SEGURIDAD INFORMÁTICA: BREVES REFLEXIONES SOBRE LA PROGRAMACIÓN SEGURA - Julio 2003

Link: http://www.virusprot.com/Art42.html

Introducción

La industria del software en su creciente y decidida penetración en todos los aspectos de los negocios y procesos de las organizaciones modernas, es una de las realidades tangibles que la sociedad, particularmente del siglo XXI, observa con inquietante y curioso análisis. Si bien, la necesidad de generar mejores soluciones sistematizadas, que aumenten la efectividad de las acciones corporativas y den mayor valor agregado a sus clientes, se contrasta con las presiones e implicaciones que esto lleva en el desarrollo mismo de dichas soluciones.

En este sentido, el balancear la necesidad de obtener un producto de software de óptima calidad que permita a la organización adelantar con oportunidad y alto contenido estratégico sus directrices de negocio, enfrenta un doble desafío para los dedicados a la programación y construcción de los sistemas de información de las organizaciones, así como para los clientes de los mencionados sistemas. Por un lado, todas las variables asociadas con la administración del proyecto en sí mismo, las cuales imprimen una importante complejidad al desarrollo de soluciones informáticas, y en segundo lugar, los altos estándares de aseguramiento de la calidad del desarrollo de software.

En esta encrucijada de formalidad administrativa y técnica, la construcción de sistemas de información ofrece un desafío práctico para las nuevas generaciones de programadores y administradores de proyectos informáticos, y una especial atención de los experimentados ingenieros de software para contextualizar sus aprendizajes en elementos conceptuales y formales que alimenten la práctica de la creación de software.

Por tanto y, considerando el gran reto de la construcción de software, la necesidad de soluciones de software intercomunicadas (orientadas al uso de redes de computadores), la utilización de modernos lenguajes de programación (JAVA, PYTHON, PERL, PHP, entre otros), la necesidad de soluciones eficientes y de alta portabilidad, el uso de metodologías de aseguramiento de calidad, se hace necesario revisar elementos relacionados con las prácticas y principios de programación segura como aspecto complementario del proceso de desarrollo de software.

Principios de un diseño seguro

Si bien las metodologías y estándares relacionados con el desarrollo y construcción de software buscan mantener altos niveles de confiabilidad y control de la solución informática, la seguridad informática y sus principios de diseño seguro no son parte formal de dichos estándares. En este sentido, la formalidad en la evaluación del diseño de la solución desde el punto de vista seguridad podría verse comprometida, dado que no se encuentran claros criterios de análisis en este sentido.

En este sentido, muchos investigadores a nivel internacional han adelantado estudios interesantes sobre este particular (LOSCOCCO, SMALLEY et al. 1999; COWAN, C., WAGLE, P et al. 1999; BISHOP, M. 1999), buscando elementos comunes que permitan orientar a los programadores en estrategias y principios que disminuyan potenciales problemas de seguridad asociado con el desarrollo del software.

Considerando éstos esfuerzos internacionales y tratando de establecer elementos básicos que orienten criterios en la programación segura de aplicaciones, se sugieren algunos principios del diseño seguro de aplicaciones que se enumeran y comentan a continuación: (GALVIN, P. 1998; BISHOP, M. 2003, cap.13)

1. Menor Privilegio
Este principio establece que un sujeto sólo debe dar a un objeto los privilegios que necesita para completar sus tareas asignadas.

2. Economía y simplificación de mecanismos de seguridad
Este principio establece que los mecanismos de seguridad que se establezcan deben ser tan sencillos como sea posible.

3. Configuraciones por defecto seguras
Este principio comenta que a menos que un sujeto haya otorgado acceso explícito a un objeto, éste no debería tenerlo. Es decir, todo lo que no está estrictamente permitido es prohibido.

4. Mediación completa
Este principio afirma que todos los accesos a un objeto(s) deben ser verificados para asegurarse de que cuentan con el permiso para hacerlo.

