International Journal of Precious Engineering Research and Applications (IJPERA)
www.ijpera.com ISSN: 2456-2734,
Volume 3, Issue 3 (Sep-Oct) 2018), PP. 05-09

Application based Quality of Service in Mobile Networks with cloud architecture

Mr.Mada Vamshi Krishna, Dr. K V G Rao
Reaearch Scholar Hydaerabad
Professor & HOD of CSE GNITS, Hyderabad.
Corresponding Author: Mr.Mada Vamshi Krishna

ABSTRACT- Cloud computing is a grid based application which facilitates "On demand network access to a shared pool of computing recourses". This environment strives to be secure, scalable and customized with guaranteed quality of service (QoS). However, QoS is guaranteed through fulfillment of non-functional requirements like Security, Scalability, Mobility and Virtualization in cloud computing System.
In this paper we proposed "the OCSA algorithm" is an approach to fulfillment of our goal. To meet the computing demands of everyday operations like Nonfunctional requirements, here we have introduced OCSA to achieve those "First security, the misuse case and Attack Tree analysis posed threat and attack surface of any cloud. Second scalability, we should overcome network traffic congestion and DoS because of rapid growth of customers. Third, storage and Service providing consequences arises and mitigated to achieve Mobility and Virtualization. The above unpredictable constraints can be modeled through DREAD analysis, So that above consequences partially annihilated and use of cloud can be extended by adding more capacity on demand.
KEY WORDS: Cloud computing, Security, Scalability, Mobility, Virtualization, Attack analysis, misuse case, SDLC, Misuse case, Attack tree, DREAD, Mitigation.

Date of Submission: xx-xx-xxxx Date of acceptance: xx-xx-xxxx

---

I. INTRODUCTION

The philosophy of software engineering emerged from different techniques have been proposed to establish software development as a part of mainstream engineering that includes formal techniques of functional and non functional requirement elicitation and maintenance of software.

All these techniques that dealt with software engineering are focused on functional requirements. Functional requirement is defined as "behavior of the software system for desired input", whereas the nonfunctional requirement relates to "how the system is expected to behave with undesired input or in an unpredicted environment". Nonfunctional requirements generally were defined as constraints in the system. In software engineering these constraints were generally handled outside of development life cycle. Security, Scalability, Mobility and Virtualization are defined as non-functional requirement and looked as constraints in software.

For long time, data networks were closed – security within these networks were ensured through isolation. The trusted private LAN (Local Area Network) was isolated from the untrusted public networks like internet through firewalls to ensure that adversaries and hackers cannot intrude into the private network and steal valuable assets. To secure assets, other security mechanisms like Proxies, IDS (Intrusion Detection System), IPS (Intrusion Prevention System), Anti-Virus, Malware catchers etc were also installed. The belief was that an applications and assets used by the organization can be secured through in-vitro perimeter security; therefore, software engineering techniques never looked into security as an important component in Software Develop Life Cycle (SDLC), therefore, defined security as nonfunctional requirement [1]. In a conventional deployment model be it mainframe, client-server, or Web – servers are isolated and run on separate hosts within a defined boundary – allowing some kind of physical security. In Cloud Computing paradigm however, platform and infrastructure are virtual making this boundary non-existent and insecure. The main motivation towards going for Cloud is scalability.

In Cloud there is no capital investment – it is mainly revenue expenses, users pay as they use. In Cloud, users use Platform as a Service (PaaS), Infrastructure as a Service (IaaS), and Software as a Service (SaaS) – supply of resources in the Cloud increase or decreases on demand. In mainframe or client-server or even in Web deployment paradigm security was in-vitro, focus was mainly on network security. However, in recent times many applications have moved out of the private network and being deployed as Web Services. These are accessible to genuine users and hackers alike; therefore, security must be part of the application to protect itself from security threats. Application security will however be over and above the perimeter network security. To achieve this, security now need to be treated as functional requirement and must be part of SDLC. Many authors [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, and 11] have identified application security and scalability as a need and proposed

ways to achieve this. All these isolated and independent techniques have been combined together in a thread to form a methodology for Security-aware Software Development Life Cycle (SaSDLC) [12] and ID Management among the Clouds [16]. Also, open-source tool Suraksha [13] and PPID Protocols is available for application security and scalability through Security Designers' Workbench and Security Testers' Workbench. SaSDLC and this tool allow a software engineer to embed security right from the beginning. For deployment in the cloud, application security must include challenges related to Cloud as well. To the best of our understanding, there is no methodology or tool available to analyze the security, scalability and mobility requirements for an application that will be deployed in a Cloud.

