La Seguridad en el Desarrollo de Aplicaciones

Esp. Miranda Trujillo Marlon Enrique.
Universidad Piloto de Colombia, *Facultad de Ingeniería de Sistemas,*
Especialización en Seguridad Informática.
*Bogotá D.C., m*emt17@gmail.com

Resumen—Este artículo sobre "La Seguridad en el Desarrollo de Aplicaciones", está enfocado en dar a conocer las diferentes vulnerabilidades de algunos de los aspectos expuestos en el top 10 de OWASP – 2013 y de los requisitos y mejores prácticas que deben ser consideradas para mitigarlas.

Índice de Términos— Aplicaciones, Desarrollo, Requisitos, Seguridad, Vulnerabilidades.

Abstract—this article on "The Security in Applications Development" is focused on publicizing the different vulnerabilities of some of the points covered in the top 10 OWASP - 2013 and the requirements and best practices that should be considered to mitigate.

Keywords— Applications, Development, Requirements, Security, vulnerabilities.

1 INTRODUCCIÓN

En este artículo se abordaran algunos de los aspectos más importantes de la seguridad expuestos en OWASP-2013[1] y otros que se consideran tener en cuenta en todo equipo de desarrollo de aplicaciones. Si bien, la seguridad de los sistemas es un tema que ha despertado amplio interés desde hace mucho tiempo, porque no sólo es necesario tener aplicaciones de alta calidad sino también que tengan seguridad. Esto debido a que actualmente la mayor parte de ataques están enfocados a las aplicaciones y teniendo en cuenta primero que todo porque no se lleva a cabo una metodología de seguridad definida durante cada una de fases del CVDS (Ciclo de Vida del desarrollo del Software) y segundo porque la mayoría de los desarrolladores no tienen el conocimiento necesario sobre los aspectos a tener en cuenta para desarrollar código seguro [2] y menos las habilidades. Por estas simples razones se hace necesario que los equipos de desarrollo en las empresas apliquen metodologías y herramientas que permitan desarrollar aplicaciones seguras que cumplan con las exigencias de seguridad en este tiempo.

Lo primero que se debe hacer para comenzar a implementar seguridad en las aplicaciones es identificar cuáles son los activos de información que se deben proteger, cómo protegerlos, cuáles son las vulnerabilidades de los elementos que interactúan con la información y como mitigarlas, al punto de reducirlas a un nivel de riesgo aceptable mediante un proceso iterativo que analice profundamente los riesgos durante todo el CVDS (Ciclo de Vida de Desarrollo del Software). También es muy importante identificar los patrones de ataque en todas las fases y almacenarlos en una base de conocimiento que permita prevenir futuros ataques en otras aplicaciones [3]. Como vemos el tema tratado aquí es bastante amplio por tanto nos centraremos en la segunda afirmación mencionada anteriormente de dar a conocer los aspectos más relevantes que se deben considerar y del que se detallara más adelante.

Hay que resaltar que la seguridad debe estar implícita desde la misma concepción del software, porque es un error dejarla para etapas o fases posteriores del desarrollo. Independientemente del lenguaje y/o plataforma de desarrollo que se utilice, hay cosas que de una o de otra forma siempre hay que tomar en cuenta si queremos tener una aplicación web "relativamente" segura. Además acoto la palabra "relativamente", porque en primer orden nunca se puede obtener una aplicación 100% segura.
Por lo tanto el presente artículo, tiene como fin el poner al lector al tanto de una serie de aspectos y/o principios de seguridad que debería seguir en el proceso de desarrollo si realmente desea obtener un resultado favorable en lo referente a este tema.

2 METODOLOGÍA

En el presente artículo nos limitaremos a exponer los puntos de seguridad que afectan al código escrito por los desarrolladores y también a las características de infraestructura de las que depende

la aplicación, como pueden ser las bibliotecas que se utilizan, la configuración del servidor o los protocolos utilizados.
Siguiendo el esquema anteriormente especificado, se deben tener en cuenta los siguientes aspectos de la seguridad:

Administración de autenticación y Sesión.