5. Diseño Abierto
Este principio establece que la seguridad de un mecanismo no debería depender del secreto o confidencialidad de su diseño o implementación.

6. Privilegios Condicionados
Este principio dice que se deben mantener los privilegios necesarios en diferentes momentos, en diferentes rutinas o programas. Es decir, los privilegios no deben ser estáticos para los programas o rutinas en el tiempo y en ejecución.

7. Menor mecanismo común
Este principio comenta que deben existir el menor número de recursos compartidos entre sujetos u objetos.

8. Aceptación psicológica
Este principio comenta que el mecanismo de seguridad que se establezca para un objeto no debe sugerir mayor dificultad a la que si el mecanismo no estuviese presente. En otras palabras, el mecanismo de seguridad deber ser fácil de usar.

Estos principios básicos enunciados, más que sugerir elementos novedosos a la programación de aplicaciones, nos recuerda que son prácticas generales que intuitivamente manejamos, pero que en el momento de la construcción de aplicaciones generalmente se dejan marginados. Por tanto, los principios de un diseño seguro, son directrices generales que deben materializarse en prácticas de programación que deberían ser parte de las formalidades del desarrollo del software mismo, donde la industria y la academia hacen parte fundamental de la misma disciplina.

Cuando estas mínimas sugerencias de diseño seguro no se consideran en la construcción de aplicaciones, la probabilidad de que surjan problemas de seguridad en el futuro es alta, dado que se compromete no solamente la funcionalidad misma de la aplicación sino las condiciones de su elaboración y ambiente de ejecución que puede socavar la confianza de los clientes frente a fallas donde se comprometa la integridad de la información de la organización.

Fuentes de Vulnerabilidades en el Software

Complementario a los elementos establecidos alrededor de los principios de diseño seguro, es importante considerar e identificar elementos prácticos al buscar y establecer vulnerabilidades o fallas de seguridad en el software. Los elementos presentados a continuación responden a experiencias en el desarrollo de software que de alguna manera materializan la ausencia de uso de estándares de desarrollo de software y de adecuadas prácticas de programación, las cuales se encuentran directamente relacionadas con los escenarios de pruebas requeridos para verificar las condiciones y confiabilidad del software: (GOLLMAN, D. 1999, CANO, J. 2000)

1. Cambios en el ambiente de ejecución
Los parches, los cambios en la configuración y variables de entorno alrededor de las aplicaciones son elementos críticos para mantener una ejecución adecuada y controlada de las rutinas y acciones previstas en el software. Al descuidar este aspecto, es probable involucrar efectos de borde o condiciones de excepción no previstas que comprometan no solamente un módulo de la aplicación sino el sistema de información mismo.

2. Desbordamientos y chequeos de sintaxis
Dos elementos importantes en la revisión y evaluación de software. Por un lado la evaluación de los desbordamientos bien sea de memoria o de variables específicas dentro de un programa y por otro lado, la verificación de buen uso de los comandos o palabras reservadas en el lenguaje de programación, que permitan al programador un uso adecuado y eficiente de las estructuras. Si este aspecto no se considera con el rigor necesario, se estará comprometiendo la integridad del ambiente de ejecución de la aplicación.

3. Convenientes pero peligrosas características del diseño del software
Esta fuente de vulnerabilidad nos presenta funcionalidades que son deseables en el software para aumentar la versatilidad de uso de las aplicaciones. Entre estas tenemos herramientas de depuración o *debugging*, conexiones remotas en puertos especiales, entre otras, las cuales ofrecen importante elementos a los programadores y usuarios, pero que generalmente abren posibilidades de ingresos no autorizados que comprometen la integridad de sistemas y socavan la confianza del usuario frente a la aplicación

4. Invocaciones no controladas
En este aparte hacemos referencia a un inadecuado manejo de errores o excepciones en las aplicaciones o exceso de privilegios de ejecución, los cuales se manifiestan en comportamientos inesperados del software que generalmente ofrecen mayores privilegios o accesos adicionales a la información del sistema. En este sentido, el control adecuado de interrupciones, mensajes de error y entorno de ejecución de los programas se vuelve crítico al ser éstos elementos los que definen la interacción del software con el usuario final y su relación con el entorno de ejecución.

