



# GUÍA DE SEGURIDAD DE LAS TIC (CCN-STIC-812)

## SEGURIDAD EN ENTORNOS Y APLICACIONES WEB

Edita:



© Editor y Centro Criptológico Nacional, 2011  
NIPO: 076-11-053-3

Tirada: 1000 ejemplares

Fecha de Edición: octubre de 2011

Raúl Siles ha participado en la elaboración y modificación del presente documento y sus anexos.

### **LIMITACIÓN DE RESPONSABILIDAD**

El presente documento se proporciona de acuerdo con los términos en él recogidos, rechazando expresamente cualquier tipo de garantía implícita que se pueda encontrar relacionada. En ningún caso, el **Centro Criptológico Nacional** puede ser considerado responsable del daño directo, indirecto, fortuito o extraordinario derivado de la utilización de la información y software que se indican incluso cuando se advierta de tal posibilidad.

### **AVISO LEGAL**

Quedan rigurosamente prohibidas, sin la autorización escrita del **Centro Criptológico Nacional**, bajo las sanciones establecidas en las leyes, la reproducción parcial o total de este documento por cualquier medio o procedimiento, comprendidos la reprografía y el tratamiento informático, y la distribución de ejemplares del mismo mediante alquiler o préstamo públicos.

## PRÓLOGO

El uso masivo de las tecnologías de la información y las telecomunicaciones (TIC), en todos los ámbitos de la sociedad, ha creado un nuevo espacio, el ciberespacio, donde se producirán conflictos y agresiones, y donde existen ciberamenazas que atentarán contra la seguridad nacional, el estado de derecho, la prosperidad económica, el estado de bienestar y el normal funcionamiento de la sociedad y de las administraciones públicas.

La Ley 11/2002, de 6 de mayo, reguladora del Centro Nacional de Inteligencia, encomienda al Centro Nacional de Inteligencia el ejercicio de las funciones relativas a la seguridad de las tecnologías de la información en su artículo 4.e), y de protección de la información clasificada en su artículo 4.f), a la vez que confiere a su Secretario de Estado Director la responsabilidad de dirigir el Centro Criptológico Nacional en su artículo 9.2.f).

Partiendo del conocimiento y la experiencia del CNI sobre amenazas y vulnerabilidades en materia de riesgos emergentes, el Centro realiza, a través de su Centro Criptológico Nacional, regulado por el Real Decreto 421/2004, de 12 de marzo, diversas actividades directamente relacionadas con la seguridad de las TIC, orientadas a la formación de personal experto, a la aplicación de políticas y procedimientos de seguridad, y al empleo de tecnologías de seguridad adecuadas.

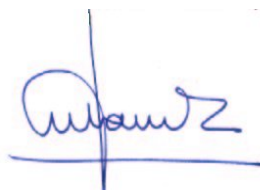
Una de las funciones más destacables del Centro Criptológico Nacional es la de elaborar y difundir normas, instrucciones, guías y recomendaciones para garantizar la seguridad de los sistemas de las tecnologías de la información y las comunicaciones de la Administración, materializada en la existencia de la serie de documentos CCN-STIC.

Disponer de un marco de referencia que establezca las condiciones necesarias de confianza en el uso de los medios electrónicos es, además, uno de los principios que establece la ley 11/2007, de 22 de junio, de acceso electrónico de los ciudadanos a los servicios públicos, en su artículo 42.2 sobre el Esquema Nacional de Seguridad (ENS).

Precisamente el Real Decreto 3/2010 de 8 de Enero de desarrollo del Esquema Nacional de Seguridad fija los principios básicos y requisitos mínimos así como las medidas de protección a implantar en los sistemas de la Administración, y promueve la elaboración y difusión de guías de seguridad de las tecnologías de la información y las comunicaciones por parte de CCN para facilitar un mejor cumplimiento de dichos requisitos mínimos.

En definitiva, la serie de documentos CCN-STIC se elabora para dar cumplimiento a los cometidos del Centro Criptológico Nacional y a lo reflejado en el Esquema Nacional de Seguridad, conscientes de la importancia que tiene el establecimiento de un marco de referencia en esta materia que sirva de apoyo para que el personal de la Administración lleve a cabo su difícil, y en ocasiones, ingrata tarea de proporcionar seguridad a los sistemas de las TIC bajo su responsabilidad.

Octubre de 2011



Félix Sanz Roldán  
Secretario de Estado  
Director del Centro Criptológico Nacional

ÍNDICE

1. INTRODUCCIÓN .....	4
2. OBJETO .....	4
3. ALCANCE .....	5
4. SEGURIDAD DE ENTORNOS Y APLICACIONES WEB.....	5
4.1. RIESGOS Y AMENAZAS DE SEGURIDAD EN ENTORNOS WEB.....	5
4.2. INCIDENTES DE SEGURIDAD EN ENTORNOS WEB .....	7
4.3. VULNERABILIDADES DE SEGURIDAD EN ENTORNOS WEB .....	8
4.4. ESTRATEGIA Y METODOLOGÍA DE SEGURIDAD DE APLICACIONES WEB .....	9
4.5. ARQUITECTURAS DE SEGURIDAD EN ENTORNOS WEB .....	12
4.6. DESARROLLO SEGURO DEL SOFTWARE DE APLICACIONES WEB.....	15
4.6.1. RECOMENDACIONES GENERALES.....	15
4.6.2. FILTRADO DE DATOS DE ENTRADA DEL USUARIO .....	15
4.6.3. MENSAJES DE ERROR Y OTROS CONTENIDOS .....	19
4.6.4. AUTENTIFICACIÓN Y GESTIÓN DE SESIONES.....	20
4.6.5. CSRF.....	21
4.6.6. GESTIÓN DE LOGS .....	21
4.7. ANÁLISIS DE SEGURIDAD DE APLICACIONES WEB.....	22
4.7.1. METODOLOGÍA DE ANALISIS DE CAJA NEGRA.....	22
4.7.2. METODOLOGÍA DE ANALISIS DE CAJA BLANCA .....	23
5. ESPECIFICACIÓN DE REQUISITOS DE AUDITORÍAS DE SEGURIDAD DE ENTORNOS WEB .....	24
5.1. ÁMBITO DE LA AUDITORÍA DE SEGURIDAD .....	26
5.2. RECONOCIMIENTO.....	26
5.2.1. INFORMACIÓN DE REGISTRO DE DOMINIOS (DNS) Y DIRECCIONES .....	26
5.2.2. SERVICIOS DE BÚSQUEDAS EN INTERNET .....	27
5.2.3. UBICACIÓN EN LA RED .....	27
5.3. ESCANEEO .....	27
5.3.1. SERVICIOS WEB .....	28
5.3.2. CONTENIDOS WEB .....	28
5.4. ANÁLISIS DE VULNERABILIDADES.....	29
5.4.1. VULNERABILIDADES DE APLICACIONES WEB.....	29
5.4.2. PRUEBAS DE CARGA Y DENEGACIÓN DE SERVICIO (DOS).....	31
ANEXO A. REFERENCIAS PRINCIPALES .....	33
ANEXO B. LISTA DE COMPROBACIÓN Y REQUISITOS .....	34
ANEXO C. GLOSARIO DE TÉRMINOS Y ABREVIATURAS.....	41
ANEXO D. REFERENCIAS.....	43

FIGURAS

FIGURA 1.- ESTRATEGIA Y METODOLOGÍA DE SEGURIDAD DE APLICACIONES WEB .....	11
--	----

## 1. INTRODUCCIÓN

1. Dada la criticidad tanto de los entornos de producción como de los dispositivos embebidos, se hace necesario establecer una metodología que permita evaluar y reforzar la seguridad de los entornos y aplicaciones Web asociados a éstos.
2. Desde el punto de vista de la seguridad, todos los elementos que conforman el entorno o aplicación Web descritos en el punto previo deben ser tenidos en cuenta a la hora de evaluar y diseñar los mecanismos de protección.
3. El presente documento establece unas pautas de carácter general enfocadas a las características propias de las aplicaciones y servicios web para establecer una política de seguridad en dichos sistemas teniendo en cuenta los requerimientos establecidos en el Real Decreto 3/2010, de 8 de enero, por el que se regula el Esquema Nacional de Seguridad en el ámbito de la Administración Electrónica. Se espera que cada organización las particularice para adaptarlas a su entorno singular.

## 2. OBJETO

4. Actuar como guía de referencia en la identificación y el análisis de los requisitos de seguridad asociados a las aplicaciones y entornos Web en el ámbito del Esquema Nacional de Seguridad, con el objetivo de reducir las posibles amenazas de seguridad asociadas a estos entornos y aplicaciones durante su diseño y antes de su paso a producción.
5. Existen dos procedimientos para mejorar la seguridad de las aplicaciones y entornos Web:
  - Durante el proceso de diseño y desarrollo de la aplicación, estableciendo unos requisitos y verificaciones de seguridad que debe cumplir toda aplicación Web.
  - Posteriormente, tras la puesta en producción, realizando el análisis de la aplicación Web mediante auditorías de seguridad, análisis de vulnerabilidades y pruebas de intrusión.
6. El objetivo del presente documento es proporcionar la información necesaria para la aplicación de ambos procedimientos, incluyendo una lista de recomendaciones que deberían ser aplicadas durante las fases de diseño, desarrollo y auditoría de la aplicación Web, dependiendo de la categoría del sistema.
7. Adicionalmente, la presente guía define los requisitos necesarios para la realización de auditorías de entornos Web por terceros, con el objetivo de obtener unos resultados que sean un reflejo completo de la seguridad de los servicios Web prestados, los resultados sean relevantes desde el punto de vista práctico, y la información obtenida se trate con la reserva oportuna.

8. Adicionalmente a la información contenida en la presente guía, en el proceso de contratación de una aplicación Web a un tercero, es necesario fijar una serie de requisitos, dentro de los cuales se encuentran los de seguridad. Esta guía pretende ofrecer recomendaciones para el establecimiento y elaboración detallada de esos requisitos mínimos, y se complementa con la guía de OWASP centrada en los contenidos propios de un contrato de estas características, y denominada *Secure Software Contract Annex* [Ref.- 17].

### 3. ALCANCE

9. Dar cumplimiento con lo establecido por el Real Decreto 3/2010, de 8 de enero, por el que se regula el Esquema Nacional de Seguridad, y más específicamente en lo relativo a las medidas de protección mp.s.2, mp.s.8 y mp.s.9.

