DETECCIÓN Y GENERACIÓN DE RECOMENDACIONES PARA EL CIERRE DE LAS VULNERABILIDADES RELACIONADAS EN EL TOP 10 OWASP IDENTIFICADAS EN LA APLICACIÓN WEB DE HISTORIAS CLÍNICAS EN LA INSTITUCIÓN PRESTADORA DE SERVICIOS DE SALUD ESPECIALIZADA EN AUDIOLOGÍA AUDIOCOM IPS

LUIS ALFONSO GIRÓN MIRANDA
HENRY ALEXANDER TORRES ROA

UNIVERSIDAD PILOTO DE COLOMBIA
FACULTAD DE INGENIERÍA, PROGRAMA INGENIERÍA DE SISTEMAS
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA.
BOGOTÁ, D.C.
2015

DETECCIÓN Y GENERACIÓN DE RECOMENDACIONES PARA EL CIERRE DE LAS VULNERABILIDADES RELACIONADAS EN EL TOP 10 OWASP IDENTIFICADAS EN LA APLICACIÓN WEB DE HISTORIAS CLÍNICAS EN LA INSTITUCIÓN PRESTADORA DE SERVICIOS DE SALUD ESPECIALIZADA EN AUDIOLOGÍA AUDIOCOM IPS

LUIS ALFONSO GIRÓN MIRANDA
HENRY ALEXANDER TORRES ROA

Trabajo de grado para optar al título de
Especialista en Seguridad Informática

Director
ÁLVARO ESCOBAR ESCOBAR
MSc.

UNIVERSIDAD PILOTO DE COLOMBIA
FACULTAD DE INGENIERÍA, PROGRAMA INGENIERÍA DE SISTEMAS
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA.
BOGOTÁ, D.C.
2015

CONTENIDO

LISTA DE FIGURAS

Pág.

LISTA DE CUADROS

GLOSARIO

AUDIOLOGÍA: es una rama de las ciencias clínicas que se encarga de diagnosticar y prevenir los problemas auditivos en los seres humanos. Además de la rehabilitación de discapacidades auditivas, ya sea mediante la adaptación de audio prótesis, mediante terapias de rehabilitación (principalmente en casos de acúfenos) y en colaboración con otras disciplinas, mediante los implantes cocleares[1].

FONOAUDIOLOGÍA: es la ciencia que estudia la comunicación humana y sus desórdenes[2].

GESTIÓN DOCUMENTAL: conjunto de programas utilizados para rastrear y almacenar documentos electrónicos o imágenes digitales de documentos originalmente soportados en papel[3].

INICIATIVA CERO PAPEL: cero papel es la iniciativa del Plan Vive Digital que busca hacer más eficiente la gestión administrativa interna en las entidades con el fin de prestar un mejor y más eficiente servicio al ciudadano, en este caso al estudiante[4].

IPS*: institución prestadora de servicios de salud[5].

LEY 1581: en la cual se dictan disposiciones generales para la protección de datos personales en Colombia[6].

---

[1] SALESA, Enrique; PERELLO, Enrique y BONAVIDA, Alfredo. Tratado de audiología. 2013. Vol. 2, No. 357, p. 27-57. [en línea], [consultado el 22 de abril de 2015]. Disponible en: https://books.google.com.co/books?isbn=8445823957

[2] Ibíd, p. 27-57.

[3] CALDERA, Jorge. Realidad Aumentada en Televisión y Propuesta de Aplicación en los Sistemas de Gestión Documental. Diciembre de 2014. [en línea], [consultado el 22 de abril de 2015]. Disponible en: http://web.a.ebscohost.com/ehost/results?sid=d0c7580a-ff4d-4b76-ba96-9fd984ebedcb%40sessionmgr4003&vid=2&hid=4201&bquery=gestion+documental&bdata=JmR

[4] MINISTERIO DE TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES, Cero Papel en la Administración Pública. Octubre 18 de 2012. [en línea], [consultado el 2 de marzo de 2014]. Disponible en: http://programa.gobiernoenlinea.gov.co/apc-aa-files/Cero_papel/guia-1-cero-papel.pdf

[5] COLOMBIA. CONGRESO DE LA REPÚBLICA. Ley 100. Diciembre 23 de 1993. Por la cual se crea el sistema de seguridad social integral y se dictan otras disposiciones. Diario Oficial. Bogotá D.C., 1993. No 41148. p. 1

[6] COLOMBIA. CONGRESO DE LA REPÚBLICA. Ley 1581. Octubre 17 de 2012. Por la cual se dictan disposiciones generales para la protección de datos personales. Diario Oficial. Bogotá D.C., 2012. No 48587. p. 1.

OPENSOURCE: promueve la libre distribución y el acceso al diseño de un producto final y sus detalles de implementación[7].
OWASP: comunidad abierta dedicada a habilitar a las organizaciones para desarrollar, comprar y mantener aplicaciones confiables[8].
TOP 10 OWASP: proyecto para crear conciencia acerca de la seguridad en aplicaciones mediante la identificación de algunos de los riesgos más críticos que enfrentan las organizaciones[9].

[7] PETRINJA, Etiel. IntroducingtheOpenSourceMaturityModel. Mayo de 2009. [en línea]. [consultado el 22 de abril de 2015]. Disponible en: http://web.a.ebscohost.com/ehost/ detail/ detail? vid=10& sid= d0c7580a-ff4d-4b76-ba96-9fd984ebedcb%40sessionmgr4003&hid=4201&bdata=

[8] OWASP Foundation. Los Diez Riesgos más Críticos en Aplicaciones Web. 2013. [en línea], [consultado el 2 de marzo de 2014]. Disponible en: https://www.owasp.org/index.php/About_OWASP

[9] OWASP Foundation. Los Diez Riesgos más Críticos en Aplicaciones Web. 2013. [en línea], [consultado el 2 de marzo de 2014]. Disponible en: https://www.owasp.org/index.php/ About_ OWASP

RESUMEN

El trabajo que se presenta a continuación muestra la implementación de la guía de pruebas de OWASP versión 3, junto al estándar de verficación y aseguramiento de aplicaciones para la evaluación y detección de las vulnerabilidades más relevantes del top 10 OWASP, que se encuentran en la aplicación web desarrollada por la IPS especializada en audiología, Audiocom, para la gestión de sus historias clínicas, con lo que finalmente se generan algunas recomendaciones que contribuyen a la supresión de las fallas de seguridad detectadas que exponen negativamente la información confidencial que allí se almacena.

Palabras clave: audiología, vulnerabilidad, historia clínica.

INTRODUCCIÓN

En el presente documento se evidencia la evaluación y detección de las vulnerabilidades más relevantes que se enuentran en la aplicación web que desarrolla y utiliza la institución prestadora de servicios de salud especializada en audiología, Audiocom, para la gestión de sus historias clínicas electrónicas. Este proyecto centra sus objetivos en el seguimiento y la aplicación de la metodología de la guía de pruebas de OWASP junto al desarrollo del estándar de verificación y aseguramiento de aplicaciones, con lo que se obtienen algunas recomendaciones que contribuyen a la minimización de riegos por exposición de las vulnerabilidades que se relacionan en el top 10 OWASP.

Adicional al trabajo realizado en la evaluación y detección de vulnerabilidades, se evidencian algunas fallas de seguridad que son inherentes al afán de Audiocom por cumplir con los requerimientos establecidos para la habilitación de funcionamiento como entidad prestadora de servicios de salud, que específicamente es la digitalización de historias clínicas. Se evidencia que la institución desarrolla su aplicativo web cumpliendo con los requerimientos funcionales, sin considerar los requerimientos de seguridad necesarios para garantizar la confidencialidad, disponibilidad e integridad de los registros clínicos almacenados.

El desarrollo de este trabajo se realizó por el interés de darle a conocer a Audiocom IPS las vulnerabilidades a las que se encuentra expuesto su aplicativo web y por tanto la información sensible que se almacena en el mismo, así como la forma de mejorar sus procesos de desarrollo de software, teniendo en cuenta que la explotación de las vulnerabilidades encontradas puede representar una pérdida monetaria significativa.

En el ámbito profesional, como especialistas en seguridad informática, el interés versó en aplicar a la realidad de las instituciones prestadoras de servicios de salud colombianas una serie de procedimientos y técnicas que muestren las falencias de los software que se desarrollan por el afán de dar cumplimiento a los requerimientos legales establecidos para el funcionamiento de estas instituciones, así como mostrar algunas recomendaciones que aportan a la minimización de riesgos.

En la realización de este trabajo se realizaron entrevistas con los funcionarios de Audiocom IPS que trabajan directamente en el desarrollo de la aplicación web para la gestión de historias clínicas, principalmente, la gerente y los líderes del equipo de desarrollo de software, de allí se obtiene la información necesaria para hacer la detección de vulnerabilidades y el desarrollo del estándar de verificación y aseguramiento de aplicaciones, para finalmente generar recomendaciones que se entregan a la gerencia y al equipo de desarrollo para minimizar los riesgos a los que se encuentra expuesta la aplicación y por tanto la empresa.

1. FORMULACIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

En la actualidad la IPS especializada en audiología, Audiocom, desarrolla y utiliza una aplicación web para la digitalización de historias clínicas que no cumple con los requerimientos de seguridad adecuados para impedir por completo que un atacante pueda explotar sus vulnerabilidades y exponer la información sensible que se almacena en ella.

La ausencia de seguridad en la aplicación se hace evidente teniendo en cuenta los reportes del antivirus usado en los computadores y los log del firewall de Audiocom, a sabiendas que la cantidad de spam y malware en general han disminuido, pero se han incrementado los ataques de contenido web malicioso, también se debe tener en cuenta que el personal que utiliza la aplicación trabaja desde casa en algunas ocasiones y por lo mismo se encontrarán ataques comunes a usuarios domésticos como suplantación de identidad y gusanos. Los usuarios domésticos se están convirtiendo en víctimas cada vez más comunes de los ataques, llegando a representar un alto porcentaje de todos los ataques dirigidos a grupos específicos, seguidos de las empresas de servicios financieros. Symantec ha identificado un incremento de los ataques dirigidos a aplicaciones cliente, además de un aumento del empleo de tácticas evasivas para evitar su detección. El empleo generalizado de gusanos en Internet ha dado paso a unos ataques menores y más centrados para realizar actividades fraudulentas, suplantación de identidad y otro tipo de delitos.

El equipo de programación que trabaja desarrollando la aplicación web de Audiocom no utiliza metodologías de desarrollo adecuadas y tampoco cuenta con gran experiencia en el campo de la programación, por lo que se sabe que esta se encuentra en riesgo permanente y más por estar basada en una arquitectura web, que se encuentra expuesta a millones de usuarios y atacantes en internet.

Es claro que los desarrolladores de software de Audiocom IPS necesitan asegurarse de que el código que entregan no es vulnerable a ataques, ya que no se puede confiar solo en los testers (que en este caso son Audiólogas y no personal técnico como debería) para que lo hagan, pues ellos nunca entenderán la aplicación tan bien como ellos mismos y por tanto nunca serán capaces de probarla tan efectivamente como ellos mismos.

1.2 JUSTIFICACIÓN

Prevenir la fuga premeditada de información sensible por explotación de vulnerabilidades en los aplicativos para el registro de información clínica utilizados por cualquier institución prestadora de servicios de salud, requiere de soluciones a nivel de software cada vez más eficientes que soporten sus procesos y procedimientos en organizaciones certificadas a nivel mundial en el desarrollo de estrategias y procedimientos que eviten estos inconvenientes.

Sabiendo que a diario la IPS especializada en audiología Audiocom, maneja cientos de historias clínicas que deben ser tratadas como información de alta sensibilidad e importancia, que contienen exámenes clínicos e información personal de los pacientes utilizados en cada consulta para dar trazabilidad a los procedimientos realizados y que de allí depende el éxito de la próxima consulta y por tanto la salud auditiva de cada paciente, por lo cual es de vital importancia el estricto cumplimiento de la ley de protección de datos en la aplicación, teniendo en cuenta que la misma exige que confidencialidad, integridad y disponibilidad en la historia clínica electrónica.

La importancia de prevenir la explotación de vulnerabilidades en el aplicativo web de Audiocom IPS, radica en que para la institución es de mayor importancia que este sistema de información clínica funcione correctamente, para evitar que se genere atención ineficiente en las consultas clínicas y la pérdida de dinero por inatención de pacientes entre otros impactos. Sin embargo, cabe resaltar que el objeto de negocio de la institución no es el desarrollo de aplicaciones ni el enfoque en la seguridad, por lo cual no se consideraron estos elementos desde el inicio del desarrollo de la aplicación.

Los beneficios de la detección de vulnerabilidades y generación de recomendaciones realizados a la aplicación, permitirán la identificación de una serie de fallas de seguridad que se explotan con mayor frecuencia en las aplicacione web y que finalmente deben ser corregidas para mantener una gestión adecuada y segura de las historias clínicas electrónicas, ayudando a garantizar la continuidad del funcionamiento de la misma.

1.3 OBJETIVOS

1.3.1 Objetivo general. Identificar las vulnerabilidades más significativas de la aplicación web de historia clínica de Audiocom IPS, considerando el enfoque de la guía de pruebas OWASP V.3 y el desarrollo del estándar de verificación y aseguramiento de aplicaciones (ASVS) 2013 V.1 para generar recomendaciones necesarias que busquen el cierre las vulnerabilidades relacionadas en el top 10 y busquen minimizar el riesgo de pérdidas monetarias por explotación de las mismas.

1.3.2 Objetivos específicos.

- Desarrollar el estándar de verificación y aseguramiento de aplicaciones (ASVS) teniendo como base la guía de pruebas de OWASP y el top 10 de las vulnerabilidades más explotadas en aplicaciones web.
- Generar recomendaciones que propongan el cierre de las vulnerabilidades encontradas.

2. MARCO REFERENCIAL

2.1 MARCO TEÓRICO

"La seguridad informática es la disciplina que se ocupa de diseñar las normas, procedimientos, métodos y técnicas destinados a conseguir un sistema de información seguro y confiable"[10], esta se encuentra enfocada en la calidad de seguro, aunque se conoce que no existe la seguridad absoluta, el fin de la seguridad informática es mantener segura al máximo la información. Esto conlleva a que para poder tener este título de seguro y de poder tener plena confianza, se tiene que validar con un análisis de riesgo sobre aquello que se quiere proteger.

Una regla básica de la ingeniería del software es que no se puede controlar lo que no se puede medir[11]. Las pruebas de Seguridad no son diferentes. Infortunadamente, medir la seguridad es notoriamente una tarea dificultosa. Un aspecto que se debe enfatizar es que las medidas de seguridad son, por necesidad, acerca de los problemas técnicos específicos (ej. que tan prevalente es una cierta vulnerabilidad) y como estas afectan las economías del software. La mayoría de la gente entiende al menos las implicaciones básicas, o tiene un conocimiento técnico más profundo de las vulnerabilidades. Por desgracia, muy pocos son capaces de realizar una conversión de dicho conocimiento en un valor monetario, y de ese cuantificar los costes que acarrea a su negocio. Se cree que hasta que eso ocurra, los responsables informáticos no podrán desarrollar un cálculo preciso del retorno sobre una inversión en seguridad, y por tanto asignar los presupuestos adecuados para la seguridad del software. El coste del software inseguro en la economía mundial es aparentemente inconmensurable. En Junio del 2002, el instituto de estándares Nacional Estadounidense (NIST) publicó un estudio sobre los costes del software inseguro para la economía estadounidense debidos a la comprobación inadecuada del software[12]. Interesantemente, ellos estiman que una mejor infraestructura de comprobación podría ahorrar un tercio de estos costes, o alrededor de $22 billones al año. Más recientemente, las relaciones entre economía y seguridad han sido estudiadas por investigadores académicos. La mayor parte de la gente no comprueba el software hasta que ya ha sido creado y se encuentra en la fase de desarrollo de su ciclo de vida. Esta es generalmente una práctica muy inefectiva y costosa. Uno de los mejores métodos

---

[10] LOPEZ, Purificación A. Seguridad informática. N.p.: Editex, n.d. 9

[11] T. De Marco, Controlling software projects: management, measurement and estimation. Londrés: Press, 1982. p. 30

[12] NIST, The economic impacts of inadequate infrastructure for software testing [enlínea], [consultado el 23 de marzo de 2015].Disponible en: http://www.nist.gov/public_affairs/releases/n02- 10.htm

para evitar la aparición de bugs de seguridad en aplicaciones en producción es mejorar el Ciclo de Vida de Desarrollo del Software (en inglés Software Development Life Cycle o SDLC). La figura 1 muestra un modelo SDLC genérico, así como los costes en incremento progresivo (estimados) de corregir bugs de seguridad en un modelo de este tipo.

Figura 1. Modelo SDLC genérico.

Fuente: OWASP[13]

Puede ser de gran ayuda pensar en el desarrollo de software como en una combinación de personas, procesos y tecnología. Si esos son los factores que "crean" el software, es lógico que esos sean los factores que deben ser probados. Hoy en día la mayoría de personas realizan pruebas a la tecnología o el propio software. De hecho, la mayoría no realiza pruebas al software hasta que ya ha sido creado y está en la fase de desarrollo de su ciclo de vida (es decir, que el código ha sido creado e instanciado en una aplicación web en uso). Generalmente, esta es una práctica muy ineficaz y prohibitiva en coste. Un programa de pruebas efectivo debería tener componentes que comprueban las personas para asegurarse de que hay la educación y concienciación adecuadas. Los procesos para asegurarse que hay las políticas y estándares adecuados y que las personas saben cómo seguir dichas políticas, la tecnología para asegurarse de que el proceso ha sido efectivo en su implementación. A menos se adopte un enfoque integral, probar tan solo la implementación técnica de una aplicación no descubrirá vulnerabilidades operacionales o de gestión que se puedan presentar. Mediante la realización de pruebas sobre las personas, políticas y procesos, se puede llegar a prever incidencias o problemas que mas tarde se manifestasen en

[13] OWASP, Guía de pruebas V3.0 [enlínea], [consultado el 23 de marzo de 2015].Disponible en: https://www.owasp.org/images/8/80/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf

forma de defectos en la tecnología, y así erradicar bugs de forma temprana e identificar las causas de los defectos. Del mismo modo que probar solamente algunas de las incidencias técnicas que pueden estar presentes en un sistema resultará en un punto de vista incompleto e incorrecto de cómo realizar una evaluación de seguridad. Denis Verdon, Responsable de Seguridad de la Información en Fidelity National Financial (http://www.fnf.com) presentó una excelente analogía para ésta concepción errónea en la conferencia del OWASP AppSec 2004 en Nueva York. “Si los coches fuesen construidos como aplicaciones... las pruebas de seguridad asumirían tan solo impactos frontales. A los coches no se les harían pruebas de rodaje, o serían probados en estabilidad en maniobras de emergencias, eficacia de los frenos, impactos laterales y resistencia a robos.”

En los últimos años, los profesionales de la seguridad han llegado a darse cuenta de la falacia que representa el modelo de parchear y penetrar, generalizado en seguridad de la información durante los 90. El modelo de parchear y penetrar comprende corregir un bug reportado, pero sin la investigación adecuada de la causa origen. El modelo de parchear y penetrar se asocia a menudo con la ventana de vulnerabilidad[14]. La figura 2 muestra el comportamiento de la ventana de exposición. La evolución de vulnerabilidades en el software común usado por todo el mundo ha demostrado la ineficacia de este modelo. Estudios de las vulnerabilidades[15] han mostrado que con el tiempo de reacción de los atacantes por todo el mundo, la ventana de vulnerabilidad típica no provee del tiempo suficiente para la instalación de parches, ya que el tiempo entre que la vulnerabilidad es descubierta y un ataque automatizado es desarrollado y divulgado decrece cada año. Existen también varias asunciones erróneas en este modelo de parchear y penetrar, los parches interfieren con la operativa normal y pueden quebrantar aplicaciones existentes, y no todos los usuarios podrían ser conscientes de la disponibilidad de un parche.

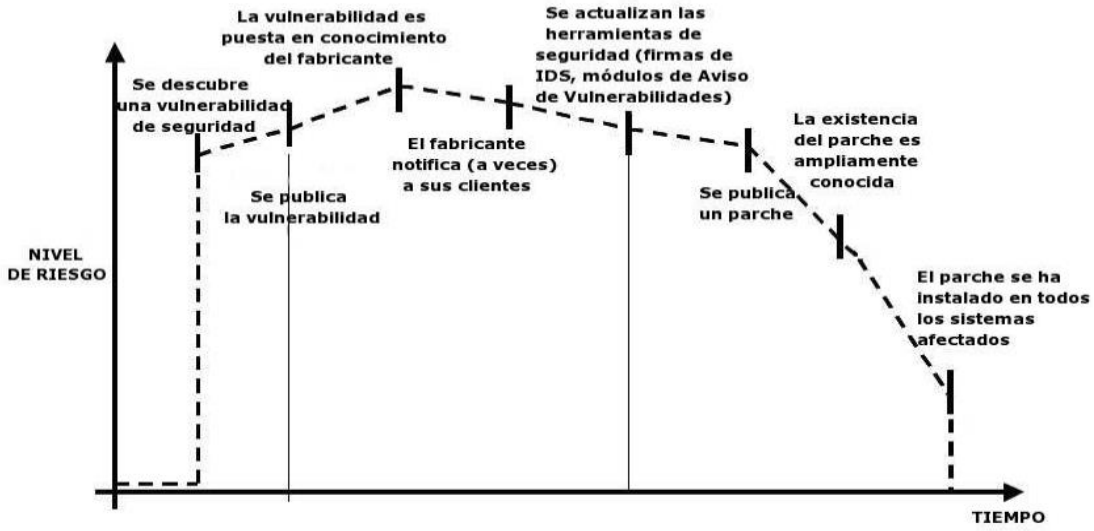
Por lo tanto no todos los usuarios del producto aplicarán los parches, debido a la incidencia o por falta de conocimiento de la existencia del mismo.

---

[14] SCHNEIER, Bruce. CryptogramIssue #9. Testing [en línea], [consultado el 23 de marzo de 2015]. Disponible en: http://www.schneier.com/crypto-gram-0009.html

[15] SYMANTEC.ThreatReportstesting [en línea], [consultado el 23 de marzo de 2015]. Disponible en: http://www.symantec.com/business/theme.jsp?themeid=threatreport

Figura 2. Ventana de exposición.



Fuente: OWASP[16]

2.1.1 Revisión del código fuente. La revisión de código fuente es el proceso de comprobar manualmente el código fuente de una aplicación web en busca de incidencias de seguridad. Muchas vulnerabilidades de seguridad serias no pueden ser detectadas con ninguna otra forma de análisis o prueba. Como dice la conocida frase ``si quieres saber qué es lo que está pasando realmente, vete directo a la fuente´´. Casi todos los expertos en seguridad están de acuerdo en que no hay nada mejor que ver realmente el código. Toda la información necesaria para identificar problemas de seguridad está en el código en algún lugar. De modo diferente a comprobar software cerrado de terceras partes, como sistemas operativos. Cuando se realizan pruebas en aplicaciones web (especialmente cuando han sido desarrolladas internamente), el código fuente debería ser puesto a disposición para comprobarlo.

