PROTECCIÓN DE ACTIVOS DE SOFTWARE

Julieta Gatica[1], Mario Berón[1], Daniel Riesco[1], Maria Joao Pereira[2], Pedro Henriques[3], Paulo Novais[3]

[1]Universidad Nacional de San Luis, San Luis, Argentina
[2]Instituto Politécnico de Braganza, Braganza, Portugal
[3]Universidade do Minho, Braga, Portugal

jag81295@gmail.com, mberon@unsl.edu.ar, driesco@unsl.edu.ar, mjoao@ipb.pt, pjon@di.uminho.pt, pedrorangelhenriques@gmail.com

RESUMEN

Una organización posee mucha información importante, conocida usualmente como activo. Estos activos pueden estar de forma física, escrita o impresa en papeles; ó de forma digital, a través de medios electrónicos. Considerando este segundo punto, se puede decir que en el software se implementan varios procesos de negocio, se utilizan algoritmos para mejorar el rendimiento, se manejan estrategias para solucionar problemas críticos, entre otras cosas. Por estas razones, el software se considera una fuente de activos muy importante.

Debido a que el software está sujeto a ataques constantes de usuarios maliciosos, se debe tener una constante actualización en materia de seguridad informática y, por ello, es necesario protegerlo utilizando distintas técnicas de protección de software.

En este artículo se presenta una línea de investigación que tiene como principal objetivo la definición de técnicas de protección y el diseño e implementación de lenguajes específicos del dominio orientados a resguardar los activos del software.

Palabras clave: Lenguaje de protección - Activos - Organización - Ofuscación - Marca de agua - Marca de nacimiento

CONTEXTO

La línea de investigación descripta en este artículo se desarrolla en el Laboratorio de Calidad e Ingeniería de Software (LaCIS) de la Universidad Nacional de San Luis; y se encuentra enmarcada dentro del proyecto: “Ingeniería de Software: Estrategias de Desarrollo, Mantenimiento y Migración de Sistemas en la Nube”, perteneciente a la misma. Dicho proyecto, es reconocido por el programa de incentivos, y es la continuación de diferentes proyectos de investigación de gran éxito a nivel nacional e internacional.

1. INTRODUCCIÓN

El avance tecnológico ha llevado a que, a través de dispositivos como computadoras y celulares, se hagan una gran cantidad de actividades que antes eran inimaginables. Hoy en día, imaginar una vida sin estos dispositivos es casi imposible, ya que se utilizan como medio de comunicación y diversión, como medio de trabajo, ayuda a que se puedan hacer actividades de manera sencilla y sin mucho esfuerzo, se pagan impuestos a través de las aplicaciones web, los estudiantes tienen acceso a sus contenidos académicos mediante el uso de aulas virtuales, se realizan transferencias bancarias, las empresas permiten a sus empleados hacer home office, etc.

Aunque no se vea a simple vista, todos los beneficios antes mencionados se dan gracias a la implementación del software, el cual se ha popularizado convirtiéndose en una herramienta muy demandada por empresas, personas, instituciones educativas, etc. De este modo, el software ha pasado de ser una herramienta sólo utilizada por algunos a estar al alcance de cualquier persona.

Sin embargo, todas estas actividades deben realizarse a través de un software que sea seguro, ya que cualquier fallo en las aplicaciones puede causar desastre.

Las empresas de desarrollo de software, en la actualidad, realizan mucho esfuerzo en producir soluciones de calidad y cada vez más seguras. A pesar de ello, los crackers que poseen alta capacidad intelectual hacen uso de sus habilidades para vulnerar las estrategias de protección y, como consecuencia, las empresas deben seguir adaptando sus procesos de desarrollo para incorporar prácticas de seguridad acordes para dificultar la violación de sus sistemas.

En este sentido, Microsoft elaboró SDL (Security Development Lifecycle), que introduce aspectos de privacidad y seguridad en todas las fases de desarrollo, permitiendo a los desarrolladores enfocarse en la creación de un software altamente seguro, capaz de cumplir con los requisitos de seguridad y reducir costos de desarrollo.

Oracle, por su lado, desarrolló OSSA (Oracle Software Security Assurance), que es una metodología para incorporar seguridad en todo el proceso de desarrollo de sus productos: diseño, construcción, pruebas y mantenimiento, cumpliendo con los requisitos de seguridad y brindando una experiencia de propiedad más rentable.

OWASP Foundation describe SAMM (Software Assurance Maturity Model), un modelo de madurez útil para proporcionar una forma efectiva y medible de analizar y mejorar el ciclo de vida de desarrollo seguro, ya que admite de manera completa el ciclo de vida del software y es independiente de la tecnología y los procesos que se utilizan.

