Métodos, técnicas y estrategias para mejorar la seguridad de los sistemas de software.

Agustín Ferrari, Edgardo Bernardis, Mario Berón, Hernán Bernardis,
Maria Joao Tinoco Varanda Pereira, Daniel Riesco

Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950, San Luis, Argentina
Departamento de Informática e Comunicações
Instituto Politécnico de Bragança
Bragança, Portugal

{ebernardis, mberon, hbernardis, driesco}@unsl.edu.ar,
ferrariagustin93@gmail.com, mjoao@ipb.pt

RESUMEN

Con el crecimiento de internet y las distintas dinámicas de la sociedad actual; ha cambiado en gran medida la forma de interactuar e intercambiar información entre las personas y las empresas. Este intercambio se vuelve blanco de ataques por parte de todos aquellos actores que quieren obtener información útil y valiosa a sus propios intereses o de terceros. Ante este panorama se vuelve imperioso implementar todo tipo de medidas y acciones tendientes a evitar estos ataques, por tal motivo nace lo que se denomina Seguridad Informática. Toda acción, herramienta o metodología enfocada a evitar, contrarrestar o retrasar ataques contra activos sensibles juega un rol sumamente importante para los diversos actores.

Por lo antes descrito, en este artículo se presenta una línea de investigación que aborda el diseño y construcción de una herramienta cuyo principal objetivo es desarrollar e incrementar el nivel de seguridad de Servicios Web.

Palabras claves: Seguridad Informática, activos sensibles, Servicios Web.

CONTEXTO

La presente línea de investigación se enmarca en el proyecto (P-031516.) de investigación: "Ingeniería de Software: Conceptos, Prácticas y Herramientas para el Desarrollo de Software con Calidad" – Facultad de Ciencias Físico-Matemáticas y Naturales, Universidad Nacional de San Luis, y el proyecto (PO/16/93.) denominado: "Fortalecimiento de la Seguridad de los Sistemas de Software mediante el uso de Métodos, Técnicas y Herramientas de Ingeniería Reversa" – Realizado en conjunto con la Universidade do Minho Braga, Portugal.

1. INTRODUCCIÓN

Actualmente se están popularizando los Web Services como artefactos de software a partir de los cuales se pueden construir sistemas más complejos. Según la W3C, un Web Service es: "Una aplicación de software identificada por una URI, cuya interfaz y enlaces son capaces de ser definidos, descritos y descubiertos como artefactos XML. Un web service soporta interacción directa con otros agentes de software usando mensajes basados en XML intercambiados a través de protocolos basados en internet". Muchas organizaciones construyen sus sistemas basándose en una arquitectura orientada a servicios web, algunos de ellos se publican al resto del mundo de manera libre, mientras que otros son utilizados de manera interna por sus equipos de

desarrollo. Esto permite que cada equipo de desarrollo elija la arquitectura que desee para construir sus proyectos sin afectar la vinculación con el resto del sistema y/o proyectos. La interacción entre proyectos se convierte en un intercambio de mensajes con la información necesaria dentro, sin necesidad de vinculación a nivel de arquitectura subyacente más que la necesaria a la invocación de los servicios web.

Construir un Web Service y que pueda ser utilizado por cualquier otra persona u organización en el mundo ha sido posible debido a la creación de estándares y lenguajes formales para la definición de los mismos.

Todo Web Service posee una especificación que provee la información necesaria para invocarlo. Uno de los estándares de descripción más conocido es WSDL (Web Service Definition Language) [6]. Las especificaciones WSDL son un dialecto XML, con reglas bien definidas para especificar cada componente del WS. Cuántos parámetros recibe y de qué tipo son, qué datos retorna y de qué tipo, qué protocolo de internet usa para su comunicación, qué operaciones posee, son entre otras tantas, características del WS que se encuentran especificadas en su WSDL asociado.

Así como el archivo WSDL sirve para que un agente de software o persona pueda interpretarlo para usar el servicio web que describe, también puede dar información a personas no deseadas o incluso exponer vulnerabilidades. Más aún si se considera que existen herramientas que generan los WSDLs de manera automática para un servicio web, con lo cual el nivel de atención a la información que se publica no siempre se encuentra bajo un estricto control. Esto se vuelve más importante para aquellos casos en donde los servicios web pertenecen a bancos, tarjetas de créditos, servicios de compra/venta online, entre otros. Incluso también para los servicios web que no se publican, son privados y necesitan mayor control y seguridad como los que pertenecen a empresas privadas y redes militares.

Empresas competidoras pueden aprender el know-how y conseguir copiar el diseño para ofrecer servicios similares y competitivos. Pero no solo se trata de competencia, los ataques de seguridad como espionaje de información, suplantación de clientes, inyección de comandos y denegación de servicio también son posibles ya que los atacantes pueden aprender sobre los datos intercambiados y los patrones de invocación de los documentos WSDL. Si bien la legibilidad de las descripciones de los servicios hace que los servicios web sean reconocibles, también contribuye a la vulnerabilidad del servicio [7]. Todos contienen información formal (código fuente) e informal (identificadores, comentarios, documentación, etc.) y es en este tipo de información en donde los atacantes hacen foco para obtener información beneficiosa a sus propósitos. Suena lógico entonces incrementar la seguridad que posee un determinado WSDL para evitar e impedir los ataques.

