Tendencias actuales en la concepción de subsistemas de hardware para la protección de aplicaciones

MSc. Humberto Díaz Pando, Profesor Instructor; MSc. Yulier Núñez Musa, Profesor Instructor; Dr. Roberto Sepúlveda Lima, Profesor Titular, Decano Facultad de Ingeniería Informática, Instituto Superior Politécnico "José Antonio Echeverría", CUJAE; y Dr. Sergio A. Cuenca Asensi, Subdirector de Investigación, Dpto. Tecnología Informática y Computación, Universidad de Alicante, España.
hdiazp@ceis.cujae.edu.cu, jnunezm@ceis.cujae.edu.cu, sepul@ceis.cujae.edu.cu, sergio@dtic.ua.es

1 Introducción

El creciente auge que ha alcanzado Internet, junto con el desarrollo de múltiples herramientas informáticas, ha posibilitado un aumento en el intercambio digital de información entre las personas; además de la posibilidad de disponer de un gran cúmulo de información de variados temas en cualquier momento. Esta situación, sin dudas, implica numerosas ventajas, aunque desafortunadamente existen procedimientos que posibilitan a ciertas personas realizar acciones ilícitas —muchas de estas acciones no están debidamente tipificadas como delitos en los códigos legales correspondientes—, que persiguen con esto beneficios propios. Dicha circunstancia se ve favorecida por el hecho de que, en Internet, es posible publicar información con relativa facilidad y con controles de seguridad prácticamente nulos. Un ejemplo son los sitios pertenecientes a personas o grupo de ellas, dedicados a divulgar la forma de explotar las vulnerabilidades y fallas de las aplicaciones. Por otra parte, se publica también información de cómo establecer procesos de ingeniería inversa a componentes ejecutables—a través de los cuales se facilita la comprensión del funcionamiento de las aplicaciones—, con el propósito de violar su integridad y obtener información confidencial acerca de algoritmos empleados.

Otra de las situaciones que existe en la actualidad está relacionada con el proceso de desarrollo de software. En el momento de desarrollar un producto informático, los requisitos vinculados con la seguridad pueden ser apreciados bajo varios enfoques. Uno de ellos es que sean considerados como requisitos no funcionales, que muchas veces son relegados a otros momentos del desarrollo del producto. Otro de los enfoques es que sean algunos requisitos que da el usuario; pero no se piensa en corregir vulnerabilidades que, en ocasiones, aparecen cuando se programa la aplicación. Estos u otros casos traen consigo la violación posterior de los mecanismos de seguridad o la explotación de vulnerabilidades, que posibilitan el acceso a los datos confidenciales o la modificación del código.

A causa de las situaciones descritas, en los últimos años han surgido una gran variedad de métodos de protección de software que tratan de evitarlas. Esta diversidad está dada, principalmente, por el desarrollo de muchos métodos de propósito específico (*ad-hoc*) por parte de la industria de software —números de licencia, registro en línea de la aplicación, llaves de hardware, etc.—, con el fin de evitar que sus productos sean copiados o analizados por atacantes o por otras compañías con fines de lucro.

El objetivo de este trabajo es proponer una taxonomía de las metodologías de protección de aplicaciones basadas en componentes de hardware.

2 Escenarios de aplicación de los mecanismos de protección de software

En este epígrafe se analizarán diversos escenarios donde se requiere la aplicación de mecanismos de protección. Varios autores [1-6] han investigado sobre este tema, a continuación se ofrecerá un análisis de los diferentes criterios existentes. Los escenarios que serán descritos no siempre se materializan de forma independiente, o sea, es posible que un escenario desencadene la aparición de otro. La descripción de cada uno se presentará bajo la óptica del atacante y del desarrollador de software. Analizar el proceder de los atacantes siempre es un método válido para los desarrolladores, porque es posible determinar la forma en que

un adversario pudiera materializar los escenarios; para, luego, implementar un mecanismo de protección que garantice un determinado nivel de seguridad.

Desestabilización de sistemas

Un atacante establecería este escenario con el objetivo de hacer fallar un sistema para obtener algún beneficio. Para lograr su objetivo, el proceder más común es a partir de la detección de alguna vulnerabilidad en el sistema, por ejemplo, un desbordamiento de *buffer*. Estas vulnerabilidades son detectadas o bien por una herramienta o por la publicación de esta por otro atacante. Generalmente aparecen en las aplicaciones porque los desarrolladores obvian una serie de requisitos de seguridad, ya sea por las restricciones temporales del desarrollo del producto, por un mal diseño o, simplemente, por no considerar los posibles ataques.

Con la intención de evitar la materialización de este escenario, el reto de los desarrolladores es, sin duda, entregar a los usuarios un producto libre de fallas. Un buen proceder para lograrlo podría ser la utilización tanto de las técnicas como las herramientas que normalmente emplean los atacantes para detectar y explotar las vulnerabilidades en las aplicaciones. Con esto se detectarían y corregirían las vulnerabilidades presentes en el desarrollo del producto antes de que sea enviado al mercado.

Obtención de privilegios en el sistema

El principal objetivo del adversario, en este escenario, es obtener información sensible que se encuentra almacenada en los repositorios de datos del sistema. La forma en que el atacante pudiera conseguirlo sería intentando ganar privilegios en el sistema. Esto podría lograrse a partir de la explotación de una vulnerabilidad ya conocida o ejecutando un ataque por ingeniería inversa —ver epígrafe siguiente—, que violen el mecanismo de autenticación. En muchas ocasiones, estas acciones son posibles debido a deficiencias en el propio proceso de desarrollo de las aplicaciones. Un ejemplo de deficiencia lo constituye el almacenamiento de la clave por parte de los desarrolladores, de forma estática, dentro del código o la presencia de puertas traseras en la aplicación —de manera premeditada o no—.