Control de Acceso.

Validación de Entradas.

Validación de Entradas (XSS).

Validación de Entradas (Inyección).

Desbordamiento de búfer.

Manejo de errores.

Protección de Datos.

Disponibilidad.

Administración de configuración de aplicación e Infraestructura.

En la especificación de cada aspecto anteriormente mencionado se seguirá el siguiente esquema:

Se nombrarán las principales amenazas y/o vulnerabilidades de cada aspecto de seguridad.

Se listarán para cada amenaza en forma de lista de chequeo todos los requisitos a cumplir para protegerse contra la amenaza.

2.1 Administración de Autenticación y sesión

Cuando hablamos del control de la autenticación y de la sesión, estamos hablando de la forma que la aplicación tiene de manejar la validación del usuario y de mantener bajo su control la sesión que se ha establecido.

2.1.1 Control de autenticación

Los métodos de autenticación existentes son variados y muchos de ellos poco seguros.

a. Autenticación Básica y Digest: Son métodos propios del protocolo http y gestionados directamente por el navegador y el servidor (sin control por parte del desarrollador). Envían el usuario y la contraseña de forma clara por la red por lo que son métodos inseguros dado que cualquiera podría escuchar en la red y recoger los datos del usuario. En aplicaciones seguras sólo se deben usar si se combina con SSL para obtener confidencialidad e integridad.

b. Autenticación basada en formulario: Permite al desarrollador mayor control sobre el interfaz de autenticación pero sufre de los mismos inconvenientes que los anteriores métodos, transmisión en texto plano y necesidad de controlar la complejidad de las contraseñas.

c. Autenticación Integrada: Propia de las aplicaciones de Intranet que usan Microsoft IIS. El navegador obliga a presentar una pantalla de autenticación que le permitirá validar al usuario contra el Active Directory a través del protocolo Kerberos. La seguridad del método depende de la seguridad intrínseca de la Intranet.

d. Autenticación basada en certificados: La autenticación entre cliente y servidor se realiza a través de certificados. La seguridad dependerá de la calidad de la clave pública y de la relación de confianza.

El método de autenticación usado dependerá del nivel de seguridad que requiera la aplicación.

Amenaza/Vulnerabilidad: Obtención de los credenciales de un usuario.
Requisitos:

- La petición de credenciales a los usuarios debe hacerse en páginas protegidas mediante HTTPS.
- El proceso de autenticación no debe poder ser evitado mediante ataques de inyección SQL.
- Autenticación fuerte. Uso de certificados en vez de autenticación usuario/contraseña. El método de autenticación fuerte exige utilizar “algo que sepas y algo que tengas” como en el caso del uso de tarjetas con certificados combinadas con claves secretas. Estos métodos dotan de mayor seguridad al sistema por encima incluso de los biométricos.
- Autenticación usuario/contraseña. Deben validarse del lado del servidor que los

credenciales cumplen la política de seguridad que se haya definido (complejidad de contraseña, reintentos, expiración, etc.).
- No usar métodos de validación a nivel del cliente (como a través de JavaScript). Son fácilmente sobrepasados. Las validaciones deben hacerse a nivel de servidor.
- No codificar en partes públicas de la aplicación (comentarios, manuales de ayuda) los credenciales de usuarios especiales.
- Evitar el cacheo de la contraseña localmente en el navegador. Usar la propiedad autocomplete = off en los campos de entrada de contraseña y usuario.
- Las páginas de autenticación deben ser marcadas con etiquetas HTML que imposibiliten el uso de caché en el navegador.
- Política de restauración de contraseñas. Se debe elegir un método que asegure que el usuario es el auténtico, resetear la contraseña y enviar por un canal seguro una clave provisional.
- Almacenamiento de contraseñas. Guardar las contraseñas cifradas (mediante MD5 o SHA1) o mediante una función de hash de forma que el proceso sea irreversible.

