Estándar de Verificación de Seguridad en Aplicaciones 3.0.1

Versión en español: Abril de 2017

Agradecimientos

Sobre el estándar

El estándar de verificación de seguridad en aplicaciones es una lista de requerimientos de seguridad o pruebas que pueden ser utilizadas por arquitectos, desarrolladores, testers, profesionales de seguridad e incluso consumidores, para definir tan segura es una aplicación.

Copyright y licencia

Versión 3.0, 2015

Líderes de Proyecto	Autores Líderes	Revisores y Colaboradores
Andrew van der Stock Daniel Cuthbert	Jim Manico	Boy Baukema Ari Kesäniemi Colin Watson François-Eric Guyomarc'h Cristinel Dumitru James Holland Gary Robinson Stephen de Vries Glenn Ten Cate Riccardo Ten Cate Martin Knobloch Abhinav Sejpal David Ryan Steven van der Baan Ryan Dewhurst Raoul Endres Roberto Martelloni
Traducción al español		Nicolás Levin Gerardo Canedo

Versión 2.0, 2014

Líderes de Proyecto	Autores Líderes	Revisores y Colaboradores
Daniel Cuthbert Sahba Kazerooni	Andrew van der Stock Krishna Raja	Antonio Fontes Colin Watson Jeff Sergeant Pekka Sillanpää Archangel Cuison Dr Emin Tatli Jerome Athias Safuat Hamdy

Líderes de Proyecto	Autores Líderes	Revisores y Colaboradores
		Ari Kesäniemi Etienne Stalmans Jim Manico Scott Luc Boy Baukema Evan Gaustad Mait Peekma Sebastien Deleersnyder

Versión 1.0, 2009

Líderes de Proyecto	Autores Líderes	Revisores y Colaboradores
Mike Boberski Jeff Williams Dave Wichers	Jim Manico	Andrew van der Stock Dr. Sarbari Gupta John Steven Pierre Parrend Barry Boyd Dr. Thomas Braun Ken Huang Richard Campbell Bedirhan Urgun Eoin Keary Ketan Dilipkumar Vyas Scott Matsumoto Colin Watson Gaurang Shah Liz Fong Shouvik Bardhan Dan Cornell George Lawless Mandeep Khera Stan Wisseman Dave Hausladen Jeff LoSapio Matt Presson Stephen de Vries Theodore Winograd Jeremiah Grossman Nam Nguyen Steve Coyle Dave van Stein John Martin Paul Douthit Terrie Diaz

Introducción

Bienvenido a la versión 3.0 del Estándar de Verificación Seguridad en Aplicaciones (ASVS por sus siglas en inglés). El ASVS es un esfuerzo comunitario por establecer un marco de referencia para los requisitos de seguridad, controles funcionales y no funcionales necesarios al diseñar, desarrollar y testear aplicaciones web modernas.

ASVS v3.0 es la culminación de un esfuerzo comunitario y de retroalimentación por parte de la industria. En esta versión, nos pareció importante estudiar las experiencias de casos de uso del mundo real relacionados con la adopción de ASVS. Esto ayudará a los recién llegados al estándar planificar la adopción de la ASVS, mientras que ayuda a las empresas existentes aprendiendo de la experiencia de otros.

Es esperable que este estándar no genere un 100% de consenso. El Análisis de riesgo es siempre subjetivo hasta cierto punto, el cual crea un reto al tratar de generalizar un estándar para todo tipo de aplicaciones. Sin embargo, esperamos que los últimos cambios en esta versión sean un paso hacia la dirección correcta, los cuales, de manera respetuosa, mejoran los conceptos introducidos en este importante estándar de la industria.

¿Qué hay de nuevo en la versión 3.0?

En la versión 3.0, hemos añadido varias secciones nuevas, incluyendo configuración, *Web Services*, aplicaciones cliente, para hacer la norma más aplicable a aplicaciones modernas, comúnmente *responsive*, con amplio uso de interfaces de usuario HTML5 o móvil, llamando a un conjunto común de *Web Services* REST, utilizando autenticación SAML.

Hemos reducido también las duplicaciones en la norma, por ejemplo, para asegurarnos de que un desarrollador móvil no necesite re-testear los mismos elementos varias veces.

Hemos proporcionado la correspondencia con el CWE (Diccionario de debilidades comunes). Esta correspondencia utilizarse para identificar información tal como la probabilidad de explotación, consecuencia de una explotación exitosa y en términos generales tener la visión de lo que podría salir mal si un control de seguridad no se utiliza o no se aplican eficazmente para mitigar la debilidad.

Por último, buscamos ayuda en la comunidad con el fin de generar sesiones para la revisión por pares durante las conferencias de AppSec EU 2015 y una sesión final de trabajo en AppSec USA 2015 con el fin de incluir una enorme cantidad de comentarios de la comunidad. Durante la revisión, si el significado de un control fue cambió sustancialmente, se creó un nuevo control dejando el antiguo como obsoleto. Hemos deliberadamente optado por no volver a utilizar los requisitos de control obsoletos, ya que podría ser una fuente de confusión. Hemos proporcionado una asignación completa de lo que ha cambiado en el apéndice A.

Tomados en conjunto, la versión numero 3.0 contiene el cambio más grande historia del estándar. Esperamos que le sea útil la actualización de la norma y que lo utilice en formas que sólo podemos imaginar.

Utilizando el Estándar de Verificación de Seguridad en Aplicaciones

ASVS tiene dos objetivos principales:

- ayudar a las organizaciones en el desarrollo y mantenimiento aplicaciones seguras
- permitir la alineación entre las necesidades y ofertas de los servicios de seguridad, proveedores de herramientas de seguridad y consumidores

Niveles de verificación de seguridad de aplicaciones

El Estándar de Verificación de Seguridad en Aplicaciones define tres niveles de verificación de seguridad, incrementando la profundidad con cada nivel.

- ASVS nivel 1 se encuentra dirigido a todo tipo de software.
- ASVS nivel 2 es para aplicaciones que contienen datos sensibles, que requieren protección.
- ASVS nivel 3 es para las aplicaciones más críticas - aplicaciones que realizan transacciones de alto valor, contienen datos médicos confidenciales, o cualquier aplicación que requiera el más alto nivel de confianza.

Cada nivel ASVS contiene una lista de requerimientos de seguridad. Cada uno de estos requisitos pueden también corresponderse a funcionalidades específicas de seguridad y capacidades que deben construirse por los desarrolladores de software.

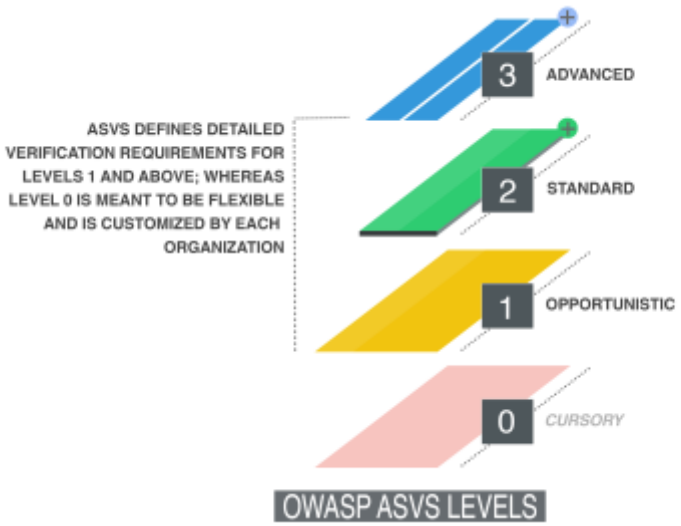

Figura 1 – Niveles del Estándar de Verificación de Seguridad en Aplicaciones de OWASP

Cómo utilizar este estándar

Una de las mejores maneras de emplear el Estándar de Verificación de Seguridad en Aplicaciones es utilizarlo como checklist de seguridad específica para su aplicación, plataforma u organización. Ajustar el ASVS a sus casos de uso le ayudará aumentar el foco en los requisitos de seguridad más importantes para sus proyectos y entornos.

Nivel 1: Oportunista

Una aplicación alcanza ASVS nivel 1 (u oportunista) si se defiende adecuadamente contra vulnerabilidades de seguridad de aplicaciones que son fáciles de descubrir y se incluyen en el OWASP Top 10 u otras listas similares.

Este nivel es apropiado típicamente para aplicaciones donde se requiere escasa confianza en el uso correcto de los controles de seguridad, o para proporcionar un análisis rápido a un conjunto de aplicaciones de una organización, o asistir en la elaboración de una lista de requerimientos de seguridad con prioridades como parte de un esfuerzo de múltiples fases. Los controles de nivel 1 pueden ser asegurados automáticamente por herramientas o manualmente sin acceso al código fuente. Consideramos a nivel 1 como el mínimo requerido para todas las aplicaciones.

Las amenazas a la aplicación probablemente provendrán de atacantes que utilizan técnicas simples y de bajo esfuerzo para identificar vulnerabilidades fáciles de encontrar y de explotar. Esto es en contraste con un ataque dirigido el cual se enfocará específicamente a la aplicación. Si los datos procesados por la aplicación tienen un alto valor, difícilmente sea suficiente con una revisión de nivel 1.

Nivel 2: Estándar

Una aplicación alcanza ASVS nivel 2 (o estándar), si se defiende adecuadamente contra la mayoría de los riesgos asociados con el software de hoy en día.

El Nivel 2 asegura que controles de seguridad se encuentran en el lugar adecuado, son efectivos y son utilizados dentro de la aplicación. Este nivel es generalmente apropiado para aplicaciones que manejan transacciones business-to-business, información de salud, implementan funciones sensibles o críticas para el negocio o incluyen el proceso de otros activos sensibles.

Las amenazas para aplicaciones de nivel 2 por lo general están motivados por atacantes los cuales se centran en objetivos concretos, utilizando herramientas y técnicas efectivas en el descubrimiento y explotación de vulnerabilidades dentro de las aplicaciones.

Nivel 3: Avanzado

El nivel 3 en ASVS es el más alto nivel de verificación dentro de ASVS. Este nivel está reservado normalmente para aplicaciones que requieren niveles significativos de

verificación de seguridad, como las que se encuentran dentro de áreas de militares, salud, seguridad, infraestructuras, etc. Las organizaciones pueden requerir del nivel 3 para aplicaciones que realizan funciones críticas, donde una falla de seguridad podría afectar significativamente sus operaciones y hasta su supervivencia. Un ejemplo en la aplicación del nivel 3 de ASVS se proporciona a continuación. Una aplicación alcanza el nivel 3 (o avanzado) si se defiende adecuadamente contra vulnerabilidades de seguridad avanzadas y también demuestra los principios de un buen diseño de seguridad.

Una aplicación en el nivel 3 de ASVS, requiere un análisis de mayor profundidad, arquitectura, codificación y Testing en todo nivel. Una aplicación segura es modularizada de forma significativa (para facilitar por ejemplo su resiliencia, escalabilidad, y sobre todo, capas de seguridad), y cada módulo (separados por conexiones de la red o instancias físicas) se encarga de sus responsabilidades de seguridad (defensa en profundidad) la que debe ser debidamente documentada. Las responsabilidades incluyen controles para asegurar la confidencialidad (cifrado, por ejemplo), integridad (transacciones, validación de la entrada), disponibilidad (manejo de carga), autenticación (incluyendo autenticación entre sistemas), no repudio, autorización y auditoría (bitácoras).

Aplicando ASVS en la práctica

Diferentes amenazas poseen diferentes motivaciones. Algunas industrias tienen activos de información únicos y valiosos y deben cumplir regulaciones y normas específicas de dichas industrias.

A continuación, se proporcionan recomendaciones con respecto a los niveles de ASVS específicas para algunas industrias.

Aunque existen criterios únicos y diferencias en las amenazas para cada sector, una amenaza común a todos los segmentos o industrias, son los ataques oportunistas. Estos buscarán cualquier aplicación vulnerable fácilmente explotable. Por esta razón el nivel 1 se recomienda para toda aplicación. Se sugiere este punto de partida para manejar los riesgos más fáciles de encontrar. También se recomienda a las organizaciones observar detenidamente sus riesgos específicos basados en la naturaleza de su negocio.

Por otro lado, el nivel 3 de ASVS se encuentra reservado para aquellos casos en que se podría poner en peligro la seguridad humana o cuando una violación a la aplicación podría impactar seriamente y por completo a la organización.

Industria	Perfil de amenaza	L1 Recomendación	L2 Recomendación	L3 Recomendación
Financiera y Seguros	Aunque este segmento experimentará intentos de atacantes oportunistas, a menudo es visto como un objetivo de alto valor por atacantes motivados y los ataques se deben muy a menudo a motivos financieros. Comúnmente, los atacantes buscan datos o credenciales de la cuenta que pueden utilizar para cometer fraudes o beneficiarse directamente aprovechando la funcionalidad de flujo de dinero en aplicaciones. Las técnicas incluyen a menudo credenciales robadas, ataques a nivel de aplicación y la ingeniería social. Algunas consideraciones importantes de cumplimiento incluyen EL ESTÁNDAR PCI DSS (PCI DSS), Gramm Leech Bliley Act y Sarbanes-Oxley (SOX).	Todas las aplicaciones accesibles desde la red.	Aplicaciones que contienen información sensible como números de tarjeta de crédito, información personal, que puede mover una cantidad limitada de dinero de manera limitada. Los ejemplos incluyen: (i) transferir dinero entre cuentas en la misma institución o (ii) una forma más lenta del movimiento de dinero (por ejemplo, ACH) con límites de transacción o (iii) transferencias en línea con límites de transferencia dentro de un período de tiempo.	Aplicaciones que contengan grandes cantidades de información sensible o que permiten que sea rápido la transferencia de grandes sumas de dinero (p. ej. transferencias) o transferencia de grandes sumas de dinero en forma de transacciones individuales o como un lote de transferencias pequeñas.

