Principios básicos del desarrollo seguro

Pablo García Pérez

Ingeniero de Proyecto. División de Seguridad Lógica de Germinus

Para comenzar habría que contestar a la pregunta ¿qué es desarrollo seguro? Cuando hablamos de desarrollo seguro nos referimos a la realización o implementación de aplicaciones de todo tipo (lo que quiere decir que no nos referimos solamente a aplicaciones Web, que es en las que se suele pensar más a menudo, sino a todo tipo de aplicaciones) que cumplan una serie de características: que sigan funcionando correctamente frente a cualquier ataque, que den acceso a la información que utilizan sólo a usuarios autorizados (autenticándolos cuando sea necesario), que custodien los datos que manejan y los protejan frente a manipulación consciente o inconsciente de los usuarios y que su disponibilidad esté garantizada. Como se puede ver, éste es un concepto muy amplio, puesto que engloba tanto la seguridad en aplicaciones, como la protección de los datos, la protección de las plataformas que soportan aplicaciones y datos y la implementación de arquitecturas seguras.

Se trata sobre todo de un cambio en el enfoque de la seguridad a la hora de desarrollar software: frente a la tradicional "securización (o bastionado) de aplicaciones" que simplemente propugna la protección del software y de los sistemas en los que corre una vez realizado el mismo, mediante elementos que reducen las posibilidades o efectos de un ataque con éxito al mismo, el desarrollo seguro implica pensar en la seguridad del software desde el primer momento y desde las primeras etapas en el ciclo de vida de éste, conociendo y analizando las amenazas que pueden afectar al software que se está desarrollando y definiendo requisitos de seguridad para éste.

El desarrollo seguro se puede definir por tanto como el proceso a través del cual se diseña, se implementa y se prueba el software para que sea seguro.

El objetivo entonces es realizar desarrollos que estén protegidos frente a ataques a priori, no a posteriori como se ha venido realizando hasta ahora. Un caso que sirve como ejemplo claro: aunque es una excelente práctica detener los ataques de sobrecarga de búfer (*buffer overflow*) en una aplicación web higienizando el tráfico HTTP que le llega, resulta todavía mucho mejor arreglar los errores que permiten este ataque y que están presentes en el código fuente de la aplicación. De esta forma se está seguro de evitar este tipo de ataques por completo.

El desarrollo seguro de aplicaciones es un campo muy reciente y que está aún bajo estudio; de hecho, las primeras referencias y los primeros libros que aparecen sobre el tema son del año 2001, lo que demuestra que el interés de todos los implicados en el campo del desarrollo sobre como realizar éste de forma segura es todavía muy nuevo. Esto explica fácilmente el hecho de que a pesar de la importancia del desarrollo seguro y de los grandes riesgos y amenazas que conlleva el no ajustarse a los principios del mismo a la hora de desarrollar las aplicaciones, las prácticas de desarrollo seguro no estén, ni mucho menos, ampliamente extendidas y, además, gran parte de los profesionales que trabajan en este campo no tienen aún clara su necesidad.

La pregunta que surge ahora es ¿cuánta seguridad es realmente necesaria?

Para dar una adecuada respuesta a esto hay que tener en cuenta varias cosas:

- ✓ ¿Cuál es la prioridad principal para el desarrollo que se esté llevando a cabo? ¿La funcionalidad o la seguridad? Sin olvidar que, aunque se quiera tener el desarrollo lo antes posible, las consecuencias de implementar software sin ningún tipo de salvaguardas de seguridad pueden ser catastróficas.
- ✓ ¿En qué entorno se va a manejar el programa desarrollado? En un entorno hostil (Internet) o en un entorno controlado (corporación).
- ✓ ¿Cuál es el coste de introducir todas las medidas de seguridad? Es necesario valorar el coste que supone proteger completamente el desarrollo, frente a las ventajas que aporta hacerlo y los riesgos que se corren al no ha-

cerlo. No tiene sentido invertir mucho en seguridad para algo que quizá no lo necesite realmente.

- ✓ ¿Qué métodos y principios del desarrollo seguro se ajustan más a la organización? Es necesario analizar y valorar las distintas maneras de realizar un desarrollo seguro y cómo pueden encajar dentro de la organización.

En definitiva, en la práctica del desarrollo seguro es fundamental tener en cuenta la seguridad desde el principio del desarrollo, a la hora de encarar un proyecto.