## 4. SEGURIDAD DE ENTORNOS Y APLICACIONES WEB

### 4.1. RIESGOS Y AMENAZAS DE SEGURIDAD EN ENTORNOS WEB

10. La seguridad en el proceso de diseño y desarrollo de una aplicación Web debe abordar principalmente dos elementos:
  - El entorno de desarrollo Web, en el que se deberá disponer de la última versión que soluciona vulnerabilidades de seguridad previas y conocidas.
  - La aplicación Web propietaria, código propio, basada en el lenguaje de programación empleado.
11. Las grandes amenazas de seguridad de las aplicaciones Web están asociadas a las siguientes características intrínsecas a este tipo de entornos:
  - Las aplicaciones Web en Internet están públicamente disponibles.
  - La ubicuidad de las aplicaciones Web es muy elevada, ya que están disponibles en prácticamente cualquier entorno de computación.
  - Las aplicaciones Web estándar utilizan el puerto TCP/80 (HTTP) y las que emplean cifrado mediante SSL (*Secure Socker Layer*) o TLS (*Transport Layer Security*) emplean el puerto TCP/443 (HTTPS). Por tanto, los firewalls tradicionales de manera general deben dejar pasar el tráfico hacia estos puertos, y son de poca utilidad en el filtrado de ataques directos sobre la aplicación Web.
  - HTTP es un protocolo complejo que permite recibir datos del usuario (en la URL mediante el método GET, en el cuerpo de la petición mediante el método POST, mediante métodos HTTP más avanzados, a través de cookies, mediante cabeceras HTTP, etc) para su procesamiento.

- Las técnicas y herramientas de ataque necesarias para explotar vulnerabilidades en los entornos Web son muy sencillas, siendo en algunos casos más que suficiente el uso exclusivo de un navegador Web estándar.
  - Los ataques desde Internet sobre las aplicaciones Web disponibles públicamente conllevan un anonimato muy elevado por parte del atacante. Este anonimato se incrementa notablemente cuando el atacante emplea dispositivos intermedios (proxies anónimos) para ocultar su origen real.
  - La mayoría de aplicaciones Web tienen carácter propietario y han sido desarrolladas internamente o mediante la solicitud de este servicio a una tercera compañía. Por estos motivos, es muy probable que la aplicación no haya pasado los controles y verificaciones de seguridad pertinentes para detectar posibles vulnerabilidades de seguridad.
  - Las capacidades de autenticación y mantenimiento de sesiones en HTTP son muy limitadas, motivo por el que es necesario mediante software proporcionar un sistema de autenticación y de gestión de sesiones robusto. El desarrollo de este sistema puede introducir nuevas vulnerabilidades de seguridad dada su complejidad.
12. Adicionalmente a los diferentes elementos que forman parte de una arquitectura Web, existen otros elementos que son potenciales objetivos para los atacantes, tales como el tráfico Web intercambiado entre clientes y servidores, y la posibilidad de realizar ataques de denegación de servicio (DoS), afectando la disponibilidad del entorno.
13. Las vulnerabilidades de seguridad más comunes y relevantes en aplicaciones Web en los últimos años son:
- XSS, *Cross-Site Scripting*
  - CSRF, *Cross-Site Request Forgery*
  - Inyección SQL
  - Otros ataques de inyección, sobre XPath (el lenguaje de consulta de información en repositorios de XML) y LDAP (servicio de directorio)
  - Publicación de información sensible
  - HTTP Response Splitting
  - Path traversal

Cada una de estas vulnerabilidades y sus ataques asociados se describe en detalle posteriormente en esta guía.

## 4.2. INCIDENTES DE SEGURIDAD EN ENTORNOS WEB

14. Diariamente se producen numerosos incidentes de seguridad sobre entornos y aplicaciones Web. Los incidentes de mayor relevancia son publicados mediante la WHID, *Web Hacking Incidents Database*, de WASC [Ref- 2].
15. Esta base de datos recoge los detalles sobre incidentes y ataques reales sobre entornos y aplicaciones Web, e identifica cada uno de ellos de forma unívoca con un código que refleja el año en el que se produjo el incidente, y el número de incidente dentro de ese año, como por ejemplo, “WHID 2007-82”.
16. Se recomienda su consulta para disponer de una visión real y actualizada sobre los ataques e incidentes de seguridad que se están llevando a cabo en entornos Web.
17. Existe una base de datos similar, XSSed.com [Ref.- 14], dónde se publican frecuentemente incidentes asociados únicamente a vulnerabilidades de XSS, *Cross-Site Scripting*, en entornos y aplicaciones Web reales.
18. Uno de los objetivos fundamentales de todo responsable de seguridad de un entorno Web debe ser el evitar aparecer en estas bases de datos de incidentes, dónde se reflejaría el entorno Web o dominio bajo su responsabilidad como vulnerable.
19. En los últimos años los incidentes de seguridad sobre aplicaciones Web tienen una relación directa con la distribución de software malicioso (malware) y la creación de complejas infraestructuras para la realización de ataques en Internet, como *botnets* [Ref.- 3].
20. En el caso de identificar un incidente de seguridad en un entorno Web, uno de los recursos de mayor relevancia es el CCN-CERT. Según el artículo 36 del RD 3/2010, el Centro Criptológico Nacional (CCN) articulará la respuesta a los incidentes de seguridad en torno a la estructura denominada CCN-CERT (Centro Criptológico Nacional-Computer Emergency Reaction Team), que actuará sin perjuicio de las capacidades de respuesta a incidentes de seguridad que pueda tener cada administración pública y de la función de coordinación a nivel nacional e internacional del CCN.
21. La finalidad principal del CCN-CERT es contribuir a la mejora del nivel de seguridad de los sistemas de información en las administraciones públicas de España. La comunidad a la que presta servicio el CCN-CERT está constituida por el conjunto de organismos de la Administración: Administración General, Autonómica y Local.
22. El registro en el portal del CCN-CERT [Ref.- 1], permite disponer de acceso a los múltiples recursos de seguridad publicados, así como a las directrices para reportar incidentes al CCN-CERT.
23. Es recomendable disponer de una serie de pautas para la implantación de una capacidad de respuesta ante incidentes informáticos, definiendo procedimientos de actuación e identificando responsabilidades para resolver en el menor tiempo posible y del modo más efectivo el incidente. Para cumplir con este objetivo se recomienda la lectura de las siguientes guías CCN:



- CCN-STIC-810 Creación de CERTs [Ref.- 6]
- CCN-STIC-817 Gestión Incidentes de Seguridad en el ENS [Ref.- 7]

### 4.3. VULNERABILIDADES DE SEGURIDAD EN ENTORNOS WEB

24. Desde el punto de vista de las vulnerabilidades de seguridad, diariamente se publican numerosos fallos de seguridad en productos y aplicaciones Web.
25. Se recomienda consultar de forma periódica las bases de datos de vulnerabilidades para estar al día sobre los últimos ataques, incidentes y vulnerabilidades sobre entornos Web y su impacto. Las fuentes recomendadas con información detallada respecto a vulnerabilidades de seguridad son CCN-CERT [Ref.- 1], SecurityFocus [Ref.- 4] y SANS [Ref.- 5].
26. Las vulnerabilidades Web más relevantes son:
  - *Cross-Site Scripting* (XSS): la aplicación Web envía los datos proporcionados por el usuario al navegador Web sin realizar ninguna validación o codificación del contenido. Este ataque permite a un atacante ejecutar código (scripts) en el navegador de la víctima: robo de sesiones, modificación de los contenidos de la Web y su configuración.
  - Ataques de inyección: los datos proporcionados por el usuario se envían a un intérprete como parte de un comando o consulta. El ejemplo más conocido es la inyección SQL en bases de datos. El atacante puede ejecutar código dañino y modificar datos a través del intérprete atacado: SQL, XPath, LDAP, etc.
  - Ejecución de ficheros y código malicioso: el código vulnerable a la inclusión de ficheros remotos (o código remoto), *Remote File Inclusion* (RFI), permite a un atacante incluir código y datos dañinos en la aplicación Web. El resultado permite tener control completo del servidor Web. Este ataque afecta a cualquier entorno de desarrollo (eg. PHP) que acepte ficheros o nombres de ficheros de los usuarios.
  - Referencias directas a objetos inseguras: una referencia directa a un objeto ocurre si el desarrollador expone un objeto interno de la implementación (fichero, directorio, registro de la BD, clave...) en forma de URL o de parámetro de un formulario. Un atacante puede manipular estas referencias para acceder a otros objetos sin autorización.
  - *Cross Site Request Forgery* (CSRF): los ataques CSRF fuerzan al navegador de la víctima a enviar peticiones pre-autenticadas a una aplicación Web vulnerable. La petición obliga al navegador de la víctima a realizar acciones hostiles no deseadas en beneficio del atacante sobre una sesión previamente establecida. Los límites de este ataque están en el tipo de aplicación Web atacada.

- Filtrado de información y gestión incorrecta de errores: las aplicaciones Web pueden revelar información sobre su configuración, detalles internos de implementación o violar la privacidad de los datos, de forma no intencionada. El atacante emplea esta debilidad para obtener información sensible o realizar ataques más avanzados.
- Autenticación y gestión de sesiones: las credenciales de acceso y los *tokens* (o identificadores) de sesión no son protegidos adecuadamente. Un atacante puede comprometer claves, secretos e identificadores de autenticación y robar la identidad de otros usuarios.
- Almacenamiento criptográfico inseguro: la aplicación Web no utiliza funciones criptográficas adecuadamente para proteger los datos y las credenciales empleadas. Un atacante puede usar los datos no protegidos para robar la identidad de otros usuarios y realizar otros ataques, como fraude con tarjetas de crédito.
- Comunicaciones inseguras: la aplicación Web no cifra el tráfico de red correctamente cuando es necesario proteger datos y comunicaciones sensibles.
- Fallo al restringir el acceso a URLs: la aplicación Web sólo protege la funcionalidad sensible no mostrando enlaces o URLs a usuarios no autorizados. Un atacante puede usar esta debilidad para acceder directamente de forma no autorizada a esas URLs.

#### **4.4. ESTRATEGIA Y METODOLOGÍA DE SEGURIDAD DE APLICACIONES WEB**

27. La estrategia y metodología de seguridad de aplicaciones Web está formada por numerosos componentes que se complementan entre sí. Este apartado detalla los aspectos fundamentales a considerar para el diseño, desarrollo, mantenimiento y evaluación de la seguridad en entornos Web.
28. Los elementos de seguridad principales de un entorno o aplicación Web deben incluir:
  - Formación en seguridad de aplicaciones Web
  - Arquitectura e infraestructura (sistemas y redes) segura
  - Metodología de seguridad de desarrollo de aplicaciones Web
  - Metodología de análisis de seguridad de aplicaciones Web
29. La estrategia y metodología de seguridad debe incluir adicionalmente los siguientes componentes:
  - Formación en seguridad

- Instalación y configuración segura de sistemas y redes (arquitectura)
    - Actualizaciones: servidor Web y de aplicación, *framework*, etc
  - Desarrollo de software seguro
    - Gestión de versiones y actualizaciones
  - Web Application Firewalls (WAF)
  - Auditorías de seguridad
    - Caja negra: pruebas de intrusión y Web Application Security Scanners (WASS)
    - Caja blanca: revisión de código manual y automático
  - Respuesta ante incidentes
30. Para alcanzar un nivel de seguridad adecuado en el entorno o aplicación Web es necesario involucrar tanto a administradores como a desarrolladores. No es posible proporcionar un entorno Web seguro sin la colaboración de ambos grupos.
31. La formación de seguridad debe centrarse en proporcionar un conocimiento adecuado a administradores y desarrolladores respecto a las vulnerabilidades y amenazas de seguridad en entornos Web, los diferentes tipos de ataques existentes y los mecanismos de defensa asociados, preferiblemente mediante ejemplos prácticos. El objetivo es disponer del conocimiento para construir una infraestructura y aplicación Web más seguros.

