Revista Ingeniería Agrícola, ISSN-2306-1545, E-ISSN-2227-8761, Vol. 9, No. 4 (octubre-noviembre-diciembre, pp. 63-68), 2019

REVISIÓN

http://opn.to/a/4SQA5

Revisión sobre técnicas de marcas de agua para software como protección a los contenidos digitales

Review of software watermarking techniques as protection for digital content

M.Sc. Yoandry Milián-Núñez*

Universidad Agraria de La Habana, Centro Universitario Municipal San Nicolás, Mayabeque, Cuba.

RESUMEN: La presente revisión bibliográfica está centrada en al área de investigación de las marcas de agua para software y la importancia que esto evidencia como conocimiento en la formación del universitario en ingeniería agrícola, agrónomo y otros, propiciando herramientas para la protección de los contenidos digitales, específicamente tiene como objetivo profundizar en la taxonomía de las marcas de agua para software las cuales pueden ser: estáticas y dinámicas. Analizar los diferentes algoritmos existentes en la actualidad sobre el tema, tales como: QP, QPS, SHKQ, huevos de pascua, D*ynamic graph-based software watermarking*, entre otros, los cuales estarían inmersos en el proceso de dar seguridad a un programa basado en la propiedad intelectual.

***Palabras clave*:** propiedad intelectual, derecho de autor, algoritmo, taxonomía.

ABSTRACT: The present bibliographic review is directed to the area of research of software watermarking and the importance that this evidences as knowledge on formation of university students of agricultural engineer, agronomy and others, bringing tools for the protection of the digital contents, and that specifically it has as objective to deepen in the taxonomy of software watermarking which can be static and dynamic. Are analyzed the different existing algorithms on the subject, such as: QP, QPS, SHKQ, Easter eggs, Dynamic graph-based watermarking software, among others, which would be immersed in the process of giving security to a program based on intellectual property.

***Keywords*:** intellectual property, copyright, algorithm, taxonomy.

INTRODUCCIÓN

En la actualidad las Tecnologías de la Información y las Comunicaciones (TIC) han cambiado los paradigmas tradicionales de procesamiento, almacenamiento y transmisión de información, requiriéndose, que los universitarios apliquen este campo del saber en la solución de problemas profesionales, vinculados a sus áreas específicas de desarrollo e investigación (Vargas *et al.*, 2016). En este sentido es importante que la conciban como instrumentos de trabajo en su modo de actuación como futuros ingenieros (González y Baserio, 2015).

La formación integral en el ingeniero agrícola, agrónomo, etc., se proyecta en la concepción de desarrollar, las habilidades informáticas en concordancia con las diferentes actividades ingenieriles donde realizan su práctica laboral a través de los diferentes niveles de formación del mismo y la aplicación de las mismas en la solución de problemas profesionales en un nivel de aplicación y de creación.

Es necesario que los estudiantes estas carreras dominen esta área del conocimiento como complemento de su formación general de las TIC, posean conocimientos más allá de un nivel informativo sobre la protección del *software,* en este caso particular las marcas de agua para *software* (*software watermarking*), pues las mismas permiten proteger los contenidos

*Autor para correspondencia: Yoandry Milián Núñez, e-mail: ymilann@unah.edu.cu

Recibido: 13/02/2019.
Aprobado: 29/07/2019.

digitales y demostrar el derecho de autoría que es producto de una creación (Milián, 2016). Propiciando el desarrollo de habilidades en términos de seguridad informática el cual tributará a evitar la violación de la propiedad intelectual del software creado.

El objetivo de este trabajo está encaminado a realizar una revisión bibliográfica concerniente a las marcas de agua para *software*, las cuales protegen el derecho de autor (*copyright*) frente a la piratería informática.

