ieee.es
Instituto Español de Estudios Estratégicos

Documento
Opinión

88/2014 11 de agosto de 2014

José Tomás Hidalgo Tarrero *

INGENIERÍA DEL SOFTWARE Y CIBERSEGURIDAD

INGENIERÍA DEL SOFTWARE Y CIBERSEGURIDAD

Resumen:

Hay una tendencia a pensar que la ciberseguridad es solo añadir subsistemas de hardware y software a nuestro sistema informático. También hay una tendencia a menospreciar la ingeniería del software, hay quien ha escrito que la ingeniería del software no debería ser ingeniería; sin embargo, desde finales de los 60, exactamente tras una conferencia sobre ingeniería del software patrocinada por la OTAN celebrada entre los días 7 y 11 de octubre de 1968 en Garmisch, Alemania, mucho se ha escrito sobre las técnicas y procedimientos de la ingeniería del software, las cuales se aplican de manera más o menos generalizada y completa.

Para demostrar la relación entre ingeniería del software y ciberseguridad se ha seguido un proceso lógico. Primero se demuestra la necesidad de la ingeniería del software mediante la demostración de la falsedad de algunos de los mitos más perniciosos sobre el desarrollo del software. El segundo paso consiste en demostrar que la ingeniería del software es necesaria para desarrollar software de calidad. En el tercero se establece la relación entre software de calidad y software seguro. Finalmente se determina la relación entre el software seguro y la ciberseguridad con lo que la relación entre la ingeniería del software y la ciberseguridad queda establecida

Por último se analiza someramente el impacto del factor humano en la ingeniería del software y, por ende, en la ciberseguridad.

Abstract:

There is a tendency to think that cybersecurity is just to add hardware subsystems and software to our computer system. There is also a tendency to belittle software engineering, some have written that software engineering should not be an engineering, however, since the late 60s, just after a conference on software engineering sponsored by NATO held between the 7th and 11th October 1968 in Garmisch, Germany, much has been written about the techniques and procedures of software engineering which are applied in a more or less generalized and complete way.

To demonstrate the relationship between software engineering and cybersecurity a logical process has been followed. First it demonstrates the need for software engineering by demonstrating the falsity of some of the most pernicious myths about software development. The second step is to show that software engineering is required to develop quality software. In the third the relationship between quality software and secure software is established. Finally, the relation between secure software and cybersecurity is determined so that the relationship between software engineering and cybersecurity is established.

Finally the impact of human factors in software engineering and therefore in cybersecurity is briefly discussed.

Palabras clave:

Ciberseguridad. Software. Ingeniería. Factor Humano.

Keywords:

Cybersafety. Software. Engineering. Human factor.

POR QUÉ INGENIERÍA DEL SOFTWARE

Para entender el porqué de la ingeniería del software primero es necesario ver alguno de los muchos mitos que hay sobre el software y su desarrollo; de entre todos ellos he elegido los cinco señalados por Cerrada[1] porque se pueden considerar de los que más daño han hecho, y siguen haciendo, al desarrollo del software de calidad y seguro.

1. El hardware es mucho más importante que el software.

   Falso. Supongamos que el coste computacional de un programa es exponencial; es decir, si N es el número de "datos" que debe procesar, el "tiempo" que tarda en ejecutarse $T = \theta(2^N)$. θ Quiere decir "del orden de".

   Supongamos ahora que duplicamos la "potencia" del ordenador, el nuevo "tiempo" sería $T_1 = \theta(2^N/2) = \theta(2^{N-1})$ que sigue siendo muy alto, del mismo orden que T.

   Supongamos ahora que reescribimos el programa para que, haciendo exactamente lo mismo, el coste computacional sea cuadrático; es decir que el nuevo "tiempo" $T_2 = \theta(N^2)$; unas sencillas operaciones matemáticas demuestran que T_2 es muchos órdenes de magnitud menor que T_1 a partir de valores de N iguales o superiores a 4, siendo 10^{3002} veces menor para N = 10000.

