Volume 3, Issue 6, June 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Reducing the Impact of Code Injection Vulnerabilities for Cloud Offerings

R. Kumar
Department of Computer Science
Jamia Millia Islamia (Central University),
New Delhi-110 025, India

Abstract: Cloud computing can be referred as a technology used to deliver software, application platform or infrastructure services through Internet in an on-demand fashion. As confidential data stored or managed by cloud, possible compromise of data and security is a serious concern. Various security concerns need to be focused to ensure a safe cloud for secure storage of information, safe use of data without any fear of compromises. Mitigation techniques based on vulnerabilities analysis provides a de facto way to develop a secure software or application. The paper suggests the same approach to develop a secure cloud on identified code injection vulnerabilities, which can be prevalent in state-of-the art Cloud Offerings by providing one of the efficient mechanism namely, security requirements.

Keywords: Cloud Computing, Security Requirements, Web Application Security, Security in Requirements Phase, Security Vulnerabilities.

I. Introduction

Cloud computing is a buzz now-a-days. It has the potential to change the way to view and use of information technology. A cloud computing can be defined as an IT technology used to deliver software, application platform or infrastructure services through Internet in an on-demand fashion. In a cloud computing environment, a service provider can play two important roles: one is the *infrastructure providers* who manage cloud platforms and lease resources according to a usage-based pricing model, and other one *service providers*, who rent resources from one or many infrastructure providers to serve the end users [1]. The new paradigm of cloud computing resulted that the many large companies such as Google, Amazon, IBM and Microsoft making every effort to provide more powerful, reliable and cost-efficient cloud platforms. Cloud computing and sensitive data is major concern among enterprises for moving their applications and data to the public cloud. Even though cloud computing can provide many significant benefits, like fast deployment, easy scalability, potential cost savings and economies of scale. For many enterprises, the essential questions about security, data privacy and compliance needs to be answered. Enterprises are resisting to migrate their applications containing sensitive data, core processes and valuable assets to publicly accessible cloud due to significant concerns with security and compliance, leaving huge untapped potential for adopting this technology. According to the Goldman Sachs Equity Research Report of 2011, 70 percent of the CIOs surveyed express major concerns about data security in the cloud [2]. Loss of transparency and control over business data, location of data and protection strategies in a given cloud infrastructure were some major concerns for them.

As cloud shall have sensitive and confidential data, possible compromise of data and security is a serious concern. Business required better and faster threat protection with increased sophistication of attacks. Various security concerns need to be focused to ensure a safe cloud for secure storage of information, safe use of data without fear of compromise. The rest of the paper is organized as follows: Section II presents a brief discussion on State of the Art Cloud Offerings, whereas in Section III, Vulnerabilities Prevalent in State of the Art Cloud Offerings is discussed. In Section IV, security requirements for invulnerable data validation are proposed. Section V presents Conclusion and Future work.

II. State of The Art Cloud Offerings

Cloud computing can used to provide three types of services: One, Software-as-a-Services (SaaS), which comprises of end-user applications delivered as a service rather than traditional on-premise software. The second service known as Platform-as-a-Service (PaaS) provides an independent platform as a service on which developers can build and deploy customer applications. Third one, Infrastructure-as-a-Service (IaaS) primarily comprises the hardware and technology for computing power, storage, operating systems or other infrastructure delivered as an on-demand service rather than a dedicated on-site resource. These services can be used to develop many types of cloud-enabled business processes. Cloud-enabled applications and business processes also require a new approach to ensuring protection and compliance. Traditional security that focuses on denying access to data is no longer enough. Cloud security has to combine controlled access with

monitoring and protection of the application as well as the infrastructure, in development as well as in production. Threat and Risk Management is needed for dynamic, application-level monitoring and vulnerability protection. Security monitoring capabilities from the infrastructure level to the network level to the application level, and together operations metrics and security metrics to expose risk at the business-process level may be consolidated. Although cloud offerings have huge benefits for enterprises but they also have some pitfalls in terms of security such as difficulty in embedding security requirements and architecture during development and difficulty in securing composite applications across hybrid environments [3].

III. Vulnerabilities Prevalent In State Of The Art Cloud Offerings

Although cloud computing is a relatively young topic, there already are myriads of cloud offerings on the market. Some of the vulnerabilities can be prevalent in state-of-the-art cloud offerings on empirical indication. Injection vulnerabilities are exploited by manipulating input to a service/application such that parts of the input are interpreted and executed as code against the intentions of the programmer. Security assessments of web components of current cloud offerings show the prevalence of injection vulnerabilities in state-of-the-art offerings. Examples of injection vulnerabilities are:

- ***SQL injection*:** The input contains SQL code that is erroneously executed in the database backend. Databases are very important support to cloud database applications. SQL (Structured Query Language) is a specialized programming language used to communicate with the database. Applications often use user-supplied data to create SQL statements. Reprehensible creation of SQL statements by an application can lead to SQL Injection attack, which in turn make achievable for an attacker to modify the statement structure and execute unintended and potentially hostile commands [4]. When such commands are executed, they do so under the context of the user specified by the application executing the statement. This capability allows attackers to gain control of all database resources accessible by that user, up to and including the ability to execute or modify commands or queries to steal, corrupt, or otherwise change underlying data on the hosting system [5, 6, 7].
  Example: A web based authentication form might build a SQL command string using the following method:
  SQLCommand = ,SELECT Username FROM Users WHERE Username = ‘‘
  SQLCommand = SQLComand & strUsername
  SQLCommand = SQLComand & ,’ AND Password = ‘‘
  SQLCommand = SQLComand & strPassword
  SQLCommand = SQLComand & ,’‘
  strAuthCheck = GetQueryResult(SQLQuery)
  In this code, the developer combines the input from the user, strUserName and strPassword, with the logic of the SQL query. Suppose an attacker submits a login and password that looks like the following:
  Username: foo
  Password: bar’ OR ‘’=’
  The SQL command string built from this input would be as follows:
  SELECT Username FROM Users WHERE Username = ‘foo’
  AND Password = ‘bar’ OR ‘’=’’
  This query will return all rows from the user’s database, regardless of whether “foo” is a real user name or “bar” is a legitimate password. This is due to the OR statement appended to the WHERE clause. The comparison ‘’=’’ will always return a “true” result, making the overall WHERE clause evaluate to true for all rows in the table. If this is used for authentication purposes, the attacker will often be logged in as the first or last user in the Users table.

- ***Command injection*:** The input contains OS commands that are erroneously executed via the operating system. OS Commanding is an attack technique used for unauthorized execution of operating system commands [8]. A cloud application is often the bridge between an outsider on the network and the internals of host operating system. When an application accepts untrusted input to build operating system commands in an insecure manner involving improper data sanitization, and/or improper calling of external programs can lead to this attack. In OS Commanding, executed commands by an attacker will run with the same privileges of the component that executed the command, (e.g. database server, web application server, web server, wrapper, application). Since the commands are executed under the privileges of the executing component an attacker can leverage this to gain access or damage parts that are otherwise unreachable (e.g. the operating system directories and files).
  Example: The example below reads the name of a shell script to execute from the system properties. It is subject to the second variant of OS command injection, as follows;
  Java Example: *Bad Code*
  String script = System.getProperty("SCRIPTNAME");
  if (script != null)
  System.exec(script);

If an attacker has control over this property, then he or she could modify the property to point to a dangerous program**.**

- ***Cross-site scripting*:** The input contains JavaScript code that is erroneously executed by a victim's browser. Cross-site scripting is an attack technique in which an attacker echoes its malicious code into a user's browser instance [9]. It occurs when a cloud application sends a page containing user supplied data to the browser without validation, filtering, or escaping [10]. Web technology like ASP, JSP, PHP, CGI, Perl, ASP.NET, VB.NET and C# may suffer with this vulnerability. Non-persistent or reflected, persistent or stored, and DOM based XSS are three variants of this attack. This attack can lead to hijacking the user account, delivery of fraudulent content into user's web browser etc.
  Example: The application uses untrusted data in the construction of the following HTML snippet without validation or escaping:
  (String) page += "<input name='creditcard' type='TEXT'value='" + request.getParameter("CC") + "'>";
  The attacker modifies the 'CC' parameter in their browser to:
  '><script>document.location='http://www.attacker.com/cgi-bin/cookie.cgi?foo='+document.cookie</script>'.
  This causes the victim's session ID to be sent to the attacker's website, allowing the attacker to hijack the user's current session.

IV. Invulnerable Data Validation

Having established a robust, secure and reliable system to identify, track and control access to resources for a user, security of the application, connected applications and systems and their underlying infrastructure must be considered. Many serious attacks on applications and their users rely on poor data validation on input from the client side and on data sent back to those clients. Such attacks include SQL injection, Cross Site Scripting (and variants), and more traditional attacks such as buffer overflows and format strings that have affected fat-client applications for some time. It is therefore essential that client input and application output is validated and sanitized, to ensure dangerous characters and inappropriate content are removed or encoded safely. At its tamest, poor data validation and sanitization (on input and output) leads to application errors that degrade the user experience. At their worst, data validation flaws allow compromise (or destruction) of the data within the application, subversion of authentication and access control, social engineering attacks on users and compromise of session tokens. Since the characteristics of the aforementioned Code Injections attacks are similar, various methods to avoid Code Injection vulnerabilities especially SQL Injections have been stated in literature [11, 12, 13, 14, 15, 16, 17, 18]. Realizing the importance of data validation, following security requirements are hereby suggested:

- ✓ *Avoid using string concatenation queries for accessing database.*
- ✓ *Use parameterized/stored procedures to access the database.*
- ✓ *Use a low privileged account to run the databases.*
- ✓ *Validate all inputs both at client and server side.*
- ✓ *Validate the data received by the application from all the potential sources of input to protect against a range of potential attacks.*
- ✓ *Validate the data on the application server before processing.*
- ✓ *Do the validation against a positive specification of what is allowed.*
- ✓ *Validate text input and URL queries string from improper characters.*
- ✓ *Limit the amount of characters for a web input fields and URL query string for a proper amount.*
- ✓ *Ensure that all valid data is accepted, while potential dangerous data is rejected or sanitized.*
- ✓ *Delete system stored procedures.*
- ✓ *Do not display the errors to user that contain the information about the database or about the actual source code.*

V. Conclusion And Future Work

Cloud computing has the potential to change the way to view and use of information technology. As a cloud managed confidential data from various sources can be a target for attackers. It has become necessary, to develop secure cloud since of the overwhelming increase of security vulnerabilities. This paper proposes security requirements for validation of data based on the security vulnerability analysis related to the same. New vulnerabilities and their corresponding security requirements shall be identified as it appears to be an evolving process. Therefore, extension/modification and proposal of new security requirements may also be done. This work may provide guidance to industry as well as academia for implementing more secure data validation process for cloud computing applications.

References

[1] Qi Zhang · Lu Cheng · Raouf Boutaba, "Cloud computing: state-of-the-art and research challenges", *Journal of Internet Services and Applications*, Vol. 1, No. 1. (May 2010), pp. 7-18. doi:10.1007/s13174-010-0007-6
[2] Goldman Sachs Equity Research, January 2011.

[3] Key capabilities for mastering the cloud, HP white paper accessed from http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA3-4693ENW.pdf

[4] Merlo, E., Letarte, D., Antoniol, G. (2007, 21-23 March). Automated Protection of PHP Applications Against SQL-injection Attacks, *IEEE CSMR 07; 11th European Conference on*. (pp. 191-202).

[5] Antunes, N., Laranjeiro, N., Vieira, M., Madeira, H. (2009, 21-25 September). Effective Detection of SQL/XPath Injection Vulnerabilities in Web Services. In proceedings of *IEEE International Conference on Services Computing SCC '09*, (pp. 260–267).

[6] Alonso, J.M., Bordon, R., Beltran, M., Guzman, A. (2008). Informatica, Mostoles; LDAP injection techniques. In proceedings of *11th IEEE Singapore International Conference on Communication Systems. ICCS 2008*, (pp. 980-986).Guangzhou.

[7] Merlo, E., Letarte, D. & Antoniol, G. (2006, October). Insider and Outsider Threat-Sensitive SQL Injection Vulnerability Analysis in PHP. IEEE WCRE 2006 13th Working Conference on. (pp. 147-156).

[8] OS Commanding. Retrieved March 21, 2009, from http://projects.webappsec.org/OS-Commanding.

[9] Cross-Site Scripting. Retrieved March 21, 2009, from http://projects.webappsec.org/Cross-Site-Scripting

[10] Zalewski, M. (2006, January). Cross Site Cooking. Whitepaper, Retrieved April 11, 2007, from http://www.securiteam.com/securityreviews/5EP0L2KHFG.html

[11] Muthuprasanna, M., Wei, K. & Kothari, S. (2006). Eliminating SQL Injection Attacks-A Transparent Defense Mechanism, wse, (pp. 22-32). *Eighth IEEE International Symposium on Web Site Evolution (WSE'06).*

[12] Bertino, E., Kamra, A., and Early, J.P. (2007). Profiling Database Application to Detect SQL Injection Attacks. In Proceedings of *IPCCC*, (pp. 449-458).

[13] Thomas, S. and Williams, L. (2007, May 20-26). Using Automated fix Generation to Secure SQL Statement. *Third International Workshop on Software Engineering for Secure Systems (SESS'07).* ICSE Workshops 2007. p. 9.

[14] Wei, K., Muthuprasanna, M. and Suraj. (2006, April 20-21). Preventing SQL Injection attacks in stored procedures. IEEE ASWEC20006. p 8

[15] Halfond, W. and Orso, A. (2005, November). AMNESIA: Analysis and Monitoring for Neutralizing SQL-Injection Attacks. In Proceedings of the *20th IEEE / ACM International conference on Automated software engine.* (pp. 174-183).

[16] Jourdan, V., G. (2008, December). Data validation, data neutralization, data footprint: A framework against injection attacks. The Open Software Engineering Journal, (2) , 45–54.

[17] Aljawarneh, S., Faisal Alkhateeb, F., and Maghayreh, A., E. (2010, April). A Semantic Data Validation Service for Web Applications. Journal of Theoretical and Applied Electronic Commerce Research. 5(1). 39-55.

[18] Xiang Fu Xin Lu Boris Peltsverger Shijun Chen Kai Qian Lixin Tao. (2007, July 24-27). A Static Analysis Framework For Detecting SQL Injection Vulnerability. 31st Annual International Computer Software and Applications Conference(COMPSAC 2007), Page(s):87 – 96