Industria	Perfil de amenaza	L1 Recomendación	L2 Recomendación	L3 Recomendación
Manufactura, profesional, transporte, tecnología, utilidades, infraestructura y defensa	Estas industrias no parecen tener mucho en común, pero los agentes de amenaza que suelen atacar a las organizaciones en este segmento son más propensos a realizar ataques enfocados con más tiempo, habilidad y recursos. A menudo la información sensible o los sistemas no son fáciles de localizar y requieren utilizar o manipular individuos que trabajen dentro de la organización, utilizando técnicas de ingeniería social. Los ataques pueden involucrar individuos que trabajan dentro de la organización, extraños a la organización, o una combinación de ambos. Sus objetivos pueden incluir acceso a la propiedad intelectual para obtener ventajas estratégicas o tecnológicas. Tampoco queremos pasar por alto a los atacantes que buscan abusar la funcionalidad de la aplicación para influenciar el comportamiento de la aplicación o alterar sistemas sensibles. La mayoría de los atacantes buscan información sensible que puede ser utilizada directa o indirectamente para beneficiarse al incluir datos personales a la información de pago. A menudo los datos pueden utilizarse para una variedad de esquemas de fraude, robo de identidad o pagos fraudulentos.	Todas las aplicaciones accesibles desde la red.	Aplicaciones que contienen información interna o información sobre empleados que pueden aprovecharse utilizando la ingeniería social. Aplicaciones que contiene información poco esencial, pero de importante propiedad intelectual o secretos comerciales.	Aplicaciones que contiene valiosa propiedad intelectual, secretos comerciales o secretos del gobierno (p. ej. en los Estados Unidos esto puede ser cualquier cosa clasificados en secreto o superior) que es fundamental para la supervivencia o el éxito de la organización. Aplicaciones que controlan funcionalidad sensible (p. ej. transporte, fabricación de equipos, sistemas de control) o que tienen la posibilidad de amenazar la seguridad

Industria	Perfil de amenaza	L1 Recomendación	L2 Recomendación	L3 Recomendación
Salud	La mayoría de los atacantes está buscando información sensible que puede ser utilizada directa o indirectamente para beneficiarse al incluir datos personales a la información de pago. A menudo los datos pueden utilizarse para una variedad de esquemas de fraude, robo de identidad y pagos fraudulentos. Para Los Estados Unidos existen el Health Insurance Portability and Accountability Act (HIPAA) http://www.hhs.gov/ocr/privacy/	Todas las aplicaciones accesibles desde la red	Aplicaciones con cantidades pequeñas o moderadas de información médica confidencial (información de salud protegida), información de identificación personal o datos de pago.	Aplicaciones utilizadas para el control de equipos médicos, dispositivos o registros que pueden poner en peligro la vida humana. Sistemas de pago de Punto de venta y (POS) que contienen grandes cantidades de datos de transacciones que podrían ser utilizados para cometer fraudes. Esto incluye las interfaces administrativas de estas aplicaciones
Venta por menor, alimento, hospitalidad	Muchos de los atacantes en este segmento utilizan tácticas oportunistas de "aplaste y agarre". Sin embargo, también existe una amenaza regular de ataques específicos en aplicaciones que contienen información de pagos, que realizan transacciones financieras o almacenan información personal que pueda ser identificable. Aunque menos probable que las amenazas antes mencionadas, también existe la posibilidad de amenazas más avanzadas las cuales atacan a este segmento de la industria para robar propiedad intelectual, obtener inteligencia competitiva o ganar una ventaja con la organización la cual se ha atacado o de un socio en negociaciones.	Todas las aplicaciones accesibles desde la red.	Adecuado para aplicaciones de negocios, Catálogo de la información del producto, información corporativa interna y aplicaciones con información limitada del usuario (p. ej. información de contacto). Aplicaciones con cantidades pequeñas o moderadas de la funcionalidad de datos o de comprobación de pago.	Sistemas de pago de Punto de venta y (POS) que contienen grandes cantidades de datos de transacciones que podrían ser utilizados para cometer fraudes. Esto incluye las interfaces administrativas de estas aplicaciones. Aplicaciones con un gran volumen de información sensible como números de tarjeta de crédito, nombres completos, documentos de indentidad, etc.

Caso de estudios

Caso de Estudio 1: ASVS como guía de prueba de seguridad

En una Universidad privada en Utah, Estados Unidos, el equipo rojo del campus utiliza el OWASP ASVS como guía al realizar tests de penetración a aplicaciones. Es utilizado a lo largo del proceso de pruebas, desde la planificación inicial, definiendo reuniones de orientación para las actividades de la prueba y como una manera de enmarcar las conclusiones del informe final entregado a los clientes. El equipo rojo también organiza capacitaciones para el equipo utilizando el ASVS.

El equipo rojo del Campus realiza tests de penetración de redes y aplicaciones para varios departamentos del campus como parte de la estrategia de seguridad de la información general de la Universidad. Durante las reuniones de planificación iniciales, los clientes suelen ser reticentes a dar autorización para que su aplicación sea puesta a prueba por un equipo de estudiantes. Al presentar el ASVS y explicando a los interesados que las actividades de pruebas son guiadas por esta estándar, y que el informe final incluirá cómo se comporta la aplicación en comparación con el estándar, muchas preocupaciones quedan inmediatamente aclaradas. Luego, el ASVS es utilizado durante la evaluación para ayudar a determinar cuánto tiempo y esfuerzo se utilizará en la prueba. Mediante el uso de los niveles de verificación predefinidas de la ASVS, el equipo rojo explica el enfoque basado en riesgo de las pruebas a realizar. Esto ayuda al cliente, los actores y el equipo para llegar a un acuerdo sobre un alcance apropiado para la aplicación en cuestión.

Una vez que el test comienza, el equipo rojo utiliza el ASVS para organizar actividades y dividir la carga de trabajo. Al realizar el seguimiento de los requisitos de verificación que han sido probados y que se encuentran pendientes, el gerente de proyecto para el equipo puede observar fácilmente el avance de las pruebas. Esto conduce a mejorar la comunicación con los clientes y permite al jefe de proyecto gestionar mejor los recursos. Dado que el equipo rojo está compuesto principalmente de estudiantes, la mayoría de los miembros del equipo tienen múltiples demandas de su tiempo proveniente de diferentes cursos. Las tareas bien definidas, basadas en los requisitos de verificación individual o categorías totales, ayudan a los miembros del equipo a saber exactamente qué debe analizarse y que estos puedan proporcionar estimaciones precisas sobre cuánto tiempo tardarán para completar las tareas. La actividad de informar también se beneficia de la clara organización del ASVS, debido a que los miembros del equipo pueden reportar sus hallazgos antes de continuar con el siguiente punto del estándar, generando el informe de forma simultánea con la ejecución de las pruebas de penetración.

El equipo rojo organiza el informe final alrededor de la ASVS, informando del estado de cada requisito de verificación y proporcionando más detalles cuando sea apropiado. Esto le da una idea a clientes e interesados de cómo se encuentra su aplicación con respecto al estándar, lo cual es extremadamente valioso desde el punto de vista del seguimiento pues permite ver cómo ha mejorado o empeorado la seguridad durante un transcurso de tiempo.

Además, los actores interesados en cómo la aplicación se comporta en una categoría específica o categorías pueden encontrar fácilmente dado que el formato del informe se alinea con el formato del ASVS. La estructura del ASVS también facilita el entrenamiento de nuevos miembros del equipo sobre cómo escribir un informe comparando el formato con el informe anterior.

Finalmente, el entrenamiento del equipo rojo ha mejorado con la adopción del ASVS. Anteriormente, los entrenamientos semanales se centraban sobre un tema elegido por el equipo o gerente del proyecto. Éstos eran seleccionados basados en las peticiones de los miembros del equipo y las necesidades percibidas. La formación basada en estos criterios tenía el potencial de ampliar las habilidades de los miembros del equipo, pero no necesariamente se relacionaban con las actividades básicas del equipo rojo. En otras palabras, el equipo no consiguía mejorar significativamente en pruebas de penetración. Después de adoptar el ASVS, el entrenamiento del equipo, se centra ahora en cómo probar los requisitos de verificación individual. Esto ha llevado a una mejora significativa y medible en las habilidades de los miembros individuales del equipo y la calidad de los informes finales.

Estudio de caso 2: un SDLC seguro

Un emprendimiento "start-up" que busca proporcionar análisis de grandes datos a instituciones financieras reconoce que implementar seguridad durante el desarrollo de su aplicación es una de las prioridades más altas de la lista de requisitos para acceder y procesar meta-datos financieros. En este caso, la Compañía "start-up" ha optado por utilizar el ASVS como base de su ciclo de desarrollo ágil.

La compañía "start-up" utiliza el ASVS para generar casos de uso e historias para cuestiones de seguridad funcional, tales como la mejor manera de implementar la funcionalidad para iniciar la sesión en la aplicación. La compañía "start-up" usa ASVS de forma diferente en comparación a la mayoría - ellos tratan de ver a través de ASVS, recogiendo los requisitos que se adhieren al sprint actual, y lo agregan directamente a la acumulación del sprint si es un requisito funcional, o como una limitación a casos de uso existentes si no son funcionales. Por ejemplo, la adición de la autenticación TOTP (Contraseña de un solo uso basada en tiempo), junto con las políticas de contraseñas y servicio web que de detección de ataques de fuerza bruta. En los sprints futuros, los requisitos adicionales se seleccionarán basados en el criterio "justo a tiempo", o que "no se va a necesitar".

Los desarrolladores utilizan el ASVS como una checklist de revisión por pares, lo que ayuda a que código inseguro no sea ingresado en el repositorio, y en retrospectiva, desafiar a los desarrolladores que hayan ingresado algún código en el repositorio que contenga una nueva característica, que hayan considerado los requisitos del ASVS y si hay algo se puede mejorar o reducir en los sprints futuros.

Por último, los desarrolladores utilizan la ASVS como parte de su unidad de seguridad de verificación automatizada y conjuntos de pruebas de integración, casos de abuso y casos de

prueba a través de "fuzzing". El objetivo es reducir el riesgo de la "metodología de catarata" el cual pone el "test de penetración al final" causando costosos esfuerzos de re-factorización de código cuando se deben entregar ciertas metas dentro del sistema. Como podrían promoverse nuevas características del software después de cada sprint, no es suficiente confiar en una actividad única de aseguramiento. Así, mediante la automatización de las pruebas, no debería haber cuestiones de suma importancia que puedan ser encontradas por un pentester calificado con tan solo semanas para probar la aplicación.

Evaluando que el software ha alcanzado un nivel de verificación

Postura de OWASP en certificaciones de ASVS y las marcas de confianza

OWASP, como organización independiente sin fines de lucro, no certifica proveedores, verificadores ni software.

Tales afirmaciones de aseguramiento, sellos de confianza o certificaciones no son oficialmente validadas, registradas o certificadas por OWASP, por lo tanto, una organización que cuenta con ese punto de vista debe ser cautelosa en depositar su confianza en cualquier tercero o sello de confianza que clame que tiene una certificación de ASVS.

Esto no debe inhibir organizaciones para ofrecer dichos servicios de garantía, con tal de que no pretendan proveer una certificación oficial de OWASP.

Guía para las organizaciones certificadoras

El estándar de verificación de seguridad en aplicaciones puede ser utilizado como un libro abierto de verificación para la aplicación, en conjunto con el acceso abierto y sin restricciones a arquitectos y desarrolladores, documentación de proyectos, código fuente, acceso autenticado al sistema (incluyendo por lo menos una cuenta en cada rol), particularmente para las verificaciones de nivel 2 y 3.

Históricamente, los test de penetración y las revisiones de código han incluido hallazgos "por excepción". – es decir, el informe final solo se compone de defectos de seguridad. Una organización certificadora debe incluir en cualquier informe cual fue el alcance de la verificación (especialmente si un componente clave está fuera de alcance, como la autenticación SSO), un resumen de resultados de la verificación, incluyendo las pruebas exitosas y las fallidas, con indicaciones claras de cómo resolver las fallidas.

Una práctica estándar en la industria es mantener papeles de trabajo detallados, imágenes o videos, registros electrónicos de las pruebas, registros proxy y notas asociadas como un script de limpieza. Pueden ser realmente útiles como pruebas de los hallazgos para la mayoría de los desarrolladores que tengan dudas. No es suficiente con ejecutar una herramienta e informar sobre de las fallas; Esto no (en absoluto) proporciona suficiente evidencia que todos los problemas han sido probados a un nivel de certificación y que estos se hayan comprobado completamente. En caso de controversia, debe existir prueba suficiente que garantice demostrar que cada requisito verificado de hecho ha sido probado.

El papel de las herramientas automáticas de pruebas de penetración

Las herramientas automatizadas buscan brindar la mayor cobertura posible y ejecutar tantos parámetros como sea posible con varias formas de entradas maliciosas.

No es posible completar totalmente la verificación ASVS utilizando solamente herramientas automáticas. Mientras que una gran mayoría de los requisitos en el nivel 1 se pueden verificar por medio de pruebas automatizadas, la mayoría no lo son.