DESARROLLO DEL TEMA

Protección de contendidos digitales

Desde el surgimiento de la informática la protección de los contenidos digitales es un área de investigación, la copia ilegal de ellos se ha convertido en una práctica habitual (Nagra *et al*., 2002). Cuando el contenido digital es un *software*, las facilidades de copia, almacenamiento y el intercambio propician la piratería: un ejemplo, en el año 2014, según el reporte de piratería de *software* brindado por la Alianza de Negocios Informáticos (*Business Software Alliance*–BSA), las pérdidas monetarias por concepto de piratería de *software* ascienden a billones de dólares y se mantiene en ascenso, al haber un incremento del mismo en el 2017 por este concepto (BSA the Software Alliance, 2018).

Para evitar la piratería de *software* es necesario garantizar ciertos atributos de seguridad, los cuales se destacan: la privacidad, integridad, autenticidad y disponibilidad.

Estos atributos son garantizados en cierta medida mediante técnicas de protección de *software*. Entre las que destacan: ofuscación, cifrado de código, auto-verificación de integridad, diversidad de código, marcas de aguas para *software*, entre otros (Nagra, 2007).

En la protección de un *software* de los ataques de un *host* malicioso, no existe ninguna defensa infalible. Una vez infectada la terminal, se puede usar cualquier técnica para extraer los datos que se requieran, o violar su integridad. En realidad, los únicos factores que limitan esta vulnerabilidad son los recursos que puede invertir el *host* para analizar el código del programa (Moruno, 2008).

Si un usuario malintencionado dispone de tiempo y recursos computacionales el cual le permita obtener el código fuente de una determinada aplicación para modificarlo y luego recompilarlo, logrando así crear su propia versión del mismo y distribuirla bajo otro nombre. Por consiguiente, la alternativa para la protección del *software* radica, encontrar esquemas de protección que resulten excesivamente costosos de romper, en términos de recursos.

Ataques a la propiedad intelectual de un software

Se definen tres tipos de ataques a la propiedad intelectual contenida en un programa, la ingeniería inversa, la piratería, y el tampering de software.

Ingeniería inversa: La ingeniería inversa es el proceso de extracción del conocimiento o del diseño de un programa desde cualquier cosa hecha por el hombre (Eilam, 2005).

Tampering: consiste en realizar modificaciones no autorizadas en un programa.

Piratería de *software*: es la copia, distribución, utilización o elaboración ilegal de programas informáticos que constituyen actos de agresión contra la propiedad intelectual y el derecho de patentes y marcas.

La piratería de softwar*e* se divide en dos grupos:

Copias realizadas por el usuario final: amigos que se prestan discos entre sí, u organizaciones que no reportan el número real de instalaciones del material que realizaron, esto se reduce a la piratería sin ánimo de lucro.

Falsificación: duplicación y distribución a gran escala de *software* copiado ilegalmente. La cual la llamaremos piratería con fines de lucro.

Medidas contra los Ataques

Ofuscación: es la técnica de transformar el código en otra menos legible y de difícil compresión al ser humano, pero manteniendo su funcionalidad, de esta forma se modifica el código para esconder su propósito original y ahogar al agresor de información (Chen *et al*., 2017).

Tamper-proofing: técnica que se utiliza cuando interesa impedir que se ejecute un programa si éste ha sido alterado. Se definen algunas situaciones en las que un programa P no debería poder ejecutarse: P contiene una marca de agua (*watermark*) dinámico, y el código que construye dicho *watermark* ha sido alterado. Un virus ha sido adjuntado a P. P es una aplicación de comercio electrónico (*e-commercey*) la seguridad de su contenido se ha visto comprometida.

Marcas de agua para *software*

Las marcas de agua permiten empotrar un mensaje en un *software* para demostrar su autoría, es decir, la autenticidad del mismo. A estas técnicas de protección se le pueden hacer varios ataques los cuales pueden ser: de sustracción, deformación, adicción y confabulación (Collberg *et al*., 2004).

Estas técnicas se amparan en la Ley de la Propiedad Intelectual[2], de esta forma se disuade a usuarios malintencionados de hacer uso de un *software* no autorizado por sus propietarios, por lo que permite certificar en caso de litigio, la propiedad del producto y para ello se inserta un *copyright*[3] que pueda resistir posibles ataques e imposibilitar su posterior extracción (Dalla y Pasqua, 2017).

