Application Security Verification Standard 4.0.3

Final

Octubre 2021

Table of Contents

Frontispicio

Acerca del Estándar

El Estándar de Verificación de Seguridad en Aplicaciones (ASVS; por sus siglas en inglés) es una lista de requisitos o pruebas de seguridad en aplicaciones que puede ser utilizado por arquitectos, desarrolladores, probadores, profesionales de la seguridad, proveedores de herramientas y consumidores para definir, construir, probar y verificar aplicaciones seguras.

Derechos de Autor y Licencia

Version 4.0.3, Octubre 2021

Líderes de Proyecto

Andrew van der Stock	Daniel Cuthbert	Jim Manico
Josh C Grossman	Elar Lang	

Principales Colaboradores

Abhay Bhargav	Benedikt Bauer	Osama Elnaggar
Ralph Andalis	Ron Perris	Sjoerd Langkemper
Tonimir Kisasondi		

Otros Colaboradores y Revisores

Aaron Guzman	Alina Vasiljeva	Andreas Kurtz	Anthony Weems	Barbara Schachner
Christian Heinrich	Christopher Loessl	Clément Notin	Dan Cornell	Daniël Geerts
David Clarke	David Johansson	David Quisenberry	Elie Saad	Erlend Oftedal
Fatih Ersinadim	Filip van Laenen	Geoff Baskwill	Glenn ten Cate	Grant Ongers
hello7s	Isaac Lewis	Jacob Salassi	James Sulinski	Jason Axley
Jason Morrow	Javier Dominguez	Jet Anderson	jeurgen	Jim Newman
Jonathan Schnittger	Joseph Kerby	Kelby Ludwig	Lars Haulin	Lewis Ardern
Liam Smit	lyz-code	Marc Aubry	Marco Schnüriger	Mark Burnett
Philippe De Ryck	Ravi Balla	Rick Mitchell	Riotaro Okada	Robin Wood
Rogan Dawes	Ryan Goltry	Sajjad Pourali	Serg Belkommen	Siim Puustusmaa
Ståle Pettersen	Stuart Gunter	Tal Argoni	Tim Hemel	Tomasz Wrobel
Vincent De Schutter	Mike Jang			

Si falta un crédito en la lista de créditos de la 4.0.3 que aparece arriba, registre un ticket en GitHub para que se le reconozca en futuras actualizaciones.

El Estándar de Verificación de Seguridad de Aplicaciones está construido sobre los hombros de aquellos que participaron desde el ASVS 1.0 en 2008 hasta el 3.0 en 2016. Gran parte de la estructura y de los elementos de verificación que todavía están en el ASVS hoy fueron escritos originalmente por Mike Boberski, Jeff Williams y Dave Wichers, pero hay muchos más contribuyentes. Gracias a todos los que han participado anteriormente. Para obtener una lista completa de todos aquellos que han contribuido a versiones anteriores, por favor consulte cada versión anterior.

Prefacio

Bienvenido al Estándar de Verificación de Seguridad en Aplicaciones (ASVS; por sus siglas en inglés) versión 4.0. El ASVS es un esfuerzo impulsado por la comunidad para establecer un marco de requisitos y controles de seguridad que se centran en definir los controles de seguridad funcionales y no funcionales requeridos al diseñar, desarrollar y probar aplicaciones web y servicios web modernos.

La versión 4.0.3 es el tercer parche menor de la v4.0 destinado a corregir errores ortográficos y hacer que los requisitos sean más claros sin realizar cambios importantes como requisitos que cambien materialmente, fortalecer los requisitos o agregar requisitos. Sin embargo, es posible que algunos requisitos se hayan debilitado ligeramente cuando nos pareció apropiado y se han eliminado algunos requisitos completamente redundantes (pero sin volver a numerarlos).

ASVS v4.0 es la culminación del esfuerzo de la comunidad y los comentarios de lecciones aprendidas en la industria durante la última década. Hemos intentado facilitar la adopción de ASVS para una variedad de diferentes casos de uso a lo largo de cualquier ciclo de vida de desarrollo de software seguro.

Lo más probable es que nunca haya un acuerdo del 100% sobre los contenidos de cualquier estándar de aplicación web, incluido el ASVS. El análisis de riesgo es siempre subjetivo hasta cierto punto, lo que crea un desafío cuando se intenta generalizar en un estándar único para todos. Sin embargo, esperamos que las últimas actualizaciones realizadas en esta versión sean un paso en la dirección correcta y mejoren los conceptos introducidos en este estándar crítico de la industria.

Novedades en la Versión 4.0

El cambio más significativo en esta versión es la adopción de las Pautas de identidad digital NIST 800-63-3, que introducen controles de autenticación avanzados, modernos y basados en evidencia. Aunque esperamos cierto retroceso en la alineación con un estándar de autenticación avanzado, creemos que es esencial que los estándares estén alineados, principalmente cuando otro estándar de seguridad de aplicaciones bien considerado se basa en evidencia.

Los estándares de seguridad de la información deben tratar de minimizar el número de requisitos únicos, de modo que las organizaciones que cumplen los requisitos no tengan que decidir sobre controles similares o incompatibles. El OWASP Top 10 2017 y ahora el Estándar De Verificación De Seguridad De Aplicaciones, se han alineado con NIST 800-63 para la autenticación y administración de sesiones. Alentamos a otros organismos que establecen estándares a trabajar con nosotros, NIST y otros para llegar a un conjunto generalmente aceptado de controles de seguridad de aplicaciones para maximizar la seguridad y minimizar los costos de cumplimiento.

ASVS 4.0 ha sido completamente renumerado de principio a fin. El nuevo esquema de numeración nos permitió eliminar brechas de capítulos desaparecidos hace mucho tiempo y permitirnos segmentar capítulos más largos para minimizar la cantidad de controles que un desarrollador o equipo debe cumplir. Por ejemplo, si una aplicación no utiliza JWT, la sección completa sobre JWT en la gestión de sesiones no es aplicable.

Lo nuevo en la versión 4.0 es un mapeo completo hacia la enumeración de debilidades comunes (CWE; por sus siglas en inglés), una de las solicitudes de mejora más pedida durante la última década. El mapeo CWE permite a los fabricantes de herramientas y aquellos que usan software de administración de vulnerabilidades comparar los resultados de otras herramientas y versiones anteriores de ASVS con 4.0 y posteriores. Para dejar espacio para la entrada CWE, hemos tenido que retirar la columna "Desde", que ahora que cambiamos la numeración por completo, tiene menos sentido que en versiones anteriores de ASVS. No todos los elementos del ASVS tienen un CWE asociado y, dado que CWE tiene una gran cantidad de duplicaciones, hemos intentado asociar el más utilizado en lugar de necesariamente el más cercano. Los controles de verificación no siempre se pueden asignar a debilidades equivalentes. Agradecemos la discusión en curso con la comunidad de CWE y el campo de la seguridad de la información en general sobre cómo cerrar esta brecha.

Hemos trabajado para cumplir y superar de manera integral los requisitos del OWASP Top 10 2017 y del OWASP Proactive Controls 2018. Dado que el OWASP Top 10 2017 es el mínimo indispensable para evitar negligencias, deliberadamente hemos hecho todo excepto los controles nivel 1 de los requisitos del Top 10, lo que facilita a los adoptadores del OWASP Top 10 avanzar a un estándar de seguridad real.

Quisimos asegurarnos de que el ASVS 4.0 Nivel 1 sea un superconjunto completo de PCI DSS 3.2.1 Secciones 6.5, para el diseño de aplicaciones, codificación, pruebas, revisiones de código seguro y pruebas de penetración. Esto requería cubrir el desbordamiento del búfer y las operaciones de memoria inseguras en V5, y los indicadores de compilación relacionados con la memoria insegura en V14, además de los requisitos de verificación de servicios web y aplicaciones líderes en la industria.

Cambiamos los controles ASVS monolíticos y solo del lado del servidor, a los nuevos controles de seguridad para todas las aplicaciones y API modernas. En los días de la programación funcional, serverless API, dispositivos móviles, nube, contenedores, CI/CD y DevSecOps, federación y más, no podemos seguir ignorando la arquitectura de aplicaciones moderna. Las aplicaciones modernas están diseñadas de manera muy diferente a las creadas cuando se lanzó el ASVS original en 2009. El ASVS siempre debe mirar hacia el futuro para poder brindar buenos consejos a nuestra audiencia principal: los desarrolladores. Hemos aclarado o eliminado cualquier requisito que suponga que las aplicaciones se ejecutan en sistemas propiedad de una sola organización.

Debido al tamaño del ASVS 4.0, así como a nuestro deseo de convertirlo en la línea base, para todos los demás esfuerzos de ASVS, hemos retirado el capítulo de "Móviles" en apoyo al Estándar de Verificación de Seguridad de Aplicaciones Móviles (MASVS; por sus siglas en inglés). El anexo de IoT (Internet de las cosas) se sumará a un futuro “IoT ASVS” dentro del Proyecto de Internet de las Cosas de OWASP. Hemos incluido una vista previa de IoT ASVS en el Apéndice C. Agradecemos tanto al Equipo de OWASP Mobile, como al Equipo de Proyecto de IoT de OWASP por su apoyo al ASVS, y esperamos trabajar con ellos a futuro para proporcionar estándares complementarios.

Por último, hemos eliminado y retirado controles menos impactantes. Con el tiempo, ASVS comenzó a ser un conjunto completo de controles, pero no todos los controles son igual de importantes para producir software seguro. Este esfuerzo por eliminar los elementos de bajo impacto podría ir más allá. En una futura edición del ASVS, el Sistema de Puntuación de Debilidad Común (CWSS; por sus siglas en inglés) ayudará a priorizar aún más los controles que son realmente importantes y los que deberían retirarse.

A partir de la versión 4.0, ASVS se centrará únicamente en ser el estándar de servicios y aplicaciones web líder, cubriendo la arquitectura de aplicaciones tradicional y moderna, y las prácticas de seguridad ágiles y la cultura DevSecOps.

Usando ASVS

ASVS tiene dos objetivos principales:

- Ayudar a las organizaciones a desarrollar y mantener aplicaciones seguras.
- Permitir que los proveedores de servicios de seguridad, los proveedores de herramientas de seguridad y los consumidores alineen sus requisitos y ofertas.

Niveles de ASVS

El estándar de verificación de seguridad de aplicaciones define tres niveles de verificación de seguridad, con cada nivel aumentando en profundidad.

- ASVS Nivel 1 es para bajos niveles de garantía, y es completamente comprobable con pentesting.
- ASVS Nivel 2 es para aplicaciones que contienen datos confidenciales, que requiere protección y es el nivel recomendado para la mayoría de las aplicaciones.
- ASVS Nivel 3 es para las aplicaciones más críticas - aplicaciones que realizan transacciones de alto valor, contienen datos médicos sensibles, o cualquier aplicación que requiere el más alto nivel de confianza.

Cada nivel ASVS contiene una lista de requisitos de seguridad. Cada uno de estos requisitos también se puede asignar a características y capacidades específicas de seguridad que los desarrolladores deben integrar en el software.

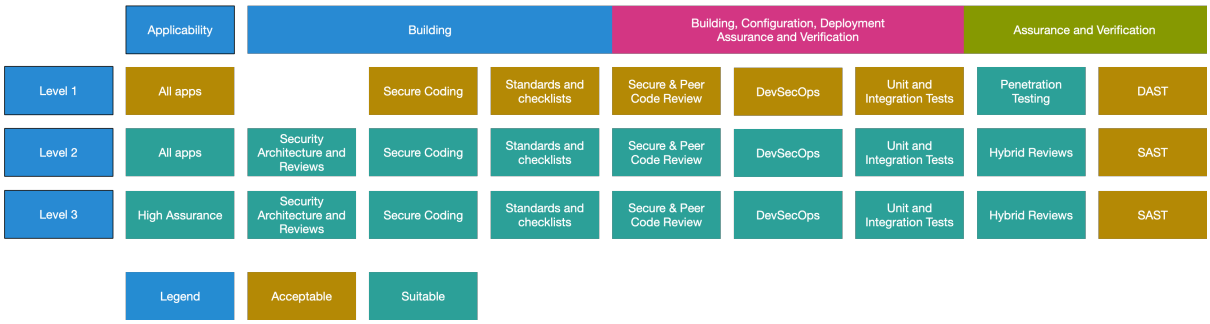

Figura 1 - Niveles de OWASP ASVS 4.0

El nivel 1 (L1) es el único nivel que es completamente comprobable de penetración usando humanos. Todos los demás requieren acceso a la documentación, el código fuente, la configuración y las personas involucradas en el proceso de desarrollo. Sin embargo, incluso si L1 permite que se realicen pruebas de "caja negra" (sin documentación ni fuente), no es una actividad de garantía eficaz y debe desalentarse activamente. Los atacantes malintencionados tienen mucho tiempo, la mayoría de las pruebas de penetración han terminado en un par de semanas. Los defensores necesitan crear controles de seguridad, proteger, encontrar y resolver todas las debilidades, y detectar y responder a actores malintencionados en un tiempo razonable. Los actores malintencionados tienen tiempo esencialmente infinito y sólo necesitan una defensa porosa, una sola debilidad, o falta de detección para tener éxito. Las pruebas de caja negra, que a menudo se realizan al final del desarrollo, se realizan rápidamente, y son completamente incapaces de hacer frente a esa asimetría.

Durante los últimos 30+ años, las pruebas de cajas negras han demostrado una y otra vez perderse problemas críticos de seguridad que llevaron directamente a brechas cada vez más masivas. Recomendamos encarecidamente el uso de una amplia gama de garantías y verificación de seguridad, incluida la sustitución de pruebas de penetración por pruebas de penetración dirigidas por código fuente (híbrido) en el nivel 1, con acceso completo a los desarrolladores y documentación durante todo el proceso de desarrollo. Los reguladores financieros no toleran auditorías financieras externas sin acceso a los libros, transacciones de muestra o a las personas que realizan los controles. La industria y los gobiernos deben exigir el mismo estándar de transparencia en el campo de la ingeniería de software.

Recomendamos encarecidamente el uso de herramientas de seguridad dentro del proceso de desarrollo. Las herramientas DAST y SAST se pueden utilizar continuamente por el canal de construcción para detectar problemas de seguridad fáciles de encontrar que nunca deben estar presentes.

Las herramientas automatizadas y los escaneos en línea sólo pueden completar casi la mitad del ASVS sin asistencia humana. Si se requiere una automatización de pruebas completa para cada compilación, se usa una combinación de pruebas de unidad personalizadas e integración, junto con escaneos en línea iniciados por la compilación. Los defectos de lógica empresarial y las pruebas de control de acceso solo son posibles mediante la asistencia humana. Estos deben convertirse en pruebas unitarias y de integración.

Cómo usar este estándar

Una de las mejores maneras de utilizar el estándar de verificación de seguridad en aplicaciones es usarlo como un plano-guía para crear una lista de comprobación de codificación segura específica para su aplicación, plataforma u organización. Adaptar el ASVS a sus casos de uso aumentará el enfoque en los requisitos de seguridad que son más importantes para sus proyectos y entornos.

Nivel 1 (L1) - Primeros pasos, vista automatizada o completa de la cartera

Una aplicación alcanza ASVS Nivel 1 si logra defenderse contra vulnerabilidades de seguridad de aplicaciones que son fáciles de descubrir, e incluido el Top 10 de OWASP y otras listas de comprobación similares.

El nivel 1 es el mínimo por el que todas las aplicaciones deben esforzarse. También es útil como primer paso en un esfuerzo multifáses o cuando las aplicaciones no almacenan ni manejan datos confidenciales y, por lo tanto, no necesitan los controles más rigurosos de Nivel 2 o 3. Los controles de nivel 1 se pueden comprobar automáticamente mediante herramientas o simplemente manualmente sin acceso al código fuente. Consideramos el Nivel 1 el mínimo requerido para todas las aplicaciones.

Las amenazas a la aplicación probablemente serán de atacantes que utilizan técnicas simples y de bajo esfuerzo para identificar vulnerabilidades fáciles de encontrar y fáciles de explotar. Esto contrasta con un atacante determinado que gastará energía enfocada para apuntar específicamente a la aplicación. Si los datos procesados por su aplicación tienen un alto valor, rara vez querrá detenerse en una revisión de Nivel 1.

Nivel 2 (L2) - Para la mayoría de las aplicaciones

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

El nivel 2 garantiza que los controles de seguridad estén en su lugar, sean eficaces y se utilicen dentro de la aplicación. El nivel 2 suele ser adecuado para aplicaciones que manejan importantes transacciones de negocio-a-negocio, incluidas aquellas que procesan información de atención médica, y/o implementan funciones críticas para el negocio, o procesan otros activos sensibles, o industrias donde la integridad es una faceta crítica para proteger su negocio, como la industria de juegos para frustrar a los tramposos y game hacks.

Las amenazas a las aplicaciones de nivel 2 suelen ser atacantes calificados y motivados que se centran en objetivos específicos utilizando herramientas y técnicas, que son altamente practicadas y eficaces para descubrir y explotar las debilidades dentro de las aplicaciones.

Nivel 3 (L3) - Alto valor, alta garantía o alta seguridad

ASVS Nivel 3 es el nivel más alto de verificación dentro del ASVS. Este nivel se reserva normalmente para aplicaciones que requieren niveles significativos de verificación de seguridad, como los que se pueden encontrar dentro de áreas de militar, salud y seguridad, infraestructura crítica, etc.

Las organizaciones pueden requerir ASVS Nivel 3 para aplicaciones que realizan funciones críticas, donde el error podría afectar significativamente las operaciones de la organización, e incluso su supervivencia. A continuación se proporcionan instrucciones de ejemplo sobre la aplicación del nivel 3 de ASVS. Una aplicación alcanza ASVS Nivel 3 (o Avanzado) si se defiende adecuadamente contra vulnerabilidades avanzadas de seguridad de aplicaciones y también demuestra principios de buen diseño de seguridad.

Una aplicación en ASVS Nivel 3 requiere un análisis más detallado de la arquitectura, la codificación y las pruebas que todos los demás niveles. Una aplicación segura se modulariza de una manera significativa (para facilitar la resiliencia, la escalabilidad y, sobre todo, las capas de seguridad), y cada módulo (separado por conexión de red y/o instancia física) se encarga de sus propias responsabilidades de seguridad (defensa en profundidad), que deben documentarse correctamente. Las responsabilidades incluyen controles para garantizar la confidencialidad (por ejemplo, cifrado), integridad (por ejemplo, transacciones, validación de

entradas), disponibilidad (por ejemplo, manejo correcto de la carga), autenticación (incluidos entre sistemas), autorización y auditoría (registros de log).

Aplicando ASVS en la Práctica

Diferentes amenazas tienen diferentes motivaciones. Algunas industrias tienen activos únicos de información y tecnología y requisitos de cumplimiento normativo específicos del dominio.

Se recomienda encarecidamente a las organizaciones que examinen profundamente sus características de riesgo únicas en función de la naturaleza de su negocio, y sobre la base de que los requisitos de riesgo y de negocio determinan el nivel adecuado de ASVS.

Cómo Hacer Referencia a los Requisitos de ASVS

Cada requisito tiene un identificador en el formato <chapter>.<section>.<requirement> donde cada elemento es un número, por ejemplo: 1.11.3.

- El elemento <chapter> corresponde al capítulo del que proviene el requisito, por ejemplo: todos los requisitos de 1.#.# son del capítulo de Arquitectura.
- El elemento <section> corresponde a la sección dentro de ese capítulo donde aparece el requisito, por ejemplo: todos los requisitos de 1.11.# están en la sección Arquitectura de la Lógica del Negocio, del capítulo de Arquitectura.
- El elemento <requirement> identifica el requisito específico dentro del capítulo y la sección, por ejemplo: 1.11.3 que a partir de la versión 4.0.3 del presente estándar es:

Compruebe que todos los flujos de lógica de negocio de alto valor, incluida la autenticación, la administración de sesiones y el control de acceso, sean seguros para subprocesos y resistentes a las condiciones de tiempo de comprobación y tiempo de uso.

Los identificadores pueden cambiar entre las versiones de la norma, por lo que es preferible que otros documentos, informes o herramientas utilicen el formato: v<version>-<chapter>.<section>.<requirement>, donde: 'versión' es la etiqueta de la versión ASVS. Por ejemplo: v4.0.3-1.11.3 se entendería que significa específicamente el 3er requisito en la sección "Arquitectura de la Lógica del Negocio" del capítulo "Arquitectura" de la versión 4.0.3. (Esto podría resumirse como v<version>-<requirement_identifier>)

Nota: La v que precede a la parte de la versión debe estar en minúsculas.

Si se utilizan identificadores sin incluir el v<version>, se debe suponer que hacen referencia al contenido más reciente del estándar de verificación de seguridad en aplicaciones. Obviamente, a medida que el estándar crece y cambia esto se vuelve problemático, por lo que los escritores o desarrolladores deben incluir el elemento de versión.

Las listas de requisitos de ASVS están disponibles en CSV, JSON y otros formatos que pueden ser útiles para el uso de referencia o mediante programación.

Evaluación y Certificación

Postura de OWASP sobre Certificaciones ASVS y Marcas de Confianza

OWASP, como una organización neutral para el proveedor sin fines de lucro, actualmente no certifica ningún proveedor, verificador o software.

Todas esas afirmaciones de garantía, marcas de confianza o certificaciones no son examinadas, registradas o certificadas oficialmente por OWASP, por lo que una organización que se base en esa opinión debe tener cuidado con la confianza depositada en cualquier tercero o marca de confianza que reclame la certificación ASVS.