5. Bypass a bajo nivel

Las implicaciones de esta fuente de vulnerabilidades hace referencia al aseguramiento que la aplicación debe tener al ser invocada o ejecutada en un ambiente computacional seguro. El programador debe fortalecer y asegurar una manera autorizada de ingreso a la aplicación por parte del usuario, estableciendo mecanismos de monitoreo y control que velen porque esto se cumpla. El sobrepasar un control de acceso a un objeto, bien sea a través de permisos deficientemente otorgados, artificios que interrumpan la normal ejecución (contraseñas de BIOS) o por la manipulación de la memoria de ejecución de la aplicación constituye un atentado directo contra la confiabilidad e integridad del software.

6. Fallas en la implementación de protocolos
Los elementos de seguridad mencionados en este apartado, hacen referencia a las medidas de seguridad en redes. Si bien, los protocolos utilizados para transmisión y control de datos, presentan múltiples fallas, éstas con frecuencia no son consideradas dentro del proceso de implementación de una aplicación. En este sentido, sabemos que las aplicaciones que se ejecutan sobre TCP/IP tienen inherentes las fallas de éste conjunto de protocolos, por tanto es menester del programador establecer junto con el encargado de la seguridad informática, analizar los posibles requerimientos de seguridad necesarios para que la aplicación funcione sobre un ambiente de red que brinde mayores niveles de seguridad y control de tráfico.

7. Fallas en software de base
Todas las aplicaciones finalmente se ejecutan bajo la supervisión de un software de base o sistema operacional. Generalmente cuando se desarrollan aplicaciones, las condiciones o aseguramiento del software de base, no es condición para la adecuada ejecución de aplicaciones. Nada ganamos con aplicar y efectuar un amplio espectro de pruebas y controles, cuando el ambiente de ejecución o el software base no ha pasado por una valoración y afinamiento necesario para asegurar un ambiente de ejecución estable y seguro. En este punto, se llama la atención tanto a proveedores como a programadores, donde el trabajo conjunto debe ser una constante para incrementar los niveles de seguridad y disminuir las vulnerabilidades frecuentes inherentes al arte y la ciencia de programar.

Como hemos revisado hasta el momento, la programación segura continúa siendo un área de investigación permanente. Los elementos planteados hasta el momento en este documento, no buscan agotar la discusión sobre esta realidad ni plantean la necesidad de ofrecer nuevas propuestas para la construcción de software, sino más bien abrir la discusión sobre la necesidad de avanzar en buenas prácticas de programación vinculadas a la disciplina de la Ingeniería de Software, que permitan una cooperación de doble vía buscando en primer lugar, una formalidad en uso y análisis de las herramientas de programación, para validar su confiabilidad y fortaleza en las rutinas, y por otro, la incorporación dentro de las prácticas de ingeniería de software, lo que podría llamarse ingeniería de construcción de software seguro (Mayor información en: SALTER, C., SAYDJARI, O., SCHNEIER, B y WALLNER, J. 1998)

Iniciativas internacionales alrededor del Software Seguro

Frente a esta realidad de la inevitabilidad de la falla, bien sea por inadecuadas implementaciones, fallas humanas o sencillamente limitaciones de la tecnología misma, se han emprendido iniciativas internacionales que aúnan esfuerzos para contar con herramientas de programación más estables y robustas, que combinadas con buenas prácticas de programación buscan disminuir la

presencia de fallas de seguridad que impacten los proyectos de desarrollo de sistemas de información.