Amenaza/Vulnerabilidad: Acceso al sistema a través de cuentas por defecto.
Requisitos:
- Evitar dentro de la infraestructura de la aplicación el uso de credenciales por defecto, ya sea en los servidores web, de aplicaciones o en base de datos.

- Evitar las puertas traseras en la aplicación (credenciales usadas por los desarrolladores para acceder a ciertas partes de la aplicación).

Amenaza/Vulnerabilidad: Ataque por fuerza bruta.
Requisitos:
- Debe intentarse que el esquema de códigos de usuarios no sea predecible ni de información pública.
- La función de cambio de contraseña debe incluir la contraseña antigua y una confirmación de la nueva.
- Las cuentas de usuario son bloqueadas si la contraseña se ingresa más de un determinado nº de veces (aunque puede provocar denegación de servicio).
- Presentar un mensaje de error que no revele que parte de las credenciales son incorrectas.

2.1.2 Administración de sesión

Al tratarse el protocolo http de un protocolo sin estado, el servidor de aplicaciones o la aplicación web se encargan de emplear algún tipo de administración de sesión que permita mantener el estado de la comunicación con el usuario. Las sesiones son almacenadas en los servidores y asociadas con los correspondientes usuarios a través de identificadores de sesión (ID's). Este identificador se convierte en un objetivo atractivo para los atacantes, ya que obteniéndolo consiguen suplantar la identidad del usuario.

El proceso básico establece que el servidor genera un identificador de sesión en algún momento de la comunicación con el usuario, le manda este identificador y espera que el navegador del usuario se lo reenvíe en cada petición. Podemos asemejar este proceso con el paso de un testigo entre usuario y servidor.
En el paso de este testigo se están empleando tres métodos con diferentes características de seguridad:

- Parámetros en la URL.
- Campos de formulario ocultos.
- Cookies. Se ha comprobado que las cookies son el método más seguro.

Amenazas y Requisitos de Seguridad.
Amenaza/Vulnerabilidad: Implementación insegura de la gestión de sesión.
- Utilizar un framework web reconocido (J2EE, .NET, PHP) con administradores de sesiones probados y en versiones que implementen testigos de sesión robustos mediante criptografía.
- Evitar guardar parámetros propios de la sesión de un usuario en campos ocultos o directamente en la URL con el fin de almacenar el estado de las peticiones. Estos parámetros se deben almacenar exclusivamente en el lado del servidor (objetos en sesión). Especial cuidado con los parámetros que definen la autorización y el rol del usuario.
- Evitar la idea de que almacenar demasiados objetos en sesión limita los recursos disponibles

del servidor. Hay que llegar a un compromiso entre la seguridad de la aplicación y los recursos necesarios.

Amenaza/Vulnerabilidad: Ataques de pre asignación de sesión.
Requisitos:
- El framework web utilizado no debe suministrar una sesión valida por sólo visitar una página sin haber sido autenticado. Sólo una vez el usuario ha sido autenticado se le debe suministrar un identificador de sesión, esto evita que cualquier usuario pueda conectar con la aplicación, obtener un identificador y utilizarlo para asignar una sesión a otro usuario.
- El servidor de aplicaciones no debe permitir establecer el id de sesión en la URL, sólo debe ser posible en una cookie (tampoco es una solución totalmente segura, herramientas como 'Paros' permiten establecer la sesión que queramos en la cookie). Se trata de ocultar lo más posible la posibilidad de asignar una sesión.
- Destrucción de sesiones en el proceso de logout. El servidor de aplicaciones no debe permitir usar id de sesiones una vez que la sesión ha expirado o se ha salido de esa sesión. En el cliente se debe seguir un mecanismo de borrado de cookies de sesión o de sobre escritura de estas cookies al salir de la aplicación. Los navegadores sólo destruyen
- las cookies de sesión cuando el proceso es eliminado. Un usuario podría utilizar un navegador abierto en el mismo PC para utilizar la cookie de sesión.
- El servidor debe borrar la sesión.