   Si pudiéramos hacer el coste computacional lineal o, mejor aún, logarítmico la ganancia sería muchísimo mayor.

   Con lo anterior se demuestra matemáticamente que la importancia del software es mucho mayor que la del hardware.

   Además, como muy bien dice Cerrada[2], el usuario no interacciona directamente con el hardware sino que lo hace a través del software.

2. Es fácil desarrollar software.

   Cualquier aplicación consta de un cierto número de módulos, que se llaman unos a otros y a sí mismos, esto último se llama recurrencia, los cuales, a su vez, están compuestos de funciones y procedimientos. Conforme aumenta la complejidad de la aplicación aumenta el número de módulos. Todo esto hay que desarrollarlo de manera que sea fácil probar la aplicación antes de que sea entregada al usuario; además, una vez que la aplicación esté implantada y funcionando, debe ser posible hacer el mantenimiento de dicha aplicación de

[1] **Cerrada Somolinos, José A., Collado Machuca, Manuel E., Gómez Palomo, Sebatián R. y Estívariz López, José F.** *Introducción a la ingeniería del software.* Madrid : Editorial universitaria Ramón Areces, 2007, 9-10.
[2] **Ibid**.

manera fácil y sencilla; de lo contrario, será muy difícil, o incluso en la práctica imposible, corregir algún error no detectado en las pruebas ni mejorarla añadiendo nuevas funcionalidades. También tiene que ser posible añadir nuevas funciones para tratar los mismos datos de entrada e incorporar datos que no estaban previstos. Todos los módulos deben estar encapsulados de tal manera que una modificación hecha en uno de ellos no implique tener que corregir los demás.

Como se ha visto anteriormente, la influencia del software en la eficiencia del sistema es mucho mayor que la del hardware; además el desarrollo de algoritmos más eficientes requiere conocimientos específicos de algoritmia, estructuras de datos y matemáticas, entre otras materias, que no hacen la tarea fácil.

3. El software sólo es un conjunto de programas ejecutables.

   Un sistema informático es un conjunto de hardware, software y personas. Software no son solo los programas ejecutables, son también los procedimientos que deben emplear las personas para utilizar ese sistema; es también toda la documentación del sistema. El diseño del sistema es parte del software porque debe diseñarse teniendo en cuenta el hardware y las personas que lo van a usar, de lo contrario ese software será, en el mejor de los casos, poco eficiente cuando no totalmente inútil.

4. El desarrollo del software es solo cuestión de programar.

   La programación o codificación es solo una de las fases, y desde luego no es la de mayor duración ni la de mayor coste, del ciclo de vida del software que, según Cerrada[3] o Pressman[4], entre otros, y de manera resumida, son: ingeniería de sistemas, análisis del software, diseño, codificación, pruebas de unidades, integración, pruebas del sistema, mantenimiento y baja del sistema. Como se ve el desarrollo del software no es cuestión baladí sino que, por su complejidad, requiere técnicas y procedimientos propios de una ingeniería que es la ingeniería informática, una de cuyas partes es la ingeniería del software.

5. Es normal que el software tenga errores.

   Ciertamente el desarrollo del software, como cualquier otra actividad humana, puede tener errores, pero no es admisible que todos los productos software tengan errores. Si compramos cualquier hardware, desde un

[3] **Cerrada et alt.** *Introducción a la ingeniería del software.* Madrid : Editorial universitaria Ramón Areces, 2007, 10-16.
[4] **Pressman, Roger S.** *Ingeniería del software. Un enfoque práctico,* segunda edición. McGraw-Hill, 1992, 23-24

ordenador a una lavadora, y tiene fallos, reclamamos y exigimos el cambio. En el caso del software, los errores se propagan durante toda la fase de producción de copias del mismo ya que esos errores tienen su origen no en el proceso de copia, fabricación en el caso del hardware, sino en etapas anteriores, fundamentalmente en las etapas iniciales hasta la codificación inclusive, por lo que hay que reestudiar todo el software para descubrir el error; además, al corregir estos errores es fácil introducir nuevos errores. Los hackers se benefician de esos errores para atacar los sistemas informáticos.

