A DYNAMIC POLICY-BASED SECURITY-AS-A-SERVICE INFRASTRUCTURE FOR CLOUD ENVIRONMENT

Sumathi D[1], Poongodi P[2], Krishnan R[3]

[1]*Asst. Prof. in CSE, Jayam College Of Engineering and Technology, Nallanur, Dharmapuri, Tamilnadu, India*
[2]*Professor in ECE, V.S.B. Engineering College, Karur, Tamilnadu, India*
[3]*Sr. Manager (Engg.), Novell Software Development India Pvt. Ltd., Bangalore, Karnataka, India*

Abstract

In the cloud environment, implementing and managing information security is extremely challenging in not only cloud end but user end also. The organizations would critically examine several factors of cloud hosting provider and security is the most important one. Cloud hosting providers have the obligation to support Quality of Service (QoS) and Service Level Agreement (SLA) parameters such as strong security, zero-tolerance downtime and intrusion detection & prevention mechanism. Cloud hosting provider may not have the required resources and mechanisms to update the security infrastructure regularly. If compromised, the hosted applications would be prone to attacks and vulnerabilities resulting in application downtime, loss of data and theft resulting in loss of trust with customers. A collaborative approach among various security providers is the need of the hour. The outcome of this collaborative approach is the design and implementation of a policy-based security as a Service (Sc-aaS) system that would have the capability of dynamically inquiring and provisioning security services for the cloud hosting provider in a transparent manner as per the requirements sought by the hosted application.

Keywords: *Software-as-a-Service, Cloud Security Provider (CSP), Cloud Hosting Provider; Security Policy, WS-security, WS-policy, WS-Agreement.*

----------***----------

1. INTRODUCTION

Cloud computing is an on-demand Internet-based virtual service model where data is stored and accessed using distributed on-demand elastic services in a transparent manner. Internet has grown at an unimaginable pace. With the advent of cloud computing, business houses have found it easy to host their applications on a virtual environment hosted and managed by a cloud service provider [1]. The notion of on-premise application deployment is fast fading out. This is known as Software-As-A-Service (SaaS) in cloud computing world.

This gives organizations the flexibility to choose the best cloud hosting provider based on parameters such as cost, reliability, performance, storage, security, availability and etc. Enterprises believe that it is possible to reduce the Total Cost of Ownership (TCO) by adopting the SaaS model. With increased and distributed computing power coupled with extensive storage, several IT majors are embracing SaaS. In spite of the benefits enjoyed in SaaS model of deployment, quite a few organizations are reluctant to adopt SaaS inorder to host their applications due to the following reasons.

- Sensitive Information
- Attacks
- Insufficient Security Levels
- Lack of Trust

The enterprises except the cloud hosting provider to get their systems and practices certified by authorized certifying bodies before signing SLAs [3]. Although hosting providers try to combat security risks by strengthening their infrastructure by means of regular security updates, it becomes insufficient and obsolete when new threats and attacks emerge.

According to Mohemed Almorsy et al (2011) [11], obtaining a security certificate such as ISO 27000 or NIST-FISMA would help cloud providers improve customers trust in their cloud platforms' security. The framework discussed is based on improving collaboration between cloud providers, service providers and service consumers in managing the security of the cloud platform and the hosted services. It is built on top of a number of security standards that assist in automating the security management process.

The security standards prevailing today are for regulat non-cloud deployments which cannot be directly applied to a cloud environment. This is because of the nature, operational artifacts and the architecture of the cloud environment.

Security officers should be aware of concerns experienced in the cloud environment. Gartner [5] has conducted an investigation regarding information security issues that should be considered when dealing with cloud computing. In their

investigation report, they have highlighted seven security issues with respect to access, compliance, data loss and recovery etc. Policy-Based Security-as-a-Service (Sc-aaS) is an approach towards provisioning security among the cloud hosting providers. It refers to security services entrusted by a security policy using the SaaS model.