Amenaza/Vulnerabilidad: Ataques de predicción de sesión.
Requisitos:
- Generar identificadores de sesión mediante algoritmos criptográficos fuertes. Los códigos de sesión generados por el servidor de aplicaciones y servidor web deben ser únicos, no predecibles y resistentes a la ingeniería inversa.

Amenaza/Vulnerabilidad: Expiración de sesión.
Requisitos:
- Exigir a los servidores HTTP que los ID's de sesión expiren. Si no se le otorga al atacante tiempo ilimitado para adivinar el ID de sesión.

Amenaza/Vulnerabilidad: Robo de sesión mediante fuerza bruta.
Requisitos:
- Existen herramientas que permiten al atacante adivinar un ID de sesión válida. Una solución posible es implantar a nivel de red un control mediante IDS que bloquee la dirección IP desde la que se produce el ataque.
- Longitud de los identificadores lo suficiente larga respecto al número de sesiones simultáneas.

Amenaza/Vulnerabilidad: Ataques de intercepción de sesión.
Requisitos:
- Escuchar la red mediante sniffers permite al atacante obtener el ID de la sesión y reutilizarlo. Se deben usar tecnologías de cifrado en la transmisión como SSL o TSL para salvaguardar la sesión de un usuario.
- Tener implantados sistemas IDS permite detectar la ejecución de sniffers en una red interna y actuar en consecuencia.

2.2 Control de Acceso

La importancia de la seguridad en este apartado estriba en que la aplicación web permitirá el acceso a contenidos y funciones dependiendo del usuario. Cualquier error en este control puede permitir que un usuario aumente sus privilegios comprometiendo la seguridad de la aplicación e incluso de la infraestructura.

La importancia de este control aumenta cuando existen interfaces administrativas que permiten administrar las aplicaciones a los Administradores y que se convierten en objetivo de los atacantes.

Amenaza/Vulnerabilidad: Eludir el control de acceso.
Requisitos:
- Documentación de la política de control de acceso de la aplicación.
- Cerciorarse que el código cumple la política de control de acceso. Es preferible que el control de acceso esté centralizado.
- Protección de los canales remotos de comunicación entre los administradores y las

aplicaciones. Sólo usuarios con permisos pueden administrar el sitio web y realizar cambios en el entorno de producción.
- Ningún parámetro disponible públicamente en la aplicación debe permitir controlar el acceso. Típicamente se trata de parámetros en la URL, de campos ocultos, de funciones validadas en el clientes (javascript, vbscript...) o de cookies.
- La aplicación debe validar las peticiones a recursos protegidos antes de permitirla.
- La aplicación no debe permitir cambiar el identificador de sesión una vez el usuario se ha validado. Si se permite cambiar, por ejemplo, el parámetro que determina el código de usuario, esto puede provocar acceso a información privilegiada.
- No se puede comprometer mediante inyección de SQL el acceso a recursos protegidos por usuario/password.
- Protección del acceso a recursos estáticos. El acceso a documentos estáticos (pdf, word, excel) guardados en el servidor en formato electrónico debe protegerse estableciendo permisos y validándolos en cada acceso. No se debe permitir que el acceso a los documentos sea público y baste con conocer la url.
- Se incluyen cabeceras http y etiquetas 'meta' para evitar que páginas sensibles se guarden en la caché del cliente.

2.3 Validación de Entradas

Las aplicaciones web utilizan parámetros en las llamadas http para saber cómo responder al usuario y poder manejar las diferentes funcionalidades de la aplicación. Estos parámetros son utilizados por los atacantes para intentar saltarse todos los mecanismos de seguridad. Los ataques que se producen utilizando los parámetros que manejan las aplicaciones web se basan, en su mayor parte, en codificar la información maligna en diferentes formatos con el fin de que los caracteres utilizados pasen las validaciones y saltarse la seguridad.

Las principales fuentes de entrada de una aplicación son:
- Cadenas de consulta en la URL.
- Campos de formulario.
- Cookies.
- Cabeceras HTTP.

