Abstract

Cloud computing is one of the emerging area in information technology nowadays in which all processing and transactions are performed in the internet “cloud”. Although it provides many advantages to many business organizations like Performance, high availability, least cost, flexibility, scalability and many more, it brings new security concerns for reliability and safety of user data. While moving to the cloud means giving up control of private and confidential data, bringing data segregation risks. Traditional on-site storage lets businesses control where data is located and exactly who can access it, but putting information in the cloud means putting location and access in the cloud provider’s hands. Hence numerous security challenges that need to be considered and addressed prior to committing to a cloud computing strategy. This paper is describing solutions to these problems and is varied and must be explored individually, but one technology shows up often: TLS or Transport Layer Security, often known by the name of the predecessor technology, SSL or Secure Sockets Layer with the use of SSL/TLS encryption, data can securely move between servers or between servers and browsers. This prevents unauthorized interceptors from reading that data.


I. INTRODUCTION

Cloud computing is one of the most hot and important technology in information technology. We can say, it is the addition of previous two technology namely grid computing and virtualization.

Cloud computing uses Scalable Data centers, which is collection of data and services provided by cloud provider that provides ubiquitous access to end user. Many large organization and companies uses cloud services to deploy their application which are web enabled. [7, 8]

1. Services:
Services in cloud computing is the concept of being able to use reusable and fine grained component across a vendor’s network. This is widely known as “as a service”. Offering with as services as a suffix include traits like Following:

1. Low barriers to entry, making them available to small businesses.
2. Large scalability.
3. Multi-tenancy, which allow resources to be shared by many users.
4. Device independence, which allows user to access the Systems on different hardware.

Basically cloud computing provides three types of services and these are classified in to the followings: Software as a service (SaaS), platform as a service (PaaS) and Infrastructure as a service (IaaS). These services dynamically provide resources like operating system, software, hardware, storage, network resources etc to the users on demand. Service layers, sometime referred as SPI model. [7]

I. SaaS (Software as a Service): This layer resides at the top of cloud stack and user can access application or software service which is hosted in cloud environment. For example ‘facebook.com’ which is simply accessed by installing web browser at client machine. Other examples are Google Apps and Salesforce.com which also provides Software as a Service.

II. PaaS (Platform as a Service): A cloud platform provides an environment for organization to develop and deploy their application without purchasing costly licensed software and user can simply access cloud platform to create various applications. Also developers can take the advantages of security features supported by integrated cloud platform.
III. IaaS (Infrastructure as a Service): it provide virtual pool of resources like storage, operating system, memory, processors etc which is dynamic in nature that is user can access resources as when needed and which can be increased and decreased. For example Amazon provides IaaS.

II. ESSENTIAL CHARACTERISTICS OF CLOUD COMPUTING

Following are the essential characteristics of cloud Computing: [8]

a. On-demand self-service: A cloud computing end user will be able to access software, hardware and network infrastructure as when needed automatically and no need to interact with cloud service provide or any others.
b. Ubiquitous network access: A user can access wide range of devices (laptop, mobile, smart phones, PDAs etc) which are having internet facility with the use of standard mechanism.
c. Resource pooling: Cloud end user will be able to increase or decrease services such as storage, processing, memory, network bandwidth, and virtual machines capacity as desired which is independent of location.
d. Rapid elasticity: To achieve scalability, without assigning specific resources to any individual user, infrastructure can be managed and controlled which is balanced across a whole and provided on demand.
e. Pay per use: Cloud service consumers/ users will be charged rent based on their resource utilization which can be storage, processing, bandwidth, and active user accounts which is transparent in nature.
f. Multi Tenacity: It refers to the need for policy-driven enforcement, segmentation, isolation, governance, service levels, and chargeback/billing models for different consumer constituencies. Consumers might utilize a public cloud provider’s service offerings or actually be from the same organization, such as different business units rather than distinct organizational entities, but would still share infrastructure.