2. THE NOTION OF TRUST

Trust is a social affair; it is a popular approach to frame the dynamics of inter-group or intra-group interactions. Since security is complex and multi-fold in nature, the notion of trust becomes even more complex. As per Gartner report, the security delivered as a cloud service will more than triple in many segments by 2013[8]

While this seems to be very encouraging and promising, the question that arises is to do with trust and collaboration. While hackers will continue to find new ways of attacks on hosted services, cloud providers have to arrive upon new and updated mechanisms to counter these attacks. Among all these, the notion of trust plays a critical role in selecting the right hosting provider who has the best of breed security infrastructure and is possibly less vulnerable.

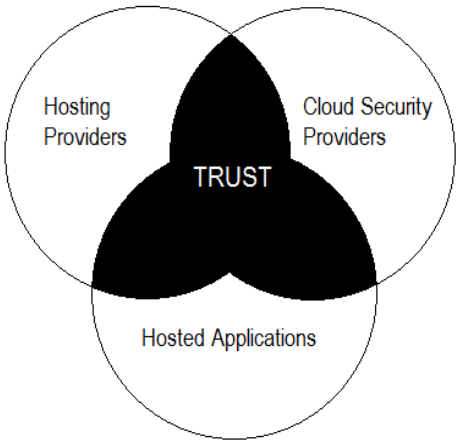

Fig 1: The Notion of Trust among the three SaaS players.

While no single hosting provider would be able to provide all kinds of security mechanisms that the enterprises want, there may be other providers who may be ready to offer security services for a nominal fee. A collaborative approach among the hosting providers by re-using the security infrastructure would make the whole SaaS model secure and trustworthy.

Figure 1 depicts the notion of trust among the three SaaS players; the Cloud Hosting Providers (CP), the Cloud Security Providers (CSP) and the Hosted Applications. The shaded area depicts the synergy, collaboration and trust that need to prevail between the three for a successful and secure implementation of Sc-aaS model.

3. SECURITY POLICIES

The objective is to design a system using which a cloud hosting provider cloud specify security service needs, select the best-fit security service from other cloud providers and provision the same into the security infrastructure for use by the hosted vendor application. The security service requirement of the end user application would be fed into the system using a well-designed interface.

The cloud provider would translate the security requirements of the application in form of an XML policy using a standard protocol called extensible Access Control Markup Language (XACML) an XML request-response language for defining security policies.

Vladimir Kolovski et al (2007) [9] have defined that a policy set is composed of a set of policies, where each policy is divided into a set of rules. Through XACML-based security policies, the system would allow a cloud hosting provider to discover, identify and collaborate with another cloud security provider in order to rent and get access to security service. The rented security services could be enforced and used to provide security service to cloud applications hosted by the cloud provider. This has to be transparent to the cloud applications.

The cloud provider will have to negotiate legitimacy with a CSP. This would be done using WS-Agreement protocol [2]. WS-Agreement protocol is used to create and manage SLAs in distributed systems [10]. The use of WS-Agreement brings in a standard nomenclature for CSPs to communicate and negotiate key SLAs which is a basic requirement to be fulfilled before collaborating and accessing the security services.

4. POLICY-BASED SC-AAS MODEL

Although cloud computing gives extremely verticals would have varied requirements and different levels of security needs. Application vendors and enterprises might have to deal with multiple hosting providers each offering different security services offerings.

It becomes extremely cumbersome and near to impossible for application vendors to statically switch between different security providers. This leads to complexity in terms of identifying and using security services. As of now there are no proven mechanisms to dynamically discover and use cloud-based security services. The system accepts the details of the security service that needs to be enforced and generates a security policy out of it. At this point, it may be assumed that the security policy uses an internal data structure.

A security policy would contain the type of security service that is needed with key QoS parameters that should be used to identify the best fit CSP who could offer the service. The system would discover the matching CSPs either from the

private/public cloud or from the local security service registry; a database of CSPs that were previously discovered. CSPs that satisfy the security policy would be recorded in the registry for future use. Once the required CSPs are discovered, the system identifies the best fit CSP. This would be based on the key QoS parameters which may be satisfied by one or more CSPs.