This paper is organized as follows. Section II, introducing basic concepts of cloud computing. Section III, describes the challenges in cloud. Section IV, describes the Achieve QoS for cloud computing. Section V arise a problem. Section VI, define OCSA algorithm. Section VII, presents the approach with an example. Section VIII, concludes the paper.

II. CLOUD COMPUTING

Cloud computing is a paradigm of computing with virtualized resources as a service to offer Dynamic scalability. Sometime some of these services could even be over Web Services accessed in Service Oriented Architecture (SOA). Users need not have knowledge of, expertise in, or control over the technology infrastructure in the Cloud that supports them. The concept incorporates infrastructure as a service (IaaS), platform as a service (PaaS) and Software as a Service (SaaS). Cloud computing services usually provide business applications online that are accessed from a web browser, while the software and data are stored on the utility servers – these servers and services may be in Clusters, or in Grids. These grids could well be Processor Grids, Data Grids, or a combination of both. The Cloud stack is shown in Figure 1 [16].

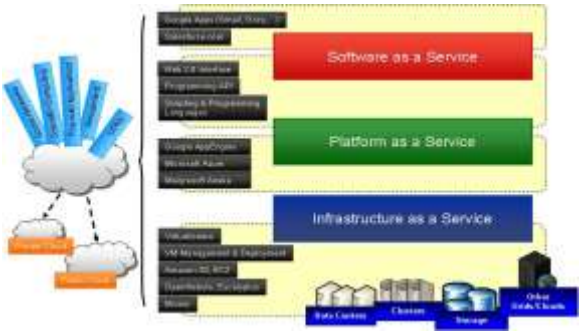

Figure 1: CLOUD COMPUTING REFERENCE MODEL ARCHITECTURE

Cloud describes the use of a collection of distributed services, applications, information and infrastructure comprised of pools of distributed compute, memory bank, network, and storage resources. These components can be rapidly orchestrated, provisioned, commissioned, and decommissioned using an on-demand utility-like model of allocation and consumption [15]. Cloud services are most likely to use virtualization technologies to provide dynamic integration, provisioning, management, mobility and scalability. The client device need not be a fixed desktop – it may even be a mobile device. The mobile device will be able to access the application in the Cloud even at a vehicular speed with seamless roaming.

Benefits from Cloud Computing are

- Quick, Cheap and Easy to Deploy without any fixed investment
- The infrastructure and platform is supplied on demand and guaranteed to scale up or down, user pays on-usage basis like any other utility services.
- Pay only for what is used or consumed – move from capex to opex.
- Reduce internal IT cost by reduction in maintenance and support cost of the Datacenter.
- Latest up to date software and the software is used on rent as pay-per-use.
- Data sharing is simpler.
- Group communication easier.
- Single signon possible for group of applications (like Google.app Engine).

To market this advantages to customers major IT vendor striving each other. Here, listed some of IT firms there providing services IaaS cloud (such as Amazon's EC2) [http://aws.amazon.com/ec2/], Rackspace [www.rackspacecloud.com], and Nimbus [http://workspace.globus.org]) run hardware virtual machine images (for example, a Xen VM image). PaaS cloud (such as google app engine[htpp://code.google.com/appengine/], Microsoft Azure [www.microsoft.com/wisdowsazure/], and Heroku [http://heroku.com]) run application VM programs (for instance JVM). Finally, SaaS cloud (such as salesforce [www.salesforce.com/platform/] and NetSuit [www.netsuit.com/portal/platform/main.stml]) runs vendor supplied services.