Por tal motivo, los esfuerzos de los desarrolladores deben estar dirigidos principalmente a buscar mecanismos de autenticación que no revelen, en ningún momento, información sobre la clave o el propio mecanismo. Además deben garantizar que la aplicación no contenga fallas que puedan desencadenar este tipo de contexto.

El resto de los escenarios se encuentra relacionado con un problema que, en el transcurso de los años, ha ido adquiriendo una magnitud significativa. Este problema es el de la piratería de software, donde los recientes estudios revelan que las pérdidas, por este concepto, han aumentado considerablemente en los últimos años [7].

Copia indiscriminada

Aquí el posible adversario adquiere el software original mediante su compra. Cuando sucede esto el adversario tiene en su poder una licencia válida del software. En este momento el "propietario" tiene libertad total para hacer cualquier cosa sobre la aplicación. Una de las posibles acciones es la replicación del software a varios usuarios, acción que no está incluida dentro del acuerdo de licencia. Este escenario ocurre principalmente en mecanismos de licencia o en mecanismos de protección que no son lo suficientemente fuertes.

Copia ilegal

En contraposición con el anterior, el adversario no cuenta con una copia válida de la aplicación, sino que posee una copia ilegal. El objetivo principal es buscar la forma de falsificar su licencia o violar su mecanismo de protección. Llegado este instante, la situación se comportaría como la descrita arriba, o sea, el atacante tendría en su poder una copia "válida" de la aplicación.

Las intenciones de los desarrolladores, en estos dos escenarios, es establecer un mecanismo de protección para garantizar que la creación de copias ejecutables de una aplicación sea un proceso poco factible [4-6].

Derechos de autor

La causa más común para que se origine este escenario es la de la utilización de algoritmos o de la aplicación, en su totalidad, por terceras partes sin el debido consentimiento del autor. Para evitar que ocurra este tipo de situaciones, el mecanismo de protección debe permitir al productor del software probar sus derechos de forma legal sobre la aplicación o algoritmo [4-6].

3 Metodologías de hardware para la protección de aplicaciones

Esta metodología surge como alternativa válida para la protección de aplicaciones. La motivación fundamental de su surgimiento se sustenta en la necesidad de la presencia de un entorno confiable donde se haga difícil la realización de ataques. En este medio serán ejecutados componentes de software pertenecientes a las aplicaciones o se almacenará algún recurso que le permita conocer a la aplicación que el dispositivo de hardware es legítimo y no ha sido suplantado. De esta forma, se trata de reducir o eliminar las posibilidades del atacante de realizar satisfactoriamente un ataque mediante análisis o modificación del código de la aplicación.

Como puede apreciarse en la figura 1, estos métodos están compuestos por tres componentes principales. El primero se refiere a la utilización de un dispositivo de hardware que garantice unicidad o exclusividad, de

modo tal que sea difícil de reproducir. El segundo componente del esquema son los segmentos especiales de código, que tienen la misión de comunicarse con el dispositivo de hardware, ya sea para chequear su presencia, almacenar datos, enviar y recibir información, etc. Mientras que el último componente está referido al canal de comunicación entre el dispositivo de hardware externo y la aplicación, el cual puede ser USB —del inglés, *Universal Serial Bus*—, PCI —del inglés, *Peripheral Component Interconnect*— o cualquier otro *bus* del sistema [8].

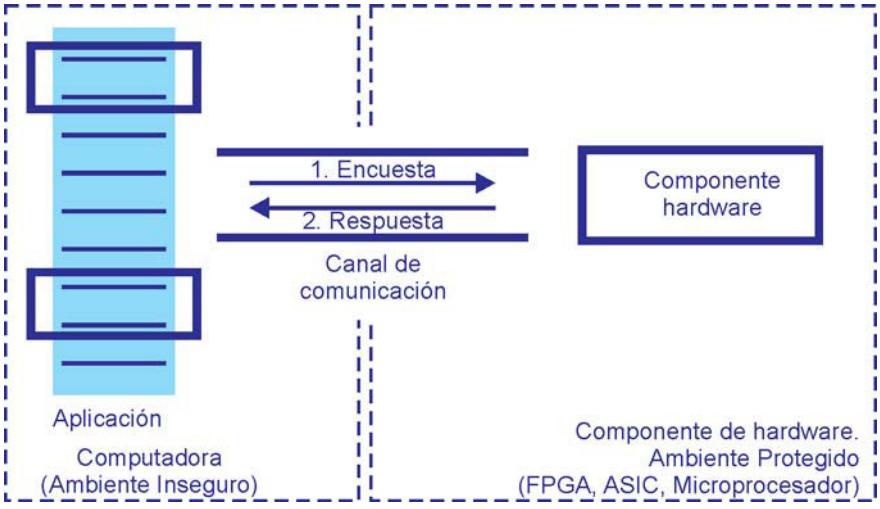

Figura 1 Componentes de un esquema de protección basado en dispositivo de hardware. (Fuente: elaboración propia).

Si bien estas metodologías mejoran ciertas deficiencias de las basadas en software, traen consigo la aparición de otras dificultades. Por ejemplo, la complicación de los esquemas de distribución de la aplicación, debido a que es necesario distribuir personalmente el dispositivo de hardware asociado a cada aplicación. Esto impide la distribución por Internet.

