CHAPTER 7

SECURE CLOUD COMPUTING SERVICES WITH VIRTUALIZATION AND SERVICE POLICY MONITORING TECHNIQUES USING CLOUD SECURITY MODEL

7.1 INTRODUCTION

This chapter emphasizes on achieving the cloud security with Virtualization and Service Policy Monitoring Techniques using Cloud Security Model (CSM). Policy Monitoring is a continuous process based on regulatory compliance, information risk, service requirements and cloud business requirement. The proposed architecture “Cloud Security Model” has two major components. Policy Monitor module (PM) and Security Auditor module (SA). PM module embeds the access control policy and enhanced message protection policies namely Homomorphic Merkle hash algorithm (HMDigestGenerator). The Security Auditor module explores the data communication network traffic using Nmap (Network Mapper). Using vulnerability analysis, risks are identified periodically by Nmap, and based on the assessment of risks; it assures security either by isolating the intruded virtual machine or by reconstructing the virtual machine images from the snapshots using hypervisor.
7.2 SECURE CLOUD COMPUTING SERVICES USING VIRTUALIZATION AND SERVICE POLICY MONITORING TECHNIQUES

Cloud Services are provided to the users through the Cloud Service Provider. The Proposed Architecture for CSM is shown in Figure.7.1

![Proposed architecture for CSM](image)

**Figure 7.1 Proposed architecture for CSM**

The Policy Monitoring technique is depicted in Chapter 5. (Figure 5.2). Here the whole concept is presented for explanation. In this chapter the entire Figure 7.1 is decomposed in to small modules for wanted clarity. This kind of decomposition facilitates the reader to understand the proposed concept clearly. Figures 7.1-7.3 depicts the proposed CSM architecture with its sub modules and working details are explained. Accesses to virtual machines are strictly through CSM modules, which will in turn provide the required security.
In the above figure, Cloud user may require PaaS or SaaS or IaaS. Cloud Services are accessed through Secure Socket Layer. Based on their request submitted through a Graphical User Interface (GUI), Virtual Machines are created to run the user services. The service is monitored through the Cloud Security Model (CSM).

Based on the user’s request, the cloud service provider instantiates the VM image and assigns it to the user. Hence, for ‘n’ user requests, ‘n’ Virtual machines are instantiated in cloud servers. Whenever required, the user can access their service through CSM. CSM has two major components, the Policy Manager (PM) and the Security Auditor (SA) as shown in Figure 7.2.

Policy Manager module is responsible for access control and message protection policy, which in turn provides the integrity of the data stored in cloud. Security Auditor explores the network traffic and audits the data stored in cloud. Requested data will be retrieved and returned to the indented user based on their request through CSM.

PM module contains the access control policy and enhanced message protection policy algorithm namely the Merkle hash algorithm. It generates a message digest along with actual data which in turn is sent to the Security Auditor. The Security Auditor explores the data communication network traffic using Nmap (Network Mapper). Vulnerability analysis and risks are identified periodically by Nmap, based on the assessment of risks, either by isolating the intruded virtual machine or by reconstructing the virtual machine images from the snapshots using hypervisor.
Services are accessed via HTTPS which assures data security at transit, and data is encrypted by the Policy Monitoring Module. Further, security is enhanced by continuously monitoring for vulnerable areas.

![Diagram of CSM components]

**Figure 7.2** Major components of CSM

Major components of CSM, PM and SA are described in figures (Figure 7.3a and 7.3b) respectively.

![Diagram of CSM’s Policy Manager (PM) module]

**Figure 7.3a** CSM’s Policy Manager (PM) module
Policy Manager module is a part of the CSM. It does the access control policy by using Java Authentication and Authorization Service. It also identifies the user accessing the service, their authorization limits and the operations being performed by them. Based on SLA, Policy is defined and agreed by the Cloud User and the Cloud Service Provider. Policy Maker and HMerkleDigest generator module generate the signature which has to be appended with the data, which assures protection policy for all the services in the cloud using message protection mechanism and in turn assures integrity of the data stored in CSP. Policy Checker verifies whether any of the data has been modified since last access by generating HMerkleDigest again and compares it with the previously generated hash which was sent along with the data.

Authentication module checks for the identity of the user accessing the service regardless of the service type and nature. Authorization module checks whether the user is authorized to access the service based on Service Level Agreement.