Muchos problemas de seguridad no intencionados pero significativos, son también extremadamente difíciles de descubrir con otras formas de comprobación o análisis, como las pruebas de intrusión, haciendo del análisis de código la técnica preferida para las comprobaciones técnicas. Con el código fuente, una persona comprobándolo puede determinar con exactitud qué está pasando (o qué se supone que está pasando), y eliminar el trabajo de adivinar de la comprobación de caja negra. Ejemplos de incidencias particularmente propicias a ser encontradas a través de las revisiones de código fuente incluyen problemas de concurrencia lógica de negocio errónea, problemas de control de acceso y

[16] OWASP, Guía de pruebas V3.0 [enlínea], [consultado el 23 de marzo de 2015]. Disponible en: https://www.owasp.org/images/8/80/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf

debilidades criptográficas, así como puertas traseras, Troyanos, Huevos de Pascua, bombas de tiempo, bombas lógicas y otras formas de código malicioso. Estas incidencias a menudo se manifiestan como las vulnerabilidades más dañinas en websites. El análisis de código fuente puede ser también extremadamente eficiente a la hora de encontrar incidencias de implementación, como localizaciones en las que la validación de entradas no es realizada o cuando pueda haber presentes procedimientos erróneos de control de fallos. Pero se debe tener en cuenta que los procedimientos operativos deben ser revisados también, ya que el código fuente usada puede no ser el mismo que el analizado[6].

Las ventajas son: eficacia, precisión y rapidez. Las desventajas requieren desarrolladores de seguridad altamente competentes; no puede detectar errores en tiempo de ejecución con facilidad, el código fuente realmente en uso puede ser diferente del que está siendo analizado.

2.1.1.1 Las pruebas de intrusión. Las pruebas de intrusión se han convertido desde hace muchos años en una técnica común empleada para comprobar la seguridad de una red. También son conocidos comúnmente como pruebas de caja negra o hacking ético. Las pruebas de intrusión son esencialmente el arte de comprobar una aplicación en ejecución remota, sin saber el funcionamiento interno de la aplicación, para encontrar vulnerabilidades de seguridad. Generalmente, el equipo de prueba de intrusión tendría acceso a una aplicación como si fuesen usuarios. Los probadores actúan como un atacante, e intentan encontrar y explotar vulnerabilidades. En muchos casos al encargado de las pruebas se le da una cuenta válida en el sistema. Mientras que las pruebas de intrusión han demostrado ser efectivos en seguridad de redes, la técnica no se traslada de forma natural al caso de aplicaciones. Cuando se realizan pruebas de intrusión en redes y sistemas operativos, la mayoría del trabajo se centra en encontrar y explotar vulnerabilidades conocidas en tecnologías específicas. Dado que las aplicaciones web son casi todas hechas a medida exclusivamente, las pruebas de intrusión en el campo de aplicaciones web son más similares a la investigación pura. Se han desarrollado herramientas de las pruebas de intrusión para automatizar el proceso pero, de nuevo, por la naturaleza de las aplicaciones web, su eficacia es a menudo escasa. Un programa de pruebas de intrusión sobre aplicaciones web, no debe ser considerado como la principal o única técnica. Gary McGraw[17] resumió bien las pruebas de intrusión cuando dijo “Si suspendes una prueba de intrusión sabes que, de hecho, tienes un problema muy grave. Si pasas una prueba de intrusión no sabes si no tienes un problema muy grave”. Sin embargo una prueba de intrusión enfocado (esto es, un test que intenta explotar

[17] GARY McGraw, Beyond the Badness-ometer.testing[en línea], [consultado el 23 de marzo de 2015]. Disponible en: testing[en línea], [consultado el 23 de marzo de 2015]. Disponible en: http://www.ddj.com/security/189500001

vulnerabilidades conocidas detectadas en revisiones anteriores) puede ser de utilidad para detectar si alguna vulnerabilidad específica está realmente corregida en el código fuente desplegado en la web.

Las ventajas son: puede ser rápido (y por tanto barato). Requiere un conocimiento relativamente menor que una revisión de código fuente. Compruebe el código que está siendo expuesto realmente.

Las desventajas son:

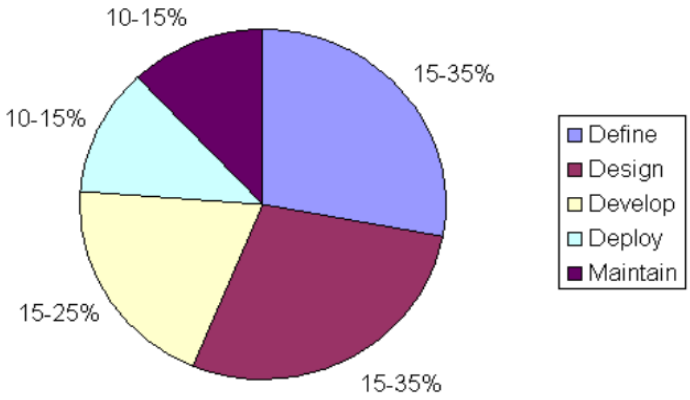
- Demasiado tardío en el SDLC.
- Pruebas solo de impactos frontales.

2.1.1.2 El enfoque equilibrado. Con tantas técnicas y enfoques para comprobar la seguridad las aplicaciones web, puede resultar difícil comprender que técnicas usar y cuando usarlas. La experiencia muestra que no hay una respuesta correcta o incorrecta a cuáles serían exactamente las técnicas que deberían usarse para construir un marco de pruebas. El hecho es que probablemente todas las técnicas deberían ser empleadas para asegurar que todas las áreas que necesitan ser probadas son cubiertas. Sin embargo, lo que está claro, es que no hay una sola técnica que cubra eficazmente toda la comprobación de seguridad que debe ser realizada.

Para asegurar que todas las incidencias son abordadas, muchas compañías adoptan un enfoque, que históricamente han sido las pruebas de intrusión, que a pesar de su utilidad, no puede abordar efectivamente muchas de las incidencias que requieren ser comprobadas, y simplemente es insuficiente y demasiado tarde dentro del ciclo de desarrollo del software, por lo cual deberían utilizarse como base del proyecto, haciendo parte de toda una metodología de desarrollo seguro que debería considerar la institución para minimizar los riesgos de su aplicación y desarrollos posteriores. El enfoque correcto es uno equilibrado que incluya varias técnicas, desde las revisiones manuales hasta las pruebas técnicas. Un enfoque equilibrado asegura la cobertura de pruebas en todas las fases del SDLC, este enfoque explota el potencial de las técnicas más apropiadas disponibles dependiendo de la fase actual del SDLC. Por supuesto hay momentos en que solo una técnica es aplicable; por ejemplo, un test en una aplicación web que ya ha sido creada y en el que el probador no tiene acceso al código fuente. En este caso, un test de intrusión es claramente mejor que no probar nada. Sin embargo, se recomienda a los responsables de realizar las pruebas, poner a prueba cualquier asunción, como el no tener acceso al código fuente y explorar la posibilidad de una comprobación completa. Un enfoque equilibrado varía dependiendo de muchos factores, como la madurez del proceso de pruebas y la cultura corporativa. En la figura 3 se muestra la proporción del esfuerzo del SDLC

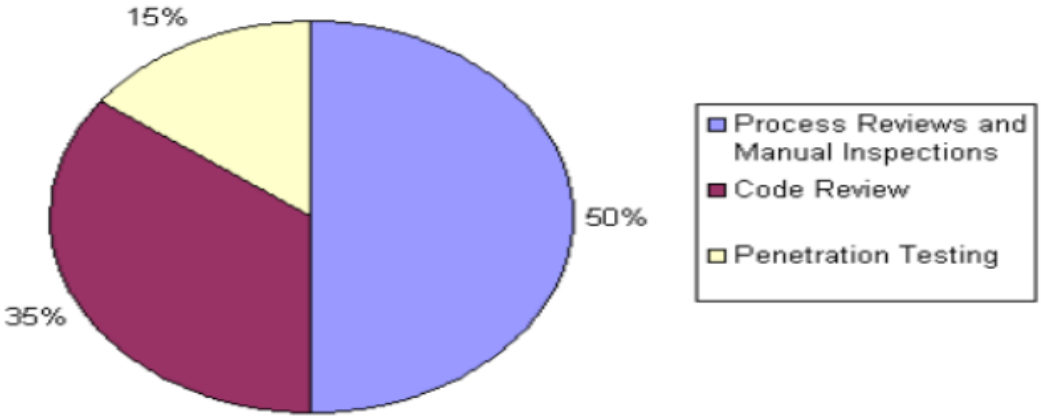
mientras que la figura 4 muestra la proporción del esfuerzo de la prueba según la técnica empleada.

Figura 3. Proporción del esfuerzo de pruebas en el SDLC.



Fuente: OWASP[18]

Figura 4. Proporción del esfuerzo de la prueba según la técnica empleada.



Fuente: OWASP[19]

2.1.1.3 Herramientas para el escaneo de aplicaciones web. Muchas organizaciones han empezado a utilizar herramientas de escaneo de aplicaciones web. Aunque sin duda tienen su lugar en un programa de pruebas, se deben remarcar algunas razones fundamentales por las que las pruebas automatizadas de caja negra no son ni serán efectivas. Resaltar estos puntos no desaconseja el

[18] OWASP, Guía de pruebas V3.0 [enlínea], [consultado el 23 de marzo de 2015].Disponible en: https://www.owasp.org/images/8/80/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf

[19] OWASP, Guía de pruebas V3.0 [enlínea], [consultado el 23 de marzo de 2015].Disponible en: https://www.owasp.org/images/8/80/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf

uso de este tipo de aplicaciones, lo que quiere decir es que sus limitaciones deberían ser comprendidas y los marcos de prueba deberían ser planeados apropiadamente.

Ejemplo 1. Parámetros mágicos: se considera una aplicación web simple que acepta una combinación nombre-valor, primero el parámetro *mágico* y luego el valor. Por simplicidad, la petición GET podría ser: http://www.host/application?magic=valor. Para simplificar más el ejemplo, en este caso los valores solo pueden ser caracteres ASCII de la "a" a la "z" (mayúsculas o minúsculas) y números del 0 al 9. Los diseñadores de esta aplicación crearon una puerta trasera administrativa durante las pruebas, pero la ofuscaron para evitar que un observador casual la descubriese. Enviando el valor sf8g7sfjd surtsdieerwqredsgnfg8d (30 caracteres), el usuario podrá iniciar sesión y se le presentará una pantalla administrativa con control total de la aplicación. La petición HTTP sería: http://www.host/application?magic= sf8g7sfjdsurtsdie erwqredsgnfg8d. Dado que todos los otros parámetros eran campos simples de dos y tres caracteres, no es posible empezar a adivinar combinaciones a partir de 28 caracteres. Un scanner de aplicación web necesitará realizar fuerza bruta (o adivinar) el espacio de claves completo de 30 caracteres. Esto suma 3028 permutaciones, o trillones de peticiones HTTP. El código podría ser como el siguiente:

```
publicvoiddoPost( HttpServletRequestrequest, HttpServletResponse) {
Stringmagic = "sf8g7sfjdsurtsdieerwqredsgnfg8d";
booleanadmin = magic.equals(request.getParameter("magic"));
if (admin) doAdmin(request, response);
else …. // proceso normal }
```

Mediante un vistazo al código fuente, la vulnerabilidad prácticamente salta a la vista como un problema potencial.

Ejemplo 2. Mala criptografía: la criptografía es usada ampliamente en aplicaciones web. Como ejemplo un desarrollador decide escribir un algoritmo de 0 cifrado simple para permitir a un usuario registrarse automáticamente del site A al site B. El desarrollador decide, desde su conocimiento, que si un usuario ha iniciado sesión en el site A, puede generar una clave usando una función de hashing MD5 que comprenda Hash usuario-fecha. Cuando un usuario pasa al site B, enviará la clave en la cadena de petición al site B en un redirect HTTP. El site B calcula independientemente el hash, y lo compara al hash que se le ha pasado en la petición. Si coinciden, el site B registra al usuario como el usuario que dice ser. Claramente, tal y como se explica en el sistema se pueden vislumbrar las insuficiencias del mismo y se puede ver como cualquiera que se lo figure (o que alguien le indique como funciona, o descargue la información de Bugtraq) puede iniciar sesión como cualquier usuario. Una revisión manual, como una entrevista, habría descubierto la incidencia de seguridad rápidamente, como también lo haría

una inspección del código. Un scanner de caja negra de la aplicación web habría visto un hash de 128 bits que cambia para cada usuario y, por la naturaleza de las funciones de hash, no cambia en ninguna forma predecible.

2.1.1.4 Herramientas de revisión de código fuente estático. Muchas organizaciones han empezado a utilizar herramientas de código fuente estático, aunque sin duda tienen lugar en un programa de pruebas exhaustivo, se remarcan algunas notas acerca de por qué no se considera que esta solución sea eficaz cuando es usada por sí sola. El análisis de código fuente estático aisladamente no puede comprender el contexto de construcciones semánticas en el código, y por tanto es propenso a hallar un número significativo de falsos positivos. La tecnología es útil en determinar partes interesantes en el código, aunque es necesario un esfuerzo significativo para validar los hallazgos.

2.1.2 Derivaciones de los requerimientos de pruebas de seguridad. Si se desea tener un programa de pruebas, es necesario saber cuáles son los objetivos de las pruebas. Estos objetivos son especificados por los requisitos de seguridad.

2.1.2.1 Objetivos de las pruebas. Uno de los objetivos de las pruebas de seguridad es validar que las funciones de control de seguridad funcionan como se esperaba. Esto está documentado a través de los requisitos de seguridad que describen la funcionalidad del control de seguridad. A alto nivel, esto significa probar la confidencialidad, integridad y disponibilidad de los datos, así como el servicio. El otro objetivo es validar que los controles de seguridad estén implementados con pocos o sin vulnerabilidades. Estas vulnerabilidades son comunes, como el OWASP Top Ten, así como las vulnerabilidades que están previamente identificadas con las evaluaciones de seguridad durante el SDLC, como el modelamiento de amenazas, análisis de código fuente, y prueba de intrusión.

2.1.2.2 Documentación de los requerimientos de seguridad. El primer paso en la documentación de los requisitos de seguridad es entender las necesidades y requerimientos del negocio. Un requisito de negocio podría proveer la información inicial y a un alto nivel de la funcionalidad de la aplicación esperada. Por ejemplo, el propósito principal de una aplicación puede ser proveer servicios financieros a los clientes o de servicios de compras y adquisiciones de bienes a partir de un catálogo en línea. Una sección de seguridad de los requerimientos de negocio debería resaltar la necesidad de proteger los datos de los clientes, así como a cumplir con la documentación de seguridad aplicable, tales como reglamentos, normas y políticas.

Una lista de control general de las regulaciones, normas y políticas aplicables sirve el propósito del cumplimiento preliminar de seguridad para aplicaciones web. Por ejemplo, el cumplimiento de las regulaciones pueden ser identificadas mediante la verificación de información sobre el sector de negocio y el país o estado donde la

aplicación necesita para funcionar u operar. Algunas de estas directrices y el cumplimiento de los reglamentos podrían traducirse en requisitos técnicos específicos para los controles de seguridad. Por ejemplo, en el caso de las aplicaciones financieras, el cumplimiento de las directrices para la autenticación FFIEC[20] requiere que las instituciones financieras implementen aplicaciones que mitiguen los riesgos de autenticación débil con control de seguridad multi-capa y múltiples factores de autenticación.

Normas de Seguridad aplicables a la industria también deben ser capturadas por la lista general de verificación de los requisitos de seguridad. Por ejemplo, en el caso de aplicaciones que manejan datos de tarjetas de crédito de los clientes, el cumplimiento del estándar PCI DSS[21] prohíbe el almacenamiento de números de identificación personal y datos CVV2, y exige que el comerciante proteja los datos almacenados en la banda magnética y su transmisión con cifrado y su visualización mediante enmascaramiento. Tales requisitos de seguridad de PCI DSS pueden ser validados a través de análisis de código fuente. Otra sección de la lista de verificación necesita reforzar requerimientos generales para el cumplimiento de las normas y políticas de seguridad de la organización. Desde la perspectiva de los requerimientos funcionales, los requerimientos de control de seguridad deben ser mapeados a una sección específica de la normativa de seguridad de información.

Un ejemplo de este requerimiento puede ser: "la complejidad de una contraseña de seis caracteres alfanuméricos debe ser forzada por el control de autenticación utilizado por la aplicación". Cuando los requisitos de seguridad mapean al cumplimiento de reglas, una prueba de seguridad puede validar el cumplimiento de la exposición al riesgo. En caso que sea encontrada una violación de las normas y políticas de seguridad de la información, éstas se traducirán en un riesgo que puede ser documentado y que la empresa tiene que hacer frente (es decir, administrar). Por esta razón, desde que los requerimientos de cumplimiento de seguridad deben ser aplicados, éstos tienen que estar bien documentados y verificados con pruebas de seguridad.

2.1.2.3 Valoración de los requerimientos de seguridad. Desde la perspectiva funcional, la validación de los requerimientos de seguridad es el principal objetivo de las pruebas de seguridad, mientras que, desde la perspectiva de gestión del riesgo, este es el objetivo de las evaluaciones de seguridad de la información. A

---

[20] FFIEC, Authentication in an Internet BankingEnvironment. testing[en línea], [consultado el 23 de marzo de 2015]. Disponible en: - http://www.ddj.com/security/18950000

[21] PCI SECURITY STANDARDS COUNCIL, PCI Data Security Standard – testing[en línea], [consultado el 23 de marzo de 2015]. Disponible en: https://www.pcisecuritystandards.org/security_standards/ pci_dss.shtml

nivel alto, el principal objetivo de las evaluaciones de seguridad de la información es la identificación de vacíos en los controles de seguridad, tales como la falta de autenticación básica, autorización, o controles de cifrado. Más en profundidad, el objetivo de la evaluación de seguridad es el análisis de riesgos, tal como la identificación de posibles deficiencias en los controles de seguridad que garanticen la confidencialidad, integridad y disponibilidad de los datos. Por ejemplo, cuando la aplicación trata con información personal identificable y datos sensibles, los requerimientos de seguridad que deberán ser validados son el cumplimiento de la política de seguridad de la información de la compañía que requiere el cifrado de los datos en tránsito y en almacenamiento. Suponiendo que el cifrado se utiliza para proteger los datos, algoritmos de cifrado y las longitudes de las claves necesitan cumplir con las normas de cifrado de la organización. Éstos podrían requerir que sólo ciertos algoritmos y longitudes de clave puedan ser utilizados. Por ejemplo, un requerimiento de seguridad puede ser probado verificando que sólo sean permitidos los algoritmos de cifrado (por ejemplo, SHA-1, RSA, 3DES) que utilizan longitudes de clave mínimas (por ejemplo, más de 128 bits para claves en cifrado simétrico y más de 1024 para claves en cifrado asimétrico). Desde la perspectiva de la evaluación de seguridad, los requerimientos de seguridad pueden ser validados en diferentes fases del SDLC usando diferentes mecanismos y metodologías. Por ejemplo, el modelamiento de amenazas enfocado a la identificación de errores durante el diseño, revisión y análisis de código fuente orientado a identificar problemas de seguridad en el código fuente durante la etapa de desarrollo, y pruebas de intrusión orientada a identificar vulnerabilidades en la aplicación durante la etapa de pruebas/validación.

Problemas de seguridad que se identifican en una fase temprana del SDLC pueden ser documentados en un plan de pruebas para que pueda ser validado posteriormente con pruebas de seguridad. Al combinar los resultados de las diferentes técnicas de prueba, es posible obtener mejores casos de pruebas de seguridad y aumentar el nivel de aseguramiento de los requerimientos de seguridad. Por ejemplo, es posible distinguir las verdaderas vulnerabilidades desde las explotables de las no explotables cuando los resultados de ensayos de intrusión y análisis de código fuente son combinados. Considerando una prueba de seguridad para una vulnerabilidad de inyección SQL, por ejemplo, una prueba de caja negra podría implicar primero un escaneo de la aplicación para descubrir y registrar la vulnerabilidad. La primera evidencia de una posible vulnerabilidad de inyección SQL que puede ser validada con la generación de una excepción de SQL. Una validación adicional de la vulnerabilidad SQL podría implicar inyectar manualmente los vectores de ataque para modificar la gramática de la consulta SQL y obtener una revelación de información. Esto podría implicar muchas pruebas de ensayo y error hasta que la consulta malintencionada sea ejecutada. Suponiendo que el ingeniero de pruebas (el que ejecuta las pruebas) tiene acceso al código fuente, él podría aprender del análisis del código fuente para construir el vector de ataque de SQL que podría explotar la vulnerabilidad (por ejemplo,

ejecutar una consulta que devuelve datos confidenciales a usuarios no autorizados).

2.1.2.4 Taxonomías de amenazas contramedidas. Una clasificación de amenaza y contramedida que tenga en cuenta las causas profundas de las vulnerabilidades, es el factor crítico para verificar que los controles de seguridad están diseñados, codificados, y construidos de tal manera que el impacto debido a la exposición de tales vulnerabilidades sea mitigado.

En el caso de las aplicaciones web, la exposición de los controles de seguridad para vulnerabilidades comunes, como el OWASP Top Ten, pueden ser un buen punto de partida para obtener los requerimientos de seguridad generales. Más concretamente, el marco de seguridad de aplicaciones web[22], establece una clasificación (por ejemplo, la taxonomía) de las vulnerabilidades que pueden ser documentadas en distintas directrices y normas y validado con pruebas de seguridad.

El foco de la categorización de amenazas y contramedidas es definir los requerimientos de seguridad en términos de amenazas y las causas de la vulnerabilidad. Una amenaza puede ser categorizada mediante el uso de STRIDE[23], por ejemplo, como suplantación o imitación (spoofing), manipulación (tampering), repudio (repudiation), revelación de información (information disclosure), denegación de servicio (denial of service), y la elevación de privilegios (elevation of privilege). La causa puede ser categorizada como defecto de seguridad en el diseño, un error de seguridad en la codificación, o un problema debido a una configuración insegura. Por ejemplo, la causa de una vulnerabilidad en la autenticación débil podría ser la falta de autenticación mutua cuando los datos cruzan los límites de confianza entre las capas de la aplicación cliente y servidor. Un requerimiento de seguridad que recoge la amenaza de no repudio durante la revisión del diseño de arquitectura permite la documentación del requerimiento de la contramedida (por ejemplo, la autenticación mutua) que puede ser validada posteriormente con pruebas de seguridad.

Una categorización de amenazas y contramedidas para vulnerabilidades puede ser utilizada para documentar requerimientos de seguridad para la codificación segura, tal como normas de codificación segura. Un ejemplo de un error común en la codificación en los controles de autenticación consiste en aplicar una función de

---

[22] MSDN, CheatSheet: Web Application Security Frame - testing[en línea], [consultado el 23 de marzo de 2015]. Disponibleen: http://msdn.microsoft.com/enus/library/ms978518. aspx# tmwacheatsheet _webappsecurityframe

[23] MSDN, Improving Web Application Security, Chapter 2, Threat And Countermeasures - testing [enlínea], [consultado el 23 de marzo de 2015]. Disponibleen: http://msdn.microsoft. com/ enus/ library/aa302418.aspx

hash para cifrar una contraseña sin usar una semilla inicial. Desde la perspectiva de la codificación segura, esta es una vulnerabilidad que afecta al cifrado usado para la autenticación con una vulnerabilidad origen en un error de codificación. Dado que la causa de es la codificación insegura, el requerimiento de seguridad puede ser documentado en la norma de codificación segura y validada a través de revisiones de código durante la fase de desarrollo del SDLC.