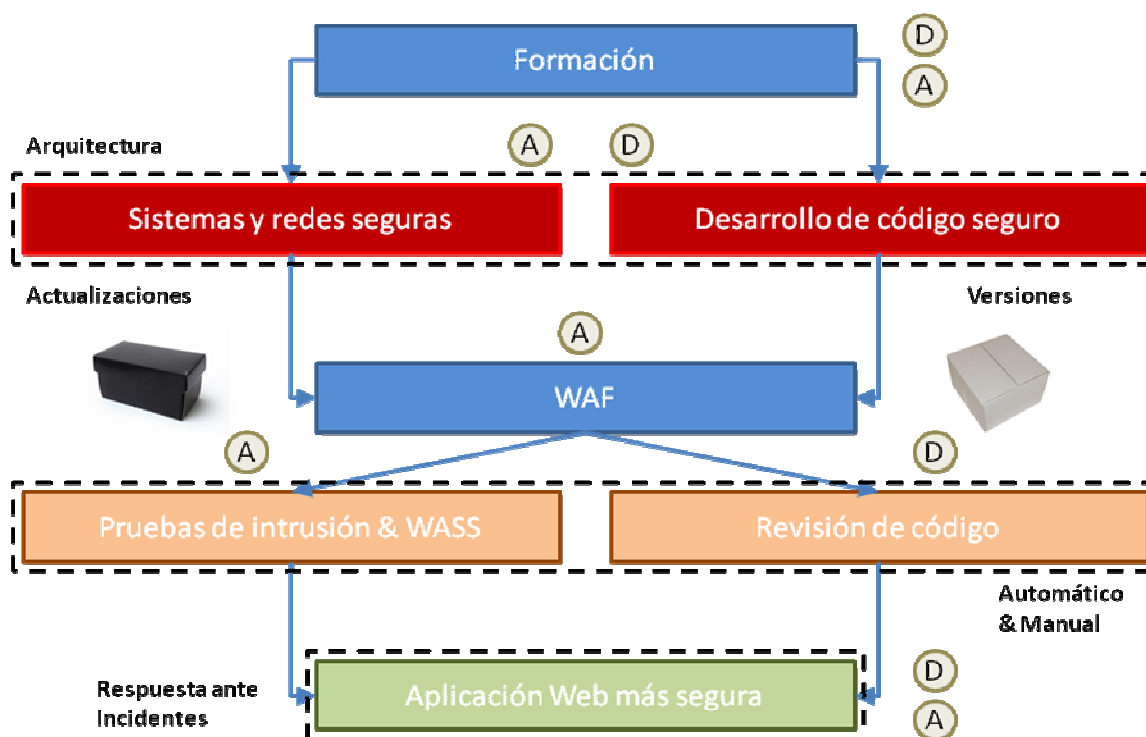


FIGURA 1.- Estrategia y metodología de seguridad de aplicaciones web

NOTA: dentro de las diferentes fases de la estrategia de seguridad de aplicaciones Web, algunas tareas están más asociadas a los administradores (A) y otras a los desarrolladores (D). Los círculos de la figura que contienen las letras A y D denotan el grupo principal al que se asocia cada tarea.

32. Entre otros, los servidores Web de una organización puede incluir:

- Servidor Web corporativo principal y públicamente disponible en Internet
- Servidores Web para terceros (socios, proveedores, clientes, etc) públicamente disponibles en Internet o desde Extranets
- Los numerosos servidores Web internos, accesibles sólo desde la red interna de la compañía o Intranet
- Servidores Web de aplicaciones comerciales: Citrix, SharePoint, VNC, etc
- Servidores Web de administración y revisión de logs
- Servidores Web de los dispositivos embebidos

## 4.5. ARQUITECTURAS DE SEGURIDAD EN ENTORNOS WEB

33. Existen fundamentalmente dos modelos de arquitectura de aplicaciones Web:
- En dos capas: donde el servidor Web y de aplicación conviven en el mismo sistema.
  - En tres capas: donde cada elemento (servidor Web, servidor de aplicación y servidor de base de datos) corresponde a un sistema independiente.
34. Desde el punto de vista de la seguridad, el modelo en tres capas es preferible ya que ofrece separación entre los distintos componentes y un mayor aislamiento frente a incidentes de seguridad en cualquiera de los elementos.
35. Adicionalmente, la arquitectura de tres capas, aunque más compleja, ofrece:
- Mayor escalabilidad para poder gestionar un mayor número de peticiones y usuarios de la aplicación.
  - La posibilidad de introducir controles de acceso avanzados, como filtrado del tráfico, y elementos avanzados de monitorización, como sistemas de detección de intrusos, entre cualesquiera elementos de la arquitectura.
  - Permite aplicar una securización más estricta sobre cada uno de los componentes, ya que cada uno tiene asignada una funcionalidad y tareas claramente definidas.
36. A la hora de diseñar una aplicación Web es necesario tener en cuenta múltiples factores: arquitectura, desarrollo de la aplicación y mecanismos de auditoría.
37. La arquitectura de la aplicación Web debe disponer de mecanismos de detección y protección a nivel de red, incluyendo elementos de seguridad tradicionales como cortafuegos o sistemas de detección de intrusos.
38. Se recomienda seguir las pautas reflejadas en las siguientes guías CCN para este tipo de elementos de seguridad perimetrales:
- CCN-STIC-661 Seguridad en Firewalls de Aplicación [Ref.- 8]
  - CCN-STIC-408 Seguridad Perimetral – Cortafuegos [Ref.- 10]
  - CCN-STIC-432 Seguridad Perimetral – IDS [Ref.- 11]
39. Uno de los elementos principales empleados en la protección de entornos ya aplicaciones Web son los cortafuegos (firewalls) de aplicaciones Web, también conocidos como WAF, *Web Application Firewall*.

40. Es necesario de disponer de dispositivos dedicados exclusivamente a la inspección y filtrado del tráfico Web, es decir, el protocolo HTTP (o HTTPS), ya que los cortafuegos tradicionales no disponen de capacidades avanzadas para analizar y bloquear los ataques recibidos a través de este protocolo.
41. Los WAF típicamente se sitúan desde el punto de vista de la arquitectura de red delante del servidor Web, con el objetivo de monitorizar y filtrar todas las peticiones maliciosas recibidas por la aplicación Web.
42. En arquitecturas Web de tres capas (servidor Web, servidor de aplicación y base de datos), existe la posibilidad de emplear WAF dedicados entre cada una de las capas.
43. Los WAF más conocidos y ampliamente utilizados sin carácter comercial son:
  - mod\_security (Apache) [Ref.- 15]
  - URLScan/Lockdown (Microsoft IIS) [Ref.- 16]
44. Algunos de los WAF más conocidos en el ámbito comercial son AppShield, InterDo, SecureSphere, F5, DenyAll, SecureIIS, Microsoft ISA, Profense, etc
45. Entre los criterios de evaluación de WAF más comunes se encuentran:
  - Modo de operación: bridge, router, proxy inverso, plug-in del servidor Web, etc
  - Gestión de SSL: terminador, capacidades de descifrado o no puede inspeccionarlo
  - Técnicas de bloqueo de tráfico
  - Tipo de solución (*appliance* o software)
  - Capacidades de reescritura de peticiones Web
  - Mecanismos de alta disponibilidad y rendimiento de la solución
  - Soporte de otros protocolos distintos a HTTP
  - Soporte de métodos de HTTP y extensiones (como WebDav)
  - Detección de ataques mediante firmas y técnicas de normalización
  - Protección frente a ataques de fuerza bruta, cookies, sesiones, campos HTML ocultos, parámetros, etc
  - Logging, notificación de alertas, informes, interfaz de gestión, , etc
46. En el caso de ser necesario asegurar la confidencialidad de las comunicaciones Web, es necesario hacer uso de la versión segura del protocolo, HTTPS.

47. HTTPS utiliza SSL (*Secure Socket Layer*) o TLS (*Transport Layer Security*) para cifrar las comunicaciones entre el navegador Web y el servidor Web.
48. Se recomienda la utilización de SSL versión 3 (no las versiones previas del protocolo) o el protocolo que lo sustituye, TLS versión 1.
49. Adicionalmente a los elementos de seguridad propios de un entorno Web, es necesario proteger todos los elementos de la infraestructura en la que reside la aplicación Web, tales como dispositivos de comunicaciones (*routers, switches, etc*) o la infraestructura de servidores de nombres (DNS).
50. Las mejores prácticas de seguridad para cada uno de los elementos que conforman el entorno Web no pueden ser detalladas específicamente en la presente guía con el objetivo de limitar su alcance. Es recomendable seguir las pautas reflejadas en las siguientes guías CCN para reforzar la seguridad de todos estos elementos:
  - CCN-STIC-5xx Conjunto de guías de seguridad de Windows [Ref.- 12]
  - CCN-STIC-6xx Conjunto de guías de seguridad de entornos Unix (HP-UX, Solaris, Linux, etc), base de datos Oracle, equipos de comunicaciones (*routers, switches, etc*), servidor DNS, servidor Web Apache y Oracle Application Server (OAS) [Ref.- 13]
51. Los mecanismos de protección a implementar deben proteger los diferentes equipos frente a:
  - Ataques directos, tales como accesos no autorizados sobre cualquiera de los elementos que conforman el entorno o aplicación Web.
  - Ataques indirectos, donde cualquiera de los elementos es empleado como herramienta en el ataque. Por ejemplo, un ataque sobre los servidores de DNS podría permitir redireccionar el tráfico de los clientes Web hacia un entorno malicioso que suplante la aplicación Web real.
  - Ataques de denegación de servicio (DoS). Las arquitecturas Web están basadas en tecnologías TCP/IP y por tanto son vulnerables a los ataques de DoS comunes en TCP/IP. Es necesario disponer de contramedidas técnicas y procedimientos de actuación frente a este tipo de ataques.
52. Se deberán aplicar los últimos parches de seguridad en cada uno de los elementos software que forman parte de la plataforma de la aplicación Web: software de los dispositivos de red y firewalls, sistema operativo de los servidores (Web, aplicación, y base de datos), y software de la plataforma de desarrollo empleada (PHP, ASP, Java, etc).
53. Desde el punto de vista de la infraestructura Web, se debe disponer de un análisis detallado del usuario, grupo, permisos y derechos con los que ejecutarán cada uno de los componentes de la aplicación Web, como por ejemplo, los procesos del servidor Web o de aplicación. Este análisis permitirá aplicar todos los mecanismos de autorización necesarios.

54. Adicionalmente, se debe disponer de un análisis detallado de las credenciales con las que unos componentes podrán acceder y obtener información de otros componentes. Por ejemplo, ¿cuáles son las credenciales y permisos de acceso del servidor de aplicación a la base de datos?