Esto no debe impedir que las organizaciones ofrezcan tales servicios de garantía, siempre y cuando no reclamen la certificación oficial de OWASP.

Orientación para las Organizaciones Certificadoras

ASVS se puede utilizar como una verificación de libro abierto de la aplicación, incluido el acceso abierto y sin restricciones a recursos clave como arquitectos, desarrolladores, documentación del proyecto, código fuente, acceso autenticado a sistemas de prueba (incluido el acceso a una o más cuentas en cada rol), especialmente para verificaciones de nivel L2 y L3.

Históricamente, las pruebas de penetración y las revisiones de código seguro han incluido cuestiones "por excepción", es decir, las pruebas fallidas aparecen en el informe final. Una organización certificadora debe incluir en cualquier informe el alcance de la verificación (especialmente si un componente clave está fuera del ámbito, como la autenticación SSO), un resumen de los resultados de la verificación, incluidas las pruebas pasadas y erróneas, con indicaciones claras de cómo resolver las pruebas con errores.

Ciertos requisitos de verificación pueden no ser aplicables al software bajo prueba. Por ejemplo, si proporciona una API de capa de servicio stateless sin una implementación de cliente a sus consumidores, muchos de los requisitos de V3 Administración de Sesiones no son directamente aplicables. En tales casos, una organización certificadora todavía puede reclamar el cumplimiento completo de ASVS, pero debe indicar claramente en cualquier informe una razón para la inaplicabilidad de dichos requisitos de verificación excluidos.

Mantener documentos de trabajo detallados, capturas de pantalla o películas, guiones para explotar de forma fiable y repetidamente un problema, y registros electrónicos de pruebas, como interceptar registros de proxy y notas asociadas, como una lista de limpieza, se considera práctica estándar de la industria y puede ser realmente útil como pruebas de los hallazgos para los desarrolladores más dudosos. No basta con simplemente ejecutar una herramienta e informar sobre los errores; esto no proporciona (en absoluto) pruebas suficientes de que todas las cuestiones a nivel de certificación han sido probadas y probadas a fondo. En caso de controversia, debe haber pruebas de garantía suficientes para demostrar que todos y cada uno de los requisitos verificados han sido probados.

Método de Pruebas

Las organizaciones de certificación son libres de elegir los métodos de prueba adecuados, pero deben indicarlas en un informe.

Dependiendo de la aplicación sometida a prueba y del requisito de verificación, se pueden utilizar diferentes métodos de prueba para obtener una confianza similar en los resultados. Por ejemplo, la validación de la eficacia de los mecanismos de verificación de entrada de una aplicación puede analizarse con una prueba de penetración manual o mediante análisis de código fuente.

El Rol de las Herramientas Automatizadas de Pruebas de Seguridad

Se recomienda el uso de herramientas de pruebas de penetración automatizadas para proporcionar la mayor cobertura posible.

No es posible completar en su totalidad la verificación de ASVS utilizando solo herramientas de pruebas de penetración automatizadas. Si bien una gran mayoría de los requisitos en N1 se pueden realizar mediante

pruebas automatizadas, la mayoría general de los requisitos no son susceptibles a las pruebas de penetración automatizadas.

Tenga en cuenta que las líneas entre las pruebas automatizadas y manuales se han difuminado a medida que la industria de seguridad de aplicaciones madura. Las herramientas automatizadas a menudo son ajustadas manualmente por expertos y los probadores manuales a menudo aprovechan una amplia variedad de herramientas automatizadas.

El rol de las Pruebas de Penetración

En la versión 4.0, decidimos hacer L1 completamente verificable con pruebas de penetración sin acceso al código fuente, documentación o desarrolladores. Dos elementos de logging, que están obligados a cumplir con OWASP Top 10 2017 A10, requerirán entrevistas, capturas de pantalla u otra captura de evidencia, al igual que lo hacen en el Top 10 2017 de OWASP. Sin embargo, las pruebas sin acceso a la información necesaria no son un método ideal de verificación de seguridad, ya que pierde la posibilidad de revisar el origen, identificar las amenazas y los controles que faltan, y realizar una prueba mucho más exhaustiva en un plazo más corto.

Siempre que sea posible, se requiere acceso a los desarrolladores, documentación, código y acceso a una aplicación de prueba con datos que no sean de producción al realizar una evaluación L2 o L3. Las pruebas de penetración realizadas a estos niveles requieren este nivel de acceso, que llamamos "revisiones híbridas" o "pruebas de penetración híbrida".

Otros usos para ASVS

Además de utilizarse para evaluar la seguridad de una aplicación, hemos identificado una serie de otros usos potenciales para ASVS.

Como una Guía Detallada de la Arquitectura de Seguridad

Uno de los usos más comunes para el estándar de verificación de seguridad en aplicaciones es como un recurso para los arquitectos de seguridad. A la arquitectura de seguridad empresarial aplicada de Sherwood (SABSA) le falta una gran cantidad de información que es necesaria para completar una revisión exhaustiva de la arquitectura de seguridad en aplicaciones. ASVS se puede utilizar para llenar esas lagunas al permitir que los arquitectos de seguridad elijan mejores controles para problemas comunes, como patrones de protección de datos y estrategias de validación de entradas.

Como un Reemplazo de Listas de Verificación de Codificación Segura Listas para Usar

Muchas organizaciones pueden beneficiarse de la adopción de ASVS, eligiendo uno de los tres niveles, o bifurcando ASVS y cambiando lo que se requiere para cada nivel de riesgo de aplicación de una manera específica del dominio. Animamos a este tipo de bifurcación siempre y cuando se mantenga la trazabilidad, de modo que si una aplicación ha pasado el requisito 4.1, esto significa lo mismo para las copias bifurcadas que el estándar a medida que evoluciona.

Como una Guía para Pruebas Automatizadas de Unidad e Integración

El ASVS está diseñado para ser altamente comprobable, con la única excepción de los requisitos de código arquitectónico y malicioso. Mediante la creación de pruebas de unidad e integración que prueban casos de fuzz y abuso específicos y relevantes, la aplicación se vuelve casi autocomprobación con todas y cada una de las compilaciones. Por ejemplo, se pueden crear pruebas adicionales para el conjunto de pruebas para un controlador de inicio de sesión, probando el parámetro username para los nombres de usuario predeterminados comunes, la enumeración de la cuenta, el forzamiento bruto, la inyección LDAP y SQL, y XSS. Del mismo modo, una prueba en el parámetro de contraseña debe incluir contraseñas comunes, longitud de contraseña, inyección de bytes nulos, eliminación del parámetro, XSS y más.

Para la Formación en Desarrollo Seguro

ASVS también se puede utilizar para definir las características del software seguro. Muchos cursos de "codificación segura" son simplemente cursos de hacking ético con un ligero barniz de tips de codificación. Esto no necesariamente ayuda a los desarrolladores a escribir código más seguro. En su lugar, los cursos de desarrollo seguro pueden utilizar el ASVS con un fuerte enfoque en los controles proactivos que provee, en lugar de las 10 cosas negativas que no se deben hacer.

Como un Conductor para la Seguridad de Aplicaciones Ágiles

ASVS se puede utilizar en un proceso de desarrollo ágil como framework para definir tareas específicas que el equipo debe implementar para tener un producto seguro. Un enfoque podría ser: A partir del nivel 1, compruebe la aplicación o el sistema específico según los requisitos de ASVS para el nivel especificado, busque qué controles faltan y genere tickets/tareas específicas en el "backlog". Esto ayuda a priorizar tareas específicas (o refinamiento) y hace que la seguridad sea visible en el proceso ágil. Esto también se puede utilizar para priorizar las tareas de auditoría y revisión en la organización, donde un requisito específico de ASVS puede ser un impulsor para la revisión, refactorización o auditoría para un miembro específico del equipo y visible como "deuda" en el backlog que debe ser hecho eventualmente.

Como Marco de Trabajo para Orientar la Adquisición de Software Seguro

ASVS es un gran marco para ayudar con la adquisición segura de software o la adquisición de servicios de desarrollo personalizados. El comprador puede simplemente establecer un requisito de que el software que desea adquirir debe desarrollarse en el nivel X de ASVS, y solicitar que el vendedor demuestre que el software cumple con el nivel X de ASVS. Esto funciona bien cuando se combina con el anexo OWASP del contrato de software seguro.

V1 Arquitectura, Diseño y Modelado de Amenazas

Objetivo de Control

La arquitectura de seguridad casi se ha convertido en un arte perdido en muchas organizaciones. Los días del arquitecto empresarial han pasado en la era de DevSecOps. El campo de la seguridad de las aplicaciones debe ponerse al día y adoptar principios de seguridad ágiles, al tiempo que reintroduce los principios de arquitectura de seguridad líderes a los profesionales del software. La arquitectura no es una implementación, sino una forma de pensar sobre un problema que tiene potencialmente muchas respuestas diferentes, y no solo una respuesta "correcta". Con demasiada frecuencia, la seguridad se ve como inflexible y exigente que los desarrolladores corrijan el código de una manera particular, cuando los desarrolladores pueden conocer una manera mucho mejor de resolver el problema. No hay una solución única y sencilla para la arquitectura, y pretender lo contrario es un flaco favor al campo de la ingeniería de software.

Es probable que una implementación específica de una aplicación web se revise continuamente a lo largo de su vida útil, pero es probable que la arquitectura general rara vez cambie, pero evoluciona lentamente. La arquitectura de seguridad es idéntica: necesitamos autenticación hoy, necesitaremos autenticación mañana y la necesitaremos dentro de cinco años. Si tomamos decisiones acertadas hoy en día, podemos ahorrar mucho esfuerzo, tiempo y dinero si seleccionamos y reutilizamos soluciones que cumplen con la arquitectura. Por ejemplo, hace una década, la autenticación multifactor rara vez se implementaba.

Si los desarrolladores hubieran invertido en un único modelo de proveedor de identidad seguro, como la identidad federada SAML, el proveedor de identidades podría actualizarse para incorporar nuevos requisitos, como el cumplimiento de NIST 800-63, sin cambiar las interfaces de la aplicación original. Si muchas aplicaciones compartían la misma arquitectura de seguridad y, por lo tanto, ese mismo componente, todas se benefician de esta actualización a la vez. Sin embargo, SAML no siempre permanecerá como la mejor o más adecuada solución de autenticación: es posible que deba intercambiarse para otras soluciones a medida que cambian los requisitos. Cambios como este son complicados y costosos como para requerir una reescritura completa, o totalmente imposible sin arquitectura de seguridad.

En este capítulo, el ASVS cubre los aspectos principales de cualquier arquitectura de seguridad sólida: disponibilidad, confidencialidad, integridad del procesamiento, no repudio y privacidad. Cada uno de estos principios de seguridad debe estar integrado y ser innato para todas las aplicaciones. Es fundamental "desplazar a la izquierda", comenzando con la habilitación del desarrollador con listas de verificación de codificación seguras, tutoría y capacitación, codificación y pruebas, creación, implementación, configuración y operaciones, y terminando con pruebas independientes de seguimiento para asegurar que todos los controles de seguridad están presentes y funcionales. El último paso solía ser todo lo que hacíamos como industria, pero ya no es suficiente cuando los desarrolladores insertan código en producción decenas o cientos de veces al día. Los profesionales de la seguridad de las aplicaciones deben mantenerse al día con técnicas ágiles, lo que significa adoptar herramientas de desarrollo, aprender a codificar, y trabajar con desarrolladores en lugar de criticar el proyecto meses después de que todos los demás siguieron adelante.

V1.1 Ciclo de Vida de Desarrollo de Software Seguro

#	Descripción	L1	L2	L3	CWE
1.1.1	Verifique el uso de un ciclo de vida de desarrollo de software seguro que aborde la seguridad en todas las etapas del desarrollo. (C1)		✓	✓	
1.1.2	Verifique el uso del modelado de amenazas para cada cambio de diseño o planificación de sprint para identificar amenazas, planificar contramedidas, facilitar respuestas de riesgo adecuadas y guiar las pruebas de seguridad.		✓	✓	1053
1.1.3	Verifique que todas las historias y características de usuario contienen restricciones de seguridad funcionales, como por ejemplo: "Como usuario, debería poder ver y editar mi perfil. No debería ser capaz de ver o editar el perfil de nadie más"		✓	✓	1110

#	Descripción	L1	L2	L3	CWE
1.1.4	Verifique la documentación y la justificación de todos los límites de confianza, componentes y flujos de datos significativos de la aplicación.		✓	✓	1059
1.1.5	Verifique la definición y el análisis de seguridad de la arquitectura de alto nivel de la aplicación y todos los servicios remotos conectados. (C1)		✓	✓	1059
1.1.6	Verifique la implementación de controles de seguridad centralizados, simples (economía del diseño), comprobados, seguros y reutilizables para evitar controles duplicados, faltantes, ineficaces o inseguros. (C10)		✓	✓	637
1.1.7	Verifique la disponibilidad de una lista de comprobación de codificación segura, requisitos de seguridad, directriz o directiva para todos los desarrolladores y evaluadores.		✓	✓	637

V1.2 Arquitectura de Autenticación

Al diseñar la autenticación, no importa si tiene una autenticación multifactor habilitada para hardware fuerte si un atacante puede restablecer una cuenta llamando a un centro de llamadas y respondiendo a preguntas conocidas. Al probar la identidad, todas las vías de autenticación deben tener la misma fuerza.

#	Descripción	L1	L2	L3	CWE
1.2.1	Verifique el uso de cuentas de sistema operativo únicas o especiales con privilegios bajos para todos los componentes, servicios y servidores de la aplicación. (C3)		✓	✓	250
1.2.2	Verifique que las comunicaciones entre los componentes de la aplicación, incluidas las API, el middleware y las capas de datos, se autentican. Los componentes deben tener los mínimos privilegios necesarios. (C3)		✓	✓	306
1.2.3	Verifique que la aplicación utiliza un único mecanismo de autenticación comprobado que se sabe que es seguro, se puede ampliar para incluir una autenticación segura y tiene suficiente logging y supervisión para detectar abuso de cuenta o brechas.		✓	✓	306
1.2.4	Verifique que todas las vías de autenticación y las API de administración de identidades implementan una fortaleza coherente del control de seguridad de autenticación, de modo que no haya alternativas más débiles por el riesgo de la aplicación.		✓	✓	306

V1.3 Arquitectura de Gestión de Sesiones

Este es un marcador de posición para los requisitos arquitectónicos futuros.

V1.4 Arquitectura de Control de Acceso

#	Descripción	L1	L2	L3	CWE
1.4.1	Verifique que los puntos de cumplimiento de confianza, tales como puertas de enlace de control de acceso, servidores y funciones serverless, exijan controles de acceso. Nunca aplique controles de acceso en el cliente.		✓	✓	602
1.4.2	[ELIMINADO, NO ACCIONABLE]				
1.4.3	[ELIMINADO, DUPLICADO CON 4.1.3]				

#	Descripción	L1	L2	L3	CWE
1.4.4	Verifique que la aplicación utilice un mecanismo de control de acceso único y bien comprobado para acceder a datos y recursos protegidos. Todas las solicitudes deben pasar por este único mecanismo para evitar copiar y pegar o rutas alternativas inseguras. (C7)		✓	✓	284
1.4.5	Verifique que se utiliza el control de acceso basado en atributos o entidades mediante el cual el código comprueba la autorización del usuario para un elemento de característica o datos en lugar de solo su rol. Los permisos deben asignarse mediante roles. (C7)		✓	✓	275

V1.5 Arquitectura de Entradas y Salidas

En 4.0, nos hemos alejado del término "server-side" como un término límite de confianza cargado. El límite de confianza sigue siendo relativo a - tomar decisiones sobre navegadores que no son de confianza o dispositivos cliente que son "bypassable". Sin embargo, en las implementaciones arquitectónicas convencionales de hoy en día, el punto de aplicación de la confianza ha cambiado drásticamente. Por lo tanto, cuando el término "trusted service layer" se utiliza en el ASVS, nos referimos a cualquier punto de aplicación de confianza, independientemente de la ubicación, como un microservicio, serverless API, server-side, una API de confianza en un dispositivo cliente que tiene secure boot, partner o API externas, etc.

El término "untrusted client" aquí se refiere a las tecnologías del lado del cliente que representan la capa de presentación, comúnmente referida como tecnologías 'front-end'. El término "serialization" aquí no sólo se refiere a enviar datos a través de las comunicaciones, como una matriz de valores o tomar y leer una estructura JSON, sino también pasar objetos complejos que pueden contener lógica.

#	Descripción	L1	L2	L3	CWE
1.5.1	Verifique que los requisitos de entrada y salida definan claramente cómo manejar y procesar datos en función del tipo, contenido y las leyes, regulaciones y otras leyes aplicables, reglamentos y otras normas de cumplimiento de políticas.		✓	✓	1029
1.5.2	Verifique que no se usa serialización al comunicarse con clientes que no son de confianza. Si esto no es posible, asegúrese de que se apliquen controles de integridad adecuados (y posiblemente cifrado si se envían datos confidenciales) para evitar ataques de deserialización, incluida la inyección de objetos.		✓	✓	502
1.5.3	Verifique que la validación de datos de entrada (input) se aplica en una capa de servicio de confianza. (C5)		✓	✓	602
1.5.4	Verifique que la codificación de salida (output encode) se produce cerca o en el intérprete para el que está destinada. (C4)		✓	✓	116

V1.6 Arquitectura Criptográfica

Las aplicaciones deben diseñarse con una arquitectura criptográfica sólida para proteger los activos de datos según su clasificación. Cifrar todo es un desperdicio, no cifrar nada es una negligencia legal. Se debe lograr un equilibrio, generalmente durante el diseño arquitectónico o de alto nivel, los "sprints" de diseño o los "spikes" arquitectónicos. Diseñar criptografía sobre la marcha o modernizarla inevitablemente costará mucho más implementar de forma segura que simplemente construirla desde el principio.

Los requisitos arquitectónicos son intrínsecos a toda la base de código y, por lo tanto, son difíciles de unir o integrar la prueba. Los requisitos arquitectónicos requieren consideración en los estándares de codificación, a lo largo de la fase de codificación, y deben revisarse durante la arquitectura de seguridad, las revisiones de pares o código, o las retrospectivas.

#	Descripción	L1	L2	L3	CWE
1.6.1	Verifique que existe una política explícita para la administración de claves criptográficas y que un ciclo de vida de clave criptográfica sigue un estándar de administración de claves como NIST SP 800-57.		✓	✓	320
1.6.2	Verifique que los consumidores de servicios criptográficos protegen el material clave y otros secretos mediante el uso de almacenes de claves o alternativas basadas en API.		✓	✓	320
1.6.3	Verifique que todas las claves y contraseñas son reemplazables y forman parte de un proceso bien definido para volver a cifrar los datos confidenciales.		✓	✓	320
1.6.4	Verifique que la arquitectura trata los secretos del lado cliente (como claves simétricas, contraseñas o tokens de API) como inseguros y nunca los usa para proteger o acceder a datos confidenciales.		✓	✓	320

V1.7 Arquitectura de Errores, Logging y Auditoría

#	Descripción	L1	L2	L3	CWE
1.7.1	Verifique que se utilice un formato común y un enfoque de logging en todo el sistema. (C9)		✓	✓	1009
1.7.2	Verifique que los registros de log se transmitan de forma segura a un sistema preferentemente remoto para análisis, detección, alertas y escalamiento. (C9)		✓	✓	

V1.8 Arquitectura de Protección de Datos y Privacidad

#	Descripción	L1	L2	L3	CWE
1.8.1	Verifique que todos los datos confidenciales se identifiquen y clasifiquen en niveles de protección.		✓	✓	
1.8.2	Verifique que todos los niveles de protección tienen un conjunto asociado de requisitos de protección, como los requisitos de cifrado, los requisitos de integridad, la retención, la privacidad y otros requisitos de confidencialidad, y que estos se aplican en la arquitectura.		✓	✓	

V1.9 Arquitectura de Comunicaciones

#	Descripción	L1	L2	L3	CWE
1.9.1	Verifique que la aplicación cifra las comunicaciones entre componentes, especialmente cuando estos componentes se encuentran en contenedores, sistemas, sitios o proveedores de nube diferentes. (C3)		✓	✓	319
1.9.2	Verifique que los componentes de la aplicación verifiquen la autenticidad de cada lado en un vínculo de comunicación para evitar ataques de "persona en el medio". Por ejemplo, los componentes de la aplicación deben validar certificados y cadenas TLS.		✓	✓	295

V1.10 Arquitectura de Software Malicioso

#	Descripción	L1	L2	L3	CWE
1.10.1	Verifique que un sistema de control de código fuente está en uso, con procedimientos para garantizar que los check-ins están respaldados tickets de issues o solicitudes de cambio. El sistema de control de código fuente debe tener control de acceso y usuarios identificables para permitir la trazabilidad de cualquier cambio.		✓	✓	284

V1.11 Arquitectura de la Lógica de Negocio

#	Descripción	L1	L2	L3	CWE
1.11.1	Verifique la definición y documentación de todos los componentes de la aplicación en términos de las funciones de negocio o de seguridad que proporcionan.		✓	✓	1059
1.11.2	Verifique que todos los flujos de lógica de negocio de alto valor, incluida la autenticación, la administración de sesiones y el control de acceso, no compartan estados no sincronizados.		✓	✓	362
1.11.3	Verifique que todos los flujos de lógica de negocio de alto valor, incluida la autenticación, la administración de sesiones y el control de acceso, sean seguros para subprocesos y resistentes a condiciones de carrera time-of-check y time-of-use (race conditions).			✓	367

V1.12 Arquitectura de Carga Segura de Archivos

#	Descripción	L1	L2	L3	CWE
1.12.1	[ELIMINADO, DUPLICADO CON 12.4.1]				
1.12.2	Verifique que los archivos subidos por el usuario, -si es necesario que se muestren o descarguen desde la aplicación-, se hace mediante descargas de secuencias de octetos o desde un dominio no relacionado, como un almacenamiento de archivos en la nube. Implemente una directiva de seguridad de contenido (CSP) adecuada para reducir el riesgo de vectores XSS u otros ataques desde el archivo cargado.		✓	✓	646

V1.13 Arquitectura de API

Este es un marcador de posición para los requisitos arquitectónicos futuros.