![Figure 7.3b CSM’s Security Auditor (SA) module](image-url)
Policy is generated based on the SLA agreement between the user and the provider. This work mainly concentrates on access control policy and message protection policy. Here policy varies with price and many other factors.

**Sub Components and their task**

**Policy definition based on SLA:** This component defines the Security Policy by adopting OWASP SMART requirement collections and follows SecSDLC for deploying the Policies. Here Policy varies with the various factors, mainly cost factor deciding the level of the Security.

**Authentication and Authorization Component:** JAAS is used here for providing access control of the Service.

**Policy Decision Maker and H-MerkleDigest Generator:** Message Protection is done by this component. It generates a signature based on a homomorphic property of Merkle Digest Generator extending with SHA1 algorithm for enhancing the security of the application. This signature is appended with the data that are sent to the cloud. Since signing is performed on the encrypted data, cloud service provider also can check the integrity of the data by once again performing the homomorphic encryption over the encrypted data and compare the signature to verify the integrity of the data. In turn this homomorphic property assures confidentiality, integrity, security to the application which is running on the cloud. This component generates a signature of the encrypted data based on homomorphic property using Merkle Hash Signature Scheme and this is appended with the data sent to cloud.
Policy Checker: This component computes the HMerkleDigest again and it compares with the previously generated hash, which was sent along with the data. If anybody has accessed or modified, hash value is different, which assures Security and Integrity.

Homomorphic Encryption: It is a form of encryption which allows specific types of computations such as combination of addition and multiplication that can be performed on cipher text and an encrypted result is obtained, which when decrypted matches the result of operations performed on the plaintext. For instance, one person could add two encrypted numbers and then another person could decrypt the result, without either of them being able to find the value of the individual numbers.

Merkle Digest: Ralph Merkle built a digital signature scheme based on hash trees and is an alternative to traditional digital signature schemes based on RSA. The Merkle signature scheme is a digital signature scheme based on hash trees developed by Ralph Merkle and is an alternative to traditional digital signatures such as the Digital Signature Algorithm or RSA.

Merkle depends on the existence of secure hash functions (SHA). SHA is based on MD4 and works like MD5, which generates 128 bit hash, whereas merkle digest generates 160 bits. This makes the Merkle Signature Scheme more secure. To sign a message $M$ with the Merkle Signature Scheme, the message $M$ is signed with a one-time signature scheme, resulting in a signature, $\text{sig}'$, first. This is done, by using one of the public and private key pairs $(x_i, y_i)$. 
The receiver knows the public key $pub$, the message $M$, and the signature $sig = (sig' || auth_0 || auth_1 || \ldots || auth_{n-1})$. At first, the receiver verifies the one-time signature $sig'$ of the message $M$. If $sig'$ is a valid signature of $M$, the receiver computes $A_0 = H(y_i)$ by hashing the private key of the one-time signature. For $j=1,\ldots,n-1$, the nodes of $A_j$ of the path $A$ are computed with $A_j = H(A_{j-1} || auth_{j-1})$. If $A_n$ equals the public key $pub$ of the merkle signature scheme, the signature is valid. i.e. here H-MerkleDigest Generator generates 160 secret values $x_i$ and 160 secret values $y_i$, whereas Merkle extends to SHA1 outputs a 160-bit value.

Nmap is the free Security Auditor for network exploration and security audit files. It is an open source program released under the GNU General Public License. A security tool which supports most of the operating systems, including Linux, Microsoft Windows, FreeBSD, OpenBSD, Solaris, IRIX, Mac OS X, HP-UX, NetBSD, Sun OS, Amiga, and more. It can be used to discover, monitor, and troubleshoot TCP/IP systems. Nmap is a free cross-platform network scanning utility tool.

JAAS can be used for viewing access information and their authorizations. JFTP (FTP website. Available on: http://j-ftp.sourceforge.net/) protocol is used to provide the service between the cloud user and cloud provider.

Proposed CSM works as follows. Upon registering with the cloud, based on access control policy through PM, virtual machine images are created and assigned to the users. So there can be ‘n’ number of virtual machines in a single provider.