Para combatir los errores propiciados por esos mitos y otros muchos y por el imparable aumento de la complejidad del hardware y del software es necesario aplicar al desarrollo del software técnicas propias de la ingeniería; esto condujo en la década de los 70, aunque la primera evidencia es una conferencia sobre ingeniería del software patrocinada por la OTAN celebrada entre los días 7 y 11 de octubre de 1968 en Garmisch, Alemania, a la creación de una nueva especialidad de la ingeniería llamada ingeniería informática, una de cuyas partes es la ingeniería del software.

- Calidad del software y seguridad del software.

  La principal tendencia en ciberseguridad es poner *artefactos* para proteger los sistemas informáticos; no está de moda el hacer software seguro. El problema es que esos *artefactos* tienen dentro software que está sujeto a ataques y si ese software no es seguro será mucho más fácil para el atacante romper la barrera y penetrar en el sistema.

  En la ingeniería del software se establecen una serie de factores de calidad del software, aunque no todos los investigadores están de acuerdo en todos ellos. Todos los factores de calidad citados por Pressman[5], Meyer[6] o Cerrada[7], entre otros, tienen relación con la ciberseguridad, pero de entre todos ellos, los de robustez, fiabilidad, seguridad, facilidad de uso, mantenibilidad y facilidad de prueba tienen una relación más directa.

  Para Howard y LeBlanc[8], una de las primeras líneas de defensa es escribir software robusto. Desde la década de los 80 se conoce un tipo de vulnerabilidad conocida como "buffer overrun"; según los mismos autores[9] la prevención de esa vulnerabilidad es principalmente una cuestión de escribir software robusto. El software robusto es capaz de funcionar correctamente incluso en situaciones no

[5] **Pressman, Roger S.** *Ingeniería del software. Un enfoque práctico,* segunda edición. McGraw-Hill, 1992, 481-482.
[6] **Meyer, Bertrand.** *Object-oriented Software Construction.* Prentice Hall International, 1988, 4-7.
[7] **Cerrada et alt.** *Introducción a la ingeniería del software.* Madrid : Editorial universitaria Ramón Areces, 2007, 27-28.
[8] **Howard, Michael y LeBlanc, David.** *Writing Secure Code,* segunda edición. Microsoft Press, 2003, 127.
[9] **Howard, Michael y LeBlanc, David.** *Writing Secure Code,* segunda edición. Microsoft Press, 2003, 155.

previstas y anormales[10]; por esta razón la prevención del "buffer overrun" y de otras muchas vulnerabilidades es escribir software robusto. Esto también es apuntado por Allen[11] y por otros autores desde hace tiempo. Alguna de las razones por las que no se utilizan técnicas, de sobra conocidas, para evitar el "buffer overrun" están implícitas en Allen[12]y de forma bastante explícita en Howard[13]; uno de los mejores remedios para paliar esas causas es la concienciación tal y como se explica en la monografía del CESEDEN nº 137.

Obviamente el factor seguridad, también llamado integridad, es el más importante de cara a obtener software seguro; si cualquiera, o cualquier módulo, puede acceder a cualquier parte del software, incluyendo el código, datos, documentación, etc., sin ninguna dificultad, evidentemente todo lo demás sobra.

La facilidad de uso tiene su importancia porque si el software es difícil de usar es más probable que el mismo usuario cometa fallos de uso que crearán brechas de seguridad. Hay que tener presente que la mayoría de los ataques informáticos empiezan con una operación de ingeniería social que aprovecha que el factor humano es el eslabón débil de la cadena de seguridad informática y es el que más fácilmente puede cometer errores que provocan brechas de seguridad.

La fiabilidad es crucial; el software debe hacer lo que está en las especificaciones, todo lo que está en las especificaciones y nada más que lo que está en las especificaciones. Obviamente si el software no hace todo lo que dicen las especificaciones, el software no será todo lo útil que debería y será rechazado.