La línea de investigación comprende los principales lineamientos de:

- *Seguridad Informática*: Se estudia el concepto de seguridad informática y sus principios generales.
- *Ofuscación*: Se estudia la ofuscación de código y sus diferentes clases.
- *Reflexión*: Se estudia el patrón de reflexión a fin de auto-incorporar algoritmos de seguridad en ejecución.

2. LÍNEAS DE INVESTIGACIÓN Y DESARROLLO

En las siguientes secciones se describen las líneas de investigación que llevan adelante los autores de este artículo.

2.1. Seguridad Informática

El objetivo de la Seguridad Informática es obtener un nivel aceptable de seguridad, entendiéndose por aceptable un nivel de protección suficiente para que la mayor parte de potenciales intrusos, interesados en los equipos con información de una organización o persona, fracasen en cualquier intento de ataque contra los mismos. Asimismo, se encarga de establecer los mecanismos para registrar cualquier evento fuera del comportamiento normal y tomar las medidas necesarias para re-

establecer las operaciones críticas a la normalidad [1].

Los principios generales de la seguridad de la información [2, 3] son:

- Integridad: implica que debe salvaguardarse la totalidad y la exactitud de la información que se gestiona. A su vez, puede incluir:
  - ❏ Autenticidad: autoría indiscutible. Definir que la información requerida es válida y utilizable en tiempo, forma y distribución.
  - ❏ No Repudio: la seguridad de que una parte no puede negar posteriormente los datos de origen; suministrando la prueba de integridad y el origen de los datos, y que puede ser verificado por un tercero. Evitar que cualquier entidad que envió o recibió información alegue, ante terceros, que no la envió o recibió.
- Confidencialidad: implica que debe protegerse la información de forma tal que sólo sea conocida y accedida por las personas autorizadas y se la resguarde del acceso de terceros.
- Disponibilidad: implica que debe protegerse la información de forma tal que se pueda disponer de ella para su gestión en el tiempo y la forma requerida por el usuario.

2.2. Ofuscación

En términos generales, se entiende por ofuscar un código fuente o un código intermedio, un proceso mediante el cual se transforma mediante la aplicación de algoritmos de reescritura, un código perfectamente legible y entendible por una persona en otro de funcionalidad equivalente en un ciento por ciento, pero, en términos ideales, totalmente ilegible e incomprensible para un lector.

En pocas palabras, algunas de las técnicas de ofuscación más comunes consisten en la inclusión de bucles irrelevantes, cálculos innecesarios, comprobaciones fuera de contexto, nombres de funciones y de variables que no tienen nada que ver con su cometido, funciones que no sirven para nada, interacciones inverosímiles entre variables y funciones, etc. Otras técnicas, sin embargo, son mucho más potentes, en el sentido de que requieren un conocimiento superior de las características del lenguaje, e incluso pueden estar diseñadas para burlar a herramientas de ingeniería inversa específicas [10].

La ofuscación de código aplica una o más transformaciones de código que hacen que el código sea más resistente al análisis y la manipulación, pero preservan su funcionalidad [4].

2.2.1. Transformaciones de Código.

Las transformaciones de código para ofuscar un programa se pueden dividir en cuatro clases principales [5], estas son:

- Transformaciones Léxicas o de Diseño: afectan la información en el código que es innecesaria para su ejecución y que reduce la información disponible para un lector humano. Ejemplo de esto es la codificación de nombres, la eliminación de comentarios, etc.
- Transformaciones de Flujo de Control: actúan modificando el flujo de control del programa. Por ejemplo inserción de predicados opacos[1], que agregan transferencias de control extra que nunca se tomarán en tiempo de ejecución.
- Transformaciones de Flujo de Datos: operan sobre las estructuras de datos usadas en el programa. Un ejemplo de una transformación de datos es desmantelar arreglos para aumentar la complejidad del código.
- Transformaciones Preventivas: intentan detener el funcionamiento correcto de los decompiladores o desofuscadores de código. Ejemplo de esto es insertar "bytes basura" entre otras instrucciones. Un algoritmo de desensamblaje interpreta estos bytes como el comienzo de una instrucción, y se arriesga a desmontar incorrectamente los bytes de la instrucción original.

[1] Informalmente, una variable V (o predicado P) es opaca si tiene alguna propiedad q que conoce a priori el ofuscador, pero que es difícil de deducir para el desofuscador [12].

2.3 Reflexión

Durante la ejecución de una aplicación el código de la misma permanece estático; las instrucciones de ejecución no cambian. Se dice que una aplicación contiene código automodificable cuando es capaz de modificar sus propias instrucciones de código, además de sus datos, durante su ejecución [11]. Al contener esta característica y hacer uso de ella de forma correcta, se pueden lograr grandes resultados como modificar funcionalidades o incorporar herramientas a un sistema en ejecución.