2.1.2.5 Análisis de riesgo y pruebas de seguridad. Los requerimientos de seguridad deben tener en cuenta la gravedad de las vulnerabilidades para apoyar una estrategia de reducción del riesgo. Suponiendo que la organización mantiene un repositorio de las vulnerabilidades encontradas en aplicaciones, es decir, una base de conocimientos de las vulnerabilidades que finalmente son problemas de seguridad reportados por tipo, los incidentes, la mitigación, la causa raíz, y mapeadas a las aplicaciones donde se encontraron. Dicha base de conocimientos de las vulnerabilidades también se pueden utilizar para establecer métricas para analizar la eficacia de las pruebas de seguridad en todo el SDLC.

Por ejemplo, considere un problema de validación de entrada, tal como una inyección SQL, que fue identificado a través de análisis de código fuente y reportado con la causa del error y el tipo de vulnerabilidad. La exposición de dicha vulnerabilidad puede evaluada a través de una prueba de intrusión, probando campos de entrada con varios vectores de ataque de inyección SQL. Esta prueba podría validar que los caracteres especiales que son filtrados antes de que se guarden en la base de datos y así, mitigar la vulnerabilidad. Al combinar los resultados de análisis de código fuente y pruebas de intrusión, es posible determinar la probabilidad y la exposición de la vulnerabilidad y el cálculo del nivel de riesgo de la vulnerabilidad. Presentando informes del cálculo de riesgo de vulnerabilidad en los resultados (por ejemplo, el informe de la prueba), es posible decidir sobre la estrategia de mitigación. Por ejemplo, vulnerabilidades de riesgo alto y medio pueden ser priorizadas para su solución, mientras que los de bajo riesgo pueden ser solucionados en próximas entregas. Al considerar los escenarios de amenaza de explotación de vulnerabilidades comunes es posible identificar los riesgos potenciales, para lo cual el control de seguridad de la aplicación necesita ser probada de manera segura. Por ejemplo, las vulnerabilidades del Top 10 OWASP pueden ser mapeadas a los ataques como el phishing, violaciones de privacidad, robo de identidad, compromiso de sistema, alteración o destrucción de datos, pérdida financiera y pérdida de reputación. Estas situaciones deben ser documentadas como parte de los escenarios de amenaza. Al pensar en términos de amenazas y vulnerabilidades, es posible elaborar una batería de pruebas que simulan los escenarios de ataques. Idealmente, la base de conocimiento de vulnerabilidad de la organización debería ser utilizada para obtener los riesgos de seguridad derivadas de los casos de prueba para validar los escenarios de ataque más probables. Por ejemplo, si el robo de identidad se considera de alto riesgo, escenario de prueba negativa deberían validar la mitigación de los impactos derivados de la explotación de

vulnerabilidades en la autenticación, control de cifrado, validación de entrada y control de autorización.

2.1.3 Requerimientos funcionales y no funcionales de las pruebas.

2.1.3.1 Requerimientos funcionales de seguridad. Desde el punto de vista de los requisitos funcionales de seguridad, las normas aplicables, las políticas y reglamentos conducen ambas a la necesidad de un tipo de control de seguridad, así como el control de la funcionalidad. Estos requerimientos también son referidos como requerimientos positivos, ya que se espera que la funcionalidad pueda ser validada a través de pruebas de seguridad. Ejemplos de requerimientos positivos son los siguientes: la aplicación se bloqueará al usuario después de seis intentos de inicio de sesión fallidos o las contraseñas deben ser de seis caracteres alfanuméricos como mínimo. La validación de los requerimientos positivos consiste en constatar la funcionalidad esperada y como tal, pueden ser probados recreando las condiciones de prueba y ejecutando las pruebas de acuerdo a las entradas predefinidas y constatando los resultados previstos como la condición de no pasa, si pasa. A fin de validar los requerimientos de seguridad con las pruebas de seguridad, los requerimientos de seguridad deben ser orientadas por la función y ponen en relieve la función esperada e implícitamente, la implementación. Ejemplos de alto nivel de diseño de requerimientos de seguridad para la autenticación pueden ser:

- Proteger las credenciales de usuarios y secretos compartidos, en tránsito y almacenadas.
- Enmascarar datos confidenciales cuando se visualicen (por ejemplo, contraseñas, cuentas).
- Bloquear la cuenta del usuario después de cierto número de intentos de acceso fallidos.
- No mostrar al usuario errores específicos de validación como resultado de un acceso fallido.
- Solamente permitir contraseñas alfanuméricas, que incluyan caracteres especiales y que tengan seis caracteres mínimos de longitud, todo esto para limitar ataques desde la interface.
- Permitir la funcionalidad de cambio de contraseña únicamente a usuarios autenticados validando la antigua contraseña, la nueva contraseña y la respuesta a la pregunta de seguridad, esto para evitar ataques de fuerza bruta a la contraseña a través de la funcionalidad de cambio de contraseña.
- El formulario de reinicio de contraseña debería validar el identificador del usuario y su correo electrónico registrado antes de enviar la contraseña temporal al usuario.
- La contraseña temporal generada debería ser una contraseña de un solo uso.
- Un enlace a la página web de reinicialización de contraseña debería ser enviado al usuario.

- La página web de reinicialización de contraseña debería validar la contraseña temporal del usuario, la contraseña temporal, la nueva contraseña además de la respuesta a la pregunta de seguridad.

2.1.3.2 Requerimientos de seguridad orientados por el riesgo. Las pruebas de seguridad deben también ser impulsadas por el riesgo, es decir, que se necesitan para validar la solicitud de un comportamiento inesperado. Estos son también llamados requisitos negativos ya que indica lo que la aplicación no debe hacer. Ejemplos de requerimientos donde se indique lo que no se debería hacer son los siguientes:

- La aplicación no debería permitir que la data sea alterada o destruida.
- La aplicación no debería ser comprometida o mal usada por un usuario malicioso para transacciones financieras no autorizadas.

Requerimientos negativos son más difíciles de probar, porque no hay comportamiento esperado a comprobar. Esto podría requerir un analista de amenazas para obtener condiciones de entrada imprevistas, causas y efectos. Esto es donde las pruebas de seguridad necesitan ser conducidas por un riesgo de análisis y un modelado de amenazas. La clave es documentar los escenarios de amenazas y la funcionalidad de las contramedidas como un factor a mitigar la amenaza. Por ejemplo, en caso del control de autenticación, los requerimientos de seguridad pueden ser documentados desde la perspectiva de las amenazas y contramedidas:

- Cifrar datos de autenticación en local y en tránsito para mitigar el riesgo de exposición de información y ataque al protocolo de autenticación.
- Cifrar contraseñas usando cifrado no reversible, como algoritmos de resumen (por ejemplo, Hash), y una semilla para evitar ataques de diccionario.
- Bloquear las cuentas después de haber alcanzado un límite de fallos en el inicio de sesión y garantizar el uso contraseñas complejas para mitigar el riesgo de ataques de fuerza bruta.
- Mostrar mensajes genéricos de error en la validación de credenciales para mitigar los riesgos de cosecha y enumeración de cuentas de usuario.
- Autenticar mutuamente al cliente y al servidor para evitar el no repudio y ataques de hombre en el medio.

Herramientas de modelado de amenazas tales como árbol de amenaza y bibliotecas de ataques pueden ser útiles para obtener escenarios de pruebas negativas. Un árbol de amenazas asumirá un ataque de raíz (por ejemplo, el atacante podría ser capaz de leer otros mensajes de usuarios), identificará diferentes exploits de controles de seguridad (por ejemplo, fallos de validación de datos debido a una vulnerabilidad de inyección SQL) y las contramedidas necesarias (por ejemplo, implementar validación de datos y consultas

parametrizadas) que podrían ser válidas para ser efectivas en la mitigación de tales ataques.

2.1.4 Derivaciones de requerimientos de seguridad a través de casos de uso y de uso indebido. Entender qué hace supuestamente una aplicación y cómo lo hace es un prerequisito en la descripción de la funcionalidad de la aplicación. Esto puede ser realizado mediante los casos de uso. Los Casos de Uso, en la forma gráfica como comúnmente es usado en la ingeniería de software, muestra las interacciones de los actores y sus relaciones, y ayuda a identificar los actores en la aplicación, sus relaciones, la secuencia de acciones para cada uno de los escenarios, acciones alternativas, requerimientos especiales y pre y post-condiciones. Similar a los casos de uso, casos de mal uso y o casos de abuso[24] describen los escenarios de mal uso y no intencionado de la aplicación.

Estos casos de uso indebido proporcionan una forma de describir los escenarios de cómo un atacante podría hacer uso indebido y abusar de la aplicación. Pasando por las diversas situaciones en un escenario de uso y pensando en cómo podría ser explotada, los posibles fallos o aspectos de la aplicación que no estén bien definidos, pueden ser descubiertos. La clave es describir todos los escenarios posibles de usos críticos o indebidos, o al menos la gran mayoría. Los escenarios de uso indebido permiten el análisis de la aplicación desde el punto de vista del atacante y contribuye a identificar vulnerabilidades potenciales y contramedidas que necesitarían ser implementadas para mitigar el impacto causado por la exposición potencial a tales vulnerabilidades. Dados todos los casos de uso y de abuso, es importante analizarlos para determinar cuál de ellos son los más críticos y que necesitan ser documentados en los requerimientos de seguridad.

La identificación de los más importantes casos de abuso y de uso indebido conduce la documentación de los requerimientos de seguridad y los controles necesarios donde los riesgos de seguridad deberían ser mitigados. Para obtener los casos de uso indebido[25] es importante definir los escenarios funcionales y los escenarios negativos, y ponerlos en forma gráfica. En el caso de la derivación de los requerimientos de seguridad para la autenticación, por ejemplo, se podrían seguir los siguientes pasos:

---

[24] GIL REGEV, Ian Alexander, Alain Wegmann, Use Cases and Misuse Cases Model the Regulatory Roles of Business Processes .testing[en línea], [consultado el 23 de marzo de 2015]. Disponibleen: http://easyweb.easynet.co.uk/~iany/consultancy/regulatory _processes/ regulatory_processes.htm

[25] SINDRE,G. Opdmal A., Capturing Security Requirements Through Misuse Cases. testing[en línea], [consultado el 23 de marzo de 2015]. Disponible en: http://folk.uio.no/nik/2001/21-sindre.pdf

Paso 1. Describir el escenario funcional: el usuario se autentica proveyendo sus datos de usuario y contraseña. La aplicación da acceso al usuario basándose en la validación de sus credenciales y provee mensajes de error específicos al usuario cuando la validación falla.

Paso 2. Describir el escenario negativo: un atacante rompe el proceso de autenticación a través de un ataque de fuerza bruta o de diccionario de contraseñas y de cuentas de usuario. Los errores proveen información específica a un atacante para adivinar qué cuentas de usuario (identificador de usuario) son actualmente válidas. El atacante, después intentará un ataque de fuerza bruta sobre la contraseña para dicha cuenta de usuario válida. Un ataque de fuerza bruta para contraseñas de cuatro dígitos de longitud mínima puede ser exitosa con un limitado número de intentos (por ejemplo, 10^4 intentos).

Paso 3. Describir escenarios funcionales y escenarios negativos con casos de uso y casos de uso indebido: el escenario funcional consiste de las acciones de usuario (ingresar identificador de usuario y contraseña) y acciones de la aplicación (autenticar el usuario y proveer un mensaje de error si la validación falla). El caso de uso indebido consta de las acciones del atacante, por ejemplo, intentar romper la autenticación por fuerza bruta sobre la contraseña a través de un ataque de diccionario, y adivinando identificadores de usuario válidos desde los mensajes de error. A través de la representación gráfica de las amenazas a las acciones del usuario (usos indebidos), es posible obtener las contramedidas como las acciones que mitiguen tales amenazas.

Paso 4. Obtener los Requerimientos de Seguridad: en este caso, los requerimientos de seguridad para autenticación son:

- Las contraseñas necesitan ser alfanuméricas, letras mayúsculas y minúsculas con una longitud mínima de siete caracteres.
- Las cuentas necesitan ser bloqueadas después de cinco intentos de acceso fallidos.
- Mensajes de error en el acceso deben ser genéricos. Estos requerimientos de seguridad necesitan ser documentados y probados.

2.1.5 Pruebas de seguridad integradas en flujo de programación y de pruebas.

2.1.5.1 Pruebas de seguridad de los programadores. Las pruebas de seguridad durante la fase de desarrollo del SDLC representan la primera oportunidad para los programadores para garantizar que los componentes de software que han desarrollado hayan sido probados a nivel de seguridad antes de que sean integrados con otros componentes e incluidos en la aplicación. Los componentes de software podrían consistir en elementos de software, tales como funciones, métodos y clases, así como interfaces de programación, bibliotecas y ejecutables.

Para probar la seguridad, los programadores pueden confiar en los resultados del análisis de código fuente para verificar estáticamente que el código fuente desarrollado no incluye potenciales vulnerabilidades y que es compatible con las normas de codificación segura. Pruebas Unitarias de seguridad pueden verificar dinámicamente (es decir, en tiempo de ejecución) que los componentes funcionan como se esperaba. Antes de integrar los nuevos y existentes cambios en el código en la construcción de la aplicación, los resultados de análisis estático y dinámico deben ser revisados y validados. La validación de código fuente antes de la integración en la aplicación es, por lo general, responsabilidad del programador senior. El programador senior es también el experto en la materia de software seguro y su función es liderar la revisión segura de código y tomar decisiones sobre si aceptará o no el código que se integrará en la construcción de la aplicación o requerir más cambios y pruebas. Este flujo de revisión segura del código puede ser realizado a través de la aceptación formal, también como una aceptación dentro de una herramienta de gestión de flujo de trabajo. Por ejemplo, asumiendo que la típica gestión de defectos usado para errores funcionales, los errores de seguridad que han sido solucionados por un programador pueden ser informados sobre el sistema de gestión de errores o de cambios. El integrador principal puede ver los resultados de las pruebas reportadas por los programadores en la herramienta y dar autorización de integrar el cambio en la aplicación.

2.1.5.2 Pruebas de seguridad para las pruebas de calidad. Después que los componentes y cambios en el código sean probados por los programadores y verificados en la construcción de la aplicación, el siguiente paso más probable en el proceso de desarrollo de software es realizar pruebas sobre la aplicación como un todo. Este nivel de pruebas es generalmente denominado pruebas de integración y pruebas del nivel del sistema.

Cuando las pruebas de seguridad son parte de estas actividades, ellas pueden ser utilizadas para validar la seguridad de la funcionalidad en la aplicación como un todo, así como la exposición del nivel de vulnerabilidades en la aplicación. Estas pruebas de seguridad sobre la aplicación incluyen tanto pruebas de caja blanca, tales como análisis de código fuente, y pruebas de caja negra, tales como pruebas de intrusión. Las pruebas de caja gris son similares a las pruebas de caja negra. En las pruebas de caja gris se puede asumir que se tiene algún conocimiento parcial de la gestión de sesiones de la aplicación y que debería ayudar a entender la desconexión y la función de timeout estén debidamente aseguradas. El objetivo de las pruebas de seguridad es el sistema completo que es el objeto que será potencialmente atacado e incluye todo el código fuente y el ejecutable. Una peculiaridad de las pruebas de seguridad en esta etapa es que es posible, para los que ejecutarán las pruebas de seguridad, determinar si las vulnerabilidades pueden ser explotadas y exponer la aplicación a riesgos reales. Estos incluyen vulnerabilidades comunes de aplicaciones Web, así como problemas de seguridad que han sido identificados antes en el SDLC con otras actividades como el

modelamiento de amenazas, análisis de código fuente y revisiones de seguridad en código. Por lo general, los ingenieros de pruebas, en lugar de los programadores de software, realizan pruebas de seguridad cuando la aplicación está en posibilidades de realizarle pruebas de integración de sistema. Estos ingenieros de pruebas de seguridad tienen conocimientos de las vulnerabilidades en aplicaciones web, técnicas de pruebas de seguridad de caja negra y caja blanca, y la propia validación de los requerimientos de seguridad en esta fase. Para la realización de tales pruebas de seguridad, es un requisito previo que los casos de prueba de seguridad se documenten en las directrices y procedimientos de pruebas de seguridad.

Un ingeniero de pruebas es quién valida la seguridad de la aplicación en el entorno de integración de sistemas y podría liberar la aplicación para pruebas en el entorno operativo (por ejemplo, pruebas de aceptación del usuario). En esta etapa de la SDLC (es decir, la validación), las pruebas funcionales de la aplicación son por lo general una responsabilidad de control de calidad, mientras que los hackers de sombrero blanco o consultores de seguridad suelen ser responsables de las pruebas de seguridad.

Algunas organizaciones se basan en su propio equipo especializado de hacking ético con el fin de llevar a cabo estas pruebas cuando no es necesaria una tercera parte evaluadora (para propósitos de auditoría). Dado que estas pruebas son el último recurso para la solucionar las vulnerabilidades antes de que la aplicación sea liberada en producción, es importante que estas pruebas sean abordadas como recomendaciones del equipo de pruebas (por ejemplo, las recomendaciones pueden incluir código, diseño o cambio de configuración). En este nivel, auditores de seguridad y responsables de la seguridad de la información discuten los problemas de seguridad reportados y analizan el riesgo potencial de acuerdo a los procedimientos de gestión de riesgos de la información. Tales procedimientos podrían exigir que el equipo de programación solucione todas las vulnerabilidades de alto riesgo antes que la aplicación sea desplegada, a menos que tales riesgos sean reconocidos y aceptados.

2.1.6 Pruebas de seguridad de los programadores.

2.1.6.1 Pruebas de seguridad en la fase de programación. Pruebas unitarias. Desde la perspectiva del programador, el principal objetivo de las pruebas de seguridad es validar que el código está siendo desarrollado en concordancia con los requerimientos estándares de codificación segura. Los elementos de código de los programadores tales como las funciones, métodos, clases, APIs y las bibliotecas necesitan ser validados funcionalmente antes de ser integrados en la aplicación. Los requerimientos de seguridad que los programadores deben seguir deberían ser documentados en las normas de codificación segura y validados con un análisis estático y dinámico. Como actividades de prueba siguiendo una revisión de seguridad en el código, las pruebas unitarias pueden validar que los

cambios en el código requeridos por la revisión de seguridad en el código sean implementados correctamente. Revisión de seguridad en el código y las herramientas de análisis de código fuente ayudan a los programadores en la identificación de los problemas de seguridad en el código fuente. Mediante el uso de pruebas unitarias y análisis dinámico (por ejemplo, la depuración), los programadores pueden validar la funcionalidad de los componentes de seguridad, así como comprobar que las contramedidas siendo desarrolladas mitigan cualquier riesgo de seguridad previamente identificadas a través del modelado de amenazas y análisis de código fuente. Una buena práctica para los programadores es construir casos de prueba de seguridad como un conjunto de pruebas generales de seguridad que forma parte del actual framework de pruebas unitarias. Un conjunto de pruebas generales de seguridad podrían ser derivadas de casos de uso y casos de uso indebido definidos anteriormente para probar la seguridad de funciones, métodos y clases. Un conjunto de pruebas generales de seguridad podrían incluir casos de prueba de seguridad para validar tanto requerimientos positivos como negativos para los controles de seguridad tales como:

- Control de acceso y autenticación.
- Codificación y validación de entrada.
- Cifrado.
- Gestión de sesiones y de usuarios.
- Gestión de errores y excepciones.
- Auditoría y registro.

Los programadores armados con una herramienta de análisis de código fuente integrada en su IDE, normas de codificación de seguridad, y un framework de pruebas de seguridad unitarias pueden evaluar y verificar la seguridad de los componentes de software que están siendo desarrolladas. Casos de prueba de seguridad pueden ser ejecutadas para detectar posibles problemas de seguridad que tienen causas profundas en el código fuente: además de validación de entrada y salida de los parámetros que entran y salen de los componentes, estas cuestiones se incluyen los controles de autenticación y autorización realizada por el componente, la protección de los datos dentro de la componente, excepciones de seguridad y manejo de errores, y auditoría y registro. Frameworks de pruebas unitarias tales como JUnit, Nunit, Cunit pueden ser adaptados para verificar los requerimientos de pruebas de seguridad. En el caso de las pruebas funcionales de seguridad, el nivel de pruebas unitarias puede probar la funcionalidad de los controles de seguridad a nivel de componentes del software, tales como funciones, métodos o clases. Por ejemplo, un caso de prueba podría validar la entrada y la salida (por ejemplo, la variable de saneamiento) y los límites de los controles de las variables preguntando por la esperada funcionalidad del componente. Los escenarios de amenaza identificados con los casos de uso y de uso indebido, pueden ser usados para documentar los procedimientos para pruebas de componentes de software. En el caso de los componentes de

autenticación, por ejemplo, pruebas unitarias de seguridad puede validar la funcionalidad de bloqueo de una cuenta, así como el hecho de que los parámetros de entrada de usuario no puede ser objeto de abuso para evitar el bloqueo de cuenta (por ejemplo, mediante el establecimiento de un contador para cuenta bloqueada a un número negativo). A nivel de componente, pruebas unitarias de seguridad pueden validar afirmaciones positivas como afirmaciones negativas, tales como los errores y manejo de excepciones. Las excepciones deben ser capturadas sin abandonar el sistema en un estado de inseguro, también como una denegación de servicio causada por los recursos que no han sido liberados (por ejemplo, manejo de conexiones no cerradas dentro de un bloque de declaración final), así como una elevación de privilegios (por ejemplo, el aumento de privilegios adquiridos antes de que la excepción sea lanzada y no volver a re-establecerse el previo nivel antes de salir de la función). El manejo seguro de errores puede evitar la divulgación de información a través de mensajes de error y registros. Casos de pruebas de seguridad a nivel unitaria pueden ser desarrolladas por un ingeniero de seguridad quien es experto en seguridad en el software, y es también responsable de validar que los problemas de seguridad en el código fuente hayan sido solucionados y pueden ser revisados en el sistema de construcción integrado. Generalmente, el constructor de la aplicación también se asegura que las librerías externas y ejecutables sean evaluadas a nivel de seguridad por vulnerabilidades potenciales antes de ser integradas en la aplicación.

Escenarios de amenazas para vulnerabilidades comunes que tienen origen en la codificación insegura también son documentadas en la guía de pruebas de seguridad del programador. Cuando una solución es implementada para un defecto de codificación identificado por una análisis de código, por ejemplo, casos de pruebas de seguridad pueden verificar que la implementación de código sigue los requerimientos de codificación segura documentada en los estándares de codificación segura. Análisis de código fuente y pruebas unitarias pueden validar que los cabios de código mitigan la vulnerabilidad expuesta por el defecto de código identificado previamente. El resultado de análisis de código seguro automatizado puede también ser usado como puertas de control para el control de versión: los “software artifacts” no pueden ser verificados en la construcción de problemas en la codificación con severidad alta o media.