V1.14 Arquitectura de Configuración

#	Descripción	L1	L2	L3	CWE
1.14.1	Verifique la segregación de componentes de diferentes niveles de confianza a través de controles de seguridad bien definidos, reglas de corta fuego, pasarelas de API, proxies reversos, grupos de seguridad basados en nube, o mecanismos similares.		✓	✓	923
1.14.2	Verifique que las firmas binarias, las conexiones de confianza y los puntos de conexión verificados se usan para el despliegue de archivos binarios a dispositivos remotos.		✓	✓	494

#	Descripción	L1	L2	L3	CWE
1.14.3	Verifique que el canal de compilación advierte de componentes obsoletos o inseguros y realiza las acciones adecuadas.		✓	✓	1104
1.14.4	Verifique que el canal de compilación contiene un paso para compilar y comprobar automáticamente el despliegue seguro de la aplicación, especialmente si la infraestructura de la aplicación está definida por software, como los scripts de compilación del entorno en la nube.		✓	✓	
1.14.5	Verifique que los despliegues de aplicaciones sean en sandbox, contenedores y/o aislados a nivel de red para retrasar e impedir que los atacantes vulneren otras aplicaciones, especialmente cuando realizan acciones sensibles o peligrosas, como la deserialización. (C5)		✓	✓	265
1.14.6	Verifique que la aplicación no utiliza tecnologías del lado cliente no compatibles, inseguras o en desuso, como NSAPI plugins, Flash, Shockwave, ActiveX, Silverlight, NACL o client-side java applets.		✓	✓	477

Referencias

Para obtener más información, consulte también:

- OWASP Threat Modeling Cheat Sheet
- OWASP Attack Surface Analysis Cheat Sheet
- OWASP Threat modeling
- OWASP Software Assurance Maturity Model Project
- Microsoft SDL
- NIST SP 800-57

V2 Autenticación

Objetivo de Control

La autenticación es el acto de establecer, o confirmar, a alguien (o algo) como auténtico y que las afirmaciones hechas por una persona o sobre un dispositivo son correctas, resistentes a la suplantación e impiden la recuperación o interceptación de contraseñas.

Cuando el ASVS fue lanzado por primera vez, "username + password" era la forma más común de autenticación fuera de los sistemas de alta seguridad. La autenticación multifactor (MFA) se aceptaba comúnmente en los círculos de seguridad, pero rara vez se requería en otros lugares. A medida que aumentaba el número de violación de contraseñas, la idea de que los nombres de usuario son de alguna manera confidenciales y las contraseñas desconocidas, hizo que muchos controles de seguridad fueran insostenibles. Por ejemplo, NIST 800-63 considera los nombres de usuario y la autenticación basada en conocimientos (KBA) como información pública, SMS y notificaciones por correo electrónico como "restricted" authenticator types, y contraseñas como pre-violadas. Esta realidad hace que los autenticadores basados en el conocimiento, la recuperación de SMS y correo electrónico, el historial de contraseñas, la complejidad y los controles de rotación sean inútiles. Estos controles siempre han sido menos que útiles, a menudo obligando a los usuarios a llegar a contraseñas débiles cada pocos meses, pero con el lanzamiento de más de 5 billones de filtraciones de "username + password", es el momento de seguir adelante.

De todas los capítulos del ASVS, los que más cambiaron son Autenticación y Administración de sesiones. La adopción de una práctica moderna, efectiva y basada en evidencia será un desafío para muchos, y eso está perfectamente bien. Tenemos que comenzar ahora la transición a un futuro post-contraseña.

NIST 800-63 - Estándar de autenticación moderno basado en evidencia

NIST 800-63b es un estándar moderno basado en la evidencia, y representa el mejor consejo disponible, independientemente de la aplicabilidad. El estándar es útil para todas las organizaciones de todo el mundo, pero es particularmente relevante para las agencias estadounidenses y para quienes trabajen con las agencias estadounidenses.

La terminología NIST 800-63 puede ser un poco confusa al principio, especialmente si solo estás acostumbrado a la autenticación de "username + password". Los avances en la autenticación moderna son necesarios, por lo que tenemos que introducir terminología que se convertirá en algo común en el futuro, pero entendemos la dificultad de entender hasta que la industria se asiente en estos nuevos términos. Se proporciona un glosario al final de este capítulo para mayor detalle. Hemos reformulado muchos requisitos para satisfacer la intención del requisito, en lugar de sólo su letra. Por ejemplo, el ASVS utiliza el término "contraseña" cuando NIST utiliza "secreto memorizado" en todo el estándar.

ASVS V2 Autenticación, V3 Administración de sesiones, y en menor medida, V4 Controles de acceso se han adaptado para que sean un subconjunto compatible de controles NIST 800-63b seleccionados, centrados en amenazas comunes y debilidades de autenticación comúnmente explotadas. Cuando se requiera el cumplimiento completo del NIST 800-63, consulte NIST 800-63.63.

Seleccionando un nivel adecuado de NIST AAL

ASVS ha intentado mapear ASVS L1 con los requisitos NIST AAL1, N2 a AAL2, y L3 a AAL3. Sin embargo, el enfoque de ASVS Level 1 como controles "esenciales" puede no ser necesariamente el nivel AAL correcto para verificar una aplicación o API. Por ejemplo, si la aplicación es una aplicación de nivel 3 o tiene requisitos reglamentarios para ser AAL3, el nivel 3 debe elegirse en los capítulos V2 y V3 Administración de sesiones. La elección del nivel de afirmación de autenticación (AAL; Por sus siglas en inglés) compatible con NIST se debe realizar según las pautas NIST 800-63b como se establece en *Seleccionando AAL* en NIST 800-63b Section 6.2.

Leyenda

Las aplicaciones siempre pueden exceder los requisitos del nivel actual, especialmente si la autenticación moderna está en el roadmap de una aplicación. Anteriormente, el ASVS requería MFA obligatorio. NIST no requiere MFA obligatorio. Por lo tanto, hemos utilizado una designación opcional en este capítulo para indicar dónde el ASVS alienta pero no requiere un control. Las siguientes claves se utilizan en todo este estándar:

Marcado	Descripción
	No requerido
o	Recomendado, pero no requerido
✓	Requerido

V2.1 Seguridad de Contraseña

Las contraseñas, llamadas "Secretos memorizados" por NIST 800-63, incluyen contraseñas, PIN, patrones de desbloqueo, elegir el gatito correcto u otro elemento de imagen y frases de contraseña. Generalmente se consideran "algo que sabes", y a menudo se utilizan como autenticadores de un solo factor. Hay desafíos significativos para el uso continuo de la autenticación de un solo factor, incluyendo miles de millones de nombres de usuario y contraseñas válidos divulgados en Internet, contraseñas predeterminadas o débiles, tablas arcoíris y diccionarios ordenados de las contraseñas más comunes.

Las aplicaciones deben incentivar el uso de autenticación multifactor y deben permitir a los usuarios re-usar los tokens que ya poseen, como tokens FIDO o U2F, o vincular a un proveedor de servicios de credenciales que proporcione autenticación multifactor.

Los proveedores de servicios de credenciales (CSPs; Por sus siglas en inglés) proporcionan identidad federada a los usuarios. Los usuarios a menudo tendrán más de una identidad con varios CSP, como una identidad empresarial con Azure AD, Okta, Ping Identity o Google, o la identidad del consumidor mediante Facebook, Twitter, Google o WeChat, por nombrar solo algunas alternativas comunes. Esta lista no es un respaldo de estas empresas o servicios, sino simplemente un estímulo para que los desarrolladores consideren la realidad de que muchos usuarios tienen muchas identidades establecidas. Las organizaciones deben considerar la integración con las identidades de usuario existentes, según el perfil de riesgo de la prueba de identidad del CSP. Por ejemplo, es poco probable que una organización gubernamental acepte una identidad de las redes sociales como un inicio de sesión para sistemas sensibles, ya que es fácil crear identidades falsas o desechadas, mientras que una compañía de juegos móviles bien puede necesitar integrarse con las principales plataformas de medios sociales para hacer crecer su base de jugadores activos.

#	Descripción	L1	L2	L3	CWE	NIST §
2.1.1	Verifique que las contraseñas del usuarios tienen al menos 12 caracteres de longitud (después de combinar varios espacios). (C6)	✓	✓	✓	521	5.1.1.2
2.1.2	Verifique que se permitan contraseñas de al menos 64 caracteres y que se denieguen contraseñas de más de 128 caracteres. (C6)	✓	✓	✓	521	5.1.1.2
2.1.3	Verifique que no se realiza el truncamiento de contraseña. Sin embargo, varios espacios consecutivos pueden ser reemplazados por un solo espacio. (C6)	✓	✓	✓	521	5.1.1.2
2.1.4	Verifique que cualquier carácter Unicode imprimible, incluidos los caracteres neutros del idioma, como espacios y Emojis esté permitido en las contraseñas.	✓	✓	✓	521	5.1.1.2
2.1.5	Verifique que los usuarios pueden cambiar su contraseña.	✓	✓	✓	620	5.1.1.2
2.1.6	Verifique que la funcionalidad de cambio de contraseña requiere la contraseña actual y nueva del usuario.	✓	✓	✓	620	5.1.1.2

#	Descripción	L1	L2	L3	CWE	NIST §
2.1.7	Verifique que las contraseñas enviadas durante el registro de la cuenta, el inicio de sesión y el cambio de contraseña se comprueban localmente contra un conjunto de contraseñas filtradas (como las 1,000 o 10,000 contraseñas más comunes que coinciden con la directiva de contraseñas del sistema) o mediante una API externa. Si se utiliza una API, una prueba de zero knowledge u otro mecanismo, asegúrese que la contraseña en texto plano no se envía ni se utiliza para verificar el estado de filtración de la contraseña. Si la contraseña esta filtrada, la aplicación debe exigir al usuario que establezca una nueva contraseña no filtrada. (C6)	✓	✓	✓	521	5.1.1.2
2.1.8	Verifique que se proporciona un medidor de fortaleza de la contraseña para ayudar a los usuarios a establecer una contraseña más segura.	✓	✓	✓	521	5.1.1.2
2.1.9	Verifique que no hay reglas de composición de contraseñas que limiten el tipo de caracteres permitidos. No debe haber ningún requisito para mayúsculas o minúsculas o números o caracteres especiales. (C6)	✓	✓	✓	521	5.1.1.2
2.1.10	Verifique que no haya rotación periódica de credenciales o solicitud del historial de contraseñas.	✓	✓	✓	263	5.1.1.2
2.1.11	Verifique que se permite la funcionalidad "pegar", las aplicaciones auxiliares de contraseñas del browser y los administradores externos de contraseñas.	✓	✓	✓	521	5.1.1.2
2.1.12	Verifique que el usuario puede elegir entre ver temporalmente toda la contraseña enmascarada o ver temporalmente el último caracter escrito de la contraseña en plataformas que no tienen esto como funcionalidad integrada.	✓	✓	✓	521	5.1.1.2

Nota: El objetivo de permitir que el usuario vea su contraseña o vea el último carácter temporalmente es mejorar la facilidad de uso de la entrada de credenciales, especialmente en torno al uso de contraseñas más largas, frases de contraseña y administradores de contraseñas. Otra razón para incluir el requisito es para limitar o impedir informes de prueba que sugieren innecesariamente que las organizaciones invaliden el comportamiento de campo de contraseña de plataforma integrada para eliminar esta experiencia de seguridad moderna y fácil de usar.

V2.2 Seguridad General del Autenticador

La agilidad del autenticador es esencial para aplicaciones preparadas para el futuro. Refactorice los verificadores de aplicaciones para permitir autenticadores adicionales según las preferencias del usuario, así como permitir la retirada de autenticadores obsoletos o inseguros de forma ordenada.

NIST considera el email y el SMS como autenticadores de tipo "restringido", y es probable que sean eliminados del NIST 800-63 y por lo tanto el ASVS en algún momento el futuro. Las aplicaciones deben planificar un roadmap que no requiera el uso de correo electrónico o SMS.

#	Descripción	L1	L2	L3	CWE	NIST §
2.2.1	Verifique que los controles anti-automatización son efectivos para mitigar las pruebas de credenciales filtradas, fuerza bruta y ataques de bloqueo de cuentas. Estos controles incluyen el bloqueo de las contraseñas filtradas más comunes, bloqueos suaves, limitación de velocidad, CAPTCHA, retrasos cada vez mayores entre intentos, restricciones de direcciones IP o restricciones basadas en riesgos, como la ubicación, el primer inicio de sesión en un dispositivo, los intentos recientes de desbloquear la cuenta o similares. Verifique que no sea posible realizar más de 100 intentos fallidos por hora en una sola cuenta.	✓	✓	✓	307	5.2.2 / 5.1.1.2 / 5.1.4.2 / 5.1.5.2
2.2.2	Verifique que el uso de autenticadores débiles (como SMS y correo electrónico) se limita a la verificación secundaria y la aprobación de transacciones y no como un reemplazo para métodos de autenticación más seguros. Verifique que se ofrezcan métodos más fuertes y no métodos débiles, que los usuarios sean conscientes de los riesgos o que se tomen las medidas adecuadas para limitar los riesgos de compromiso de la cuenta.	✓	✓	✓	304	5.2.10
2.2.3	Verifique que las notificaciones seguras se envían a los usuarios después de las actualizaciones de los detalles de autenticación, como restablecimientos de credenciales, cambios de correo electrónico o dirección, inicio de sesión desde ubicaciones desconocidas o de riesgo. Se prefiere el uso de notificaciones push - en lugar de SMS o correo electrónico - , pero en ausencia de notificaciones push, SMS o correo electrónico es aceptable siempre y cuando no se divulgue información confidencial en la notificación.	✓	✓	✓	620	
2.2.4	Verifique la resistencia a la suplantación contra el phishing, como el uso de la autenticación multifactor, los dispositivos criptográficos con intención (como las claves conectadas con un push para autenticarse) o en niveles AAL más altos, certificados del lado cliente.			✓	308	5.2.5
2.2.5	Verifique que donde se separan un proveedor de servicios de credenciales (CSP) y la aplicación que comprueba la autenticación, el TLS mutuamente autenticado está en su lugar entre los dos endpoints.			✓	319	5.2.6
2.2.6	Verifique la resistencia a la reproducción mediante el uso obligatorio de dispositivos de one-time password (OTP), autenticadores criptográficos o códigos de búsqueda.			✓	308	5.2.8
2.2.7	Verifique la intención de autenticarse exigiendo la entrada de un token de OTP o una acción iniciada por el usuario, como una pulsación de botón en un teclado de hardware FIDO.			✓	308	5.2.9

V2.3 Ciclo de Vida del Autenticador

Los autenticadores son contraseñas, tokens de software, tokens de hardware y dispositivos biométricos. El ciclo de vida de los autenticadores es fundamental para la seguridad de una aplicación: si alguien puede registrar automáticamente una cuenta sin evidencia de identidad, puede haber poca confianza en la aserción de identidad. Para sitios de redes sociales como Reddit, está perfectamente bien. Para los sistemas bancarios,

un mayor enfoque en el registro y la emisión de credenciales y dispositivos es fundamental para la seguridad de la aplicación.

Nota: Las contraseñas no deben tener una duración máxima ni estar sujetas a la rotación de contraseñas. Las contraseñas deben comprobarse si han sido filtradas, no pedir que se reemplacen regularmente.

#	Descripción	L1	L2	L3	CWE	NIST §
2.3.1	Verifique que las contraseñas iniciales o los códigos de activación generados por el sistema DEBEN ser generados de forma aleatoreamente segura, DEBE tener al menos 6 caracteres de largo y PUEDE contener letras y números, y expirar después de un corto período de tiempo. Estos secretos iniciales no deben permitirse su re-utilización para convertirse en la contraseña a largo plazo.	✓	✓	✓	330	5.1.1.2 / A.3
2.3.2	Verifique que se admite la inscripción y el uso de dispositivos de autenticación proporcionados por el suscriptor, como tokens U2F o FIDO.		✓	✓	308	6.1.3
2.3.3	Verifique que las instrucciones de renovación se envían con tiempo suficiente para renovar los autenticadores con límite de tiempo.		✓	✓	287	6.1.4

V2.4 Almacenamiento de Credenciales

Los arquitectos y desarrolladores deben adherirse a esta sección al crear o refactorizar código. Esta sección solo se puede verificar completamente mediante la revisión del código fuente o mediante pruebas de unidad o integración seguras. Las pruebas de penetración no pueden identificar ninguno de estos problemas.

La lista de funciones de derivación aprobadas de one-way key se detalla en la sección 5.1.1.2 del NIST 800-63 B, y en BSI Kryptographische Verfahren: Empfehlungen und Schlussellängen (2018). El último algoritmo nacional o regional y los estándares de longitud de clave se pueden elegir en lugar de estas opciones.

Esta sección no se puede probar en penetración, por lo que los controles no se marcan como L1. Sin embargo, esta sección es de vital importancia para la seguridad de las credenciales si son robadas, por lo que si bifurca el ASVS para una arquitectura o directriz de codificación o lista de comprobación de revisión de código fuente, coloque estos controles de nuevo en L1 en su versión privada.

#	Descripción	L1	L2	L3	CWE	NIST §
2.4.1	Verifique que las contraseñas se almacenan en un forma tal que resisten ataques sin conexión. Las contraseñas DEBERÁN usar hash con salto mediante una derivación de llave de una sola vía aprobada o función de hash de contraseña. Las funciones derivación de llave y hash de contraseñas toman una contraseña, una salto y un factor de costo como entradas al generar un hash de contraseña. (C6)		✓	✓	916	5.1.1.2
2.4.2	Verifique que el salto tiene al menos 32 bits de longitud y que se elige arbitrariamente para minimizar las colisiones de valor de salto entre los hashes almacenados. Para cada credencial, se DEBE almacenar un único valor de salto y el hash resultante. (C6)		✓	✓	916	5.1.1.2
2.4.3	Verifique que si se utiliza PBKDF2, el recuento de iteraciones DEBE ser tan grande como el rendimiento del servidor de verificación lo permita, normalmente de al menos 100,000 iteraciones. (C6)		✓	✓	916	5.1.1.2
2.4.4	Verifique que si se utiliza bcrypt, el factor de trabajo DEBE ser tan grande como lo permita el rendimiento del servidor de verificación, con un mínimo de 10. (C6)		✓	✓	916	5.1.1.2

#	Descripción	L1	L2	L3	CWE	NIST §
2.4.5	Verifique que se realiza una iteración adicional de una función de derivación de claves, utilizando un valor de salto que es secreto y que solo conoce el verificador. Genere el valor de salto utilizando un generador de bits aleatorios aprobado [SP 800-90Ar1] y proporcione al menos la fuerza de seguridad mínima especificada en la última revisión del SP 800-131A. El valor secreto del salto se almacenará por separado de las contraseñas hash (p. ej., en un dispositivo especializado como un módulo de seguridad de hardware).		✓	✓	916	5.1.1.2

Cuando se mencionan las normas estadounidenses, se puede utilizar una norma regional o local en lugar de la norma estadounidense o además de la norma estadounidense según sea necesario.

V2.5 Recuperación de Credenciales

#	Descripción	L1	L2	L3	CWE	NIST §
2.5.1	Verifique que un secreto de activación o recuperación inicial generado por el sistema no se envíe en texto claro al usuario. (C6)	✓	✓	✓	640	5.1.1.2
2.5.2	Verificar sugerencias de contraseña o autenticación basada en conocimientos (las llamadas "preguntas secretas") no están presentes.	✓	✓	✓	640	5.1.1.2
2.5.3	Verificar la recuperación de credenciales de contraseña no revela la contraseña actual de ninguna manera. (C6)	✓	✓	✓	640	5.1.1.2
2.5.4	Verificar que las cuentas compartidas o predeterminadas no estén presentes (por ejemplo. "root", "admin", o "sa").	✓	✓	✓	16	5.1.1.2 / A.3
2.5.5	Verifique que si se cambia o reemplaza un factor de autenticación, se notifica al usuario de este evento.	✓	✓	✓	304	6.1.2.3
2.5.6	Verifique la contraseña olvidada y otras rutas de recuperación utilizan un mecanismo de recuperación seguro, como OTP basado en el tiempo (TOTP) u otro token de software, mobile push u otro mecanismo de recuperación sin conexión. (C6)	✓	✓	✓	640	5.1.1.2
2.5.7	Verifique que si se pierden factores de autenticación OTP o multifactor, esa evidencia de prueba de identidad se realiza al mismo nivel que durante la inscripción.		✓	✓	308	6.1.2.3

V2.6 Verificador de Secretos de Look-up

Secretos de Look up (secretos de búsqueda) son listas generadas previamente de códigos secretos, similares a los números de autorización de transacción (TAN), los códigos de recuperación de redes sociales o una cuadrícula que contiene un conjunto de valores aleatorios. Estos se distribuyen de forma segura a los usuarios. Estos códigos de búsqueda se utilizan una vez y, una vez que se utilizan todos, se descarta la lista secreta de búsqueda. Este tipo de autenticador se considera "algo que tienes".