#### **4.6. DESARROLLO SEGURO DEL SOFTWARE DE APLICACIONES WEB**

55. El elemento fundamental en la metodología de seguridad de desarrollo de software de aplicaciones Web pasa por incluir todos los aspectos de seguridad en el ciclo de vida de desarrollo de software (SDLC, *Software Development Life Cycle*).
56. Las aplicaciones Web raramente son gestionadas con un sistema exhaustivo de control de versiones que permita controlar en todo momento cual es la versión existente en producción, tal y como ocurre con otro tipo de software.
57. Debido al dinamismo y la demanda de nueva funcionalidad en este tipo de aplicaciones, nuevo código y contenidos son añadidos a la aplicación Web en tiempo real sin realizar un control detallado de los cambios.

##### **4.6.1. RECOMENDACIONES GENERALES**

58. De forma general, se recomienda el uso del método GET de HTTP sólo para la consulta de información, y el método POST para el intercambio y envío de información por parte de los clientes Web a la aplicación Web.
59. Las cabeceras HTTP pueden ser manipuladas fácilmente por un atacante y no deben emplearse como método de validación o de envío de información.
60. Todos los mecanismos de interacción entre los distintos componentes del entorno Web (servidor Web, de aplicación y base de datos) deben realizarse de forma segura. Las comunicaciones entre estos elementos deberán estar cifradas, autenticadas y asegurarse su integridad.
61. El almacenamiento de información sensible, tanto propia de la lógica de la aplicación como las credenciales de acceso, debe de almacenarse cifrada en todos los servidores, y especialmente en el de base de datos.

##### **4.6.2. FILTRADO DE DATOS DE ENTRADA DEL USUARIO**

62. Es necesario ser consistente en la aplicación de los mecanismos de seguridad, como por ejemplo el filtrado de datos enviados por el usuario.



63. Algunos de los ataques Web mencionados, como por ejemplo XSS o inyección SQL, pueden ser mitigados filtrando los datos maliciosos (código HTML y scripts, o sentencias SQL, respectivamente) en la entrada de la aplicación, en la salida, o en ambas. Se recomienda, aplicando criterios de defensa en profundidad, aplicar los mecanismos de filtrado tanto en la entrada como en la salida.
64. Los datos de entrada proporcionados por el usuario (o atacante) deben ser considerados dañinos por naturaleza, por ello es necesaria su verificación y análisis antes de ser procesados por la aplicación.
65. Existen dos modelos para el filtrado de datos de entrada:
  - Eliminar los caracteres maliciosos y permitir el resto.
  - Permitir sólo los caracteres válidos para cada entrada en la aplicación.

Desde el punto de vista de la seguridad, el segundo modelo es el más restrictivo y recomendado, aunque no siempre puede ser aplicado al desconocerse el conjunto total de caracteres válidos.

66. Se recomienda establecer una única librería de código que contemple toda la funcionalidad necesaria para filtrar cualquier entrada del usuario. Toda entrada recibida por parte del usuario debe ser procesada y filtrada por esta librería, independientemente del método empleado: variables en métodos GET o POST, cookies, cabeceras HTTP (Referer, User-Agent, etc), etc
67. La disponibilidad de una única librería central permite aplicar de forma homogénea los mismos mecanismos de filtrado en cualquier parte del código de la aplicación. Para ello sólo hace falta invocar desde código a la función correspondiente de la librería.
68. Asimismo, la existencia de una única librería central simplifica el mantenimiento y las actualizaciones de los mecanismos de filtrado en la totalidad de la aplicación Web.
69. Los mecanismos de filtrado de la entrada del usuario permiten mitigar ataques como XSS, inyección SQL, desplazamiento por directorios, referencias directas a ficheros, ejecución de comandos del sistema operativo o HTTP Response Splitting. A continuación se analiza en detalle las recomendaciones de seguridad en el código de la aplicación para evitar este tipo de ataques.
70. La librería debería proporcionar funciones para filtrar la entrada del usuario según:
  - El tipo de datos: debería haber funciones sólo para letras, números, caracteres alfanuméricos, así como otros tipos de datos más concretos como fechas, DNI, números de teléfono, etc.
  - Funciones de filtrado frente a ataques de XSS:

- Se deben filtrar cualquier contenido recibido por el usuario que tenga validez en el lenguaje HTML, es decir, etiquetas (o *tags* HTML) de representación de objetos, como por ejemplo: `<script>`, `<img>`, `<a href=...>`, `<object>`, `<iframe>`, etc.
  - Adicionalmente y de manera general si es posible, se recomienda filtrar los caracteres para la creación de etiquetas HTML, “<” y “>”, y sus múltiples representaciones: `&lt;` y `&gt;`, `%60` y `%62`, `&#60;` y `&#62;`, junto a otros caracteres propios de código de scripts: `=` “ ’ ( ) ; &.
  - Los ataques de XSS no se centran únicamente en la inyección de la etiqueta `<script>`, sino que hacen uso de muchos otros elementos para referenciar scripts remotos mediante etiquetas HTML que permiten referenciar otras fuentes, “`src=`” (*source*).
  - Se recomienda filtrar no sólo la entrada del usuario, sino también la salida de la aplicación, para eliminar aquellos contenidos no deseados de la misma, como por ejemplo scripts o iframes que referencian sitios Web remotos.
- Funciones de filtrado frente a ataques de inyección SQL: el filtrado depende en gran medida del tipo de base de datos empleada.
    - Se debe filtrar el envío de caracteres especiales para la base de datos y palabras SQL reservadas, como comentarios (`--`, `#`, `/*` o `*/`), “`;`” (fin de consulta SQL), caracteres comodín (`*`, `%`, o `_`), concatenación (`+` o `||`), u operadores SQL: `OR`, `TRUE`, `1=1`, `SELECT`, `JOIN`, `UPDATE`, `INNER`, `INSERT`, `PRINT`, `waitfor delay “xx”`, etc.
    - Adicionalmente es necesario filtrar todas las representaciones de los caracteres especiales. Por ejemplo, la “`“`” es igual a `%27` o el “`=`” es `%3D`. Asimismo, deben traducirse las funciones propias de la base de datos, como la función `CHAR()`. Por ejemplo, `CHAR(77)` es igual al carácter ‘M’ (valor ASCII decimal: 77).
    - Criterios de filtrado similares deben aplicarse a la inyección de comandos en otros lenguajes de consulta, como por ejemplo LDAP, con sus operadores asociados: `AND (&)`, `OR (|)`, `NOT (!)`, `<=`, `>=`, `=` y `~=` y el carácter comodín (`*`).
    - Otro de los lenguajes de consulta comúnmente empleados en la actualidad en servicios Web es XPath, que permite la consulta de repositorios de datos en XML.

- Una de las técnicas recomendadas para evitar la inyección SQL es la utilización de procedimientos almacenados, donde la entrada del usuario se convierte en parámetros específicos del procedimiento. Deben filtrarse los parámetros correctamente y no emplear SQL dinámico en el procedimiento almacenado para evitar ataques de inyección.
- Funciones de filtrado frente al desplazamiento por directorios:
  - Se deben filtrar referencias relativas a los directorios padre, como “..” y todas sus representaciones (Unicode: %c0%af, %c1%9c, %255c,, etc), o referencias absolutas a directorios mediante “/” o “\”.
  - Ejemplo de funciones vulnerables: fopen() en PHP.
- Funciones de filtrado frente a referencias directas a ficheros (locales o remotas):
  - Debe limitarse el acceso a ficheros o recursos en URLs (es decir, remotos) como si fueran ficheros o recursos locales.
  - Ejemplo de funciones vulnerables: include() en PHP.
- Funciones de filtrado frente a ejecución de comandos del sistema operativo:
  - Debe limitarse el envío de caracteres especiales para el sistema operativo, como por ejemplo “;”, “>”, “<”, “|”, etc.
  - Ejemplo de funciones vulnerables: exec(), shell\_exec(), system(), o passthru() en PHP.
- Funciones de filtrado frente a HTTP Response Splitting:
  - Se deben filtrar los caracteres de fin de línea que podrían permitir añadir cabeceras HTTP adicionales o inyectar datos en la cabecera, tales como “\r”, “\n” o ambas:
    - \r: CR – carriage return
    - \n: LF – line feed
- Una opción común a cualquier función de filtrado en el caso de no disponer de funciones asociadas en el lenguaje o entorno de desarrollo empleados es aplicar filtros mediante expresiones regulares.
- Adicionalmente debe realizarse el filtrado de los datos de entrada frente a ataques más tradicionales, como por ejemplo desbordamientos de buffer. Para ello es necesario fijar y comprobar la longitud de los datos de entrada.

71. El filtrado de los datos de entrada del usuario se debería realizar tanto en el cliente como en el servidor, de nuevo, aplicando criterios de defensa en profundidad. En caso de ser necesario un único nivel de validación, por ejemplo por motivos de rendimiento, este siempre se llevará a cabo en el servidor, ya que la validación en el cliente puede ser fácilmente manipulada por parte del atacante.
72. Dado que es posible representar los datos de entrada a las aplicaciones Web de múltiples formas, como por ejemplo ASCII, codificación de la URL en hexadecimal (mediante el código %XX del carácter a representar), en Unicode, etc, existen ataques que pretenden evadir los filtros empleando diferentes técnicas de codificación. Por ejemplo, la herramienta “nikto” implementa múltiples técnicas de evasión de sistemas de detección de intrusos en el tráfico Web que genera.
73. Previo a la verificación y filtrado de los datos de usuario, se recomienda realizar una normalización de los mismos, en ese orden: primero se debe normalizar y posteriormente filtrar. El objetivo de la normalización es convertir los datos en un lenguaje común, en concreto, el empleado por los filtros, independientemente de cómo los haya codificado el usuario o atacante.

#### **4.6.3. MENSAJES DE ERROR Y OTROS CONTENIDOS**

74. El entorno Web debe considerar una gestión de errores adecuada, minimizando la cantidad de información que se proporciona al usuario o atacante.
75. La información contenida en los errores puede ser empleada durante el reconocimiento del entorno, y proporcionar información del software y las versiones empleadas, información del sistema de ficheros y detalles de dónde se encuentran ubicados los recursos empleados por la aplicación Web. Adicionalmente puede contener información más detallada de la base de datos que puede ser empleada por un atacante para ejecutar ataques de inyección de código más efectivos.
76. Es necesario en todo momento capturar las condiciones de error y mostrar mensajes de error personalizados con la mínima cantidad de información posible. No se recomienda mostrar directamente los mensajes de error detallados generados por el servidor Web, el servidor de aplicaciones o la base de datos.
77. La gestión de errores debe aplicarse tanto en los accesos a contenidos estáticos como dinámicos, tras la ejecución de código y scripts.
78. Con el objetivo de minimizar la cantidad de información disponible a través de la aplicación Web, se recomienda no disponer del código fuente de la aplicación en el entorno de producción, así como eliminar todos los recursos existentes por defecto en el software empleado.
79. No debe existir ningún tipo de información confidencial en los datos proporcionados al usuario, tal como campos ocultos en documentos HTML o claves de la aplicación, en los documentos enviados a los clientes Web. Si en algún caso es necesario enviar algún campo sensible hacia el cliente Web, este campo debería estar cifrado, expirar tras cierto tiempo y no ser reutilizable.