2.1.7 Pruebas de seguridad para ingenieros de pruebas funcionales.

2.1.7.1 Pruebas de seguridad durante la fase de integración y validación. Pruebas de integración del sistema y operación. El objetivo principal de las pruebas de integración es validar el concepto de defensa en profundidad, esto es, que la implementación de controles de seguridad provéen seguridad a niveles diferentes. Por ejemplo, la falta de validación de entrada cuando se llama un componente integrado con la aplicación es a veces un factor que puede ser probado con pruebas de integración. El entorno para las pruebas de integración del sistema es

también el primer entorno donde los ingenieros de pruebas pueden simular escenarios de ataques reales, cómo pueden ser ejecutados potencialmente por un usuario malicioso de la aplicación, interno o externo. Pruebas de seguridad en este nivel pueden validar cuándo las vulnerabilidades son reales y cuándo pueden ser explotados por los atacantes. Por ejemplo, una vulnerabilidad potencial en el código fuente puede ser calificada como de alto riesgo debido a la exposición a los posibles usuarios malintencionados, así como por el impacto potencial (por ejemplo, el acceso a información confidencial). Escenarios reales de ataque pueden ser probados con técnicas de pruebas manuales y herramientas de pruebas de intrusión. Pruebas de seguridad de este tipo también se denominan pruebas de hacking ético. Desde la perspectiva de las pruebas de seguridad, se trata de pruebas conducidas por el riesgo y tienen el objetivo de probar la aplicación en el entorno operativo. El objeto es la aplicación que es representativa de la versión de la aplicación que está siendo desplegada en producción.

La ejecución de seguridad en la fase de integración y de validación es fundamental para la identificar vulnerabilidades debido a la integración de componentes así como la validación de la exposición de tales vulnerabilidades. Dado que la seguridad de las aplicaciones de pruebas requieren de un conjunto de conocimientos especializados, que incluye tanto el conocimiento de software y de seguridad que no son típicos entre los ingenieros de seguridad, las organizaciones a menudo requieren formación de seguridad para los desarrolladores de software sobre técnicas de hacking ético, procedimientos de evaluación de seguridad y herramientas. Un escenario realista es desarrollar dichos recursos en casa y documentarlos en guías de pruebas de seguridad y procedimientos que tengan en cuenta el conocimiento de pruebas de seguridad del programador.

A las llamadas listas de trucos o listas de control de casos de prueba de seguridad, por ejemplo, puede proporcionar simples casos de prueba y vectores de ataque que pueden ser utilizados por los ingenieros de pruebas para validar la exposición a vulnerabilidades comunes tales como la suplantación de identidad, la divulgación de información, desbordamientos de búfer, formateo de cadenas, inyección de SQL e inyección XSS, XML, SOAP, problemas de canonización, denegación de servicio, código administrado y los controles ActiveX (por ejemplo, NET). Una primera batería de estas pruebas se pueden realizar manualmente con un conocimiento básico de seguridad de software.

El primer objetivo de las pruebas de seguridad podría ser la validación de un conjunto de requisitos mínimos de seguridad. Estos casos de prueba de seguridad podrían consistir en forzar manualmente en la aplicación errores y estado excepcional y la recopilación de información del comportamiento de la aplicación. Por ejemplo, las vulnerabilidades de inyección SQL pueden ser probadas manualmente mediante la inyección de vectores de ataque a través de la entrada de usuario y mediante la verificación de si las excepciones de SQL son devueltas al usuario. La evidencia de un error de excepción de SQL puede ser una

manifestación de una vulnerabilidad que podría ser explotada. Pruebas de seguridad más profunda pueden requerir al ingeniero de pruebas el conocimiento de técnicas de prueba y de herramientas especializadas.

Además del análisis de código fuente y pruebas de intrusión, estas técnicas incluyen, por ejemplo, inyección de fallas en código fuente y binario, análisis de la propagación de fallas y cobertura de código, pruebas de datos aleatorios, e ingeniería inversa. La guía de pruebas de seguridad debe proporcionar los procedimientos y recomendar las herramientas que pueden ser utilizadas por los ingenieros de pruebas de seguridad para la realización de tales evaluaciones de seguridad a fondo.

El siguiente nivel de pruebas de seguridad después de las pruebas de integración del sistema es llevar a cabo pruebas de seguridad en el entorno de aceptación de usuario. Hay ventajas únicas para la realización de pruebas de seguridad en el entorno operativo. El entorno de pruebas de aceptación de usuario (UAT) es la más representativa de la liberación de configuración, con la excepción de los datos (por ejemplo, datos de pruebas son usados en lugar de datos reales). Una característica de pruebas de seguridad en UAT es probar los problemas de configuración de seguridad. En algunos casos, estas vulnerabilidades podrían representar alto riesgo. Por ejemplo, el servidor que aloja la aplicación web no puede ser configurado con un mínimo de privilegios, sin certificado SSL válido y sin configuración segura, los servicios esenciales deshabilitados y directorio raíz web sin páginas web de ejemplo y sin administración.

2.1.8 Análisis de resultados de pruebas de seguridad y reportes.

2.1.8.1 Objetivos de mediciones y métricas de las pruebas de seguridad. La definición de los objetivos de las métricas y mediciones de las pruebas de seguridad son un requisito previo para la utilización de resultados de las pruebas de seguridad para el análisis de riesgos y gestión de procesos. Por ejemplo, una medida como el número total de vulnerabilidades de seguridad encontradas con las pruebas de seguridad podría cuantificar el nivel de seguridad de la aplicación. Estas medidas también ayudan a identificar los objetivos de las pruebas de seguridad del software: por ejemplo, reducir el número de vulnerabilidades a un aceptable número (mínimo) antes que la aplicación sea desplegada en producción. Otro objetivo manejable podría ser comparar el nivel de seguridad de la aplicación contra una línea de base para evaluar las mejoras en los procesos de seguridad de la aplicación. Por ejemplo, la métrica de seguridad de base podría consistir en una aplicación que sólo se probó con pruebas de intrusión. Los resultados obtenidos a partir de una aplicación que también fue probada durante la codificación debería mostrar una mejora (por ejemplo, menor número de vulnerabilidades) en comparación con la base de referencia. En las pruebas de software tradicionales, el número de defectos del software, tales como los errores encontrados en una aplicación, podría proporcionar una medida de calidad de

software. Del mismo modo, pruebas de seguridad pueden proporcionar una medida de seguridad de software. Desde la perspectiva de la gestión de defectos y presentación de informes, la calidad del software y de seguridad puede usar similar categorización de las causas de defectos y los esfuerzos de remediación. Desde la perspectiva de las causas de errores, un defecto de seguridad puede ser debido a un error en el diseño (por ejemplo, fallo de seguridad) o debido a un error en la codificación (por ejemplo, error de seguridad). Desde la perspectiva del esfuerzo requerido para corregir un defecto, los defectos de seguridad y de calidad pueden ser medidos en términos de horas de programadores para implementar la solución, las herramientas y los recursos necesarios para solucionarlo y, por último, el coste de implementar la solución.

Una peculiaridad de los resultados de las pruebas de seguridad, en comparación con los resultados de calidad, es la clasificación en términos de la amenaza, la exposición de la vulnerabilidad y el impacto potencial que plantea la vulnerabilidad a fin de determinar el riesgo. Probar la seguridad de las aplicaciones consiste en la gestión de riesgos técnicos para asegurarse de que las de contramedidas satisfacen los niveles aceptables de la aplicación. Por esta razón, los resultados de las pruebas de seguridad necesitan apoyar la estrategia de la gestión del riesgo de seguridad en los puntos de control crítico en el SDLC. Por ejemplo, las vulnerabilidades encontradas en el código fuente con el análisis del código fuente representan una medida inicial de riesgo. Esta medida de riesgo (por ejemplo, alto, medio, bajo) para la vulnerabilidad se puede calcular determinando la exposición y factores de riesgo, además, validando dicha vulnerabilidad con pruebas de intrusión. Las métricas del riesgo asociado a las vulnerabilidades encontradas con pruebas de seguridad permiten al negocio hacer decisiones sobre la gestión del riesgo, tales como decidir si un riesgo puede ser aceptado, mitigado o transferido a distintos niveles dentro de la organización (por ejemplo, negocio como técnico).

Al evaluar el nivel de seguridad de las aplicaciones, es importante tener en cuenta ciertos factores, tales como el tamaño de la aplicación en desarrollo. El tamaño de la aplicación se ha demostrado estadísticamente que se relaciona con el número de problemas encontrados con las pruebas. Una de las medidas del tamaño de la aplicación es el número de línea de código (LOC) de la aplicación. Generalmente, los defectos de calidad del software van desde unos 7 a 10 defectos por cada mil líneas de código nuevo y modificado[26]. Dado que las pruebas pueden reducir el número global de alrededor del 25% con una prueba por sí sola, es lógico que las aplicaciones de mayor tamaño deben ser probadas mucho más y con frecuencia más que las aplicaciones de menor tamaño. Cuando las pruebas de seguridad se

[26] SECURITY ACROSS THE SOFTWARE DEVELOPMENT LIFECYCLE TASK FORCE, REFERRED DATA FROM CAPER JOHNS, Software Assessments, Benchmarks and Best Practices -[enlínea], [consultado el 23 de marzo de 2015]. Disponibleen: http://www. cyberpartnership.org/SDLCFULL.pdf

realizan en varias fases de la SDLC, los datos de las pruebas podrían demostrar la capacidad de las pruebas de seguridad en la detección de vulnerabilidades tan pronto como se hayan introducido, y demostrar la eficacia de la eliminación de ellos mediante la implementación de contramedidas en los diferentes puntos de control de la SDLC. Una medida de este tipo también es definido como medidas de contención y proporciona una medida de la capacidad de una evaluación de seguridad realizada en cada fase del proceso de desarrollo para mantener la seguridad dentro de cada fase. Estas medidas de contención son también un factor crítico en la reducción del coste en solucionar vulnerabilidades, ya que es menos costoso hacer frente a las vulnerabilidades cuando se encuentren (en la misma fase de la SDLC), en lugar de la solucionarlos más adelante en otra etapa. Métricas de las pruebas de seguridad permiten el análisis del riesgo de seguridad, el coste y la gestión de defectos cuando es asociado con objetivos tangibles y oportunos tales como:

- Reducir el número total de vulnerabilidades en un 30%.
- Los problemas de seguridad serán solucionados antes de un plazo determinado (por ejemplo, antes de la versión beta).

Los resultados de las pruebas de seguridad pueden ser absolutos, como el número de vulnerabilidades detectadas durante la revisión manual de código, así como comparativos, tales como el número de vulnerabilidades detectadas en la revisión de código frente a las pruebas de intrusión. Para responder a las preguntas acerca de la calidad del proceso de seguridad, es importante determinar una línea de base para lo que se podría considerar aceptable y bueno. Los resultados de las pruebas de seguridad también ayudan a los objetivos específicos del análisis de la seguridad, tales como el cumplimiento de las normas de seguridad y estándares de la seguridad de la información, la gestión de los procesos de seguridad, la identificación de las causas de seguridad y procesos de mejora, y los costes de la seguridad frente a los beneficios.

Cuando el resultado de las pruebas de seguridad es reportado, deben proporcionar métricas para apoyar el análisis. El alcance del análisis es la interpretación de los resultados de las pruebas para encontrar pistas sobre la seguridad del software que está siendo producido, así como la eficacia del proceso. Algunos ejemplos de pistas apoyadas por los resultados de las pruebas de seguridad pueden ser:

- ¿Las vulnerabilidades son reducidas a un nivel aceptable para el “release”?
- ¿Cómo el nivel de seguridad del producto se compara con productos similares de software?
- ¿Son todos los requerimientos de las pruebas de seguridad cumplidos?
- ¿Cuáles son las principales causas de los problemas de seguridad?

- ¿Cómo son de numerosos los fallos de seguridad frente a los errores de seguridad?
- ¿Qué actividad de seguridad es más efectiva para encontrar vulnerabilidades?
- ¿Qué equipo es más productivo solucionando defectos de seguridad y vulnerabilidades?
- ¿Qué porcentaje de todas las vulnerabilidades son de alto riesgo?
- ¿Qué herramienta es más efectiva detectando vulnerabilidades de seguridad?
- ¿Qué tipo de prueba de seguridad es más efectiva para encontrar vulnerabilidades (por ejemplo, pruebas de caja blanca frente a pruebas de caja negra?
- ¿Cuántos problemas de seguridad son encontrados durante la revisión de la seguridad en el código?
- ¿Cuántos problemas de seguridad son encontrados durante la revisión de la seguridad del diseño?

Con el fin de hacer un buen juicio utilizando los resultados de las pruebas, es importante tener un buen conocimiento del proceso de pruebas así como de las herramientas. Una taxonomía de herramientas debería ser adoptada para decidir qué herramientas de seguridad deben ser usadas. Herramientas de seguridad pueden ser calificadas como buenas para encontrar vulnerabilidades conocidas y comunes orientadas a los distintos elementos del software. La cuestión es que los problemas de seguridad desconocidas no son probados. El hecho de que la aplicación salga limpia no significa que sea buena.

Se ha demostrado que en las mejores herramientas se puede encontrar poco menos de la mitad de la cantidad total de vulnerabilidades. Incluso las más sofisticadas herramientas automáticas no encajan con un ingeniero de pruebas de seguridad experimentado. Confiando en los resultados de éxito de la pruebas de seguridad de herramientas automáticas dan a los profesionales una falsa sensación de seguridad. Generalmente, los más experimentados ingenieros de pruebas de seguridad usan una metodología y las herramientas de pruebas, en este caso, los mejores resultados y un buen análisis serán obtenidos. Es importante que los administradores inviertan en herramientas de pruebas de seguridad, también se consideran una buena inversión la contratación de recursos humanos cualificados, así como formación en pruebas de seguridad.

2.1.8.2 Requerimientos en la presentación de informes. El nivel de seguridad de una aplicación puede ser caracterizada desde el punto de vista de los efectos, tales como el número de vulnerabilidades y el nivel de riesgo de las vulnerabilidades, así como desde el punto de vista de la causa (es decir, el origen), como los errores de codificación, defectos arquitectónicos y problemas de configuración. Las vulnerabilidades pueden ser clasificadas de acuerdo a diferentes criterios. Podría ser una clasificación estadística, como la OWASP Top 10 o la WASC Web Security Statistics project, o relacionados con los controles de

defensa como en el caso de la categorización WASF (marco de seguridad de aplicaciones web). Al reportar los resultados de las pruebas de seguridad, la mejor práctica es incluir la siguiente información, además de la categorización de cada tipo de vulnerabilidad:

- La amenaza a la seguridad de que el problema expone.
- La causa raíz del problema de seguridad (por ejemplo, errores de seguridad, fallo de seguridad).
- La técnica de prueba usada para encontrarla.
- La remediación de la vulnerabilidad (por ejemplo, la contramedida).
- La calificación de riesgo de la vulnerabilidad (alta, media, baja).

Al describir lo que es la amenaza de seguridad, es posible comprender por qué el control no es eficaz en la mitigación de la amenaza Presentado la causa raíz del problema puede ayudar a identificar lo que necesita ser solucionado. En el caso de pruebas de caja blanca, por ejemplo, la causa raíz de la seguridad en el software será el código fuente problemático. Una vez informado los problemas, también es importante proporcionar orientación a los programadores de software sobre la forma de volver a probar y encontrar las vulnerabilidades. Esto podría implicar la técnica de pruebas caja blanca (por ejemplo, la revisión de seguridad del código con una analizador estático de código) para encontrar si el código es vulnerable. Si una vulnerabilidad es encontrada a través de técnicas de pruebas de caja negra (prueba de intrusión), el informe de la prueba también debe proporcionar información sobre cómo validar la exposición de la vulnerabilidad al usuario (por ejemplo, el cliente). La información acerca de cómo solucionar la vulnerabilidad debe ser lo suficientemente detallada para un programador para implementar una solución. Debería proporcionar ejemplos de codificación segura, cambios de configuración y proporcionar adecuadas referencias. Por último, la calificación del riesgo ayuda a priorizar el esfuerzo de remediación. Generalmente, la asignación de una calificación de riesgo a la vulnerabilidad implica un análisis de riesgo basado en factores tales como el impacto y la exposición.

2.1.9 El Top 10 OWASP. Las detecciones pueden ser online o en entorno de laboratorio, pudiendo arrojar cualquiera de los dos, los resultados esperados, basándose en los controles de OWASP, los resultados esperados a encontrar pueden ser.

- A1 – Inyección.
- A2 – Pérdida de autenticación y gestión de sesiones.
- A3 – Secuencia de comandos en sitios cruzados (XSS).
- A4 – Referencia directa insegura a objetos.
- A5 – Configuración de seguridad incorrecta.
- A6 – Exposición de datos sensibles.
- A7 – Ausencia de control de acceso a las funciones.

- A8 – Falsificación de peticiones en sitios cruzados (CSRF).
- A9 – Uso de componentes con vulnerabilidades conocidas.
- A10 – Redirecciones y reenvíos no válidos.

Los riesgos mencionados en la lista anterior son los más encontrados en las aplicaciones web y su objetivo es crear conciencia acerca de la seguridad en aplicaciones mediante la identificación de algunos de los riesgos más críticos que se encuentran en las organizaciones y serán la base de este trabajo.

2.2 MARCO INSTITUCIONAL

Desde 1995, se funda el Centro de Audición y lenguaje con servicios de audiología básica, electrofisiología auditiva y amplificación, en la ciudad de Pasto (Nariño). Dirigido por una integrante de la primera promoción de la Especialización en audiología[27].

Desde el año de su fundación Audiocom IPS ha crecido para ser líder en el área de la audiología, en el 2002 se funda su sede en Medellín y 4 años más tarde en el 2006 inaugura su sede principal ubicada en Bogotá, en el 2009 moderniza su unidad de producción y obtiene la certificación CCAA (certificado de Capacidad de Almacenamiento y Acondicionamiento), la cual es requerida por el INVIMA para la importación de los audífonos, en el 2010 abre su programa de audiología pediátrica llamada Audiokids, ya con los avances tecnológicos y la experiencia obtenida durante los años de servicio en el 2012 abre el primer centro de evaluación y reentrenamiento de Tinnitus y Vértigo y a su vez crea el departamento de metrología para mantenimiento y calibración de equipos médicos audiológicos, en la actualidad tiene la representación de los audífonos SIEMENS y a través de su departamento de desarrollo actualiza el software de las historias clínicas subiéndolo a la nube, haciéndolo disponible las 24 horas de los 7 días de la semana, a su vez crea su propio departamento de atención al cliente a través de su propio call center a nivel nacional y centralizado en Bogotá y crea un programa propietario para la atención a los usuarios de ARL.

Los audilógos de Audiocom IPS están siendo apoyados constantemente por profesionales de la dirección científica, ingenieros de sistemas, diseñadores gráficos, abogados, ingenieros biomédicos, arquitectos, contadores y especialistas en la calidad de la atención en salud, dando la garantía del trabajo en equipo, no es solo en la comunidad científica médica, si no demostrando que es necesario tener el apoyo de diferentes áreas para prestar un servicio mejor y estandarizado.

---

[27] IPS AUDIOCOM [Web en línea]. [en línea], [consultado el 23 de marzo de 2015]. Disponible en: http://www.audiocom-ips.com [Consulta: 01-06-2014]

Audiocom IPS, presta sus servicios audiológicos en todo el país, dirigida por profesionales capacitados y entrenados los cuales proporcionan la atención necesaria para una atención eficaz y segura a todos los usuarios.

La empresa se basa en pilares como lo son la precisión, la honestidad y la calidad de servicio y estos a su vez son la base para prestar los servicios a los pacientes

La IPS presta sus servicios orientando a pacientes con desórdenes audio-vestibulares, contando con los expertos en las técnicas existentes para atender y brindar un diagnóstico y una solución completa, Audiocom IPS es consciente del avance de la tecnología, pero a su vez sabe y conoce que la labor humana es fundamental para el manejo de pacientes. Los expertos de la IPS tienen como requisito escencial una especialización en audiología y reciben capacitación nacional e internacional relacionada con la atención de usuarios. Además de tener las capacidades técnicas y el conocimiento para ejercer su labor, los médicos de Audiocom IPS tienen características que hacen que la empresa crezca cada vez más. Habilidades como el trabajo en equipo, la seguridad en el manejo de la información, la conciencia situacional y la comunicación de liderazgo hacen que se controle mejor situaciones adversas y cualquier eventualidad que se presente.

Las historias clínicas son de uso privado y son documentos confidenciales, los cuales serán accedidos únicamente por personal autorizado, como el médico tratante y el usuario al cual pertenece, y solo puede ser visto por terceros con previa autorización del paciente o en casos explícitos y previstos por la ley, en caso de ser solicitado por terceros deberá seguirse una serie de pasos y requisitos para poder hacer efectiva la solicitud, la historia clínica del paciente está protegida en Colombia por la Ley Estatutaria 1581 de 2012 Reglamentada parcialmente por el Decreto Nacional 1377 de 2013. Que la Ley 1581 de 2012 constituye el marco general de la protección de los datos personales en Colombia.

Es aquí donde radica la importancia de proteger la información de los pacientes, la información de las historias clínicas está protegida por la ley 1581 de 2012 (Ley de protección de datos), ya que la información es sensible y podría afectar la reputación del paciente, clínica u hospital y el médico tratante, debido a que se violarían fundamentos de la seguridad informática en sus pilares principales, confidencialidad, integridad y disponibilidad, la ley obliga a todos los campos de la salud a proteger las historias clínicas de los pacientes.

Audiocom es consciente de esta realidad y esta cobijada en esta ley, Audiocom IPS en su constante mejoramiento quiere brindar seguridad a los pacientes y dar lo mejor de sí para lograr ser una de las mejores y reconocidas instituciones del país. Hoy en día cualquier corporación o empresa tiene un contacto directo con sus empleados, clientes y usuarios del común a través de la tecnología y una de las formas más fáciles de llegar a las personas, es a través de los aplicativos web, donde un usuario podría conocer la información sobre la empresa, sobre sus

empleados y si le es posible sobre (en este caso en particular) su historia clínica, teniendo acceso a ella.

2.3 MARCO TECNOLÓGICO

La información de los expedientes clínicos dejó de encontrarse exclusivamente en folios y carpetas físicas por lo cual se debe considerar el alcance de las medidas de protección y los desarrollos tecnológicos que satisfagan las necesidades de controlar la trazabilidad de los registros clínicos de los pacientes, por lo cual se deben evaluar con detalle las medidas de seguridad aplicadas en la programación de aplicaciones web que almacenan información clínica.

Los datos clínicos se encuentran entre los datos que más pueden afectar a la intimidad de las personas por lo que deben ser objeto de una adecuada política de salvaguarda de la privacidad. De hecho, la Ley de Protección de Datos los incluye entre los que precisan de un mayor nivel de reserva. La información permite un mayor grado de protección de la información que las historias en papel, y es responsabilidad de las instituciones sanitarias garantizar las máximas medidas de seguridad, pero estos sistemas de custodia pueden resultar inútiles sino se acompañan de la adopción de una serie de hábitos por parte de los usuarios.

En concreto, el software que actualmente desarrolla la institución prestadora de servicios de salud especializada en audiología, Audiocom, identifica a los profesionales mediante la combinación de nombre de usuario y clave de acceso, lo que permite al sistema asignarles un perfil que les autoriza el acceso a un espectro definido de la información incluida en el mismo. Sin embargo, existen otros riesgos técnicos en el desarrollo de la aplicación que comprometen la custodia y el secreto de los datos, por lo cual no se puede garantizar que una persona no autorizada pueda visualizar la información que allí se almacena.

En el caso particular de Audiocom, existe un equipo de programación que trabaja desarrollando la aplicación web para la digitalización de historias clínicas. Sin embargo, este no utiliza metodologías de desarrollo adecuadas y tampoco cuenta con mucha experiencia en el desarrollo de los mismos, por lo que se sabe que este encuentra en riesgo permanente sobre todo por su exposición permanente a millones de atacantes en Internet.

