide hop-by-hop handling of data packets across the network.



5. THE KEY EXCHANGE PROTOCOL

5.1 The SSL Handshake Protocol

The SSL handshake-protocol message flow involves client and server negotiating a common cipher suite acceptable to both parties, exchanging random once, and the client sending an encrypted master secret. Then each verifies that their protocol runs match by authenticating all messages with the master secret, and assuming that the check succeeds, both generate session keys from the master secret and proceed to send application data.

The handshake protocol is responsible for selecting a CipherSpec and generating a MasterSecret, which together comprise the primary cryptographic parameters associated with a secure session. The handshake protocol can also optionally authenticate parties who have certificates signed by a trusted certificate authority [13].

5.2 Authentication and key exchange

SSL supports three authentication modes: authentication of both parties, server authentication with an unauthenticated client, and total anonymity. Whenever the server is authenticated, the channel should be secure against man-in-the-middle attacks, but completely anonymous sessions are inherently vulnerable to such attacks. Anonymous servers cannot authenticate clients, since the client signature in the certificate verify message may require a server certificate to bind the signature to a particular server. If the server is authenticated, its certificate message must provide a valid certificate chain leading to an acceptable certificate authority. Similarly, authenticated clients must supply an acceptable certificate to the server. Each party is responsible for verifying that the other's certificate is valid and has not expired or been revoked. The general goal of the key exchange process is to create a pre MasterSecret known to the communicating parties and not to attackers. The pre MasterSecret will be used to generate the MasterSecret. The MasterSecret is required to generate the finished messages, encryption keys, and MAC secrets. By sending a correct finished message, parties thus prove that they know the correct pre MasterSecret [14].

5.3 Anonymous key exchange

Completely anonymous sessions can be established using RSA, Diffie-Hellman, or Fortezza for key exchange. With anonymous RSA, the client encrypts a pre MasterSecret with the server's uncertified public key extracted from the server key exchange message. The result is sent in a client key exchange message. Since eavesdroppers do not know the server's private key, it will be infeasible for them to decode the pre MasterSecret. With Diffie-Hellman or Fortezza, the server's public parameters are contained in the server key exchange message and the clients are sent in the client key exchange message. Eavesdroppers who do not know the private values should not be able to find the Diffie-Hellman result (*i.e.* the pre MasterSecret) or the Fortezza token encryption key (TEK). Warning: Completely anonymous connections *only* provide protection against passive eavesdropping. Unless an independent tamper-proof channel is used to verify that the finished messages were not replaced by an attacker, server authentication is required in environments where active man-in-the-middle attacks are a concern [14].

5.4 RSA key exchange and authentication

With RSA, key exchange and server authentication are combined. The public key may be either contained in the server's certificate or may be a temporary RSA key sent in a server key exchange message. When temporary RSA keys are used, they are signed by the server's RSA or DSS certificate. The signature includes the current ClientHello.random, so old signatures and temporary keys cannot be replayed. Servers may use a single temporary RSA key for multiple negotiation sessions.

When RSA is used for key exchange, clients are authenticated using the certificate verify message. The client signs a value derived from the MasterSecret and all preceding handshake messages. These handshake messages include the server certificate, which binds the signature to the server, and ServerHello.random, which binds the signature to the current handshake process [15].

5.5 Diffie-Hellman key exchange with authentication

When Diffie-Hellman key exchange is used, the server can either supply a certificate containing fixed Diffie-Hellman parameters or can use the client key exchange message to send a set of temporary Diffie-Hellman parameters signed with a DSS or RSA certificate. Temporary parameters are hashed with the hello.random values before signing to ensure that attackers do not replay old parameters. In either case, the client can verify the certificate or signature to ensure that the parameters belong to the server. If the client has a certificate containing fixed Diffie-Hellman parameters, its certificate contains the information required to complete the key exchange. Note that in this case the client and server will generate the same Diffie-Hellman result (*i.e.*, pre MasterSecret) every time they communicate. To prevent the pre MasterSecret from staying in memory any longer than necessary, it should be converted into the MasterSecret as soon as possible. Client Diffie-Hellman parameters must be compatible with those supplied by the server for the key exchange to work. If the client has a standard DSS or RSA certificate or is unauthenticated, it sends a set of temporary parameters to the server in the client key exchange message, then optionally uses a certificate verify message to authenticate itself [16].