Una violación del *copyright* trae como consecuencia pérdidas económicas a las fábricas de *software* y por consiguiente un deterioro de la economía.

En el lenguaje coloquial es la vía para proteger un *software* a la falsificación de la propiedad intelectual definiéndose el *watermarking* en empotrar un mensaje secreto dentro de otro mensaje.

[2] La ley de Propiedad Intelectual: el reconocimiento de un derecho particular en favor de un autor u otros titulares de derechos, sobre las obras del intelecto humano.

[3] Expresa los derechos de autor, cuya intención es proteger la propiedad intelectual de un uso ilegal y evitar que se alegue dichos derechos a una persona o institución que no le corresponda

Esta técnica no impide realizar copias ilegales de un objeto, sino que nos permite probar la autoría de un contenido digital. El *watermarking* multimedia, el mensaje secreto es la información de *copyright*, y el mensaje que la contiene es una imagen, audio o vídeo (Collberg y Thomborson, 1998).

El *software watermarking* se puede describir básicamente como empotrar una estructura W en un programa P de modo que W puede ser localizado y extraído de P de una forma fiable incluso después que P haya sido sometido a transformaciones

Para que el *software watermarking* sea útil, éste debe ser resistente a la variedad de posibles ataques que puedan surgir. Se estima que un determinado atacante puede lograr su propósito dado a los suficientes recursos y el tiempo necesario. Por consiguiente, la meta de las técnicas de *software watermarking* es diseñar marcas que precisen de un tiempo, un esfuerzo y recurso considerable para ser destruidas. De este modo se intenta que resulte menos costoso adquirir una copia de forma legal, o escribir un programa propio, al de adquirirlo de forma ilegal (Dey *et al*., 2018).

Las técnicas de *software watermarking* contienen tres premisas esenciales:

- La primera es la tasa de datos, que no es más que la mayor cantidad de datos ocultos posibles que pueden ser empotrados en el mensaje (programa). Es vital que la marca codifique la mayor cantidad de información posible sin incrementar considerablemente el tamaño del código o del entorno de ejecución del programa.
- La segunda premisa es la imperceptibilidad, que de esta forma demuestra el grado de capacidad que tiene la información embebida para pasar desapercibida ante los ojos de un observador.
- Y por último la marca debe tener la resistencia frente a diferentes ataques de un adversario. En cualquier técnica existe un compromiso entre estas tres premisas, de modo que por ejemplo una tasa de datos alta implica baja imperceptibilidad y resistencia, o también incrementar la resistencia aplicando redundancia en la información tiene como consecuencia una baja tasa de datos.

Taxonomía de las marcas de agua para *software*

Las técnicas *watermarking* pueden clasificarse atendiendo a varios criterios, según definiciones de diferentes autores que plantean que se pueden clasificar en dos grandes grupos: frágiles y robustas; las frágiles se desglosan de tipo de validación y de licencia y en el caso de las robustas se clasifican en autoría y de tipo *fingerprinting* (huella digital) las cuales a su vez se derivan en simples o múltiples (Nagra, 2007).

La clasificación hecha por el autor del artículo: Un modelo para la seguridad de componentes de *software*, coincide en muchos aspectos con lo definido por Collberg *et al.* (2004), que plantea que debe adicionársele el criterio de visibilidad es decir si son visibles o invisibles, además de considerar según la técnica de extracción si son estáticas o dinámicas.

Existen dos vías para extraer la marca en un programa: de forma estática o dinámica. Los estáticos son guardados en el ejecutable de programa que pudiera ser la sección de datos inicializados (donde se guardan los *strings* estáticos), la sección de texto (código ejecutable) o la sección simbólica (información del *debug*) del ejecutable.