#	Descripción	L1	L2	L3	CWE	NIST §
2.6.1	Verifique que los secretos de búsqueda solo se pueden usar una vez.		✓	✓	308	5.1.2.2
2.6.2	Verifique que los secretos de búsqueda tengan suficiente aleatoriedad (112 bits de entropía), o si menos de 112 bits de entropía, saltados con un única y aleatoria salto de 32 bits y hasheado con un hash aprobado de una sola vía.		✓	✓	330	5.1.2.2

#	Descripción	L1	L2	L3	CWE	NIST §
2.6.3	Verifique que los secretos de búsqueda son resistentes a los ataques sin conexión, como los valores predecibles.		✓	✓	310	5.1.2.2

V2.7 Verificador Fuera de Banda

En el pasado, un verificador fuera de banda común habría sido un correo electrónico o SMS que contiene un enlace de restablecimiento de contraseña. Los atacantes utilizan este mecanismo débil para restablecer las cuentas que aún no controlan, como tomar el control de la cuenta de correo electrónico de una persona y volver a usar los vínculos de restablecimiento descubiertos. Hay mejores maneras de manejar la verificación fuera de banda.

Los autenticadores seguros fuera de banda son dispositivos físicos que pueden comunicarse con el verificador a través de un canal secundario seguro. Algunos ejemplos son las notificaciones push a dispositivos móviles. Este tipo de autenticador se considera "algo que tienes". Cuando un usuario desea autenticarse, la aplicación de verificación envía un mensaje al autenticador fuera de banda a través de una conexión al autenticador directa o indirectamente a través de un servicio de terceros. El mensaje contiene un código de autenticación (normalmente un número aleatorio de seis dígitos o un cuadro de diálogo de aprobación modal). La aplicación de comprobación espera a recibir el código de autenticación a través del canal principal y compara el hash del valor recibido con el hash del código de autenticación original. Si coinciden, el verificador fuera de banda puede suponer que el usuario se ha autenticado.

ASVS asume que sólo unos pocos desarrolladores desarrollarán nuevos autenticadores fuera de banda, como notificaciones push, y por lo tanto los siguientes controles ASVS se aplican a los verificadores, como la API de autenticación, las aplicaciones y las implementaciones de inicio de sesión único. Si está desarrollando un nuevo autenticador fuera de banda, por favor refiérase a NIST 800-63B § 5.1.3.1.

No se permiten autenticadores inseguros fuera de banda, como el correo electrónico y VoIP. La autenticación RTC y SMS está actualmente "restringida" por NIST y debe estar en desuso en favor de las notificaciones push o similares. Si necesita utilizar la autenticación telefónica o sms fuera de banda, por favor consulte § 5.1.3.3.

#	Descripción	L1	L2	L3	CWE	NIST §
2.7.1	Verifique que los autenticadores de texto sin cifrar fuera de banda tales como PSTN o SMS ("restringido por NIST") no se ofrecen de forma predeterminada, y que en primer lugar se ofrecen alternativas más sólidas, como las notificaciones push.	✓	✓	✓	287	5.1.3.2
2.7.2	Verifique que el verificador fuera de banda expira después de 10 minutos, fuera de las solicitudes de autenticación de banda, códigos o tokens.	✓	✓	✓	287	5.1.3.2
2.7.3	Verifique que las solicitudes de autenticación, los códigos o los tokens de verificador fuera de banda solo se pueden usar una vez y solo para la solicitud de autenticación original.	✓	✓	✓	287	5.1.3.2
2.7.4	Verifique que el autenticador y el verificador fuera de banda se comuniquen a través de un canal independiente seguro.	✓	✓	✓	523	5.1.3.2
2.7.5	Verifique que el verificador fuera de banda conserva solo una versión hasheada del código de autenticación.		✓	✓	256	5.1.3.2
2.7.6	Verifique que el código de autenticación inicial sea generado por un generador de números aleatorios seguro, que contiene al menos 20 bits de entropía (normalmente un número aleatorio digital de seis es suficiente).		✓	✓	310	5.1.3.2

V2.8 Verificador de Una Sola Vez

Las contraseñas de una sola vez de un solo factor (OTPs) son tokens físicos o flexibles que muestran un desafío pseudoaleatorio que cambia continuamente. Estos dispositivos hacen que el phishing (suplantación) sea difícil, pero no imposible. Este tipo de autenticador se considera "algo que tienes". Los tokens multifactor son similares a los OTP de un solo factor, pero requieren un código PIN válido, desbloqueo biométrico, inserción USB o emparejamiento NFC o algún valor adicional (como calculadoras de firma de transacciones) que se introduzcan para crear el OTP final.

#	Descripción	L1	L2	L3	CWE	NIST §
2.8.1	Verifique que los OTP basados en el tiempo tienen una duración definida antes de expirar.	✓	✓	✓	613	5.1.4.2 / 5.1.5.2
2.8.2	Verifique que las claves simétricas utilizadas para comprobar los OTP enviados están altamente protegidas, por ejemplo, mediante el uso de un módulo de seguridad de hardware o almacenamiento seguro de claves basadas en el sistema operativo.		✓	✓	320	5.1.4.2 / 5.1.5.2
2.8.3	Verifique que los algoritmos criptográficos aprobados se utilizan en la generación, siembra y verificación de OTP.		✓	✓	326	5.1.4.2 / 5.1.5.2
2.8.4	Verifique que el OTP basado en el tiempo se pueda utilizar solamente una vez dentro del período de validez.		✓	✓	287	5.1.4.2 / 5.1.5.2
2.8.5	Verifique que si se reutiliza un token OTP multifactor basado en el tiempo durante el período de validez, se registra en logs y se rechaza con notificación segura enviada al titular del dispositivo.		✓	✓	287	5.1.5.2
2.8.6	Verifique que el generador OTP de un solo factor físico pueda ser revocado en caso de robo u otra pérdida. Asegúrese de que la revocación es efectiva inmediatamente en todas las sesiones iniciadas, independientemente de la ubicación.		✓	✓	613	5.2.1
2.8.7	Verifique que los autenticadores biométricos se limitan a usarlos solo como factores secundarios junto con algo que Ud. tiene y algo que Ud. sabe.		o	✓	308	5.2.3

V2.9 Verificador Criptográfico

Las claves de seguridad criptográficas son tarjetas inteligentes o claves FIDO, donde el usuario tiene que conectar o emparejar el dispositivo criptográfico al equipo para completar la autenticación. Los verificadores envían un mensaje de desafío a los dispositivos o software criptográficos, y el dispositivo o software calcula una respuesta basada en una clave criptográfica almacenada de forma segura.

Los requisitos para los dispositivos y software criptográficos de un solo factor, y los dispositivos y software criptográficos multifactor son los mismos, ya que la verificación del autenticador criptográfico demuestra la posesión del factor de autenticación.

#	Descripción	L1	L2	L3	CWE	NIST §
2.9.1	Verifique que las claves criptográficas utilizadas en la verificación se almacenan de forma segura y protegidas contra la divulgación, como el uso de un módulo de plataforma segura (TPM) o un módulo de seguridad de hardware (HSM) o un servicio de sistema operativo que puede utilizar este almacenamiento seguro.		✓	✓	320	5.1.7.2

#	Descripción	L1	L2	L3	CWE	NIST §
2.9.2	Verifique que el mensaje de desafío tenga al menos 64 bits de longitud y sea estadísticamente único o sea único a lo largo de la vida útil del dispositivo criptográfico.		✓	✓	330	5.1.7.2
2.9.3	Verifique que se utilizan algoritmos criptográficos aprobados en la generación, la semilla y la verificación.		✓	✓	327	5.1.7.2

V2.10 Autenticación de Servicio

Esta categoría no es comprobable con test de penetración, por lo que no tiene ningún requisito L1. Sin embargo, si se utiliza en una arquitectura, codificación o revisión de código segura, suponga que el software (al igual que Java Key Store) es el requisito mínimo en L1. El almacenamiento de texto claro de los secretos no es aceptable bajo ninguna circunstancia.

#	Descripción	L1	L2	L3	CWE	NIST §
2.10.1	Verifique que los secretos dentro del servicio no se basan en credenciales invariables, como contraseñas, claves de API o cuentas compartidas con acceso con privilegios.		OS assisted	HSM	287	5.1.1.1
2.10.2	Verifique que si las contraseñas son necesarias para la autenticación de servicio, la cuenta de servicio utilizada no es una credencial predeterminada. (p. ej., root/root o admin/admin son predeterminados en algunos servicios durante la instalación).		OS assisted	HSM	255	5.1.1.1
2.10.3	Verifique que las contraseñas se almacenan con suficiente protección para evitar ataques de recuperación sin conexión, incluido el acceso al sistema local.		OS assisted	HSM	522	5.1.1.1
2.10.4	Verifique que las contraseñas, las integraciones con bases de datos y sistemas de terceros, las semillas y los secretos internos y las claves de API se administran de forma segura y no se incluyen en el código fuente ni se almacenan en los repositorios de código fuente. Dicho almacenamiento DEBE resistir ataques fuera de línea. Se recomienda el uso de un almacén de claves de software seguro (L1), TPM de hardware o un HSM (L3) para el almacenamiento de contraseñas.		OS assisted	HSM	798	

Requisitos Adicionales de Agencias de EE.UU.

Las agencias de EEUU tienen requisitos obligatorios relativos al NIST 800-63. ASVS siempre ha sido casi el 80% de los controles que se aplican a casi el 100% de las aplicaciones, y no el último 20% de los controles avanzados o aquellos que tienen una aplicabilidad limitada. Como tal, el ASVS es un subconjunto estricto de NIST 800-63, especialmente para las clasificaciones IAL1/2 y AAL1/2, pero no es lo suficientemente completo, especialmente en lo que respecta a las clasificaciones IAL3/AAL3.

Instamos encarecidamente a las agencias de EEUU a que revisen e implementen NIST 800-63 en su totalidad.

Glosario de términos

Termino	Significado
CSP	Proveedor de servicios de credenciales también llamado proveedor de identidades
Authenticator	Código que autentica una contraseña, un token, MFA, una aserción federada, etc.

Termino	Significado
Verifier	"Una entidad que verifica la identidad del reclamante verificando la posesión y el control del reclamante de uno o dos autenticadores mediante un protocolo de autenticación. Para ello, es posible que el verificador también necesite validar las credenciales que vinculan el autenticador con el identificador del suscriptor y comprobar su estado"
OTP	Contraseña de una sola vez
SFA	Autenticadores de un solo factor, como algo que conoces (secretos memorizados, contraseñas, frases de contraseña, PIN), algo que eres (biometría, huellas dactilares, escaneos faciales) o algo que tienes (tokens OTP, un dispositivo criptográfico como una tarjeta inteligente)
MFA	Autenticación multifactor, que incluye dos o más factores individuales

Referencias

Para obtener más información, véase también:

- NIST 800-63 - Digital Identity Guidelines
- NIST 800-63 A - Enrollment and Identity Proofing
- NIST 800-63 B - Authentication and Lifecycle Management
- NIST 800-63 C - Federation and Assertions
- NIST 800-63 FAQ
- OWASP Testing Guide 4.0: Testing for Authentication
- OWASP Cheat Sheet - Password storage
- OWASP Cheat Sheet - Forgot password
- OWASP Cheat Sheet - Choosing and using security questions

V3 Gestión de sesiones

Objetivo de Control

Uno de los componentes principales de cualquier aplicación basada en web o stateful API es el mecanismo por el cual controla y mantiene el estado de un usuario o dispositivo que interactúa con él. La gestión de sesiones cambia un protocolo sin estado (stateless) a uno con estado (stateful), lo que es fundamental para diferenciar diferentes usuarios o dispositivos.

Asegúrese de que una aplicación verificada cumple los siguientes requisitos de gestión de sesiones de alto nivel:

- Las sesiones son únicas para cada individuo y no se pueden adivinar ni compartir.
- Las sesiones se invalidan cuando ya no son necesarias y se agota el tiempo de espera durante los períodos de inactividad.

Como se ha señalado anteriormente, estos requisitos se han adaptado para ser un subconjunto compatible de controles NIST 800-63b seleccionados, centrados en amenazas comunes y debilidades de autenticación comúnmente explotadas. Los requisitos de verificación anteriores han sido retirados, eliminado redundancias, o en la mayoría de los casos, adaptados para estar fuertemente alineados con los requisitos obligatorios de NIST 800-63b.

Requisitos de Verificación de Seguridad

V3.1 Seguridad Fundamental en la Gestión de Sesiones

#	Descripción	L1	L2	L3	CWE	NIST §
3.1.1	Verifique que la aplicación nunca revela tokens de sesión en parámetros de dirección URL.	✓	✓	✓	598	

V3.2 Binding de Sesión

#	Descripción	L1	L2	L3	CWE	NIST §
3.2.1	Verifique que la aplicación genera un nuevo token de sesión en la autenticación de usuario. (C6)	✓	✓	✓	384	7.1
3.2.2	Verifique que los tokens de sesión posean al menos 64 bits de entropía. (C6)	✓	✓	✓	331	7.1
3.2.3	Verifique que la aplicación solo almacena tokens de sesión en el navegador mediante métodos seguros, como proteger las cookies adecuadamente (consulte la sección 3.4) o el almacenamiento de sesión en HTML 5.	✓	✓	✓	539	7.1
3.2.4	Verifique que los tokens de sesión se generan mediante algoritmos criptográficos aprobados. (C6)		✓	✓	331	7.1

TLS u otro canal de transporte seguro es obligatorio para la gestión de sesiones. Esto se trata en el capítulo Seguridad de las comunicaciones.

V3.3 Terminación de Sesión

Los tiempos de espera de sesión se han alineado con NIST 800-63, lo que permite tiempos de espera de sesión mucho más largos de lo permitido tradicionalmente por los estándares de seguridad. Las organizaciones deben revisar la tabla siguiente y, si es deseable un tiempo de espera más largo en función del riesgo de la aplicación, el valor NIST debe ser los límites superiores de los tiempos de espera de inactividad de sesión.

L1 en este contexto es IAL1/AAL1, L2 es IAL2/AAL3, L3 es IAL3/AAL3. Para IAL2/AAL2 e IAL3/AAL3, el tiempo de espera de inactividad más corto es, el límite inferior de los tiempos de inactividad para ser cerrado o re-autenticado para reanudar la sesión.

#	Descripción	L1	L2	L3	CWE	NIST §
3.3.1	Verifique que el cierre de sesión y la expiración invalidan el token de sesión, de modo que el botón "Atrás" o un usuario de confianza posterior no reanude una sesión autenticada, incluso entre los usuarios de confianza. (C6)	✓	✓	✓	613	7.1
3.3.2	Si los autenticadores permiten a los usuarios permanecer conectados, compruebe que la re-autenticación se produce periódicamente tanto cuando se utiliza activamente o después de un período de inactividad. (C6)	30 días	12 horas o 30 minutos de inactividad, 2FA opcional	12 horas o 15 minutos de inactividad, con 2FA	613	7.2
3.3.3	Verifique que la aplicación ofrece la opción de terminar todas las demás sesiones activas después de un cambio de contraseña correcto (incluido el cambio mediante el restablecimiento/recuperación de contraseña), y que esto es efectivo en toda la aplicación, el inicio de sesión federado (si está presente) y cualquier usuario de confianza.		✓	✓	613	
3.3.4	Verifique que los usuarios pueden ver y (habiendo vuelto a introducir las credenciales de inicio de sesión) cerrar sesión en cualquiera o todas las sesiones y dispositivos activos actualmente.		✓	✓	613	7.1

V3.4 Gestión de Sesión Basada en Cookie

#	Descripción	L1	L2	L3	CWE	NIST §
3.4.1	Verifique que los tokens de sesión basados en cookies tengan el atributo 'Secure' establecido. (C6)	✓	✓	✓	614	7.1.1
3.4.2	Verifique que los tokens de sesión basados en cookies tienen el atributo 'HttpOnly' establecido. (C6)	✓	✓	✓	1004	7.1.1
3.4.3	Verifique que los tokens de sesión basados en cookies utilizan el atributo 'SameSite' para limitar la exposición a ataques de falsificación de solicitudes entre sitios. (C6)	✓	✓	✓	16	7.1.1
3.4.4	Verifique que los tokens de sesión basados en cookies utilizan el prefijo "__Host-" para que las cookies solo se envíen al host que configuró inicialmente la cookie.	✓	✓	✓	16	7.1.1

#	Descripción	L1	L2	L3	CWE	NIST §
3.4.5	Verifique que si la aplicación se publica bajo un nombre de dominio con otras aplicaciones que establecen o usan cookies de sesión que podrían revelar las cookies de sesión, establezca el atributo de ruta en tokens de sesión basados en cookies utilizando la ruta más precisa posible. (C6)	✓	✓	✓	16	7.1.1

V3.5 Administración de Sesiones Basada en Tokens

La gestión de sesiones basada en tokens incluye JWT, OAuth, SAML y API keys. De estos, se sabe que las API keys son débiles y no se deben usar en código nuevo.

#	Descripción	L1	L2	L3	CWE	NIST §
3.5.1	Verifique que la aplicación permite a los usuarios revocar tokens de OAuth que forman relaciones de confianza con aplicaciones vinculadas.		✓	✓	290	7.1.2
3.5.2	Verifique que la aplicación utiliza tokens de sesión en lugar de claves y secretos de API estáticos, excepto con implementaciones heredadas.		✓	✓	798	
3.5.3	Verifique que los tokens de sesión sin estado utilizan firmas digitales, cifrado y otras contramedidas para protegerse contra ataques de manipulación, envolvente, reproducción, cifrado nulo y sustitución de claves.		✓	✓	345	

V3.6 Reautenticación Federada

Esta sección se relaciona con aquellos que escriben el código de Partes de confianza (RP) o proveedores de servicios de credenciales (CSP). Si confía en el código que implementa estas características, asegúrese de que estos problemas se manejen correctamente.

#	Descripción	L1	L2	L3	CWE	NIST §
3.6.1	Verifique que las partes de confianza (RP) especifiquen el tiempo máximo de autenticación para los proveedores de servicios de credenciales (CSP) y que los CSP vuelvan a autenticar al usuario si no han utilizado una sesión dentro de ese período.			✓	613	7.2.1
3.6.2	Verifique que los proveedores de servicios de credenciales (CSP) informan a las partes de confianza (RP) del último evento de autenticación, para permitir que los RP determinen si necesitan volver a autenticar al usuario.			✓	613	7.2.1

V3.7 Defensas Contra las Vulnerabilidades de Gestión de Sesiones

Hay un pequeño número de ataques de gestión de sesiones, algunos relacionados con la experiencia del usuario (UX) de las sesiones. Anteriormente, sobre la base de los requisitos ISO 27002, el ASVS ha requerido bloquear varias sesiones simultáneas. El bloqueo de sesiones simultáneas ya no es adecuado, no sólo porque los usuarios modernos tienen muchos dispositivos o la aplicación es una API sin una sesión de navegador, pero en la mayoría de estas implementaciones, el último autenticador gana, que a menudo es el atacante. En esta categoría se proporcionan instrucciones principales sobre cómo disuadir, retrasar y detectar ataques de gestión de sesiones mediante código.

Descripción del Ataque semi-abierto

A principios de 2018, varias instituciones financieras se vieron comprometidas usando lo que los atacantes llamaron "ataques semi abiertos". Este término se ha atascado en la industria. Los atacantes golpearon múltiples instituciones con diferentes bases de código propietario, y de hecho parece diferentes bases de código dentro de las mismas instituciones. El ataque semi-abierto explota un defecto de patrón de diseño que se encuentra comúnmente en muchos sistemas existentes de autenticación, gestión de sesiones y control de acceso.

Los atacantes inician un ataque semi-abierto al intentar bloquear, restablecer o recuperar una credencial. Un patrón de diseño de gestión de sesiones popular reutiliza los objetos/modelos de sesión de perfil de usuario entre objetos/modelos no autenticados y autenticados a semi-autorizado (restablecimiento de contraseña, nombre de usuario olvidado) y código totalmente autenticado. Este patrón de diseño rellena un objeto de sesión o token válido que contiene el perfil de la víctima, incluidos los roles y hashes de contraseña. Si las comprobaciones de control de acceso en controladores o routers no comprueban correctamente que el usuario ha iniciado sesión por completo, el atacante podrá actuar como si fuera el usuario. Los ataques podrían incluir cambiar la contraseña del usuario a un valor conocido, actualizar la dirección de correo electrónico para realizar un restablecimiento de contraseña válido, deshabilitar la autenticación multifactor o inscribir un nuevo dispositivo MFA, revelar o cambiar claves de API, etc.

#	Descripción	L1	L2	L3	CWE	NIST §
3.7.1	Verifique que la aplicación garantiza una sesión de inicio de sesión completa y válida o requiere una re-autenticación o verificación secundaria antes de permitir cualquier transacción confidencial o modificaciones de la cuenta.	✓	✓	✓	306	

Referencias

Para obtener más información, véase también:

- OWASP Testing Guide 4.0: Session Management Testing
- OWASP Session Management Cheat Sheet
- Set-Cookie Host- prefix details

V4 Control de Acceso

Objetivo de Control

La autorización es el concepto de permitir el acceso a los recursos solo a aquellos a los que se les permite utilizarlos. Asegúrese de que una aplicación verificada cumple los siguientes requisitos de alto nivel:

- Las personas que acceden a los recursos tienen credenciales válidas para hacerlo.
- Los usuarios están asociados a un conjunto bien definido de roles y privilegios.
- Los metadatos de roles y permisos están protegidos contra la reproducción o la manipulación.

