RESUMEN ANALÍTICO EN EDUCACIÓN - RAE -	UNIVERSIDAD CATÓLICA de Colombia Vigilada Mineducación RIUCaC

FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA DE SISTEMAS
BOGOTÁ D.C.

AÑO DE ELABORACIÓN: 2016

TÍTULO: Seguridad de JAX-RS frente a ataques por inyección de código.

AUTOR (ES):

Fuyo, Juan Carlos y Rivera, Jael Alexander.

DIRECTOR(ES)/ASESOR(ES):

Velandia, John y Lopez, Alexandra.

MODALIDAD: Trabajo de investigación.

PÁGINAS: 105 **TABLAS:** 18 **CUADROS:** 0 **FIGURAS:** 9 **ANEXOS:** 4

CONTENIDO:

RESUMEN ANALÍTICO EN EDUCACIÓN - RAE -	 UNIVERSIDAD CATÓLICA de Colombia Vigilada Mineducación RIUCaC

DESCRIPCIÓN:

Se implementan tres algoritmos de ataques de inyección de código. Teniendo en cuenta estos algoritmos se definen los requerimientos para diseñar y desarrollar el prototipo, se describe la arquitectura de la aplicación, además del escenario de pruebas donde se encuentran los servicios Web a ser atacados. Por último se indican los niveles de vulnerabilidad encontrados en cada una de las implementaciones seleccionadas y se finaliza con un análisis de los resultados y algunas conclusiones de la investigación.

METODOLOGÍA:

En un principio es un estudio de tipo exploratorio, que tiene como objetivo la formulación del problema, que en este caso es encontrar vulnerabilidades iniciales en el sistema, para posibilitar una investigación más precisa y el desarrollo de un proceso de ataque que logre vulnerarlo. En la segunda fase se convierte en un estudio de tipo descriptivo, mediante en el cual se presentan y describen las vulnerabilidades identificadas, de acuerdo a los indicadores definidos para tal fin.

PALABRAS CLAVE:

Ejemplo: API, REST, CODE-INJECTION, JAX-RS, SEGURIDAD, HTTP.

CONCLUSIONES:

Actualmente no se cuenta con un software que realice análisis de vulnerabilidades a Rest Services implementados con JAX-RS, la aproximación a este tipo de soluciones son herramientas que realizan pruebas de penetración de seguridad a plataformas Web.

Con el fin de realizar un aporte a las evaluaciones realizadas a REST Service implementadas con JAX-RS, se desarrolló un prototipo que implementa inyección de código a las peticiones que consumen estos REST Services, esto con el fin de identificar el nivel de vulnerabilidad de API.

Durante esta investigación se determinó que los principales inconvenientes de vulnerabilidad presentados en aplicaciones JAX-RS, son causados por la forma en la cual es codificado su código fuente, esto se debe al desconocimiento de la API JAX-RS y del estilo de arquitectura REST.

Otro factor que está ligado a la mala codificación del servicio, es la falta de controles específicos como son los filtros, validaciones y las malas prácticas de programación, las cuales logran revelar y exponer los detalles del servicio, estos son indicios que informan al atacante las vulnerabilidades presentadas en los REST Services.

Debido a que REST es un estilo de arquitectura aplicado a sistemas distribuidos, podemos determinar al servidor y al cliente como actores de cada petición/respuesta, este último tiene total control sobre la representación del recurso, quien también puede inyectar algún método que altere el funcionamiento del REST Service.

Estas inyecciones de código son posibles debido a la forma en la cual se deben de extraer los datos, es decir a través de mensajes HTTP. Las inyecciones son realizadas a través del cliente, ya que como mencionamos anteriormente, él es quien tiene control total sobre recurso, desde la petición hasta la respuesta.
Los textos de representación XML y JSON son otra entrada de vulnerabilidad, ya que por defecto los analizadores comunes XML y JSON de Java, contienen una configuración insegura que no contempla protección sobre estas representaciones, permitiendo al atacante inyectar valores que alteren la representación y flujo de los recursos.

Por otra parte se pudo determinar que la implementación Restlet genera un buen nivel de seguridad. Esto se debe gracias a la forma propia en la cual se codifica, captura y procesan los datos de las peticiones, logrando que no sea tan factible la inyección de parámetros HTTP.

FUENTES:

[1] OWASP, “OWASP,” OWASP.ORG, 2015. .
[2] E. J and C. Linksource, “SOA, Web Services, And RESTful Systems.,” World, 2007.
[3] E. J. Bruno, “SOA, Web Services, and RESTful Systems,” Dr. Dobb’s J. - ProQuest Sci. Journals, vol. 1, p. 5, 2007.
[4] C. Davis, “What if the Web Were not RESTful?,” EMC Corp., vol. 1, p. 8, 2011.
[5] R. T. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” Building, vol. 54, p. 162, 2000.