Existen dos tipos principales de *watermarks* estáticos: *watermarks* de código, los cuales son almacenados en la sección del ejecutable que contiene instrucciones, y *watermarks* de datos, los cuales se almacenan en cualquier otra sección, incluyendo *headers*, secciones de *strings*, secciones de información del *debug*.

Los *watermark* dinámicos, se almacenan en el estado de ejecución de un programa, en lugar de en el propio código. Esto hará que en muchos de ellos podamos aplicar *tamper-proofing* para neutralizar ataques por ofuscación.

En los *watermark* de *software* dinámico la aplicación "O" es ejecutada con una secuencia de entrada predeterminada que hace que el programa entre en un estado que representa el *watermark*. Los distintos métodos se diferencian según la parte del estado del programa en la que se almacenen, y según la manera de extraerlos.

Tipos de Ataques a marcas de agua para *software*:

En general, no existe ninguna técnica de *watermarking* inmune a todo tipo de posibles ataques, y muchas veces deben emplearse varias técnicas simultáneamente para alcanzar el grado deseado de resistencia. Ejemplo de los diferentes tipos de ataques, se asume el siguiente escenario: Alice embebe una marca W con una clave K en un objeto "O" y después lo vende a Bob. Antes de que Bob pueda vender "O" a Douglas, Bob tiene que asegurarse de que ha conseguido inutilizar el *watermark*, de lo contrario, Alice será capaz de demostrar que sus derechos de la propiedad intelectual han sido violados. Los tres principales tipos de ataques que Bob puede lanzar contra el *watermark* son:

Sustracción**:** Si Bob puede detectar la presencia y la localización (aunque aproximada) de W, podría intentar extraerlo de "O".

Deformación**:** Si Bob no puede localizar W y está dispuesto a aceptar una pequeña degradación en la calidad de "O", puede aplicar transformaciones que distorsionen uniformemente el objeto y por extensión cualquier *watermark* que pueda contener.

Adición: Bob puede aumentar "O" insertando su propio *watermark* W' de modo que Alice no puede demostrar que su marca precede temporalmente a la de Bob.

Confabulación: es cuando el ataque consiste en comparar varias copias de un programa, que sean de diferentes proveedores, lo cual permitirá al atacante estar en condiciones de detectar dónde está la marca.

Diseño de una técnica de marcas de agua para *software*

El diseño de una técnica de *software watermarking* debe dar respuesta de forma general a las premisas esenciales del mismo, se consideran tres aspectos a tener en cuenta en el proceso de diseño (Haoyu *et al*., 2019):

1. Tasa de datos: ¿Qué tamaño tiene la marca en comparación con el tamaño del programa?
2. Forma del programa que lo contiene: ¿En qué tipo de código

será distribuido el programa: Código interpretado, para ejecutarse sobre una máquina virtual, o código binario nativo?
3. Modelo de amenazas esperado: ¿Qué tipos de ataques podemos esperar por parte de Bob para atacar el *watermark*s?. Dependiendo de los recursos necesarios que pueda tener Bob para llevarlos a cabo.

Marcas de agua para *software* estático y dinámico

*Watermark*s estáticos de datos

Los *watermark*s estáticos de datos tienen una gran facilidad para ser construidos y reconocidos, ejemplo de lo antes mencionado es cuando la información del *copyright* puede estar contenida en una imagen JPEG lo cual permite fácilmente su extracción del código de un navegador (*netscape*) obteniendo todos los *strings* definidos y filtrando por los que contienen la palabra "*copyright*". Un segundo ejemplo es empotrar el *watermark* en una imagen (o audio o video digital) usando un algoritmo de *watermarking* de multimedia y luego almacenar la imagen en la sección de datos estáticos del programa. Este tipo de *watermark*s son muy vulnerables a ataques de deformación por ofuscación. Por lo que un ofuscador podría dividir todos los *strings* (y otros datos estáticos) en *substrings* que serían repartidos por el ejecutable; esto haría el reconocimiento del *watermark* casi imposible. Un método complejo sería convertir todos los datos estáticos en un programa que produjese los datos (Hsu & Tu, 2019).