Se debe de tener en cuenta que a medida que madura la industria de la seguridad en aplicaciones, la línea entre pruebas automatizadas y manuales se torna borrosa. Las herramientas automatizadas son a menudo adaptadas manualmente por expertos para realidades puntuales.

El rol del test de penetración

Es posible realizar una prueba de penetración manual y verificar todos los puntos del nivel L1 sin necesidad de acceso al código fuente, aunque esta no es una práctica muy utilizada. Para el nivel L2 se requiere al menos algún acceso a los desarrolladores, documentación, código y acceso autenticado al sistema. Una cobertura completa de pruebas de penetración del nivel 3 no es posible, como la mayoría de los problemas adicionales incluyen revisión de configuración del sistema, revisión de código malicioso, modelado de amenazas y otros artefactos de prueba de penetración.

Como una guía de arquitectura de seguridad detallada

Uno de los usos más comunes para el estándar de verificación de seguridad en aplicaciones es su utilización como un recurso para arquitectos de seguridad. Los dos marcos de la arquitectura de seguridad, SABSA o TOGAF, carecen de una gran cantidad de información necesaria para completar por medio de una revisión de arquitectura aspectos de seguridad de la aplicación. ASVS puede utilizarse mitigar esas carencias permitiendo a los arquitectos de seguridad elegir controles adecuados para problemas comunes, tales como patrones de protección de datos y estrategias de validación de entradas de datos.

Como reemplazo para checklist de verificación generadas por terceros

Muchas organizaciones pueden beneficiarse de la adopción del ASVS, eligiendo uno de los tres niveles, o utilizar ASVS y refinar únicamente lo que se requiere para cada nivel de riesgo de la aplicación en un dominio específico. Recomendamos este tipo de combinación o adaptación del ASVS original mientras se mantiene la trazabilidad, por lo que si una aplicación ha pasado requisito 4.1, esto significa lo mismo tanto para el refinamiento como para el estándar a medida que éste evolucione.

Como una guía para pruebas unitarias y de integración automatizadas

El ASVS está diseñado para ser altamente verificable, con la sola excepción de requisitos de arquitectónicos y de código malicioso. A través de la generación de pruebas unitarias y de integración (tanto específicas como de fuzzing), la aplicación se aproxima a auto verificase y validarse con cada construcción. Por ejemplo, se pueden generar pruebas adicionales para el paquete de pruebas para el control de acceso, para testear el parámetro nombre de usuario para nombres de usuario comunes, enumeración de cuenta, ataques de fuerza

bruta, inyección LDAP y SQL y XSS. Asimismo, se deben incluir pruebas al parámetro de contraseña para las más comunes, su longitud, inyección de byte nulo, eliminación del parámetro, XSS, enumeración de cuentas y mucho más.

Como capacitación para el desarrollo seguro

ASVS también puede utilizarse para definir características de software seguro. Muchos cursos de «codificación segura» son simplemente cursos éticos de hacking con un ligero toque de consejos sobre codificación. Esto no ayuda a los desarrolladores de sistemas. En cambio, cursos de desarrollo seguro pueden usar la guía ASVS con un fuerte enfoque en los controles pro-activos en esta, en lugar de cosas negativas del OWASP Top 10.

Proyectos OWASP que utilizan ASVS

Security Knowledge Framework

https://www.owasp.org/index.php/OWASP_Security_Knowledge_Framework

Para entrenar desarrolladores en la escritura de código seguro - El proyecto SKF es una aplicación completamente gratis escrita en Python-Flask que utiliza el estándar de verificación de seguridad OWASP para aplicaciones, pensada para entrenar en la escritura de código seguro desde su diseño.

OWASP Zed ataque Proxy (ZAP)

https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

OWASP Zed (ZAP) es una herramienta de fácil uso e integrada para la búsqueda de vulnerabilidades en aplicaciones web. Está diseñada tanto para ser utilizado por personas con amplia experiencia en seguridad, así como desarrolladores y testers funcionales que son nuevos en pruebas de penetración. ZAP ofrece escaneos automatizados e integra un conjunto de herramientas que permiten encontrar vulnerabilidades de seguridad manualmente.

OWASP Cornucopia

https://www.owasp.org/index.php/OWASP_Cornucopia

Cornucopia de OWASP es un mecanismo en forma de un juego de cartas que ayudar a equipos de desarrollo de software a identificar requisitos de seguridad utilizado metodologías ágiles y procesos de desarrollo formales. Es un lenguaje agnótico de la tecnología y la plataforma. El contenido de Cornucopia fue seleccionado en base a la estructura de la Guía Práctica de Codificación Segura de OWASP - guía de referencia rápida (SCP), considerando adicionalmente el estándar de verificación de seguridad de OWASP para aplicaciones, la guía de pruebas de OWASP y los principios de programación segura de David Rook.

Requisitos de verificación detallada

V1. Arquitectura, diseño y modelado de amenazas

V2. Autenticación

V3. Gestión de sesiones

V4. Control de acceso

V5. Manejo de entrada de datos maliciosos

V7. Criptografía en el almacenamiento

V8. Gestión y registro de errores

V9. Protección de datos

V10. Comunicaciones

V11. Configuración de seguridad HTTP

V13. Controles Maliciosos

V15. Lógica de negocio

V16. Archivos y recursos

V17. Móvil

V18. Servicios Web (Nuevo en 3.0)

V19. Configuración (nuevo en 3.0)

V1: Arquitectura, diseño y modelado de amenazas

Objetivo de control

Asegurar que una aplicación verificada satisfaga los siguientes requisitos de alto nivel:

- Nivel 1, los componentes de la aplicación son identificados y tienen una razón de ser.
- Nivel 2, se ha definido la arquitectura y el código se adecúa a ésta.
- Nivel 3, la arquitectura y el diseño son los indicados, se utilizan y resultan eficaces.

Nota: Esta sección se ha introducido nuevamente en la versión 3.0, pero se utilizan en esencia los mismos controles arquitectónicos de la versión 1.0 del ASVS.

Requisitos

#	Descripción	1	2	3	Desde
1.1	Verificar que todos los componentes de la aplicación se encuentran identificados y asegurar que son necesarios.	✓	✓	✓	1.0
1.2	Verificar todos los componentes, tales como bibliotecas, módulos y sistemas externos, que no son parte de la aplicación pero que la misma los necesita para funcionar se han identificado.		✓	✓	1.0
1.3	Verificar que se ha definido una arquitectura de alto nivel para la aplicación.		✓	✓	1.0
1.4	Verificar que todos los componentes de la aplicación se definen de acuerdo a las funciones de negocio o de seguridad que proporcionan.			✓	1.0
1.5	Verificar que todos los componentes que no son parte de la aplicación pero que son necesarios para su funcionamiento, sean definidos de acuerdo a las funciones de negocio o de seguridad que proporcionan.			✓	1.0
1.6	Verificar que se ha realizado un modelo de amenazas para la aplicación en cuestión y que éste cubre riesgos asociados con la suplantación de identidad, manipulación, repudio, revelación de información y elevación de privilegios (STRIDE).			✓	1.0
1.7	Verificar que todos los controles de seguridad (incluyendo las bibliotecas que llaman a servicios de seguridad externos) tienen una implementación centralizada.		✓	✓	3.0
1.8	Verificar que los componentes están separados unos de		✓	✓	3.0

#	Descripción	1	2	3	Desde
	otros mediante controles de seguridad, tales como segmentación de la red, reglas de firewall, o grupos de seguridad basados en la nube.				
1.9	Verificar que la aplicación tiene una clara separación entre la capa de datos, la capa de control y la capa de presentación, tal que las decisiones de seguridad pueden aplicarse en sistemas confiables.		✓	✓	3.0
1.10	Verificar que no hay ninguna lógica de negocio sensible, claves secretas u otra información propietaria en el código del lado del cliente.		✓	✓	3.0
1.11	Verificar que todos los componentes de la aplicación, bibliotecas, módulos, frameworks, plataformas y sistemas operativos se encuentran libres de vulnerabilidades conocidas		✓	✓	3.0.1

Referencias

Para obtener más información, consulte:

- “Cheat Sheet” de Modelado de Amenazas
  https://www.owasp.org/index.php/Application_Security_Architecture_Cheat_Sheet
- “Cheat Sheet” de Análisis de Superficie de ataque
  https://www.owasp.org/index.php/Attack_Surface_Analysis_Cheat_Sheet

V2: Requisitos de verificación de autenticación

Objetivo de control

Autenticación es el acto de establecer o confirmar, algo (o alguien) como auténtico, esto es, que lo que reclama sobre aquello es verdadero. Se debe asegurar que la aplicación satisface los siguientes requisitos de alto nivel:

- Verifica la identidad digital del remitente de una comunicación.
- Asegura que sólo los usuarios autorizados son capaces de autenticarse y que las credenciales sean transportadas de forma segura.

Requisitos

#	Descripción	1	2	3	Desde
2.1	Verificar que todas las páginas y recursos requieran autenticación excepto aquellos que sean específicamente destinados a ser públicos (Principio de mediación completa).	✓	✓	✓	1.0
2.2	Verificar que todos los campos de credenciales no reflejen las contraseñas del usuario. Cargar la credencial por parte de la aplicación implica que la misma fue almacenada de forma reversible o en texto plano, lo que se encuentra explícitamente prohibido.	✓	✓	✓	3.0.1
2.4	Verificar que todos los controles de autenticación se realicen del lado del servidor.	✓	✓	✓	1.0
2.6	Verificar que los controles de autenticación fallan de forma segura para evitar que los atacantes no puedan iniciar sesión.	✓	✓	✓	1.0
2.7	Verificar que los campos de contraseñas permiten o fomentan el uso de frases como contraseñas (passphrases) y no impiden el uso de gestores de contraseñas, contraseñas largas o altamente complejas.	✓	✓	✓	3.0.1
2.8	Verificar que toda función relacionada con la autenticación (como registro, actualización del perfil, olvido de nombre de usuario, recuperación de la contraseña, token perdido / deshabilitado, funciones de help desk o IVR) que pueda ser utilizada de forma indirecta como mecanismo de autenticación, sea al menos tan resistente a ataques como el mecanismo primario.	✓	✓	✓	2.0

#	Descripción	1	2	3	Desde
2.9	Verificar que la funcionalidad de cambio de contraseña solicite la contraseña anterior, la nueva contraseña y una confirmación de la contraseña.	✓	✓	✓	1.0
2.12	Verificar que todas las decisiones de autenticación son registradas en la bitácora sin almacenar información sobre la contraseña o el identificador de la sesión. Esto debería incluir los metadatos necesarios para investigaciones de seguridad.		✓	✓	3.01
2.13	Verificar que las contraseñas de las cuentas se encuentren almacenadas utilizando una rutina de hashing con una sal, y que requiera un factor de trabajo lo suficientemente alto para evitar un ataque de fuerza bruta.		✓	✓	3.0.1
2.16	Verificar que las credenciales son transportadas mediante un enlace cifrado adecuadamente y que todas las páginas/funciones que requieren que el usuario introduzca credenciales se realicen utilizando enlaces cifrados.	✓	✓	✓	3.0
2.17	Verificar que las funciones de recuperar contraseña y acceso no revelen la contraseña actual y que la nueva contraseña no se envíe en texto plano al usuario.	✓	✓	✓	2.0
2.18	Verificar que no es posible enumerar información mediante las funcionalidades de: inicio de sesión, reinicio o recuperación contraseñas.	✓	✓	✓	2.0
2.19	Verificar que no se utilizan contraseñas por defecto en la aplicación o cualquiera de los componentes utilizados por la misma (como "admin/password").	✓	✓	✓	2.0
2.20	Verificar que existen mecanismos de anti-automatización que previenen la verificación de credenciales obtenidas de forma masiva, ataques de fuerza bruta y ataques de bloqueos de cuentas.	✓	✓	✓	3.0.1
2.21	Verificar que todas las credenciales de autenticación para acceder a servicios externos a la aplicación se encuentran cifradas y almacenadas en un lugar protegido.		✓	✓	2.0
2.22	Verificar que las funcionalidades de recuperar contraseña y otras formas de recuperar la cuenta utilizan mecanismos de TOTP (Time-Based One-Time Password) u otro tipo de soft token, push a dispositivo móvil u otro tipo de mecanismo de recuperación offline. El uso de un valor aleatorio en un correo electrónico o SMS debe ser la última opción ya que son conocidas sus debilidades.	✓	✓	✓	3.0.1

#	Descripción	1	2	3	Desde
2.23	Verificar que el bloqueo de la cuenta se divida en estado de bloqueo suave y duro, y éstas no son mutuamente excluyentes. Si una cuenta está temporalmente bloqueada de forma suave debido a un ataque de fuerza bruta, esto no debe restablecer el bloqueo de estado duro.		✓	✓	3.0
2.24	Verificar que si la aplicación hace uso de conocimiento basado en preguntas (también conocido como "secreto"), las preguntas no violan leyes de privacidad y son lo suficientemente fuertes para proteger la cuenta de recuperaciones maliciosas.	✓	✓	✓	3.0.1
2.25	Verificar que el sistema puede configurarse para no permitir el uso de un número de contraseñas utilizadas anteriormente.		✓	✓	2.0
2.26	Verificar que la re-autenticación basada en riesgo, autenticación de dos factores o firma de transacciones se encuentra implementada en los lugares adecuados.		✓	✓	3.0.1
2.27	Verificar que existen medidas para bloquear el uso de contraseñas comúnmente utilizadas y contraseñas débiles.	✓	✓	✓	3.0
2.28	Verificar que todos los desafíos de autenticación, ya sea exitosa o fallida, responden en el mismo tiempo promedio.			✓	3.0
2.29	Verificar que secretos, llaves de API y contraseñas no se incluyen en el código fuente o en los repositorios en línea de código fuente.			✓	3.0
2.31	Verificar que, si una aplicación permite a los usuarios autenticarse, puedan hacerlo mediante autenticación de dos factores u otra autenticación fuerte, o cualquier esquema similar que proporcione protección contra la divulgación de nombres de usuario y contraseñas.		✓	✓	3.0
2.32	Verificar que las interfaces administrativas de la aplicación no sean accesibles a intrusos.	✓	✓	✓	3.0
2.33	Autocompletar de navegadores e integración con gestores de contraseñas deben estar permitidos a no ser que se encuentren prohibidos por políticas de riesgos.	✓	✓	✓	3.0.1

Referencias

Para obtener más información, consulte:

- Guía de pruebas OWASP 4.0: Prueba de autenticación
  https://www.owasp.org/index.php/Testing_for_authentication
- Cheat Sheet - Almacenamiento de Contraseña
  https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet
- Cheat sheet - Olvido de Contraseña
  https://www.owasp.org/index.php/Forgot_Password_Cheat_Sheet
- Escoger y usar preguntas de seguridad
  https://www.owasp.org/index.php/Choosing_and_Using_Security_Questions_Cheat_Sheet

V3: Requisitos de verificación de gestión de sesiones

Objetivo de control

Uno de los componentes básicos de cualquier aplicación web es el mecanismo por el cual controla y mantiene el estado de un usuario al interactuar con ésta. Esto se refiere a manejo de sesiones y se define como el conjunto de todos los controles que rigen el estado completo de interacción entre un usuario y la aplicación basada en la web.