III. CHALLENGES IN THE CLOUD

Along with advantages, there are some challenges in Cloud as well – security, scalability and mobility are the most important ones. Many of these challenges are common to any Web application. However, in the cloud unlike a Web environment, due to virtualization, one runtime instance may run on different processor at different point in time. This makes it mandatory that an application security and mobility are additionally aware of interface security. Challenges in public Cloud are higher in magnitude compared to a private network or datacenter or even a private cloud. This is because the major areas of vulnerabilities in Cloud are,

1. Untrusted public network (e.g., internet) makes intruders in to system.
2. Session less HTTP protocol produce DoS attack.
3. Extensive usage of virtualization leads to storage consequences.
4. Often uses SOA architecture increases internal attackers.
5. Often uses Web Services.
6. Service providing problem from scalability.

Cloud carries higher security risks because the application is deployed on the Web and accessible by a hacker with many free hacking tools available. Also, Web applications have many security vulnerabilities starting from session hijacking to SQL injection. Web applications also increase the security risk because there are too many entry and exit points in the application unlike a mainframe or client-server application. Adding virtualization, mobility and scalability of software, increases the security risk even further. Bugs in the virtualized or mobilized middleware can give a local attacker root or local system access to the entire system. Thus, we have to give up the protection we once had in isolated systems by going virtual. An exploit in one virtualized server can provide unfettered access to all servers, as they are hosted on the same hardware. Deployment using SOA or Web services also increases the security risk as the attack surface in such cases grow substantially.

IV. ACHIEVE QoS FOR CLOUD

For any application that is Cloud-ready, security, scalability and mobility requirements must be inbuilt within the application. The design of the application also must be such that the attack surface for this application should be as minimum as possible. In computers, only that part of the program will be a target of attack that is accessible to a hacker. A piece of code or part of a program is exposed to the public as an API (application programming interface) that can be the target of attack. The attack surface [2] of an application is the union of code, interfaces, services, protocols, and practices exposed to a user (or attacker alike). In security design, therefore, the attempt is always to analyze the attack surface and reduce it. Reduction of attack surface is always advisable for any application. For a conventional application where the deployment scenario is static, attack surface is generally reduced through perimeter security; however, for cloud-ready application attack surface reduction will be part of functional requirement. If the attack surface is reduced, the security is automatically increased by reducing the risk of attack. Attack surface reduction focuses on reducing the area of the code accessible to unauthorized users. This is achieved by understanding the system's entry and exit points and the trust level required to access them through authentication and authorization checks and through escalation of privileges.

To reduce the attack surface, one needs to get answers to the following questions:

- Question 1. Is this feature really necessary? Who are the users that need this feature? If this feature is not necessary to a majority of users, it should be hidden by default.
- Question 2. Is it necessary to offer this feature from remote location? If yes, determine from where and what type of access mechanism this feature will be provided. Also, determine the type of networks the feature will be available from.
- Question 3. Who are the users that need to access this feature? Determine the legitimate users and a mechanism to validate them so that unauthorized users cannot access this feature.
- Question 4. What type of privilege does this feature need to provide the service? If it needs escalated privilege, determine how long it needs the escalated privilege for.
- Question 5. What are the interfaces this feature has with other services, interfaces, and protocols? If this feature crashes, what impact it will have on other services or the system as a whole.
- Question 6. What are present problems triggered in system? Related to security, scalability, mobility and virtualization.

Attack surface analysis helps understand the areas that can be target of attack and through threat modeling analyze possible threats. Combining these two will guide the action plan to build a secured Cloud-ready application.

V. PROBLEM STATEMENT

Researchers have given various kind of approaches for QoS on Cloud applications, but all of them are proposed outside of the SDLC, which requires additional time and cost. Here we are proposing a new approach that is imbibed within the SDLC Requirement process.

The main focus of this paper is on the holistic approach to identify and reduce attack surface for cloud application. An Algorithm called OCSA also introduced to support this analysis. This paper also explains how misuse case and attack trees are use to identify attack surface on Cloud applications and how to reduce it.

VI. THE OCSA ALGORITHM WITH ATTACK SURFACE ANALYSIS

The OCSA algorithm follows below step to figure out list of threats, problem to fulfill security, scalability, mobility, and virtualization requirements.