Un método simple de *software watermarking* estático fue descrito por Holmes (1994), este método consistía en la copia maestra del programa que contiene un segmento de datos que es no usado por el mismo. La localización y tamaño de este segmento sin usar son determinados cuando el programa es conectado. Cuando una copia de este programa está hecha para distribución autorizada, este segmento es sobrescrito con la información del *watermark*s, como la fecha, la hora, y el destino de la copia. Holmes propone este método para la distribución en *Internet* de un programa.

*Watermark*s estáticos de código

Los *watermark*s de código se construyen de forma similar, dado que el código de objeto también contiene información redundante. Si no existen dependencias de datos o de control entre dos instrucciones de código consecutivas, éstas pueden ser dispuestas en cualquier orden. Entonces, un *bit* de *watermarking* podría ser codificado dependiendo de si las dos instrucciones están en orden lexicográfico o no. Este método fue usado por *International Business Machines Corporation* (IBM) el cual usó una variación de esta técnica para construir una *watermark*s en su *software* consistente en el orden en que los registros eran puestos y sacados de la pila.

Otro método propuesto consiste en codificarlo en el orden de las opciones de varios bloques. Davidson propuso un método innovador en el cual un número de serie se codificaría en la secuencia básica de bloques de los grafos de control de flujo del programa (Davidson & Myhrvold, 1996). Esto consistía en un método para generar y auditar firmas para módulos ejecutables en el que cada copia contiene una firma única que la distingue. La firma de cada copia autorizada, se codifica en el orden de las instrucciones del módulo ejecutable teniendo en cuenta que cada módulo ejecutable se compone de múltiples bloques de instrucciones.

Para insertar la firma se selecciona un grupo de bloques que sigan un flujo de ejecución y se reordenan, posteriormente se modifican adecuadamente para no romper el flujo de ejecución, y por último se substituyen los bloques originales por los modificados. La copia modificada es funcionalmente equivalente a la original pero ahora los bloques reordenados proporcionan una firma única.

Este método fue una de las primeras propuestas formales de *software_watermarking*. Su principal característica es su sencillez, pero al mismo tiempo su debilidad, el mismo es fácil de atacar mediante optimizaciones de código, mediante simples reordenaciones aleatorias de bloques básicos, o mediante ofuscaciones de código: insertar sentencias condicionales que siempre sean ciertas. En el caso de reordenaciones aleatorias de bloques básicos, se trata de un ataque por adición, es decir, si un atacante reordena de nuevo los bloques básicos del módulo ejecutable, estaría introduciendo su propia firma y por tanto sería totalmente imposible recuperar la firma original.

En este ámbito existen varios algoritmos de protección como son: QP, QPS, QPI, SHKQ, *Graph Theoretic Watermarking* (GTW).

Los autores Dey *et al.* (2018) y Hsu & Tu (2019) esgrimen que los algoritmos QP, QPS y QPI utilizan la asignación de registros como herramienta para codificar un mensaje, estos dos últimos surgen como mejora del algoritmo QP, lo que aumenta su grado de imperceptibilidad por lo que dificulta la ubicación del *watermark*, tienen una baja tasa de datos, aunque tienen poca resistencia ante ataques simples.

En el caso del algoritmo SHKQ y sus dos variantes se puede concluir que la primera variante del mismo se basa en la modificación de los índices de la tabla de las variables locales de los métodos, haciendo variar los *opcodes* que incluyen dichos índices, al contrario de la segunda variante que requiere de un diccionario de modificaciones más complejo para sustituir o insertar grupos de hasta cuatro instrucciones (Collberg *et al.*, 2004). Este último es más flexible a la hora de escoger elementos para el vector de frecuencia, ya que el primero se reduce a los *opcodes* que incluyen el operando de la tabla de variables locales.