Se debe asegurar que la aplicación verificada satisface los siguientes requerimientos de manejo de sesiones de alto nivel:

- Las sesiones son únicas para cada individuo y no conjeturadas o compartidas
- Las sesiones son invalidadas cuando ya no son necesarias y el tiempo es limitado durante los períodos de inactividad.

Requisitos

#	Descripción	1	2	3	Desde
3.1	Verificar que no se utiliza un gestor de sesiones personalizado, o que, si el gestor de sesiones es personalizado, éste sea resistente contra los ataques más comunes.	✓	✓	✓	1.0
3.2	Verificar que las sesiones se invalidan cuando el usuario cierra la sesión.	✓	✓	✓	1.0
3.3	Verificar que las sesiones se invalidan luego de un período determinado de inactividad.	✓	✓	✓	1.0
3.4	Verificar que las sesiones se invalidan luego de un período determinado de tiempo, independientemente de que se esté registrando actividad (timeout absoluto).			✓	1.0
3.5	Verificar que todas las páginas que requieren autenticación poseen acceso fácil y visible a la funcionalidad de cierre de sesión.	✓	✓	✓	1.0
3.6	Verificar que el identificador de sesión nunca se revele en URLs, mensajes de error o registros de bitácora. Esto incluye verificar que la aplicación no es compatible con la re-escritura de URL incluyendo el identificador de sesión.	✓	✓	✓	1.0
3.7	Verificar que toda autenticación exitosa y re-autenticaciones generen un nuevo identificador de sesión.	✓	✓	✓	1.0

3.10	Verificar que sólo los identificadores de sesión generados por la aplicación son reconocidos como activos por ésta.		✓	✓	1.0
3.11	Verificar que los identificadores de sesión son suficientemente largos, aleatorios y únicos para las sesiones activas.	✓	✓	✓	1.0
3.12	Verificar que los identificadores de sesión almacenados en cookies poseen su atributo “path” establecido en un valor adecuadamente restrictivo y que además contenga los atributos "Secure" y "HttpOnly"	✓	✓	✓	3.0
3.16	Verificar que la aplicación limita el número de sesiones concurrentes activas.	✓	✓	✓	3.0
3.17	Verificar que una lista de sesiones activas esté disponible en el perfil de cuenta o similar para cada usuario. El usuario debe ser capaz de terminar cualquier sesión activa.	✓	✓	✓	3.0
3.18	Verificar que al usuario se le sugiera la opción de terminar todas las otras sesiones activas después de un proceso de cambio de contraseña exitoso.	✓	✓	✓	3.0

Referencias

Para obtener más información, consulte:

- Guía de pruebas OWASP 4.0: sesión de prueba de manejo
  https://www.OWASP.org/index.php/Testing_for_Session_Management
- Cheat Sheet – Gestión de sesiones
  https://www.OWASP.org/index.php/Session_Management_Cheat_Sheet

V4: Requisitos de verificación del Control de acceso

Objetivo de control

Autorización es el concepto de permitir acceso a los recursos únicamente a aquellos que les ha sido permitido utilizarlos. Se debe asegurar que la aplicación verificada satisface los siguientes requisitos de alto nivel:

- Personas que acceden a recursos poseen credenciales válidas para hacerlo.
- Los usuarios se encuentran asociados con un conjunto bien definido de roles y privilegios.
- Los metadatos de Roles y permisos se encuentran protegidos de ataques de reutilización o manipulación.

Requisitos

#	Descripción	1	2	3	Desde
4.1	Verificar que existe el principio de privilegio mínimo - los usuarios sólo deben ser capaces de acceder a las funciones, archivos de datos, URL, controladores, servicios y otros recursos, para los cuales poseen una autorización específica. Esto implica protección contra suplantación de identidad y elevación de privilegios.	✓	✓	✓	1.0
4.4	Verificar que el acceso a registros sensibles esté protegido, tal que sólo objetos autorizados o datos sean accesibles por cada usuario (por ejemplo, proteger contra la posible manipulación hecha por usuarios sobre un parámetro para ver o modificar la cuenta de otro usuario).	✓	✓	✓	1.0
4.5	Verificar que la navegación del directorio esté deshabilitada a menos que esto sea deliberadamente deseado. Además, las aplicaciones no deben permitir el descubrimiento o divulgación de metadatos de archivos o directorios, como carpetas que contengan Thumbs.db, DS_Store, o directorios .git o SVN.	✓	✓	✓	1.0
4.8	Verificar que los controles de acceso fallen de forma segura.	✓	✓	✓	1.0
4.9	Verificar que las mismas reglas de control de acceso implícitas en la capa de presentación son aplicadas en el servidor.	✓	✓	✓	1.0

#	Descripción	1	2	3	Desde
4.10	Verificar que todos los atributos de usuario, datos e información de las políticas utilizadas por los controles de acceso no puedan ser manipulados por usuarios finales a menos que sean específicamente autorizados.		✓	✓	1.0
4.11	Verificar que existe un mecanismo centralizado (incluyendo las bibliotecas que requieren servicios de autorización externa) para proteger el acceso a cada tipo de recursos protegidos.			✓	1.0
4.12	Verificar que todas las acciones de control de acceso pueden ser registradas y que todas las acciones fallidas son registradas.		✓	✓	2.0
4.13	Verificar que la aplicación o su infraestructura emite tokens anti-CSFR aleatorios existe otro mecanismo de protección de la transacción.	✓	✓	✓	2.0
4.14	Verificar que el sistema se pueda proteger contra el acceso permanente a funciones aseguradas, recursos o datos. Por ejemplo, que el sistema utilice un recurso gobernante que limite el número de ediciones por hora o para prevenir que la base de datos sea sobre-utilizada por un único usuario.		✓	✓	2.0
4.15	Verificar que la aplicación disponga de autorización adicional (como autenticación aumentada o adaptación de autenticación) para sistemas de valores bajos, y/o segregación de funciones para aplicaciones de alto valor para cumplir con los controles anti fraude según el análisis de riesgo de la aplicación y fraudes cometidos en el pasado.		✓	✓	3.0
4.16	Verificar que la aplicación aplique correctamente la autorización contextual para no permitir la manipulación de parámetros de la URL.	✓	✓	✓	3.0

Referencias

Para obtener más información, consulte:

- Guía de pruebas OWASP 4.0: Autorización
  https://www.OWASP.org/index.php/Testing_for_Authorization
- Cheat Sheet – Control de acceso
  https://www.OWASP.org/index.php/Access_Control_Cheat_Sheet

V5: Requisitos de verificación para Manejo de entrada de datos maliciosos

Objetivo de control

La debilidad más común de seguridad de las aplicaciones web es la falla en validar apropiadamente el ingreso de datos que provienen del cliente o del ambiente antes de ser utilizada. Esta debilidad conduce a casi todas las vulnerabilidades encontradas en aplicaciones web, tales como cross site scripting (XSS), inyecciones SQL, inyección de intérprete, ataques locale/Unicode, ataques a sistemas de archivos y desbordamientos de búfers.

. Se debe asegurar que la aplicación verificada satisface los siguientes requisitos de alto nivel:

- Todas las entradas son correctamente validadas y adecuadas para el propósito previsto.
- No debe confiarse en datos de una entidad externa o del cliente y deben ser tratados como tales.

Requisitos

#	Descripción	1	2	3	Desde
5.1	Verificar que el entorno de ejecución no es susceptible a desbordamientos de búfer, o que los controles de seguridad previenen desbordamientos de búfer.	✓	✓	✓	1.0
5.3	Verificar que las fallas de validación de entradas de datos del lado del servidor sean rechazadas y registradas.	✓	✓	✓	1.0
5.5	Verificar que se aplican las rutinas de validación de entradas de datos del lado del servidor.	✓	✓	✓	1.0
5.6	Verificar que un único control de validación de entrada es utilizado por la aplicación para cada tipo de datos que es aceptado.			✓	1.0
5.10	Verificar que todas las consultas de SQL, HQL, OSQL, NOSQL y procedimientos almacenados, llamadas de procedimientos almacenados están protegidos por la utilización de declaraciones preparadas o parametrización de consultas, y por lo tanto no sean susceptibles a la inyección de SQL	✓	✓	✓	2.0

#	Descripción	1	2	3	Desde
5.11	Verificar que la aplicación no es susceptible a la inyección LDAP, o que los controles de seguridad previenen inyección LDAP.	✓	✓	✓	2.0
5.12	Verificar que la aplicación no es susceptible a la inyección de comandos del sistema operativo, o que los controles de seguridad previenen la inyección de comandos del sistema operativo.	✓	✓	✓	2.0
5.13	Verificar que la aplicación no es susceptible a la inclusión de archivo remoto (RFI) o inclusión de archivo Local (LFI) cuando el contenido es utilizado como una ruta a un archivo.	✓	✓	✓	3.0
5.14	Verificar que la aplicación no es susceptible a ataques comunes de XML, como manipulación de consultas XPath, ataques de entidad externa XML, y ataques de inyección XML.	✓	✓	✓	2.0
5.15	Asegurar que todas las variables string utilizadas dentro de HTML u otro lenguaje web interpretado en cliente se encuentra apropiadamente codificada manualmente o se utiliza plantillas que automáticamente codifican contextualmente para asegurar que la aplicación no sea susceptible a ataques DOM Cross-Site Scripting (XSS).	✓	✓	✓	2.0
5.16	Si el framework de la aplicación permite asignación automática de parámetros en masa (también llamada enlace automático de variables o variable biding) desde la petición entrante a un modelo, verificar que campos sensibles de seguridad como "*accountBalance*", "*role*" o "password" sean protegidos de enlaces automáticos maliciosos.		✓	✓	2.0
5.17	Verificar que la aplicación contenga defensas contra los ataques de contaminación de parámetros HTTP (agregar parámetros a la URL), particularmente si el framework de la aplicación no hace distinción sobre el origen de los parámetros de la petición (GET, POST, cookies, cabeceras, ambiente, etc.)		✓	✓	2.0
5.18	Verificar que las validaciones del lado del cliente se utilizan como una segunda línea de defensa, en adición a la validación del lado del servidor.		✓	✓	3.0
5.19	Verificar que todos los datos de entrada sean validados, no solamente los campos de formularios HTML sino también todos los orígenes de entrada como las llamadas REST, parámetros de consulta, encabezados HTTP, cookies, archivos por lotes, fuentes RSS, etc.; mediante validación positiva (lista blanca), o utilizando otras formas de validación menos eficaces tales como listas de rechazo transitorio (eliminando símbolos defectuosos), o rechazando malas entradas (listas negras).		✓	✓	3.0

#	Descripción	1	2	3	Desde
5.20	Verificar que datos estructurados fuertemente tipados son validados un esquema definido incluyendo; caracteres permitidos, longitud y patrones (p. ej. tarjeta de crédito o teléfono o validando que dos campos relacionados son razonables, tales como validación de coincidencia entre localidad y código postal).		✓	✓	3.0
5.21	Verificar que los datos no estructurados sean sanitizados cumpliendo medidas genéricas de seguridad tales como caracteres permitidos, longitud y que caracteres potencialmente dañinos en cierto contexto sean anulados (p. ej. nombres naturales con Unicode o apóstrofos, como ねこ o O'Hara)		✓	✓	3.0
5.22	Verificar que HTML no confiable proveniente de editores WYSIWYG o similares sean debidamente sanitizados con un sanitizador de HTML y se manejen apropiadamente según la validación de entrada y codificación.	✓	✓	✓	3.0
5.23	Para tecnologías de plantilla de codificación automática, si ésta se ha deshabilitado, asegurar que la sanitización de HTML esté habilitada en su lugar.		✓	✓	3.0
5.24	Verificar que los datos transferidos desde un contexto DOM a otro, utilice métodos de JavaScript seguro, como pueden ser .innerText y .val		✓	✓	3.0
5.25	Verificar que cuando se interprete JSON en navegadores, que JSON.parse sea el utilizado para interpretarlo y no eval().		✓	✓	3.0
5.26	Verificar que los datos de autenticación se eliminen del almacenamiento del cliente, tales como el DOM del navegador, después de terminada la sesión.		✓	✓	3.0