Requisitos de Verificación de Seguridad

V4.1 Diseño de Control de Acceso General

#	Descripción	L1	L2	L3	CWE
4.1.1	Verifique que la aplicación aplica las reglas de control de acceso en una capa de servicio de confianza, especialmente si el control de acceso del lado cliente está presente y podría ser bypaseado.	✓	✓	✓	602
4.1.2	Verifique que todos los atributos de usuario y datos y la información de directiva utilizada por los controles de acceso no pueden ser manipulados por los usuarios finales a menos que se autorice específicamente.	✓	✓	✓	639
4.1.3	Verifique que existe el principio de privilegios mínimos: los usuarios solo deben poder acceder a funciones, archivos de datos, direcciones URL, controladores, servicios y otros recursos, para los que poseen una autorización específica. Esto implica protección contra la suplantación y elevación de privilegios. (C7)	✓	✓	✓	285
4.1.4	[ELIMINADO, DUPLICADO DE 4.1.3]				
4.1.5	Verifique que los controles de acceso fallan de forma segura, incluso cuando se produce una excepción. (C10)	✓	✓	✓	285

V4.2 Control de Acceso a Nivel de Operación

#	Descripción	L1	L2	L3	CWE
4.2.1	Verifique que los datos confidenciales y las API están protegidos contra ataques de referencia insegura directa de objetos (IDOR; por sus siglas en inglés) dirigidos a la creación, lectura, actualización y eliminación de registros, como la creación o actualización del registro de otra persona, la visualización de los registros de todos o la eliminación de todos los registros.	✓	✓	✓	639
4.2.2	Verifique que la aplicación o el framework aplica un mecanismo anti-CSRF seguro para proteger la funcionalidad autenticada, y eficaz anti-automatización o anti-CSRF protege la funcionalidad no autenticada.	✓	✓	✓	352

V4.3 Otras Consideraciones de Control de Acceso

#	Descripción	L1	L2	L3	CWE
4.3.1	Verifique que las interfaces administrativas utilicen la autenticación multifactor adecuada para evitar el uso no autorizado.	✓	✓	✓	419

#	Descripción	L1	L2	L3	CWE
4.3.2	Verifique que la exploración de directorios está deshabilitada a menos que se desee deliberadamente. Además, las aplicaciones no deben permitir la detección o divulgación de metadatos de archivos o directorios, como Thumbs.db, .DS_Store, .git o .svn.	✓	✓	✓	548
4.3.3	Verifique que la aplicación tiene autorización adicional (como la autenticación paso a paso o adaptativa) para sistemas de menor valor y/ o segregación de tareas para aplicaciones de alto valor para aplicar controles antifraude según el riesgo de aplicación y fraudes previos.		✓	✓	732

Referencias

Para obtener más información, véase también:

- OWASP Testing Guide 4.0: Authorization
- OWASP Cheat Sheet: Access Control
- OWASP CSRF Cheat Sheet
- OWASP REST Cheat Sheet

V5 Validación, Desinfección y Codificación

Objetivo de Control

La debilidad más común de la seguridad de la aplicación web es la falta de validación adecuada de la entrada procedente del cliente o del entorno antes de usarla directamente sin ninguna codificación de salida (output encoding). Esta debilidad conduce a casi todas las vulnerabilidades significativas en aplicaciones web, como el scripting entre sitios (XSS), la inyección de SQL, la inyección del intérprete, los ataques de configuración regional/Unicode, los ataques del sistema de archivos y los desbordamientos de búfer.

Asegúrese de que una aplicación verificada cumple los siguientes requisitos de alto nivel:

- La validación de entrada y la arquitectura de codificación de salida tienen un canal acordado para evitar ataques de inyección.
- Los datos de entrada están fuertemente tipados, validados, de rango o longitud comprobados, o en el peor de los casos, desinfectados o filtrados.
- Los datos de salida se codifican o escapan según el contexto de los datos lo más cerca posible del intérprete.

Con la arquitectura moderna de aplicaciones web, la codificación de salida es más importante que nunca. Es difícil proporcionar una validación de entrada sólida en determinados escenarios, por lo que el uso de una API más segura, como consultas parametrizadas, frameworks de plantillas de auto-escaping automático o codificación de salida cuidadosamente elegida, es fundamental para la seguridad de la aplicación.

V5.1 Validación de Entrada

Controles de validación de entrada implementados correctamente, utilizando listas de permisos positivas y un fuerte tipado de datos, puede eliminar más del 90% de todos los ataques de inyección. Las comprobaciones de longitud y rango pueden reducir esto aún más. Se requiere la creación de validación de entrada segura durante la arquitectura de la aplicación, los sprints de diseño, la codificación y las pruebas de unidad e integración. Aunque muchos de estos elementos no se pueden encontrar en las pruebas de penetración, los resultados de no implementarlos se encuentran generalmente en V5.3 - Codificación de Salida y Requisitos de Prevención de Inyección. Se recomienda a los desarrolladores y revisores de código seguro que traten esta sección como si se requiriera L1 para todos los elementos para prevenir inyecciones.

#	Descripción	L1	L2	L3	CWE
5.1.1	Verifique que la aplicación tiene defensas contra los ataques de contaminación de parámetros HTTP, especialmente si el marco de la aplicación no hace ninguna distinción sobre el origen de los parámetros de solicitud (GET, POST, cookies, encabezados o variables de entorno).	✓	✓	✓	235
5.1.2	Verifique que los frameworks protegen contra ataques de asignación de parámetros masivos o que la aplicación tiene contramedidas para proteger contra la asignación de parámetros no seguros, como marcar campos privados o similares. (C5)	✓	✓	✓	915
5.1.3	Verifique que todas las entradas (campos de formulario HTML, solicitudes REST, parámetros de URL, encabezados HTTP, cookies, archivos por lotes, fuentes RSS, etc.) se validan mediante validación positiva (lista de permitidos). (C5)	✓	✓	✓	20
5.1.4	Verifique que las estructuras de datos están fuertemente tipados y validados con un esquema definido que incluya caracteres permitidos, longitud y patrón (p. ej., números de tarjeta de crédito, direcciones de correo electrónico, números de teléfono, o validar que dos campos relacionados son razonables, como comprobar que el suburbio y el código postal coinciden). (C5)	✓	✓	✓	20

#	Descripción	L1	L2	L3	CWE
5.1.5	Verifique que las redirecciones y reenvíos de URL solo permiten destinos que aparecen en una lista de permitidos, o muestra una advertencia al redirigir a contenido potencialmente no confiable.	✓	✓	✓	601

V5.2 Requisitos de Sanitización y Sandboxing

#	Descripción	L1	L2	L3	CWE
5.2.1	Verifique que todas las entradas HTML que no son de confianza de los editores WYSIWYG o similares se sanitizan correctamente con una biblioteca de sanitización HTML o una función de marco de trabajo. (C5)	✓	✓	✓	116
5.2.2	Verifique que los datos no estructurados están sanitizados para aplicar medidas de seguridad, como caracteres permitidos y longitud.	✓	✓	✓	138
5.2.3	Verifique que la aplicación sanitiza la entrada del usuario antes de pasar a los sistemas de correo para protegerse contra la inyección SMTP o IMAP.	✓	✓	✓	147
5.2.4	Verifique que la aplicación evita el uso de eval() u otras características de ejecución de código dinámico. Cuando no hay alternativa, cualquier entrada de usuario debe sanitizarse, y ponerlo en sandbox antes de ejecutarse.	✓	✓	✓	95
5.2.5	Verifique que la aplicación protege contra ataques de inyección de plantilla asegurándose que cualquier entrada de usuario que se incluya está sanitizada o en un lugar controlado.	✓	✓	✓	94
5.2.6	Verifique que la aplicación protege contra ataques SSRF, validando o desinfectando datos que no son de confianza o metadatos de archivos HTTP, como nombres de archivo y campos de entrada de URL, y utiliza listas de protocolos permitidos, dominios, rutas de acceso y puertos.	✓	✓	✓	918
5.2.7	Verifique que la aplicación desinfecta, deshabilita o pone en sandbox el contenido proporcionado por el usuario, con scripts de gráficos vectoriales escalables (SVG; por sus siglas en inglés) especialmente en lo que se refiere a XSS resultante de scripts en línea y foreignObject.	✓	✓	✓	159
5.2.8	Verifique que la aplicación desinfecta, deshabilita o pone en sandbox el contenido proporcionado por el usuario, con expresiones en lenguaje de plantilla o script como Markdown, CSS o las hojas de estilo XSL, BBCode o similares.	✓	✓	✓	94

V5.3 Codificación de Salida y Prevención de Inyección

La codificación de salida cercana o adyacente al intérprete en uso es fundamental para la seguridad de cualquier aplicación. Normalmente, la codificación de salida no se conserva, pero se usa para hacer que la salida sea segura en el contexto de salida adecuado para su uso inmediato. Si no se codifica la salida, se producirá una aplicación insegura, inyectable e insegura.

#	Descripción	L1	L2	L3	CWE
5.3.1	Verifique que la codificación de salida es relevante para el intérprete y el contexto requerido. Por ejemplo, utilice codificadores específicamente para valores HTML, atributos HTML, JavaScript, parámetros de URL, encabezados HTTP, SMTP y otros según lo requiera el contexto, especialmente a partir de entradas que no sean de confianza (por ejemplo, nombres con Unicode o apóstrofes, como ねこ u O'Hara). (C4)	✓	✓	✓	116

#	Descripción	L1	L2	L3	CWE
5.3.2	Verifique que la codificación de salida conserva el juego de caracteres y la configuración regional elegidos por el usuario, de modo que cualquier punto de caracteres Unicode sea válido y se maneje de forma segura. (C4)	✓	✓	✓	176
5.3.3	Verifique que el escape de salida basado en contexto, preferiblemente automatizado - o en el peor de los casos, manual - protege contra XSS reflejado, almacenado y basado en DOM. (C4)	✓	✓	✓	79
5.3.4	Verifique que la selección de datos o las consultas de base de datos (por ejemplo, SQL, HQL, ORM, NoSQL) utilizan consultas parametrizadas, ORM, marcos de entidades o están protegidas de los ataques de inyección de base de datos. (C3)	✓	✓	✓	89
5.3.5	Verifique donde los mecanismos parametrizados o más seguros no están presentes, la codificación de la salida en el contexto específico se utiliza para proteger contra ataques de inyección, como el uso de escape SQL para proteger contra la inyección SQL. (C3, C4)	✓	✓	✓	89
5.3.6	Verifique que la aplicación protege contra ataques de inyección de JSON, ataques de "eval" en JSON y evaluación de expresiones de JavaScript. (C4)	✓	✓	✓	830
5.3.7	Verifique que la aplicación protege contra vulnerabilidades de inyección LDAP o que se han implementado controles de seguridad específicos para evitar la inyección LDAP. (C4)	✓	✓	✓	90
5.3.8	Verifique que la aplicación protege contra la inyección de comandos del sistema operativo y que las llamadas al sistema operativo utilizan consultas de sistema operativo parametrizadas o utilicen codificación de salida de línea de comandos contextual. (C4)	✓	✓	✓	78
5.3.9	Verifique que la aplicación protege contra ataques de inclusión de archivos locales (LFI) o de inclusión remota de archivos (RFI).	✓	✓	✓	829
5.3.10	Verifique que la aplicación protege contra ataques de inyección XPath o de inyección XML. (C4)	✓	✓	✓	643

Nota: El uso de consultas parametrizadas o el escape de SQL no siempre es suficiente; los nombres de tabla y columna, ORDER BY, etc., no se pueden escapar. La inclusión de datos proporcionados por el usuario con escape en estos campos da como resultado consultas con errores o inyección SQL.

Nota: El formato SVG permite explícitamente el script ECMA en casi todos los contextos, por lo que puede que no sea posible bloquear todos los vectores SVG XSS completamente. Si se requiere la carga SVG, se recomienda encarecidamente que se sirvan estos archivos cargados como texto/sin formato o que utilicen un dominio de contenido proporcionado por un usuario independiente para evitar que un XSS exitoso tome el control de la aplicación.

V5.4 Memoria, Cadena y Código No Administrado

Los siguientes requisitos solo se aplicarán cuando la aplicación utilice un lenguaje de sistemas o código no administrado.

#	Descripción	L1	L2	L3	CWE
5.4.1	Verifique que la aplicación utiliza cadenas de memoria segura, copia de memoria más segura y aritmética de puntero para detectar o evitar desbordamientos de pila, búffer o heap.		✓	✓	120

#	Descripción	L1	L2	L3	CWE
5.4.2	Verifique que las cadenas de formato no toman entradas potencialmente hostiles y son constantes.		✓	✓	134
5.4.3	Verifique que se utilizan técnicas de validación de signos, intervalos y entradas para evitar desbordamientos de enteros.		✓	✓	190

V5.5 Prevención de Deserialización

#	Descripción	L1	L2	L3	CWE
5.5.1	Verifique que los objetos serializados utilizan comprobaciones de integridad o están cifrados para evitar la creación de objetos hostiles o la manipulación de datos. (C5)	✓	✓	✓	502
5.5.2	Verifique que la aplicación restringe correctamente los analizadores XML para que solo usen la configuración más restrictiva posible y para asegurarse de que las características no seguras, como la resolución de entidades externas, están deshabilitadas para evitar ataques XML eXternal Entity (XXE).	✓	✓	✓	611
5.5.3	Verifique que la deserialización de datos que no son de confianza se evita o está protegida tanto en código personalizado como en bibliotecas de terceros (como analizadores JSON, XML y YAML).	✓	✓	✓	502
5.5.4	Verifique que al analizar JSON en exploradores o backends basados en JavaScript, JSON.parse se utiliza para analizar el documento JSON. No utilice eval() para analizar JSON.	✓	✓	✓	95

Referencias

Para obtener más información, véase también:

- OWASP Testing Guide 4.0: Input Validation Testing
- OWASP Cheat Sheet: Input Validation
- OWASP Testing Guide 4.0: Testing for HTTP Parameter Pollution
- OWASP LDAP Injection Cheat Sheet
- OWASP Testing Guide 4.0: Client Side Testing
- OWASP Cross Site Scripting Prevention Cheat Sheet
- OWASP DOM Based Cross Site Scripting Prevention Cheat Sheet
- OWASP Java Encoding Project
- OWASP Mass Assignment Prevention Cheat Sheet
- DOMPurify - Client-side HTML Sanitization Library
- XML External Entity (XXE) Prevention Cheat Sheet

Para obtener más información sobre el escape automático, consulte:

- Reducing XSS by way of Automatic Context-Aware Escaping in Template Systems
- AngularJS Strict Contextual Escaping
- AngularJS ngBind
- Angular Sanitization
- Angular Template Security

- ReactJS Escaping
- Improperly Controlled Modification of Dynamically-Determined Object Attributes

Para obtener más información sobre la deserialización, consulte:

- OWASP Deserialization Cheat Sheet
- OWASP Deserialization of Untrusted Data Guide

V6 Criptografía almacenada

Objetivo de Control

Asegúrese que una aplicación verificada cumple los siguientes requisitos de alto nivel:

- Todos los módulos criptográficos fallan de forma segura y que los errores se gestionan correctamente.
- Se utiliza un generador de números aleatorios adecuado.
- El acceso a las claves se administra de forma segura.

V6.1 Clasificación de Datos

El activo más importante son los datos procesados, almacenados o transmitidos por una aplicación. Realice siempre una evaluación de impacto en la privacidad para clasificar correctamente las necesidades de protección de datos de los datos almacenados.

#	Descripción	L1	L2	L3	CWE
6.1.1	Verifique que los datos privados regulados se almacenan cifrados mientras están en reposo, como información de identificación personal (PII), información personal confidencial o datos evaluados que puedan estar sujetos al RGPD de la UE.		✓	✓	311
6.1.2	Verifique que los datos de salud regulados se almacenen cifrados mientras están en reposo, como registros médicos, detalles de dispositivos médicos o registros de investigación anonimizados.		✓	✓	311
6.1.3	Verifique que los datos financieros regulados se almacenen cifrados mientras están en reposo, como cuentas financieras, impagos o historial de crédito, registros fiscales, historial de pagos, beneficiarios o registros de mercado o de investigación anonimizados.		✓	✓	311

V6.2 Algoritmos

Los avances recientes en criptografía significan que los algoritmos y longitudes de clave previamente seguros ya no son seguros o suficientes para proteger los datos. Por lo tanto, debe ser posible cambiar algoritmos.

Aunque esta sección no es fácil de demostrar con prueba de penetración, los desarrolladores deben considerar toda esta sección como obligatoria aunque L1 falta en la mayoría de los elementos.

#	Descripción	L1	L2	L3	CWE
6.2.1	Verifique que todos los módulos criptográficos fallan de forma segura y que los errores se gestionan de forma que no se habiliten los ataques "Padding Oracle".	✓	✓	✓	310
6.2.2	Verifique que se utilicen algoritmos, modos y bibliotecas criptográficas probados por la industria o aprobados por el gobierno, en lugar de criptografía codificada personalizada. (C8)		✓	✓	327
6.2.3	Verifique que los modos de vector de inicialización de cifrado, configuración de cifrado y bloque están configurados de forma segura mediante los últimos consejos vigentes.		✓	✓	326
6.2.4	Verifique que los algoritmos de número aleatorio, cifrado o hash, longitudes de clave, rondas, cifrados o modos, se puedan reconfigurar, actualizar o intercambiar en cualquier momento, para protegerse contra ruptura criptográficas. (C8)		✓	✓	326

#	Descripción	L1	L2	L3	CWE
6.2.5	Verifique que los modos de bloque inseguros conocidos (i.e., ECB, etc.), los modos de relleno (i.e. PKCS#1 v1.5, etc.), los cifrados con tamaños de bloque pequeños (i.e. Triple-DES, Blowfish, etc.), y los algoritmos de hashing débiles (i.e. MD5, SHA1, etc.) no se utilizan a menos que sea necesario para la compatibilidad con versiones anteriores.		✓	✓	326
6.2.6	Verifique que los "nonces", los vectores de inicialización y otros números de uso único no se deben usar más de una vez con una clave de cifrado determinada. El método de generación debe ser adecuado para el algoritmo que se está utilizando.		✓	✓	326
6.2.7	Verifique que los datos cifrados se autentiquen a través de firmas, modos de cifrado autenticados, o HMAC para asegurarse de que el texto cifrado no sea alterado por una parte no autorizada.			✓	326
6.2.8	Verifique que todas las operaciones criptográficas son de tiempo constante, sin operaciones de "cortocircuito" en comparaciones, cálculos o devoluciones, para evitar fugas de información.			✓	385

V6.3 Valores Aleatorios

True Pseudo-random Number Generation (PRNG) es increíblemente difícil de hacer bien. Generalmente, las buenas fuentes de entropía dentro de un sistema se agotarán rápidamente si se utilizan en exceso, pero las fuentes con menos aleatoriedad pueden conducir a claves y secretos predecibles.

#	Descripción	L1	L2	L3	CWE
6.3.1	Verifique que todos los números aleatorios, nombres de archivo aleatorios, GUID aleatorios y cadenas aleatorias se generan utilizando el generador de números aleatorios criptográficamente seguro aprobado por el módulo criptográfico cuando estos valores aleatorios están destinados a no ser adivinables por un atacante.		✓	✓	338
6.3.2	Verifique que los GUID aleatorios se crean mediante el algoritmo GUID v4 y un generador de números pseudoaleatorio (CSPRNG) criptográficamente seguro. Los GUID creados con otros generadores de números pseudoaleatorios pueden ser predecibles.		✓	✓	338
6.3.3	Verifique que los números aleatorios se crean con la entropía adecuada incluso cuando la aplicación está bajo carga pesada, o que la aplicación se degrada correctamente en tales circunstancias.			✓	338

V6.4 Gestión de Secretos

Aunque esta sección no se desmuestra facilmente con prueba de penetración, los desarrolladores deben considerar toda esta sección como obligatoria, aunque L1 falta en la mayoría de los elementos.

#	Descripción	L1	L2	L3	CWE
6.4.1	Verifique que una solución de gestión de secretos, como un almacén de claves, se utiliza para crear, almacenar, controlar el acceso y destruir secretos de forma segura. (C8)		✓	✓	798
6.4.2	Verifique que el material de claves no está expuesto a la aplicación, sino que utiliza un módulo de seguridad aislado como un almacén para operaciones criptográficas. (C8)		✓	✓	320

Referencias

Para obtener más información, véase también:

- OWASP Testing Guide 4.0: Testing for weak Cryptography
- OWASP Cheat Sheet: Cryptographic Storage
- FIPS 140-2

V7 Manejo y Registro de Errores

Objetivo de Control

El objetivo principal del control y registro de errores es proporcionar información útil para el usuario, los administradores y los equipos de respuesta a incidentes. El objetivo no es crear cantidades masivas de registros, sino registros de alta calidad, con más señal que el ruido descartado.

Los registros de alta calidad a menudo contendrán datos confidenciales y deben protegerse según las leyes o directivas locales de privacidad de datos. Esto debe incluir:

- No recopilar o registrar información confidencial a menos que sea específicamente necesario.
- Garantizar que toda la información registrada se maneje de forma segura y protegida según su clasificación de datos.
- Asegurarse de que los registros no se almacenan para siempre, pero tienen una duración absoluta que es lo más corta posible.

