ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"ANÁLISIS DE VULNERABILIDADES Y ACCIONES CORRECTIVAS SOBRE UN SISTEMA WEB"

EXAMEN DE GRADO (COMPLEXIVO)

Previo a la obtención del Título de:

MAGISTER EN SEGURIDAD INFORMÁTICA APLICADA

DANNY OMAR PINOS SOLANO

GUAYAQUIL – ECUADOR

AÑO: 2017

AGRADECIMIENTOS

Agradezco a Dios, a mi padre y a mi esposa por su amor y apoyo incondicional, quienes fueron participes para la culminación del presente proyecto complexivo.

A todas las personas que directa o indirectamente han brindado su apoyo para la consecución de esta meta.

DEDICATORIA

A Dios por iluminarme día a día, en todas las metas planteadas.

A mi papa, a mi esposa y familiares por el apoyo incondicional brindado y por ser mi fortaleza en todo momento.

A todas las personas que de una u otra manera han estado y estarán presentes en mi crecimiento personal y profesional.

TRIBUNAL DE EVALUACIÓN

Mgs. Lenin Freire	**Mgs. Juan Carlos García**
PROFESOR EVALUADOR	PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, me corresponde exclusivamente; y doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

.....

Danny Omar Pinos Solano

RESUMEN

Actualmente la empresa XYZ cuenta con un sistema web el cual almacena información referente a la gestión de ventas, cobranzas, y datos sensibles del staff de clientes de la organización, esta aplicación a pesar de ser de uso interno se encuentra publicada en internet. En una inspección realizada se pudo detectar que existen algunos fallos de seguridad los cuales pueden ser explotados por hackers maliciosos. El objetivo primordial es identificar todas las falencias tanto de infraestructura como de aplicación y proponer mejoras en cada una de ellas. Para ello nos basaremos en análisis de código estático, pentesting con Nessus y validando contra los estándares desarrollo seguro de OWASP Top 10 y OWASP testing guide.

Del análisis de la infraestructura se pudo evidenciar que existen deficiencias, dentro del equipo se encuentra el servidor de aplicaciones y base de datos, dado la alta demanda operativa que tiene por parte de los usuarios y de otros aplicativos de la organización existen lapsos de tiempos que se producen colapsos comprometiendo la disponibilidad de la información. Del análisis de vulnerabilidades se pudieron detectar varias amenazas que pueden ser muy perjudiciales, entre las principales encontradas están: propensa a ataques de inyección de código (SQL, XSS), control ineficiente de gestión de sesiones y de autorización, incorrecta validación de datos de entrada, permisos y privilegios de acceso a la base de datos inadecuados, entre otros.

Una vez identificadas todas las deficiencias se procedió a implementar mejoras sobre cada una de ellas empezando por una restructuración de la arquitectura, aplicando mejoras de seguridad en el código fuente y elaborando una “política y estándar de programación segura” para futuros desarrollos con el objetivo de mitigar huecos de seguridad.

ÍNDICE GENERAL

INTRODUCCIÓN

En la actualidad cualquier computadora conectada a internet está expuesta a diversas amenazas, el riesgo sobre los sistemas informáticos ha aumentado debido a la variedad de herramientas de hacking usadas con fines maliciosos y sobre todo por el poco nivel de conocimientos que se necesita para utilizarlas, la consecuencia de esto es un incremento en el número de ataques por parte de los ciberdelincuentes ya que utilizan estas herramientas y aprovechan cualquier vulnerabilidad que encuentren en los sistemas y pueden llegar a comprometer uno de los activos más importantes de las organizaciones: “la información”.

Una forma de disminuir el impacto de futuros ataques informáticos es tomar acciones de manera anticipada, es recomendable realizar análisis de forma continua con el objetivo de detectar las vulnerabilidades potenciales en nuestros sistemas y sobre estas poder tomar las medidas necesarias para mitigarlas, de esta manera se reduce la probabilidad de éxito en los ataques realizados.

Nuestro objetivo principal en este proyecto es realizar un análisis de vulnerabilidades sobre el sistema web de ventas de la compañía XYZ, para esto utilizaremos diferentes técnicas como análisis de código estático y escaneo automático a través de la herramienta Nessus, finalmente vamos a elaborar un plan de acciones correctivas sobre cada debilidad encontrada y un documento con políticas y estándares de seguridad.

CAPÍTULO 1

GENERALIDADES.