2.4 MARCO LEGAL

La principal ventaja del uso de los computadores en medicina es el acceso rápido a la información. Sin embargo, la facilidad para localizar datos es al mismo tiempo su máxima debilidad. En principio, la información clínica solo debe estar disponible para aquellos interesados en el cuidado médico del paciente; no obstante, la información puede estar a disposición de personal de salud que no la necesita, o abierta a terceros que pueden usarla para infringir daño físico, emocional o

financiero al paciente. Por tanto, a los objetivos clínicos de un sistema se le deben añadir y verificar características de seguridad que permitan garantizar el mejoramiento de la atención, accediendo a la información oportunamente y protegiendo la confidencialidad a través de la restricción del acceso.

Revisando la normatividad vigente se encuentra que en la ley 1438 de 2011 en el parágrafo transitorio dice textualmente que la historia clínica única electrónica será de obligatoria aplicación antes del 31 de Diciembre del año 2013. Esta tendrá plena validez probatoria[29]. Por otro lado, la ley estatutaria 1581 de 2012 por la cual se dictan disposiciones para la protección de datos personales, decreta en el artículo 10 numeral c) que la autorización del paciente para el registro en la base de datos en su atención médica solo dejará de ser necesaria en casos de urgencia médica o sanitaria.

En el mundo del software, entender el marco jurídico que regula la propiedad intelectual e industrial es fundamental para conocer bajo qué condiciones se pueden ceder los programas informáticos o utilizar los de terceros. Al ser usuarios, es importante comprender qué derechos y obligaciones se reciben al adquirir una determinada aplicación o paquete, y si se es creador hay que entender cómo ceder los programas a los usuarios y qué derechos y obligaciones se les está ofreciendo[30].

[29] (2011, Enero 19). Ley 1438 de 2011 [Online]. Disponible en: http://www.unipamplona.edu.co/unipamplona/portalIG/home_54/recursos/01general/04122012/ley_1438_2011.pdf

[30] MAS, Jordi (2005). «Marco jurídico y oportunidades de negocio en el software libre». UOC Papers [artículo en línea]. N.º 1. UOC. [Fecha de consulta: dd/mm/aa]. ISSN 1885-1541

3. DISEÑO METODOLÓGICO

3.1 HIPÓTESIS DE INVESTIGACIÓN

La metodología de pruebas OWASP para la identificación de vulnerabilidades en la aplicación para la gestión de historias clínicas electrónicas de Audiocom IPS es un elemento fundamental para identificar las diferentes debilidades derivadas de la arquitectura de la aplicación para lograr la optimización de un método eficiente y seguro que proteja la información vital y confidencial de los usuarios para dar cumplimiento a la ley de protección de datos exigida para la habilitación de los servicios de salud.

3.2 VARIABLES

Variable independiente**:** Detección, generación.
Variable dependiente**:** Vulnerabilidades, amenazas, riesgos.
Variable cuantitativa: Software, historia clínica.
Variable cualitativa: IPS, audiología.

3.3 METODOLOGÍA

Con el fin de cumplir los objetivos propuestos en el proyecto se definieron cuatro fases de actividades específicas, orientadas a realizar con éxito la detección de vulnerabilidades del aplicativo web de historias clínicas de Audiocom IPS para la generación de recomendaciones. Las fases que se consideraron fueron:

- Planeación.
- Recolección de información.
- Detección de vulnerabilidades.
- Desarrollo del ASVS.
- Generación de recomendaciones.

En la fase de planeación se proyectó la realización de entrevistas a los miembros del equipo de desarrollo web de Audiocom IPS. En este espacio se discutiría en términos generales las diferentes actividades relacionadas en el proceso realizado para la programación del software y la definición del cronograma de los módulos entregables.

En la fase 2 se plantea la recolección de información operativa suministrada por la gerencia e información técnica suministrada por el departamento de desarrollo.

Posterior al levantamiento de información se realiza la detección de vulnerabilidades, procediendo con las técnicas descritas en la guía de pruebas de

Owasp y el top 10 de la misma como parte de una verificación exhaustiva de la aplicación.

En la siguiente fase se ejecuta el desarrollo del estándar de verificación y aseguramiento de aplicaciones (ASVS) con lo que se normaliza la gama de cobertura de seguridad y el nivel de rigor disponible a la hora de realizar la verificación de la seguridad de la aplicación web, con ello se establece un nivel de confianza.

Al culminar las fases anteriores se proyectó la generación de las recomendaciones basadas en la evidencia recolectada para el mejoramiento de la aplicación y su nivel de confianza.

4. DETECCIÓN DE VULNERABILIDADES EN EL APLICATIVO WEB

A continuación se presentan los hallazgos de las pruebas de vulnerabilidad realizadas sobre la aplicación web para la gestión de pacientes e historia clínica de Audiocom IPS, basados en la versión 3.0 de la guía de pruebas de OWASP, para ello se han trabajado pruebas de caja negra y de caja gris, dependiendo de los fragmentos de código suministrados por el equipo de desarrollo de Audiocom IPS, quienes autorizaron realizar el análisis de vulnerabilidades, facilitando los recursos necesarios, con el fin de buscar, evidenciar y evaluar las brechas potenciales y vulnerabilidades que presenta actualmente su aplicativo web, de tal forma que se generen recomendaciones para remediarlas y se contribuya en la mitigación de los riesgos asociados, ya que hasta el momento, la IPS no ha trabajado en ningún tipo de control de seguridad.

A1 - Inyección. Las pruebas que se extraen de la guía de OWASP para verificar este ítem son las siguientes:

OWASP-DV-005 Inyección SQL.
OWASP-DV-012 Inyección de código.
OWASP-DV-013 Inyección de órdenes del sistema operativo.

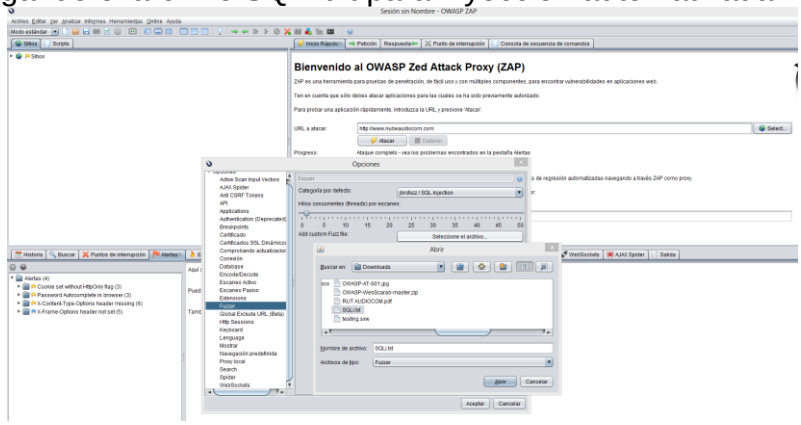
Verificando OWASP-DV-005. Se habla de pruebas de Inyección SQL cuando se intenta inyectar una determinada consulta SQL directamente en la Base de Datos. Se encuentra una vulnerabilidad SQL si la aplicación utiliza entradas de usuario para crear consultas SQL sin que la aplicación haga una validación adecuada de los datos. Una explotación exitosa de ésta clase de vulnerabilidad permite a un usuario no autorizado acceder o manipular datos en la base de datos. El objetivo es manipular los datos en la base de datos, un recurso vital para todas las empresas.

Para realizar esta prueba se utiliza la herramienta ZAP y en ella se cargan las siguientes consultas que se inyectan en un archivo de texto llamado SQLi.txt. Como evidencia se puede ver que la figura 5 muestra la carga del archivo SQLi.txt para la inyección automatizada y la figura 6 muestra la respuesta negativa del servidor a la inyección de SQL.

order by	asc
desc	delete
update	distinct
having	truncate
replace	like
handler	bfilename
or username like %	oruname like %
oruserid like %	execxp
execsp	; exec master..xp_cmdshell

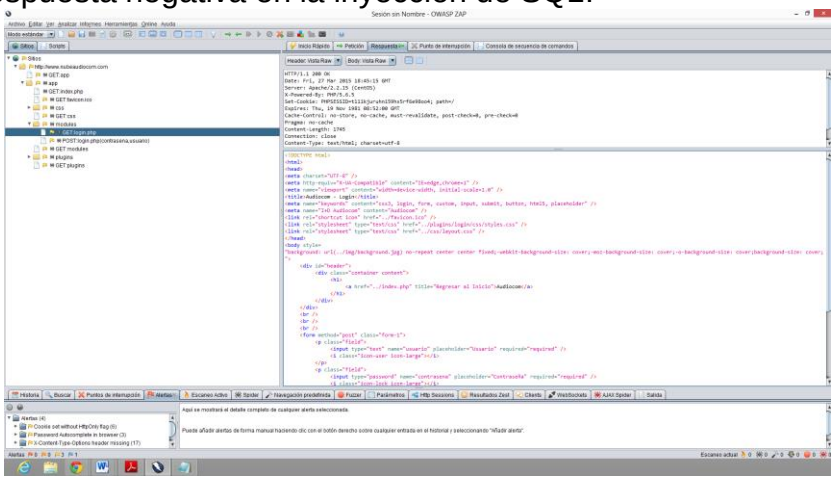
; execxp_regread	t'exec master..xp_cmdshell
'nslookup www.google.com'-	#¿NOMBRE?
\x27UNION SELECT	UNION SELECT
UNION ALL SELECT	or (EXISTS)
(select top 1	||UTL_HTTP.REQUEST
1;SELECT%20*	to_timestamp_tz
tz_offset	<>"'%;)(&+
%20or%201=1	%27%20or%201=1
%20$(sleep%2050)	%20'sleep%2050'
char%49%41%2b%40SELECT	'%20OR
sqlattempt1	(sqlattempt2)
'or 'x'='x	

Figura 5. Cargando el archivo SQLi.txt para inyección automatizada.



Fuente: autores

Figura 6. Respuesta negativa en la inyección de SQL.

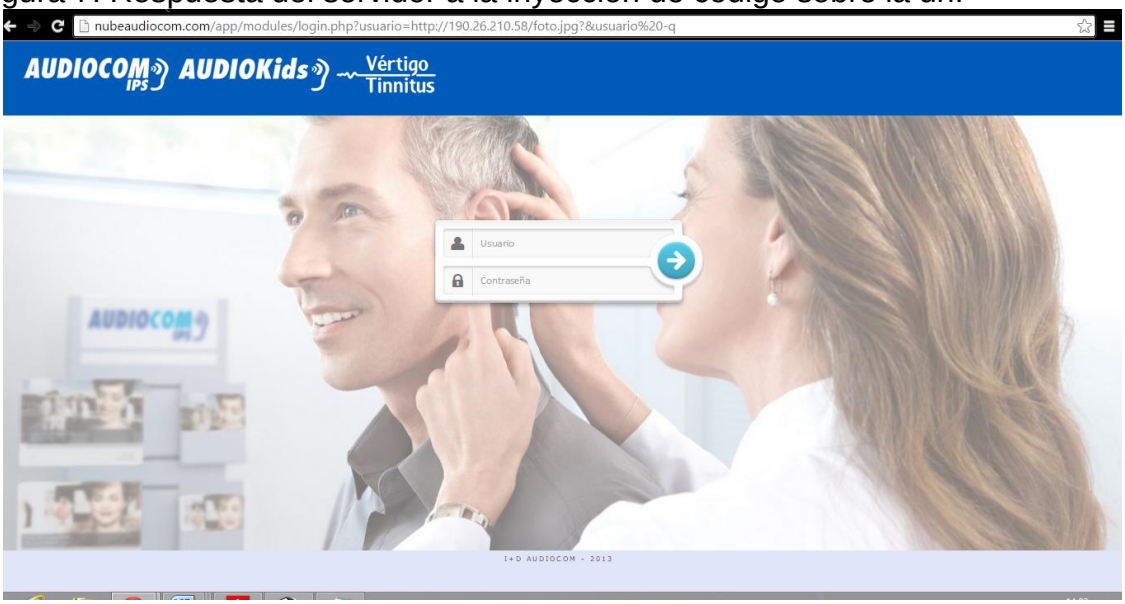


Fuente: autores

Con la herramienta automatizada no se obtuvo resultados que reflejen posibles vulnerabilidades de inyección SQL.

Verificando OWASP-DV-012. Con esta prueba se pretende comprobar si es posible introducir el siguiente código como entrada en la aplicación y que éste lo ejecute el servidor: http://nubeaudiocom.com/app/modules/login. php?usuario= http://190.26.210.58/foto.jpg?&usuario%20-q. Se ha verificado que independientemente del código introducido, la aplicación toma como indiferentes todos los parámetros inesperados. La figura 7 muestra la respuesta del servidor a la inyección de código sobre la URL.

Figura 7. Respuesta del servidor a la inyección de código sobre la url.

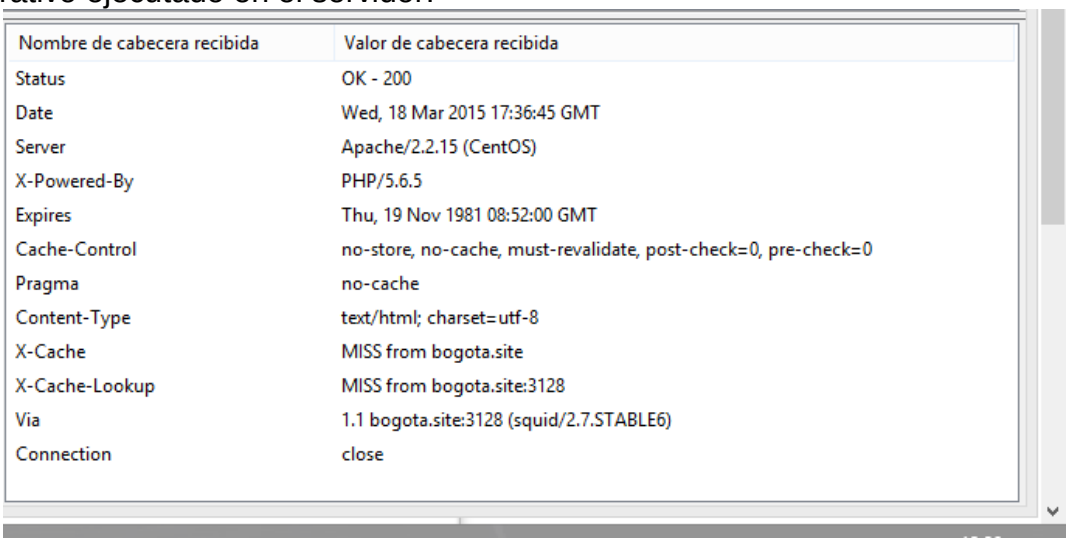


Fuente: autores

Verificando OWASP-DV-013. Con esta prueba se describe cómo comprobar si la aplicación es vulnerable a la inyección de órdenes del sistema operativo.

Interceptando el encabezado de una petición al servidor se puede saber que éste corre sobre el sistema operativo Centos (figura 8) y con esto se saben los comandos que pueden tratar de inyectarse de manera lógica. La figura 8 muestra el encabezado de la aplicación interceptado para identificar el sistema operativo ejecutado en el servidor y la figura 9 muestra la respuesta del servidor a inyección de código.

Figura 8. Encabezado de la aplicación interceptado para identificar el sistema operativo ejecutado en el servidor.



Nombre de cabecera recibida	Valor de cabecera recibida
Status	OK - 200
Date	Wed, 18 Mar 2015 17:36:45 GMT
Server	Apache/2.2.15 (CentOS)
X-Powered-By	PHP/5.6.5
Expires	Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control	no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma	no-cache
Content-Type	text/html; charset=utf-8
X-Cache	MISS from bogota.site
X-Cache-Lookup	MISS from bogota.site:3128
Via	1.1 bogota.site:3128 (squid/2.7.STABLE6)
Connection	close

Fuente: autores

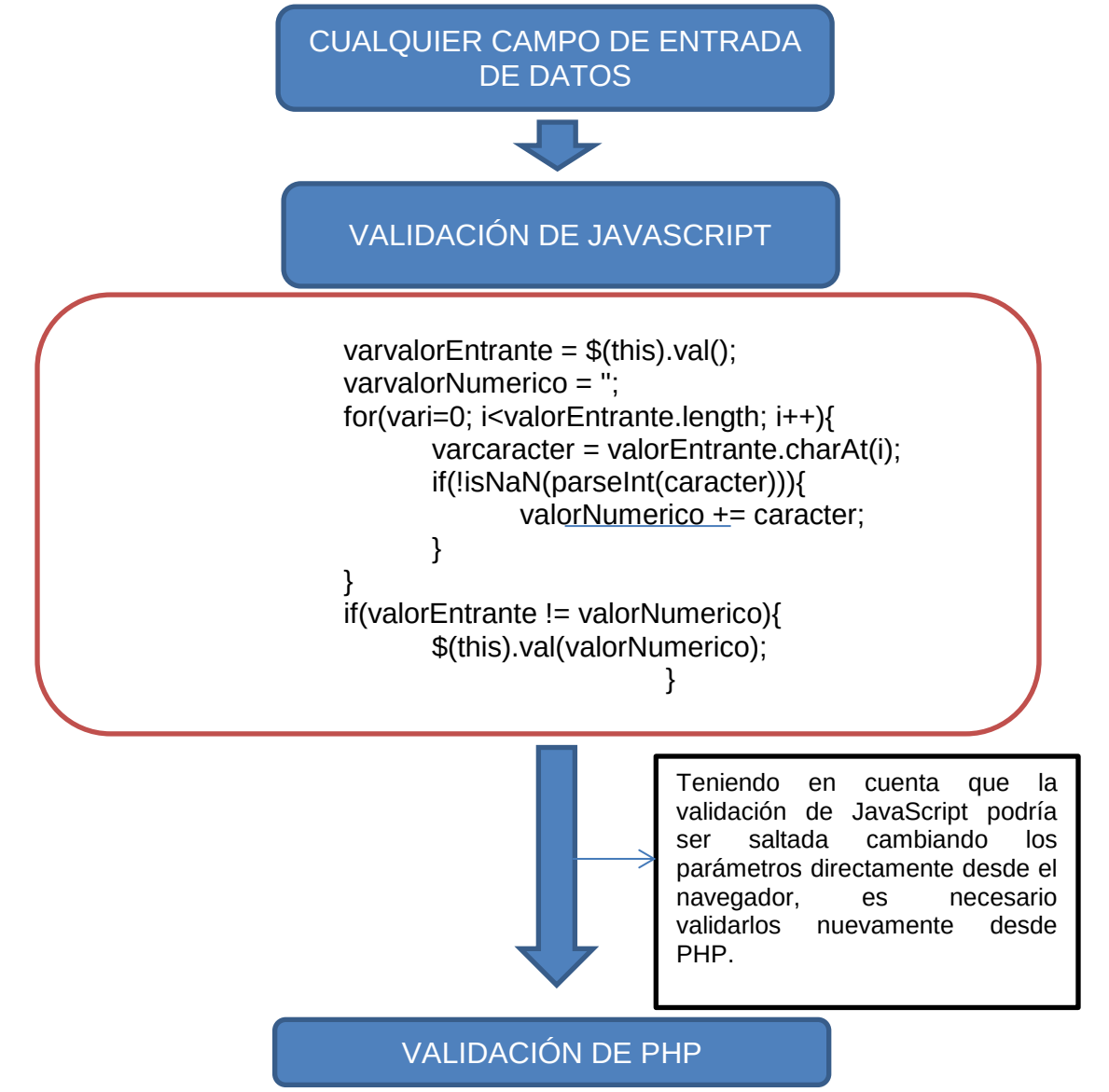
Teniendo en cuenta lo anterior y que la aplicación fue desarrollada en PHP, se ha escrito punto y coma al final de la URL seguida de una orden del sistema operativo esperando si este la ejecutaba o no, con lo cual se obtuvo que la respuesta del servidor fuera indiferente.

Figura 9. Respuesta del servidor a inyección de código.

Fuente: autores

Para finalizar la prueba, se muestra el diagrama de flujo que describe cómo se filtran los todos los campos de ingreso de datos a la aplicación mediante recursos de programación. Para entenderlo de mejor forma, la figura 10 muestra la validación que se realiza con JavaScript y PHP para filtrar los parámetros no deseados en el ingreso de datos a la aplicación.

Figura 10. Validación JavaScript – PHP.



Fuente: autores

A2 - Pérdida de Autenticación y Gestión de Sesiones. Las pruebas que se extraen de la guía de OWASP para verificar este ítem son las siguientes:

OWASP-AT-001 Transporte de credenciales sobre un canal cifrado.
OWASP-AT-002 Pruebas para la enumeración de usuarios.
OWASP-AT-003 Pruebas para cuentas de usuario posibles.
OWASP-AT-004 Pruebas de fuerza bruta.
OWASP-AT-006 Pruebas para contraseñas recordadas vulnerables y su reinicio.
OWASP-AT-007 Pruebas para el cierre de sesión y la gestión del cache del explorador.
OWASP-SM-002 Pruebas de atributos de las cookies.
OWASP-SM-003 Pruebas de fijación de sesión.

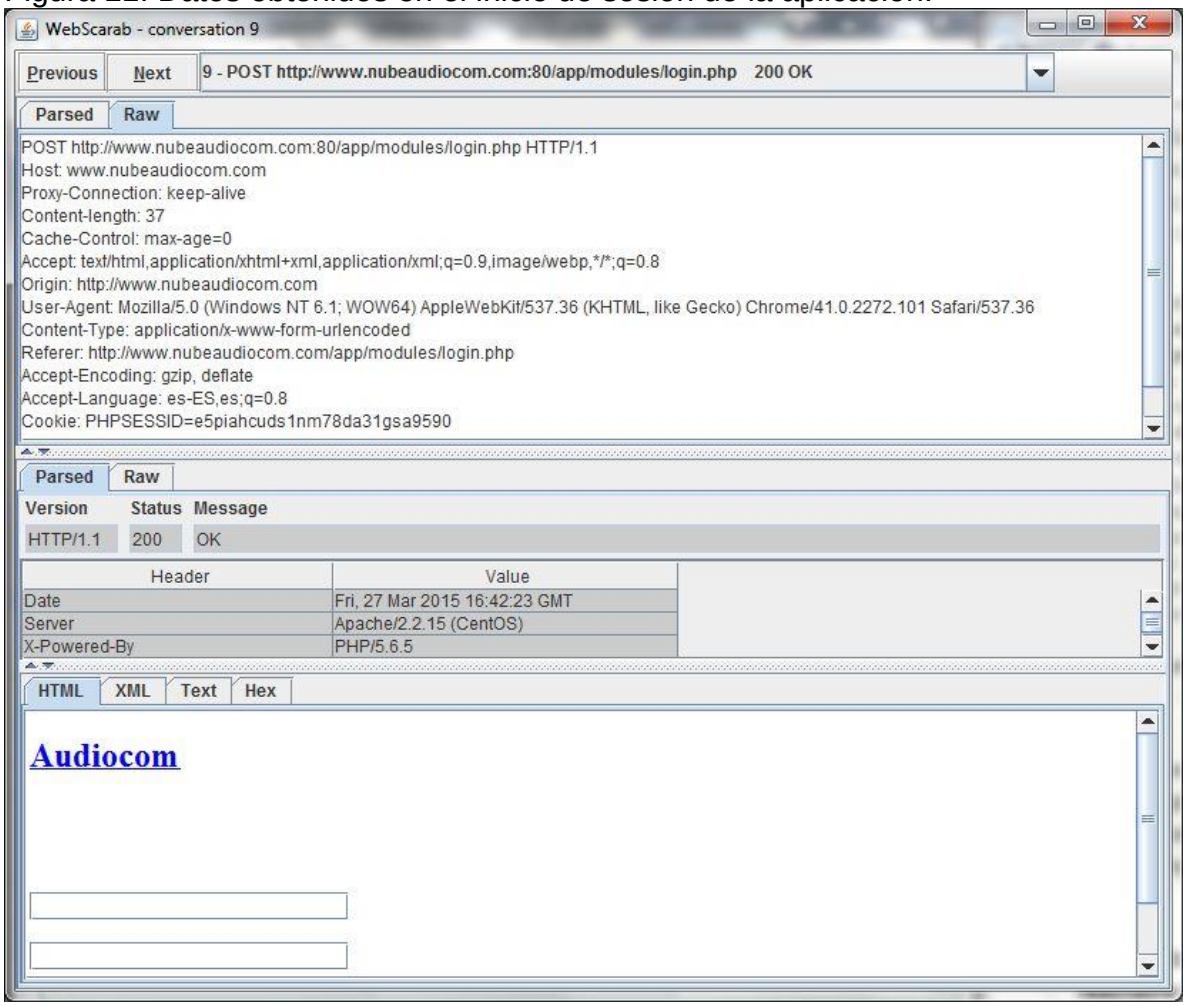
Para estas pruebas primero se debe averiguar si la información que introducen los usuarios en los formularios, con el fin de autenticarse en la aplicación, es transmitida utilizando protocolos seguros que la protegen de un atacante o no.