Las aplicaciones web deben protegerse de la utilización de estos parámetros de forma indebida. Para ello los desarrolladores deben decodificar todos los parámetros a su forma más simple antes de validarlos.

Amenaza/Vulnerabilidad: Falta de validación de los parámetros de entrada.
Requisitos:
- Validar del lado del servidor todos aquellos parámetros que provengan de una petición del usuario. Lo más efectivo sería crear una librería centralizada que implementara la validación contra un formato estricto que especifique el tipo de dato, el conjunto de caracteres permitidos, la longitud, entre otros. Ya existen firewalls de nivel 7 que pueden llevar a cabo este cometido, pero para ello deben estar configurados con una definición estricta de lo que es válido para cada parámetro del sitio.
- Las validaciones a nivel de cliente sólo deben ser realizadas para dar mayor sensación de usabilidad y rendimiento al usuario pero deben llevar por detrás una validación en el servidor.

2.4 Validación de Entradas (XSS).

Dentro del control de validación de entradas, se dedica un apartado especial a la validación de entradas que pueden provocar ejecución de código malicioso en un usuario diferente a través de lenguajes como el Javascript, Vbscript, Flash, etc.

En este tipo de ataque, se utiliza la existencia de entradas no validadas, para introducir scripts que el navegador del usuario ejecutará. Este script puede hacer uso de las cookies del usuario y de cualquier información de éste y enviársela al atacante.

Se pueden clasificar en dos categorías, **almacenados y reflejados**. Cuando hablamos de **XSS almacenado** estamos diciendo que el código inyectado se almacena permanentemente en el servidor objetivo, ya sea BBDD, un foro, de forma que cuando un usuario requiere esa información se encuentra con un script que se ejecuta en su máquina.

Los ataques reflejados se caracterizan porque es el servidor el que refleja el script malicioso como resultado de búsquedas incorrectas, mensajes de error, enviándolo a las víctimas por otros medios como pueden ser el correo u otro servidor web.

Cualquiera que sea la forma de ataque los problemas pueden ser graves, desde obtener la sesión del usuario, instalar puertas traseras o modificar las páginas web del sitio. Para ello el atacante hace uso de estrategias diferentes para codificar el script de forma que no sea interceptado, usando Unicode en vez de ASCII.

Amenaza/Vulnerabilidad: Ejecución de código malicioso en clientes.
Requisitos:
- Revisión de código buscando aquellos sitios donde una solicitud http podría encontrar camino hacia la salida de html.
- Definición de una política de seguridad que determine los valores permitidos en los parámetros de entrada (cookies, campos ocultos, formularios, cadenas URL).
- Conversión en las salidas generadas dinámicamente de los caracteres siguientes:

Convertir el Carácter	Carácter convertido
>	&lt;
<	&gt;
(	&#40;
)	&#41;
#	&#35;
&	&#38;

Fig. 1. Tabla de Caracteres dinámicos. [5]

2.5 Validación de Entradas (Inyección).

A través de las entradas a una aplicación se puede conseguir ejecutar código en los sistemas que forman parte de la infraestructura de la aplicación, ya sea el Sistema operativo, las Bases de Datos (inyección SQL), el servidor de correo. Por ello hay que tener especial cuidado cuando una aplicación utiliza los parámetros de una petición http para solicitar el servicio de una aplicación externa.

Amenaza/Vulnerabilidad: Inyección de comandos del Sistema Operativo.
Requisitos:
- Evitar llamadas a intérpretes externos a través de comandos como System, fork, exec, Runtime.exec. Si es necesario hacerlo utilizar librerías propias del framework utilizado.
- La aplicación debe ejecutarse con los privilegios mínimos para realizar su función. Esto exige no ejecutar el servidor como Administrador.

Amenaza/Vulnerabilidad: Inyección SQL.
Requisitos:
- Validación de los parámetros dinámicos suministrados a las consultas SQL.
- Uso de procedimientos almacenados y sentencias preparadas (en java prepared statements) para realizar las llamadas a BBDD de modo que los parámetros sean tratados como datos y no como código ejecutable.