Si los registros contienen datos privados o confidenciales, la definición de cuyos varían de un país a otro, los registros se convierten en parte de la información más sensible de la aplicación y, por lo tanto, son muy atractivos para los atacantes por derecho propio.

También es importante asegurarse de que la aplicación falla de forma segura y que los errores no revelan información innecesaria.

V7.1 Contenido de Registro de Log

El registro de log de información confidencial es peligroso: los registros se clasifican a sí mismos, lo que significa que deben cifrarse, estar sujetos a políticas de retención y deben divulgarse en las auditorías de seguridad. Asegúrese de que solo la información necesaria se mantiene en logs, y ciertamente no se registre información de pagos, credenciales (incluidos los tokens de sesión), información confidencial o de identificación personal.

V7.1 cubre OWASP Top 10 2017:A10. Como 2017:A10 y esta sección no son comprobables por prueba de penetración, es importante para:

- Desarrolladores para asegurar el cumplimiento total de esta sección, como si todos los elementos estuvieran marcados como L1.
- Probadores de penetración para validar el cumplimiento total de todos los elementos en V7.1 a través de entrevistas, capturas de pantalla o aserción.

#	Descripción	L1	L2	L3	CWE
7.1.1	Verifique que la aplicación no registra las credenciales ni los detalles de pago. Los tokens de sesión solo deben almacenarse en registros de forma irreversible y hasheados. (C9, C10)	✓	✓	✓	532
7.1.2	Verifique que la aplicación no registra otros datos confidenciales tal como se definen en las leyes de privacidad locales o la política de seguridad pertinente. (C9)	✓	✓	✓	532
7.1.3	Verifique que la aplicación registra eventos relevantes para la seguridad, incluidos los eventos de autenticación correctos y con errores, los errores de control de acceso, los errores de deserialización y los errores de validación de entrada. (C5, C7)		✓	✓	778
7.1.4	Verifique que cada evento de registro incluye la información necesaria que permitiría una investigación detallada de la escala de tiempo cuando se produce un evento. (C9)		✓	✓	778

V7.2 Procesamiento del Log

El registro oportuno es fundamental para los eventos de auditoría, el triage y la escalada. Asegúrese de que los registros de la aplicación sean claros y se puedan monitorear y analizar fácilmente, ya sea localmente o el registro se envíe a un sistema de monitoreo remoto.

V7.2 cubre OWASP Top 10 2017:A10. Como 2017:A10 y esta sección no son comprobables por prueba de penetración, es importante para:

- Desarrolladores para asegurar el cumplimiento total de esta categoría, como si todos los elementos estuvieran marcados como L1.
- Probadores de penetración para validar el cumplimiento completo de todos los elementos en V7.2 a través de entrevistas, capturas de pantalla o aserción.

#	Descripción	L1	L2	L3	CWE
7.2.1	Verifique que se registran todas las decisiones de autenticación, sin almacenar tokens o contraseñas de sesión confidenciales. Esto debe incluir solicitudes con los metadatos relevantes necesarios para las investigaciones de seguridad.		✓	✓	778
7.2.2	Verifique que se pueden registrar todas las decisiones de control de acceso y que se registran todas las decisiones erróneas. Esto debe incluir solicitudes con los metadatos pertinentes necesarios para las investigaciones de seguridad.		✓	✓	285

V7.3 Protección de Logs

Los registros que se pueden modificar o eliminar trivialmente son inútiles para investigaciones y procesamientos. La divulgación de registros puede exponer detalles internos sobre la aplicación o los datos que contiene. Se debe tener cuidado al proteger los registros de la divulgación, modificación o eliminación no autorizadas.

#	Descripción	L1	L2	L3	CWE
7.3.1	Verifique que todos los componentes de registro codifiquen adecuadamente los datos para evitar la inyección de registros. (C9)		✓	✓	117
7.3.2	[ELIMINADO, DUPLICADO DE 7.3.1]				
7.3.3	Verifique que los registros de seguridad están protegidos contra el acceso y la modificación no autorizados. (C9)		✓	✓	200
7.3.4	Verifique que la fuente donde se lee el tiempo están sincronizados con la hora y la zona horaria correctas. Considere firmemente el registro solo en UTC si los sistemas son globales para ayudar con el análisis forense posterior al incidente. (C9)		✓	✓	

Nota: La codificación de registros (7.3.1) es difícil de probar y revisar mediante herramientas dinámicas automatizadas y pruebas de penetración, pero los arquitectos, desarrolladores y revisores de código fuente deben considerarlo un requisito de L1.

V7.4 Control de Errores

El propósito del control de errores es permitir que la aplicación proporcione eventos relevantes para la seguridad para el monitoreo, el triage y la escalada. El propósito no es crear registros. Al registrar eventos relacionados con la seguridad, asegúrese de que hay un propósito para el registro y que se puede distinguir por SIEM o software de análisis.

#	Descripción	L1	L2	L3	CWE
7.4.1	Verifique que se muestra un mensaje genérico cuando se produce un error inesperado o sensible a la seguridad, potencialmente con un identificador único que el personal de soporte técnico puede usar para investigar. (C10)	✓	✓	✓	210
7.4.2	Verifique que el control de excepciones (o un equivalente funcional) se utiliza en todo el código base para tener en cuenta las condiciones de error esperadas e inesperadas. (C10)		✓	✓	544
7.4.3	Verifique que se define un controlador de errores de "último recurso" que detectará todas las excepciones no controladas. (C10)		✓	✓	431

Nota: Ciertos lenguajes, como Swift and Go - y a través de la práctica de diseño común - muchos lenguajes funcionales, no admiten excepciones o controladores de eventos de último recurso. En este caso, los arquitectos y desarrolladores deben usar una forma amigable de patrón, lenguaje o framework para garantizar que las aplicaciones puedan controlar de forma segura eventos excepcionales, inesperados o relacionados con la seguridad.

Referencias

Para obtener más información, véase también:

- OWASP Testing Guide 4.0 content: Testing for Error Handling
- OWASP Authentication Cheat Sheet section about error messages

V8 Protección de Datos

Objetivo de Control

Hay tres elementos clave para una protección de datos sólida: Confidencialidad, Integridad y Disponibilidad (CIA; por sus siglas en inglés). Este estándar supone que la protección de datos se aplica en un sistema de confianza, como un servidor, que se ha reforzado y tiene suficientes protecciones.

Las aplicaciones tienen que asumir que todos los dispositivos de usuario están comprometidos de alguna manera. Cuando una aplicación transmite o almacena información confidencial en dispositivos inseguros, como ordenadores compartidos, teléfonos y tabletas, la aplicación es responsable de garantizar que los datos almacenados en estos dispositivos estén cifrados y no puedan obtenerse, modificarse o divulgarse de forma ilícita.

Asegúrese de que una aplicación verificada cumple los siguientes requisitos de protección de datos de alto nivel:

- Confidencialidad: Los datos deben protegerse de la observación o divulgación no autorizada tanto en tránsito como cuando se almacenan.
- Integridad: Los datos deben protegerse de ser creados, alterados o eliminados maliciosamente por atacantes no autorizados.
- Disponibilidad: los datos deben estar disponibles para los usuarios autorizados según sea necesario.

V8.1 Protección General de Datos

#	Descripción	L1	L2	L3	CWE
8.1.1	Verifique que la aplicación protege los datos confidenciales de la caché en componentes del servidor, como balanceadores de carga y cachés de aplicaciones.		✓	✓	524
8.1.2	Verifique que todas las copias almacenadas en caché o temporales de datos confidenciales almacenados en el servidor están protegidas contra el acceso no autorizado o purgadas/invalidadas después de que el usuario autorizado acceda a los datos confidenciales.		✓	✓	524
8.1.3	Verifique que la aplicación minimiza el número de parámetros de una solicitud, como campos ocultos, variables Ajax, cookies y valores de encabezado.		✓	✓	233
8.1.4	Verifique que la aplicación puede detectar y alertar sobre números anormales de solicitudes, como por IP, usuario, total por hora o día, o lo que tenga sentido para la aplicación.		✓	✓	770
8.1.5	Verifique que se realizan copias de seguridad periódicas de datos importantes y que se realizan pruebas de la restauración de datos.			✓	19
8.1.6	Verifique que las copias de seguridad se almacenan de forma segura para evitar que los datos sean robados o se dañen.			✓	19

V8.2 Protección de datos del lado del cliente

#	Descripción	L1	L2	L3	CWE
8.2.1	Verifique que la aplicación establece suficientes encabezados anti-almacenamiento en caché para que los datos confidenciales no se almacenen en caché en los navegadores modernos.	✓	✓	✓	525

#	Descripción	L1	L2	L3	CWE
8.2.2	Verifique que los datos almacenados en el almacenamiento del navegador (como localStorage, sessionStorage, IndexedDB o cookies) no contengan datos confidenciales.	✓	✓	✓	922
8.2.3	Verifique que los datos autenticados se borran del almacenamiento del cliente, como el DOM del explorador, después de que se termine el cliente o la sesión.	✓	✓	✓	922

V8.3 Datos Privados Confidenciales

Esta sección ayuda a proteger los datos confidenciales de la creación, lectura, actualización o eliminación sin autorización, especialmente en cantidades masivas.

El cumplimiento de esta sección implica el cumplimiento del control de acceso V4 y, en particular, del V4.2. Por ejemplo, para protegerse contra actualizaciones no autorizadas o divulgación de información personal confidencial requiere el cumplimiento de V4.2.1. Por favor, cumpla con esta categoría y V4 para la cobertura completa.

Nota: Las regulaciones y leyes de privacidad, como los Principios de Privacidad de Australia APP-11 o GDPR, afectan directamente a la forma en que las aplicaciones deben abordar la implementación del almacenamiento, el uso y la transmisión de información personal confidencial. Esto va desde sanciones severas hasta simples consejos. Consulte sus leyes y regulaciones locales y consulte a un especialista en privacidad o abogado calificado según sea necesario.

#	Descripción	L1	L2	L3	CWE
8.3.1	Verifique que los datos confidenciales se envían al servidor en el cuerpo o encabezados del mensaje HTTP y que los parámetros de cadena de consulta de cualquier verbo HTTP no contienen datos confidenciales.	✓	✓	✓	319
8.3.2	Verifique que los usuarios tienen un método para eliminar o exportar sus datos sobre demanda (on demand).	✓	✓	✓	212
8.3.3	Verifique que se proporciona a los usuarios un lenguaje claro con respecto a la recopilación y el uso de la información personal suministrada y que los usuarios han proporcionado el consentimiento de aceptación para el uso de esos datos antes de que se utilicen de alguna manera.	✓	✓	✓	285
8.3.4	Verifique que se han identificado todos los datos confidenciales creados y procesados por la aplicación, y asegúrese de que existe una política sobre cómo tratar los datos confidenciales. (C8)	✓	✓	✓	200
8.3.5	Verifique que el acceso a los datos confidenciales se audita (sin registrar los datos confidenciales en sí), si los datos se recopilan en las directivas de protección de datos pertinentes o donde se requiere el registro del acceso.		✓	✓	532
8.3.6	Verifique que la información confidencial contenida en la memoria se sobrescribe tan pronto como ya no sea necesaria para mitigar los ataques de volcado de memoria, utilizando ceros o datos aleatorios.		✓	✓	226
8.3.7	Verifique que la información confidencial o privada que se requiere que se cifre, se cifra mediante algoritmos aprobados que proporcionan confidencialidad e integridad. (C8)		✓	✓	327

#	Descripción	L1	L2	L3	CWE
8.3.8	Verifique que la información personal confidencial está sujeta a la clasificación de retención de datos, de forma que los datos antiguos o desactualizados se eliminen automáticamente, según una programación o según la situación lo requiera.		✓	✓	285

Al considerar la protección de datos, una consideración principal debe ser la extracción masiva o la modificación o el uso excesivo. Por ejemplo, muchos sistemas de redes sociales solo permiten a los usuarios agregar 100 nuevos amigos por día, pero el sistema del que provienen estas solicitudes no es importante. Una plataforma bancaria podría querer bloquear más de 5 transacciones por hora transfiriendo más de 1000 euros de fondos a instituciones externas. Es probable que los requisitos de cada sistema sean muy diferentes, por lo que decidirse por "anormal" debe tener en cuenta el modelo de amenaza y el riesgo empresarial. Los criterios importantes son la capacidad de detectar, disuadir o preferentemente bloquear tales acciones masivas anormales.

Referencias

Para obtener más información, véase también:

- Consider using Security Headers website to check security and anti-caching headers
- OWASP Secure Headers project
- OWASP Privacy Risks Project
- OWASP User Privacy Protection Cheat Sheet
- European Union General Data Protection Regulation (GDPR) overview
- European Union Data Protection Supervisor - Internet Privacy Engineering Network

V9 Comunicación

Objetivo de Control

Asegúrese de que una aplicación verificada cumpla con los siguientes requisitos de alto nivel:

- Requiere TLS o cifrado fuerte, independientemente de la sensibilidad del contenido.
- Siga la guía más reciente, que incluye:
  - Consejos de configuración
  - Algoritmos y cifrados preferidos
- Evite los algoritmos y cifrados débiles o que pronto quedarán obsoletos, excepto como último recurso
- Deshabilite los algoritmos en desuso, o que se sabe son de cifrado inseguro.

Dentro de estos requisitos:

- Manténgase actualizado con los consejos recomendados por la industria sobre la configuración segura de TLS, ya que cambia con frecuencia (a menudo debido a fallas catastróficas en los algoritmos y cifrados existentes).
- Utilice las versiones más recientes de las herramientas de revisión de la configuración de TLS para configurar el orden preferido y la selección de algoritmos.
- Verifique su configuración periódicamente para asegurarse de que la comunicación segura esté siempre presente y sea efectiva.

V9.1 Seguridad de la Comunicación del Cliente

Asegúrese de que todos los mensajes de los clientes se envíen a través de redes cifradas, utilizando TLS 1.2 o posterior. Utilice herramientas actualizadas para revisar la configuración del cliente de forma regular.

#	Descripción	L1	L2	L3	CWE
9.1.1	Verifique que TLS se utilice para toda la conectividad del cliente y que no recurra a comunicaciones inseguras o no cifradas. (C8)	✓	✓	✓	319
9.1.2	Verifique con herramientas de prueba TLS actualizadas que solo estén habilitados los conjuntos de cifrado fuertes, con los conjuntos de cifrado más fuertes configurados como preferidos.	✓	✓	✓	326
9.1.3	Verifique que solo estén habilitadas las últimas versiones recomendadas del protocolo TLS, como TLS 1.2 y TLS 1.3. La última versión del protocolo TLS debería ser la opción preferida.	✓	✓	✓	326

V9.2 Seguridad de la Comunicación del Servidor

Las comunicaciones de servidor son algo más que HTTP. Las conexiones seguras hacia y desde otros sistemas, como sistemas de supervisión, herramientas de administración, acceso remoto y ssh, middleware, bases de datos, mainframes, sistemas de origen externo o de socios, deben estar en su lugar. Todos estos deben ser cifrados para evitar "dureza exterior, trivialmente fácil de interceptar en el interior".

#	Descripción	L1	L2	L3	CWE
9.2.1	Verifique que las conexiones hacia y desde el servidor utilizan certificados TLS de confianza. Cuando se utilizan certificados generados internamente o autofirmados, el servidor debe configurarse para que solo confíe en las CA internas específicas y en los certificados autofirmados específicos. Todos los demás deben ser rechazados.		✓	✓	295

#	Descripción	L1	L2	L3	CWE
9.2.2	Verifique que las comunicaciones cifradas, como TLS, se utilizan para todas las conexiones entrantes y salientes, incluidos los puertos de administración, monitoreo, la autenticación, la API o las llamadas a servicios web, la base de datos, la nube, el serverless, el mainframe, ya sean externos o de conexiones de asociados. El servidor no debe volver a protocolos inseguros o no cifrados.		✓	✓	319
9.2.3	Verifique que se autentican todas las conexiones cifradas a sistemas externos que implican información o funciones confidenciales.		✓	✓	287
9.2.4	Verifique que la adecuada revocación de certificación, como la comprobación de Online Certificate Status Protocol (OCSP), esté habilitada y configurada.		✓	✓	299
9.2.5	Verifique que se hace logging de errores de conexión TLS de back-end.			✓	544

Referencias

Para obtener más información, véase también:

- OWASP – TLS Cheat Sheet
- OWASP - Pinning Guide
- Notas sobre "Modos aprobados de TLS":
  - En el pasado, la ASVS se refería al estándar estadounidense FIPS 140-2, pero como estándar global, aplicar los estándares estadounidenses puede ser difícil, contradictorio o confuso de aplicar.
  - Un mejor método para lograr el cumplimiento de la sección 9.1 sería revisar guías como Mozilla's Server Side TLS o generate known good configurations, y utilice herramientas de evaluación de TLS conocidas y actualizadas para obtener el nivel de seguridad deseado.

V10 Código Malicioso

Objetivo de Control

Asegúrese de que el código cumple los siguientes requisitos de alto nivel:

- La actividad maliciosa se controla de forma segura y adecuada para no afectar al resto de la aplicación.
- No tiene bombas de tiempo u otros ataques basados en el tiempo.
- No permite "llamar a casa" a destinos maliciosos o no autorizados.
- No tiene puertas traceras, huevos de pascua, salami attacks, rootkits o código no autorizado que pueda ser controlado por un atacante.

Encontrar código malicioso es una prueba de lo negativo, que es imposible de validar por completo. Se deben realizar los mejores esfuerzos para asegurarse de que el código fuente no contiene código malicioso o funcionalidades no deseadas.

V10.1 Integridad de Código

La mejor defensa contra el código malintencionado es "confiar, pero verificar". Introducir código no autorizado o malicioso en el código fuente es a menudo un delito en muchas jurisdicciones. Las políticas y procedimientos deben dejar claras las sanciones relativas al código malintencionado.

Los lideres de áreas de desarrolladores deben revisar regularmente las comprobaciones de código, especialmente aquellas que podrían tener acceso a las funciones de tiempo, I/O o funciones de red.

#	Descripción	L1	L2	L3	CWE
10.1.1	Verifique que se está utilizando una herramienta de análisis de código que puede detectar código potencialmente malintencionado, como funciones de tiempo, operaciones de archivos no seguras y conexiones de red.			✓	749

V10.2 Búsqueda de Código Malicioso

El código malicioso es extremadamente raro y es difícil de detectar. La revisión manual línea por línea de código puede ayudar a buscar bombas lógicas, pero incluso el revisor de código más experimentado tendrá problemas para encontrar código malicioso, incluso si supiera que existe.

Cumplir con esta sección no es posible sin acceso completo al código fuente, incluidas las bibliotecas de terceros.

#	Descripción	L1	L2	L3	CWE
10.2.1	Verifique que el código fuente de la aplicación y las bibliotecas de terceros no contienen capacidades no autorizadas de recopilación de datos o de "llamadas a casa". Cuando detecte dicha funcionalidad, obtenga el permiso explicito del usuario para que sea operado así, antes de recopilar cualquier dato.		✓	✓	359
10.2.2	Verifique que la aplicación no solicite permisos innecesarios o excesivos para funciones o sensores relacionados con la privacidad, como contactos, cámaras, micrófonos o ubicación.		✓	✓	272
10.2.3	Verifique que el código fuente de la aplicación y las bibliotecas de terceros no contienen puertas traseras, como cuentas, claves o código ofuscado, blobs binarios no documentados, rootkits o anti-depuración, características de depuración inseguras o de otro modo funcionalidades desactualizadas, inseguras u ocultas que podrían usarse maliciosamente si se descubren.			✓	507

#	Descripción	L1	L2	L3	CWE
10.2.4	Verifique que el código fuente de la aplicación y las bibliotecas de terceros no contienen bombas de tiempo mediante la búsqueda de funciones relacionadas con la fecha y la hora.			✓	511
10.2.5	Verifique que el código fuente de la aplicación y las bibliotecas de terceros no contienen código malintencionado, como salami attacks, logic bypasses o bombas lógicas.			✓	511
10.2.6	Verifique que el código fuente de la aplicación y las bibliotecas de terceros no contienen huevos de pascua ni ninguna otra funcionalidad potencialmente no deseada.			✓	507

V10.3 Integridad de Aplicación

Una vez que se implementa una aplicación, todavía se puede insertar código malintencionado. Las aplicaciones deben protegerse contra ataques comunes, como la ejecución de código sin firmar desde orígenes que no son de confianza y tomas de control de subdominios.

Cumplir con esta categoría, es probable que sea una tarea operativa y continua.