SAMM es de naturaleza evolutiva, está impulsada por el riesgo y es un marco abierto para ayudar a las organizaciones a formular e implementar una estrategia para la seguridad del software. Ayuda a i) evaluar las prácticas de seguridad de software existentes de una organización; ii) construir un programa de garantía de seguridad de software equilibrado en iteraciones bien definidas; iii) demostrar mejoras concretas a un programa de aseguramiento de la seguridad; y iv) definir y medir las actividades relacionadas con la seguridad en toda la organización.

Con las mejoras mencionadas con anterioridad, el software ha aumentado su calidad y seguridad. Sin embargo, todavía sigue siendo vulnerable, es decir, toda estrategia de seguridad tarde o temprano puede ser violada.

2. LÍNEAS DE INVESTIGACIÓN Y DESARROLLO

La Protección del Software asume que no existen estrategias totalmente seguras, y que siempre el cracker puede encontrar la forma de superarlas. Por lo tanto, es una disciplina destinada a proteger el software de los ataques constantes de los crackers. Se entiende por protección a la estrategia de retrasar el éxito del ataque el mayor tiempo posible. Para lograr este objetivo, la Protección de Software utiliza varias estrategias como: *ofuscación de código*, *marca de agua de software*, *huellas dactilares*, *marca de nacimiento*, etc.

Cada estrategia tiene su campo de aplicación, por ejemplo, la *ofuscación de código* es útil para evitar el análisis del programa. Es bien sabido que los especialistas en informática

pueden comprender rápidamente el software con una alta calidad y solo necesitan crear herramientas de análisis de programas para acelerar el proceso. La ofuscación oculta las propiedades del software al agregar ruido al código fuente, lo que dificulta la extracción de información a través de herramientas de análisis de programas y por lo tanto retrasa el objetivo del cracker.

Las *marcas de agua*, las *huellas dactilares* y las *marcas de nacimiento* de software tienen como objetivo detectar y rastrear la distribución ilegal de software. Por ejemplo, es posible detectar la manipulación del software agregando una marca de agua dentro del código fuente.

En la presente investigación, se realizó un estado del arte acerca de las técnicas de protección de software y la utilización de las mismas.

Muchos autores presentan trabajos acerca de cuáles son las principales técnicas de protección utilizadas y cómo las mismas se utilizan dentro del ciclo de vida del software[1] para elaborar una alta tolerancia a fallos en el mismo[5] y mejorar su seguridad. Los autores mencionan que las técnicas más utilizadas son: serial number, empaquetado, protección de máquina virtual, antidepuración[1], strong name, watermarking, software dog, code obfuscation[8], entre otros.

Por otro lado, existen estudios que hacen uso de las técnicas antes mencionadas, y las combinan para proteger diversas partes del software. Algunos las utilizan a nivel de clase[2]; otros utilizan una combinación de algoritmos, ofuscación y marca de agua en una serie de pasos establecidos[7,10]; otros proponen métodos para la construcción o mejora de las estrategias de protección ya existentes[9,13,14,17]; otros establecen un método de cifrado para proteger el derecho intelectual[6], o utilizan criptografía para generar un mecanismo de protección más novedoso[4]; y otros especifican modelos para medir la resistencia de un software dedicado a proteger sistemas[16].

Además de las estrategias ya mencionadas hay otras que utilizan una combinación de software y hardware, que también se construyen con el objetivo de evitar la manipulación. Hay autores que proponen un nuevo esquema de resistencia a la manipulación basado en encriptación y haciendo uso de un dispositivo USB[3].

Por otro lado, hay autores que crean su propia técnica o modelo de protección[11,12,15].

Dado el estado del arte realizado hasta el momento podemos concluir en que, si bien se han estudiado diversas formas de proteger el software, no existe aún un lenguaje que, en base a diversas primitivas de protección y funciones de alto nivel, genere estrategias de protección.

En la actualidad, los integrantes de la presente línea de investigación, han implementado algunas primitivas de protección con la intención de utilizarlas en un futuro. Estas primitivas serán utilizadas en la definición de un lenguaje específico del dominio que, haciendo uso de funciones de alto nivel como map, filter, reduce, etc., se puedan combinar las mismas para crear estrategias de protección.

3. RESULTADOS ESPERADOS

A través del trabajo realizado por los integrantes de la presente línea de investigación, se esperan obtener los siguientes resultados:

- Obtener un estado del arte aún más amplio acerca del uso de técnicas y estrategias de protección de software.
- Obtener conocimientos más profundos acerca del diseño y la construcción de lenguajes específicos del dominio.

- La generación de un lenguaje específico del dominio compuesto por diversas primitivas de protección que, haciendo uso de funciones de alto nivel, se combinen para crear estrategias de protección.