Con base en la información obtenida del equipo de desarrollo de Audiocom IPS, se sabe que en la aplicación las credenciales de los usuarios se almacenan cifradas en un algoritmo whirpool con hash de 512 bits, lo que es presentado en 128 caracteres hexadecimales. Sin embargo, la aplicación carece de un certificado de seguridad, por lo que prácticamente se pierde el trabajo de cifrado, ya que las claves quedan en texto plano para el atacante y esto se puede verificar interceptando los datos enviados por el usuario a la aplicación por medio de la interceptación de peticiones.

En este punto, la persona a cargo de las pruebas intenta averiguar si la información que introducen los usuarios en un formulario web, con el fin de poder autenticarse en la aplicación, es transmitida utilizando protocolos seguros que la protegen de un atacante o no.

Verificando OWASP-AT-001. En la figura 11 pueden verse los datos obtenidos en el inicio de sesión de la aplicación, que fueron extraídos de la página de inicio, donde se muestra un formulario con los campos usuario, contraseña y un botón de inicio para la autenticación de acceso. Si se observa el encabezado del pedido de la figura anterior, capturado con la herramienta ZAP de OWASP, se entiende que el método POST envía los datos a la página www.nubeaudiocom.com/app/modules/login.php simplemente utilizando HTTP. Por lo tanto los datos son transmitidos sin cifrado y un usuario malicioso puede leer el usuario y contraseña monitoreando la red con una herramienta como la utilizada anteriormente.

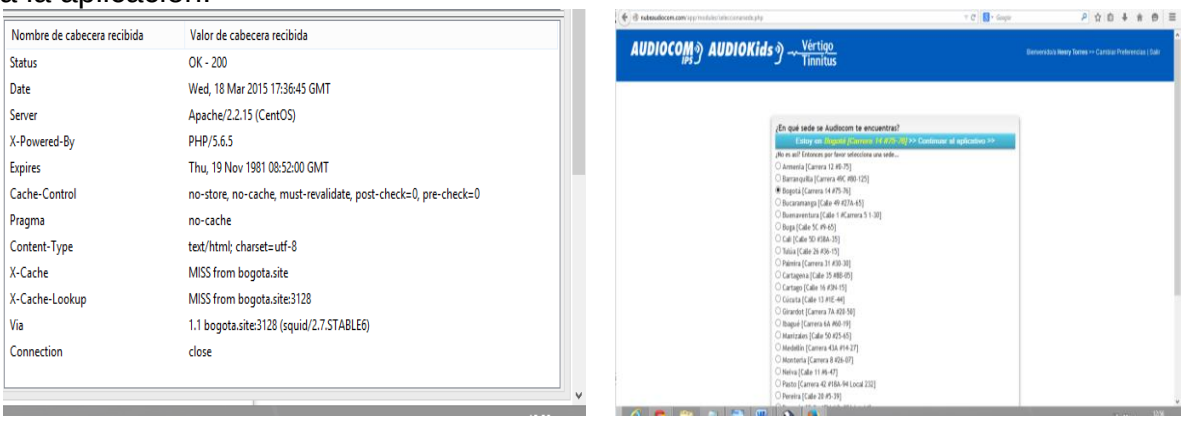
Figura 11. Datos obtenidos en el inicio de sesión de la aplicación.



Fuente: autores

Verificando OWASP-AT-002. Para probar la enumeración de usuarios se verifica si es posible recolectar un conjunto válido de usuarios simplemente interactuando con los mecanismos de autenticación de la aplicación. Esto luego es útil para realizar ataques de fuerza bruta. Las pruebas se pueden ver como sigue: la figura 12 muestra la respuesta del servidor cuando se envían usuario y contraseña válidos a la aplicación; la figura 13 muestra la respuesta del servidor a usuario existente y contraseña inválida y la figura 14 muestra la respuesta del servidor a usuario y contraseña incorrectos.

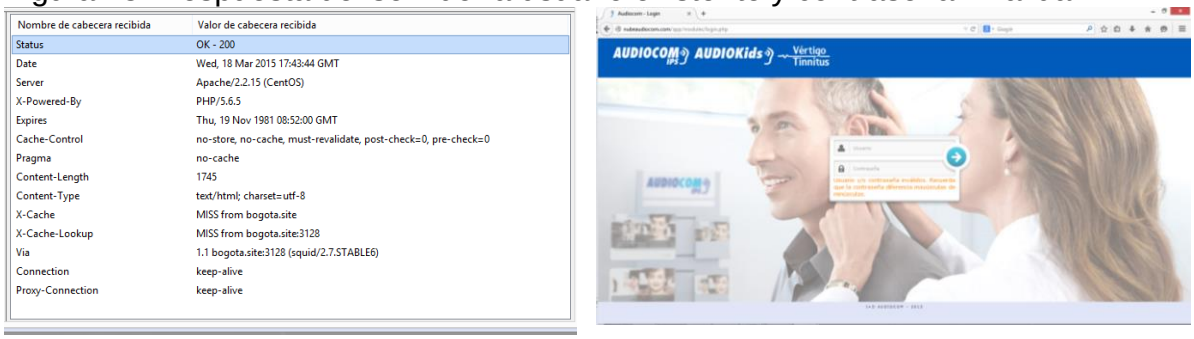
Figura 12. Respuesta del servidor cuando se envían usuario y contraseña válidos a la aplicación.



Nombre de cabecera recibida	Valor de cabecera recibida
Status	OK - 200
Date	Wed, 18 Mar 2015 17:36:45 GMT
Server	Apache/2.2.15 (CentOS)
X-Powered-By	PHP/5.6.5
Expires	Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control	no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma	no-cache
Content-Type	text/html; charset=utf-8
X-Cache	MISS from bogota.site
X-Cache-Lookup	MISS from bogota.site:3128
Via	1.1 bogota.site:3128 (squid/2.7.STABLE6)
Connection	close

Fuente: autores

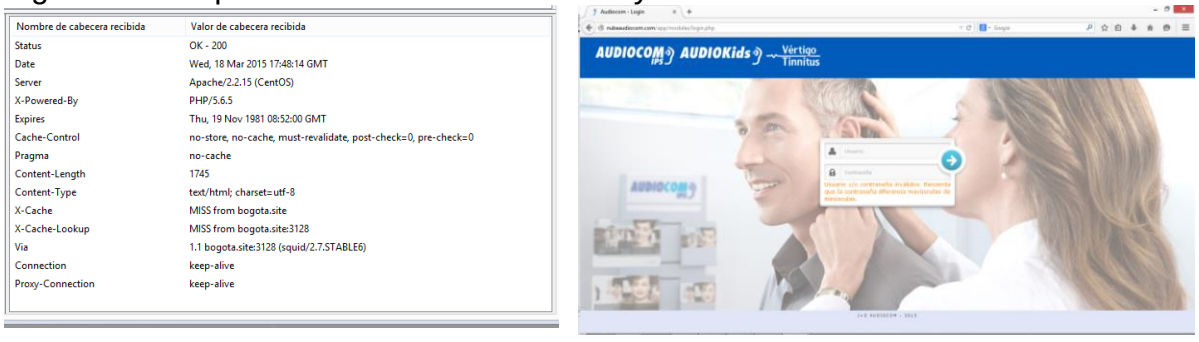
Figura 13. Respuesta del servidor a usuario existente y contraseña inválida.



Nombre de cabecera recibida	Valor de cabecera recibida
Status	OK - 200
Date	Wed, 18 Mar 2015 17:43:44 GMT
Server	Apache/2.2.15 (CentOS)
X-Powered-By	PHP/5.6.5
Expires	Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control	no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma	no-cache
Content-Length	1745
Content-Type	text/html; charset=utf-8
X-Cache	MISS from bogota.site
X-Cache-Lookup	MISS from bogota.site:3128
Via	1.1 bogota.site:3128 (squid/2.7.STABLE6)
Connection	keep-alive
Proxy-Connection	keep-alive

Fuente: autores

Figura 14. Respuesta del servidor a usuario y contraseña incorrectos.



Nombre de cabecera recibida	Valor de cabecera recibida
Status	OK - 200
Date	Wed, 18 Mar 2015 17:48:14 GMT
Server	Apache/2.2.15 (CentOS)
X-Powered-By	PHP/5.6.5
Expires	Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control	no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma	no-cache
Content-Length	1745
Content-Type	text/html; charset=utf-8
X-Cache	MISS from bogota.site
X-Cache-Lookup	MISS from bogota.site:3128
Via	1.1 bogota.site:3128 (squid/2.7.STABLE6)
Connection	keep-alive
Proxy-Connection	keep-alive

Fuente: autores

Verificando OWASP-AT-003. En el servidor la aplicación está programada para que no entregue información sobre los errores. Sin embargo, gracias a la información entregada por el equipo de desarrollo, se sabe que la política administrativa para la creación de usuarios es:

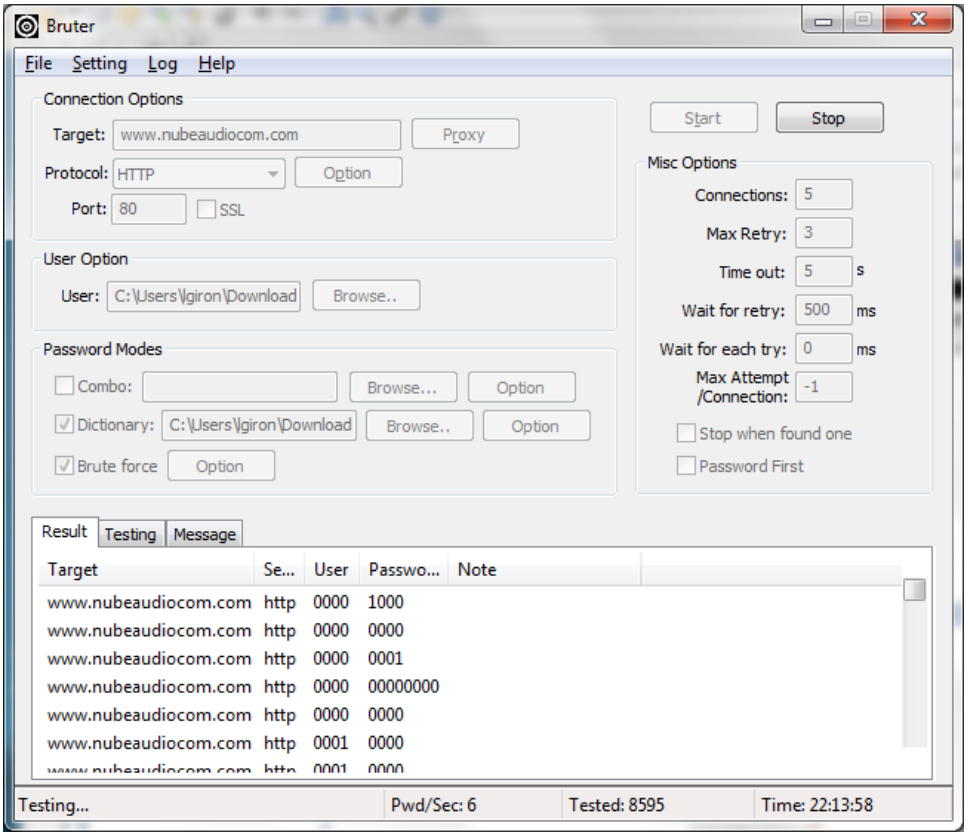
usuario = primer_letra_del_nombre + primer_apellido

contraseña = primer_letra_del_nombre + primer_apellido + 123

Con lo anterior se pueden adivinar los usuarios con algunos conocimientos básicos del personal asistencial que trabaja en la aplicación.

Verificando OWASP-AT-004. En la figura 15 se muestra el proceso realizado escaneando contraseñas con fuerza bruta. Esta prueba se realiza con la ayuda de la herramienta brutus y un diccionario de 500MB, que no lograron revelar credenciales válidas.

Figura 15. Escaneando contraseñas con fuerza bruta.



Fuente: autores

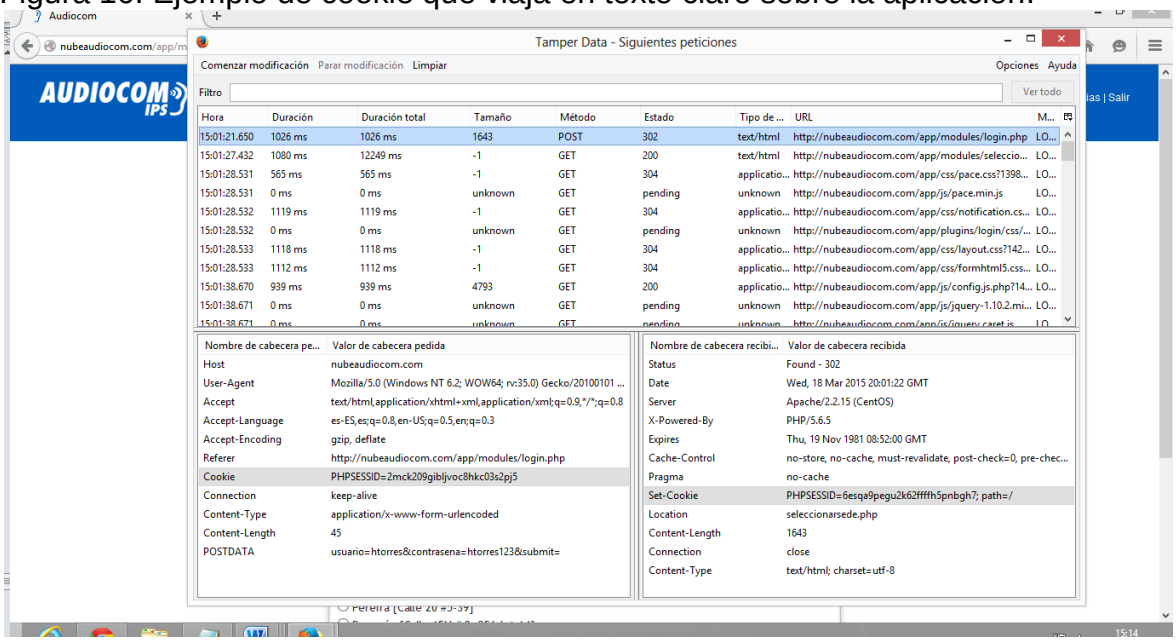
Verificando OWASP-AT-006. La aplicación no cuenta con la opción para el recordatorio de contraseñas, o pruebas de captcha y por tanto no es posible hacer esta comprobación.

Verificando OWASP-AT-007. Por medio del equipo de desarrollo se sabe que la aplicación está haciendo el manejo de sesiones propuesto en las guías de PHP. Sin embargo, se ha probado el manejo de caché en el navegador, verificando el cierre de sesión con el cierre del mismo y haciendo peticiones directamente sobre

la URL, para comprobar que no es posible ingresar a la aplicación por situaciones adversas o por mal manejo de la caché.

Para probar la manipulación de cookies, en la figura 16 se muestra cómo viaja la cookie en texto claro y por lo cual puede reemplazarse en otro equipo con el mismo navegador para establecer otra sesión con la misma cookie.

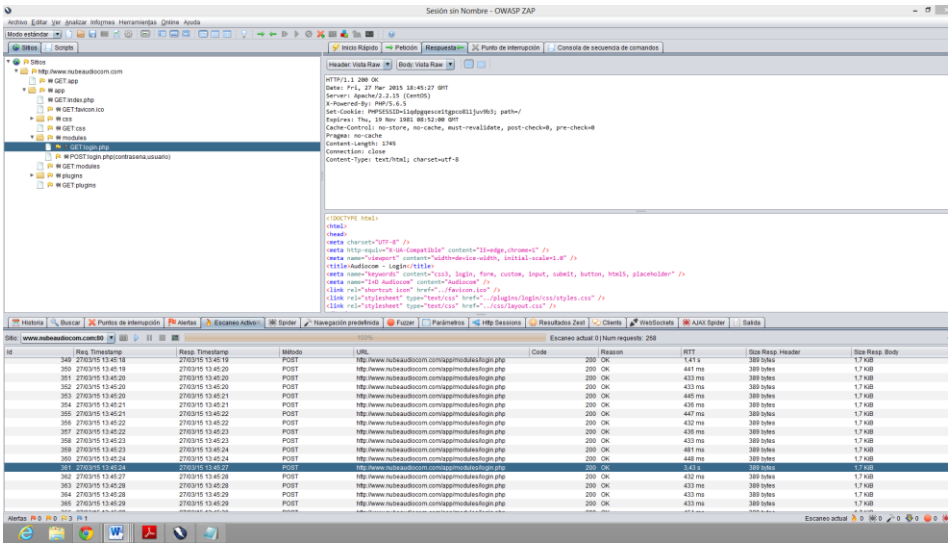
Figura 16. Ejemplo de cookie que viaja en texto claro sobre la aplicación.



Fuente: autores

Verificando OWASP-SM-002. Para esta prueba se usa la herramienta ZAP como proxy interceptor en el navegador con el fin de analizar las respuestas del servidor que establece las cookies. La figura 17 muestra la Interceptación del encabezado para verificar los parámetros de las cookies, con lo que se encuentra que ninguna de ellas contiene los atributos secure, HTTPOnly *o* expires y por lo cual se sabe que pueden ser fácilmente robadas o accedidas con un script local. Estas tampoco se aseguran con un tiempo de expiración.

Figura 17. Interceptación de encabezado para verificar los parámetros de las cookies.



Fuente: autores

Verificando OWASP-SM-003. Realizando una petición al sitio www.nubeaudiocom.com/modules/login.phpse obtiene la siguiente respuesta:

HTTP/1.1 200 OK
Date: Fri, 27 Mar 2015 18:45:27 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.6.5
Set-Cookie: PHPSESSID=i1qdpgqesce1tgpco811juv9b3; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 1745
Connection: close
Content-Type: text/html; charset=utf-8

Se observa que la aplicación establece un nuevo identificador de sesión para el cliente PHPSESSID=i1qdpgqesce1tgpco811juv9b3

Ahora se realiza una autenticación exitosa en la aplicación para observar la siguiente respuesta del servidor.

HTTP/1.1 200 OK
Date: Fri, 27 Mar 2015 18:48:09 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.6.5
Set-Cookie: PHPSESSID=u2fvorr4vccbkaf661tf0jf0q7; path=/

Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 1745
Connection: close
Content-Type: text/html; charset=utf-8

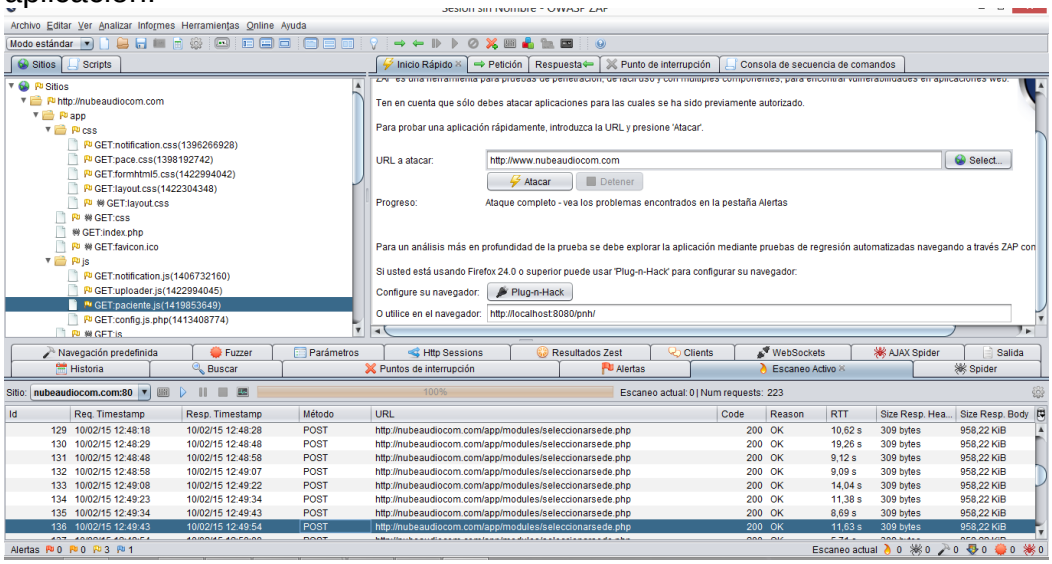
Se puede ver que se generó una nueva cookie después de la autenticación exitosa y por ello se sabe que no es posible realizar un robo de sesión.

A3 - Secuencia de comandos en sitios cruzados (XSS). Las pruebas que se extraen de la guía de OWASP para verificar este ítem son las siguientes:

OWASP-DV-001 Prueba de Cross site scripting reflejado.
OWASP-DV-002 Prueba de Cross site scripting almacenado.

Para detectar si la aplicación es vulnerable a cross site scripting se inyecta el siguiente script en cada una de las entradas de datos al sistema: <script>alert(“vulnerabilidad detectada”)</script> y se espera que se ejecute en el navegador del usuario revelando la vulnerabilidad. Esta operación se realiza con la ayuda de la herramienta ZAP de OWASP. La figura 18 muestra la implementación de la herramienta ZAP para la detección de cross site scripting.

Figura 18. Implementación de ZAP para la detección de cross site scripting en la aplicación.