II.I CLOUD COMPUTING DEPLOYMENT MODELS

Cloud deployment model can be basically categorized in three ways by which cloud services can also be deployed and described. [8]

Public Cloud: In this type of cloud, end user can access their own resources via cloud services which are provided and managed by cloud service provider (third party cloud vendor) that can be access over the internet on demand. Users have no knowledge about how the cloud services are managed and controlled, what infrastructures are provided or available.

These kinds of provider support multiple end users by using multiple data centers and provide their security features to users. These kinds of cloud users are not trusted.

Examples of Public Cloud:
• Google App Engine
• Microsoft Windows Azure
• IBM Smart Cloud
• Amazon EC2

Private Cloud: In this model, cloud infrastructure can be exclusively operated for any business. It can be managed by third party cloud vendor and may exist off premises. Based on that two private cloud scenarios can be exist, as follows:

On-site Private Cloud
It can be implemented at a customer’s premises.

Outsourced Private Cloud
Where the server side is outsourced to a hosting company.

Examples of Private Cloud:
• Eucalyptus
• Ubuntu Enterprise Cloud - UEC (powered by Eucalyptus)
• Amazon VPC (Virtual Private Cloud)
• VMware Cloud Infrastructure Suite
• Microsoft ECI data center.
Hybrid Cloud: The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load-balancing between clouds).

Examples of Hybrid Cloud:
- Windows Azure (capable of Hybrid Cloud)
- VMware vCloud (Hybrid Cloud Services)

Community Cloud: The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). Government departments, universities, central banks etc. often find this type of cloud useful. Community cloud also has two possible scenarios:

On-site Community Cloud Scenario
- Implemented on the premises of the customers composing a community cloud.

Outsourced Community Cloud
- Server side is outsourced to a hosting company.

Examples of Community Cloud:
- Google Apps for Government
- Microsoft Government Community Cloud

III. CLOUD SECURITY CHALLENGES
Cloud computing opens up a new world of opportunities for businesses, but mixed in with these opportunities are numerous security challenges that need to be considered and addressed prior to committing to a cloud computing strategy. [3, 4]

Cloud computing security challenges fall into three broad categories:

1) Data Protection: Securing your data both at rest and in transit
2) User Authentication: Limiting access to data and monitoring who accesses the data. [1]
3) Disaster and Data Breach Contingency Planning

III. CLOUD DATA SECURITY THROUGH SSL/TLS PROTOCOL
A popular implementation of public-key encryption is the Secure Sockets Layer (SSL). It is originally developed by Netscape. SSL is an Internet security protocol used by Internet browsers and Web servers to transmit sensitive information. SSL has become part of an overall security protocol known as Transport Layer Security (TLS).

In web browser, we can tell when we are using a secure protocol, such as TLS, in a couple of different ways. We can notice that the "http" in the address line is replaced with "https" as shown in Figure 2.
We should see a small padlock in the status bar at the bottom of the browser window as shown in figure 3. When we are accessing sensitive information over cloud environment (basically software as a service) using internet, such as an online bank account or a payment transfer service like PayPal or Google Checkout, chances are you’ll see this type of format change and know your information will most likely pass along securely.[3,4]

Both TLS and SSL make significant use of certificate authorities (CA). Once browser requests a secure page and adds the "s" onto "http," the browser sends out the public key and the certificate, it will check three things:

1) The certificate comes from a trusted party
2) The certificate is currently valid
3) Certificate has a relationship with the site from which it’s coming.

The browser then uses the public key to encrypt a randomly selected symmetric key. Public-key encryption takes a lot of computing, so most systems use a combination of public-key and symmetric key encryption.

When two computers initiate a secure session, one computer creates a symmetric key and sends it to the other computer using public-key encryption. The two computers can then communicate using symmetric-key encryption. Once the session is finished, each computer discards the symmetric key used for that session. Any additional sessions require that a new symmetric key be created, and the process is repeated.