Referencias

Para obtener más información, consulte:

- Guía de pruebas OWASP 4.0: Validación de entradas de datos
  https://www.OWASP.org/index.php/Testing_for_Input_Validation
- Cheat Sheet: Validación de entradas de datos
  https://www.OWASP.org/index.php/Input_Validation_Cheat_Sheet

- Guía de pruebas OWASP 4.0: pruebas de contaminación de parámetro HTTP https://www.OWASP.org/index.php/Testing_for_HTTP_Parameter_pollution_%28OTG-INPVAL-004%29
- Cheat Sheet: Inyección LDAP https://www.OWASP.org/index.php/LDAP_Injection_Prevention_Cheat_Sheet
- Guía de pruebas OWASP 4.0: Pruebas en el Cliente https://www.OWASP.org/index.php/Client_Side_Testing
- Cheat Sheet: Prevención de Cross Site Scripting https://www.OWASP.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet
- OWASP: Proyecto codificación de Java https://www.OWASP.org/index.php/OWASP_Java_Encoder_Project

Para obtener más información sobre codificación automática, consulte:

- Reducción de XSS a través escape Context-Aware automático en sistemas de plantilla http://googleonlinesecurity.blogspot.com/2009/03/Reducing-XSS-by-Way-of-Automatic.html
- AngularJS Escape Contextual Estricto https://docs.angularjs.org/Api/ng/Service/ $sce

V6: Codificación / escape de salidas de datos

Esta sección se incorporó en V5 en el Estándar de Verificación de Seguridad en Aplicaciones 2.0. Requisitos de ASVS 5.16 tratan la codificación contextual de salida para ayudar a prevenir Cross Site Scripting.

V7: Requisitos de verificación para la criptografía en el almacenamiento

Objetivo de control

Asegure que una aplicación verificada satisfaga los siguientes requisitos de alto nivel:

- Que todos los módulos criptográficos fallen de forma segura y que los errores sean gestionados correctamente.
- Que se utilice un generador de números aleatorios adecuado cuando se requiere la aleatoriedad.
- Que el acceso a claves se gestiona de forma segura.

Requisitos

#	Descripción	1	2	3	Desde
7.2	Verificar que todos los módulos criptográficos fallen de forma segura, y que los errores sean manejados de tal manera que no permitan ataques Oracle padding.	✓	✓	✓	1.0
7.6	Verificar que todos los números aleatorios, nombres aleatorios de archivo, GUIDs aleatorios y cadenas aleatorias sean generados usando un módulo criptográfico aprobado del generador de números aleatorios cuando se pretende que estos valores no puedan ser adivinados o predecibles para un atacante.		✓	✓	1.0
7.7	Verificar que los algoritmos criptográficos utilizados por la aplicación hayan sido validados contra FIPS 140-2 o un estándar equivalente.	✓	✓	✓	1.0
7.8	Verificar que los módulos criptográficos operen en su modo aprobado según sus políticas de seguridad publicadas.			✓	1.0
7.9	Verificar que existe una política explícita para el manejo de las claves criptográficas (por ejemplo, generadas, distribuidas, revocadas y vencidas). Verificar que el ciclo de vida de las clave se aplique correctamente.		✓	✓	1.0

#	Descripción	1	2	3	Desde
7.11	Verificar que los consumidores de servicios criptográficos no poseen acceso directo a los datos de la clave. Aislar procesos criptográficos, incluyendo secretos maestros y considerar el uso de un módulo de seguridad de hardware (HSM).			✓	3.0.1
7.12	*La información de identificación personal* debe almacenarse de forma cifrada y verificar que la comunicación se lleve a cabo utilizando de canales protegidos.		✓	✓	3.0
7.13	Verificar que contraseñas y claves criptográficas sean sobreescritas con ceros en memoria tan pronto no sean necesarias, con el fin de mitigar ataques de volcado de memoria.		✓	✓	3.0.1
7.14	Verificar que todas las claves y contraseñas sean reemplazables, y sean generadas o reemplazadas durante la instalación.		✓	✓	3.0
7.15	Verificar que los números aleatorios sean creados con adecuada entropía, incluso cuando la aplicación se encuentre bajo carga intensa, o que la aplicación se degrade armoniosamente en tales circunstancias.			✓	3.0

Referencias

Para obtener más información, consulte:

- Guía de pruebas OWASP 4.0: Pruebas para criptografía débil https://www.owasp.org/index.php/Testing_for_weak_Cryptography
- Cheat Sheet: Almacenamiento criptográfico https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet

V8: Requisitos de verificación de gestión y registro de errores

Objetivo de control

El objetivo principal de la gestión y registro de errores es proporcionar una reacción útil para los usuarios, administradores y equipos de respuesta a incidentes. El objetivo no es crear cantidades masivas de registros, sino crear registros de alta calidad, con información útil y desechando ruido.

Los registros de bitácora de alta calidad a menudo contienen datos confidenciales y también deben ser protegidos según las leyes de privacidad de datos o directivas. Esto debe incluir:

- No recoger o registrar información confidencial si no es necesaria.
- Garantizar que toda la información registrada se gestiona de forma segura y es protegida según su clasificación de datos.
- Asegurar que los registros de bitácora no sean almacenados indeterminadamente, sino que posean un ciclo de vida útil lo más corta posible.

Si los registros contienen datos privados o confidenciales, cuya definición varía de país a país, éstos se convierten en parte de la información sensible y por lo tanto resulta muy atractiva para los atacantes.

Requisitos

#	Descripción	1	2	3	Desde
8.1	Verificar que la aplicación no emita mensajes de error o rastros de pilas que contengan datos sensibles que podrían ayudar a un atacante, incluyendo el identificador de sesión, versiones de software/entorno y datos personales.	✓	✓	✓	1.0
8.2	Verificar que la lógica de manejo de errores en controles de seguridad niegue el acceso por defecto.		✓	✓	1.0
8.3	Verificar que los controles del registro de seguridad proporcionen la capacidad para registrar los eventos de éxito y sobre todo los eventos de falla que son identificados como relevantes para la seguridad.		✓	✓	1.0
8.4	Verificar que cada registro de evento incluya la información necesaria para permitir una eventual investigación y correlación con otros eventos.		✓	✓	1.0

8.5	Verificar que todos los eventos que incluyen datos no confiables no se ejecuten como código en el software destinado a la visualización del registro.			✓	1.0
8.6	Verificar que los registros de seguridad estén protegidos contra modificación y acceso no autorizado.		✓	✓	1.0
8.7	Verificar que la aplicación no registre datos sensibles definidos en las leyes o regulaciones de privacidad local, datos organizacionales sensibles definidos por una evaluación de riesgos, o datos de autenticación sensible que podrían ayudar a un atacante, incluyendo identificadores de sesión del usuario, contraseñas, hashes o tokens de APIs.		✓	✓	3.0
8.8	Verificar que todos los símbolos no imprimibles y separadores de campos estén codificados correctamente en las entradas del registro, para evitar la inyección del registro que no permita seguir las pistas de un acto malicioso.			✓	2.0
8.9	Verificar que los campos del registro de fuentes confiables y no confiables sean identificables en las entradas del registro.			✓	2.0
8.10	Verificar que un registro de auditoría o similar permita la no repudiación de transacciones claves.		✓	✓	3.0
8.11	Verificar que los registros de seguridad poseen alguna forma de verificación o control de integridad para prevenir modificaciones no autorizadas.			✓	3.0
8.12	Verificar que los registros estén almacenados en una partición diferente a donde ejectua la aplicación con una rotación de registros adecuada.			✓	3.0

Referencias

Para obtener más información, consulte:

- OWASP Testing Guide 4.0: Pruebas de gestión de Errores
  https://www.owasp.org/index.php/Testing_for_Error_Handling

V9: Requisitos de Verificación de Protección de Datos

Objetivo de control

Hay tres elementos clave para la protección de datos: Confidencialidad, Integridad y Disponibilidad (CIA por sus siglas en inglés). Este estándar asume que la protección de datos se aplica en un sistema de confianza, como un servidor, que ha sido protegido debidamente y dispone de protecciones suficientes.

Las aplicaciones web deben asumir que todos los dispositivos de un usuario puedan ser comprometidos de alguna manera. Cuando una aplicación transmite o almacena información sensible dentro de dispositivos inseguros, como equipos compartidos, teléfonos y tabletas, la aplicación es responsable de que los datos almacenados en estos dispositivos sean cifrados y no pueden ser fácilmente o ilícitamente obtenidos, alterados o divulgados.

Se debe asegurar que la aplicación verificada satisface los siguientes requisitos de protección de datos de alto nivel:

- **Confidencialidad**: los datos deben ser protegidos de observación no autorizada o la divulgación tanto en tránsito como cuando están almacenados.
- **Integridad**: los datos deben protegerse siendo creados maliciosamente, alterados o eliminados por los intrusos no autorizados.
- **Disponibilidad**: los datos deben estar disponibles para usuarios autorizados cuando sea necesario

Requisitos

#	Descripción	1	2	3	Desde
9.1	Verificar que todos los formularios que contengan información sensible se les haya desactivado el almacenamiento de caché en el cliente, incluyendo funciones de autocompletar.	✓	✓	✓	1.0
9.2	Verificar que la lista de datos sensibles procesados por la aplicación se encuentra identificada, y que existe una política explícita de cómo debe controlarse el acceso a estos datos, cifrarse y reforzarse bajo las directivas de protección de datos pertinentes.			✓	1.0

#	Descripción	1	2	3	Desde
9.3	Verificar que toda información sensible es envíada al servidor en el cuerpo o cabeceras del mensaje HTTP (por ejemplo, los parámetros de la URL nunca se deben utilizar para enviar datos sensibles).	✓	✓	✓	1.0

9.4	Verificar que la aplicación establece encabezados anti-caché adecuados según el riesgo de la aplicación, tales como las siguientes:
	`Expires: Tue, 03 Jul 2001 06:00:00 GMTT`
	`Last-Modified: {now} GMT`
	`Cache-Control: no-store, no-cache, must-revalidate, max-age=0`
	`Cache-Control: post-check = 0, pre-check = 0`
	`Pragma: no-cache`
		✓	✓	✓	1.0

9.5	Verificar que, en el servidor, todas las copias almacenadas en caché o temporales de datos sensibles estén protegidos de accesos no autorizados o son purgados/invalidados después del acceso por parte del usuario autorizado.		✓	✓	1.0
9.6	Verificar que existe un mecanismo para eliminar de la aplicación todo tipo de dato sensible luego de transcurrido el tiempo definido por la política de retención.			✓	1.0
9.7	Verificar que la aplicación reduce al mínimo el número de parámetros en una solicitud, como campos ocultos, variables de Ajax, cookies y valores en encabezados.		✓	✓	2.0
9.8	Verificar que la aplicación tenga la capacidad para detectar y alertar sobre un número anormal de solicitudes para la recolección de datos por medio de extracción de pantalla (screen scrapping).			✓	2.0
9.9	Verificar que datos almacenados en el cliente (como almacenamiento local de HTML5, almacenamiento de la sesión, IndexedDB, cookies normales o las cookies de Flash) no contengan información sensible o información personal identificable.	✓	✓	✓	3.0.1
9.10	Verificar que el acceso a datos sensibles es registrado en bitácora, los datos son registrados acorde a las directivas de protección de datos o cuando el registro de los accesos es requerido.		✓	✓	3.0
9.11	Verificar que la información sensible mantenida en memoria es sobre escrita con ceros tan pronto como no es requerida, para mitigar ataques de volcado de memoria.		✓	✓	3.0.1

Referencias

Para obtener más información, consulte:

- Cheat Sheet - Protección de la privacidad del usuario
  https://www.owasp.org/index.php/User_Privacy_Protection_Cheat_Sheet

V10: Requisitos de Verificación de Seguridad de las Comunicaciones

Objetivo de control

Se debe asegurar que la aplicación verificada satisfaga los siguientes requisitos de alto nivel:

- Que se utilice TLS donde se transmite información sensible
- Que se utilicen algoritmos y cifradores fuertes en todo momento.