En este capítulo se detallan los antecedentes, objetivos y problemas que dieron origen al análisis de vulnerabilidades de una aplicación web de la empresa XYZ, adicionalmente se define el plan de acciones y mejoras implementadas sobre la infraestructura y sobre el código fuente del aplicativo.

1.1 Antecedentes.

En la actualidad las tecnologías de información han evolucionado de manera considerable, esto ha sido un factor influyente para que existan múltiples amenazas y riesgos sobre los activos de las organizaciones, teniendo como consecuencia un aumento en el número de ataques aprovechándose de las vulnerabilidades o fallos de seguridad de nuestros activos o sistemas informáticos. Entre los principales problemas de seguridad de los aplicativos webs podemos destacar los siguientes: Control de acceso, Autentificación de usuarios, Gestión de la sesión, Gestión de la configuración, Visibilidad de datos sensibles, Validación de entradas de datos de usuario, Inyección en diferentes ámbitos (SQL, XSS, HTML), entre otros.

Existen varios factores que hacen posible este tipo de eventos entre ellos está el desconocimiento de políticas empresariales por parte de los usuarios internos, publicación incorrecta de servicios, configuraciones por defecto, software

malicioso dentro de la red interna y hasta fallos de seguridad en las aplicaciones web.

El análisis de vulnerabilidades nos ayuda a identificar las amenazas que existen en nuestros sistemas informáticos y a partir de los resultados obtenidos podemos tomar las medidas y controles necesarios para garantizar la disponibilidad, confidencialidad e integridad de la información.

1.2 Objetivo General

Nuestro objetivo principal es realizar un análisis de vulnerabilidades sobre el sistema de ventas y elaborar un plan de acciones correctivas sobre las principales encontradas.

1.3 Objetivos específicos

Los objetivos principales que se presentan en este caso de estudio son los siguientes:

- Realizar un análisis de código estático de la aplicación web.
- Realizar escaneo dinámico de la aplicación web mediante la herramienta Nessus.
- Realizar un informe con las principales vulnerabilidades encontradas durante la fase de escaneo.
- Implementar un plan de acciones correctivas sobre las principales vulnerabilidades encontradas.
- Elaborar un estándar de codificación para tratar de mitigar futuros agujeros de seguridad en los nuevos desarrollos que se realicen.

1.4 Descripción del problema

Actualmente la empresa XYZ cuenta con un sistema web el cual almacena toda la información referente a la gestión de ventas y cobranzas de la compañía, adicional a esto se registran datos sensibles de personas que forman parte de su staff de clientes, datos de clientes corporativos y proveedores.

Este sistema web de ventas a pesar de ser de uso interno se encuentra publicado en el internet ya que personal de TI de la matriz (ubicada en otro país) necesita acceder a cierta información para toma de decisiones corporativas. En una sencilla inspección realizada sobre el sistema se pudo detectar que existen algunos fallos de seguridad y dado que esta se encuentra publicado en internet es un blanco fácil para ataques de diferentes tipos.

1.5 Solución propuesta

Una forma de prevenir futuros ataques a nuestros sistemas informáticos es tomar acciones de manera anticipada, realizando un análisis y detectando las vulnerabilidades potenciales que podrían ser aprovechadas por los atacantes, de esta manera se reduce la probabilidad de éxito en los ataques realizados. Por este motivo vamos a tomar el siguiente plan de acción sobre el sistema:

- Realizar un análisis de vulnerabilidades sobre el sistema web de ventas y elaborar un informe con las principales debilidades encontradas.
- Diseñar e implementar un plan de acciones correctivas sobre las vulnerabilidades encontradas y especialmente sobre aquellas que se encuentren en el top 10 de OWASP 2013.
- Definir un estándar de codificación con normativas básicas de seguridad para tratar de mitigar vulnerabilidades en los desarrollos futuros.
- Definir políticas de acceso para la administración de la base de datos del sistema web.

CAPÍTULO 2

MARCO TEÓRICO.

El objetivo de este capítulo es conocer diferentes conceptos que utilizaremos en el desarrollo del caso de estudio: auditoria de seguridad, hacking ético, metodologías de análisis de vulnerabilidades, OWASP y terminologías esenciales que debemos abarcar para introducirnos de forma breve en el mundo de la seguridad informática.

2.1 Fundamentos de Seguridad Informática

2.1.1 Auditorías de seguridad

Una auditoria de seguridad es un proceso que permite evaluar e identificar de forma sistemática el estado de la seguridad en relación a una serie de criterios o normas.