No es infrecuente encontrar en el software funcionalidades no descritas y que no están en las especificaciones; buena parte de ellas son puertas traseras, *backdoors* en inglés, hechas por el desarrollador de esa parte del software para facilitar la depuración y, posteriormente, el mantenimiento y que olvidó eliminar. Esas *backdoors* permiten a un atacante entrar en el sistema y alcanzar privilegios de administrador para luego hacer lo tenga por conveniente para sus intereses.

Esto nos lleva a la importancia de las especificaciones: deben hacerse teniendo en cuenta también la seguridad. Como muy bien explica Allen[14] la seguridad no es un añadido, los cortafuegos, la protección mediante palabra clave, las herramientas de detección de virus u otras parecidas no son requisitos de sistema seguro sino que son

[10] **Meyer, Bertrand.** *Object-oriented Software Construction.* Prentice Hall International, 1988, 4.
[11] **Allen, Julia H; Barnum, Sean; Ellison, Robert J.; McGraw, Gary; y Mead, Nancy R.** *Software Security Engineering: A Guide for Project Managers,* Addison-Wesley Professional, 1st edition, 2008, xi.
[12] **Allen et alt.** *Software Security Engineering: A Guide for Project Managers,* Addison-Wesley Professional, 1st edition, 2008, xii.
[13] **Howard, Michael y LeBlanc, David.** *Writing Secure Code,* segunda edición. Microsoft Press, 2003, 23.
[14] **Allen et alt.** *Software Security Engineering: A Guide for Project Managers,* Addison-Wesley Professional, 1st edition, 2008, 73-78.

mecanismos implementados para satisfacer requisitos no especificados; debe considerarse como un defecto el no introducir bien los requisitos de seguridad en la especificaciones pues los costes añadidos por no hacerlo, según Allen[15], son muy altos.

Es importante no cambiar las especificaciones durante el desarrollo, hay que fijarlas antes de empezar la siguiente fase; si durante el desarrollo, cuando se van mostrando los diversos prototipos, en una metodología de desarrollo en espiral, surge la necesidad de incorporar nuevas funciones o tratar datos distintos de los inicialmente previstos, estas deben tratarse como añadidos y seguir todas las fases de la ingeniería del software ya nombradas. La seguridad debe seguir siendo parte integral de este proceso.

Según Allen[16], especificar los requisitos de seguridad es difícil porque hay que pensar en lo impensable, en situaciones anormales. Esta dificultad en la especificación se traslada al diseño de las pruebas de seguridad. Cuando las especificaciones están muy bien hechas esta dificultad es menor. Además, si el software tiene una gran facilidad de prueba será más fácil encontrar vulnerabilidades durante las mismas y, si el software es muy mantenible, es decir que tiene facilidad para ser mantenido, que es otro de los factores de calidad mencionados, dichas vulnerabilidades podrán ser eliminadas antes de que el producto software llegue al usuario.

La mantenibilidad nos permitirá corregir fácilmente errores y vulnerabilidades no detectadas en la fase de pruebas y que surgen a lo largo de la vida del software. Cuanto menos tiempo se tarde en corregir esos errores y vulnerabilidades menos tiempo tendrán los atacantes para sacar provecho de las mismas.

Para certificar la seguridad del software existen unos Common Criteria (Criterios Comunes) que establecen siete niveles de los cuales solo los cuatro primeros están realmente aceptados por todas las naciones adheridas, mientras que los tres últimos son implementación nacional. En septiembre de 2012 una mayoría de los miembros del Common Criteria Recognition Arrangement (CCRA) se pusieron de acuerdo para rebajar el mutuo reconocimiento a los dos primeros niveles. Estos criterios comunes han tenido críticas importantes, en este sentido es necesario recordar un artículo de William Jackson[17] en el que, basado en entrevistas a ingenieros y gestores de sistemas informáticos, ponía de relieve que los Common Criteria realmente no aseguran que el software sea seguro, sino que en el proceso de desarrollo se han seguido unas determinadas pautas. Además, los Common Criteria no evalúan todo el