Requisitos

#	Descripción	1	2	3	Desde
10.1	Verificar que puede construirse la cadena de confianza desde una CA (Autoridad de Certificación) para cada certificado TLS (Transport Layer Security) del servidor, y que cada certificado del servidor sea válido.	✓	✓	✓	1.0
10.3	Verificar que se utiliza TLS para todas las conexiones (incluyendo conexiones back-end y externas) autenticadas o que involucran funciones o información sensible, y no recaigan en protocolos inseguros o sin cifrado. Asegúrese de que la alternativa más fuerte es el algoritmo preferido.	✓	✓	✓	3.0
10.4	Verificar que se registran los fallos de conexiones TLS en el backend.			✓	1.0
10.5	Verificar que se construyen las cadenas de confianza para todos los certificados de clientes mediante anclajes de confianza e información de revocación de certificados.			✓	1.0
10.6	Verificar que todas las conexiones a sistemas externos que involucran acciones o información sensible sean autenticadas.		✓	✓	1.0
10.8	Verificar que haya una sola implementación estándar de TLS utilizada por la aplicación la cual esté configurada para operar en un modo aprobado de operación.			✓	1.0
10.10	Verificar que el certificado de clave pública se encuentre fijado (Certificate Pinning) con la clave de producción y la clave pública de respaldo. Para obtener más información, vea las referencias abajo.			✓	3.0.1
10.11	Verificar que los encabezados HTTP Strict Transport Security sean incluidos en todas las peticiones y para todos los subdominios, como Strict-Transport-Security: max-age = 15724800; includeSubdomains	✓	✓	✓	3.0

#	Descripción	1	2	3	Desde
10.12	Verificar que la URL del sitio web de producción haya sido enviada a una lista precargada de dominios de Strict Transport Security(STS) mantenidos por proveedores de navegadores web. Para obtener más información, vea las referencias abajo.			✓	3.0
10.13	Asegurar que *foward secrecy* se esté utilizando para mitigar que atacantes pasivos puedan grabar el tráfico.	✓	✓	✓	3.0
V10.14	Verificar que una adecuada revocación de certificados, tal como el protocolo de estatus de certificado en línea (OSCP), está habilitado y configurado para determinar el estado de vigencia del certificado.	✓	✓	✓	3.0
V10.15	Verificar que se utilicen únicamente algoritmos, cifradores y protocolos fuertes, a través de toda la cadena de confianza, incluyendo certificados raíz y certificados intermediarios de la autoridad certificadora seleccionada.	✓	✓	✓	3.0
V10.16	Verificar que la configuración de TLS esté en línea con las mejores prácticas actuales, particularmente debido a que configuraciones comunes se convierten en inseguras a medida que transcurre el tiempo.	✓	✓	✓	3.0

Referencias

Para obtener más información, consulte:

- Cheat Sheet: TLS https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet
- Notas sobre "Modos aprobados de TLS": En el pasado, el ASVS hizo referencia al estándar estadounidense FIPS 140-2, pero como es un estándar global, la aplicación de estándares americanos puede resultar difícil, contradictorio o confuso de aplicar. Un mejor método de cumplimiento para el punto 10.8 es revisar guías tales como (https://wiki.mozilla.org/Security/Server_Side_TLS), generar configuraciones bien conocidas (https://mozilla.github.io/server-side-tls/ssl-config-generator/) y utilizar herramientas de evaluación de TLS conocidas, como sslyze, diversos escáneres de vulnerabilidades o servicios confiables de evaluación en línea de TLS para obtener un nivel de seguridad deseado. En general, vemos el incumplimiento de esta sección

debido al uso de cifradores anticuados, algoritmos inseguros, la falta de *perfect secret fowarcy*, protocolos de SSL obsoletos o inseguros, algoritmos de Cifrado débiles y así sucesivamente.

- Fijación de Certificado: Por mas información consulte https://tools.ietf.org/html/rfc7469 . La razón de ser tras el fijado de certificados para producción y copia de claves es la continuidad del negocio - vea https://noncombatant.org/2015/05/01/about-http-public-key-pinning/
- Pre-Cargamiento de HTTP Transporte de Seguridad Estricto: https://www.Chromium.org/hsts

V11: Requisitos de verificación de configuración de seguridad HTTP

Objetivo de control

Asegure que la aplicación verificada satisfaga los siguientes requisitos de alto nivel:

- El servidor de aplicaciones está convenientemente endurecido de una configuración preestablecida
- Toda respuesta HTTP contiene su tipo de contenido establecido utilizando un conjunto de caracteres seguro.

Requisitos

#	Descripción	1	2	3	Desde
11.1	Verificar que la aplicación acepte solo un conjunto definido de métodos de solicitud HTTP y que son necesarios, como GET y POST, y métodos no utilizados (por ejemplo: TRACE, PUT y DELETE) se encuentran explícitamente bloqueados.	✓	✓	✓	1.0
11.2	Verificar que cada respuesta HTTP contenga una cabecera content-type en la que se especifique un conjunto utilizando un conjunto de caracteres seguros (Ejemplo: UTF-8, ISO 8859-1).	✓	✓	✓	1.0
11.3	Verificar que los encabezados HTTP agregados por un proxy confiable o dispositivos SSO, tales como un token de portador (bearer), son autenticados por la aplicación.		✓	✓	2.0
11.4	Verificar que el cabezal X-FRAME-OPTIONS se encuentra especificado para los sitios que no deben ser embebidos en X-Frame en sitios de terceros		✓	✓	3.0.1
11.5	Verificar que los encabezados HTTP o cualquier parte de la respuesta HTTP no expongan información detallada de la versión de los componentes del sistema.	✓	✓	✓	2.0
11.6	Verificar que todas las respuestas del API contienen opciones X-Content-Type: nosniff y Content-Disposition: attachment; filename="api.json" (u otro nombre de archivo apropiado para el tipo de contenido).	✓	✓	✓	3.0

11.7	Verificar que la política de seguridad de contenido (CSPv2) está en uso de tal manera que ayude a mitigar vulnerabilidades de inyección comunes de DOM, XSS, JSON y Javascript	✓	✓	✓	3.0.1
11.8	Verificar que el encabezado "X-XSS-Protection: 1; mode=block" esté presente para habilitar a los navegadores a filtrar XSS reflejados	✓	✓	✓	3.0

Referencias

Para obtener más información, consulte:

- Guía de pruebas OWASP 4.0: Testeo para la manipulación de verbos HTTP https://www.owasp.org/index.php/Testing_for_HTTP_Verb_Tampering_%28OTG-INPVAL-003%29
- Adición de Content-Disposition a las respuestas API ayuda a prevenir muchos de los ataques basados en errores o malinterpretaciones en el tipo MIME entre cliente y servidor, y la opción de "nombre de archivo" específicamente ayuda a prevenir ataques de tipo Descarga de Archivos Reflejada. https://www.blackhat.com/docs/eu-14/materials/eu-14-Hafif-Reflected-File-Download-A-New-Web-Attack-Vector.pdf

V12: Requisitos de verificación de configuración de seguridad

Esta sección se incorporó en V11 en la versión 2.0 del Estándar de Verificación de Seguridad en Aplicaciones.

V13: Requisitos de verificación para Controles Malicioso

Objetivo de control

Asegure que la aplicación verificada satisfaga los siguientes requisitos de alto nivel:

- La actividad maliciosa se debe manejar con seguridad y adecuadamente para no afectar el resto de la aplicación.
- No posee bombas de tiempo ni otros ataques basados en tiempo
- No realiza "phone home" a destinos malintencionados o no autorizados
- La aplicación no posee puertas traseras, huevos de Pascua, ataques salami o fallos de lógica que pueden ser controlados por un atacante

El código malicioso es extremadamente raro difícil de detectar. La revisión manual línea por línea del código puede ayudar a encontrar bombas lógicas, pero incluso el más experimentado revisor de código tendrá que esforzarse para encontrar código malicioso aunque sepa que existe.

Requisitos

#	Descripción	1	2	3	Desde
13.1	Verificar que toda actividad maliciosa sea adecuadamente aislada o encajonada para retrasar y disuadir a los atacantes de atacar a otras aplicaciones.			✓	2.0
13.2	Verificar que el código fuente de la aplicación y tantas bibliotecas de terceros como sean posibles, no poseen puertas traseras, huevos de pascua, o fallas de lógica en la autenticación, control de acceso, validaciones de entrada y lógica de negocio en transacciones de alto valor.			✓	3.0.1

Referencias

Para obtener más información, consulte:

- http://www.dwheeler.com/essays/apple-goto-fail.html

V14: Requisitos de verificación de seguridad interna

Esta sección se incorporó en V13 en la versión 2.0 del Estándar de Verificación de Seguridad en Aplicaciones.

V15: Requisitos de verificación para lógica de negocios

Objetivo de control

Asegure que la aplicación verificada satisfaga los siguientes requisitos de alto nivel:

- El flujo de la lógica de negocio es secuencial y en orden
- La Lógica de negocios incluye límites para detectar y evitar ataques automatizados, como las continuas transferencias de fondos pequeños, agregando 1 millón amigos uno a uno y así sucesivamente.
- Flujos de lógica de negocios de alto valor han considerado casos de abuso y agentes maliciosos y poseen protecciones contra la falsificación, alteración, repudio, revelación de información y ataques a la elevación de privilegios.

Requisitos

#	Descripción	1	2	3	Desde
V15.1	Verificar que la aplicación sólo procese flujos lógicos de negocios en orden secuencial, con todos los pasos procesados en tiempo humano realista, y no procesados fuera de orden, con pasos salteados, con pasos del proceso de otro usuario, o de transacciones muy rápidamente enviadas.		✓	✓	2.0
V15.2	Verificar que la aplicación tiene límites de negocio y los aplique correctamente por cada usuario, con alertas configurables y reacciones automatizadas ante ataques inusuales o automáticos.		✓	✓	2.0

Referencias

Para obtener más información, consulte:

- Guía de pruebas OWASP 4.0: Pruebas de lógica de negocios https://www.OWASP.org/index.php/Testing_for_business_logic
- Cheat Sheet: Lógica de Negocio https://www.OWASP.org/index.php/Business_Logic_Security_Cheat_Sheet

V16: Requisitos de verificación de archivos y recursos

Objetivo de control

Asegure que la aplicación verificada satisfaga los siguientes requisitos de alto nivel:

- Datos no confiables deben ser gestionados como tales y de forma segura
- Datos Obtenidos de fuentes no confiables sean almacenan fuera del webroot y posean permisos limitados.

Requisitos

#	Descripción	1	2	3	Desde
16.1	Verificar que las URL de redireccionen y reenvíen sólo a destinos clasificados en la lista blanca, o mostrar una advertencia cuando se redirija a contenido potencialmente no confiable.	✓	✓	✓	2.0
16.2	Verificar que archivos no confiables enviados a la aplicación no sean utilizados directamente por comandos de I/O (Entrada/Salida) de archivos, especialmente para proteger contra manipulaciones de rutas, archivo local incluido, manipulación de tipo mime y vulnerabilidades de inyección de comandos de sistema operativo.	✓	✓	✓	2.0
16.3	Verificar que los archivos procedentes de fuentes no confiables sean validados para ser del tipo del cual se espera y sean analizados por escáneres antivirus para evitar la carga de contenido malicioso conocido.	✓	✓	✓	2.0
16.4	Verificar que datos no confiables no se utilicen en funcionalidades de reflexión, cargado de clases o inserción para prevenir vulnerabilidades de inclusión de archivos remotos/locales.	✓	✓	✓	2.0
16.5	Verificar que datos no confiables no se utilicen en recursos de dominios compartidos (CORS) para proteger contra el contenido remoto arbitrario.	✓	✓	✓	2.0
16.6	Verificar que los archivos obtenidos de fuentes no confiables se almacenen fuera del webroot, con permisos limitados, preferiblemente con una fuerte validación.		✓	✓	3.0

#	Descripción	1	2	3	Desde
16.7	Verificar que el servidor web o de aplicación se encuentra configurado por defecto para negar el acceso a recursos remotos o sistemas fuera del servidor web o de aplicación.		✓	✓	2.0
16.8	Verificar que el código de la aplicación no ejecuta datos cargados obtenidos de fuentes no confiables.	✓	✓	✓	3.0
16.9	Verificar que no utiliza Flash, Active-X, Silverlight, NACL, Java del lado del cliente u otras tecnologías del lado del cliente que no sean soportadas de forma nativa a través de los estándares de navegador W3C.	✓	✓	✓	2.0

Referencias

Para obtener más información, consulte:

- Manejo de Extensión de Archivo para información confidencial: https://www.owasp.org/index.php/Unrestricted_File_Upload

V17: Requisitos de verificación Móvil

Objetivo de control

Esta sección contiene controles específicos para aplicaciones móviles. Estos controles han sido de-duplicados de la versión 2.0, por lo que deben tomarse en conjunto con el resto de las secciones de los niveles correspondientes de Verificación ASVS.

Las aplicaciones móviles deben:

- Deben tener el mismo nivel de controles de seguridad tanto en el cliente móvil como en el servidor, mediante la aplicación de controles de seguridad en un entorno de confianza.
- Activos de Información sensible almacenados en el dispositivo debe realizarse de un modo seguro.
- Todos los datos sensibles transmitidos desde el dispositivo deben ser hechos teniendo en mente la seguridad en la capa de transporte.