Mediante este proceso se trata de identificar diversas vulnerabilidades que pudieran afectar a la confidencialidad, integridad y/o disponibilidad de un aplicativo y de los sistemas asociados con éste, lo más importante es garantizar que los 3 pilares de la seguridad de la información no se vean comprometidos por un ataque. En la figura 2.1 presentamos de forma visual el triángulo CIA [1] (por sus siglas en inglés):

- **Confidencialidad (Confidentiality):** se consigue asegurando que los objetos del sistema solo pueden ser accedidos por elementos autorizados.

- **Integridad (Integrity):** significa que los objetos de un sistema solo pueden ser modificados por elementos autorizados.
- **Disponibilidad (Availability):** indica que los elementos del sistema tienen que permanecer accesibles a los elementos autorizados

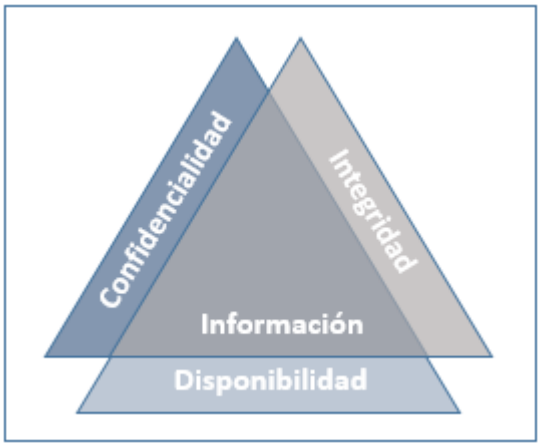


Figura 2.1: Triángulo CIA Seguridad Información

Existen 2 tipos de auditorías que pueden realizarse en una organización: internas y externas.

- **Auditorías internas:** Es realizada con recursos materiales y personas que pertenecen a la empresa auditada. La auditoría interna existe por expresa decisión de la organización, por lo que se puede optar por su disolución en cualquier momento.
- **Auditorías externas:** Es realizada por personas afines a la empresa auditada. Se presupone una mayor objetividad que en la auditoría interna, debido al mayor distanciamiento entre auditores y auditados.

2.1.2 Amenazas informáticas

Las amenazas son aquellas condiciones del entorno que, dada la oportunidad, pueden llegar a generar una violación de la seguridad en los sistemas informáticos. Existen muchas formas de clasificar las amenazas, sin embargo, la clasificación más extendida es según su tipo, y para ello se modeliza un flujo de información desde una fuente a un destino [2]. La figura 2.2 muestra este hecho:

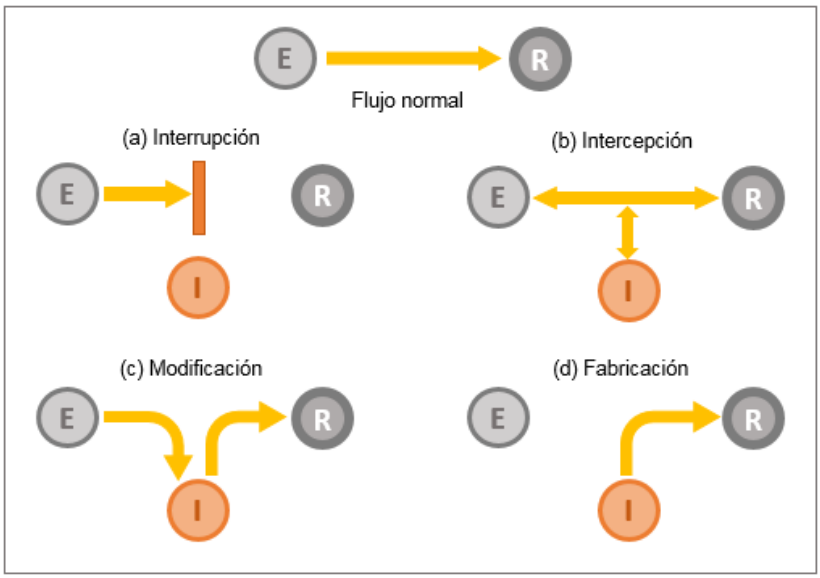


Figura 2.2: Tipos de amenaza

Los tipos de amenaza son:

a) **Interrupción:** Afecta a la disponibilidad. Se produce cuando un recurso del sistema es destruido o deja de estar disponible.

b) **Intercepción:** Afecta a la confidencialidad. Se produce cuando una entidad no autorizada consigue acceso a un recurso.

c) **Modificación:** Afecta a la confidencialidad e integridad. Se produce cuando una entidad no autorizada intercepta y manipula los datos.

d) **Fabricación:** Afecta a la autenticidad. Se produce cuando una entidad no autorizada se hace pasar por una que sí lo está y genera datos evidentemente falsos.