[15] **Ibid**.
[16] **Ibid**.
[17] **William Jackson**. Under Attack: Common Criteria has loads of critics, but is it getting a bum rap? By William Jackson, http://gcn.com/Articles/2007/08/10/Under-attack.aspx?p=1. Consultado el 17/12/2013

sistema, sino las partes que el fabricante quiere y de acuerdo con los escenarios y las amenazas que quiere el fabricante. No obstante, los Common Criteria es lo único que hay acordado a nivel internacional para evaluar el software; aunque no sean todo lo buenos que sería deseable, al menos dan una cierta indicación de la seguridad del software. Es necesario siempre leer bien toda la documentación relativa a la evaluación del mismo para saber en qué nivel, en qué entorno y ante qué amenazas ha sido certificado.

Con todo lo anterior de lo que se trata es de proteger el software desde dentro mismo sin renunciar a las líneas de defensa externas, que también tienen mucho software dentro, el cual, a su vez, debe protegerse desde dentro. Esto suena, y mucho, a proceso recursivo.

Siempre hay que tener en cuenta que no se puede alcanzar la seguridad absoluta y que en informática, como en la guerra, es necesario establecer varias líneas de defensa por lo que no es necesario elegir entre proteger el software desde fuera o desde dentro; por el contrario, hay que protegerle desde dentro y desde fuera en todas las líneas de protección. En los sistemas existentes, muchos de los cuales tienen software diseñado sin pensar en la seguridad, implementar la seguridad desde dentro puede ser entre muy difícil e imposible, por lo que habrá que pensar sólo en protección externa.

Otro aspecto importante en el diseño del sistema informático es el principio del privilegio mínimo, que también se aplica en otras áreas de la seguridad. Cada proceso, cada función y procedimiento debe funcionar con el privilegio mínimo para ejecutar la tarea que debe hacer. Esto reducirá la ventana de posibilidad para la explotación de vulnerabilidades[18].

SOFTWARE SEGURO Y CIBERSEGURIDAD

Para hacer software seguro, que tal y como hemos visto es forzosamente software de calidad, es necesario, en todas las fases del desarrollo, tener presentes dos puntos de vista fundamentales en la ingeniería del software seguro: los usuarios y los atacantes. Los primeros focalizan su atención en la funcionalidad del sistema y asumen que el sistema es seguro; los segundos, excepto en algunos casos muy específicos, focalizan su atención en las características de seguridad del sistema y asumen que el sistema tiene cierta funcionalidad[19].

[18] **Howard et alt.** *Writing Secure Code,* segunda edición. Microsoft Press, 2003, 207.
[19] **Allen et alt.** *Software Security Engineering: A Guide for Project Managers,* Addison-Wesley Professional, 1st edition, 2008, 73-78.

La ciberseguridad, al menos en teoría, defiende nuestros sistemas y reacciona contra los atacantes. Para defender nuestros sistemas necesita promocionar el software seguro y, como hemos visto, para hacer software seguro se necesitan los puntos de vista del atacante y del usuario. Para reaccionar contra los atacantes es necesario también, como en cualquier batalla, tener en cuenta tanto el punto de vista del atacante como del defensor que en un sistema informático sería el usuario. Como vemos, en ambos casos se necesitan los dos puntos de vista.

También hemos visto que aplicando en el desarrollo del software las técnicas de ingeniería del software de manera integral, es decir, incluyendo la seguridad del sistema desde el principio, reduciremos las vulnerabilidades del sistema.

Se debe suponer que el atacante está dentro de un sistema que tiene software seguro, la presunción de que el atacante no ha hecho sus deberes puede calificarse de suicida tal y como demuestra la historia de los conflictos humanos, por lo que tener el conocimiento de cómo hacer software seguro nos dará una cierta ventaja o, al menos, nos pondrá al nivel del atacante. También se debe suponer, por las mismas razones, que el atacante está familiarizado con el software seguro y conoce técnicas de ataque sofisticadas.