Requisitos

#	Descripción	1	2	3	Desde
17.1	Verificar que los valores de identificadores almacenados en el dispositivo y recuperables por otras aplicaciones, como el número de UDID o IMEI no se utilicen como tokens de autenticación.	✓	✓	✓	2.0
17.2	Verificar que la aplicación móvil no almacene datos sensibles en recursos compartidos potencialmente no cifrados en el dispositivo (tarjeta SD o carpetas compartidas).	✓	✓	✓	2.0
17.3	Verificar que los datos sensibles no se almacenen sin protección en el dispositivo, incluso en áreas protegidas del sistema como llaveros.	✓	✓	✓	2.0
17.4	Verificar que las contraseñas, claves secretas y tokens de APIs se generan dinámicamente en la aplicación móvil.		✓	✓	2.0

#	Descripción	1	2	3	Desde
17.5	Verificar que la aplicación móvil evite fugas de información sensible (por ejemplo, capturas de pantalla de la aplicación actual sean guardadas mientras la aplicación está en segundo plano o escribiendo información sensible en consola).		✓	✓	2.0
17.6	Verificar que la aplicación solicita permisos mínimos para la funcionalidad y recursos requeridos.		✓	✓	2.0
17.7	Verificar que el código sensible de la aplicación sea cargado de forma no predecible en memoria (utilizando ASLR por ejemplo).	✓	✓	✓	2.0
17.8	Verificar que se encuentren presentes técnicas anti depuración suficientes para impedir o retrasar a posibles atacantes de inyectar depuradores en la aplicación móvil (GDB, por ejemplo).			✓	2.0
17.9	Verificar que la aplicación no exporte actividades intents, o proveedores de contenido con información sensible a otras aplicaciones móviles posibles de ser explotados por otras aplicaciones en el mismo dispositivo.	✓	✓	✓	2.0
17.10	Verificar que la información sensible almacenada en memoria es sobreescrita con ceros tan pronto como no deje de ser requerida, con el fin de mitigar ataques de volcado de memoria.			✓	3.0.1
17.11	Verificar que las actividades expuestas, intents, proveedores de contenido realicen validaciones de los datos de entrada.	✓	✓	✓	3.0.1

Referencias

Para obtener más información, consulte:

- Proyecto OWASP de Seguridad Móvil:
  https://www.OWASP.org/index.php/OWASP_Mobile_Security_Project
- iOS Developer Cheat Sheet:
  https://www.OWASP.org/index.php/IOS_Developer_Cheat_Sheet

V18: Requisitos de verificación de servicios Web

Objetivo de control

Asegúrese de que la aplicación verificada, de utilizar servicios web REST o SOAP posean:

- Autenticación adecuada, gestión de sesión y autorización de todos los servicios web
- Validación de entrada de datos de todos los parámetros que transiten de zonas de menor a mayor confianza
- Interoperabilidad básica de la capa de servicios web SOAP para promover el uso de la API.

Requisitos

#	Descripción	1	2	3	Desde
18.1	Verificar que el mismo estilo de codificación se utiliza tanto en el cliente como el servidor.	✓	✓	✓	3.0
18.2	Verificar que el acceso a las funciones de administración y gestión de la aplicación proveedora de servicios web sea limitado a los administradores.	✓	✓	✓	3.0
18.3	Verificar que existen esquemas XML o JSON y que éstos son verificados por la aplicación antes de aceptar datos de entrada.	✓	✓	✓	3.0
18.4	Verificar que todos los datos de entrada se encuentren limitados a un tamaño adecuado.	✓	✓	✓	3.0
18.5	Verificar que los servicios web basados en SOAP son compatibles con el perfil básico de interoperabilidad de servicios Web (WS-I) como mínimo. Esencialmente, eso implica compatibilidad con cifrado TLS.	✓	✓	✓	3.0.1
18.6	Verificar el uso de autenticación y autorización basada en sesiones. Por favor refiérase a las secciones 2, 3 y 4 para mayor orientación. Evite el uso de "claves de API" estáticas y enfoques similares.	✓	✓	✓	3.0

#	Descripción	1	2	3	Desde
18.7	Verificar que los servicios REST se encuentren protegidos de Falsificación de Peticiones en Sitos Cruzados (CSRF), mediante el uso de al menos uno o mas de los siguientes mecanismos: Verificaciones de ORIGIN, CSRF nonces, verificaciones de referrer o el envío doble de valores en cookie y en el servicio (double submit cookie pattern)	✓	✓	✓	3.0.1
18.8	Verificar que los servicios REST comprueben explícitamente que el Content-Type entrante sea el que se espera, como aplicación/xml o aplicación/json.		✓	✓	3.0
18.9	Verificar que el contenido de los mensajes se encuentra firmado para asegurar el transporte confiable entre el cliente y el servicio, utilizando JSON Web Signing o WS-Security para servicios SOAP.		✓	✓	3.0.1
18.10	Verificar que no existen rutas de acceso alternativas y menos seguras.		✓	✓	3.0

Referencias

Para obtener más información, consulte:

- Guía de pruebas de OWASP 4.0: Configuración y Despliegue
  https://www.owasp.org/index.php/Testing_for_configuration_management
- Cheat Sheet: Falsificación de Peticiones en Sitos Cruzados
  https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet
- JSON Web Tokens (y su firma)
  https://jwt.io/

V 19. Requisitos de Configuración

Objetivo de control

Asegúrese de que la aplicación verificada:

- Utilice bibliotecas y una plataforma actualizada.
- Una configuración segura por omisión.
- Un Hardening suficiente de tal forma que los cambios realizados por un usuario no resulten en exposiciones innecesarias o creen debilidades de seguridad o fallas a los sistemas subyacentes.

Requisitos

#	Descripción	1	2	3	Desde
19.1	Todos los componentes deben estar actualizados a las configuraciones y versiones de seguridad adecuadas. Esto debería incluir la eliminación de configuraciones y carpetas innecesarias como aplicaciones de ejemplo, documentación de plataforma y usuarios pre-establecidos o de ejemplo.	✓	✓	✓	3.0
19.2	Las comunicaciones entre componentes, tales como entre el servidor de aplicaciones y el servidor de base de datos, deberían ser cifradas, particularmente cuando los componentes están en diferentes contenedores o en sistemas diferentes.		✓	✓	3.0
19.3	Las comunicaciones entre componentes, tales como entre el servidor de aplicaciones y el servidor de base de datos deberían autenticarse utilizando una cuenta con los mínimos privilegios necesarios.		✓	✓	3.0
19.4	Verificar que los despliegues de la aplicación se encuentren dentro de Sandboxes, en contenedores o aislados para retrasar y disuadir a los atacantes de atacar a otras aplicaciones.		✓	✓	3.0
19.5	Verificar que los procesos de compilación y despliegue de la aplicación se realizan de forma segura.		✓	✓	3.0

#	Descripción	1	2	3	Desde
19.6	Verificar que los administradores autorizados posean la capacidad de verificar la integridad de todas las configuraciones de seguridad pertinentes para garantizar que no hayan sido manipuladas.			✓	3.0
19.7	Verificar que todos los componentes de aplicación se encuentren firmados.			✓	3.0
19.8	Verificar que los componentes de terceros proceden de repositorios de confianza.			✓	3.0
19.9	Verificar que los procesos de compilación para los lenguajes de nivel de sistema operativo tengan todas las banderas de seguridad activas, tales como controles de seguridad, DEP y ASLR.			✓	3.0
19.10	Verificar que todos los recursos de la aplicación se encuentran alojados en la aplicación en vez de confiar en un CDN o proveedores externos, tales como bibliotecas JavaScript, estilos CSS o web fonts			✓	3.0.1

Referencias

Para obtener más información, consulte:

- Guía de pruebas de OWASP 4.0: Configuración y Despliegue
  https://www.owasp.org/index.php/Testing_for_configuration_management

Apéndice A: Qué sucedió con …

# Original	Descripción	Estado	Eliminado	Razón
2.3	Verificar que, si se supera un número máximo de intentos de autenticación, la cuenta será bloqueada por un período de tiempo suficiente para disuadir ataques de fuerza bruta.	Obsoleto	2.0	Substituido por un requerimiento más complejo (v2.20)
2.5	Verificar que todos los controles de autenticación (incluyendo las bibliotecas que requieren servicios de autenticación externos) tengan una implementación centralizada.	Combinado	3.0	Generalizado para incluir todos los controles de seguridad y se trasladó a 1.10
2.10	Verificar que autenticación sea requerida antes de que se permitan cualquier operación sensible específicas de la aplicación.	Obsoleto	2.0	Se quita el control debido a que la re- autenticación se observa raramente.
2.11	Verificar que después de un período de tiempo configurable por un administrador, las credenciales de autenticación expiren.	Obsoleto	2.0	Expiración absoluta de tiempos de espera y expiración credencial ha sido eliminado por no ser un control eficaz.
2.14	Verificar que todas las credenciales de autenticación para acceder a servicios externos a la aplicación se encuentren cifradas y almacenadas en un lugar protegido (no en código fuente).	Actualizado	2.0	Se convirtió en V2.21
2.15	Verificar que todo el código de aplicación o uso de controles de autenticación no es este afectado por cualquier código malicioso.	Se trasladó	2.0	Se trasladó a V13 - código malicioso
2.30	Verificar que si la aplicación permite a los usuarios autenticarse, se provee un mecanismo de autenticación seguro	Obsoleto	3.0.1	Muy ambiguo para ser probado, de hecho, es un resumen de los requerimientos de V2
3.8	Verificar que se modifique el identificador de sesión cuando haya re-autenticación	Actualizado	3.0	Contemplado en 3.7

3.9	Compruebe que el identificador de sesión se cambie o elimine al salir	Actualizado	3.0	Contemplado en 3.7
3.13	Verificar que todo el código de aplicación o uso de controles de gestión de sesión no se encuentre afectado por código malicioso	Se trasladó	2.0	Se trasladó a V13 - código malicioso
3.14	Verificar que los tokens de sesión autenticados usando cookies están protegidos por el uso de "HttpOnly".	Actualizado	3.0	Se trasladó a 3.13
3.15	Verificar que los tokens de sesión autenticados usando cookies están protegidos con el atributo "secure".	Actualizado	3.0	Se trasladó a 3.13
4.2	Verificar que los usuarios sólo pueden acceder a URLs seguras que poseen una autorización específica.	Actualizado	3.0	Contemplado en 4.1
4.3	Verificar que los usuarios sólo puedan acceder a los archivos de datos asegurados para los que poseen una autorización específica.	Actualizado	3.0	Contemplado en 4.1
4.13	Compruebe que las limitaciones en entrada y acceso impuestas por reglas de negocios sobre la aplicación (como límites de transacciones diarias o secuenciación de las tareas) no sean sobrepasadas.	Se trasladó	3.0	Se trasladó a la lógica de negocio de V15
4.15	Verificar que todo el código de aplicación o uso de controles de acceso no se vea afectado por cualquier código malicioso.	Se trasladó	2.0	Se trasladó a controles maliciosos V13
5.2	Verificar que un patrón de validación positivo es definido y aplicado a todas las entradas de datos	Obsoleto	2.0	Removido debido a que es demasiado difícil de aplicar particularmente para entradas de texto de forma libre
5.4	Verificar que se haya especificado un conjunto de caracteres como UTF-8, para todas las fuentes de entrada	Obsoleto	3.0	Removido debido a que es demasiado difícil de aplicar en la mayoría de los lenguajes de programación
5.7	Verificar que todos los fallos de validación de entrada se registren.	Obsoleto	3.0	Removido, debido a que crearía muchos registros

				inútiles que serían ignorados
5.8	Verificar que toda entrada de datos es canónica para todos los decodificadores downstream o intérpretes antes de la validación.	Obsoleto	3.0	Removido ya que el tipo 1 JSP tecnología específica no es un problema para los marcos de desarrollo más modernos
5.9	Verificar que todos los controles de validación de entrada dedatos no sean afectados por cualquier código malicioso	Se trasladó	2.0	Se trasladó a controles maliciosos V13
5.14	Verificar que el entorno de ejecución no es susceptible a las inyecciones de XML o que controles de seguridad impidan inyecciones de XML	Combinado	3.0	Se fusionó con V5.13
5.15	--REQUISITO VACÍO--	Eliminado	3.0	Este requisito nunca existió
5.19	Verificar que para cada tipo de codificación/escape de salida realizado por la aplicación, haya un control de seguridad único para ese tipo de salida para el destino intencionado	Combinado	3.0	Generalizado para incluir todos los controles de seguridad y se trasladó a 1.10
7.1	Verificar que todas las funciones criptográficas utilizadas para proteger secretos del usuario de la aplicación se ejecutan del lado del servidor	Obsoleto	3.0	Muchas de las aplicaciones modernas de respuesta rápida y móvil incluyen esto por diseño
7.3	Verificar que el acceso a cualquier secreto(s) maestro está protegido de accesos no autorizados (un secreto maestro es una credencial de aplicación almacenada como texto simple en el disco que se utiliza para proteger el acceso a la información de configuración de seguridad).	Se trasladó	3.0	Se trasladó a V2.29
7.4	Verificar que los hashes de contraseñas son salados cuando sean creados	Se trasladó	2.0	Se trasladó a V2.13
7.5	Verificar que se registran en una bitácora las fallas de módulo criptográfico	Obsoleto	2.0	Creación de registros log innecesarios que no son revisados nunca es contraproducente