5.6 Fortezza

Fortezza's design is classified, but at the protocol level it is similar to Diffie-Hellman with fixed public values contained in certificates. The result of the key exchange process is the token encryption key (TEK), which is used to wrap data encryption keys, client write key, server write key, and master secret encryption key. The data encryption keys are not derived from the pre MasterSecret because unwrapped keys are not accessible outside the token. The encrypted pre MasterSecret is sent to the server in a client key exchange message. The asymmetric algorithms are used in the handshake protocol to authenticate parties and to generate shared keys and secrets.

6. CONCLUSION

SSL has become the universal standard for authenticating web sites to web browsers, and for encrypting communications between web browsers and web servers. The data which is sent over the communication network is totally secure and that total security is provided by SSL Protocol. It also discusses the role of key exchange protocol through handshake protocol authenticate parties who have certificates signed by a trusted certificate authority. Thus, the paper describes the usage of SSL in securing the data transmitted through web pages.

7. REFERENCE:-

1. Mariana Carroll, Paula Kotzé, Alta van der Merwe (2012). "Securing Virtual and Cloud Environments". In I. Ivanov et al. *Cloud Computing and Services Science, Service Science: Research and Innovations in the Service Economy*. Springer Science+Business Media. doi:10.1007/978-1-4614-2326-3
2. A. O. Freier P. Karlton and P. C. Kocher. The SSL Protocol, Version 3.0. Netscape Communications, 1996, http://wp.netscape.com/eng/ssl3/draft302.txt (2003).
3. Wikipedia,"*Encryption*",http://en.wikipedia.org/wiki/Encryption, modified on 13 December 2006.
4. Freeman J., Neely R., and Megalo L. "*Developing Secure Systems: Issues and Solutions*". IEEE Journal of Computer and Communication, Vol. 89, PP. 36-45. 1998
5. Beth T. and Gollmann D. "*Algorithm Engineering for Public Key Algorithms*". IEEE Journal on Selected Areas in Communications; Vol. 7, No 4, PP. 458-466. 1989
6. IBM. "*The Data Encryption Standard (DES) and its strength against attacks*". IBM Journal of Research and Development, Vol. 38, PP. 243-250. 1994
7. National Institute of Standards and Technologies. Data Encryption Standard. U.S. Dpt. of Commerce, December 1993.
8. ANSI. American National Standard for Financial Institution Key Management (wholesale). ANSI, 1985.
9. R. Rivest. A Description of the RC2(r) Encryption Algorithm, RFC 2268. Network Working Group, 1998, ftp://ftp.rfc-editor.org/in-notes/rfc2268.txt (2003).
10. B. Schneier. Applied Cryptography, 2ed. John Wiley and Sons, 1996
11. R. Rivest. The MD-5 Message-Digest Algorithm, RFC 1321. John Wiley and Sons, 1996, ftp://ftp.rfc-editor.org/in-notes/rfc1321.txt (2003).
12. National Institute of Standards and Technologies. National Institute of Standards and Technologies, and Secure Hash Standard. U.S. Dpt. of Commerce, 1994.
13. [BCK96] M. Bellare, R. Canetti, and H. Krawczyk, \Keying Hash Functions for Message Authentication," *Advances in Cryptology|CRYPTO '96 Proceedings*, Springer-Verlag, 1996, pp. 1-15.
14. [FKK96] A. Freier, P. Karlton, and P. Kocher, The SSL Protocol Version 3.0", ftp://ftp.netscape.com/pub/review/ ssl-spec.tar.Z, March 4 1996, Internet Draft, work in progress.
15. MERKLE, R., AND HELLMAN, M. 1981. On the security of multiple encryptlon*Commun. ACM* 24, 7 (July), 465-467.
16. [RSA93] RSA Data Security, Inc., \Public-Key Cryptography Standards (PKCS)," Nov 93.