Amenaza/Vulnerabilidad: Inyección LDAP.
Requisitos:
- La aplicación no debe procesar comandos LDAP del usuario.

2.6 Desbordamiento de Buffer

Dentro de los ataques que aprovechan los parámetros de entrada a una aplicación éste puede ser el más complicado pero el más peligroso. Si el atacante logra encontrar un desbordamiento de búfer puede tomar el control de la máquina.

El desbordamiento de búfer se aprovecha de la falta de validación de un parámetro de entrada para introducir en la pila de la aplicación código arbitrario que luego pasa a ejecutar. Este código da la posibilidad al atacante de tomar el control remoto de la máquina mediante la apertura de una puerta trasera.

Lo más habitual es aprovechar las fallas descubiertas en productos o librerías estándar que todavía no han sido parcheadas correctamente. Cabe destacar que la mayoría de servidores de aplicaciones y servidores web son susceptibles a estos errores si exceptuamos los ambientes Java y J2EE (omitiendo los desbordamientos propios de la JVM).

Amenaza/Vulnerabilidad: Ejecución de código arbitrario inyectado por el atacante.

Requisitos:
- Utilización de productos y librerías actualizados, cuyos reportes de vulnerabilidades han sido parcheados.
- Si es una aplicación a medida, validación de que la aplicación soporta peticiones http con entradas arbitrariamente grandes.
- Examen periódico con escáneres que busquen desbordamientos en los elementos de la infraestructura o en la aplicación. Parcheado de las posibles vulnerabilidades.

2.7 Manejo de Errores

El manejo inadecuado de errores introduce problemas de seguridad a un sitio web. El error más común se presenta cuando se muestra información detallada del error (trazas de la pila, errores de BBDD, códigos de error). Esto mensajes le dan al atacante información privilegiada de la implementación de la aplicación y pistas para poder comprometerla.

Amenaza/Vulnerabilidad: Manejo inadecuado de errores.
Requisitos:
- Cumplimiento de una política específica del manejo de errores que incluye los tipos de errores a ser manejados, la información que se debe guardar para auditar y la que se debe presentar al usuario.
- Respuestas específicas a los errores evitando descubrir datos innecesarios.

2.8 Protección de Datos

Muchas de las aplicaciones web necesitan guardar información sensible, en Base de Datos o en un sistema de archivos. Para evitar exponer esta información a posibles ataques se necesita poner atención en el desarrollo de los módulos de la aplicación que manejan esta información.

Amenaza/Vulnerabilidad: Fallo al cifrar información sensible o confidencial.
Requisitos:
- Elección de una biblioteca expuesta a la revisión pública y sin vulnerabilidades abiertas.
- Revisión de los módulos que empleen criptografía.

Amenaza/Vulnerabilidad: Almacenamiento inseguro de llaves, certificados y contraseñas.
Requisitos:
- Almacenamiento de certificados, llaves y contraseñas en ubicaciones seguras, incluso almacenarlas en distintas ubicaciones.

Amenaza/Vulnerabilidad*:* Datos sensibles en el código.
Requisitos:
Revisar la existencia de datos sensibles en el código que puedan ser guardados en la caché del navegador y permitir montar un ataque.

2.9 Disponibilidad

Las aplicaciones web son susceptibles a ataques de negación de servicio, que provocan la indisponibilidad de la misma para un usuario o para la totalidad. A diferencia de la negación de servicio a nivel de red, la aplicación no puede determinar si se trata de un ataque o de un pico de tráfico generado por muchos usuarios realizando la misma petición. Es fácil para el atacante realizar tantas peticiones al servidor que recursos como los hilos que comunican con BBDD o el ancho de banda, sean consumidos y el resto de los usuarios no puedan acceder, o bloquear el acceso de un usuario a un sitio restringido mandando credenciales inválidas hasta que quede bloqueada la cuenta.

Amenaza/Vulnerabilidad: Denegación por tráfico pesado.
Requisitos:
- Realización de pruebas de carga.