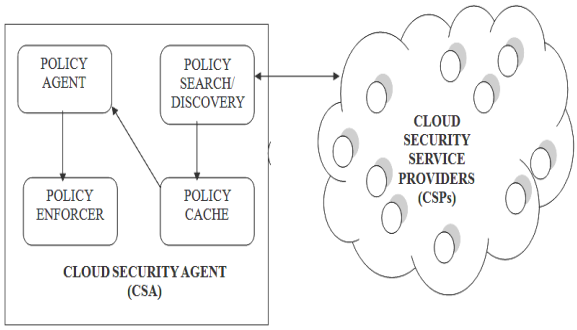

Fig 2: The interacting modules of the policy-based Sc-aaS system.

The Cloud Security Agent (CSA) is the core component of the system. It is responsible for discovering, caching, identifying and enforcing security services obtaining from the third-party CSPs.

Policy Agent is the core component of the service for use with the hosted application. It uses policy search/discovery to search and discover security policies from CSPs. It uses the policy enforcer to enforce the policy on the hosting provider. It caches the discovered policies in the policy cache before selecting and enforcing it.

Policy enforcer enforces the security policy on the hosting provider. The process of policy enforcement contains parsing the policy. Reading the policy details and transacting with other CSPs to access the security service from them. Once the policy is enforced, the hosting provider can use the inherited security service against hosted applications till it expires.

Policy cache stores the security policies which may not have been enforced by the policy agent. The policy cache must be designed to accommodate several security policies that could be discovered in the cloud.

Policy discovery searches/discovers security services from different security providers. It is used by the CSA when the hosted application requires different security service which is not available in the policy cache.

4.1 Structure of a Security Policy

A security policy may have one or more security services that it could render. Every security service has an area representing which area this security service caters to. For example, **Area="Security Scanning"**, indicates that this security policy has services which can scan the hosted application for possible vulnerabilities. The status attribute indicates if the service is currently active or inactive. The expiry attribute would contain the date by when the service would expire beyond which the cloud provider has to renew the service after re-negotiation. A security service area may have the ability to offer more than one security service.

4.2 System Modules and Interactions

A Cloud Vendor Application requests security service by creating a security policy and sends it to the cloud provider which hosts the application. The Cloud Provider invokes the CSA. Once invoked, the cloud security agent does multitude of tasks. It uses the security policy to discover the list of Cloud Security Providers who have the capability of serving the requested security service.

The Cloud Registry Service may perform a lookup of matching CSPs from the Cloud Service Registry. This would fetch responses from several CSPs. Depending upon the availability of the security services; one cloud get responses from thousands of CSPs The security policies would also be stored in the Cloud Service Registry for further lookups. It is the responsibility of the CSA to select the best fit security policy.

The Cloud Security Agent then negotiates with the CSPs using the WS-Agreement protocol. It sends a Security Token Service (STS) token to the CSP to establish its identity and creating a mutual trust between the CP and the CSP.

Once an agreement is reached and it the CP decides to go with CSP, the CSP sends an STS-ACK token concluding its final agreement. Thus, the CSP is identified else, the CP would start negotiating with the next CSP and the process continues till an appropriate CSP is selected.

The CSA enforces the security service on the CP which returns the access point of the security service to the cloud application. Having obtained the access point, the cloud application invokes and accesses the security service. This access is allowed till the expiry time decided during the negotiation process between the CP and CSP after which the service expires.

4.3 Ranking of Ordering of Cloud Security Providers

There will be several matching security policies during the search/discovery process. So, the system needs to identify the best-fit security policy based on several QoS parameters so that the most preferred security service could be provided to the vendor application. It is necessary to rank order the obtained CSPs. Saurabh Kumar Garg et al (2011) [8] have proposed a framework and mechanism, which measure the quality and prioritize Cloud Services.

Table 1: QoS Parameters and Their Weights

S.No	QoS Parameter	Weight Out of 10
1	Security Service Type	6
2	Security Service Strength	9
3	Number of Key Deployments	5
4	Reputation index	7
5	Cost	6
6	Reliability Quotient	7
7	Service Management (Self/Third party managed)	4
8	Response Time Index	5
9	Ease of Access	6
10	Duration	7
11	Number of Cloud Providers using this service	4
12	Satisfaction Index	4
13	Serviceability Index	6

A process called Analytical Hierarchical Process (AHP) based ranking mechanism is used to implement this. The CSP ranking index is computed based on the weights assigned to each QoS parameter as specified in Table 1. The higher the weight has higher the importance of the parameter.