*Software Watermark*ing dinámicos

Los *watermark*ing dinámicos, se almacenan en el estado de ejecución de un programa, en lugar del propio código. Esto hará que en muchos de ellos podamos aplicar *tamper-proofing* para neutralizar ataques por ofuscación. En los *watermark*s de *software* dinámico la aplicación O es ejecutada con una secuencia de entrada predeterminada que hace que el programa entre en un estado que representa el *watermark*. Los distintos métodos se diferencian según la parte del estado del programa en la que se almacenen, y según la manera de extraerlos (Collberg y Thomborson, 1999).

*Watermark*s de Huevos de Pascua

Este *watermark*s realiza una acción que es inmediatamente perceptible para el usuario, haciendo que la extracción de la marca sea trivial. El *watermark* consiste en una porción de código que sólo se ejecuta si el usuario ejecuta la aplicación con una secuencia de entrada determinada.

*Watermark*s de Estructuras Dinámicas de Datos

Este tipo de *watermark* se ubica en tiempo de ejecución en la *heap* (memoria dónde se cargan los objetos), la pila (*stack*) o en los datos globales. Esto sólo ocurre cuando se ejecuta dicho programa con una secuencia de entrada determinada. La forma de extracción es examinando los valores que contienen las variables del programa una vez se ha alcanzado el final de los datos de entrada, esto se logra con una rutina específica que se ejecuta en el programa, o con el debuggado del programa. El mismo no produce ninguna salida de datos, no es inmediatamente perceptible para un atacante cuando se introduce la secuencia específica de entrada.

Watermarks de Traza de Ejecución

El *watermark* es empotrado en la traza de ejecución del programa (instrucciones, direcciones, o ambas) cuando éste se ejecuta con una secuencia de entrada determinada. El *watermark* se extrae mediante la monitorización de algunas propiedades estadísticas de la traza de direcciones de entradas/salidas de la secuencia de operadores ejecutada (Collberg y Thomborson, 1999).

Dynamic Graph-Based *Software watermarking*

Este algoritmo fue desarrollado por los autores Collberg y Thomborson al cual denominaron CT. El mismo se encuentra implementado en Java e incluido en la herramienta SandMark. Dicho algoritmo utiliza estructuras dinámicas de datos para codificar la marca, esto consiste en incrustar la marca en la topología de un grafo construido de forma dinámica en memoria, y en tiempo de ejecución, el cual se divide en subgrafos. En la extracción de la marca, se asume una clave secreta K, que es la secuencia de entrada de la aplicación (Collberg *et al.*, 2004).

Un diseño basado en grafos dinámicos trae grandes ventajas:

- El parecido entre el código de la aplicación y del *watermark*
- La posibilidad de empotrar *watermark*s de gran tamaño
- Las posibilidades de aplicar técnicas de *tamper-proofing*

Otra gran ventaja de este diseño viene dada por la posibilidad de dividir un grafo muy grande en varios subgrafos más pequeños. Lo cual permite esparcir el *watermark* a lo largo de toda la aplicación, lo que posibilita el gran tamaño del *watermark* sin que disminuya el grado de imperceptibilidad del mismo. El *watermark* se encuentra codificada en forma de datos, el código que generado, es el resultado de su ejecución (Chen *et al.*, 2017).

El resultado de la ejecución de este código genera una estructura de datos que se almacena en memoria, si se diseña apropiadamente estas estructuras de datos, se pueden dotar de propiedades específicas para luego insertar código que verifique las propiedades de las mismas.

Threading Software watermarking

Esta técnica se basa en introducir nuevos *threads* en secciones del programa en las que sólo se ejecuta un único *thread*. En un programa con múltiples *threads* ejecutándose simultáneamente, se puede dar el caso en el que varios de ellos intenten leer o escribir sobre datos compartidos, o de forma más general, intenten usar recursos compartidos simultáneamente. Cuando se da esta situación, el resultado final de la ejecución dependerá únicamente de cómo han sido programados los distintos *threads*. Lo cual permite que múltiples *threads* compartan recursos de manera controlada (Nagra, 2007).

Este algoritmo consta de tres fases y está realizado sobre *Java Bytecode*

Tracing: Se analiza y captura la traza de ejecución del programa dada la entrada secreta I.