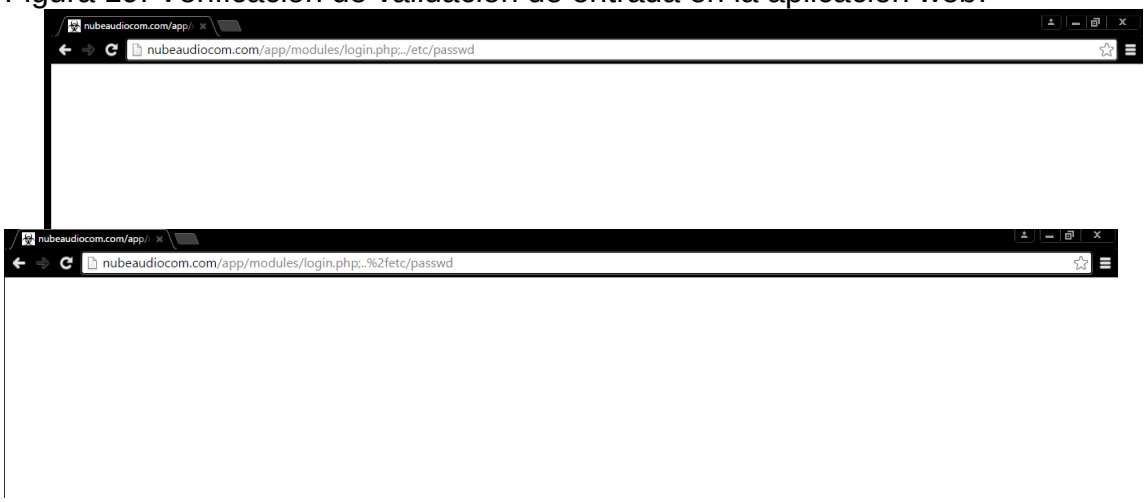
Fuente: autores

En un examen más profundo, se analiza el código para verificar que todos los campos de entrada de datos a la aplicación están filtrados para el fin dispuesto únicamente. Por ejemplo, los campos correspondientes al nombre del paciente,

que son primer nombre, segundo nombre, primer apellido y segundo apellido, solo admite cada uno un máximo de 15 caracteres que solo pueden ser letras y en general, ningún campo de entrada de datos puede contener caracteres especiales, lo cual es válido ya que la aplicación no lo requiere entre las reglas del negocio.

A4 - Referencia directa insegura a objetos. La aplicación web en cuestión solo maneja un tipo de usuario y por lo tanto todos ellos navegan por la aplicación con los mismos privilegios, de esta forma, se sabe que no se podrá dar una escalación de privilegios o un salto en el esquema de autorización. Todas las operaciones administrativas, como la creación de usuarios y asignación de algunos permisos especiales se hacen directamente sobre el código de la aplicación, lo cual corresponde a una mala práctica en la aplicación. Sin embargo, se realizan pruebas de ruta transversal (OWASP-AZ-001), verificando que la aplicación ignora todos los parámetros desconocidos, por ejemplo la modificación de los parámetros POST en los encabezados. La figura 19 muestra la verificación en la validación de entrada de la aplicación.

Figura 19. Verificación de validación de entrada en la aplicación web.



Fuente: autores

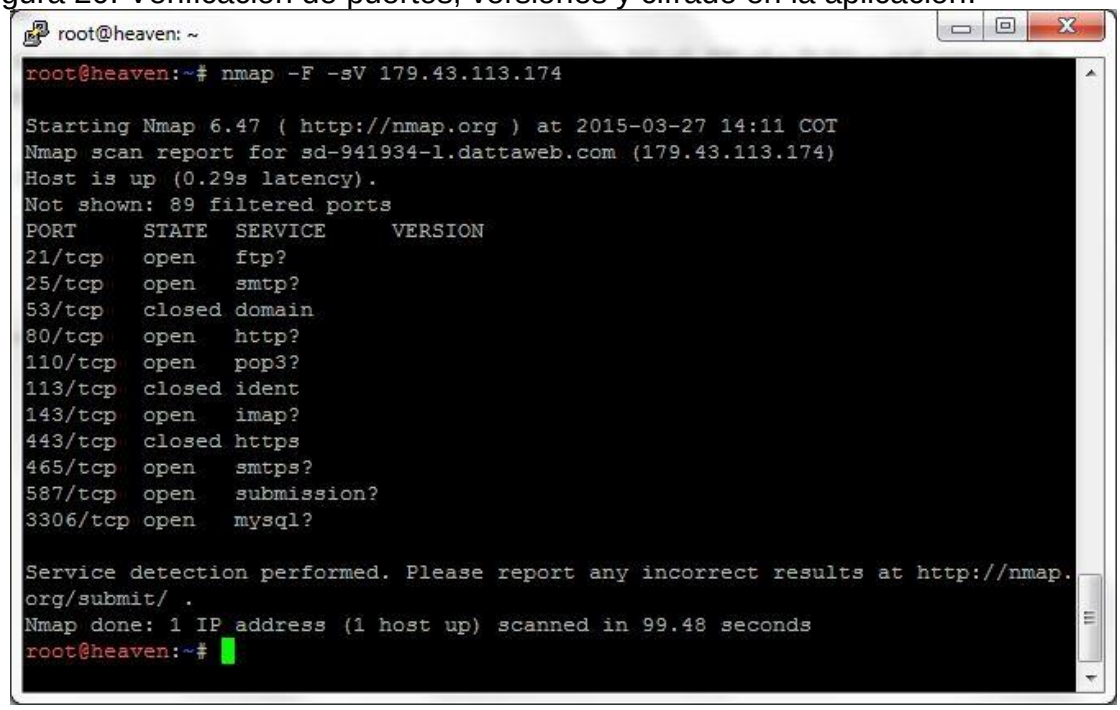
A5 - Configuración de seguridad incorrecta. Las pruebas que se extraen de la guía de OWASP para verificar este ítem son las siguientes:

OWASP-CM-001 Pruebas SSL/TLS (SSL version, algoritmos, longitud de claves, validez de certificado digital).
OWASP-CM-003 Prueba de gestión de configuración de infraestructura.
OWASP-CM-004 Prueba de gestión de configuración de aplicación.
OWASP-CM-008 Prueba de métodos HTTP y XST.

Verificando OWASP-CM-001. Con el fin de detectar un posible apoyo de cifrados débiles, los puertos asociados a SSL / TLS envueltos servicios estén identificados.

Estos incluyen típicamente el puerto 443, que es el puerto https estándar. Sin embargo, esto puede cambiar porque los servicios https pueden ser configurados para ejecutarse en puertos no estándar. Con el scanner nmap, a través de la opción de escaneo –sV se identifican los servicios SSL. La figura 20 muestra la verificación de puertos, versiones y cifrados en la aplicación. Se puede ver que además de realizar el descubrimiento de servicios se pueden incluir comprobaciones de cifrados débiles, evidenciando además que la url no cuenta con SSL.

Figura 20. Verificación de puertos, versiones y cifrado en la aplicación.



Fuente: autores

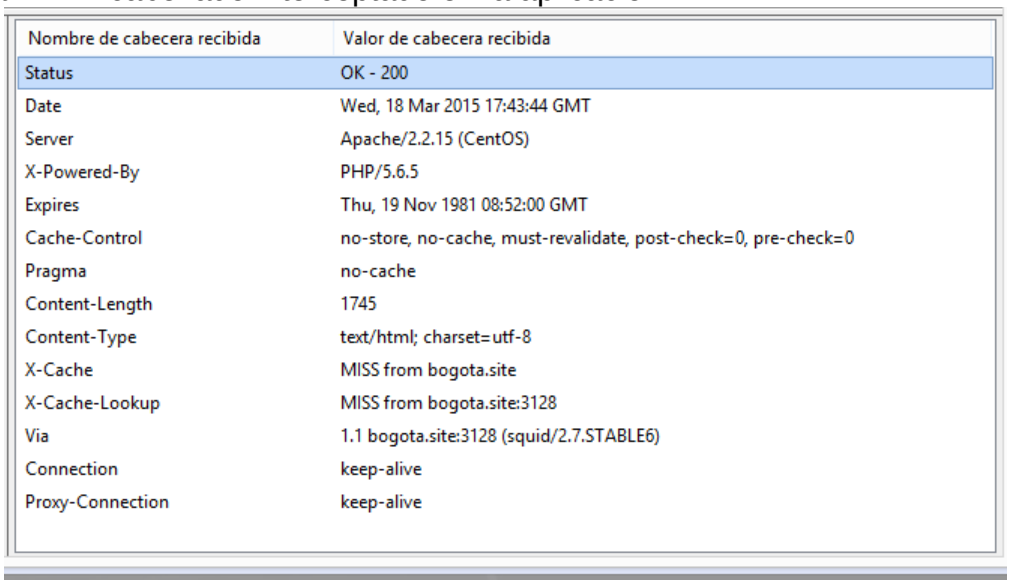
Verificando OWASP-CM-003. La figura 21 muestra un encabezado interceptado en la aplicación y en él se observa que la versión de php usada en la programación es la 5.6.5, para la cual ya existen vulnerabilidades críticas identificadas como: CVE-2015-0231, CVE-2014-9427 Y CVE-2015-0232[31]. Estas vulnerabilidades permiten la ejecución de código arbitrario y denegación de servicios, respectivamente.

También existen diversas vulnerabilidades conocidas para la versión de Apache server instalada, que podrían permitir a un atacante remoto ejecutar código

[31] UNAALDIA [Web en línea].<>. [en línea], [consultado el 17 de marzo de 2015]. Disponible en: http://unaaldia.hispasec.com/2015/01/diversas-vulnerabilidades-en-php.html

arbitrario o causar denegación de servicios. Estas vulnerabilidades se conocen con los CVE-2014-0118, CVE-2014-0231, CVE-2014-0226, CVE-2013-5704.

Figura 21. Encabezado interceptado en la aplicación.

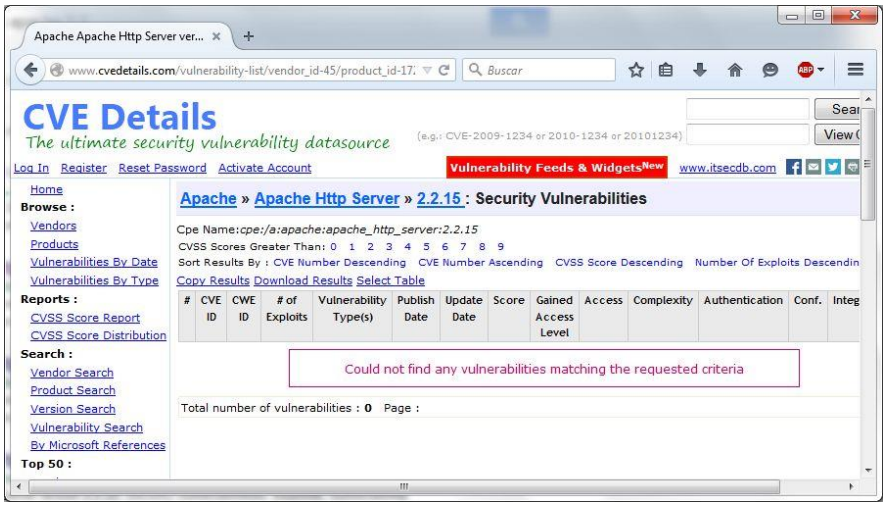


Nombre de cabecera recibida	Valor de cabecera recibida
Status	OK - 200
Date	Wed, 18 Mar 2015 17:43:44 GMT
Server	Apache/2.2.15 (CentOS)
X-Powered-By	PHP/5.6.5
Expires	Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control	no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma	no-cache
Content-Length	1745
Content-Type	text/html; charset=utf-8
X-Cache	MISS from bogota.site
X-Cache-Lookup	MISS from bogota.site:3128
Via	1.1 bogota.site:3128 (squid/2.7.STABLE6)
Connection	keep-alive
Proxy-Connection	keep-alive

Fuente: autores

Verificando OWASP-CM-004. A partir de las vulnerabilidades conocidas en las versiones instaladas, basándose en National Vulnerability Database del NIST y en este caso del performance del servidor Apache se puede sellar y hacer un hardening de la máquina. En la figura 22 se muestra la comprobación de algunas vulnerabilidades detectadas con su respectivo código CVE.

Figura 22. Comprobación de algunas vulnerabilidades con su respectivo código CVE.



Fuente: autores

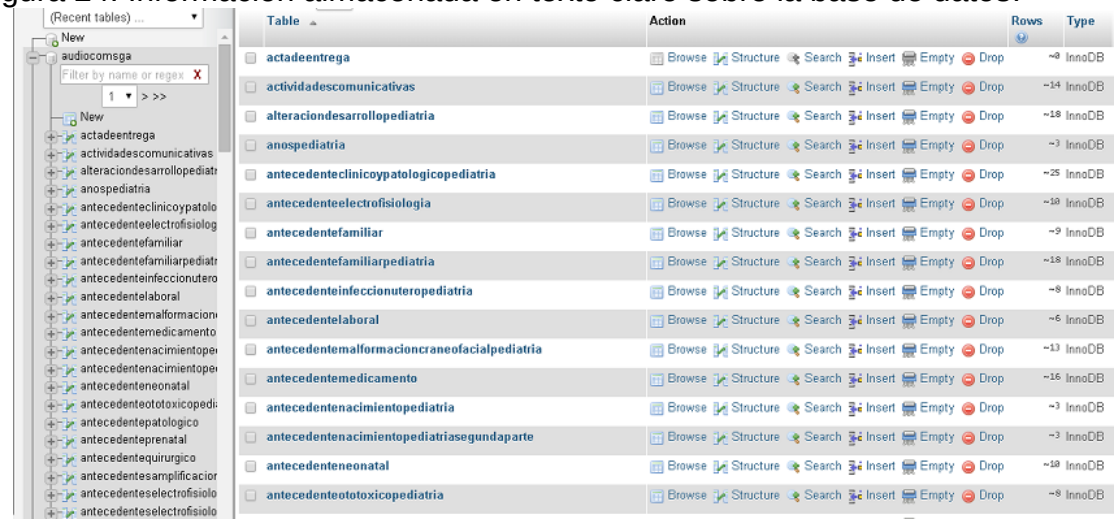
Verificando OWASP-CM-008. En la figura 23 se muestra la respuesta del servidor a las peticiones de netcat, que son necesarias para la realización de esta prueba, con lo que se sabe cuáles son los métodos HTTP compatibles con el servidor web que está siendo examinado.

Figura 23. Respuesta del servidor a las peticiones de netcat.

Fuente: autores

A6 - Exposición de datos sensibles. Siguiendo las pruebas que se realizaron para los puntos anteriores, se sabe que la aplicación expone datos sensibles, debido a la carencia de un certificado de seguridad en la transmisión de datos, ya que la información viaja en texto claro y puede ser interceptada con facilidad por un atacante. Este riesgo se complementa sabiendo que la información en la base de datos está almacenada en texto claro. La figura 24 muestra la información almacenada en texto claro sobre la base de datos.

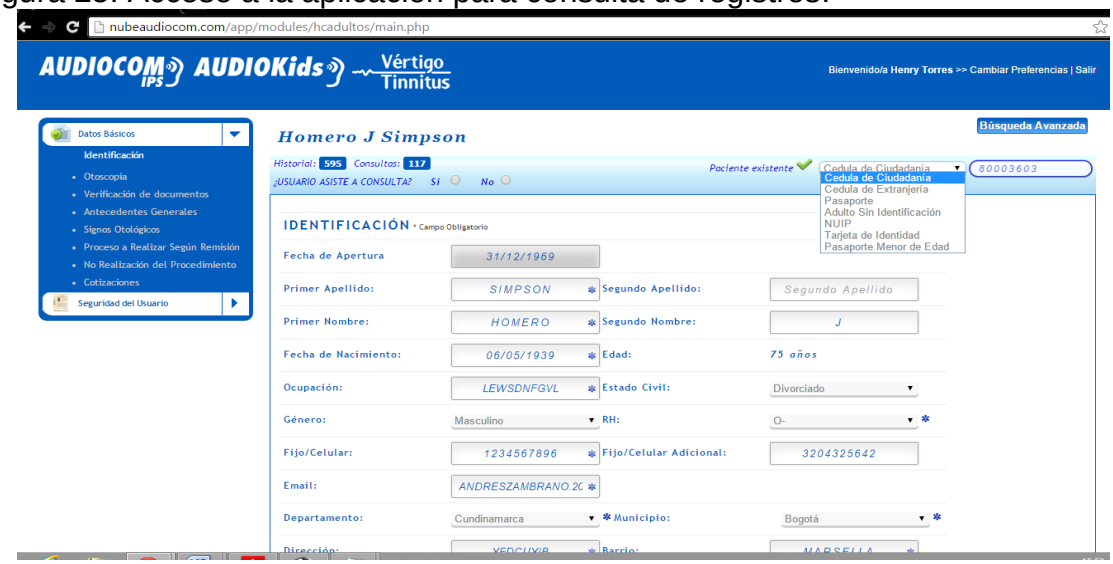
Figura 24. Información almacenada en texto claro sobre la base de datos.



Fuente: autores

A7 - Inexistente control de acceso a nivel de funcionalidades. Siguiendo las pruebas realizadas con anterioridad que permiten comprobar la referencia a objetos, la autenticación y gestión de sesiones, las probabilidades de inyección y XSS, con las cuales se verifican las funcionalidades de la aplicación manualmente y a través de un proxy, no se encuentran vulnerabilidades en este punto. Sin embargo, se sabe que la aplicación maneja el mismo perfil para todos los usuarios y por ello, si es vulnerada, un atacante podría tener acceso a la información de la historia clínica de los pacientes, sabiendo previamente su número de cédula, ya que la aplicación no entrega resultados de varios pacientes a la vez. La figura 25 muestra el acceso a la aplicación para la consulta de registros, con lo que se evidencia que cuando se accede solo se puede consultar 1 (un) número de identificación de paciente por vez.

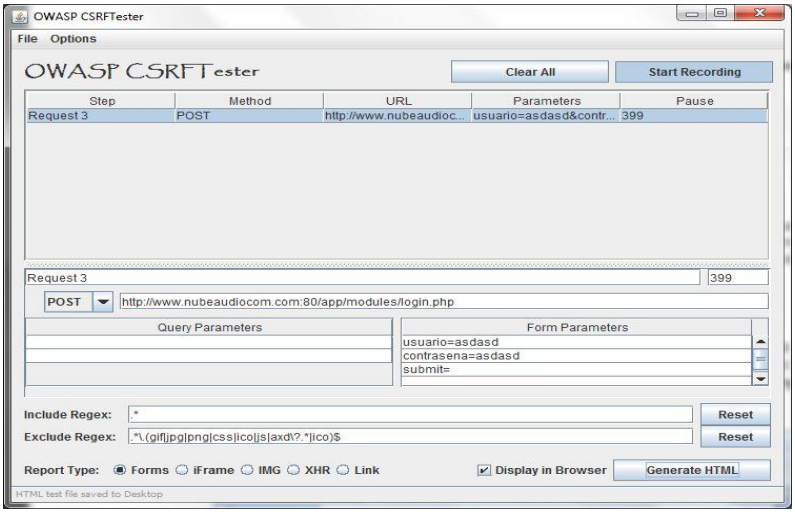
Figura 25. Acceso a la aplicación para consulta de registros.



Fuente: autores

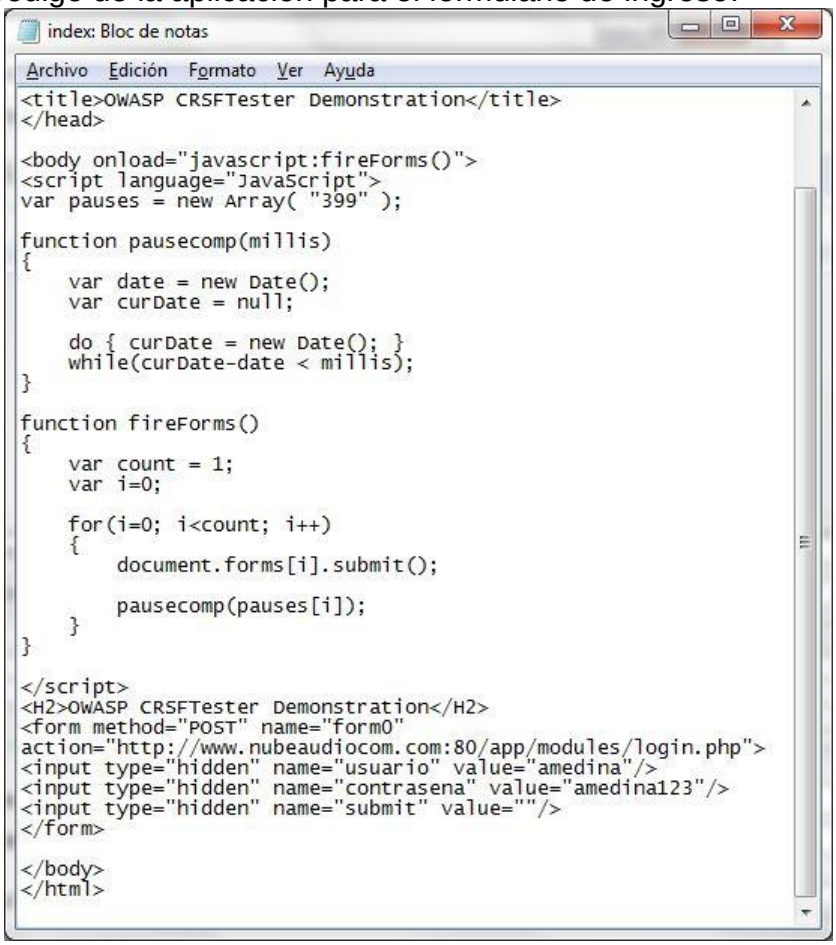
A8 - Falsificación de peticiones en sitios cruzados (CSRF). Teniendo como base que las pruebas realizadas en el apartado de inyección no arrojan resultados que evidencien la existencia del mismo, se sabe que tampoco es posible la realización de ataques de falsificación de peticiones en sitios cruzados. La figura 26 muestra la respuesta del tester CSRF de OWASP, con lo que se evidencia que no hay vulnerabilidad a este tipo de inyección. La figura 27 muestra el código de la aplicación para el formulario de ingreso, con lo que se pueden evidenciar los filtros descritos en apartados anteriores. Finalmente, la figura 28 muestra el resultado de la prueba de CSRF, con lo que se ratifica que no existen inyecciones de ese tipo.

Figura 26. OWASP CSRF Tester



Fuente: autores

Figura 27. Código de la aplicación para el formulario de ingreso.



Fuente: autores

Figura 28. Resultado de la prueba de CSRF

Fuente: autores

A9 - Uso de componentes con vulnerabilidades conocidas. La aplicación en cuestión es un desarrollo propio en su totalidad, en la cual no se usan componentes o frameworks de terceros y por tanto no aplican vulnerabilidades de este tipo.

A10 - Redirecciones y reenvíos no válidos. La aplicación en cuestión no utiliza redirecciones a otros sitios y tampoco se le detectaron posibles inyecciones, por lo cual no aplica este ítem.

4.1 DESARROLLO DEL ESTÁNDAR DE VERIFICACIÓN Y ASEGURAMIENTO DE APLICACIONES (ASVS)

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones.