4.4 Negotiation SLAs

SLA negotiation is critical to the system and is used to negotiate key service agreement before a contract is established between two providers; the CP and CSP in this case. The implementation uses WS-Agreement protocol as a negotiation algorithm which is available as part of WSAG4J toolkit. The key to WS-Agreement is the availability of an agreement template. The agreement template is akin to the SLAs that we have been talking about is filled by the CP when offered by the CSP.

The template that comes from the CSP contains the service description of the security service as well as a set of options that the CP can choose from. The service description contains the available resource description and the time-frame till the resource is available. The CP fills the details of the template, it sends it back to the CSP as an offer. The CSP checks whether the requested service can be provisioned (in WS-Agreement terminology). In case the requested service can be provided, the CSP sends back a completed counter-offer to the CP.

After receiving the counter-offer the CP would created a negotiated agreement with the CP based on the offer. The CP and CSP can authenticate each other using a STS in order to form a security channel and the CSP sends back the security services detailed as an XML message. The XML message contains all the details that the CP needs to identify, select and enforce the service for the vendor application.

One thing that is critical to understand here is that the period of service availability is finite. This is decided by the expiry time that is attached to the service. After the elapse of the expiry time, the CSP would require a re-negotiation of the agreement which would need a new template to be filled in by the CP. The process starts all over again till a new agreement is obtained a XML message with service description is passes on from CSP to CP.

4.5 Selecting the Best-Fit Policy

Once CSPs are ordered based on their ranking commensuration with the QoS weights, it becomes pretty straight forward to identify the best CSP. The procedure to identify the best CSP would be as follows.

- Sort and Retrieve the top 5 CSPs from the service registry. The number of CSPs to retrieve could be a configuration option in the system.
- Check if all services offered by the CSP are active and not expired.
- If the services are not active or expired, check the next CSP.
- If top 5 CSPs have inactive or expired services, perform the inquiry operation with the next 5 CSPs.
- If any of the CSPs have both active and unexpired service, then return the CSP as the best CSP.

The best CSP evaluation procedure ensures that the best active CSP with their services intact gets a chance to participate in service offerings although it doesn't stand higher in the ranking index.

4.6 Enforce Security Policy

Depending upon the demand of the hosting provider, one or more security policies can be selected and enforced by the Policy Enforcer. That would fall under the bracket of 'group policies'. The process of policy enforcement is to make the security services from different vendors available to the hosting provider so that it could be used by the hosted applications.

Once enforced, the security services can be accessed by the hosting provider by parsing the security policy. The cloud hosting provider has to establish some kind of trust with the CSP. This could be done

- By registering as a valid hosting provider with the CSP and authenticating before accessing the security service.
- By means of exchanging digital certificates.
- Using secure token service (STS)

WS-Trust is a specification from OASIS standards group that deals with the issuing, renewing, and validating security tokens to establish mutual trust between two parties for secure message exchange. Validating the security policy is important

and acts as a filtering mechanism to disregard security policies that have been tampered, corrupted/altered or with inactive services.

Since the security policy is got from a CSP, it also acts as a mechanism to perform compliance check to see if there are missing elements which might be mandatory for establishing trust and accessing the security service. The enforcement of security service by the policy agent is shown in the below procedure.

- Validate security policy to check if confirms to standard norms
- Parse the policy XML document
- Identify the security service that is required to be enforced
- Check the status is inactive, it would be required to go back and fetch the next best CSP.
- If the status is active and expiry date shows service expired, then re-negotiate with CSP and get the service re-issued.
- If the status is active and the service has not expired, get the service access URL
- Pass the service access URL to the cloud hosting provider.

Enforcing the security service onto the CP makes it available and accessible to the hosted application. It is interesting to note that the entire enforcement architecture acts like a service broker. What the security service does and its intended use by the hosted application is unknown to the CSA and the policy agent within it. Once the service access point is passed onto the cloud provider, it is the responsibility of the hosted application to make best use of it. However, if the service expires, it is the responsibility of the CSA to ensure that the service is no longer made available to the hosted application.