- *Embedding*: Se selecciona un número W (el *watermark*) y se empotra en la secuencia de ejecución de los distintos *threads*.
- *Recognition*: Se ejecuta el código con la entrada secreta I, se analiza la traza de ejecución y se extrae el *watermark* W.

Diferencias entre los distintos tipos de algoritmos de *software watermarking dinámico*

El algoritmo CT consta de pocas prestaciones al no dividirse el grafo en varios subgrafos que se distribuyan a lo largo del camino (*path*) de ejecución dada la entrada secreta. Ya que las operaciones necesarias para construir el grafo no resultan sospechosas por sí solas, un gran número de ellas concentradas en un mismo punto pueden llegar a serlo. El algoritmo CT está diseñado para resistir ataques automatizados (optimizaciones, ofuscaciones), que el marcado es demasiado largo para realizar ataques por inspección manual.

El algoritmo *Threading Software watermarking* muestra que el *watermark*s es resistente especialmente a ataques por ofuscación, y ataques por decompilación y recompilación. En cambio, los ataques por adición representan la amenaza más eficaz para este tipo de *watermark*. (Haoyu *et al.*, 2019).

CONCLUSIONES

- En la revisión del estado del arte referente al tema evidencia que las propuestas hasta el momento en términos de seguridad sufren de falta de robustez en las mismas. Dichas técnicas no son infalibles en su totalidad a posibles ataques es necesario llevar a un análisis profundo en la selección de la misma: estática o dinámica, para tener un mayor grado de seguridad a lo que se requiere proteger.
- Es necesario que los estudiantes de la carrera de ingeniería agrícola, agronomía y otros, dominen esta área del conocimiento como complemento de su formación general de las TIC. Propiciando el desarrollo de habilidades en términos de seguridad informática el cual tributará a evitar la violación de la propiedad intelectual del software creado.

REFERENCIAS BIBLIOGRÁFICAS

BSA THE SOFTWARE ALLIANCE: "Software management: Security imperative, business opportunity", *[en línea]*, En: *BSA*, 2018, *Disponible en: https://www.bsa.org/files/2019-02/2018_BSA_GSS_Report_en_.pdf*, *[Consulta: 2 de febrero de 2019]*.

CHEN, Z.; JIA, C.; XU, D.: "Hidden path: dynamic software watermarking based on control flow obfuscation", *[en línea]*, En: *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, Ed. IEEE, vol. 2, pp. 443-450, 2017, ISBN: 1-5386-3221-7, *Disponible en: https://ieeexplore.ieee.org/abstract/document/8006048*, *[Consulta: 3 de enero de 2019]*.

COLLBERG, C.; CARTER, E.; DEBRAY, S.; HUNTWORK, A.; KECECIOGLU, J.; LINN, C.: "Dynamic path-based software watermarking", *[en línea]*, En: *ACM sigplan 2004 conference on programming language design and implementation*, Ed. ACM, vol. 39, pp. 107-118, 2004, DOI: 10.1145/996841.996856, ISBN: 1-58113-807-5, *Disponible en: https://dl.acm.org/citation.cfm?id=996856*, *[Consulta: 10 de julio de 2015]*.

COLLBERG, C.; THOMBORSON, C.: *On the limits of software watermarking*, no. 1173-3500, Inst. Department of Computer Science, The University of Auckland, New Zealand, Technical Report #164, New Zealand, 1998.

COLLBERG, C.; THOMBORSON, C.: "Software watermarking: Models and dynamic embeddings", *[en línea]*, En: *Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, Ed. ACM, pp. 311-324, 1999, DOI: 10.1145/292540.292569, ISBN: 1-58113-095-3, *Disponible en: https://dl.acm.org/citation.cfm?id=292569*, *[Consulta: 10 de enero de 2016]*.

DALLA, P.M.; PASQUA, M.: "Software watermarking: a semantics-based approach", *Electronic Notes in Theoretical Computer Science*, 331: 71-85, 2017, ISSN: 1571-0661, DOI: https://doi.org/10.1016/j.entcs.2017.02.005.