Amenaza/Vulnerabilidad: Denegación de acceso a recursos.
Requisitos:
- Limitar recursos asignados a un usuario. Establecer cuotas de carga sobre el sistema por usuario.

Amenaza/Vulnerabilidad: Denegación por manejo inadecuado de errores.
Requisitos:
- Cumplimiento de la política de manejo errores. Cada posible error debe ser controlado para no provocar caídas de la aplicación.

2.10 Administración de Configuración, de Aplicación e Infraestructura

El servidor Web y el servidor de aplicaciones tienen un papel clave en la seguridad de una aplicación Web, cometer errores en la configuración de estos servidores da lugar a diferentes fallas de seguridad.

Amenaza/Vulnerabilidad: Errores de seguridad no parcheados en el software del servidor.
Requisitos:
- Aplicar últimos parches y vigilar nuevas vulnerabilidades publicadas.
- Búsquedas regulares de vulnerabilidades desde máquinas externas a la red y desde máquinas en la red. Usar escáneres de seguridad tipo 'Nessus' o 'Nikto'.

Amenaza/Vulnerabilidad: Mala configuración de los parámetros del servidor y servicios innecesarios habilitados.
Requisitos:
- Cumplimiento de guía de bastionado de los servidores tanto de producción como de desarrollo.
- Revisiones de la configuración de seguridad del servidor.

Amenaza/Vulnerabilidad: Existencia de archivos por defecto y rutas comunes.
Requisitos:
- Eliminación de los archivos y rutas por defectos de cada entorno Web.
- Verificación de que no se incluyen copias de seguridad de los ficheros de la aplicación.

Amenaza/Vulnerabilidad: Certificados SSL y opciones de encriptación mal configuradas.
Requisitos:
- Validación de los certificados y de los algoritmos de encriptación usados.

Amenaza/Vulnerabilidad: Acceso a interfaces administrativas.
Requisitos:
- Asegurar que las interfaces administrativas de las aplicaciones y servidores no son accesibles desde Internet.

3 RESPONSABILIDADES

En este apartado se identifican las responsabilidades de los diferentes colectivos involucrados en el desarrollo de aplicaciones a medida, bien sean para su entrega a clientes o para uso Corporativo o departamental.

3.1 Gerentes y Gestores de Proyecto

- Difundir entre los equipos de desarrollo a su cargo las normas detalladas en este artículo.
- Asegurar que el personal conoce la Guía de Requisitos de Seguridad en Aplicaciones y llamar su atención cuando tome conocimiento de su falta de cumplimiento.

3.2 Equipos de Desarrollo

- Conocer y aplicar lo establecido en la Guía de Requisitos de Seguridad en Aplicaciones.
- Tomar conciencia que se está actuando en representación de Indra que es un proveedor crítico para sus clientes. Cualquier riesgo derivado de una mala política en cuanto a la Seguridad en aplicaciones puede afectar al posicionamiento de Indra ante sus clientes.

4 CONCLUSIONES

Proponer el presente artículo como fuente de información y buenas prácticas, que permita apoyar a las organizaciones y sus aéreas de TI, especialmente a sus equipos de desarrollo en lo relacionado a la seguridad en el desarrollo de sus aplicaciones y los riesgos que implica el no tenerlas en cuenta.

5 REFERENCIAS

[1] Foundation OWASP. Top 10-2013 https://www.owasp.org/images/5/5f/OWASP_Top_10_-_2013_Final_-_Espa%C3%B1ol.pdf [2013]

[2] M. Brown & A. Paller. "Secure software development: Why the development world awoke to the challenge". Information Security Technical Report, Vol. 13, No. 1, pp. 40-43, 2008.

[3] J. H. Allen et al. "Software Security Engineering: A Guide for Project Managers". Addison-Wesley. 2008.

[4] Código de buenas prácticas para la Gestión de la Seguridad de la Información. UNE-ISO/IEC 17799

[5] Tabla de Códigos HTML, Caracteres y Símbolos http://ascii.cl/es/codigos-html.htm