#### 4.6.4. AUTENTIFICACIÓN Y GESTIÓN DE SESIONES

80. Un caso concreto de mensajes de error son los asociados al mecanismo de autenticación. En ningún caso la aplicación debe desvelar los detalles de que componentes de las credenciales de acceso no son válidos. Por ejemplo, mensajes de error indicando que el usuario es válido, pero la clave no, facilitan enormemente a un atacante las tareas de adivinación de claves.
81. El mecanismo de autenticación debe proporcionar capacidades para definir una política de acceso, definir la longitud y complejidad de las claves, los mecanismos de log de acceso, y la política de bloqueo temporal de cuentas en base al tiempo tras un número de intentos de acceso fallidos.
82. La gestión de sesiones en la aplicación Web debe realizarse empleando identificadores de sesión o *tokens* no predecibles (es decir, suficientemente aleatorios), de suficiente longitud para que no puedan ser adivinados mediante técnicas de fuerza bruta, y que caduquen tras cierto tiempo.
83. Los identificadores de sesión pueden intercambiarse como parte de la URL (variables en el método GET), como campos HTML ocultos, *cookies* (método más comúnmente utilizado y recomendado), cabeceras HTTP estándar o propietarias, o como variables y valores en el método POST.
84. Para sistemas de nivel ALTO se deberán implantar comprobaciones de integridad sobre el identificador de sesión para detectar y evitar su manipulación. Para ello se deben emplear hashes criptográficos como MD5 o SHA-1, y cifrados, no codificados (en por ejemplo, base64). El identificador de sesión puede ser contrastado constantemente con otros datos que no son fácilmente manipulables por el usuario por parte de la aplicación, como por ejemplo la dirección IP origen empleada en el momento de la creación de la sesión.
85. Se deberá hacer un uso adecuado de los dos tipos de cookies existentes, persistentes y no persistentes, en función de su propósito y de la duración estimada de la sesión del usuario; además del uso de las capacidades extendidas en la definición de cookies, como “secure” (cookies sólo disponibles a través de SSL) o “httponly” (cookies sólo disponible para la página Web, y no para los scripts).
86. Se deberán emplear los mecanismos de gestión de sesiones existentes en el lenguaje o entorno de programación empleado. En el caso en el que sean necesarios mecanismos de seguridad más avanzados, se deberán extender las capacidades del mecanismo existente, y no construir desde cero el sistema de gestión de sesiones.

#### 4.6.5. CSRF

87. Los ataques de CSRF sólo pueden ser evitados mediante la implementación de elementos conocidos como *tokens* de formulario, o *tokens* anti-CSRF. La aplicación debe ser capaz de seguir la propia lógica de la aplicación antes de realizar una acción crítica. Es posible llevar a cabo este control mediante *tokens* dinámicos, creados por cada sesión, usuario y formulario crítico, suficientemente aleatorios, y que deben tener una fecha o tiempo de expiración.
88. Otras soluciones como el sustentar la validez de las peticiones sobre la aplicación en el campo “Referer” (cabecera HTTP) no son válidas ya que este campo puede ser fácilmente modificado por el atacante.
89. Adicionalmente, es posible evitar este tipo de ataques y proporcionar más control de la aplicación Web al usuario mediante la solicitud de verificación para acciones críticas. Una implementación común de esta verificación es el uso de imágenes CAPTCHA, dónde el usuario debe introducir el texto existente en una imagen, o la utilización de autenticación de dos factores mediante tarjetas de coordenadas o dispositivos de claves de un solo uso.

#### 4.6.6. GESTIÓN DE LOGS

90. Existen numerosos motivos para la generación, almacenamiento y gestión de logs, tales como motivos legales y/o de regulación, de contabilidad, para la resolución problemas, auditoría, estadísticas, respuesta ante incidentes y análisis forense.
91. Se recomienda la aplicación de las mejores prácticas en la gestión de logs, no sólo a nivel de los diferentes elementos que componen la plataforma del entorno Web, tales como los dispositivos de red, firewalls, IDS, WAF, servidor Web, servidor de aplicación o servidor de base de datos, sino también en el propio código de la aplicación Web.
92. Dentro de las mejores prácticas de gestión de logs es necesario considerar su rotación, ubicación (local y remota), espacio necesario en disco, permisos, rendimiento, política de retención de logs, centralización y correlación de logs, y las herramientas y soluciones para su análisis detallado (automático y manual).
93. A nivel de la aplicación Web es necesario definir qué acciones van a generar eventos de logs y con qué nivel de detalle.
94. Para generar los logs en la aplicación Web deben emplearse los mecanismos estándar del lenguaje o entorno de desarrollo Web empleado, como por ejemplo Log4J (Java), Log4PHP (PHP), etc.
95. Se recomienda centralizar la función de *logging* en una librería para la totalidad de la aplicación Web, de forma que se disponga de un mecanismo único y consistente de *logging* para el conjunto global de la aplicación. En cualquier punto del código donde sea necesario generar logs sólo será necesario invocar a las funciones correspondientes de la librería de logs.

## 4.7. ANÁLISIS DE SEGURIDAD DE APLICACIONES WEB

96. Una vez se ha implementado una metodología de desarrollo de software para aplicaciones Web que contempla los aspectos de seguridad pertinentes, es necesario complementarla con análisis y auditorías del entorno y la aplicación Web.
97. El objetivo del análisis de seguridad de una aplicación Web es pasar de una aplicación Web que presenta vulnerabilidades, a una aplicación que es segura y no tiene vulnerabilidades conocidas.
98. Es importante reseñar que la seguridad al 100% no se puede asegurar, por lo que el objetivo de la metodología de análisis es aumentar lo más posible el nivel de seguridad de la aplicación Web mediante el estudio de aquellas vulnerabilidades y errores conocidos.
99. La metodología de análisis debe incluir dos áreas (descritas en detalle a continuación):
  - Caja negra
  - Caja blanca

### 4.7.1. METODOLOGÍA DE ANALISIS DE CAJA NEGRA

100. El análisis de caja negra se centra en estudiar las vulnerabilidades de seguridad de la aplicación Web desde el punto de vista de un atacante externo.
101. El analista, actuando como atacante, no dispone de ningún tipo de información previa relativa a la aplicación y mucho menos del código de la misma.
102. El análisis se basa en la interacción del analista con la aplicación Web. Mediante la generación de diferentes estímulos o datos de entrada, se analiza la respuesta o datos de salida, con el objetivo de identificar posibles vulnerabilidades.
103. Los datos de entrada deben ser especialmente manipulados para provocar la ejecución de excepciones y condiciones no esperadas, como errores en la ejecución de scripts, errores provenientes de la base de datos, errores del servidor Web (HTTP 500), etc. Este proceso implica típicamente la utilización de caracteres especiales según los elementos que conforman la aplicación Web.
104. Ejemplos:
  - Para detectar la existencia de vulnerabilidades de inyección SQL es necesario enviar como datos de entrada caracteres especiales para la base de datos, como por ejemplo el carácter “ ’ ” (comilla simple) empleado en las consultas SQL.



- Para detectar la existencia de vulnerabilidades de XSS (*Cross-Site Scripting*) es necesario enviar como datos de entrada caracteres especiales en el procesamiento de scripts en los navegadores Web, como por ejemplo la etiqueta HTML <script> empleada en la inclusión de scripts en páginas Web.

105. La metodología a seguir es similar a la empleada en las pruebas de intrusión, donde el análisis desde el punto de vista del atacante se divide en varias fases que permiten obtener información de la aplicación y sus posibles vulnerabilidades.

106. Las fases típicas de este proceso son:

- Reconocimiento, también conocida como descubrimiento o identificación
- Enumeración o escaneo
- Detección y verificación de vulnerabilidades

107. Para el estudio necesario en cada una de estas fases existen numerosas herramientas que permiten analizar e identificar vulnerabilidades en aplicaciones Web de forma automática. Se recomienda la lectura de la siguiente guía CCN-STIC:

- CCN-STIC-818 Herramientas de Seguridad [Ref.- 9]

108. El análisis mediante herramientas automáticas debe ser complementado con análisis manuales (pruebas de intrusión), ya que dada la naturaleza y complejidad de las aplicaciones Web, se estima que estas herramientas sólo pueden encontrar entre un 20-60% de las vulnerabilidades existentes en la aplicación.

#### **4.7.2. METODOLOGÍA DE ANALISIS DE CAJA BLANCA**

109. El análisis de caja blanca se centra en estudiar las vulnerabilidades de seguridad de la aplicación Web desde el punto de vista del desarrollador.

110. El analista, actuando como desarrollador, dispone de acceso completo al código fuente de la aplicación para su revisión.

111. La metodología a seguir se centra en realizar un análisis exhaustivo del código de la aplicación en busca de funciones vulnerables o de la ausencia de métodos que permitan, por ejemplo, validar la entrada recibida por el usuario.

112. Desde un punto de vista general, a la hora de establecer controles específicos sobre una aplicación Web es necesario conocer las recomendaciones y soluciones disponibles para el lenguaje de programación empleado: Java, PHP, .NET, etc.

113. Por este motivo, es necesario disponer de documentación adicional relativa al lenguaje de programación de la aplicación Web, tal como funciones y librerías que permiten validar y normalizar las entradas del usuario en el lenguaje empleado.



114. Dentro de las áreas de revisión de código se deberá analizar el código de la aplicación Web en busca de vulnerabilidades de:

- Desbordamientos de memoria (*buffer overruns* y *overflows*)
- Inyección de comandos en el sistema operativo
- Inyección de comandos SQL en la base de datos
- Validación de los datos de entrada
- XSS (*Cross-Site Scripting*)
- CSRF, *Cross Site Request Forgery*
- Manejo y generación de errores
- Gestión de logs
- Autenticación
- Autorización
- Gestión de sesiones
- Cifrado, tanto en almacenamiento como en tránsito
- Condiciones de carrera (*race conditions*)

## 5. ESPECIFICACIÓN DE REQUISITOS DE AUDITORÍAS DE SEGURIDAD DE ENTORNOS WEB

115. El presente apartado pretende proporcionar un mayor nivel de detalle sobre la metodología a emplear en la realización de análisis de seguridad de caja negra sobre entornos Web.

116. El objetivo principal es definir los requisitos necesarios para que una auditoría de seguridad de los entornos Web de la administración llevada a cabo por terceros obtenga unos resultados que sean un reflejo completo y realista de la seguridad de los servicios Web prestados, los resultados sean relevantes desde el punto de vista práctico, y la información obtenida se trate con la reserva oportuna.

- Completitud: la auditoría debe reflejar el estado real y completo de la seguridad de los servicios Web ofrecidos por el entorno objetivo, sin importar su base tecnológica a efectos de alcance.