DAVIDSON, R.I.; MYHRVOLD, N.: *Method and system for generating and auditing a signature for a computer program*, no. US5559884, Inst. U.S. Microsoft Corporation, United States, 1996.

DEY, A.; BHATTACHARYA, S.; CHAKI, N.: "Software watermarking: Progress and challenges", *Indian National Academy of Engineering*, 4(1): 65-75, 2018, ISSN: 2366-326X, DOI: https://doi.org/10.1007/s41403-018-0058-8.

EILAM, E.: *Reversing: secrets of reverse engineering*, *[en línea]*, Ed. Wiley Published Inc, Indianapolis, Indiana, 589 p., 2005, ISBN: 978-0-7645-7481-8, *Disponible en: https://www.amazon.com/Reversing-Secrets-Engineering-Eldad-Eilam/dp/0764574817*, *[Consulta: 25 de marzo de 2015]*.

GONZÁLEZ, T.; BASERIO, B.: "La integración de las tecnologías de la información y las comunicaciones en la carrera de agronomía", *Cuadernos de Educación y Desarrollo*, 2015, ISSN: 1989-4155.

HAOYU, M.A.; CHUNFU, J.I.A.; SHIJIA, L.; WANTONG, Z.; DINGHAO, W.: "Xmark: Dynamic Software Watermarking using Collatz Conjecture", *IEEE Transactions on Information Forensics and Security*, 2019, DOI: 10.1109/TIFS.2019.2908071.

HOLMES, K.: *Computer software protection*, no. US5287407, Inst. U.S. International Buisness Machines, United States, 71-85 p., 1994.

HSU, C.S.; TU, S.F.: "Digital Watermarking Scheme for Copyright Protection and Tampering Detection", *International Journal on Information Technologies & Security*, 11(1), 2019, ISSN: 1313-8251.

MILIÁN, N.Y.: *Algoritmo de marcas de aguas para software*, *[en línea]*, Universidad Agraria de La Habana, MSc. Thesis, San José de las Lajas, Mayabeque, Cuba, 100 p., 2016, *Disponible en: http://ebiblio.unah.edu.cu:8080/jspui/handle/123456789/3951*.

MORUNO, M.: *Estudio de técnicas de inserción de marcas de agua sobre software (Software watermarking)*, *[en línea]*, Universidad Politécnica de Cataluña, Eng. Thesis, Cataluña, España, 157 p., 2008, *Disponible en: http://hdl.handle.net/2099.1/5151*, *[Consulta: 20 de septiembre de 2015]*.

NAGRA, J.: *Threading Software Watermarks*, *[en línea]*, Auckland University, PhD. Thesis, New Zealand, 158 p., 2007, *Disponible en: https://www.cs.auckland.ac.nz/~cthombor/Students/jnagra/jnagrathesis.pdf*, *[Consulta: 9 de febrero de 2019]*.

NAGRA, J.; THOMBORSON, C.; COLLBERG, C.: "A functional taxonomy for software watermarking", *[en línea]*, En: *Australian computer science communications*, Ed. Australian Computer Society, Inc., vol. 24, Australia, pp. 177-186, 2002, ISBN: 0-909925-82-8, *Disponible en: https://dl.acm.org/citation.cfm?id=563822*, *[Consulta: 20 de abril de 2015]*.

VARGAS, L.M.; DE PAYER, V.E.; DI GIANANTONIO, A.: "Marcas de agua: una contribución a la seguridad de archivos digitales", *Revista de la Facultad de Ciencias Exactas, Físicas y Naturales*, 3(1): 49-54, 2016, ISSN: 2362-2539.

Yoandry Milián Núñez, Profesor de la Universidad Agraria de La Habana, Centro Universitario Municipal San Nicolás, Mayabeque, Cuba, e-mail: ymilann@unah.edu.cu

El autor de este trabajo declara no presentar conflicto de intereses.