De los tres componentes del esquema, el más vulnerable es el canal de comunicación entre el dispositivo de hardware externo y la aplicación. Esto se debe a que la aplicación, al ser protegida a través de esquemas basados en software, resulta más compleja para hacer un ataque mediante análisis y modificación de la misma. Por su parte, el dispositivo de hardware cuenta con mecanismos de protección que dificultan la realización de cualquier tipo de ataque. Los esfuerzos principales de los atacantes están dirigidos sobre el canal. El propósito esencial es suplantar el dispositivo de hardware, lo que trae como consecuencia la anulación por completo del mecanismo.

4 Taxonomía de las metodologías de hardware para la protección de aplicaciones

En los últimos años se ha producido un incremento en el desarrollo de los métodos de protección por parte de la industria de software y de grupos de investigación. Muchos de estos métodos no obedecen o se rigen por una clasificación determinada lo que hace difícil el desarrollo de métodos posteriores a partir de las deficiencias de los que les precedieron. Apoyado en la idea anterior se requiere definir formalmente una taxonomía que agrupe los principales mecanismos de protección. La clasificación

mostrada en la figura 2 se obtuvo a partir del análisis de varios trabajos relacionados con el tema [2, 3, 5, 8-10]. Además, estos métodos se dirigen hacia el aseguramiento de uno o varios de los escenarios descritos en el punto 1.

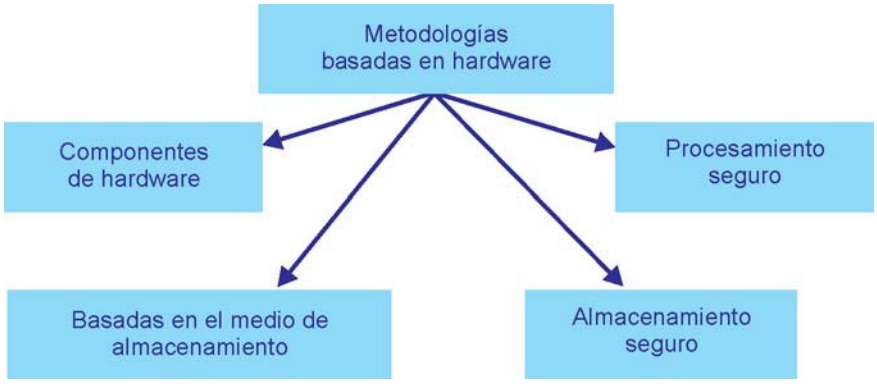

Figura 2 Principales estrategias en la protección de software. (Fuente: elaboración propia).

4.1 Basada en el medio de almacenamiento

El método se basa en la modificación o en la medición de características físicas o comportamiento específico del medio de almacenamiento —discos flexibles, CD, DVD— en que es distribuida la aplicación. De esta forma, el programa se encarga de chequear el medio y confirmar que es el original [5], [11-12]. Una de las variantes implantadas en los discos flexibles fue la creación de sectores especiales en los discos y la verificación de esos sectores al ejecutar el programa. Si el programa fuera copiado en un disco nuevo, este detectaría que no tiene los sectores especiales y, entonces, abortaría su ejecución [12].

Con el surgimiento de los discos compactos fue necesario buscar una variante de solución que permitiera proteger las aplicaciones. Una de las soluciones aplicadas a discos compactos es la creada por la compañía Laserlock, que consiste en embeber en el disco una firma digital durante el proceso de grabación. Dicha firma está formada por un conjunto de sectores corruptos con información oculta, la cual puede ser solo reconocida por el software original. Por otra parte, se le adiciona al software una rutina que será la encargada de chequear la presencia de la firma. Esta firma no puede ser copiada por grabadores de CD estándares [13].

Desde el punto de vista de seguridad, el método en general es vulnerable a ataques sobre su componente software. Esto se debe a que es posible detectar y eliminar las rutinas que hacen el chequeo del medio de almacenamiento. Por otra parte, también existen programas —LaserLock Import Fixer v0.1 y LL32ICA v1.35 Beta - Generic LaserLock Patcher— que logran duplicar la información almacenada, incluso, los sectores especiales [14].

4.2 Componente de hardware

Llamado también método de pregunta/respuesta, surge con la necesidad de encontrar un mecanismo que no involucre totalmente a la computadora donde se ejecuta la aplicación. El funcionamiento general consiste en que la aplicación realice algún tipo de consulta al dispositivo de hardware y, en dependencia de su resultado, la aplicación seguirá o no su funcionamiento.

El punto más vulnerable de este método resulta el propio código de la aplicación debido a que el atacante, al ejecutar un ataque pasivo sobre el código de la aplicación, puede determinar qué segmento de código es el encargado de establecer la comunicación con el dispositivo. Al identificar esto, realizaría un ataque activo o por modificación para, así, quitar o ignorar la comprobación [12].

4.3 Almacenamiento seguro

En este esquema se incluye, dentro del dispositivo externo, algún secreto que además de ser único, es importante para la ejecución satisfactoria de algún algoritmo. De esta forma, se mantiene la comprobación periódica para determinar si el dispositivo está conectado e invoca, cuando sea necesario, al secreto que hay almacenado. En este caso el atacante tiene que desarrollar otros métodos, pues no basta con deshabilitar la pregunta de comprobación, sino que tiene que reproducir el secreto almacenado en el dispositivo. Dentro de este esquema de protección aparece ya la criptografía como elemento importante [15-16].

Una de sus variantes de solución es que el desarrollador cifre el código de la aplicación y almacene la clave de descifrado dentro del dispositivo. En el momento en que se vaya a ejecutar el programa, se invoca un componente de software encargado de comunicarse con el dispositivo para obtener la clave. A continuación, este mismo se encarga de descifrar el código de la aplicación y, por último, se ejecuta [12]. Esta variante, a pesar de incluir elementos criptográficos, no da solución al problema de la copia de software porque, una vez que el código descifrado se encuentre en la memoria de la computadora, puede ser copiado por el atacante y generar una nueva aplicación con el código descifrado.