- Relevancia: la auditoría deberá reflejar, de forma concisa y práctica, las posibilidades, teóricas o prácticas, de que un atacante altere las características de Disponibilidad, Integridad, Confidencialidad, Autenticidad, y Trazabilidad ofrecidas por los servicios Web.
  - Secreto: se deberá comunicar toda la información personal de los terceros implicados en la auditoría. Los auditores tendrán un acuerdo de confidencialidad que reforzará en lo posible la legislación de referencia, con penalización económica en caso de incumplimiento y avales suficientes.
117. Los resultados de la auditoría deben ser reproducibles. Con este objetivo se almacenará una captura de los resultados obtenidos tras la ejecución de las diferentes pruebas. El formato de los resultados dependerá del tipo de prueba, tal como páginas Web obtenidas, capturas de tráfico en formato PCAP, listas de URLs auditadas y vulnerables, etc.
118. La información detallada contenida en el informe de resultados debe incluir las fechas concretas en las que se llevaron a cabo las pruebas.
119. Es recomendable disponer en el informe de resultados de una lista detallada de las vulnerabilidades y problemas de seguridad encontrados, clasificados por orden de criticidad.
120. El tipo de análisis de seguridad de un entorno Web planteado se centra en los aspectos técnicos y tecnológicos de la aplicación Web [Ref.- 18]. De forma complementaria es necesario analizar la inclusión de pruebas de denegación de servicio y de ingeniería social.
121. El análisis de seguridad debe identificar complementariamente alguno de los aspectos asociados a la gestión del entorno e infraestructura Web, tales como:
- La política de actualizaciones de software para todos los componentes del entorno Web.
  - El ciclo de vida de la aplicación Web y la metodología de actualizaciones, resolución de errores (*bugs*), e inclusión de nuevas funcionalidades.
  - Análisis de seguridad de los mecanismos de copia de seguridad (*backup*) de los contenidos de la aplicación Web, incluyendo todos los elementos que la componen (servidor Web, de aplicación y base de datos), así como los mecanismos de cifrado de las copias de seguridad.

## 5.1. ÁMBITO DE LA AUDITORÍA DE SEGURIDAD

122. El determinar y cerrar el ámbito o alcance de la auditoría de seguridad de un entorno Web no es una tarea sencilla debido a las relaciones existentes entre múltiples dominios y aplicaciones Web.
123. Se recomienda cerrar el alcance inicialmente mediante una lista de dominios objetivo de la auditoría. Todas las aplicaciones Web existentes en el dominio o los dominios del alcance, serían objeto de las pruebas englobadas dentro de la auditoría.
124. Adicionalmente, es posible limitar aún más el alcance en entornos muy complejos mediante la definición de un límite máximo en la profundidad del número de enlaces recorridos desde la página Web principal. Por ejemplo, la auditoría puede limitarse a los recursos y aplicaciones Web existentes en la página Web principal y hasta una distancia de 3 enlaces para sus organismos dependientes.
125. Las auditorías así definidas se realizarán sobre un nivel de profundidad de enlaces de 3 niveles, tanto en el dominio y subdominios principales como en los organismos dependientes accesibles dentro de esos 3 niveles, sin importar el puerto que presta el servicio y si es HTTP o HTTPS.
126. Serán objeto de análisis todos los subdominios detectados a partir del dominio objetivo inicial, salvo excepciones explícitas reflejadas en el alcance. Las páginas que se analizarán serán las que se puedan encontrar en estos dominios y subdominios, independientemente de si están enlazadas.
127. Las fases de la metodología de análisis recomendada son:
- Reconocimiento, también conocida como descubrimiento o identificación
  - Enumeración o escaneo
  - Análisis (detección y verificación) de vulnerabilidades
128. Cada una de estas fases deben incluir información detallada sobre los elementos y fuentes de datos relacionadas con el entorno o aplicación Web objetivo de la auditoría de seguridad.

## 5.2. RECONOCIMIENTO

### 5.2.1. INFORMACIÓN DE REGISTRO DE DOMINIOS (DNS) Y DIRECCIONES

129. Es necesario analizar en detalle toda la información disponible en los servicios de registro de dominios y rangos de direcciones IP.
130. La información está disponible a través del servicio WHOIS, de los registradores de dominios del nivel de nombres principal, y del proveedor del rango de direcciones IP.

131. Información administrativa contenida en el servicio de nombres (DNS): personas de contacto, servidores de nombres, dominios, etc.
132. Seguridad de los DNS: versión del software de DNS, vulnerabilidades de seguridad asociadas al software de DNS, tipos de registros de DNS, transferencias de zona (por dominio y por rango de direcciones IP), etc.

### 5.2.2. SERVICIOS DE BÚSQUEDAS EN INTERNET

133. Información confidencial y sensible en buscadores accesibles públicamente (Google, Yahoo, etc) mediante técnicas de búsqueda avanzadas, conocidas como “Google Hacking”.
134. Identificación de relaciones con otras organizaciones, entornos Web y dominios.
135. Obtención de un mapa Web completo en modo texto y/o gráfico del entorno objetivo, separando claramente partes estáticas y dinámicas.
136. Análisis de correspondencias y discrepancias entre el mapa Web obtenido previamente y el mapa descrito en el sitio Web objetivo.

### 5.2.3. UBICACIÓN EN LA RED

137. Información de la ubicación en la red del entorno Web objetivo y tráfico ICMP permitido.
138. Identificación de los sistemas de comunicaciones y dispositivos de red: routers, balanceadores, etc.
139. Identificación de los sistemas de protección de perímetro (firewalls, IDS, etc).

## 5.3. ESCANEEO

140. Es necesario obtener la mayor información posible sobre los servicios y recursos del entorno Web objetivo, incluyendo tanto su ubicación en la red como los detalles de los elementos que lo conforman.
141. Enumeración de los servicios disponibles en el entorno objetivo, obtenida mediante escaneos de puertos (TCP y UDP) exhaustivos.
142. Información detallada de los servicios disponibles en el entorno objetivo, obtenida mediante técnicas de *fingerprinting* sobre cada servicio/puerto (TCP y UDP) descubierto.
143. Identificación de la plataforma y sistema operativo (OS *fingerprinting*) de los servidores objetivo, al menos, servidor Web, servidor de aplicación y base de datos.

144. Identificación de la política de tráfico permitido en los sistemas de protección de perímetro.

### 5.3.1. SERVICIOS WEB

145. Identificación e información sobre el tipo de servicios, aplicación y versión para el servidor Web, servidor de aplicación y base de datos.
146. Soporte de los métodos HTTP soportados por el entorno Web. Algunos métodos, como TRACE, están asociados a ataques Web conocidos, como XST, *Cross-Site Tracing*.
147. Soporte SSL del servidor Web: versión de SSL, protocolos soportados y versiones de SSL o TLS, algoritmos de cifrado e integridad soportados (incluyendo longitudes de clave de cifrado admitidas), certificados digitales y validez (expiración del certificado), Autoridad Certificadora (CA), etc.

### 5.3.2. CONTENIDOS WEB

148. Identificación de recursos existentes en el entorno Web, tanto enlazados como adivinados/obtenidos por fuerza bruta o técnicas de diccionario.
149. Información de la estructura de la Web de los dominios objetivo, diferenciando contenidos y recursos estáticos y dinámicos.
150. Listado de las páginas dinámicas detectadas, sus parámetros de entrada, los tipos de los parámetros y el método de transferencia al servidor.
151. Identificación de los lenguaje(s) y entorno(s) de programación empleado(s).
152. Acceso y análisis de los códigos de error generados por la aplicación Web.
153. Identificación de las extensiones de ficheros empleadas en los recursos Web y la gestión de las diferentes extensiones de ficheros.
154. Identificación de contenidos por defecto propios de las tecnologías empleadas en el entorno Web.
155. Acceso público a páginas administrativas, de gestión, estadísticas, etc.
156. Identificación de recursos con control de accesos, es decir, que requieren autenticación (páginas de *login*).
157. Identificación de copias de seguridad/versiones anteriores de recursos accesibles, tanto enlazadas como adivinadas/obtenidas por fuerza bruta o técnicas de diccionario.
158. Identificación de relaciones con otros entornos y aplicaciones Web, y análisis de seguridad en la invocación y referencias a servicios Web tanto internos y externos.

159. Análisis de los mecanismos de control de almacenamiento de contenidos en cachés y dispositivos de red intermedios, tales como proxies.
160. Análisis de los mecanismos de control de publicación de contenidos en los servidores de búsqueda de Internet: Google, Yahoo, etc.
161. Análisis de los mecanismos existentes centrados en ofrecer una superficie de exposición mínima a la totalidad del entorno Web.
162. Identificación de las capacidades de la base de datos, contenidos y funcionalidad disponible por defecto.

## 5.4. ANÁLISIS DE VULNERABILIDADES

### 5.4.1. VULNERABILIDADES DE APLICACIONES WEB

163. Identificación de vulnerabilidades de filtrado mediante el chequeo de parámetros de entrada en los componentes dinámicos de la aplicación, comprobando la existencia de vulnerabilidades de los siguientes tipos:
  - Inyección SQL
  - Inyección SQL ciega, incluyendo análisis de las diferencias en los tiempos de respuesta
  - Inyección LDAP y XPath
  - XSS, reflejado y persistente
  - CSRF
  - HTTP Response Splitting
  - Inyección de comandos en el sistema operativo
  - Desplazamiento por directorios
  - Referencias directas a ficheros
  - Parámetros y contenidos reflejados en la respuesta del servidor Web
  - Métodos para evitar comprobaciones de tipo CAPTCHA
  - Desbordamiento de buffers
164. Como resultado de estas operaciones y comprobaciones se obtendrá una lista de los parámetros chequeados y sus resultados.

## 165. Identificación de vulnerabilidades en los mecanismos de autenticación:

- Descripción de la seguridad de los métodos de autenticación
- Transporte de credenciales sobre canales de comunicación seguros
- Determinación de los elementos de protección frente a ataques de enumeración y adivinación de credenciales: número máximo de intentos de acceso fallidos, limitaciones por tiempo, etc
- Identificación de escenarios de denegación de servicio por bloqueo de cuentas de usuarios tras un número determinado de intentos de acceso fallidos
- Análisis de la fortaleza de las claves y de los mecanismos de generación de claves por defecto
- En el caso de emplearse certificados de cliente, análisis de los certificados digitales, y de los procedimientos de gestión de los certificados (alta, verificación, revocación de certificados, etc)
- Ataques de diccionario y fuerza bruta sobre las credenciales de acceso
- Determinación de métodos para evitar el sistema de autenticación
- Determinación de los mecanismos de renovación de claves e identificación de vulnerabilidades en los mismos

## 166. Análisis de seguridad en los mecanismos de control de sesiones:

- Descripción de la seguridad de las sesiones
- Determinación de la duración y ámbito de las sesiones
- Determinación de los elementos empleados para implementar el mantenimiento de sesiones
- Determinación del formato y contenido del identificador o *token* de las sesiones
- Identificación y análisis de seguridad del uso de *tokens* (cookies, variables, cabeceras HTTP, etc) por parte de la aplicación Web
- Determinación de mecanismos de manipulación del *token* de sesión
- Identificación de ataques de fijación de sesión
- Identificación y análisis de seguridad de escenarios *Single Sign On* (SSO)
- Determinación del mecanismo de cierre de sesiones y gestión de la información de sesión cacheada en los clientes Web

167. Análisis de los mecanismos de control de acceso (ACLs):

- Acceso a contenidos de usuarios conocidos o típicos (existentes por defecto)
- Acceso a ficheros de configuración del entorno Web
- Acceso a versiones renombradas (*bakups*) de archivos en producción
- Acceso a la información de otros usuarios con credenciales
- Trazabilidad de los accesos de usuario
- Determinación de métodos para evitar el sistema de autorización
- Escalada de privilegios

168. En todos aquellos entornos Web dónde existe el concepto de usuarios autenticados, se recomienda la realización de los análisis de seguridad desde dos vertientes:

- Mediante un usuario externo sin credenciales de acceso
- Mediante un usuario con credenciales de acceso, y que por tanto puede acceder a contenidos protegidos o privados

#### **5.4.2. PRUEBAS DE CARGA Y DENEGACIÓN DE SERVICIO (DOS)**

169. Se recomienda incluir en el alcance de la auditoría, como elemento opcional, la realización de pruebas de carga que podrían provocar una denegación de servicio (DoS) sobre el entorno objetivo.