Access Control Policy code snippet is given in Figure 7.4a. Message Protection Policy code snippet is given in Figure 7.4b.
public int getServerUsers (int srvid, String uname, String passwd) 
.
.
.
ResultSet rscure = st.executeQuery ("select * from serverusers where srvid='"+srvid+"' and srvusername='"+uname+"'and srvpasswd='"+passwd+"'");
.
.
stmt = "UPDATE `filetable` SET `flmodate` = "'+dts+'", flsignature="'+SHA1.getDigest (file, clrsid+":"+loc+":"+dts) +'", flstatus='updated' WHERE `fileid` ="'+fileid; 
.
.
Figure 7.4a Code snippet of access control policy

```
public class HMerkleDigest extends SHA1Digest {
    public String getDigest (String file, String other) {
        MerkleSHA1 merkle = new MerkleSHA1 ();
        int max = 10;
        byte [] message = other.getBytes ();
        merkle.append (file, message);
        return merkle.digest ();
    }
}
...
```

Figure 7.4b Code snippet of message protection policy using homomorphic function calculating merkle hash
public class CloudFTP
{
  .
}
class VerifyFile implements Runnable
{
  private SHA1Digest SHA1 = new MerkleDigest();
  ..
}
class UpdateFile implements Runnable
{
  private SHA1Digest SHA1 = new MerkleDigest();
  .
}
public final class DbCon
{
  public static Connection getConnection () {
    String DB_CONN_STRING = "jdbc:mysql:// ";
    String DRIVER_CLASS_NAME = "com.mysql.jdbc.Driver";
    .
  }
}
public class CloudMain
{
  static CloudMain mt;
  .
}
public class CloudLogin
{
  ..
}
public class SecAud
{
  ..
}
public class HMerkleDigest extends SHA1Digest {
  public String getDigest (String file, String other) {
    ..
  }
}

Figure 7.5 Code snippets of CSM cloud
CSM Module Code snippets are provided in figure 7.5. This code snippet highlights the some of the modules used to implement the CSM cloud.

- CloudFTP module is used to transfer data securely to and from cloud to the service user.

- VerifyFile module checks for modifications in the file since last access.

- Update file module updates the data file with Merkle hash in the cloud.

- DbCon Module takes care of the database connectivity between the service provider and the service user.

- CloudLogin module allows the user securely to access their service using single sign on.

- SecAud module generates the monitoring and vulnerability analysis report.

- HMD module generates the homomorphic digital signature which has to be appended with the service data to provide the integrity of the data.

The code snippets given in Figures 7.4 and 7.5 are used to provide secure service. JFtp is used to provide service between the cloud user and the cloud service provider. JFtp used here is free and distributed under the GNU public license.
The following Pseudo code highlights only the major steps involved in this to implement the CSM cloud.

**Pseudo code of CSM**

Let ‘A’ be the Cloud user and ‘B’ be the Cloud Service Provider.

1. ‘A’ sends request to ‘B’ through CSM about his requirement

2. Based on the requirements, virtual machine(s) is (are) created using hypervisor.

3. ‘A’ can send his data to ‘B’ by applying Homomorphic function by calculating the Merkle Hash

4. CSM provides enhanced secure cloud services using the following method

   a. PM’s Authentication and Authorization component checks whether the user is authorized one, to do the operation such as managing the data stored in CSP.

   b. PM’s Policy Maker and HMerkleDigest generator component generates the signature which has to be appended with the data which assures the integrity of the data stored in CSP; Policy Checker component checks for whether data has been modified since last access.
c. Security Auditor (SA) responsible for Vulnerability Assessment. It is performed using the following steps.

i. For \( i = 1 \) to \( n \), scan and monitor all the ports of \( \text{VM}_i \)

ii. Check whether services are accessed by authorized user

iii. Take the snapshots of \( \text{VM}_i \) periodically

iv. If there is an unknown service, take proper security control, Worst case destroy that \( \text{VM}_x \)

v. Reconstruct the \( \text{VM}_x \) using the snapshots taken already or from the fresh image

vi. If there is heavy load clone the \( \text{VM}_c \) to manage that particular VM

vii. Repeat the process from step i.

5. ‘A’ can access its data anytime from anywhere through CSM.