7.10	Verificar que todo el código de soporte o que esté utilizando un módulo criptográfico no sea afectado por cualquier código malicioso	Se trasladó	2.0	Se trasladó a V13
8.2	Verificar que toda la gestión de errores se realice en dispositivos confiables		3.0	Obsoleto
8.3	Verificar que todos los controles de registro se ejecuten en el servidor.	Se trasladó	3.0	Se convirtió en un control arquitectónico más genérico en V1.13
8.9	Verifique que haya una implementación única de registro de bitácora a nivel de la aplicación que sea utilizada por el software.	Se trasladó	3.0	Se convirtió en un control arquitectónico más genérico en V1.13
8.11	Verificar que una herramienta de análisis del registros de bitácora se encuentre disponible al analista el cual le permita buscar eventos basados en combinaciones de criterios de búsqueda en todos los campos en el formato de registro log compatibles con este sistema.	Obsoleto	3.0	Removido ya que no requiere de software seguro
8.12	Verificar que todo el código de la aplicación o uso de controles de registro de bitácora y control de errores no sea afectado por cualquier código malicioso.	Se trasladó	2.0	Se trasladó a controles maliciosos V13
8.15	Verificar que el registro de bitácora se realice antes de ejecutar la transacción. Si el registro de bitácora no tuvo éxito (p. ej. disco completo, permisos insuficientes) la aplicación falla de forma segura. Esto es, para cuando la integridad y no repudio son imprescindibles.	Obsoleto	3.0	Eliminado debido a que es control demasiado detallado que sólo sería aplicable a un pequeño porcentaje de todas las applicaciones
10.2	Verificar que conexiones TLS no caigan de nuevo a una conexión insegura HTTP	Combinado	3.0	Se fusionó con 10.3
10.7	Verificar que todas las conexiones a sistemas externos que involucran información confidencial o usen una cuenta que se ha establecido con privilegios mínimos necesarios para que la aplicación funcione correctamente			
10.9	Verificar que codificaciones de caracteres específicos se definen para todas las			

	conexiones (por ejemplo, UTF-8).			
V11.1	Obsoleto			
V11.4	Obsoleto			
V11.5	Obsoleto			
V11.6	Obsoleto			
V11.7	Obsoleto			
V11.8	Obsoleto			
V11.4	Obsoleto			
V13.1	Obsoleto			
V13.2	Obsoleto			
V13.3	Obsoleto			
V13.4	Obsoleto			
V13.5	Obsoleto			
V13.6	Obsoleto			
V13.7	Obsoleto			
V13.8	Obsoleto			
V13.9	Obsoleto			
15.1-15.7 15.9	Sección de la lógica de negocios.	Combinado	3.0	La mayor parte de la sección 15 se ha fusionado en 15.8 y 15.10.
15.11	Verificar que la aplicación se protege de los riesgos asociados con la suplantación de identidad, manipulación, repudio, revelación de información y elevación de privilegios	Duplicar	3.0	Requisito de duplicados. Capturado por V1.6

	(STRIDE).			
16.4	Verificar que no se utilizan parámetros obtenidos de fuentes no confiables en la manipulación de los nombres de archivo, rutas de acceso o cualquier objeto de sistema de archivo sin ser la primera canónica y entrada validada para prevenir ataques de inclusión de archivo local.	Se trasladó	3.0	Se trasladó a V16.2
17.1	Verificar que el cliente valide certificados SSL	Obsoleto	3.0	Requisito duplicado. El requisito general se encuentra ya capturado por V10.
V17.7	Obsoleto			
V17.8	Obsoleto			
V17.10	Obsoleto			
V17.11	Obsoleto			
V17.12	Obsoleto			
V17.13	Obsoleto			
V17.14	Obsoleto			
V17.15	Obsoleto			
V17.16	Obsoleto			
V17.17	Obsoleto			
V17.18	Obsoleto			
V17.19	Obsoleto			
V17.20	Obsoleto			
V17.22	Obsoleto			

V17.23	Obsoleto			
V17.24	Obsoleto			

Apéndice B: Glosario

- **Control de acceso** – una forma de restringir el acceso a archivos, funciones de referencias, URLs y datos basados en la identidad de los usuarios o grupos a que pertenecen.
- **Address Space Layout Randomization (ASLR) (ASLR)** – una técnica para ayudar a proteger contra ataques de desbordamiento de búfer.
- **Aplicaciones de seguridad**- La seguridad a nivel de aplicación se centra en el análisis de los componentes que conforman la capa de aplicación del -Modelo de Referencia de Interconexión de sistema abierto (modelo OSI), en lugar de centrarse en por ejemplo el sistema operativo subyacente o redes conectadas.
- **Verificación de seguridad en aplicaciones**, la evaluación técnica de una aplicación contra el ASVS.
- **Informe de verificación de seguridad de aplicación** – un informe que documenta los resultados generales y análisis de apoyo producido por el verificador para una aplicación particular.
- **Autenticación**: verificación de la identidad reivindicada de usuario de una aplicación.
- **Verificación automatizada** – el uso de herramientas automatizadas (herramientas de análisis dinámico, las herramientas de análisis estático o ambos) que utilizan firmas de vulnerabilidad para encontrar problemas.
- **Puertas traseras** – un tipo de código malicioso que permite el acceso no autorizado a una aplicación.
- **Lista negra** – una lista de datos u operaciones que no se permiten, por ejemplo, una lista de caracteres que no se permite como entrada.
- **Hojas de Estilos en Cascada (CSS)** - un lenguaje de hoja de estilo utilizado para describir la semántica de la presentación del documento escrito en un lenguaje de marcado, como HTML.
- **Autoridad de certificación (CA)** – una entidad que emite los certificados digitales.

- **Seguridad de las comunicaciones**, la protección de datos de aplicaciones cuando se transmite entre los componentes de aplicación, entre clientes y servidores y entre sistemas externos y la aplicación.
- **Componente** – una unidad independiente de código, con interfaces de red y disco asociadas que se comunica con otros componentes.
- **Cross-Site Scripting (XSS)** – una vulnerabilidad de seguridad se encuentra típicamente en aplicaciones que permiten la inyección de secuencias de comandos de cliente en contenido web.
- **Módulo criptográfico** – Hardware, software o firmware que implementa algoritmos criptográficos o genera claves criptográficas.
- **Ataques de denegación de servicio (DoS)** –la inundación de una aplicación con más peticiones que puede manejar.
- **Verificación del diseño** – la evaluación técnica de la arquitectura de seguridad de una aplicación.
- **Verificación dinámica** – el uso de herramientas automatizadas que utilizan firmas de vulnerabilidad para encontrar problemas durante la ejecución de una aplicación.
- **Huevos de Pascua** – un tipo de código malicioso que no se ejecuta hasta que se produzca un evento de entrada de usuario específico.
- **Sistemas externos** – una aplicación de servidor o servicio que no es parte de la aplicación.
- **FIPS 140-2**, un estándar/norma que puede utilizarse como base para la verificación del diseño y aplicación de módulos criptográficos
- **Identificador único global (GUID)** – un número de referencia único utilizado como un identificador de software.
- **Lenguaje de marcado de hipertexto (HTML)** – el lenguaje de marcado principal para la creación de páginas web y otra información mostrada en el navegador web.
- **Hyper Text Transfer protocolo (HTTP)** –un protocolo de aplicación para sistemas de información distribuido, colaborativo hipermedia. Es la base de datos de comunicación de la World Wide Web.

- **Validación de entrada** – la canonización y la validación de entrada de usuario no es de confianza.
- **Protocolo ligero de acceso a directorios (LDAP)** – un protocolo de aplicación para el acceso y mantenimiento de servicios de información de directorio distribuida sobre una red.
- **Código malicioso** – código introducido en una aplicación durante su desarrollo desconocido al dueño de la aplicación, que elude la política de seguridad de la aplicación. ¡No es lo mismo un software malicioso a un virus o un gusano!
- Malware – código ejecutable que se introduce en una aplicación en tiempo de ejecución sin el conocimiento del usuario de la aplicación o el administrador.
- **Proyecto Abierto de Seguridad en aplicación Web (OWASP)** –es una comunidad libre y abierta en todo el mundo enfocada en mejorar la seguridad de software en aplicaciones. Nuestra misión es hacer «visible», la seguridad de aplicaciones para que personas y organizaciones pueden tomar decisiones informadas sobre los riesgos de seguridad de la aplicación. Ver: http://www.owasp.org/
- **Codificación de salida** – la canonización y la validación de solicitud de salida para los navegadores Web y sistemas externos.
- **Información de identificación personal (IPI)** – es información que puede utilizarse por sí solo o con otra información para identificar, contactar o localizar a una sola persona, o para identificar a un individuo en contexto.
- V**alidación Positiva** – véase lista blanca.
- **Arquitectura de seguridad** – una abstracción del diseño de una aplicación que identifica y describe los controles de seguridad sobre dónde y cómo se utilizan, identifica y describe la ubicación y la sensibilidad de los datos de usuario y la aplicación.
- **Configuración de seguridad** –la configuración de tiempo de ejecución de una aplicación que afecta a cómo se utilizan los controles de seguridad.
- **Control de seguridad** – una función o componente que realiza una comprobación de seguridad (por ejemplo, una verificación de control de acceso) o cuando resultados

llamados resultan en efecto de seguridad (por ejemplo, generando un registro de auditoría).

- **SQL Injection (SQLi)** – una técnica de inyección de código utilizada para atacar aplicaciones de datos, en que se insertan sentencias SQL maliciosas en un punto de entrada.
- **Verificación estática** – el uso de herramientas automatizadas que utilizan firmas de vulnerabilidad para encontrar problemas en el código fuente de la aplicación.
- **Objetivo de la verificación (TOV)** – si usted está realizando una verificación de seguridad en la aplicación según los requisitos del ASVS, la verificación será de un uso particular. Esta aplicación se llama "El objetivo de la verificación" o simplemente el TOV.
- **Modelado de Amenazas** – una técnica que consiste en desarrollar cada vez más arquitecturas refinadas de seguridad para identificar agentes de amenaza, las zonas de seguridad, controles de seguridad y recursos técnicos y de negocios importantes.
- **Seguridad de capa de transporte** – protocolos criptográficos que proporcionan la seguridad de las comunicaciones por Internet
- **Fragmentos de URL/URI/URL** – un identificador uniforme de recursos es una cadena de caracteres utilizado para identificar un nombre o un recurso web. Un localizador uniforme de recursos a menudo se utiliza como una referencia a un recurso.
- **Aceptación la prueba del usuario (UAT)** – tradicionalmente un entorno de prueba que se comporta como el entorno de producción donde se realizan todas las pruebas de software antes de desplegar la aplicación en vivo.
- **Verificador** – la persona o equipo que se está revisando una aplicación contra los requisitos del ASVS.
- **Lista blanca** – una lista de datos permitidos u operaciones, por ejemplo, una lista de caracteres permitidos para realizar la validación de entrada.
- **XML** – un lenguaje de marcado que define un conjunto de normas de codificación de documentos.

Anexo C: Referencias

Los siguientes proyectos de OWASP son muy probablemente útiles para los usuarios/adoptantes de esta norma:

- Guía de pruebas de OWASP

  https://www.owasp.org/index.php/OWASP_Testing_Project
- Guía de revisión de código de OWASP

  http://www.owasp.org/index.php/Category:OWASP_Code_Review_Project
- Cheat Sheets de OWASP

  https://www.owasp.org/index.php/OWASP_Cheat_Sheet_Series
- Controles proactivos de OWASP

  https://www.owasp.org/index.php/OWASP_Proactive_Controls
- OWASP Top 10

  https://www.owasp.org/index.php/Top_10_2013-Top_10
- OWASP Mobile Top 10

  https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks

Del mismo modo, los siguientes sitios web probablemente sean de utilidad para los usuarios/adoptantes de esta norma:

- MITRE Common Weakness Enumeration

  http://cwe.mitre.org/
- PCI Security Standards Council

  https://www.pcisecuritystandards.org
- PCI Data Security Standard (DSS) v3.0 Requirements and Security Assessment Procedures

  https://www.pcisecuritystandards.org/documents/PCI_DSS_v3.pdf

Apéndice D: Correlación con otras normas

PCI DSS 6.5 se deriva de la OWASP Top 10 2004/2007, con algunas extensiones recientes del proceso. El ASVS es un superconjunto estricto del OWASP Top 10 2013 (154 artículos a 10 artículos), así que todas las cuestiones cubiertas por OWASP Top 10 y PCI DSS 6.5.x son manejadas finamente por requisitos de control ASVS. Por ejemplo, «Pérdida de autenticación y gestión de sesiones» conecta exactamente a las secciones de autenticación V2 y V3 Manejo de la sesión.

La correlación completa se logra al nivel de verificación 3, aunque el nivel de verificación 2 abordará más los requisitos de PCI DSS 6.5 excepto 6.5.3 y 6.5.4. Problemas de proceso, tales como PCI DSS 6.5.6, no están cubiertos por el ASVS.

PCI-DSS 3.0	ASVS 3.0	Descripción
6.5.1 fallas de inyección, particularmente de inyección SQL. También se consideran fallas de inyección de comandos del sistema operativo y LDAP, XPath, así como otras fallas de inyección	5.11, 5.12, 5.13, 8.14, 16.2	Correlación exacta.
6.5.2 Desbordamientos de buffer	5.1	Correlación exacta
6.5.3 Almacenamiento criptográfico inseguro	v7 - todos	Correlación completa de nivel 1
6.5.4 Comunicaciones Inseguras	v10 - todos	Correlación completa de nivel 1
6.5.5 Manejo incorrecto de errores	3.6, 7.2, 8.1, 8.2	Correlación exacta
6.5.7 Cross-site scripting (XSS)	5.16, 5.20, 5.21, 5.24, 5.25, 5.26, 5.27, 11.4,11.15	ASVS desglosa XSS en varios requisitos destacando la complejidad de la defensa XSS, en especial para aplicaciones legadas
6.5.8 Controles de acceso inapropiados (como referencias directas a objetos inseguros, no restringir acceso URL, el salto de directorio y falta restringir acceso de usuario a las funciones).	v4 - todos	Correlación completa de nivel 1

6.5.9 Falsificación de peticiones en sitios cruzados (CSRF).	4.13	Correlación exacta. ASVS considera la defensa de CSRF un aspecto de control de acceso.
6.5.10 Pérdida de autenticación y gestión de sesiones.	v2 y v3 - todos	Correlación completa de nivel 1