Si los gestores e ingenieros han sido previsores y el sistema informático está basado en software seguro, desarrollado siguiendo los principios de ingeniería del software seguro, el atacante se encontrará con una defensa bien integrada y con muchas menos vulnerabilidades de lo que es habitual hoy en día. Solo con esto la ciberseguridad ya ha conseguido una cierta ventaja.

No podemos eliminar los ataques, pero usando software seguro y de calidad nuestro sistema podrá soportar mucho mejor los ataques, factor de calidad robustez que implica que el sistema funcionará bien incluso en situaciones anormales y no previstas; será más difícil que escale privilegios, principio del mínimo privilegio que implica que el uso de los privilegios siempre estará muy limitado en el tiempo y en la cantidad, con lo que será más difícil que pueda obtener información o usar recursos del sistema.

Introducir desde el principio la seguridad reduce vulnerabilidades intrínsecas del sistema, pero hay que recordar una frase célebre de Albert Einstein: "*Every day, man is making bigger and better fool-proof things, and every day, nature is making bigger and better fools. So far, I think nature is winning*". ("Cada día, el hombre está haciendo cosas más grandes y mejores a prueba de tontos, y cada día, la naturaleza está haciendo tontos más grandes y mejores. Hasta el momento, creo que la naturaleza está ganando"). Esta frase nos recuerda la importancia del factor humano en la ingeniería del software seguro y en la seguridad de los sistemas informáticos.

FACTOR HUMANO EN LA INGENIERÍA DEL SOFTWARE SEGURO

El factor humano en la ingeniería del software hay considerarlo en los siguientes aspectos: los gerentes de las empresas de desarrollo, los ingenieros y personal de desarrollo, los gerentes de las empresas usuarias y los usuarios propiamente dichos.

Entre todos hacen una especie de círculo vicioso que actúa, normalmente, en contra de los principios de la ingeniería del software seguro.

Los gerentes de las empresas usuarias y los usuarios propiamente dichos presionan a los gerentes de las empresas de desarrollo de software para que el producto que hacen sea más barato, se desarrolle en menos tiempo y que la seguridad, si existe o se trata de alguna manera, sea transparente. Los gerentes de las empresas de desarrollo presionan a los ingenieros y personal de desarrollo para que el software se haga deprisa, aunque haya que saltarse fases y las pruebas puedan ser sumarias y no completas; a menudo el cliente cambia de parecer durante el desarrollo y modifica los requisitos, y por tanto las especificaciones, pero no los plazos ni el precio final. En el caso del software más comercial la presión de los usuarios finales por tener nuevas versiones más baratas, más pronto y con más funcionalidades es similar a la descrita para el software hecho a medida.

Los usuarios finales, además, suelen tener la tendencia de no leer los manuales de uso del software con lo que antes o después, más bien antes, usan el software de manera inadecuada saltándose indicaciones o normas de seguridad que pueden estar incluidas en los manuales. Einstein parece que tenía razón también con la frase *ut supra*.

La mejor forma de combatir esta especie de círculo vicioso es la concienciación. Para aquellos interesados en la concienciación de ciberseguridad, la monografía nº 137 del CESEDEN, dedicada a la conciencia de ciberseguridad, es un documento digno de leer.

i

*José Tomás Hidalgo Tarrero**
Coronel E.A.
Profesor de la EALEDE

BIBLIOGRAFÍA

Garfinkel, Simson y Spafford, Gene. *Practical UNIX & Internet Security,* Second Edition, April 1996.

Gómez Bule, Juan Antonio; Roldán Tudela, José Manuel; Jiménez Muñoz, Luis; Pastor Acosta, Oscar; Sánchez de Rojas Díaz, Emilio; Hidalgo Tarrero, José Tomás. *Necesidad de una conciencia nacional de ciberseguridad. La ciberdefensa: un reto prioritario. Monografía 137 de CESEDEN,* septiembre 2013.

Common Criteria Recognition Arrangement (CCRA). *Common Criteria for Information Technology Security Evaluation (CC) versión 3, edición 4*.

Common Criteria Recognition Arrangement (CCRA). *Common Methodology for Information Technology Security Evaluation (CEM) versión 3, edición 4*.