Existen otros tipos de solución, a partir de la utilización de la criptografía, que intentan elevar la seguridad aunque con resultados discretos. Por ejemplo, podría dividirse la aplicación en pequeñas piezas, cada una cifrada con una clave diferente, en este caso el dispositivo actuaría como repositorio seguro de las claves; de este modo, en la memoria sólo existirá, descifrada, la porción de código que se está ejecutando y el resto se descifraría en el momento de su ejecución solicitando la clave correspondiente al repositorio [12].

Desde el punto de vista del dispositivo de hardware, es usado como motor criptográfico que utiliza, generalmente, el algoritmo AES —*Advanced Encryption Standard* / Estándar de Cifrado Avanzado— [15-16]. Dicho motor es empleado para establecer un canal de comunicación seguro entre la aplicación y el dispositivo. Puede decirse que, entre las bondades que brinda el algoritmo, se encuentra la rapidez de su ejecución —no ocurre lo mismo con un algoritmo de clave pública como el RSA—; por otra parte, de acuerdo con su fortaleza también es superior siempre y cuando la clave se mantenga como un secreto. Este último requisito es bastante difícil de lograr del lado de la aplicación, porque la clave se almacenaría dentro de la computadora y puede ser recuperada por un atacante.

Las soluciones comerciales que implementan el método de almacenamiento seguro [15-16] complementan su solución con mecanismos de protección basados en software. Uno de estos mecanismos es denominado Envoltura (*Envelope*) —herramienta de protección automática desplegada en el fichero que se desea proteger—. De esta manera, no es necesario tener conocimiento del código fuente de la aplicación, debido a que esta Envoltura se adiciona al fichero como tal y no al código fuente. Una posible vulnerabilidad en este mecanismo consiste en el punto de unión entre el fichero de la aplicación y el código de protección adicionado. Esto viene dado porque, una vez anulado, se perderá el vínculo entre la llave de hardware y la aplicación. Puede ocurrir que la aplicación sea ejecutada sin la presencia del dispositivo. Con el objetivo de asegurar la permanencia de la Envoltura, se pueden encontrar implementaciones de diversos mecanismos, tales como [15-16]:

- Dividir el código de protección en diferentes capas, organizadas cada una de forma aleatoria en cada sesión de protección. Esto garantiza que para un mismo fichero la cantidad de capas y su disposición sea distinta. Las capas son cifradas de modo diferente y cada una se encarga, durante la ejecución de la aplicación, de descifrar la próxima en la secuencia usando una clave aleatoria.
- El código en cada capa es ofuscado para impedir la aplicación de técnicas de ingeniería inversa.
- Utilización de mecanismos de detección de depuración, basados en que el sistema operativo y un depurador ejecutan aplicaciones de manera diferente. Estos mecanismos están constantemente a la espera de depuradores activos, cuando es encontrado uno activo envía comandos confusos y basura para desviar la atención del depurador. El resultado de la aplicación de este mecanismo es que los depuradores activos son descubiertos y manipulados por la Envoltura.

También son usados mecanismos de protección de memoria para lectura y escritura, lo que implica que la aplicación externa sólo puede acceder a la zona de memoria que le corresponde dentro del dispositivo; en ese sentido, existirá una zona de memoria para uso exclusivo del dispositivo.

Estos elementos descritos no constituyen un patrón a emplear obligatoriamente. Existen soluciones comerciales que sólo implementan determinados mecanismos. La cantidad de mecanismos implementados podría ser directamente proporcional a la seguridad que gana la aplicación; aunque, se corre el riesgo de afectar el rendimiento de la aplicación si se hace un uso indiscriminado de los mismos.

4.4 Procesamiento seguro

Este método puede verse como una evolución lógica de los métodos anteriores porque mantiene algunas de sus características e incluye otras nuevas. El punto en común es que, en este tipo de método, la aplicación va a continuar haciendo encuestas al dispositivo. Lo nuevo que incorpora es la ejecución de cierto componente de software, perteneciente a la aplicación a proteger, dentro del dispositivo de hardware externo.

La mayoría de las soluciones que implementan este método dotan al dispositivo de hardware de un motor criptográfico. Al tener el motor criptográfico junto con la clave de cifrado, el dispositivo podría recibir los bloques de código cifrado, descifrarlos y devolverlos en este estado para que el microprocesador de la computadora pueda ejecutar las instrucciones. Esta podría ser una solución al problema de exponer las claves de cifrado a un ambiente no seguro, aunque todavía se estaría exponiendo el código en texto claro a este ambiente. Al ocurrir esto el atacante, en lugar de capturar las claves y emular el dispositivo, obtendría de la memoria los fragmentos de código [12].

Una solución al problema sería almacenar, dentro del dispositivo externo, segmentos de código involucrados en alguna lógica de negocio importante para la aplicación. Estos segmentos pueden ser almacenados en texto claro o cifrado, la principal diferencia radica en que la variante del código cifrado es más resistente a los ataques físicos descritos al

inicio. La utilización de una de estas variantes traería consigo que el atacante debe saber qué funcionalidad está implementada para poder emular el dispositivo.

En [1] los autores proponen un esquema de protección que utiliza como dispositivo externo una Smart Card (Tarjeta Inteligente) —tarjeta que contiene un procesador y memoria y que efectúa diferentes acciones—, el cual es el encargado de procesar determinados segmentos de código de la aplicación. Estos segmentos se encuentran cifrados o bien con un algoritmo asimétrico o con uno simétrico. En el cifrado asimétrico se utiliza la clave pública de la tarjeta; en el otro, la clave de cifrado simétrico se codificaría con la clave pública de la tarjeta. La diferencia de ambas variantes radica en que los algoritmos de cifrado simétrico tienen una mejor eficiencia que los asimétricos.