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V1.1	Verifique que todas las páginas y recursos requieren autenticación, excepto los destinados específicamente a ser pública (Principio de la mediación completa).	S	S	S	Cumple	OWASP-AT-002

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V1.2	Verifique todos los campos de contraseña no se hacen eco de la contraseña del usuario cuando se introduce, y que los campos de contraseña (o las formas que los contienen) tienen autocompletar deshabilitados.	S	S	S	No Cumple	OWASP-AT-002
V1.3	Verifique que todos los controles de seguridad funcionan correctamente, asegúrese de que un atacante no se puede loguear.	S	S	S	Cumple	OWASP-AT-002, OWASP-AT-006
V1.4	Compruebe que las credenciales y toda otra información de identidad que maneja la aplicación no atraviesan enlaces no cifrados o encriptados débilmente.	S	S	S	No Cumple	OWASP-AT-001
V1.5	Compruebe que olvidó la contraseña y otras vías de recuperación no envíen las contraseñas existentes o nuevas en texto claro para el usuario.	S	S	S	No Aplica	No Aplica
V1.6	Verifique que la enumeración nombre de usuario no es posible a través de la conexión de restablecimiento de contraseña, o se olvidó funcionalidad cuenta.	S	S	S	Cumple	OWASP-AT-002

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V1.7	Verifique que no hay contraseñas por defecto en el uso de la estructura de aplicaciones o componentes utilizados por la aplicación (como " admin / password").	S	S	S	Cumple	OWASP-AT-004, OWASP-AT-006
V1.8	Compruebe que un gobernador de recursos está en su lugar para proteger contra vertical (una sola cuenta probado contra todas las contraseñas posibles) y bruta horizontal forzando (todas las cuentas analizadas con la misma contraseña por ejemplo, " Password1 "). Una entrada de credenciales correcto debería incurrir en ningún retraso. Por ejemplo, si un atacante intenta fuerza bruta todas las cuentas con la contraseña única " Password1 ", cada intento incorrecto incurre en un lineal de marcha atrás (digamos 5, 25, 125, 625 segundos) con un bloqueo suave de decir 15 minutos.	N	S	S	Cumple	OWASP-AT-004, OWASP-AT-007
V1.9	Verifique todos los controles de autenticación se aplican en el lado del servidor.	N	S	S	Cumple	OWASP-AT-005, OWASP-AT-010

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V1.10	Verifique los campos de entrada de contraseña permiten o alientan el uso de frases de paso, y no impiden contraseñas largas o contraseñas de alta complejidad que entra, y proporcionan una resistencia mínima suficiente para proteger contra el uso de contraseñas comúnmente elegidos.	N	S	S	Cumple	OWASP-AT-006
V1.11	Compruebe todas las funciones de administración de cuentas (como el registro, actualización del perfil, se olvidó el nombre de usuario. Olvidó su contraseña, token de discapacitados / perdida, mesa de ayuda o IVR) que podría volver a tener acceso a la cuenta son por lo menos tan resistentes al ataque como mecanismo de autenticación primaria.	N	S	S	No Aplica	No Aplica
V1.12	Verificar que los usuarios pueden cambiar sus credenciales de forma segura utilizando un mecanismo que es al menos tan resistente al ataque como el mecanismo de autenticación primaria.	N	S	S	No Aplica	No Aplica

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V1.13	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Cumple	OWASP-AT-007
V1.14	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Cumple	OWASP-AT-007
V1.15	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-AT-003
V1.16	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Cumple	OWASP-AT-001
V1.17	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V1.18	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V1.19	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V1.20	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V1.21	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V1.22	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V2.1	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Aplica	No se manejan Frameworks
V2.2	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-SM-002, OWASP-SM-003
V2.3	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Cumple	OWASP-SM-001
V2.4	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-SM-001
V2.5	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-SM-001, OWASP-SM-004

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V2.6	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-SM-001, OWASP-SM-003
V2.7	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Cumple	OWASP-SM-001, OWASP-SM-002
V2.8	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Cumple	OWASP-SM-001, OWASP-SM-002
V2.9	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-SM-003
V2.10	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-SM-001, OWASP-SM-002
V2.11	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V2.12	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V2.13	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V2.14	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Cumple	OWASP-SM-003
V2.15	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V3.1	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-AT-005, OWASP-SM-001
V3.2	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-AT-005, OWASP-SM-003
V3.3	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-AT-005
V3.4	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-AZ-002, OWASP-DV-003
V3.5	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-IG-005

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V3.6	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-AZ-002
V3.7	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-AZ-001, OWASP-AZ-002, OWASP-AZ-003
V3.8	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-AZ-001, OWASP-AZ-002, OWASP-AZ-003
V3.9	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-AZ-003
V3.10	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-AT-005, OWASP-AT-010
V3.11	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-CM-007

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V3.12	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-DV-001, OWASP-DV-002, OWASP-SM-005
V3.13	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Cumple	OWASP-CM-007, OWASP-AT-002, OWASP-AT-003
V3.14	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V4.1	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-DV-014
V4.2	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-DV-005
V4.3	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-DV-001, OWASP-DV-002, OWASP-DV-003, OWASP-DV-004, OWASP-DV-015,

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V4.4	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Aplica	No Aplica
V4.5	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-DV-013
V4.6	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-AT-010
V4.7	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-AT-005, OWASP-AT-010
V4.8	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-DV-001, OWASP-DV-002, OWASP-DV-012
V4.9	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-DV-012, OWASP-DS-001
V4.10	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V4.11	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V4.12	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V4.13	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S		
V4.14	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S		
V4.15	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S		
V5.1	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V5.2	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V5.3	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V5.4	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V5.5	Verificar que las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S		
	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.					
	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.					
V5.6	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S		
V5.7	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S		
V6.1	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-AT-002
V6.2	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V6.3	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-SM-004
V6.4	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-SM-004
V6.5	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-CM-004
V6.6	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Cumple	OWASP-CM-004
V6.7	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-SM-004
V6.8	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-SM-004
V6.9	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V6.10	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V6.11	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V7.1	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-AT-007, OWASP-SM-004
V7.2	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Cumple	OWASP-AT-001, OWASP-DV-016
V7.3	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-AT-007
V7.4	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-AT-007
V7.5	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V7.6	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V7.7	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V7.8	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V8.1	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Cumple	OWASP-AT-001
V8.2	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Cumple	OWASP-AT-001
V8.3	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Cumple	OWASP-AT-001
V8.4	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-AT-002, OWASP-AT-005, OWASP-AT-006
V8.5	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-AT-002, OWASP-AT-005, OWASP-AT-006
V8.6	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V8.7	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V8.8	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V8.9	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V9.1	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Cumple	OWASP-AT-001, OWASP-DV-016
V9.2	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-AT-001, OWASP-SM-004, OWASP-AZ-001
V9.3	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-SM-003
V9.4	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Cumple	OWASP-AT-004

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V10.1	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V10.2	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V10.3	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V10.4	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V10.5	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V10.6	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V10.7	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V10.8	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V10.9	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V10.10	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V10.11	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V11.1	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V11.2	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V11.3	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V11.4	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V11.5	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Cumple	OWASP-AT-001, OWASP-AT-004, OWASP-DV-016
V11.6	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Cumple	OWASP-AT-001, OWASP-DS-003
V11.7	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	Cumple	OWASP-AZ-001, OWASP-AZ-003
V11.8	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V11.9	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V11.10	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V12.1	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Aplica	No Aplica

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V12.2	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-AZ-001, OWASP-AZ-002
V12.3	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Cumple	OWASP-CM-005, OWASP-DV-002
V12.4	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Cumple	OWASP-CM-005, OWASP-DV-002
V12.5	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Cumple	OWASP-CM-005, OWASP-DV-002
V12.6	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	Cumple	OWASP-DV-001, OWASP-DV-002
V12.7	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Cumple	OWASP-DV-015
V12.8	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Cumple	OWASP-DV-015
V12.9	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Cumple	OWASP-DV-015

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V12.10	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V13.1	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Cumple	OWASP-AT-001
V13.2	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Aplica	No Aplica
V13.3	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Aplica	No Aplica
V13.4	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	S	S	S	No Aplica	No Aplica
V13.5	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V13.6	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V13.7	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V13.8	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V13.9	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V13.10	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V13.11	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V13.12	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V13.13	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V13.14	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	S	S	No Aplica	No Aplica
V13.15	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica

Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V13.16	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V13.17	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V13.18	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V13.19	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V13.20	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V13.21	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V13.22	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V13.23	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica

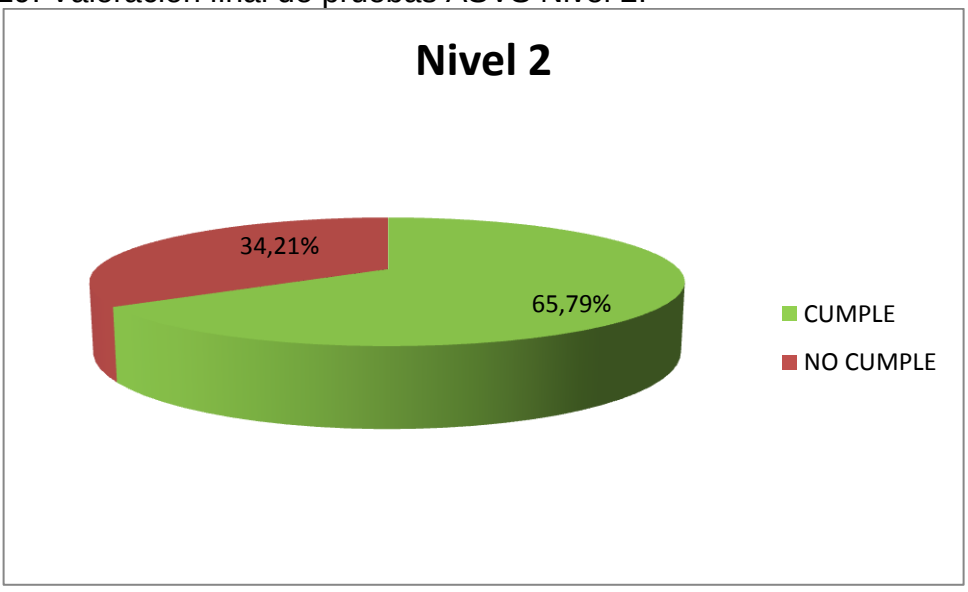
Cuadro 1. Estándar de verificación y aseguramiento de aplicaciones (Continuación)

Item	Requerimientos	Nivel 1	Nivel 2	Nivel 3	Resultado	Evidencia
V13.24	Verificar las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
V13.25	Verificar que las credenciales de autenticación pueden expirar después de un período administrativo configurable.	N	N	S	No Aplica	No Aplica
Fuente: autores						

4.1.2 RESULTADOS DEL ESTÁNDAR DE VERIFICACIÓN Y AEGURAMIENTO DE APLICACIONES.

Finalmente, para complementar las pruebas anteriores, se ubica la aplicación en el nivel 2 con 65.79% de cumplimiento, según el estándar de verificación para la seguridad de aplicaciones (ASVS - 2013). La figura 29 muestra la valoración final de las pruebas ASVS que se encuentran en el nivel 2.

Figura 29. Valoración final de pruebas ASVS Nivel 2.



Fuente: autores

5. CONCLUSIONES

- Como resultado de la aplicación de la metodología de pruebas OWASP en la verificación exhaustiva de la aplicación web para la gestión de historias clínicas de Audiocom IPS, se concluye que el equipo de programación debe empezar a utilizar la ingeniería de software como mecanismo de aplicación y evaluación de la eficiencia y calidad operacional del sistema, especialmente en la inclusión de los casos de abuso en el levantamiento de requerimientos.

- Después de haber desarrollado el estándar de verificación y aseguramiento de aplicación basado en la versión 3 de la guía de pruebas de OWASP, se concluye que la aplicación web de Audiocom IPS se encuentra en el nivel 2, ya que hace parte de un procedimiento que involucra el cuidado de la salud y en donde los atacantes podrían encontrar datos sensibles que pueden utilizar directa o indirectamente en el robo de identidad, pagos fraudulentos o una variedad de esquemas de fraude. Los resultados también muestran que actualmente la aplicación web tiene un porcentaje de inclumplimiento 34.21%, que debe ser minimizado siguiendo las recomendaciones que se incluyen en este trabajo.

- En la actualidad, mantener los registros de la historia clínica actualizados es un imperativo legal que contribuye con el mejoramiento en la calidad de la atención, ya que permite rápido acceso a los registros vitales que definen o intervienen en la salud del paciente sin tener que esperar el documento físico, por lo que se concluye que este trabajo resultante debería ser un requisito obligatorio para todos los desarrollos de software de las instituciones prestadoras de servicios de salud que almacenen información clínica de pacientes.

- Desde los principios de la medicina se tuvo la necesidad de llevar un registro escrito de la historia clínica del paciente, si bien siempre el mismo se realizó en papel, hoy surge la necesidad de cambiar de medio de registro de los datos médicos, por lo cual, el desarrollo de software y la programación se han convertido en uno de los pilares fundamentales de la informática y al cual se dedican muchas horas y esfuerzos en las instituciones prestadoras de servicios de salud, que entre otras, no consideran la importancia de la seguridad en el tráfico de información.

- Toda organización que desarrolle software para el registro de datos clínicos debe contar con personal capacitado para la implementación y desarrollo de software seguro. Esta política acertada permite que las unidades organizativas garanticen la confidencialidad de los datos personales y sensibles de los usuarios, dando cumplimiento a la ley de protección de datos.

- Podría considerarse que la clave para mantener la seguridad de la información de los expedientes clínicos electrónicos está en contar con soluciones informáticas o herramientas óptimas de seguridad en la información. Sin embargo, aunque esto es necesario, la realidad demuestra que también se necesitan los aspectos físicos que generalmente se omiten y son imprescindebles para la seguridad de cualquier negocio. Por tanto, es posible minimizar los riesgos de la información con la combinación de la tecnología y los medios físicos y la participación de los responsables de la información.

6. RECOMENDACIONES

- Crear un módulo de administración de sesión de usuarios, en donde se definan roles, permisos y accesos autorizados en el aplicativo web, con lo que se excluiría una mala práctica en el desarrollo de software y se evitarían accesos innecesarios al código fuente de la aplicación. Además, se debe tener en cuenta que la práctica actual para la creación de usuarios requiere la modificación de archivos directamente en el servidor que almacena la aplicación y por tanto hay pérdida de disponibilidad mientras culmina el proceso. Esto le cuesta a la empresa en la mala atención de pacientes por retraso en la consulta y por tanto en la pérdida de buen nombre ante los pacientes y las EPS contratantes, sin contar el tiempo extra de los profesionales en audiología.

- Adquirir un certificado SSL con un proveedor líder mundial de seguridad en línea, por ejemplo Symantec. Con ello, la empresa puede evitar el robo de información de los pacientes que se almacenan en la base de datos, por lo cual podría acarrear multas de hasta 2000 salarios mínimos vigentes, es decir, $12.887 millones de pesos, tal como se estipula en el decreto 1377 de 2013. Si el certificado se adquiere con el proveedor actual que suministra el servidor virtual, tendría un costo por año de $328 mil pesos; siendo necesario para un solo dominio como en este caso.

- Instalar por lo menos semanalmente los parches de actualización y de seguridad establecidos en el sistema operativo y las versiones de los servicios instalados sobre el mismo, que en este caso son PHP, Apache Server y CentOS. La funcionalidad de estas actualizaciones es cubrir los bugs y sellar vulnerabilidades conocidas en el sistema evitando vulnerabilidades del día cero, teniendo en cuenta que actualmente el sistema está expuesto a varias que podrían ocasionar robo de información y denegación de servicios.

- Implementar a futuro un Web Aplication Firewall para analizar y proteger el tráfico web entre el servidor y la WAN. Este gasto no sería solo aplicado para el portal web, también aplica para la infraestructura de la empresa. Esta no debe considerarse como una inversión alta, si se tiene en cuenta que actualmente protege los datos de más de 250000 pacientes, que representan un ingreso anual de $19000000000 de pesos.

- Capacitar al personal de desarrollo de software para que implemente los métodos que se requieran para dar cumplimiento absoluto a la verificación de requerimientos realizada en el estándar de verificación para seguridad de aplicaciones (ASVS) realizado en este trabajo, ya que este incrementaría el nivel de cumplimiento de las buenas prácticas de desarrollo y cerraría las vulnerabilidades detectadas.

- Brindar capacitación especializada al personal que desarrolla la aplicación para que implementen buenas prácticas de desarrollo y metodologías de desarrollo ágil, teniendo en cuenta que las exigencias de la gerencia para este departamento incluyen que los requerimientos se entreguen en plazos cortos.

- Documentar el proceso y la aplicación ya que para ello no se tienen requerimientos funcionales y no funcionales, al igual que el manual funcional de la aplicación, por lo que es difícil incluir nuevos desarrolladores en el equipo de trabajo o hacer modificaciones sobre los módulos ya desarrollados, lo cual puede generar pérdidas monetarias en tiempo de desarrollo y capacitación de usuarios profesionales en el uso de la aplicación.

- Realizar auditorías periódicas que incluyan escaneos automáticos sobre todo el sistema y especialmente a las nuevas funcionalidades incluidas en la aplicación, de esta manera se podrá evaluar el avance obtenido a nivel de seguridad y se garantizará el constante cumplimiento de los parámetros establecidos para tal fin. Estas auditorías también deben incluir las actualizaciones que genere la fundación OWASP en sus publicaciones de los diez riesgos más comunes de aplicaciones web y la guía de pruebas que se utilizaron en este trabajo, sabiendo que estas se pueden adquirir gratuitamente.

BIBLIOGRAFÍA

- CALDERA, Jorge. Realidad Aumentada en Televisión y Propuesta de Aplicación en los Sistemas de Gestión Documental. Diciembre de 2014. [en línea], [consultado el 22 de abril de 2015]. Disponible en: http://web. a.ebscohost.com/ehost/results?sid=d0c7580a-ff4d-4b76-ba96-9fd984 ebedcb%40 sessionmgr4003&vid=2&hid=4201&bquery=gestion+documental&bdata=JmR

- COLOMBIA. CONGRESO DE LA REPÚBLICA. Ley 100. Diciembre 23 de 1993. Por la cual se crea el sistema de seguridad social integral y se dictan otras disposiciones. Diario Oficial. Bogotá D.C., 1993. No 41148. p. 1

- _____. _____. Ley 1581. Octubre 17 de 2012. Por la cual se dictan disposiciones generales para la protección de datos personales. Diario Oficial. Bogotá D.C., 2012. No 48587. p. 1.

- FFIEC, Authentication in an Internet BankingEnvironment. Testing [en línea], [consultado el 23 de marzo de 2015]. Disponible en: http://www.ddj. com security /18950000

- GARY McGraw, Beyond the Badness-ometer.testing[en línea], [consultado el 23 de marzo de 2015]. Disponible en: testing[en línea], [consultado el 23 de marzo de 2015]. Disponible en: http://www.ddj.com/security/189500001

- GIL REGEV, Ian Alexander, Alain Wegmann, Use Cases and Misuse Cases Model the Regulatory Roles of Business Processes .testing[en línea], [consultado el 23 de marzo de 2015]. Disponible en: http://easyweb.easynet. co.uk/~ iany/ consultancy/regulatory _processes/ regulatory_processes.htm

- IPS AUDIOCOM La Web [en línea], [consultado el 23 de marzo de 2015]. Disponible en: http://www.audiocom-ips.com

- LÓPEZ, M. (2012). Efectividad de técnicas de OWASP para asegurar aplicaciones web contra inyección de SQL. México. Universidad de Guadalajara, Tesis de maestría en tecnologías de la información, 120 p.

- MINISTERIO DE TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES, Cero Papel en la Administración Pública. [en línea], [consultado el 2 de marzo de 2014]. Disponible en: http://programa.gobierno enlinea. gov.co/apc-aa-files/Cero_papel/guia-1-cero-papel. pdf

- MITRE, BeingExplicitAboutWeaknesses, Slide 30, Coverage of CWE [en línea], [consultado el 23 de marzo de 2015]. Disponible en:

http://cwe.mitre.org/ documents/ beingexplicit/BlackHat DC_ BeingExplicit_Slides.ppt

- MSDN, CheatSheet: Web Application Security Frame - testing[en línea], [consultado el 23 de marzo de 2015]. Disponibleen: http://msdn.microsoft. com/enus/library/ms978518. aspx# tmwacheatsheet _webappsecurityframe

- _____. Improving Web Application Security, Chapter 2, Threat And Countermeasures - testing [enlínea], [consultado el 23 de marzo de 2015]. Disponible en: http://msdn.microsoft. com/ enus/ library/aa302418.aspx

- NIST, The economic impacts of inadequate infrastructure for software testing [enlínea], [consultado el 23 de marzo de 2015].Disponible en: http://www.nist.gov/public_affairs/releases/ n02- 10.htm

- OWASP Foundation (2008). Guía de pruebas OWASP. [en linea], [consultado el 23 de marzo de 2015]. Disponible en https://www.owasp.org/images/8/80/Guía_de _pruebas_de_OWASP_ver_3.0.pdf. [Citado 22-Mar-2015].

- _____ (2013).OWASP Application Security Verification Standard 2013.[enlinea], [consultado el 23 de marzo de 2015]. Disponible en: https://www.owasp.org/ images/9/96/Introducing_ASVS_2013_Sahba_Kazerooni%2BDaniel_Cuthbert. pdf.

- _____. (2013). OWASP Top 10 2013. [en linea], [consultado el 23 de marzo de 2015]. Disponible en:http://owasptop10.googlecode.com/files/OWASP%20 Top% 2010%20-%202013.pdf.

- PCI SECURITY STANDARDS COUNCIL, PCI Data Security Standard – testing[en línea], [consultado el 23 de marzo de 2015]. Disponibleen: https://www.pci securitystandards.org/security _standards/ pci_dss.shtml

- PETRINJA, Etiel. IntroducingtheOpenSourceMaturityModel. Mayo de 2009. [en línea]. [consultado el 22 de abril de 2015]. Disponible en: http://web.a. ebscohost. com/ehost/ detail/ detail? vid=10& sid= d0c7580a-ff4d-4b76-ba96-9fd984ebedcb%

- SALESA, Enrique; PERELLO, Enrique y BONAVIDA, Alfredo. Tratado de audiología. 2013. Vol. 2, No. 357, p. 27-57. [en línea], [consultado el 22 de abril de 2015]. Disponible en: https://books.google.com.co/books?isbn=8445823957

- SALLIS E., CARACCIOLO C., RODRÍGUEZ M. Etical Hacking. (1ª. Ed.).Buenos Aires, Bogotá, México, Santiago de Chile, 2009.

- SCHNEIER, Bruce. CryptogramIssue #9. Testing [en línea], [consultado el 23 de marzo de 2015]. Disponible en: http://www.schneier.com/crypto-gram-0009.html

- SECURITY ACROSS THE SOFTWARE DEVELOPMENT LIFECYCLE TASK FORCE, REFERRED DATA FROM CAPER JOHNS, Software Assessments, Benchmarks and Best Practices -[enlínea], [consultado el 23 de marzo de 2015]. Disponibleen: http://www. cyberpartnership.org/SDLCFULL.pdf

- SINDRE,G. Opdmal A., Capturing Security Requirements Through Misuse Cases. testing[en línea], [consultado el 23 de marzo de 2015]. Disponible en: http://folk.uio.no/nik/2001/21-sindre.pdf

- SYMANTEC. ThreatReportstesting[en línea], [consultado el 23 de marzo de 2015]. Disponibleen: http://www.symantec.com/business/theme.jsp? themeid =threatre port

- T. De Marco, Controlling software projects: management, measurement and estimation. Londrés: Press, 1982. p. 30.

- UNAALDIA.Web en línea. [consultado el 17 de marzo de 2015]. Disponible en: http://unaaldia.hispasec.com/2015/01/diversas-vulnerabilidades-en-php.html