Dentro de algunos de los proyectos internacionales planteados en este sentido tenemos:

CYCLONE

<http://www.research.att.com/projects/cyclone/>

SPSMM - Standard de Programación Segura

<http://www.isecom.org/projects/spsmm-es.htm>

CCURED DOCUMENTATION

<http://manju.cs.berkeley.edu/ccured/>

OPEN WEB APPLICATION SECURITY PROJECT (OWASP)

<http://www.owasp.org/>

Conclusiones

Como hemos revisado en este breve documento la programación segura responde al deseo permanente del hombre por avanzar en medio de la dificultad y confrontar el reto de la perfección. Sabemos que el software libre de errores es una meta prácticamente inalcanzable, dado que, por un lado no sabemos nada de las rutinas o herramientas que utilizamos para crearlas y por otro, los buenos hábitos y prácticas de programación no se han desarrollado suficientemente para avanzar en sólidos desarrollos de sistemas de información vistos tanto desde la perspectiva técnica como desde la óptica de administración de proyectos.

En este sentido, es menester de los experimentados profesionales del desarrollo de software, orientar y comunicar su práctica en esta disciplina que ofrezca un contexto práctico para las nuevas generaciones de inquietos “cacharreros”, para que desde sus inicios fomenten acciones formales y buenos hábitos al enfrentarse al reto de un desarrollo a la medida o una adecuación de productos. Así mismo, alertar a los nuevos ingenieros de software a revisar constantemente los paradigmas de programación, para revisar en sus profundidades importantes mecanismos de control y pruebas del software, que permitan contrastar la experiencia de los ingenieros experimentados y los procesos expuestos en los estándares de ingeniería de software.

En razón a lo anterior, avanzar en la búsqueda de opciones y elementos en el área de la programación segura, no es una adición o aderezo de la disciplina inherente al desarrollo de software, sino una fuerza interna de la mencionada disciplina, que pone de manifiesto que el software al igual que las organizaciones es un mundo de relaciones que le dan sentido a sus acciones y le permiten avanzar en el logro de su razón de ser.

Referencias

* LOSCOCCO, SMALLEY et al.(1999) *The inevitability of failure: The flawed assumption of security in modern computing environments*. National Agent Security. http://www.nsa.gov/selinux/doc/inevitability/inevitability.html

* COWAN, C., WAGLE, P et al. (1999) *Buffer overflows: attacks and defenses for the vulnerability of the decade*. Department of computer science and Engineering. Oregon University. http://www.immunix.org/StackGuard/discex00.pdf

* BISHOP, M. (1999) *Principles of Secure Programming*. 1999 USENIX Security Symposium. <http://web.cs.mun.ca/courses/cs3718-f99/overheads/secure_coding_principles.ppt>.

* CANO, J. (2000) *Programación Segura? Conceptos y Aspectos Técnicos*. Conferencia Magistral. Congreso Nacional de Estudiantes de Ingeniería de Sistemas. Santafe de Bogotá. Colombia. Universidad Distrital.

* SALTER, C., SAYDJARI, O., SCHNEIER, B y WALLNER, J. (1998) *Toward a Secure System Engineering Methodology*. http://www.counterpane.com/secure-methodology.html

* GALVIN, P. (1998) *Unix Secure programming FAQ*. Sunworld Magazine. <http://sunsite.uakom.sk/sunworldonline/swol-08-1998/swol-08-security.html>

* GOLLMAN, D. (1999) Computer security. John Wiley and Sons.

* GAFINKEL y SPAFFORD. (1998) Practical Unix and Internet Security. O´Really.

* BISHOP, M. (2003) Computer Security. Art and Science. Addison Wesley

* GRAFF, M y VAN WYK, K. (2003) Secure Coding: Principles & Practices. O' Really. <http://www.securecoding.org/>

D. Jeimy J. Cano
jcano@uniandes.edu.co
Ingeniero de Sistemas y Computación
Universidad de los Andes
COLOMBIA