IV. TLS/SSL PROTOCOL OVERVIEW

Why has the TLS/SSL protocol been so widely adopted? Like me, many of you have probably studied the TCP/IP protocols family and learnt that, even if an official standard protocol exists (the ISO/OSI model), TCP/IP became the “de facto” standard for all network communications due to its simplicity and its ability to work well in every situation.[2]

The success of the TLS/SSL protocol relies on the fact that it was designed to perfectly match the TCP/IP protocol architecture. It maintains the TCP/IP’s layer-based design principles, thus inheriting all their advantages. It maybe for this reason that protocols like S-HTTP have not had the same success. Figure 4 shows how TLS/SSL protocol interacts with the TCP/IP protocol. [2, 5]
Transport Layer Security, a protocol that guarantees privacy and data integrity between client/server applications communicating over the Internet and application hosted in cloud environment.

TLS/SSL adds a new layer to the TCP/IP layers stack. As with all the other layers in the stack, the TLS/SSL protocols are independent of the protocols above and below, but the layer “speaks the same language” as the same layer on the other side of the communications channel. This design not only ensures full compatibility with all the network technologies based on TCP/IP, but it also enables them to easily “switch” to their secure versions without having to reinvent them from the ground up. So, HTTP became HTTPS without having to modify its specifications, FTP became FTPS, and so on.

The TLS protocol is made up of two layers:

The TLS Record Protocol -- layered on top of a reliable transport protocol, such as TCP, it ensures that the connection is private by using symmetric data encryption and it ensures that the connection is reliable. The TLS Record Protocol also is used for encapsulation of higher-level protocols, such as the TLS Handshake Protocol.

The TLS Handshake Protocol -- allows authentication between the server and client and the negotiation of an encryption algorithm and cryptographic keys before the application protocol transmits or receives any data.

TLS is application protocol-independent. Higher-level protocols can layer on top of the TLS protocol transparently. [4] To keep the data secure during the communication, TLS/SSL uses cryptographic techniques. Among the four goals of cryptography (confidentiality, integrity, authentication, and non-repudiation) TLS/SSL is able to guarantee confidentiality, integrity, and authentication. This can be done principally in two steps:

Step 1: Authentication of the entities involved in the data exchange and negotiation of the cryptographic parameters to be used during the communication. This step uses asymmetric cryptography and X509 digital certificates.

Step 2: Symmetric encryption of exchanged data, and message authentication code (MAC) calculation and verification of each packet transmitted. The first assures confidentiality, the second integrity.

### IV.I THE HANDSHAKE MECHANISM

The first step occurs by using a series of messages that the two communicating entities (client and server) exchange to start the secure communication. The protocol that specifies the kind of messages to be exchanged and their relative order is the Handshake Protocol.

During the handshake, the client and the server agree on the TLS/SSL protocol version to adopt, they decide on the cryptographic algorithms and the relative parameters that they will use, they (optionally) authenticate each other, and finally they exchange shared secrets to use during the second step.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Handshake messages exchanged during a secure communication with TLS/SSL protocol. [6]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client Message: Client Hello =&gt;</strong></td>
<td><strong>Server Message: &lt;= Server Hello</strong></td>
</tr>
<tr>
<td>Exchange data:</td>
<td>Exchanged data:</td>
</tr>
<tr>
<td>Max SSL Protocol version allowed</td>
<td>SSL Protocol Selected</td>
</tr>
<tr>
<td>SessionID</td>
<td>Session ID</td>
</tr>
<tr>
<td>Set of Cipher Suites</td>
<td>Selected Cipher Suite</td>
</tr>
<tr>
<td>Set of Compression Methods</td>
<td>Selected Compression Method</td>
</tr>
<tr>
<td>Random number</td>
<td>Random Number</td>
</tr>
</tbody>
</table>

The client sends the maximum number of the SSL protocol version it understands, a session id to be used to resume the communication if needed, a set of cipher suites and compression methods that it is able to use, and a random generated number to be used during the key exchange.

The server sends the SSL protocol version selected (not exceeding the maximum supported by the client), a session id, the cipher suite and compression method selected from those suggested by the client, and a random number to be used during the key exchange.

| **Server Message: <= Server Certificate** |
### Exchanged data: Server’s X509 Certificate(s)
The server sends its X509 certificate (or X509 certificates chain) to use during the key exchange to encrypt key materials and to authenticate itself.