Principios del desarrollo seguro

A continuación se enumeran algunos principios generales aplicables al desarrollo seguro de aplicaciones:

1. Conocer y comprender todas las amenazas que pueden afectar al desarrollo que se vaya a iniciar.
2. Describir los requisitos de seguridad. Es imprescindible detallar quién puede acceder a la aplicación y a los datos, en qué entorno se va a gestionar, etc. Realizando, si es posible, un análisis de los riesgos de posibles ataques a la aplicación.
3. Integrar los requisitos de seguridad con todas las etapas del ciclo de desarrollo de software, y no realizar únicamente pruebas al final, como es habitual.
4. Formar al personal de desarrollo en los problemas de seguridad típicos en los entornos de desarrollo software. Hay que educar a todos los implicados en el mismo en los hábitos de desarrollo de software seguro a través de charlas ilustrativas de los problemas comunes y sus consecuencias inmediatas.
5. Utilizar arquitecturas seguras de despliegue de aplicaciones y servicios. El hecho de que desarrollemos software seguro no debe hacernos olvidar el resto de medidas de seguridad (cortafuegos, bastionado de máquinas, etc.), que también son imprescindibles. En muchos casos, la seguridad de un aplicativo va a depender de las arquitecturas y sistemas que den soporte al mismo, pudiendo éstas incrementar o rebajar el nivel de seguridad según se hayan diseñado e implementado.
6. Realizar pruebas de seguridad de programas en los momentos adecuados. Si se utiliza para realizar el desarrollo un modelo de ciclo de vida del desarrollo (que facilita el poder implementar seguridad en el desarrollo), la norma es realizar las pruebas cuanto antes dentro del ciclo de vida de una aplicación determinada, y además realizarlas lo más a menudo posible, con lo que será más fácil identificar los problemas y fallos, y también más fácil corregirlos. Una prueba de seguridad adecuada debería tener mecanismos para probar a las personas que realizan el desarrollo (en cuanto a su educación en desarrollo seguro y su preocupación por el mismo), los procesos que se siguen, la tecnología utilizada y por supuesto la implementación que se está llevando a cabo.

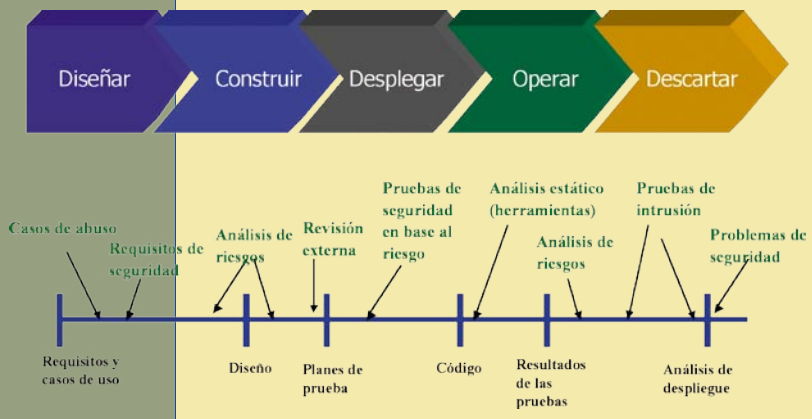

Figura 1: Pruebas de seguridad en distintos momentos del ciclo de vida

7. Desarrollar métricas para evaluar la seguridad de las aplicaciones realizadas. Para poder medir y evaluar el impacto de las distintas actividades del proceso de desarrollo seguro y su efecto en el nivel de seguridad final de la aplicación.

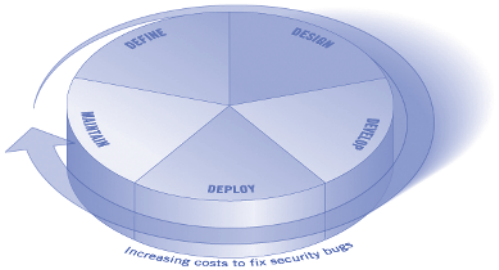

Figura 2. Ejemplo de ciclo de vida del Desarrollo Software y los incrementos de coste derivados del arreglo de errores de seguridad (Fuente: Proyecto OWASP)

Automatización del desarrollo seguro

Existen herramientas y procesos diseñados para automatizar algunos de los principios anteriores:

- ➦ *Métricas:* es importante ser capaces de determinar si en un proyecto las cosas están yendo mejor o peor. A través de las métricas, mediante el análisis de los resultados de las distintas pruebas, se pueden ver las tendencias en seguridad del desarrollo y saber si es necesaria más formación para los integrantes del mismo o si algún proceso no se está realizando

de forma correcta. En definitiva, cuales son los puntos fuertes y débiles en la implantación del desarrollo seguro.