STEP 1 – Identification of Functional Requirements
STEP 2 – Identifying of Nonfunctional Requirement.
STEP 2.1 – Identify attack surface.
STEP 2.2 – Preventing DoS and Traffic congestion.
Step 2.3 – Storage and service providing consequences
Step 3 – Perform DREAD Analysis.
Step 4 – Mitigation Strategies
Step 5 – Convert Nonfunctional requirements to Function requirements.
Step 7 – Iterate

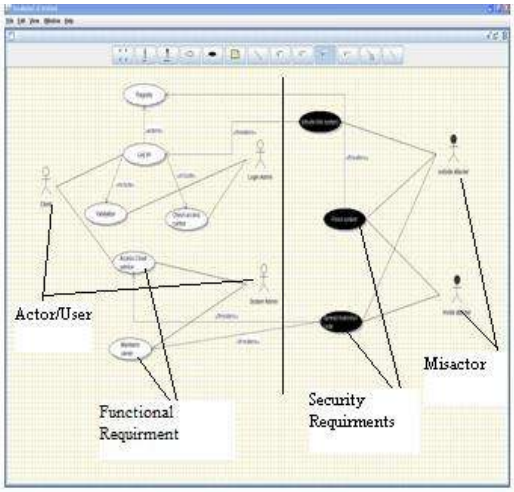

Figure 2: LOGIN FUNCTION SECURITY REQUIREMENTS (SURAKSHA)

For Cloud-ready systems, it is necessary that the attack surface is minimized. For this, we used Suraksha tool[12,13]. Suraksha uses UML to elicit requirement. We take the use case diagram of the suraksha to derive the deploy functional requirement scenario of login application as showing figure 2. and, enhance it to evaluate trust boundaries. The trust boundary is then examined to evaluate pathways of interactions between user and system function.

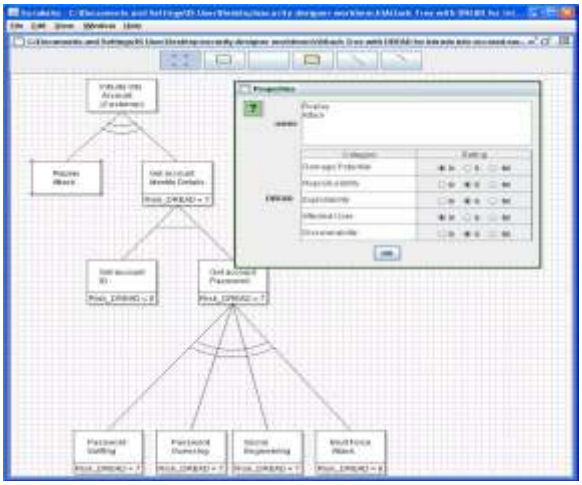

Figure 3 ATTACK TREE WITH DREAD RATING FOR INTRUDE INTO COSTOMER ACCOUNT LOGIN FUNCTION (SURAKSHA)

From misuse case diagram we calculate the number of entry points and exit points. Higher weight age is given to entry point compared to exit points. Higher weight age is used for sensitive out-data. Additional weight age is given for these entry-points where some of the fields can be used as parameters in SQL query and can potentially be vulnerable for SQL injection. For each entry-point and exit-point, a security requirement diagram (Figure 2) is used to assess the possibility of a security attack. Through STRIDE, the possibilities for Spoofing Identity, Tampering with Data, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege are considered; and, through CI5A, Confidentiality, Integrity, Availability, Authentication, Authorization, Accounting, and Anonymity in the given case by misactors are explored. This yields a list of possible abstract threats. "A use case generally describes behavior that the system/entity owner wants it to provide. A misuse case is a special kind of use case, describing behavior that the system/entity owner does not want to occur." A misactor is the inverse of an actor, who does not want the system to function in an expected fashion. A misactor can also be defined as a special kind of actor (hacker) who initiates misuse case [1].

Using the security requirements diagram as shown in Figure 2, Suraksha helps analyze the entry and exit points and determine whether there is a possible attack scenario. If yes, refine it during the design and go for information hiding as much as possible. Information hiding will reduce the attack surface. Reduction of attack surface will reduce security risk and in turn increase the security of the application.