**Server Message: <= Server Key Exchange**

<table>
<thead>
<tr>
<th>Exchanged data:</th>
<th>Public key to use for the key exchange algorithm (optional)</th>
</tr>
</thead>
</table>

If the server’s certificate is not able to perform encryption, the server sends a public key to be used to encrypt key materials.

**Server Message: <= Certificate Request**

<table>
<thead>
<tr>
<th>Exchanged data:</th>
<th>None</th>
</tr>
</thead>
</table>

(Optional) If client authentication is required, the server sends the Certificate Request message to the client.

**Server Message: <= Server Hello Done**

<table>
<thead>
<tr>
<th>Exchanged data:</th>
<th>None</th>
</tr>
</thead>
</table>

End of the Server Hello.

**Client Message: Certificate =>**

<table>
<thead>
<tr>
<th>Exchanged data:</th>
<th>Client’s X509 Certificate (optional)</th>
</tr>
</thead>
</table>

If client authentication is required, the client sends to the server its X509 certificate (or X509 certificates chain).

**Client Message: Client Key Exchange =>**

<table>
<thead>
<tr>
<th>Exchanged data:</th>
<th>Key material encrypted with the server’s certificate</th>
</tr>
</thead>
</table>

The client sends all the information needed to agree with the server the symmetric key to use during the communication. The type of information depends on the key exchange algorithm selected.

**Client Message: Certificate Verify =>**

<table>
<thead>
<tr>
<th>Exchanged data:</th>
<th>Signed handshake messages sent or received until now (optional)</th>
</tr>
</thead>
</table>

If client authentication is required, the client sends all the handshake messages exchanged until now signed with its private key. The server verifies the client’s identity using the public key received.

**Client Message: Change Cipher Spec + Finished =>**

<table>
<thead>
<tr>
<th>Exchanged data:</th>
<th>None</th>
</tr>
</thead>
</table>

End of the negotiation.

**Server Message: <= Change Cipher Spec + Finished**

<table>
<thead>
<tr>
<th>Exchanged data:</th>
<th>None</th>
</tr>
</thead>
</table>

End of the negotiation.

### IV.II DATA EXCHANGE
In the second step, the data exchange takes place. In the same way as all the other TCP/IP layers, TLS/SSL maintains the fragment-based nature of the communication. Information is divided into small data units, and each unit is encapsulated with a header that contains the information needed by the other side of the communication channel to reconstruct the original message.

A MAC is appended to each data fragment, and the overall content is encrypted using the symmetric algorithm and the key materials negotiated in first step. Figure 4 illustrates the composition of a TLS/SSL data unit.

Figure 4: TLS/SSL Data Unit

V. REQUIREMENT SPECIFICATIONS

Table 2: Summary of software and Hardware for implementation of SSL/TLS using .NET framework.

<table>
<thead>
<tr>
<th>Software Specification</th>
<th>Hardware Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Processor: Pentium-IV</td>
</tr>
<tr>
<td>Windows XP/7</td>
<td>Processor Speed: 1.1GHz</td>
</tr>
<tr>
<td>Tools and Technology</td>
<td>RAM: 1GB</td>
</tr>
<tr>
<td>ASP.NET, C#.NET, WCF</td>
<td>Hard Disk: 40 GB</td>
</tr>
<tr>
<td>IDE(Integrated Development Environment</td>
<td>Visual Studio 2010/2012</td>
</tr>
<tr>
<td>.NET Framework</td>
<td>4.0/4.5</td>
</tr>
</tbody>
</table>

VI. FUTURE SCOPE

The future work is to implement a secure communication using the TLS/SSL protocol using Microsoft .NET Framework comes with the SslStream class under the System.Net.Security namespace.

VII. CONCLUSION

In this paper, we describe the security issues related to the cloud computing; help you to better understand the TLS/SSL protocol and the principles behind it thus simplifying your job when dealing with it. As we have seen, to implement at low level a secure communication channel by using the TLS/SSL protocol isn’t a trivial task. It requires at least a basic knowledge about symmetric and asymmetric cryptography, hash calculation and public key infrastructure (PKI). Even if in the last few years a large demand for cloud data security has taken place, the complexity behind it sometimes is the reason why it is put aside.

REFERENCES