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 maestría en Ingeniería de Software de la Universidad Nacional de San Luis, en el marco de los proyectos de investigación mencionados en el contexto del presente documento.

5. BIBLIOGRAFÍA

[1]. L. Guoyuan, T. Jiutao, and G. Bing. "A survey on shareware protection schemes". In: 2010 2nd International Conference on Signal Processing Systems. Vol. 3. 2010, pp. 697–700. doi: 10.1109/ICSPS.2010.5555515.

[2]. X. Guangli and C. Zheng. "The Code Obfuscation Technology Based on Class Combination". In: 2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science. 2010, pp. 479–483. doi: 10.1109/DCABES.2010.104.

[3]. S. Zheng and J. Liu. "An USB-Key based approach for software tamper resistance". In: 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE). Vol. 6. 2010, pp. 506–509. doi: 10.1109/ICACTE.2010.5579199.

[4]. H. Lin, X. Mo, and Y. Gao. "Based on RSA and Self-Modifying Mechanism of Software Protection". In: 2010 International Symposium on Parallel and Distributed Processing with Applications. 2010, pp. 474–477. doi: 10.1109/ISPA.2010.12.

[5]. L. Xiong and Q. Tan. "A Configurable Approach to Tolerate Soft Errors via Partial Software Protection". In: 2011 IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications Workshops. 2011, pp. 260–265. doi: 10.1109/ISPAW.2011.45.

[6]. J. Hu et al. "A key hiding based software encryption protection scheme". In: 2011 IEEE 13th International Conference on Communication Technology. 2011, pp. 719–722. doi: 10.1109/ICCT.2011.6157970.

[7]. L. Chen and C. Zhang. "A novel algorithm for .NET programs watermarking based on obfuscation". In: 2012 International Symposium on Instrumentation Measurement, Sensor Network and Automation (IMSNA). Vol. 2. 2012, pp. 583–586. doi: 10.1109/MSNA.2012.6324652.

[8]. B. Chen et al. "Cloud licensing model for .NET software protection". In: 2012 7th International Conference on Computer Science Education (ICCSE). 2012, pp. 1069–1074. doi: 10.1109/ICCSE.2012.6295248.

[9]. Y. Wang et al. "CHI Based Instruction-Words Based Software Birthmark Selection". In: 2012 Fourth International Conference on Multimedia Information Networking and Security. 2012, pp. 892–895. doi:10.1109/MINES.2012.84

[10]. C. Collberg, A. Gibson, S. Martin, N. Shinde, A. Herzberg and H. Shulman, "Provenance of exposure: Identifying sources of leaked documents". In: 2013

IEEE Conference on Communications and Network Security (CNS), National Harbor, MD, USA, 2013, pp. 367-368, doi: 10.1109/CNS.2013.6682731.

[11]. C. Basile and M. Ceccato, "Towards a unified software attack model to assess software protections". In: 2013 21st International Conference on Program Comprehension (ICPC), San Francisco, CA, USA, 2013, pp. 219-222, doi: 10.1109/ICPC.2013.6613852.

[12]. A. Averbuch, M. Kiperberg and N. J. Zaidenberg, "Truly-Protect: An Efficient VM-Based Software Protection". In: 2013 IEEE Systems Journal, vol. 7, no. 3, pp. 455-466, Sept. 2013, doi: 10.1109/JSYST.2013.2260617.

[13]. P. Chan, L. Hui, S. Yiu, "Dynamic Software Birthmark for Java Based on Heap Memory Analysis". In: 2011 Communications and Multimedia Security, pp. 94-107

[14]. A. Kulkarni and R. Metta, "A New Code Obfuscation Scheme for Software Protection". In: 2014 IEEE 8th International Symposium on Service Oriented System Engineering, Oxford, UK, 2014, pp. 409-414, doi: 10.1109/SOSE.2014.57.

[15]. A. A. Osti, B. A. P. Botelho and N. Uto, "Implementation aspects of software guards: A case study". In: 2014 The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014), London, UK, 2014, pp. 262-267, doi: 10.1109/ICITST.2014.7038818.

[16]. S. Banescu, C. Collberg, V. Ganesh, Z. Newsham, A. Pretschner. "Code obfuscation against symbolic execution attacks". In: 2016 Proceedings of the 32nd Annual Conference on Computer Security Applications, pp. 189-200. doi: 10.1145/2991079.2991114.

[17]. B. F. Demissie, M. Ceccato and R. Tiella, "Assessment of Data Obfuscation with Residue Number Coding". In: 2015 IEEE/ACM 1st International Workshop on Software Protection, Florence, Italy, 2015, pp. 38-44, doi: 10.1109/SPRO.2015.15.