In the next phase of the analysis and design of the application security, **threat modeling** of Suraksha is used. In threat modeling, each abstract threat in the Misuse case diagram is Considered as a root node and corresponding attack tree (Figure 3) is constructed to understand what are the AND and OR relationship in the threat path. Here the user goes through each and every Misuse case; a node in the attack-tree is an actual threat. In an AND relationship each and every attack path needs to be successful for a successful attack whereas, in OR path anyone needs to be successful. Threat model is used to include proper security classes and methods in the application. Once the threat model is known, the attack path is also known. During design, all these attack paths need to be covered. In a non-Cloud environment it may be possible to cover an attack path through network security; but, in Cloud-ready application this must be covered through proper countermeasures within the application through proper security-awareness.

VII. CONCLUSION

This paper focuses mainly on QoS on cloud based applications. We proposed an algorithm which uses step wise analysis of security, scalability, mobility and virtualization requirements Identification through Attack surface analysis. We used an open source tool called "Suraksha" which will help a software designer to draw all UML diagrams in a simple way and help Threat modeling process using Misuse case and Attack tree. And also it helps to elicit the Security use cases using threat modeling and DREAD rating.

REFERENCE:

[1]. Asoke K Talukder and Manish Chaitanya, Architecting Secure Software Systems, CRC Press,2008.

[2]. Shawn Hernan, Scott Lambert, Tomasz Ostwald, Adam Shostack, "Uncover Security Design Flaws using The STRIDE Approach" msdn.microsoft.com, Nov. 2006. [Online]. Available:http://msdn.microsoft.com/en-us/magazine/cc163519.aspx. [Accessed: July 21, 2008].

[3]. F. Swiderski and W. Snyder, Threat Modeling. Washington: Microsoft Press, 2004.

[4]. Bruce Schneir, "Modeling Security Threats", December 1999. [Online]. Available: http://www.schneier.com/paper-attacktrees-ddj-ft.html [Accessed: Aug. 17, 2008].

[5]. Fredrik Moberg, "Security Analysis of an Information System using an Attack tree-based Methodology," Master's Thesis, Chalmers University of Technology, Goteborg, Sweden, 2000.

[6]. Mamadou H. Diallo, Jose Romero-Mariona, Susan Elliott Sim and Debra J. Richardson, "A Comparative Evaluation of Three Approaches to Specifying Security Requirements," presented at 12th Working Conference on Requirements Engineering: Foundation for Software Quality, Luxembourg, 2006.

[7]. Guttorm Sindre and Andreas L Opdahl, "Capturing Security Requirements by Misuse Cases," in Proc. 14th Norwegian Informatics Conference (NIK'2001),Troms, Norway, Nov. 2001.

[8]. G. Sindre and A.L. Opdahl, "Eliciting Security Requirements by Misuse Cases," in Proc. 37th Conf. Techniques of Object-Oriented Languages and Systems, TOOLS Pacific 2000, 2000, pp.

[9]. G. Sindre and A.L. Opdahl, "Eliciting security requirements with misuse cases," Requirements Engineering, Vol. 10, No. 1, pp. 34-44, Jan.2005.

[10]. J.D. Meier, et al., "Improving Web Application Security: Threats and Countermeasures," msdn.microsoft.com, June [Online].Available:www.msdn.microsoft.com/enus/library/aa302419.aspx [Accessed: July.29, 2008].

[11]. Guttorm Sindre and Andreas L. Opdahl, "Templates for Misuse Case Description," in Proceedings of the 7 th International Workshop on Requirements Engineering, Foundation for Software Quality (REFSQ'2001), Interlaken, Switzerland, June 2001.

[12]. Asoke K Talukder, Vineet Kumar Maurya, Santhosh Babu G, Jangam Ebenezer, Muni Sekhar V, Jevitha K P, Saurabh Samanta, Alwyn Roshan Pais, "Security-aware Software Development Life Cycle (SaSDLC) – Processes and Tools", accepted for presentation at IEEE WOCN 2009, April 28-30, Cairo, Egypt.

[13]. Suraksha (http://isea.nitk.ac.in/suraksha/)

[14]. Asoke K Talukder, Vineet Kumar Maurya, Santhosh Babu G, Jangam Ebenezer, Muni Sekhar V, Alwyn Roshan Pais , and PrahaladH. A Managing Functional & Nonfunctional Requirements for Collaborative Clouds.

[15]. Cloud computing Reference Model Architecture from "http://texdexter.files.wordpress.com/2009/09/figure-1-cloud-computing-architecture.jpg&imgrefurl=http://texdexter.wordpress.com" .