Por otra parte, Yee [17] afirma que la utilización del procesamiento seguro es una buena opción para la protección de una aplicación contra copias no autorizadas. En un inicio plantea la importancia de utilizar coprocesadores seguros en el desarrollo de aplicaciones seguras —verificación de la integridad de la computadora, almacenamiento de trazas, protección de copias, etc.—. En la protección de copias, refiere que es posible utilizar el coprocesador para proteger ejecutables de ser copiados y usados ilegalmente. El autor propone que el código sensible o parte de él sea distribuido y almacenado de forma cifrada y que sea ejecutado solamente dentro del coprocesador. Para cifrar el código, pueden ser empleados esquemas simétricos o asimétricos, con la peculiaridad de que si son usados los primeros, es necesario ejecutar una administración de claves con cifrado asimétrico. Además, se requiere autenticar a ambas partes para evitar la suplantación de algunas de ellas y dar al traste con la protección. Desde el punto de vista de arquitectura física, la propuesta utiliza un procesador, memoria RAM volátil y RAM respaldada por baterías, una memoria EPROM y una EEPROM encargadas de almacenar los programas de inicialización.

Smith y Weingart [18-19] documentan el desarrollo de un dispositivo de hardware que establezca un ambiente seguro y, así, poder ejecutar las aplicaciones fuera de los ambientes tradicionales. Entre sus funcionalidades se encuentran: algoritmos de cifrado asimétrico y simétrico y algoritmos de generación de números aleatorios. El dispositivo utiliza memoria RAM, un procesador, memoria RAM respaldada por baterías y memoria ROM. Estos estudios permitieron el desarrollo del coprocesador seguro 4758 por parte de IBM [20].

Otra solución, en este caso más comercial, es el producto Rockey [21-22] de la compañía Feitian. Este producto utiliza, para proteger las aplicaciones, la combinación de dos tecnologías conocidas, una de ellas es el conjunto de tecnologías básicas empleadas por los métodos anteriormente descritos y, por otra parte, usa las tecnologías de las Tarjetas Inteligentes. Su estructura interna está compuesta por un controlador de entrada/salida, un sistema operativo específico para el dispositivo, una memoria EEPROM y una RAM, y el procesador de 32 bits. Para el intercambio de mensajes con la aplicación, utiliza algoritmos de cifrado simétrico y asimétrico —*Data Encryption Standard* / Estándar de Cifrado de Datos (DES) y RSA—.

En [29-30], el procesamiento seguro no se realiza en un dispositivo de hardware externo porque plantean que, a partir de la modificación de los componentes estándares de la placa base de la computadora, puede ejecutarse el procesamiento seguro de instrucciones cifradas. El punto de partida es la distribución del código cifrado por Internet, donde la clave para descifrarlo, además de ser única, se encuentra dentro del componente de hardware. Antes de la ejecución del código, es descifrado por el circuito. Además, cuando las instrucciones son cargadas, se verifica su integridad.

Por su parte [23-24], apuntan la utilización de técnicas de ofuscación acopladas a un soporte de hardware reconfigurable. Esta solución se fundamenta en los métodos de Infraestructura de Clave Pública —del inglés, *Public Key Infrastructure* (PKI)— y de ofuscación a partir de la mutilación de código, con el propósito de hacerlo menos comprensible.

El método se basa en ubicar un dispositivo de hardware entre el nivel más alto de la memoria caché y la memoria principal. Este dispositivo es el responsable de validar la integridad de las instrucciones antes de que lleguen al procesador. Si la validación es satisfactoria, entonces las instrucciones se pasan a la caché para que el procesador las ejecute. Mediante esta técnica se garantiza la integridad de la aplicación, o sea, que no sea manipulada y modificada. En caso positivo, se vería imposibilitada su ejecución. Lo que si no se llega a evitar totalmente es que sea el microprocesador de la computadora el que ejecute la instrucción. Esto introduce un riesgo a partir del hecho de que cualquier arquitectura convencional es un medio inseguro para ejecutar cualquier aplicación, debido a que el atacante podría interceptar las instrucciones cuando sean entregadas para su ejecución al microprocesador.

Este problema se solucionaría si las instrucciones importantes se ejecutaran dentro del dispositivo de hardware y esta devolvería el resultado de la operación. Los mecanismos de protección planteados en esta técnica serían válidos también porque brindarían un mecanismo de che-

queo para saber si la instrucción fue manipulada o no.

Actualmente existe una fuerte alianza a nivel internacional, formada por un grupo de compañías líderes en el sector de la informática, que aboga por la inclusión de componentes de hardware dentro de la arquitectura de las computadoras, es decir, estandarizar un módulo que no pueda ser removido de la computadora y que sea capaz de procesar información de forma segura [9]. Las compañías involucradas son: Hewlett Packard, IBM, Intel, Compaq, Microsoft —grupo conocido como Trusted Computing Platform Alliance—. Esta organización ha publicado un conjunto de especificaciones las cuales definen un módulo de hardware que cumple con los requisitos de seguridad establecidos por las compañías miembros [25-27].

El módulo desarrollado está orientado a brindar protecciones basadas en hardware para información de autenticación y claves de cifrado sensibles a ataques. Entre las aplicaciones más importantes que se reportan de este componente, están las relacionadas con protección de claves de autenticación y protección de ficheros y sistemas de ficheros. La protección de copias no se encuentra entre sus principales aplicaciones [25-27]. La arquitectura de este módulo de hardware incluye memoria volátil y no volátil, motor para cifrado de clave pública con RSA, generador de claves y de números aleatorios, función resumen (sha-1), etc.