170. Este tipo de pruebas debe de ser meticulosamente planeado, especialmente sobre entornos en producción, para evitar problemas de disponibilidad en el servicio del entorno Web.

171. Pueden realizarse dos tipos de pruebas de carga:

- DoS simple: empleando un único cliente, y definiendo el ancho de banda disponible, por ejemplo 20Mbps.
- DoS distribuida (DDoS): empleando múltiples clientes, por ejemplo 20, y definiendo el ancho de banda disponible para cada uno de ellos, por ejemplo 20Mbps.

172. Las pruebas de DDoS requieren disponer de elevados recursos para su consecución, por lo que no siempre es posible su realización.



173. Adicionalmente, los ataques de DoS pueden incluir otro tipo de pruebas relacionadas:

- Bloqueo de cuentas mediante intentos de acceso fallidos
- Existencia de errores no recuperables en la aplicación, como desbordamiento de buffers
- Reserva de recursos en base a las peticiones de un usuario, y errores de liberación de recursos
- Existencia de bucles de proceso infinitos en función de los parámetros de entrada
- Consumo excesivo de recursos en los elementos de búsqueda
- Diferencias de demanda de recursos en los diferentes componentes del entorno Web: servidor Web, de aplicación o base de datos.
- Denegación de servicio por rellenado automático de formularios
- Consumo de recursos de almacenamiento (espacio en disco) en base a las peticiones de un usuario

## ANEXO A. REFERENCIAS PRINCIPALES

Las tres organizaciones de mayor renombre centradas en la seguridad de las aplicaciones Web son:

- Web Application Security Consortium (WASC):  
<http://www.webappsec.org>
- Open Web Application Security Project (OWASP):  
<http://www.owasp.org>
- SANS Software Security Institute (ésta cubre no sólo el desarrollo seguro de aplicaciones Web, sino también el desarrollo de cualquier otro tipo de aplicaciones software)  
<http://www.sans-ssi.org>

## ANEXO B. LISTA DE COMPROBACIÓN Y REQUISITOS

El objetivo de este apartado es proporcionar una herramienta que permita la evaluación sistemática de un entorno u aplicación Web antes de su desarrollo y adquisición.

Mediante esta lista se pretende evaluar las capacidades y elementos de seguridad empleados internamente o por un tercero en el desarrollo de aplicaciones Web, dependiendo del nivel en el que se ha categorizado el sistema.

		CATEGORÍA DEL SISTEMA			COMPROBACIÓN
		BAJA	MEDIA	ALTA	
<b>GESTIÓN DE RIESGOS Y AMENAZAS WEB</b>					
1.	Análisis de Riesgos no formal.	X			
2.	Análisis de Riesgos semi formal.		X		
3.	Análisis de Riesgos formal.			X	
<b>INCIDENTES DE SEGURIDAD EN ENTORNOS WEB</b>					
4.	Sistema de registro de incidentes de seguridad.		X	X	
<b>ESTRATEGIA Y METODOLOGÍAS EN ENTORNOS WEB</b>					
5.	Metodología de Seguridad en el desarrollo de aplicaciones web.		X	X	
6.	El almacenamiento de información sensible, tanto propia de la lógica de la aplicación, debe de almacenarse cifrada en todos los servidores, y especialmente en el de base de datos.			X	
7.	El almacenamiento de las credenciales de acceso deberá realizarse en forma de hash.			X	
<b>ARQUITECTURA DE LA APLICACIÓN WEB</b>					
8.	Deben existir recomendaciones de seguridad asociadas a los dispositivos y entorno de red necesario para el despliegue de la aplicación Web: <i>switches, routers, firewalls, etc.</i>	X	X	X	
9.	Deben existir recomendaciones de seguridad asociadas a los servidores y sistema operativo de los equipos necesarios para implantar la aplicación Web: servidor Web, servidor de aplicaciones o servidor de base de datos.	X	X	X	
10.	Deben existir recomendaciones respecto a las actualizaciones de seguridad de los diferentes elementos software que conforman la plataforma de la aplicación Web: software de los equipos de red, sistema operativo de los servidores, software del entorno Web, etc.	X	X	X	
11.	Determinar el tipo de arquitectura se emplea en la aplicación Web, dos o tres capas, así como el				

	porqué y los aspectos considerados para aplicar este diseño.		X	X	
12.	Deben existir elementos de detección (sistema de detección de intrusos) y de protección (firewall de aplicación Web, WAF) en la arquitectura. Identificar dónde están situados en la topología de red y la justificación de esa ubicación.	X	X	X	
13.	Deben existir elementos de detección (sistema de detección de intrusos) y de protección (firewall de aplicación Web, WAF) específicos entre el servidor Web y el de aplicaciones. Verificar si se dispone de estos elementos.	X	X	X	
14.	Deben existir elementos de detección (sistema de detección de intrusos) y de protección (firewall de aplicación Web, WAF) específicos entre el servidor de aplicaciones y el de base de datos. Verificar si se dispone de estos elementos.			X	
15.	Identificar qué dispositivo de tipo WAF se emplea y por qué, así como los criterios empleados para su elección.	X	X	X	
16.	Se deberá utilizar tráfico cifrado HTTPS en la aplicación web, e identificar si existen dispositivos de seguridad para el análisis de este tipo de tráfico.		X	X	
17.	Identificar cuál es el servidor Web empleado para la aplicación Web: Apache, IIS, etc, y verificar cuáles son las recomendaciones de seguridad para la instalación y configuración del servidor Web.	X	X	X	
18.	Identificar cuál es el servidor de aplicación empleado para la aplicación Web: Tomcat, WebSphere, Weblogic, Jakarta, etc, y verificar cuáles son las recomendaciones de seguridad para la instalación y configuración del servidor de aplicación.	X	X	X	
19.	Identificar cuál es el servidor de base de datos empleado para la aplicación Web: MySQL, Microsoft SQL Server, Oracle, DB2, PostgreSQL, etc, y verificar cuáles son las recomendaciones de seguridad para la instalación y configuración del servidor de base de datos.	X	X	X	
20.	Comprobar si se emplean aplicaciones Web de terceros para proporcionar parte de la infraestructura básica de la aplicación Web, tal como PHPBB, WordPress, etc. En caso afirmativo, verificar cuáles son las recomendaciones de seguridad para la instalación y configuración del	X	X	X	

	software Web de terceros.				
21.	Identificar si se emplea HTTPS para el acceso a todos los recursos confidenciales de la aplicación, tales como interfaces de administración.	X	X	X	
22.	Identificar si se emplea HTTPS para todas las transacciones Web que requieren reforzar la confidencialidad de la comunicación, tales como sesiones de usuario con información sensible.		X	X	
23.	Verificar si se emplea SSL o TLS en algún módulo de la aplicación y el porqué. Comprobar la versión concreta del protocolo empleada.		X	X	
24.	Verificar si existe algún mecanismo en el diseño o arquitectura de la aplicación para hacer frente a ataques de denegación de servicio (DoS).			X	
25.	Verificar si se aplican en la infraestructura de filtrado tanto filtros de tráfico de entrada ( <i>ingress</i> ) como de salida ( <i>egress</i> ).		X	X	
26.	Identificar si se dispone de información detallada sobre las credenciales y permisos de acceso necesarios para la ejecución de cada componente de software existente en todos los servidores, así como para las interacciones entre éstos.	X	X	X	
<b>DESARROLLO SEGURO DE LA APLICACIÓN WEB</b>					
27.	Verificar si existe una metodología para incluir la seguridad como elemento fundamental del ciclo de vida de desarrollo de software, como por ejemplo, CLASP de OWASP. Obtener detalles sobre la metodología empleada y las acciones que conlleva.	X	X	X	
28.	Identificar el entorno de desarrollo empleado en la aplicación Web: PHP, ASP, ASP .NET, Java (J2EE, JSP...), Python, Ruby on Rails, Perl, etc, así como las recomendaciones de seguridad específicas para la instalación y configuración del entorno de desarrollo.	X	X	X	
29.	Identificar el lenguaje de programación empleado en la aplicación Web, en la parte servidor: PHP, ASP, ASP .NET, Java, Python, Ruby, Perl, etc, así como la metodología de seguridad específica empleada en el desarrollo de código mediante el/los lenguaje/s empleados.	X	X	X	
30.	Identificar el lenguaje de programación empleado en la aplicación Web, en la parte				

	cliente: AJAX, JavaScript, VBScript, ActiveX, Flash, Java <i>applets</i> , etc, así como la metodología de seguridad específica empleada en el desarrollo de código mediante el/los lenguaje/s empleados.	X	X	X	
31.	Verificar si el desarrollo contempla la protección frente a las vulnerabilidades Web más comunes.		X	X	
32.	En caso de respuesta afirmativa en el punto anterior, identificar qué metodología se emplea para desarrollar una aplicación Web segura y protegida frente a las vulnerabilidades más comunes.		X	X	
33.	Comprobar si existe algún mecanismo de soporte tras el paso a producción de la aplicación que permita obtener actualizaciones software que resuelvan nuevas vulnerabilidades de seguridad encontradas en la aplicación Web.	X	X	X	
34.	Identificar el modelo de filtrado se emplea en la aplicación (eliminación de caracteres maliciosos o aceptación de caracteres válidos únicamente) y los motivos para la aplicación de este modelo.			X	
35.	Identificar los mecanismos de filtrado de los datos del usuario empleados por todos los componentes de la aplicación Web que deben interactuar con el cliente.			X	
36.	Verificar si se normaliza la entrada del usuario antes de proceder a su verificación y filtrado.	X	X	X	
37.	Identificar los métodos de normalización contemplados en la aplicación: ASCII, codificación de URLs en hexadecimal, Unicode, etc.	X	X	X	
38.	Verificar si se dispone de una librería única y centralizada que proporcione todas las funciones de normalización y filtrado de los datos recibidos por el usuario. Asimismo, identificar para que tipo de ataques proporciona funciones específicas esta librería: XSS, inyección, referencias a ficheros, etc, así como el conjunto de caracteres que se filtran o aceptan para cada tipo de ataque.			X	
39.	Verificar dónde se realiza el filtrado de los datos de entrada: en el cliente, en el servidor o en ambos, así como el motivo por el que se			X	