If they are new users, they can register and continue to access their service. Even resources can be added or scaled up based on the request from the user. The HMD generator component allows the users to access their service in a secured way. It generates a merkle hash and appends it with the user data which has to be stored in cloud, and also checks whether the service is modified since the last access or not, service is intruded or not. It can be verified through HMD generator component to assure the
enhanced secure cloud services. Data transferred to the cloud is shown in Figure 8.7

Table 7.1 Monitoring and vulnerability analysis report

<table>
<thead>
<tr>
<th>Port</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>tcp tcmux</td>
</tr>
<tr>
<td>3</td>
<td>tcp compressnet</td>
</tr>
<tr>
<td>7</td>
<td>tcp echo</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td>tcp ftp-data</td>
</tr>
<tr>
<td>21</td>
<td>tcp ftp</td>
</tr>
<tr>
<td>22</td>
<td>tcp ssh</td>
</tr>
<tr>
<td>23</td>
<td>tcp telnet</td>
</tr>
<tr>
<td>24</td>
<td>tcp priv-mail</td>
</tr>
<tr>
<td>25</td>
<td>tcp smtp</td>
</tr>
<tr>
<td>26</td>
<td>tcp rsftp</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>80</td>
<td>tcp http</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>8080</td>
<td>tcp http-proxy</td>
</tr>
<tr>
<td>8081</td>
<td>tcp blackice-icecap</td>
</tr>
<tr>
<td>8082</td>
<td>tcp blackice-alerts</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>65129</td>
<td>tcp unknown</td>
</tr>
<tr>
<td>65389</td>
<td>tcp unknown</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>65535</td>
<td>tcp unknown</td>
</tr>
</tbody>
</table>

Table 7.1 shows the Monitoring and Vulnerability Analysis Report generated by Security Auditor. This table data is obtained by summarizing the results of port and service scanning by Nmap Security
Auditor. SA monitors and checks for a user’s authentication and authorization along with the port being accessed. If there is any hidden or unknown service, appropriate steps can be taken, based on the vulnerability assessment report. Even by taking appropriate snapshots, the system can be brought back to the previously running state. In worst case scenario, the entire Virtual Machine (VM) can be destroyed and it can be reconstructed with the help of the snapshots taken, without interrupting other users.

Services and their port usage are shown. By carefully monitoring the resource utilization like CPU, memory, network, port, isolating the intruded Virtual Machine is possible. By monitoring specific port as well as services, threats can be identified. So based on risk assessment, CSM reacts either by preventing further running of these machines or by reconstructing these machines from the images taken periodically as per policy.

By monitoring Internet protocol ports along with all other ports, vulnerability report can be generated. Port such as 80 for Hypertext Transfer Protocol (http) which is commonly used for Web servers, http secure sockets (https) in port 443 for secure Web Communication, Domain Name Service (DNS) for resolving names to IP addresses in port 53, File Transfer Protocol (FTP) for transferring data between systems in ports 20 and 21, Secure Shell Protocol (SSH) in port 22 for encrypting communications, telnet in 23, Simple Mail Transfer Protocol (SMTP) in port 25, Windows file sharing, login and Remote Procedure Call (RPC) in ports range 135 to 139 and 445, Network Time Protocol (NTP) for network time synchronization in port 123, Internet security Association and Key Management Protocol (ISAKMP) key negotiation for Secure Internet
Protocol (IPSec) in port 500, Session Initiation Protocol (SIP) for Voice over IP (VoIP) in port 5060. All the above protocols will be used along with either TCP or UDP. Continuous Scanning of well known Ports ranges from 1 to 1024 and in special cases scanning specific ports like most commonly used internet ports is possible, since cloud computing itself accesses service through the internet. The above discussed monitoring technique along with virtualization, which helps the provider to achieve full security of the virtual machine, is achievable.

The next phase is to perform process level monitoring. Process level monitoring, involves logging every transmission and transaction that occurs at the process level and hence help us analyze the data and arrive at much fine grained information which could be further used for enforcing SLA/Policy Violations at various levels of the service platform. Tier wise monitoring/consolidation with separate focus on tier specific metrics and tier specific processes will help better understand and enforce policy monitoring at a finer level thereby ensuring better coverage of all the elements of a large system.

7.3 SUMMARY

In this chapter, the cloud security is achieved by using virtualization and policy monitoring by effectively using “Cloud Security Model (CSM)”. The proposed CSM has two major components. Policy Manager module (PM) and Security Auditor module (SA). PM module embeds the access control policy and enhanced message protection policies namely Homomorphic Merkle hash algorithm (HMDigest Generator).
The Security Auditor module explores the data communication network traffic using Nmap (Network Mapper). The risks are identified by using vulnerability analysis. Based on the assessment of risks, counter measures are taken and the security of the cloud services is assured. It has been identified that by combining the two techniques, the system is made more robust and effective in terms of security and functionality. Next Chapter 8 discusses the deployment details of Hypervisor, CSM and results obtained in detail.