#	Descripción	L1	L2	L3	CWE
10.3.1	Verifique si la aplicación tiene una característica de actualización automática de cliente o servidor, las actualizaciones deben obtenerse a través de canales seguros y firmados digitalmente. El código de actualización debe validar la firma digital de la actualización antes de instalar o ejecutar la actualización.	✓	✓	✓	16
10.3.2	Verifique que la aplicación emplea protecciones de integridad, como la firma de código o la integridad de subrecursos. La aplicación no debe cargar ni ejecutar código de fuentes que no sean de confianza, como la carga de includes, plugins, módulos, código o bibliotecas de fuentes que no sean de confianza o de Internet.	✓	✓	✓	353
10.3.3	Verifique que la aplicación tiene protección contra takeovers de subdominios si la aplicación se basa en entradas DNS o subdominios DNS, como nombres de dominio expirados, punteros DNS obsoletos o CNAME, proyectos expirados en repositorios de código fuente públicos o API de nube transitorias, funciones serverless o buckets de almacenamiento (*autogen-bucket-id*.cloud.example.com) o similares. Las protecciones pueden incluir asegurarse de que los nombres DNS utilizados por las aplicaciones se comprueban regularmente para comprobar su caducidad o cambio.	✓	✓	✓	350

Referencias

Para obtener más información, véase también:

- Hostile Subdomain Takeover, Detectify Labs
- Hijacking of abandoned subdomains part 2, Detectify Labs

V11 Lógica de Negocio

Objetivo de Control

Asegúrese de que una aplicación verificada cumple los siguientes requisitos de alto nivel:

- El flujo de lógica de negocios es secuencial, se procesa en orden y no se puede omitir.
- La lógica empresarial incluye límites para detectar y prevenir ataques automatizados, como transferencias continuas de montos pequeños, o agregar un millón de amigos de uno en uno, etc.
- Los flujos de lógica de negocios de alto valor han considerado casos de abuso y actores malintencionados, y tienen protecciones contra la suplantación, manipulación, divulgación de información y ataques de elevación de privilegios.

V11.1 Seguridad de la Lógica de Negocio

La seguridad de la lógica de negocio es tan individual en todas las aplicaciones, que ningún checklist se puede aplicar. La seguridad de la lógica empresarial debe diseñarse para proteger contra amenazas externas probables: no se puede agregar mediante firewalls de aplicaciones web ni comunicaciones seguras. Recomendamos el uso de modelado de amenazas durante los sprints de diseño, por ejemplo, utilizando la herramienta Cornucopia OWASP o herramientas similares.

#	Descripción	L1	L2	L3	CWE
11.1.1	Verificar que la aplicación solo procesará flujos de la lógica de negocio para el mismo usuario en orden de pasos secuenciales y sin omitir pasos.	✓	✓	✓	841
11.1.2	Verificar que la aplicación solo procesará flujos de lógica de negocios con todos los pasos que se procesan en tiempo humano realista, es decir, las transacciones no se envían demasiado rápido.	✓	✓	✓	799
11.1.3	Verificar que la aplicación tiene límites adecuados para acciones o transacciones de negocio específicas, y que se aplican correctamente con base en los usuarios.	✓	✓	✓	770
11.1.4	Verifique que la aplicación tenga controles anti-automatización para proteger contra llamadas excesivas, como exfiltración masiva de datos, solicitudes de lógica empresarial, carga de archivos o ataques de denegación de servicio.	✓	✓	✓	770
11.1.5	Verificar que la aplicación tiene límites de lógica empresarial o validación para protegerse contra riesgos o amenazas empresariales probables, identificados mediante el modelado de amenazas o metodologías similares.	✓	✓	✓	841
11.1.6	Verifique que la aplicación no tenga problemas de "Time Of Check to Time Of Use" (TOCTOU) u otras race conditions para operaciones sensibles.		✓	✓	367
11.1.7	Verificar que la aplicación supervisa eventos o actividades inusuales desde una perspectiva de lógica de negocios. Por ejemplo, los intentos de realizar acciones fuera de servicio o acciones que un usuario normal nunca intentaría. (C9)		✓	✓	754
11.1.8	Verificar que la aplicación tiene alertas configurables cuando se detectan ataques automatizados o actividad inusual.		✓	✓	390

Referencias

Para obtener más información, véase también:

- OWASP Web Security Testing Guide 4.1: Business Logic Testing

- Anti-automation can be achieved in many ways, including the use of OWASP AppSensor and OWASP Automated Threats to Web Applications
- OWASP AppSensor can also help with Attack Detection and Response.
- OWASP Cornucopia

V12 Archivos y Recursos

Objetivo de Control

Asegúrese de que una aplicación verificada cumple los siguientes requisitos de alto nivel:

- Los datos de archivo que no son de confianza deben manejarse en consecuencia de una manera segura.
- Los datos de archivos que no son de confianza obtenidos de fuentes no confiables se almacenan fuera de la raíz web y con permisos limitados.

V12.1 Carga de Archivos

Aunque las bombas zip son eminentemente comprobables utilizando técnicas de prueba de penetración, se consideran L2 y superiores para fomentar la consideración de diseño y desarrollo con pruebas manuales cuidadosas, y para evitar pruebas de penetración automatizadas o del tipo manual pero no calificadas de una condición de denegación de servicio.

#	Descripción	L1	L2	L3	CWE
12.1.1	Verifique que la aplicación no aceptará archivos grandes que puedan llenar el almacenamiento o provocar una denegación de servicio.	✓	✓	✓	400
12.1.2	Verifique que la aplicación compruebe los archivos comprimidos (p. ej. zip, gz, docx, odt) contra el tamaño máximo sin comprimir permitido y con el número máximo de archivos antes de descomprimir el archivo.		✓	✓	409
12.1.3	Verifique que se aplica una cuota de tamaño de archivo y un número máximo de archivos por usuario para asegurarse de que un solo usuario no puede llenar el almacenamiento con demasiados archivos o archivos excesivamente grandes.		✓	✓	770

V12.2 Integridad de Archivos

#	Descripción	L1	L2	L3	CWE
12.2.1	Verifique que los archivos obtenidos de orígenes que no son de confianza se validan para que sean del tipo esperado en función del contenido del archivo.		✓	✓	434

V12.3 Ejecución de Archivos

#	Descripción	L1	L2	L3	CWE
12.3.1	Verifique que los metadatos del nombre de archivo enviados por el usuario no se utilizan directamente por los sistemas de archivos del sistema o del marco de trabajo y que se utiliza una API de dirección URL para proteger contra el recorrido de ruta de acceso.	✓	✓	✓	22
12.3.2	Verifique que los metadatos del nombre de archivo enviados por el usuario se validan o ignoran para evitar la divulgación, creación, actualización o eliminación de archivos locales (LFI).	✓	✓	✓	73
12.3.3	Verifique que los metadatos del nombre de archivo enviados por el usuario se validan o omiten para evitar la divulgación o ejecución de archivos remotos a través de ataques de inclusión remota de archivos (RFI) o falsificación de solicitudes del lado del servidor (SSRF).	✓	✓	✓	98

#	Descripción	L1	L2	L3	CWE
12.3.4	Verifique que la aplicación protege contra la descarga de archivos reflectantes (RFD) validando o ignorando los nombres de archivo enviados por el usuario en un parámetro JSON, JSONP o URL, el encabezado Content-Type de respuesta debe establecerse en text/plain y el encabezado Content-Disposition debe tener un nombre de archivo fijo.	✓	✓	✓	641
12.3.5	Verifique que los metadatos de archivos que no son de confianza no se utilizan directamente con la API del sistema o las bibliotecas, para proteger contra la inyección de comandos del sistema operativo.	✓	✓	✓	78
12.3.6	Verifique que la aplicación no incluye ni ejecuta funcionalidad desde orígenes que no son de confianza, como redes de distribución de contenido no verificadas, bibliotecas de JavaScript, bibliotecas node npm o archivos DLL server-side.		✓	✓	829

V12.4 Almacenamiento de Archivos

#	Descripción	L1	L2	L3	CWE
12.4.1	Verifique que los archivos obtenidos de fuentes no confiables se almacenen fuera de la raíz web, con permisos limitados.	✓	✓	✓	552
12.4.2	Verifique que los escáneres antivirus analicen los archivos obtenidos de fuentes no confiables para evitar la carga y el servicio de contenido malicioso conocido.	✓	✓	✓	509

V12.5 Descarga de Archivos

#	Descripción	L1	L2	L3	CWE
12.5.1	Verifique que la capa web está configurado para transmitir solo archivos con extensiones específicas, para evitar la filtración accidental de información o código fuente. Por ejemplo, los archivos de copia de seguridad (p. ej. .bak), los archivos de trabajo temporales (p. ej. .swp), los archivos comprimidos (.zip, .tar.gz, etc.) y otras extensiones utilizadas comúnmente por los editores deben bloquearse a menos que sea necesario.	✓	✓	✓	552
12.5.2	Verifique que las solicitudes directas a los archivos cargados nunca se ejecutarán como contenido HTML/JavaScript.	✓	✓	✓	434

V12.6 Protección SSRF

#	Descripción	L1	L2	L3	CWE
12.6.1	Verifique que el servidor web o de aplicaciones está configurado con una lista de permisos de recursos o sistemas a los que el servidor puede enviar solicitudes o cargar datos o archivos.	✓	✓	✓	918

Referencias

Para obtener más información, véase también:

- File Extension Handling for Sensitive Information
- Reflective file download by Oren Hafif
- OWASP Third Party JavaScript Management Cheat Sheet

V13 API y Servicios Web

Objetivo de Control

Asegúrese de que una aplicación verificada que utiliza APIs de capa de servicio de confianza (normalmente mediante JSON o XML o GraphQL) tiene:

- Autenticación adecuada, gestión de sesiones y autorización de todos los servicios web.
- Validación de entrada de todos los parámetros que transitan de un nivel de confianza inferior a superior.
- Controles de seguridad eficaces para todos los tipos de API, incluida la nube y los Serverless API

Lea este capítulo en combinación con todos los demás capítulos de este mismo nivel; ya no duplicamos los problemas de autenticación o administración de sesiones de API.

V13.1 Seguridad Genérica de Servicios Web

#	Descripción	L1	L2	L3	CWE
13.1.1	Verifique que todos los componentes de la aplicación utilizan las mismas codificaciones y analizadores para evitar el análisis de ataques que explotan un comportamiento de análisis de archivos o URI diferente que se podría usar en ataques SSRF y RFI.	✓	✓	✓	116
13.1.2	[DELETED, DUPLICATE OF 4.3.1]				
13.1.3	Verifique que las direcciones URL de la API no exponen información confidencial, como API keys, los tokens de sesión, etc.	✓	✓	✓	598
13.1.4	Verifique que las decisiones de autorización se toman en el URI, se aplican mediante seguridad programática o declarativa en el controlador o enrutador, y en el nivel de recursos, se aplican mediante permisos basados en modelos.		✓	✓	285
13.1.5	Verifique que las solicitudes que contienen tipos de contenido inesperados o contenido que falta se rechazan con encabezados adecuados (estado de respuesta HTTP 406 Inaceptable o 415 Tipo de medio no compatible).		✓	✓	434

V13.2 Servicio Web RESTful

La validación del esquema JSON se encuentra en una etapa preliminar de estandarización (ver referencias). Cuando considere usar la validación del esquema JSON, que es la mejor práctica para los servicios web RESTful, considere usar estas estrategias de validación de datos adicionales en combinación con la validación del esquema JSON:

- Análisis de la validación del objeto JSON, ya sea que faltan elementos o hay elementos adicionales.
- Validación de los valores de objeto JSON mediante métodos de validación de entrada estándar, como el tipo de datos, el formato de datos, la longitud, etc.
- y la validación formal del esquema JSON.

Una vez que se formalice el estándar de validación de esquemas JSON, ASVS actualizará sus consejos en esta área. Supervise cuidadosamente las bibliotecas de validación de esquemas JSON en uso, ya que tendrán que actualizarse periódicamente hasta que se formalice el estándar y se eliminan los errores de las implementaciones de referencia.

#	Descripción	L1	L2	L3	CWE
13.2.1	Verifique que los métodos HTTP RESTful habilitados son una opción válida para el usuario o la acción, como impedir que los usuarios normales usen DELETE o PUT en recursos o API protegidos.	✓	✓	✓	650
13.2.2	Verifique que la validación del esquema JSON está en su lugar y se comprueba antes de aceptar la entrada.	✓	✓	✓	20
13.2.3	Verifique que los servicios web RESTful que utilizan cookies están protegidos contra la falsificación de solicitudes entre sitios, mediante el uso de una o más de las siguientes formas: patrón de cookies de doble envío, "nonces" CSRF o comprobaciones de encabezado de solicitud de origen.	✓	✓	✓	352
13.2.4	[DELETED, DUPLICATE OF 11.1.4]				
13.2.5	Verifique que los servicios REST comprueben explícitamente que el tipo de contenido entrante sea el esperado, como application/xml o application/json.		✓	✓	436
13.2.6	Verifique que los encabezados de mensaje y la carga útil (payload) son confiables y no se modifican en tránsito. Requerir un cifrado seguro para el transporte (solo TLS) puede ser suficiente en muchos casos, ya que proporciona confidencialidad y protección de integridad. Las firmas digitales por cada mensaje pueden proporcionar una garantía adicional sobre las protecciones de transporte para aplicaciones de alta seguridad, pero conllevan complejidad y riesgos adicionales para compensar los beneficios.		✓	✓	345

V13.3 Servicio Web SOAP

#	Descripción	L1	L2	L3	CWE
13.3.1	Verifique que la validación del esquema XSD tiene lugar para garantizar un documento XML formado correctamente, seguido de la validación de cada campo de entrada antes de que se realice cualquier procesamiento de esos datos.	✓	✓	✓	20
13.3.2	Verifique que el payload del mensaje está firmada mediante WS-Security para garantizar un transporte fiable entre el cliente y el servicio.		✓	✓	345

Nota: Debido a los problemas con los ataques XXE contra los DTD, la validación DTD no se debe utilizar, y debe deshabilitar la evaluación DTD del framework, según los requisitos establecidos en la configuración V14.

V13.4 GraphQL

#	Descripción	L1	L2	L3	CWE
13.4.1	Verifique que se utiliza una lista de permisos de consulta o una combinación de limitación de profundidad y limitación de cantidad para evitar que GraphQL o la expresión de la capa de datos provoque una denegación de servicio (DoS) como resultado de costosas consultas anidadas. Para escenarios más avanzados, se debe usar el análisis de costos de consulta.		✓	✓	770
13.4.2	Verifique que GraphQL u otra lógica de autorización de capa de datos podría implementarse en la capa de lógica de negocio en lugar de la capa GraphQL.		✓	✓	285

Referencias

Para obtener más información, véase también:

- OWASP Serverless Top 10
- OWASP Serverless Project
- OWASP Testing Guide 4.0: Configuration and Deployment Management Testing
- OWASP Cross-Site Request Forgery cheat sheet
- OWASP XML External Entity Prevention Cheat Sheet - General Guidance
- JSON Web Tokens (and Signing)
- REST Security Cheat Sheet
- JSON Schema
- XML DTD Entity Attacks
- Orange Tsai - A new era of SSRF Exploiting URL Parser In Trending Programming Languages

V14 Configuración

Objetivo de Control

Asegúrese de que una aplicación verificada tiene:

- Un entorno de compilación seguro, repetible y automatizable.
- La aplicación no incluye la biblioteca de terceros reforzada, la dependencia y la administración de la configuración, de modo que la aplicación no incluya componentes obsoletos o no seguros.

La configuración de la aplicación "desde fábrica" debe ser segura para estar en Internet, lo que significa una configuración segura desde la caja.

V14.1 Compilación y Despliegue

Las pipelines de compilación son la base para la seguridad repetible - cada vez que se detecta algo inseguro, se puede resolver en el código fuente, compilar o desplegar scripts y probarse automáticamente. Enfatizamos el uso de pipelines de compilación con comprobaciones automáticas de seguridad y chequeo de dependencia que alerten o interrumpen la compilación para evitar que los problemas de seguridad conocidos se implementen en producción. Los pasos manuales realizados de forma irregular directamente conducen a errores de seguridad evitables.

A medida que la industria se encamina hacia un modelo DevSecOps, es importante garantizar la disponibilidad continua y la integridad de la implementación y la configuración para lograr un estado "bueno conocido". En el pasado, si un sistema fuera hackeado, tardaría días o meses en demostrar que no se habían producido más intrusiones. Hoy en día, con la llegada de la infraestructura definida por software, las implementaciones rápidas de A/B con zero downtime y las compilaciones automatizadas en contenedores, es posible compilar, hardenizar y desplegar continuamente un reemplazo "bueno conocido" para cualquier sistema comprometido.

Si aún persisten modelos tradicionales, se deben tomar medidas manuales para hardenizar y hacer una copia de seguridad de esa configuración para permitir que los sistemas comprometidos se reemplacen rápidamente por sistemas de alta integridad y sin compromisos, de manera oportuna.

El cumplimiento de esta categoria ASVS requiere un sistema de compilación automatizado y acceso a scripts de compilación e implementación.

#	Descripción	L1	L2	L3	CWE
14.1.1	Verifique que los procesos de compilación y despliegue de aplicaciones se realizan de forma segura y repetible, como la automatización de CI/CD, la administración de configuración automatizada y los scripts de despliegue automatizado.		✓	✓	
14.1.2	Verifique que los indicadores del compilador están configurados para habilitar todas las protecciones y advertencias de desbordamiento de búfer disponibles, incluida la aleatorización de la pila, la prevención de la ejecución de datos y para interrumpir la compilación si se encuentra un puntero no seguro, memoria, cadena de formato, entero u operaciones de cadena.		✓	✓	120
14.1.3	Verifique que la configuración del servidor está hardenizada según las recomendaciones del servidor de aplicaciones y los framewors en uso.		✓	✓	16
14.1.4	Verifique que la aplicación, la configuración y todas las dependencias se pueden volver a implementar mediante scripts de implementación automatizada, crearse a partir de un runbook documentado y probado en un tiempo razonable o restaurarse a partir de copias de seguridad de forma oportuna.		✓	✓	

#	Descripción	L1	L2	L3	CWE
14.1.5	Verifique que los administradores autorizados pueden verificar la integridad de todas las configuraciones relevantes para la seguridad para detectar una posible manipulación.			✓	

V14.2 Dependencias

La administración de dependencias es fundamental para el funcionamiento seguro de cualquier aplicación de cualquier tipo. No mantenerse al día con dependencias obsoletas o inseguras es la causa raíz de los ataques más grandes y costosos hasta la fecha.

Nota: En el nivel 1, el cumplimiento 14.2.1 se relaciona con observaciones o detecciones de bibliotecas y componentes del lado cliente y otros, en lugar del análisis de código estático en tiempo de compilación o análisis de dependencia más preciso. Estas técnicas más precisas podrían ser detectables por entrevistas, según sea necesario.

#	Descripción	L1	L2	L3	CWE
14.2.1	Verifique que todos los componentes estén actualizados, preferiblemente utilizando un comprobador de dependencias durante el tiempo de compilación. (C2)	✓	✓	✓	1026
14.2.2	Verifique que se eliminen todas las funciones, documentación, aplicaciones de muestra y configuraciones innecesarias.	✓	✓	✓	1002
14.2.3	Verifique que si los activos de la aplicación, como bibliotecas JavaScript, fuentes CSS o web, se hospedan externamente en una red de entrega de contenido (CDN) o un proveedor externo, se usa la integridad de subrecursos (SRI) para validar la integridad del activo.	✓	✓	✓	829
14.2.4	Verifique que los componentes de terceros provienen de repositorios predefinidos, de confianza y mantenidos continuamente. (C2)		✓	✓	829
14.2.5	Verifique que se mantenga una Lista de materiales de software (SBOM; por sus siglas en inglés) de todas las bibliotecas de terceros en uso. (C2)		✓	✓	
14.2.6	Verifique que la superficie de ataque se reduce mediante sandboxing o encapsular bibliotecas de terceros para exponer solo el comportamiento necesario en la aplicación. (C2)		✓	✓	265

V14.3 Divulgación de Seguridad Involuntaria

Las configuraciones para la producción deben endurecerse contra ataques comunes, como consolas de depuración, elevar el nivel de defensa contra los ataques de secuencias de comandos entre sitios (XSS) e inclusión remota de archivos (RFI) y eliminar las "vulnerabilidades" de detección de información trivial que son el sello no deseado de muchos informes de pruebas de penetración. Muchos de estos problemas rara vez se clasifican como un riesgo significativo, pero se encadenan junto con otras vulnerabilidades. Si estos problemas no están presentes de forma predeterminada, se debe elevar el nivel de defensa antes de que la mayoría de los ataques puedan realizarse exitosamente.