Vulnerabilidades

Bajo este mecanismo, los ataques asociados a la modificación del código de la aplicación no tendrán éxito. El único ataque posible que involucra el código es el análisis de los datos transmitidos desde y hacia el dispositivo de hardware. El objetivo de este ataque sería adivinar las funcionalidades que ejecuta dicho dispositivo.

Otro tipo de ataque es el de manipular físicamente el dispositivo [17, 28-29]. Se pretende obtener la información que se encuentra almacenada dentro del dispositivo de hardware. Estos pueden ser ataques mecánicos, eléctricos, a las condiciones de operación —temperatura, voltaje, nivel de radiación permisible, etc.—.

Entre los métodos descritos, este logra asegurar aún más a la aplicación, debido a que restringe la gama de posibles ataques a los relacionados con el análisis de los datos intercambiados y la manipulación física del dispositivo. No obstante, en caso de que alguno sea satisfactorio, no se tiene el modo de determinar si la aplicación realmente se está comunicando con el dispositivo correcto y no con el dispositivo manipulado o suplantado.

5 Ataques a los mecanismos de protección basados en dispositivos de hardware

Se han mencionado varios inconvenientes de estos mecanismos de protección. Como inconveniente principal, se encuentra la posibilidad de que el dispositivo sea suplantado. En esta sección se describirán los ataques a que pueden ser sometidos estos dispositivos y las consecuencias que traerían para el dispositivo de hardware y para el buen funcionamiento del mecanismo de protección. El esquema de protección basado en dispositivos de hardware, descrito en el punto 3, es susceptible a 2 tipos de ataque: ataques sobre el dispositivo de hardware y el canal de comunicación, y ataques sobre la aplicación o los segmentos especiales.

5.1 Ataques sobre el dispositivo de hardware y el canal de comunicación

En este tema, el trabajo de Rae, A. J., Wildman, L. P. [30], resulta uno de los más abarcadores. Los autores proponen una taxonomía a partir del análisis de trabajos anteriores relacionados con los ataques a dispositivos seguros. En la figura 3 se observa la taxonomía propuesta. Los ataques se encuentran agrupados según el nivel de acceso del atacante sobre el dispositivo y según su propósito.

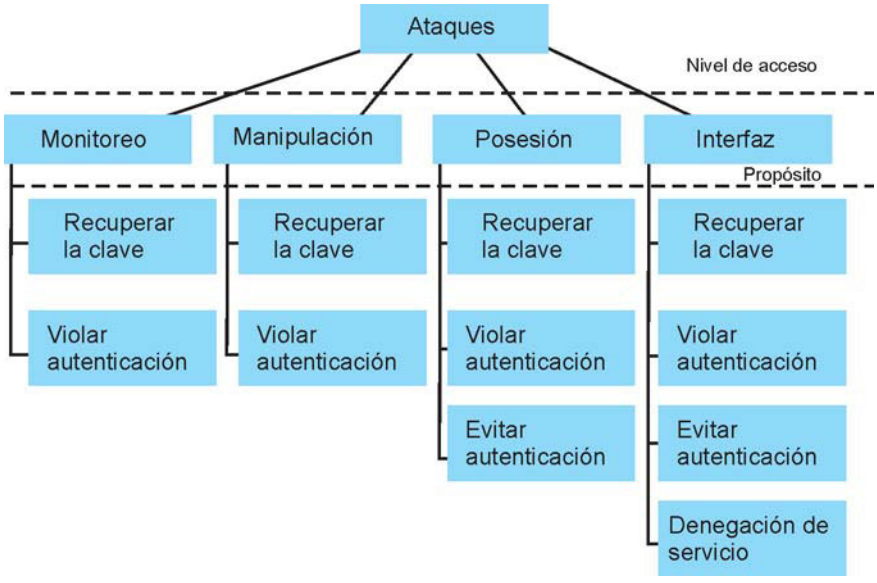

Figura 3 Taxonomía de los ataques a dispositivos seguros. (Fuente: elaboración propia).

Ataques de monitoreo al dispositivo

En este ataque, el atacante tiene la posibilidad de monitorear las características externas del dispositivo pero no puede tocarlo. Por monitoreo de las características externas se refiere al análisis de las señales electro-

magnéticas emitidas por fuentes de energía o por el propio procesador. Dentro de este ataque existen 2 tipos, determinados por el propósito que persigue el atacante: ataques para **recuperar la clave** o para **violar la autenticación**. Los primeros pueden tener 4 variantes diferentes que están dadas a partir de la observación de 4 parámetros —consumo de potencia, radiaciones electromagnéticas, análisis de tiempos y emisiones ópticas—. A estos ataques se les denominan **ataques de escucha** o **ataques de** *side-channel*. Si el propósito es **violar la autenticación,** se utilizan **ataques de radiofrecuencia**.

Ataques con manipulación del dispositivo

En estos se manipula el ambiente del dispositivo incluyendo el establecimiento de valores de entrada y monitoreo de las señales de salida. No contiene abrir el dispositivo para modificar su funcionamiento. Para **recuperar la clave** se utilizan los denominados ataques por análisis de fallas o los de análisis diferencial de fallas, que consisten en la generación de fallas transitorias al dispositivo. Si se pretende **violar la autenticación**, es posible ejecutar ataques dirigidos a las variables ambientales del dispositivo —temperatura, radiación, etc.—.