4.7 Fault Tolerance to Cloud Providers.

Since the security service rented is transparent to the vendor application, it is the responsibility of the CP to provide uninterrupted and quality service as per the SLA signed between the vendor application and the CP. CSPs are external entities with which the system has negotiated number of SLAs based on key QoS parameters; it is absolutely necessary to keep track of the health of the CSP.

The policy agent would send a dummy beacon policy indicating the status of each service. Since the policy agent keeps track of the CPs on which the security services have been enforced, it is easy to ascertain those services which needs to be active at all times. The following procedure gives the details of the health check process

- For each CSP that have been enlisted, send a beacon policy
- If the reply is not received within specific timeout period, it may be assumed that the CSP is down. One of the two actions could be performed.
  1. Quickly identify the next CSP in the service registry which could provide similar service and enforce the service immediately.
  2. Inform the CP that the CSP is down and the service may not be available for some time.
- If the CSP responds, check the value of status and expiry attributes that are returned by the CSP.
- If the status is inactive, then the procedure to ensure fault tolerance may be adopted or the CP can be informed that the service is inactive and may not be available for some time.

5. TESTING

5.1 Agent Policy Interface Testing

The mechanism here is to create a database of several policy descriptors each catering to a specific security policy need. The following tests were carried out

- Test with policies cached in the service registry
- Tests with policies not cached in the service registry
- Tests with one policy that is bound to match the search criteria
- Tests without the matching policies in the service registry
- Tests with more than 100 matching policies in the service registry
- Tests with varying sizes of the security policies
- Test with invalid security policies
- Negative test condition where the discovery service is stopped to verify the behavior of the agent

5.2 Testing by Varying QoS Weights

There will be several matching security policies during the search/discovery process. So, the system needs to identify the best-fit security policy based on several QoS parameters so that the most preferred security service could be provided to the vendor application. Testing is done by computing the CSP ranking index from the weights assigned to each QoS parameter. The following tests were conducted.

- Tests by randomizing the weights
- Tests by assigning fixed weights with giving importance to specific parameters
- Tests with refresh rates of QoS Parameters

5.3 Testing the Selection of Best-Fit Policy

Once the CSPs are ordered based on their ranking commensuration with the QoS weights, it becomes very easy and pretty straight forward to identify the best CSP. When the agent performed a discovery operation, a query was sent to all the gateway web servers which in turn picked up information from different CSPs to which it had access to. When the

gateways were provisioned dynamically and were sufficiently large, the CSPs obtained were faster than a static list of gateways. Each gateway had to queue to CSP information and send back the details to the agent one after another which reduced the system response time. Hence, we could conclude that it is necessary to provision in the gateway web servers depending upon the load of the system. In a cloud-based provisioning model, this can be easily done by cloning and deploying a Virtual Machine (VM) from the snapshop.

The MapReduce algorithm helped in quickly processing CSPs on the agent side. It is seen that when the number of CSPs returned from gateway servers is large (in the order of several hundred), the regular string compare or even comparing using regular expression takes upto a minute. When the same thing is done using MapReduce, the performance is reduced to few seconds as mentioned in the below table

Table 2. The Performance of the CSP Discovery Process with Varying Loads.

#Matching CSPs	Regular String Compare (Avg. Search Time)	Regex (Avg. Search Time)	MapReduce (Avg. Search Time)
50	Upto 5 Secs	Upto 5 Secs	More than 5 secs
500	Upto 8 Secs	Around 5 Secs	Aroung 5 Secs
2000+	Upto 54 secs	Upto 23 secs	Upto 6 secs

5.4 Testing SLA Negotiation and Fault Tolerance

In order to test the SLA negotiation, a negotiation template was created as per WS-Agreement protocol with the required SLA details and sent to the best-fit CSP identified. It was found that whenever the SLA negotiation performance of the system is directly proportional to the number of failed negotiations plus the number of CSPs found and cached in the service registry. While conducting the tests with negotiated SLAs, two attributes were modified dynamically, Service Status and Expiry Date. The system date was deliberately modified to a date after the expiry date. It was found there was no disruption in the transaction that is in progress. The system rejected new transactions demanding renegotiation of the SLAs. After the SLAs were re-negotiated successfully, the system processed the next transaction. During this time, the cloud vendor cloudlet got suspended as the security service failed to start. This is a normal scenario and is working as designed.