En el caso de esta investigación, se propone que el sistema permita al usuario incorporar algoritmos de ofuscación en ejecución y así poder aumentar la seguridad de los servicios web, es decir, incorporar un módulo donde el usuario pueda desarrollar sus propios algoritmos que luego serán utilizados para la protección de los Servicios Web. Para llevar a cabo esta funcionalidad se hace uso de la Reflexión.

La Reflexión es: *la capacidad integral de un programa para observar o cambiar su propio código, así como todos los aspectos de su lenguaje de programación (sintaxis, semántica o implementación), incluso en tiempo de ejecución. Se dice que un lenguaje de programación es reflexivo cuando proporciona a sus programas la capacidad de reflexión* [8]. La palabra integral es muy importante, la verdadera reflexión no impone límites a lo que un programa puede observar o modificar. En el caso de esta investigación, se hace uso de esta capacidad a fin de permitirle al usuario incorporar algoritmos personalizados que aumenten la seguridad de los servicios web.

3. RESULTADOS OBTENIDOS/ESPERADOS

El trabajo de investigación de esta línea permitió obtener los siguientes resultados:

- Se desarrolló ATENOS una herramienta que permite recorrer Servicios Web, extraer sus identificadores, protegerlos y generar un nuevo servicio con funcionalidad equivalente pero más seguro[2].
- Se logró mejorar significativamente la seguridad del WSDL al ofuscar los tags del Servicio Web dificultando su entendimiento y por consiguiente disminuyendo considerablemente el tipo de ataques que se pueden realizar sobre el mismo.
- Se desarrollaron e implementaron tres algoritmos para ofuscar, los tags de un WSDL, mediante la aplicación de diferentes técnicas de transformaciones léxicas sobre la información tomada como objetivo para incrementar su seguridad.
- Se logró crear una plataforma que permite agregar modificar y eliminar algoritmos propios de los usuarios; siendo la misma un rasgo distinguible del resto de las herramientas de similares características.
- Se permite utilizar también la encriptación de los archivos al brindar la opción de desencriptar el archivo.
- Se utilizó ATENOS con un ejemplo de aplicación de la vida real el caso de la Facturación Electrónica perteneciente a la AFIP [9].

4. FORMACIÓN DE RECURSOS HUMANOS

Los progresos obtenidos en esta línea de investigación sirven como base para el desarrollo de tesis de posgrado, ya sea de doctorado o maestrías en Ingeniería de Software y desarrollo de trabajos finales de las carreras Licenciatura en Ciencias de la Computación, Ingeniería en Informática e Ingeniería en Computación de la Universidad Nacional de San Luis, en el marco de los Proyectos de Investigación.

[2] Cuando se menciona la extracción de identificadores, se hace referencia al proceso de construcción del árbol de sintaxis abstracta (AST) y la aplicación de diferentes recorridos para recuperar/modificar la información.

5. BIBLIOGRAFÍA

[1] Alejandra Stolk. Técnicas de seguridad informática con software libre, 2013. Parque Tecnológico de Mérida. ESLARED.

[2] Salton, International Telecommunication Union. SERIES X: DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY. Telecommunication security. Overview of cybersecurity. 2008.

[3] Salton, Borghello, Cristian F. Seguridad Informática, sus implicancias e implementación. Tesis Licenciatura en Sistemas. Universidad Tecnológica Nacional. 2001.

[4] Code obfuscation techniques for software protection. Cappaert, Jan. Katholieke Universiteit Leuven. 2012.

[5] A taxonomy of obfuscating transformations. Collberg, Christian and Thomborson, Clark and Low, Douglas. 1997.

[6] WSDL Specification for W3C https://www.w3.org/TR/wsdl..

[7] Pananya Sripairojthikoon, Twittie Senivongse. "Concept-Based Readability Measurement and Adjustment for Web Services Descriptions". ICACT Transactions on Advanced Communications Technology (TACT) Vol. 3, Issue 1, January 2014.

[8] Malenfant, J., Jacques, M., & Demers, F. N. (1996, April). A tutorial on behavioral reflection and its implementation. In Proceedings of the Reflection (Vol. 96, pp. 1-20).

[9]AFIP.https://wswhomo.afip.gov.ar/wsfev1/service.asmx. 2018.

[10] DOLZ, Daniel; PARRA, Gerardo. Ofuscadores de código intermedio. En *VIII Workshop de Investigadores en Ciencias de la Computación*. 2006.

[11] XIANYA, Mi, et al. A survey of software protection methods based on self-modifying code. En *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*. IEEE, 2015. p. 589-593.

[12] Christian Collberg, Clark Thomborson, and Douglas Low. Manufacturing cheap, resilient, and stealthy opaque constructs. ACM, New York, NY, USA, 184-196. 1998.