Ataques con posesión del dispositivo

El atacante tiene la capacidad de manipular completamente el dispositivo, incluso, modificarlo o someterlo a exploraciones sofisticadas. Esta manipulación puede hacerse manteniendo el dispositivo en funcionamiento o destruyéndolo para comprender su funcionamiento. Las alternativas de este ataque para **recuperar la clave** comprenden la observación de la información durante la ejecución de las funciones dentro del dispositivo, el análisis del microprocesador a partir de un proceso de ingeniería inversa y la destrucción de zonas del circuito asociadas a la implementación de un determinado algoritmo.

Para dar al traste con la autenticación puede realizarse un ataque que sobrescriba una determinada zona en la memoria del dispositivo, ya sea para reemplazar la clave o para modificar la ejecución de los algoritmos. En [28- 29] se expone un ataque de esta índole sobre soluciones comerciales, en este caso el eToken y el iKey 1000. Para **evitar la autenticación** se emplean ataques a partir de la utilización de técnicas avanzadas que permiten poner el *chip* en modo de prueba y, de esta manera, descargar su contenido.

Ataques sobre la interfaz del dispositivo

Estos ataques no se enfocan en el dispositivo, sino que están dirigidos sobre los protocolos que el dispositivo usa para comunicarse con el mundo exterior. Es posible **recuperar la clave** a partir de la realización de criptoanálisis sobre los paquetes de información intercambiados. Para **violar la autenticación**, el ataque del hombre en el medio juega un papel importante, el cual se materializa en ataques por repetición y ataques dirigidos a modificar la estructura interna del mensaje.

Los ataques para **evitar la autenticación** son específicos de la implementación, porque estos se basan en la explotación de errores en el diseño o configuración del dispositivo. Su variante más difundida son los ataques por desbordamiento de *buffer*, donde el atacante es capaz de usar cierta información en la entrada para causar operaciones inseguras del dispositivo. Otra variante es la del canal secreto, mecanismo a través del cual la información puede ser deliberadamente comunicada entre partes de un sistema que se encuentran en diferentes niveles de seguridad. Por último, los ataques semánticos y por fuerza bruta provocarían una **denegación de servicios** por parte del dispositivo. En el caso de los ataques semánticos, el atacante envía paquetes o mensajes estructurados incorrectamente hacia el dispositivo para que se consuman sus recursos. Por su parte, los ataques por fuerza bruta consisten en enviar grandes cantidades de mensajes correctamente formados.

5.2 Ataques sobre la aplicación o los segmentos especiales

Los segmentos especiales de código —tercer componente— también se incluyen dentro de los objetivos de un atacante. Los ataques sobre este componente se clasifican como ataques pasivos y ataques activos. Estos pretenden anular los segmentos especiales de código para evitar la comunicación con el dispositivo de hardware externo. Los ataques pasivos están dirigidos contra la privacidad de la aplicación, o sea, para revelar sus secretos. Dentro de ellos, se encuentra el análisis de la aplicación mediante un proceso de ingeniería inversa. Este ataque, el adversario puede ejecutarlo de forma estática a partir del análisis de los datos y de la aplicación cuando no se está efectuando; o de forma dinámica, que implica que el adversario ejecute la aplicación para analizar el flujo de control y los valores de las variables del programa. Otro ejemplo consiste en el análisis diferencial del software, donde son examinadas diferentes versiones de una aplicación con el objetivo de identificar qué partes han cambiado [3].

Los ataques activos van dirigidos a modificar el código de la aplicación, es decir, atentan contra la integridad de la aplicación. Al igual que el ataque por análisis, puede efectuarse de forma estática y dinámica. La primera se refiere a la modificación del código de la aplicación cuando no se está cumplimentando, mientras que la variante dinámica implica que se modifique el código de la aplicación cuando se está ejecutando [3].

Luego de haber analizado los principales ataques a que pueden ser sometidos los componentes de un esquema de protección basado en hardware, deben destacarse varias cuestiones: los ataques sobre los dispositivos seguros exigen del atacante cierto nivel de especialización y tecnológico en cuanto a los tipos de herramientas utilizadas. Por lo que se ve limitada la cantidad de personas o empresas capaces de ejecutarlos. Las ideas expuestas presuponen un elevado gasto en tiempo y dinero para realizar un ataque satisfactorio sobre una determinada aplicación.

Los ataques sobre los segmentos especiales de código y el canal de comunicación no exigen un gasto tecnológico elevado, aunque sí un elevado consumo de tiempo. En los primeros el gasto aumentaría si el software es protegido por algunos de los mecanismos basados en software. En ambos casos, existen herramientas y documentación de los procedimientos que facilitan, en cierta medida, la ejecución de ataques satisfactorios sobre los 2 componentes.

6 Conclusiones

Al concluir este trabajo es posible asegurar que los métodos de protección basados en dispositivos de hardware garantizan un nivel de seguridad superior a sus antecesores basados en software. El empleo satisfactorio de este tipo de estrategias implica la protección de sus tres componentes —la integridad de la aplicación, el canal de comunicación entre el software y el hardware, y la integridad del dispositivo de hardware—.

De los métodos analizados, se aprecia que los mecanismos de autenticación empleados no evitan que el dispositivo de hardware sea emulado, lo cual elimina el principio de unicidad del método y, con ello, facilita su violación.

7 Referencias bibliográficas

[1] Maña, A. and Pimentel, E. "An Efficient Software Protection Scheme" in Proceedings of the IFIP TC11 Sixteenth Annual Working Conference on Information Security, Deventer, The Netherlands, The Netherlands, 2001, pp. 385-402.

[2] Van Oorschot, Paul C. "Revisiting Software Protection". Paper presented at the Proceedings of the 6th International Information Security Conference, Bristol, United Kingdom, 2003, pp. 1-13.