#	Descripción	L1	L2	L3	CWE
14.3.1	[ELIMINADO, DUPLICADO DE 7.4.1]				
14.3.2	Verifique que los modos de depuración del servidor web o de aplicaciones y del framework de aplicaciones están deshabilitados en producción para eliminar las características de depuración, las consolas de desarrollador y las divulgaciones de seguridad no deseadas.	✓	✓	✓	497

#	Descripción	L1	L2	L3	CWE
14.3.3	Verifique que los encabezados HTTP o cualquier parte de la respuesta HTTP no exponen información detallada de la versión de los componentes del sistema.	✓	✓	✓	200

V14.4 Encabezados de Seguridad HTTP

#	Descripción	L1	L2	L3	CWE
14.4.1	Verifique que cada respuesta HTTP contenga un encabezado de tipo de contenido. También especifique un conjunto de caracteres seguro (p. ej., UTF-8, ISO-8859-1) si los tipos de contenido son texto/*, /+xml y aplicación/xml. El contenido debe coincidir con el encabezado de tipo de contenido proporcionado.	✓	✓	✓	173
14.4.2	Verifique que todas las respuestas de API contienen un encabezado Content-Disposition: attachment; filename="api.json" (u otro nombre de archivo apropiado para el tipo de contenido).	✓	✓	✓	116
14.4.3	Verifique que existe un encabezado de respuesta de Directiva de Seguridad de Contenido (CSP) que ayuda a mitigar el impacto de los ataques XSS como vulnerabilidades de inyección de HTML, DOM, JSON y JavaScript.	✓	✓	✓	1021
14.4.4	Verifique que todas las respuestas contienen un encabezado X-Content-Type-Options: nosniff.	✓	✓	✓	116
14.4.5	Verifique que se incluye un encabezado Strict-Transport-Security en todas las respuestas y para todos los subdominios, como Strict-Transport-Security: max-age-15724800; includeSubdomains.	✓	✓	✓	523
14.4.6	Verifique que se incluya adecuadamente un encabezado de Referrer-Policy para evitar exponer información confidencial en la URL a través del encabezado de referencia a partes que no son de confianza.	✓	✓	✓	116
14.4.7	Verifique que el contenido de una aplicación web no se puede incrustar en un sitio de terceros de forma predeterminada y que la inserción de los recursos exactos solo se permite cuando sea necesario mediante el uso adecuado de Content-Security-Polic: frame-ancestors y encabezados de respuesta X-Frame-Options.	✓	✓	✓	1021

V14.5 Validación de Encabezado de Solicitud HTTP

#	Descripción	L1	L2	L3	CWE
14.5.1	Verifique que el servidor de aplicaciones solo acepta los métodos HTTP que utiliza la aplicación/API, incluidas las pre-flight OPTIONS, y los Logs/alertas en cualquier solicitud que no sea válida para el contexto de la aplicación.	✓	✓	✓	749
14.5.2	Verifique que el encabezado Origin proporcionado no se utiliza para las decisiones de autenticación o control de acceso, ya que un atacante puede cambiar fácilmente el encabezado Origin.	✓	✓	✓	346
14.5.3	Verifique que el encabezado Cross-Origin Resource Sharing (CORS) Access-Control-Allow-Origin utiliza una estricta lista de permisos de dominios y subdominios de confianza para que coincidan entre si, y no se permita el origen "nulo".	✓	✓	✓	346

#	Descripción	L1	L2	L3	CWE
14.5.4	Verifique que la aplicación autentica los encabezados HTTP agregados por un proxy de confianza o dispositivos SSO, como un token de portador.		✓	✓	306

Referencias

Para obtener más información, véase también:

- OWASP Web Security Testing Guide 4.1: Testing for HTTP Verb Tampering
- Adding Content-Disposition to API responses helps prevent many attacks based on misunderstanding on the MIME type between client and server, and the "filename" option specifically helps prevent Reflected File Download attacks.
- Content Security Policy Cheat Sheet
- Exploiting CORS misconfiguration for BitCoins and Bounties
- OWASP Web Security Testing Guide 4.1: Configuration and Deployment Management Testing
- Sandboxing third party components

Apéndice A: Glosario

- **Randomnización del diseño del espacio de direcciones** (ASLR) - Una técnica para dificultar la explotación de errores de corrupción de memoria.
- **Lista de permitidos** - Una lista de datos u operaciones permitidas, por ejemplo, una lista de caracteres que pueden realizar la validación de entrada.
- **Seguridad de la aplicación** - La seguridad a nivel de aplicación se centra en el análisis de los componentes que componen la capa de aplicación del modelo de referencia de interconexión de sistemas abiertos (modelo OSI), en lugar de centrarse, por ejemplo, en el sistema operativo subyacente o las redes conectadas.
- **Verificación de seguridad de aplicaciones** - La evaluación técnica de una solicitud contra el ASVS de OWASP.
- **Informe de Verificación de Seguridad en la aplicación** - Un informe que documenta los resultados generales y el análisis de soporte generado por el verificador para una aplicación determinada.
- **Autenticación** - La verificación de la identidad reclamada de un usuario de la aplicación.
- **Verificación Automatizada** - El uso de herramientas automatizadas (ya sea herramientas de análisis dinámico, herramientas de análisis estático o ambas) que utilizan firmas de vulnerabilidad para encontrar problemas.
- **Prueba de Caja Negra** - Es un método de prueba de software que examina la funcionalidad de una aplicación sin mirar sus estructuras internas o su funcionamiento.
- **Componente** - Una unidad de código autónoma, con interfaces de disco y de red asociadas que se comunica con otros componentes.
- **Cross-Site Scripting** (XSS) - Una vulnerabilidad de seguridad que se encuentra típicamente en las aplicaciones web y que permite la inyección de scripts del lado del cliente en el contenido.
- **Módulo criptográfico** - Hardware, software y/o firmware que implementa algoritmos criptográficos y/o genera claves criptográficas.
- **Enumeración de debilidades comunes** (CWE) - Lista desarrollada por la comunidad, con las debilidades comunes de seguridad de software. Sirve como un lenguaje común, una vara de medir para las herramientas de seguridad de software, y como una línea base para la identificación de debilidades, la mitigación y los esfuerzos de prevención.
- **Verificación del diseño** - La evaluación técnica de la arquitectura de seguridad de una aplicación.
- **Prueba de seguridad de aplicaciones dinámicas** (DAST) - Las tecnologías están diseñadas para detectar las condiciones indicativas de una vulnerabilidad de seguridad en una aplicación en su estado de ejecución.
- **Verificación dinámica** - El uso de herramientas automatizadas que utilizan firmas de vulnerabilidad para encontrar problemas durante la ejecución de una aplicación.
- **Fast IDentity Online** (FIDO) - Conjunto de normas de autenticación que permiten utilizar una variedad de métodos de autenticación diferentes, como la biometría, los módulos de plataformas de confianza (TPM), los tokens de seguridad USB, etc.
- **Identificador globalmente único** (GUID) - Un número de referencia único utilizado como identificador en el software.
- **Protocolo de transferencia de hipertexto** (HTTPS) - Un protocolo de aplicación para sistemas de información hipermedia distribuidos y en colaboración. Es la base de la comunicación de datos para la World Wide Web.
- **Claves en código duro** - Claves criptográficas que se almacenan en el sistema de archivos, ya sea en código, comentarios o archivos.

- **Módulo de seguridad de hardware** (HSM) - Componente de hardware que puede almacenar claves criptográficas y otros secretos de forma protegida.
- **Hibernate Query Language** (HQL) - Lenguaje de consulta de apariencia similar al SQL utilizado por la biblioteca de Hibernate ORM.
- **Validación de entrada** - La canonicalización y validación de entradas de usuario que no son de confianza.
- **Código malicioso** - Código introducido en una aplicación durante su desarrollo sin saberlo el propietario de la aplicación, que elude la directiva de seguridad prevista de la aplicación. No es lo mismo que un malware, asi como un virus es diferente de un gusano!
- **Malware** - Código ejecutable que se introduce en una aplicación durante el tiempo de ejecución sin el conocimiento del usuario o administrador de la aplicación.
- **Open Web Application Security Project** (OWASP) - La fundación OWASP es una comunidad abierta y libre en todo el mundo centrada en mejorar la seguridad de las aplicaciones de software. Nuestra misión es hacer que la seguridad de las aplicaciones sea "visible", para que las personas y las organizaciones puedan tomar decisiones informadas sobre los riesgos de seguridad de las aplicaciones. Ver: https://www.owasp.org/
- **Contraseña de un solo uso** (OTP) - Una contraseña que se genera de manera única para ser usada en una sola ocasión.
- **Mapeo Objeto-Relación** (ORM) - Un sistema utilizado para permitir que una base de datos relacional/basada en tablas sea referenciada y consultada dentro de un programa de aplicación usando un modelo de objeto compatible con la aplicación.
- **Password-Based Key Derivation Function 2** (PBKDF2) - Algoritmo especial unidireccional utilizado para crear una clave criptográfica robusta a partir de un texto de entrada (como una contraseña) y un valor de “salto” aleatorio adicional y que, por lo tanto, puede utilizarse para dificultar el descifrado de una contraseña fuera de línea si el valor resultante se almacena en lugar de la contraseña original.
- **Información de identificación personal** (PII) - Es la información que puede utilizarse por sí sola o junto con otra información para identificar, contactar o localizar a una sola persona, o para identificar a un individuo en un contexto.
- **Position-independent executable** (PIE) - Un cuerpo de código máquina que, al ser colocado en algún lugar de la memoria primaria, se ejecuta correctamente independientemente de su dirección absoluta.
- **Infraestructura de Clave Pública** (PKI) - Un arreglo que vincula las claves públicas con las respectivas identidades de las entidades. La vinculación se establece mediante un proceso de registro y emisión de certificados en y por una autoridad de certificación (CA).
- **Red telefónica pública conmutada** (PSTN) - La red telefónica tradicional que incluye tanto teléfonos de línea fija como móviles.
- **Relying Party** (RP) - Generalmente una aplicación que se basa en que un usuario se ha autenticado contra un proveedor de autenticación independiente. La aplicación se basa en algún tipo de token o conjunto de afirmaciones firmadas proporcionadas por ese proveedor de autenticación para confiar en que el usuario es quien dice ser.
- **Pruebas de seguridad de aplicaciones estáticas** (SAST) - Un conjunto de tecnologías diseñadas para analizar el código fuente de la aplicación, el código de bytes y los binarios para la codificación y las condiciones de diseño que son indicativos de vulnerabilidades de seguridad. Las soluciones SAST analizan una aplicación desde "adentro hacia afuera" en un estado de "no ejecución".
- **Ciclo de vida del desarrollo de software** (SDLC) - El proceso paso a paso por el cual el software es desarrollado yendo desde los requerimientos iniciales hasta el despliegue y mantenimiento.
- **Arquitectura de seguridad** - Una abstracción del diseño de una aplicación que identifica y describe dónde y cómo se utilizan los controles de seguridad, y también identifica y describe la ubicación y la sensibilidad de los datos tanto del usuario como de la aplicación.

- **Configuración de seguridad** - La configuración en tiempo de ejecución de una aplicación que afecta a la forma en que se utilizan los controles de seguridad.
- **Control de seguridad** - Una función o componente que realiza una comprobación de seguridad (por ejemplo, una comprobación de control de acceso) o cuando una ejecución resulta en un efecto de seguridad (por ejemplo, la generación de un registro de auditoría).
- **Falsificación de Peticiones del Lado del Servidor** (SSRF) - Un ataque que abusa de la funcionalidad del servidor para leer o actualizar los recursos internos suministrando o modificando una URL que el código que se ejecuta en el servidor leerá o a la que enviará datos.
- **Autenticación de inicio de sesión único** (SSO) - Esto ocurre cuando un usuario se conecta a una aplicación y luego se conecta automáticamente a otras aplicaciones sin tener que volver a autenticarse. Por ejemplo, cuando se inicia la sesión en Google, al acceder a otros servicios de Google como YouTube, Google Docs y Gmail se iniciará la sesión automáticamente.
- **Inyección SQL** (SQLi) - Técnica de inyección de código utilizada para atacar aplicaciones basadas en datos, en la que se insertan sentencias SQL maliciosas en un punto de entrada.
- **SVG** - Gráficos vectoriales escalables
- **OTP basado en el tiempo** (TOTP) - Método de generación de un OTP en el que el tiempo actual actúa como parte del algoritmo para generar la contraseña.
- **Modelado de amenazas** - Técnica que consiste en desarrollar arquitecturas de seguridad cada vez más refinadas para identificar agentes de amenazas, zonas de seguridad, controles de seguridad y activos técnicos y comerciales importantes.
- **Seguridad de la capa de transporte** (TLS) - Protocolos criptográficos que proporcionan seguridad a las comunicaciones a través de una conexión de red.
- **Módulo de Plataforma de Confianza** (TPM) - Un tipo de HSM que suele estar conectado a un componente de hardware más grande como una placa madre y actúa como la "raíz de la confianza" para ese sistema.
- **Autenticación de dos factores** (2FA) - Esto agrega un segundo nivel de autenticación al inicio de un login en una cuenta.
- **2do Factor Universal** (U2F) - Una de las normas creadas por FIDO específicamente para permitir el uso de una llave de seguridad USB o NFC como 2do factor de autenticación.
- **Fragmentos de URI/URL/URL** - Un Identificador Uniforme de Recursos es una cadena de caracteres utilizada para identificar un nombre o un recurso web. Un Localizador Uniforme de Recursos se utiliza a menudo como referencia a un recurso.
- **Verificador** - La persona o equipo que está revisando una aplicación contra los requerimientos de OWASP ASVS.
- **Lo que ves es lo que obtienes** (WYSIWYG) - Un tipo de editor de contenido enriquecido que muestra cómo se verá realmente el contenido cuando se renderice en lugar de mostrar la codificación usada para gobernar la renderización.
- **Certificado X.509** - Un certificado X.509 es un certificado digital que utiliza la norma internacional de infraestructura de clave pública (PKI) X.509, ampliamente aceptada, para verificar que una clave pública pertenece a la identidad del usuario, la computadora o el servicio contenido en el certificado.
- **Entidad eXterna XML** (XXE) - Un tipo de entidad XML que puede acceder al contenido local o remoto mediante un identificador de sistema declarado. Esto puede cargarse a varios ataques de inyección.

Apéndice B: Referencias

Es más probable que los siguientes proyectos de OWASP sean útiles para los usuarios/adoptantes de esta norma:

Principales Proyectos OWASP

1. OWASP Top 10 Project: https://owasp.org/www-project-top-ten/
2. OWASP Web Security Testing Guide: https://owasp.org/www-project-web-security-testing-guide/
3. OWASP Proactive Controls: https://owasp.org/www-project-proactive-controls/
4. OWASP Security Knowledge Framework: https://owasp.org/www-project-security-knowledge-framework/
5. OWASP Software Assurance Maturity Model (SAMM): https://owasp.org/www-project-samm/

OWASP Cheat Sheet Series project

Este proyecto tiene una serie de hojas de trucos que serán relevantes para diferentes temas en el ASVS.

Hay un mapping entre cheat sheet y ASVS, que se puede encontrar aquí : https://cheatsheetseries.owasp.org/cheatsheets/IndexASVS.html

Proyectos relacionados con la seguridad de móviles

1. OWASP Mobile Security Project: https://owasp.org/www-project-mobile-security/
2. OWASP Mobile Top 10 Risks: https://owasp.org/www-project-mobile-top-10/
3. OWASP Mobile Security Testing Guide and Mobile Application Security Verification Standard: https://owasp.org/www-project-mobile-security-testing-guide/

Proyectos de OWASP relacionados con el Internet de las cosas

1. OWASP Internet of Things Project: https://owasp.org/www-project-internet-of-things/

Proyectos OWASP Serverless

1. OWASP Serverless Project: https://owasp.org/www-project-serverless-top-10/

Otros

Del mismo modo, es más probable que los siguientes sitios web sean útiles para los usuarios/adoptantes de este estándar

1. SecLists Github: https://github.com/danielmiessler/SecLists
2. MITRE Common Weakness Enumeration: https://cwe.mitre.org/
3. PCI Security Standards Council: https://www.pcisecuritystandards.org
4. PCI Data Security Standard (DSS) v3.2.1 Requirements and Security Assessment Procedures: https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1.pdf
5. PCI Software Security Framework - Secure Software Requirements and Assessment Procedures: https://www.pcisecuritystandards.org/documents/PCI-Secure-Software-Standard-v1_0.pdf
6. PCI Secure Software Lifecycle (Secure SLC) Requirements and Assessment Procedures: https://www.pcisecuritystandards.org/documents/PCI-Secure-SLC-Standard-v1_0.pdf

Apéndice C: Requisitos de verificación de Internet de las cosas

Esta capítulo estaba originalmente en la rama principal, pero con el trabajo que el equipo de IoT de OWASP ha realizado, no tiene sentido mantener dos subprocesos diferentes en el tema. Para la versión 4.0, estamos trasladando esto al Apéndice, e instamos a todos los que lo requieran, a utilizar más bien el proyecto principal de OWASP IoT project

Objetivo de Control

Los dispositivos integrados/IoT deben:

- Tener el mismo nivel de controles de seguridad dentro del dispositivo que se encuentra en el servidor, aplicando controles de seguridad en un entorno de confianza.
- Los datos confidenciales almacenados en el dispositivo deben realizarse de forma segura mediante el almacenamiento respaldado por hardware, como elementos seguros.
- Todos los datos confidenciales transmitidos desde el dispositivo deben utilizar la seguridad de la capa de transporte.

Requisitos de verificación de seguridad

#	Description	L1	L2	L3	Desde
C.1	Verifique que las interfaces de depuración de capa de aplicación, como USB, UART y otras variantes seriales, estén deshabilitadas o protegidas por una contraseña compleja.	✓	✓	✓	4.0
C.2	Verifique que las claves criptográficas y los certificados son únicos para cada dispositivo individual.	✓	✓	✓	4.0
C.3	Verifique que los controles de protección de memoria como ASLR y DEP están habilitados por el sistema operativo integrado/IoT, si procede.	✓	✓	✓	4.0
C.4	Verifique que las interfaces de depuración en chip como JTAG o SWD estén deshabilitadas o que el mecanismo de protección disponible esté habilitado y configurado adecuadamente.	✓	✓	✓	4.0
C.5	Verifique que la ejecución de confianza está implementada y habilitada, si está disponible en el SoC o CPU del dispositivo.	✓	✓	✓	4.0
C.6	Verifique que los datos confidenciales, las claves privadas y los certificados se almacenan de forma segura en un elemento seguro, TPM, TEE (Trusted Execution Environment) o se protegen mediante criptografía segura.	✓	✓	✓	4.0
C.7	Verifique que las aplicaciones de firmware protegen los datos en tránsito mediante la seguridad de la capa de transporte.	✓	✓	✓	4.0
C.8	Verifique que las aplicaciones de firmware validan la firma digital de las conexiones de servidor.	✓	✓	✓	4.0
C.9	Verifique que las comunicaciones inalámbricas se autentiquen mutuamente.	✓	✓	✓	4.0
C.10	Verifique que las comunicaciones inalámbricas se envíen a través de un canal cifrado.	✓	✓	✓	4.0
C.11	Verifique que cualquier uso de funciones C prohibidas se sustituye por las funciones equivalentes seguras adecuadas.	✓	✓	✓	4.0

#	Description	L1	L2	L3	Desde
C.12	Verifique que cada firmware mantiene una lista de materiales de software que cataloga componentes de terceros, control de versiones y vulnerabilidades publicadas.	✓	✓	✓	4.0
C.13	Verifique que todo el código, incluidos los archivos binarios de terceros, las bibliotecas y los marcos de trabajo, se revisen para las credenciales codificadas de forma hardcoded (backdoors).	✓	✓	✓	4.0
C.14	Verifique que la aplicación y los componentes de firmware no son susceptibles a la inyección de comandos del sistema operativo invocando contenedores de comandos de shell, scripts o que los controles de seguridad impiden la inyección de comandos del sistema operativo.	✓	✓	✓	4.0
C.15	Verifique que las aplicaciones de firmware anclan la firma digital a un servidor de confianza.		✓	✓	4.0
C.16	Verifique la presencia de la resistencia a la manipulación y/o las características de detección de manipulaciones.		✓	✓	4.0
C.17	Verifique que las tecnologías de protección de propiedad intelectual disponibles proporcionadas por el fabricante del chip estén habilitadas.		✓	✓	4.0
C.18	Verifique que los controles de seguridad estén en su lugar para obstaculizar la ingeniería inversa del firmware (por ejemplo, remueva los símbolos de depuración detallados).		✓	✓	4.0
C.19	Verifique que el dispositivo valide la firma de la imagen de arranque antes de cargarla.		✓	✓	4.0
C.20	Verifique que el proceso de actualización del firmware no es vulnerable a los ataques de tiempo de comprobación frente a los ataques de time-of-check vs time-of-use.		✓	✓	4.0
C.21	Verifique que el dispositivo utiliza la firma de código y valida los archivos de actualización de firmware antes de instalar.		✓	✓	4.0
C.22	Verifique que el dispositivo no se pueda degradar a las versiones antiguas (anti-rollback) del firmware válido.		✓	✓	4.0
C.23	Verifique el uso del generador de números pseudoaleatorios criptográficamente seguro en un dispositivo integrado (p. ej., utilizando generadores de números aleatorios proporcionados por chip).		✓	✓	4.0
C.24	Verifique que el firmware pueda realizar actualizaciones automáticas de firmware según una programación predefinida.		✓	✓	4.0
C.25	Verifique que el dispositivo borra el firmware y los datos confidenciales al detectar la manipulación o la recepción de mensajes no válidos.			✓	4.0
C.26	Verifique que solo se utilicen microcontroladores que admitan la desactivación de interfaces de depuración (por ejemplo, JTAG, SWD).			✓	4.0
C.27	Verifique que solo se utilizan microcontroladores que proporcionan una protección sustancial contra ataques de des encapsulación (decapping) y de canal lateral.			✓	4.0
C.28	Verifique que las trazas sensibles no estén expuestas a las capas externas.			✓	4.0

#	Description	L1	L2	L3	Desde
C.29	Verifique que la comunicación entre chips esté cifrada (p. ej., comunicación de la placa principal a la placa hija).			✓	4.0
C.30	Verifique que el dispositivo usa código firmado y valida el código antes de la ejecución.			✓	4.0
C.31	Verifique que la información confidencial mantenida en la memoria se sobrescribe con ceros tan pronto como ya no sea necesaria.			✓	4.0
C.32	Verifique que las aplicaciones de firmware utilizan contenedores de kernel para el aislamiento entre aplicaciones.			✓	4.0
C.33	Verifique que los indicadores seguros del compilador como -fPIE, -fstack-protector-all, -Wl,-z,noexecstack, -Wl,-z,noexecheap están configurados para compilaciones de firmware.			✓	4.0
C.34	Verifique que los microcontroladores estén configurados con protección de código (si corresponde).			✓	4.0

Referencias

Para obtener más información, véase también:

- OWASP Internet of Things Top 10
- OWASP Embedded Application Security Project
- OWASP Internet of Things Project
- Trudy TCP Proxy Tool