La definición de estas métricas de forma automática a partir del código fuente puede ser complicada (aunque existen ya herramientas con esa capacidad), así que es recomendable utilizar métricas estándares, como las que proporciona el Proyecto OWASP (*Open Web Application Security Project*) de Métricas u otras organizaciones similares.

En la tabla 1 se muestran algunas herramientas de análisis automático de código fuente.

- *Herramientas de pruebas:* existe toda una serie de herramientas que permiten automatizar una parte considerable de las tareas a realizar en un desarrollo seguro, ayudando a los integrantes del proyecto a la hora de realizar las tareas de seguridad necesarias. Sin embargo, es importante conocer con exactitud las pruebas que pueden realizar y las que no pueden abarcar. Tradicionalmente, las herramientas automáticas pueden ser aplicadas en la detección de algunos fallos comunes específicos una vez implementada la aplicación pero no pueden ser utilizadas en puntos previos del desarrollo software.

En la tabla 2 se muestran algunas de las herramientas automáticas que se pueden utilizar para las pruebas de seguridad de aplicaciones Web:

El desarrollo seguro se encuentra, en la práctica, en una fase inicial, muy verde todavía, a pesar de que, como se puede ver consultando las referencias adjuntas, existe ya numerosa información con respecto al desarrollo seguro de aplicaciones, no existiendo ninguna razón para que no se empiecen a aplicar los principios aquí descritos por las organizaciones encargadas de desarrollar aplicaciones sea cual sea su entorno. ❑

Herramienta	URL
Software libre (Open Source)	
RATS	http://www.securesoftware.com
FlawFinder	http://www.dwheeler.com/flawfinder
Microsoft's FXCop	http://www.gotdotnet.com/team/fxcop
Split	http://splint.org/
Boon	http://www.cs.berkeley.edu/~daw/boon/
Pscan	http://www.striker.ottawa.on.ca/~aland/pscan/
Comerciales	
Fortify	http://www.fortifysoftware.com
Ounce labs Prexis	http://www.ouncelabs.com
GrammaTech	http://www.grammatech.com
ParaSoft	http://www.parasoft.com
ITS4	http://www.cigital.com/its4/
CodeWizard	http://www.parasoft.com/products/wizard/

Tabla 1: Herramientas de análisis automático de código fuente

Nombre	URL
Software Libre (Open Source)	
SPIKE	http://www.immunitysec.com
WebScarab	http://www.owasp.org
Paros	http://www.proofsecure.com
Comerciales	
ScanDo	http://www.kavado.com
WebSleuth	http://www.sandsprite.com
SPI Dynamics	http://www.spidynamics.com

Tabla 2: Herramientas de análisis de caja negra de aplicaciones web

Referencias:

- Seguridad en el SDLC (NIST)
  http://csrc.nist.gov/publications/nistpubs/800-64/NIST-SP800-64.pdf
- The OWASP Guide to Building Secure Web Applications (Version 1.0)
  http://www.owasp.org/documentation/guide
- The OWASP Guide to Building Secure Web Applications (Working Draft Version 2.0)
  http://www.owasp.org/documentation/guide
- The Economic Impacts of Inadequate Infrastructure for Software Testing
  http://www.nist.gov/director/prog-ofc/report02-3.pdf
- Threats and Countermeasures – Improving Web Application Security
  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/threatcounter.asp
- The Security of Applications: Not All Are Created Equal
  http://www.atstake.com/research/reports/acrobat/atstake_app_unequal.pdf
- The Security of Applications Reloaded
  http://www.atstake.com/research/reports/acrobat/atstake_app_reloaded.pdf
- Use Cases: Just the FAQs and Answers
  http://www-106.ibm.com/developerworks/rational/library/content/RationalEdge/jan03/UseCaseFAQS_TheRationalEdge_Jan2003.pdf

Sitios Web:

- OWASP — http://www.owasp.org
- Código Seguro — http://www.securecoding.org
- Código Seguro para la arquitectura .NET — http://msdn.microsoft.com/security/securecode/bestpractices/default.aspx?pull=/library/en-us/dnnetsec/html/seccodeguide.asp
- Seguridad en la plataforma Java — http://java.sun.com/security/