Otra forma importante de clasificar las amenazas es según de donde provengan. Estas pueden ser clasificadas como amenazas internas y externas [3].

- **Amenazas internas:** es un ataque que se realiza desde el interior del perímetro de seguridad del objetivo, generalmente son realizados por personas dentro de la organización (empleados inconformes o clientes con acceso temporal).
- **Amenazas externas:** es un ataque que se realiza desde una fuente externa al perímetro de seguridad del objetivo, generalmente son realizados al azar desde internet o por ex-empleados de las organizaciones.

Los ataques externos son más fáciles de detectar y repeler que los ataques internos, el atacante interno ya tiene acceso a la red privada e incluso al servidor que desea atacar.

2.1.3 Ataques informáticos

Un ataque informático es un método por el cual un individuo, mediante un sistema informático, intenta tomar el control, desestabilizar o dañar otro sistema informático. Los ataques los podemos clasificar en 2 tipos: pasivos y activos [4].

Ataques pasivos: en los ataques pasivos el atacante no altera la comunicación, únicamente la escucha o monitoriza, para obtener información que está siendo transmitida. Sus objetivos son la intercepción de datos y el análisis de tráfico. Ejemplos: Sniffing, Snooping, Ports Scan, Backdoors, Keyloggers.

Ataques activos: Estos ataques implican algún tipo de modificación del flujo de datos transmitido o la creación de un falso flujo de datos, pudiendo subdividirse en cuatro categorías:

- **Suplantación de identidad:** el intruso se hace pasar por una entidad diferente. Por ejemplo, realizar una transferencia bancaria en nombre de otra persona.
- **Reactuación:** uno o varios mensajes legítimos son capturados y repetidos para producir un efecto no deseado, como por ejemplo ingresar dinero repetidas veces en una cuenta bancaria.
- **Modificación de mensajes:** una porción del mensaje legítimo es alterada, o los mensajes son retardados o reordenados, para producir un efecto no deseado. Por ejemplo, cambiar la cantidad de un ingreso bancario.
- **Denegación de servicio:** impide o inhibe el uso normal o la gestión de recursos informáticos y de comunicaciones. Por ejemplo, impedir el acceso a una aplicación determinada.

Los ataques activos afectan la disponibilidad, la integridad y la autenticidad de los datos; mientras que los ataques pasivos son violaciones directas a la confidencialidad de la información.

2.1.4 Hacking ético

Hacking ético es una forma de referirse al acto de una persona a usar sus conocimientos de informática y seguridad para realizar pruebas en redes y encontrar vulnerabilidades, la idea es tener el conocimiento de cuales elementos dentro de la red son vulnerables y corregirlos con anticipación. Estas pruebas se llaman "pen tests" o "pruebas de penetración" en donde se intenta de múltiples formas burlar la seguridad para robar información sensitiva, para luego reportarlo a la organización y así mejorar su seguridad.

Las personas que realizan estas “Pruebas de penetración” se denominan hackers. Los hackers pueden ser divididos en tres grupos: white hats, black hats y grey hats.

- **White hats:** son hackers éticos que utilizan sus habilidades con objetivos defensivos.

- **Black hats:** son hackers maliciosos o crackers quienes utilizan sus habilidades con fines ilegales, antimorales o propósitos maliciosos.
- **Gray hats:** son hackers que pueden trabajar de manera ofensiva o defensiva según las circunstancias y la situación.

Un hacker ético sigue procesos similares a los de un atacante malicioso. Los pasos para obtener y mantener acceso al sistema informático son similares más allá de las intenciones del atacante. La figura 2.3 ilustra las cinco fases que los atacantes [5], en general, siguen para acceder a un sistema.