	aplica un método u otro.				
40.	Verificar si se han implementado mecanismos de protección en la aplicación frente a ataques de CSRF, como por ejemplo <i>tokens</i> de formulario.			X	
41.	Verificar si se implementan mecanismos avanzados de verificación de operaciones y acciones críticas, como por ejemplo CAPTCHA o tarjetas de coordenadas.			X	
42.	Verificar si se dispone de un sistema de gestión de versiones para la aplicación Web (así como un programa más general de gestión de cambios). Obtener detalles de este sistema y de como se emplea en el control de vulnerabilidades de seguridad y en la generación de actualizaciones y nuevas versiones.	X	X	X	
43.	Comprobar si existe un criterio para la utilización de métodos HTTP GET y POST en los diferentes componentes de la aplicación Web.		X	X	
44.	Comprobar si se emplean las cabeceras HTTP por parte de la aplicación Web como mecanismo de envío de información confidencial o de verificación del estado del cliente.	X	X	X	
45.	Verificar si se hace uso de peticiones y capacidades asíncronas, AJAX, en la aplicación Web, así como los mecanismos de seguridad empleados para proteger este tipo de comunicaciones.	X	X	X	
46.	Identificar los mecanismos y tecnologías que se emplean para la autenticación de usuarios: métodos de autenticación HTTP básico o <i>digest</i> , NTLM, autenticación mediante certificados SSL (cliente y/o servidor), autenticación mediante formularios y base de datos, o autenticación de múltiples factores.		X	X	
47.	Identificar si se hace uso de las capacidades de autenticación del entorno de desarrollo Web empleado. Obtener detalles sobre los mecanismos de autenticación empleados.	X	X	X	
48.	Identificar qué mecanismos y tecnologías se emplean para el mantenimiento de sesiones de usuario: URLs, cookies, cabeceras HTTP	X	X	X	

	propietarias, etc.				
49.	Verificar si se hace uso de las capacidades de mantenimiento de sesiones del entorno de desarrollo Web empleado. Obtener detalles sobre los mecanismos de gestión de sesiones empleados.	X	X	X	
50.	Verificar cómo se lleva a cabo la gestión de errores de las tres capas: servidor Web, de aplicación y de base de datos, así como si existe una librería centralizada para la gestión y control de los errores.			X	
51.	Identificar cómo se realiza la gestión del código fuente de la aplicación en el entorno de producción, así como de los recursos existentes por defecto en el software empleado, con el objetivo de minimizar la información disponible.			X	
52.	Identificar si se realiza algún tipo de comprobación para asegurarse de que no existe ningún tipo de información confidencial, tal como campos ocultos en documentos HTML o claves de la aplicación, en todos los documentos y datos enviados a los clientes Web.	X	X	X	
53.	Verificar qué mecanismos se han implantado a nivel de la aplicación Web para la generación, almacenamiento y gestión de logs, así como si se dispone de una librería única y centralizada para la gestión de logs en la totalidad de la aplicación.		X	X	
54.	Identificar qué entorno de generación y gestión de logs se ha utilizado para implementar el mecanismo de <i>logging</i> de la aplicación Web.		X	X	
55.	Identificar qué flexibilidad existe para determinar de qué eventos generar logs, así como el nivel de detalle de los logs. Obtener un listado detallado con todos los campos estándar de cualquier entrada de log estándar.			X	
<b>AUDITORÍAS DE SEGURIDAD DE LA APLICACIÓN WEB</b>					
56.	Verificar si se realiza, o se ha planificado la realización de, una auditoría de seguridad sobre la aplicación Web una vez completado su desarrollo y antes de su puesta en producción. Identificar el tipo de auditoría: caja blanca,	X	X	X	



	negra o ambas.				
57.	Verificar si se realizan auditorías de seguridad sobre la aplicación Web durante las fases de desarrollo. Identificar el tipo (caja blanca, negra o ambas) y con qué frecuencia se realizan.	X	X	X	
58.	Para las auditorías de caja negra, verificar si se emplean herramientas automáticas, pruebas manuales o ambas.	X	X	X	
59.	Identificar la metodología que se sigue en las auditorías de caja negra.			X	
60.	Contrastar la metodología y elementos de la auditoría de caja negra con el listado de recomendaciones proporcionado en el apartado “ESPECIFICACIÓN DE REQUISITOS DE AUDITORÍAS DE SEGURIDAD DE ENTORNOS WEB” de la presente guía.			X	
61.	Para las auditorías de caja blanca, verificar si se emplean herramientas automáticas, pruebas manuales o ambas.			X	
62.	Identificar la metodología que se sigue en las auditorías de caja blanca.			X	
63.	Identificar qué herramientas de tipo WASS se emplean y el por qué, así como cuáles han sido los criterios empleados para su elección.			X	
64.	Identificar qué herramientas de análisis de código se emplean y el por qué, así como cuáles han sido los criterios empleados para su elección.			X	
65.	Identificar qué pautas, cláusulas y requisitos se aplican para la evaluación y contratación de una aplicación Web a un tercero.	X	X	X	
<b>FORMACIÓN Y CONCIENCIACIÓN</b>					
66.	Se deberán realizar cursos de formación de seguridad centrados en proporcionar un conocimiento adecuado a administradores y desarrolladores respecto a las vulnerabilidades y amenazas de seguridad en entornos Web, los diferentes tipos de ataques existentes y los mecanismos de defensa asociados.	X	X	X	
67.	Se deberán realizar tareas de concienciación en materia de seguridad, relativas al desarrollo seguro de aplicaciones y servicios Web.	X	X	X	

## ANEXO C. GLOSARIO DE TÉRMINOS Y ABREVIATURAS

ACL	Access Control List
AJAX	Asynchronous JavaScript and XML
ASP	Active Server Pages
BD	Base de datos
CA	Certification Authority
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CCN	Centro Criptológico Nacional
CCN-CERT	Centro Criptológico Nacional – Computer Emergency Response Team
CLASP	Comprehensive, Lightweight Application Security Process
CMS	Content Management Systems
CR	Carriage Return
CSRF	Cross-Site Request Forgery
DDoS	Distributed Denial of Service
DNS	Domain Name Service (o System)
DoS	Denial of Service
FAQ	Frequently Asked Questions
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	Secure Hyper Text Transfer Protocol
IDS	Intrusion Detection System
IIS	Internet Information Server
J2EE	Java 2 Enterprise Edition
JSP	Java Server Pages
LAMP	Linux, Apache, MySQL y PHP

LDAP	Lightweight Directory Access Protocol
LF	Line Feed
OWASP	Open Web Application Security Project
PCAP	Packet Capture
PHPBB	PHP Bulletin Board
RFI	Remote File Inclusion
SDLC	Software Development Life Cycle
SQL	Structured Query Language
SSL	Secure Socket Layer
SSO	Single Sign On
STIC	Seguridad de las Tecnologías de la Información y las Comunicaciones
TIC	Tecnologías de la Información y las Comunicaciones
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VNC	Virtual Network Computing
VoIP	Voz sobre IP
WAF	Web Application Firewall
WASC	Web Application Security Consortium
WASS	Web Application Security Scanner
Webmail	Correo electrónico accesible vía Web
WHID	Web Hacking Incidents Database
XML	eXtended Markup Language
XSS	Cross-Site Scripting

## ANEXO D. REFERENCIAS

- [Ref.- 1] CCN-CERT  
URL: <https://www.ccn-cert.cni.es>
- [Ref.- 2] “WHID, Web Hacking Incidents Database”. WASC  
URL: <http://www.webappsec.org/projects/whid/>
- [Ref.- 3] Artículos Known Your Enemy (KYE) relacionados con entornos Web. Honeynet Project:  
- “Web Application Threats”  
URL: <http://www.honeynet.org/papers/webapp/>  
- “Malicious Web Servers”  
URL: <http://www.honeynet.org/papers/mws/>  
- “Behind the Scenes of Malicious Web Servers”  
URL: <http://www.honeynet.org/papers/wek/>
- [Ref.- 4] Security Focus Newsletter.  
URL: <http://www.securityfocus.com/newsletters/>
- [Ref.- 5] SANS Newsletters: @Risk.  
URL: <http://www.sans.org/newsletters/>
- [Ref.- 6] CCN-STIC-810 Creación de CERTs. CCN  
URL: [https://www.ccn-cert.cni.es/publico/seriesCCN-STIC/series/800-Esquema\\_Nacional\\_de\\_Seguridad/810\\_Creacion-de-CERTs\\_sep-11.pdf](https://www.ccn-cert.cni.es/publico/seriesCCN-STIC/series/800-Esquema_Nacional_de_Seguridad/810_Creacion-de-CERTs_sep-11.pdf)
- [Ref.- 7] CCN-STIC-817 Gestión de Incidentes de Seguridad en el ENS. CCN  
URL: <https://www.ccn-cert.cni.es/ens>
- [Ref.- 8] CCN-STIC-661 Seguridad en Firewalls de Aplicación. CCN  
URL: <https://www.ccn-cert.cni.es/protegido/ccn-stic/CCN-STIC-661.htm>
- [Ref.- 9] CCN-STIC-818 Herramientas de Seguridad. CCN  
URL: <https://www.ccn-cert.cni.es/ens>
- [Ref.- 10] CCN-STIC-408 Seguridad Perimetral - Cortafuegos. CCN  
URL: <https://www.ccn-cert.cni.es/protegido/ccn-stic/CCN-STIC-408.htm>
- [Ref.- 11] CCN-STIC-432 Seguridad Perimetral – IDS. CCN  
URL: <https://www.ccn-cert.cni.es/protegido/ccn-stic/CCN-STIC-432.htm>
- [Ref.- 12] CCN-STIC-5xx Guías de seguridad de Windows. CCN  
URL: <https://www.ccn-cert.cni.es/protegido/ccn-stic/500windows.htm>
- [Ref.- 13] CCN-STIC-6xx Guías de seguridad de múltiples entornos. CCN  
URL: <https://www.ccn-cert.cni.es/protegido/ccn-stic/600entornos.htm>
- [Ref.- 14] XSSed.com  
URL: <http://xssed.com>
- [Ref.- 15] mod\_security  
URL: <http://www.modsecurity.org>
- [Ref.- 16] IIS Lockdown. Microsoft  
URL: <http://www.microsoft.com/technet/security/tools/locktool.mspx>

- [Ref.- 17] OWASP Secure Software Contract Annex  
URL: [http://www.owasp.org/index.php/OWASP\\_Secure\\_Software\\_Contract\\_Annex](http://www.owasp.org/index.php/OWASP_Secure_Software_Contract_Annex)  
URL: [http://www.owasp.org/index.php/OWASP\\_Legal\\_Project](http://www.owasp.org/index.php/OWASP_Legal_Project)
- [Ref.- 18] Penetration Testing Framework 0.50. Kevin Orrey  
URL: <http://www.vulnerabilityassessment.co.uk/Penetration%20Test.html>