[6] M. Little, “A Comparison of JAX-RS Implementations,” infoq/news, 2008. .
[7] Q. U. É. Son, L. A. S. Vulnerabilidades, and D. E. L. Software, “¿Qué son las vulnerabilidades del software?,” pp. 1–11, 2014.
[8] Edward Hunt, “US Government Computer Penetration Programs and the gImplications for Cyberwar,” IEEE Ann. Hist. Comput., vol. 34, no. undefined, pp. 4–21, 2012.
[9] OWASP, “OWASP Zed Attack Proxy Project,” OWASP Zed Attack Proxy Proj., 2016.
[10] A. B. L. y Andrea Alarc�n Aldana y Mauro Callejas Cuervo, “Vulnerabilidad de Ambientes Virtuales de Aprendizaje utilizando SQLMap, RIPS, W3AF y Nessus [Vulnerability in Virtual Learning Environments using SQLMap, RIPS, W3AF and Nessus],” Vent. Inform�tica, vol. 0, no. 30, 2014.
[11] W. G. J. Halfond and A. Orso, “AMNESIA: Analysis and Monitoring for NEutralizing SQL-injection Attacks,” in Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, 2005, pp. 174–183.
[12] S. W. Boyd and A. D. Keromytis, “SQLrand: Preventing SQL Injection Attacks,” in Applied Cryptography and Network Security: Second International Conference, ACNS 2004, Yellow Mountain, China, June 8-11, 2004. Proceedings, M. Jakobsson, M. Yung, and J. Zhou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 292–302.
[13] M. SQL, “SQL Injection,” 2011.
[14] O. and/or its Affiliates, “Building RESTful Web Services with JAX-RS,” Oracle and/or its affiliates, 2013. .
[15] J. Evdemon, “understanding services,” in SOA in the real World, 1st ed., 2011, p. 195.
[16] H. Li, “RESTful Web Service Frameworks in Java,” Informatiz. Off. Shanghai Lixin Univ. Commer. Shanghai, 201620, China, vol. 1, p. 4, 2016.
[17] N. Balani and R. Hathi, Apache CXF Web Service Development. 2009.
[18] R. W. Services, “RESTEasy JAX-RS.”
[19] I. Rest, “7 implementaciones,” pp. 130–137.
[20] A. Wink, “Apache Wink 1.1,” pp. 3–121, 2010.
[21] M. Alfonso, Cifrado de las comunicaciones digitales, de la cifra clasica al algoritmo RSA. 2006.
[22] I. K. Center, “Protección de recursos JAX-RS,” IBM Documentation, 2015. .
[23] C. Pautasso and E. Wilde, “RESTful Web Services: Principles, Patterns, Emerging Technologies,” Raleigh • NC • USA, vol. 1, p. 2, 2010.
[24] U. Yael, “Entorno para Experimentación de Vulnerabilidades en la Ensenanza de Buenas Practicas de Programación.”
[25] M. Fuksa and M. Gajdo, “Securing JAX-RS RESTful services.”

[26] Cwe, “CWE - 2011 cwe/sans top 25 most dangerous software errors,” SANS Inst., p. 41, 2011.
[27] A. Klein, “Blind XPath Injection,” pp. 1–10, 2004.
[28] “Live Experiments depicting SQL Injection Attacks,” pp. 91–93.
[29] J. M. Alonso, R. Bordon, M. Beltran, and A. Guzman, “LDAP injection techniques,” in Communication Systems, 2008. ICCS 2008. 11th IEEE Singapore International Conference on, 2008, pp. 980–986.
[30] Z. Su and G. Wassermann, “The Essence of Command Injection Attacks in Web Applications,” in Conference Record of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 2006, pp. 372–382.
[31] B. Sullivan, “Server-side JavaScript injection,” Black Hat USA, 2011.
[32] K.-J. Lin, “SMTP-1: The First Functionalized Metalloporphyrin Molecular Sieves with Large Channels,” Angew. Chemie Int. Ed., vol. 38, no. 18, pp. 2730–2732, 1999.
[33] B. M. Bowen, V. P. Kemerlis, P. Prabhu, A. D. Keromytis, and S. J. Stolfo, “Automating the injection of believable decoys to detect snooping,” in Proceedings of the third ACM conference on Wireless network security, 2010, pp. 81–86.
[34] D. Html and C. S. S. Javascript, “Índice.”
[35] E. Janot and P. Zavarsky, “Preventing SQL Injections in Online Applications : Study , Recommendations and Java Solution Prototype Based on the SQL DOM,” 2008.
[36] C. Commons, “OWASP Top 10 2013 Los Diez Riesgos Más Críticos en Aplicaciones Web,” p. 22, 2013.
[37] M. Montenegro, “Interfaces gráficas con Swing Introducción,” vol. 467.
[38] E. Moreno, “Web Page Development Web Page.”
[39] “Getting Started with AWS.”
[40] J. Yances and S. Murillo, “Software Requirements Specification,” pp. 1–31, 2008.
[41] F. De Ciencias and D. Administración, “Universidad del Azuay,” no. Vdi, 2015.
[42] F. Barnsteiner and M. Theis, “2011 [RESTful Webservices mit,” 2011.
[43] C. Servlets, “Controlling g Web Application Behavior,” 2009.
[44] “Generic Webshop System with JPA.”

LISTA DE ANEXOS:
Anexo1.ProcesoDeRecolecciónDeDocumentos.
Anexo2.PAIC_APLICATION
Anexo3.Manual_de_usuario
Anexo4.Manual_de_configuracion