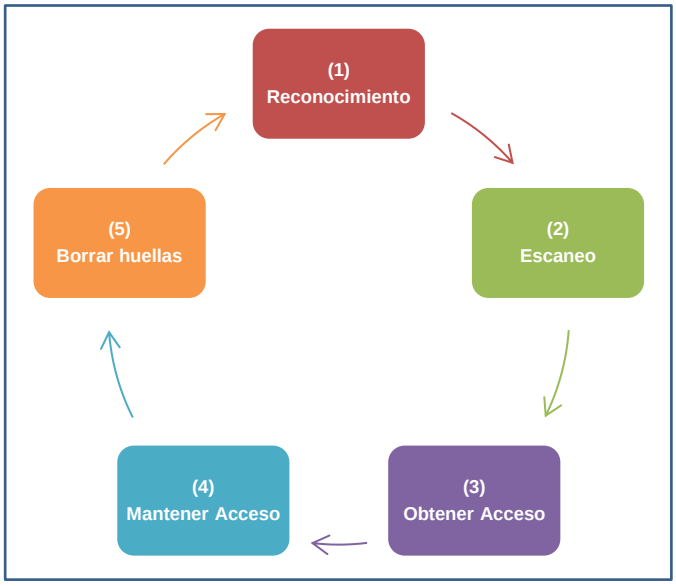


Figura 2.3: Fases del Hacking

1. **Reconocimiento:** Se intenta obtener toda la información sobre el objetivo o componente relacionado de fuentes públicas o sin utilizar métodos invasivos.
2. **Escaneo:** En esta fase se escanea y enumeran todos los servicios, aplicaciones o elementos de red a raíz del descubrimiento en la primera fase.
3. **Obtener acceso:** Esta es la fase en la que el verdadero hacking tiene lugar. Las vulnerabilidades descubiertas en las 2 fases previas ahora son explotadas para obtener acceso al sistema.

4. **Mantener acceso:** En esta fase se crea o modifica el sistema para permitir el acceso continuado al mismo, se intentan escalar privilegios de acceso, se establecen canales de comunicación para permitir descargar y ejecutar software malicioso, transferir información o analizar todo el tráfico que gestione el dispositivo.
5. **Borrar Huellas:** Se intenta eliminar cualquier evidencia del proceso de intrusión en el sistema con el fin de asegurar la fase 4, y evitar dejar cualquier información que pudiera ser utilizada legalmente en su contra, al igual que proteger el método utilizado para romper la seguridad en su acceso.

El proceso de Hacking Ético se realiza de forma similar, pero deteniéndose en la Fase 3 con el fin de realizar los informes técnicos y ejecutivos de los riesgos, vulnerabilidades encontradas y las soluciones para su eliminación o mitigación del impacto en cada caso.

2.1.5 Técnicas para utilizar en un ataque

A continuación detallamos un listado de las principales técnicas de ataques que son utilizadas por los hackers:

- Denegación de servicio (DoS)
- Denegación de servicio distribuidos (DDoS)
- Crackeo de contraseña
- Explotación de vulnerabilidades
- Phising
- Secuestro de secciones en redes wifi
- Hijacking
- Spoofing
- Ingeniera social
- Confianza transitiva
- Ataques dirigidos por datos
- Troyanos

2.2 Open Web Security Project (OWASP)

2.2.1 Introducción

Con el fin de agrupar y clasificar todo el conocimiento en relación al desarrollo seguro y seguridad en aplicaciones web, así como de proveer documentación, herramientas y metodologías libres y de código abierto nace en 2004 un organismo sin fines de lucro bajo el nombre de “Open Web Application Security Project” (OWASP).

Dicho organismo está formado tanto por empresas como instituciones académicas e individuales de todo el mundo. Algunos de los proyectos más conocidos se pueden encontrar en la siguiente lista:

- OWASP Application Security Verification Standard (ASVS)
- OWASP Enterprise Security API(ESAPI)
- OWASP Development guide
- OWASP Top 10
- OWASP Testing guide
- OWASP Zed Attack Proxy(ZAP)

2.2.2 OWASP Top 10 Vulnerabilidades

El OWASP Top 10 [6] es una lista consensuada de las vulnerabilidades más críticas que se pueden encontrar en aplicativos webs. Se debe distinguir y notar que son las diez más críticas y no las diez más comunes. Esta lista se actualiza continuamente y la última versión liberada oficial es del año 2013 e identifica las siguientes vulnerabilidades:

- **A1 Inyección:** Es un ataque en el cual se inserta código malicioso en las cadenas para aprovecharse del intérprete de datos y ejecutar comandos no deseados. Estos ataques pueden venir tanto de fuentes externas de datos, como podrían ser formularios en la aplicación web.
- **A2 Perdida de autenticación y Gestión de sesiones:** en este caso los atacantes se aprovechan de debilidades en el sistema de autenticación o de control de sesiones (enumeración de cuentas, contraseñas, identificadores de sesión, etc) para suplantar usuarios.