[3] Main, A. and Van Oorschot, P.C. "Software Protection and Application Security: Understanding the Battleground". http://www.scs.carleton.ca/~paulv/papers/pubs.html. (acceso abril, 2008).

[4] Lee, B. and Kim, K. ""Software Protection Using Public Key Infraestructure". Paper presented at the Symposium on Cryptography and Information Security, Kobe, Japan, 1999.

[5] Cronin, G. *A Taxonomy of Methods for Software Piracy Prevention*. New Zealand: Department of Computer Science, University of Auckland, 2002.

[6] Goldreich, O. "Towards a Theory of Software Protection and Simulation by Oblivious RAMs". Paper presented at the STOC '87 Proceedings of the Nineteenth Annual ACM Conference on Theory of Computing, New York, United States, 1987.

[7] *Fourth Annual BSA and IDC Global Software Piracy Study*. (2007). http://w3.bsa.org/globalstudy//upload/2007-Global-Piracy-Study-EN.pdf. (acceso noviembre, 2007).

[8] Gosler, James R. "Software Protection: Myth or Reality?" Lecture notes in Computer Sciences, 218 on Advances in Cryptology, CRYPTO 85, Springer-Verlag New York, Inc., New York, NY, USA, 1985, pp. 140-157.

[9] Hachez, G. A. "Comparative Study of Software Protection Tools Suited for E-Commerce with Contributions to Software Watermarking and Smart Cards". Tesis de Doctorado, Universidad Católica de Louvain, Bélgica, marzo, 2003.

[10] Chen, M. "Software Product Protection". Paper presented at the Seminar on Network Security, University of Technology, Helsinki, Finland, 2001.

[11] Albanese, J. and Sonnenreich, W. *Network Security Illustrated*. New York: McGraw-Hill, 2004, pp. 398.

[12] Eilam, E. *Reversing, Secrets of Reverse Engineering*. Indianapolis, Indiana: Wiley Publishing, Inc., 2005, pág. 589.

[13] "LaserLock Product Features". http://www.laserlock.com/product_features.html. (acceso junio, 2007).
[14] "Estudio colectivo de desprotecciones". (octubre/2001). http://ciberia.ya.com/keyviboro/ECD/laserlok5.htm. (acceso junio, 2007).
[15] *Sentinel™ UltraPro™ 1.0 Developer's Guide*. Baltimore, USA: SafeNet, Inc., 2004. http://www.safenetinc.com/products/sentinel/ultraPro.asp. (acceso abril, 2007).
[16] "HASP HL". http://www.aladdin.com/HASP/HaspHL.asp. (acceso noviembre, 2006).
[17] "Sentinel UltraPro". http://www.safenetinc.com/products/sentinel/ultraPro.asp. (acceso abril, 2007).
[17] Yee, B. "Using Secure Coprocessors". Tesis de Doctorado, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 1994.
[18] Smith, S.W. and Weingart, S. "Building a High-Performance, Programmable Secure Coprocessor". *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 31, no. 9 (April 23/1999): 831-860.
[19] Smith, S.W., Palmer, E.R. and Weingart, S. "Using a High-Performance, Programmable Secure Coprocessor". Paper presented at the Second International Conference on Financial Cryptography, FC'98, Anguilla, British West Indies, 1998.
[20] Dyer, J.G., Lindemann, M., Perez, R., Sailer, R., v. Doorn, L., Smith, S.W., and Weingart, S. "Building the IBM 4758 Secure Coprocessor". *Computer,* vol. 34 no.10 (October/1991): 57-66.
[21] "ROCKEY6 Smart". http://www.rockey.com.my/prod-dongle-rockey6.php. (acceso abril, 2007).
[22] "ROCKEY6 User Manual". (18/11/2005). http://www.rockey.com.my/dl_dongle_rockey6.php. (acceso abril, 2007).
[23] Joseph, Z., Alok, C., Rahul, S., and Bhagirath, N. "Flexible Software Protection Using Hardware/Software Codesign Techniques". Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1, IEEE Computer Society, 2004.
[24] Zambreno, J., Honbo, D., Choudhary, A., Simha, R., and B. Narahari. "High-Performance Software Protection Using Reconfigurable Architectures". *Proceedings of the IEEE*, vol. 94, no. 2 (2006): 419-431.
[25] Berger, B. "Trusted Computing Group History". *Information Security Technical Report*, vol. 10 no. 2 (2005): 59-62.
[26] Kallath, D. "Trust in Trusted Computing - the End of Security as We Know It. *Computer Fraud & Security*, no. 1 (enero/2005): 4-7.
[27] Mason, S. "Trusting Your Computer to Be Trusted". *Computer Fraud & Security*, no. 12 (diciembre/2005): 7-11.
[28] Grand, J. "Attacks on and Countermeasures for USB Hardware Token Devices". Paper presented at the Proceedings of the Fifth Nordic Workshop on Secure IT Systems Encouraging Co-operation, Reykjavik, Iceland, October 12-13, 2000.
[20] Grand, Joe. "Advanced Hardware Hacking Techniques". Presented in DEFCON® Hacking Conference, DEFCON 12, Las Vegas, NV, USA, Julio 30-August 1, 2004. http://www.defcon.org/images/defcon-12/dc-12-presentations/Grand/grand_advanced_hardware_hacking_DC12_handouts.pdf. (acceso abril, 2007).
[30] Rae, A.J. and Wildman, L.P. "A Taxonomy of Attacks on Secure Devices". Paper presented at the Proceedings of the Fourth Australian Information Warfare and IT Security Conference, University of South Australia, Adelaide, Australia, 2003, pp. 251-263.