In order to test the service status, the web service running against the gateway service was brought down and the policy agent was refreshed by giving an external refresh command. The output showed that the status field changed to inactive and the system was unable to talk to the security service. When the gateway service was brought up and the policy agent refreshed, the output showed that the status field changed to active and the system was able to talk to the security service and started functioning normally.

The tests to check the fault tolerance of the system was done in the following way. The service registry was filled with three negotiated CSPs offering the same security service. The best-fit CSP was shutdown. After few seconds, it was noticed that the system started reading the security information from the second CSP. The second CSP was shutdown and the system started reading information from the third CSP.

6. CONCLUSIONS AND FUTURE WORK

The debut of the Cloud Computing has made the management of information security the most significant and critical issue to research and solve. The concept of policy-based security service offering in cloud computing has deeper and wider impact on enterprise computing managing information security. As more vendors jump into the cloud hosting band-wagon trying to provide hosted services on the internet cloud, dealing with security continues to be a major challenge. Collaboration among different CSPs and hosting providers to leverage each other's strengths and hosting providers to quickly and transparently resolve security issues and maintain the SLAs.

In this paper, we have illustrated the design of model of the CSA by using a Policy-Based Sc-aaS Infrastructure, which specified the requirements by means of a Sc-aaS poicy, to efficiently discover security services among cloud security providers.

REFERENCES

[1] Ahmed E. Youssef, (2012), "Exploring Cloud Computing Services and Applications", Journal of Emerging Trends in Computing and Information Sciences, ISSN 2079-8407

[2] Alian Andrieux, Asit Dan et al, (2007), "Web Services Agreement Specification (WS-Agreement)", Open Grid Forum, GFD-R-P.107, pp 17-28

[3] Balachandra Reddy Kandukuri, Ramakrishna Paturi and Atanu Rakshit, (2009), "Cloud Security Issues," in Proceedings of the 2009 IEEE International Conference on Services Computing, pp. 517-520.

[4] Boudaoud. K, Guessoum Z, McCathie Nevile C, (2010), "Policy-based Security Management using a Multiagent System"

[5] Jay Heiser, Mark Nicolett, (2008), "Assessing the Security Risks of Cloud Computing", Gartner Research, ID Number: G00157782

[6] Mohamed Al Morsy, John Grundy and Ingo Müller,

(2010), An Analysis of The Cloud Computing Security Problem", APSEC 2010 Cloud Workshop, 2010

[7] Rajkumar Buyya, Saurabh Kumar Garg, Rodrigo N. Calheiros, (2011), "SLA-Oriented Resource Provisioning for Cloud Computing: Challenges, Architecture, and Solutions", International Conference on Cloud and Service Computing, 978-1-4577-1637-9/11

[8] Saurabh Kumar Garg, Steve Versteeg and Rajkumar Buyya, (2011), "SMICloud: A Framework for Comparing and Ranking Cloud Services", Fourth IEEE International Conference on Utility and Cloud Computing, 978-0-7695-4592-9/11

[9] Vladimir Kolovski, James Hendler, Bijan Parsia, (2007), "Analyzing Web Access Control Policies", International World Wide Web Conference Committee (IW3C2), ACM 9781595936547/07/0005

[10] Wolfgang Ziegler, Philipp Wieder, Dominic Battr´e, (2008), "Extending WS-Agreement for dynamic negotiation of Service Level Agreements", CoreGRID Technical Report Number TR-0172, Project no. FP6-004265

BIOGRAPHIES

Mrs. D. Sumathi is working as Assistant Professor in the Department of CSE, Jayam College of Engineering and Technology, Dharmapuri since 2010. Her research interests include Cloud Computing, Information and Network Security, Mobile Ad Hoc Networks.

Dr. P. Poongodi is currently working as Professor in the Department of ECE in V.S.B. Engineering College, Karur. Her research interests include Wireless Sensor Networks, VLSI and Cloud Computing.

Mr. R. Krishnan is working as Senior Manager (Engineering) in Novell Software Development India Pvt. Ltd., Bangalore. His research interests include Big Data Analysis, Network Security, Virtualization Techniques and Cloud Computing.