- **A3 Secuencia de comandos en sitios cruzados (XSS):** Son aquellas en las que el atacante se aprovecha de los fallos de validación de datos que la aplicación realiza permitiéndole ejecutar código malicioso en el navegador del usuario pudiendo secuestrar sesiones de usuarios, desfigurar sitios web o redirigir a sitios maliciosos.
- **A4 Referencia directa insegura a objetos:** Este tipo de vulnerabilidades permite a los atacantes aprovecharse de la aplicación alterando parámetros que referencian directamente a elementos del sistema o usuarios sobre los cuales no tiene autorización.
- **A5 Configuración de seguridad incorrecta:** Corresponde a configuraciones no adecuadas que pueden impactar en la seguridad de la propia aplicación, generalmente se debe a configuraciones predeterminadas inseguras como usuarios y contraseñas por defecto, páginas de administraciones accesibles, vulnerabilidades conocidas sin parchear, ficheros y directorios sin proteger.
- **A6 Exposición de datos sensibles:** Se clasifican en esta categoría aquellas vulnerabilidades en la que los atacantes se pueden aprovechar de datos críticos, como tarjetas de créditos, que han sido expuestos por una mala gestión de estos.
- **A7 Ausencia de control de acceso a las funciones:** Son todas aquellas vulnerabilidades donde el atacante mediante la alteración del valor de una URL o de un parámetro acceda a una función sobre la cual no tiene privilegios
- **A8 Falsificación de peticiones en sitios cruzados (CSRF):** Permite a un atacante generar peticiones sobre una aplicación vulnerable a partir de la sesión de la víctima, estas incluyen la cookie de sesión de la víctima y otra información de autenticación.

- **A9 Uso de componentes con vulnerabilidades conocidas:** Corresponde a la explotación de librerías, frameworks y otros componentes vulnerables por parte de un atacante con el fin de obtener acceso o combinar con otros ataques.
- **A10 Redirecciones y reenvíos no validados:** Esto ataques aprovechan el uso de redirecciones de sitios web a otros sitios utilizando información no confiable (untrusted) para redirigir a las víctimas a sitios de phishing o que contienen malware.

Actualmente existe un nuevo listado del OWASP TOP 10 pero aún no se encuentra oficialmente liberado continúa en proceso de revisión, en la figura 2.3 podemos visualizar los 2 listados (2013 y 2017).

Figura 2.3: OWASP Top 10 2013 - 2017

2.2.3 OWASP Testing guide

Las Auditorías de Seguridad sobre Aplicaciones se han vuelto imprescindibles para evaluar la seguridad de los desarrollos. Uno de las metodologías con más reconocimiento internacional a la hora de realizar auditorías de seguridad sobre aplicaciones es la OWASP Testing Guide. La OWASP lanzó este proyecto con el objetivo de crear un marco que incluyera las mejores prácticas en el desarrollo de test de intrusión en aplicaciones.

La versión que revisaremos en este caso de estudio es la “OWASP Testing Guide v3” [7], esta guía se enfoca en diez áreas sobre las que se

deberá trabajar con tal de identificar potenciales fallos de seguridad que pudieran afectar a la aplicación. Estas áreas se presentan de la siguiente manera:

- **Recopilación de información:** Esta etapa de la auditoría de seguridad consiste en recopilar toda la información que se pueda sobre la aplicación cuya seguridad está siendo auditada.
- **Pruebas de gestión de configuración de la infraestructura:** En esta etapa se realiza un análisis de la infraestructura tecnológica sobre la que se encuentra desplegada la aplicación que se desea auditar.
- **Pruebas de autenticación:** en esta etapa se evaluarán todas las secciones de la web que estén relacionadas con los procesos de autenticación, el objetivo es tratar de eludir este mecanismo de ingreso.
- **Pruebas de administración de sesión:** En esta etapa se evalúan los controles de seguridad sobre la gestión de sesiones de una aplicación, el objetivo es verificar si se puede obtener acceso a cuentas de usuario sin necesidad de proporcionar credenciales correctas.
- **Pruebas de autorización:** En esta etapa se evalúa si es posible evadir los controles de autorización, si existe una vulnerabilidad en el traspaso de rutas o si es posible realizar un escalado de privilegios.
- **Pruebas de la lógica del negocio:** En esta etapa se trata de buscar aquellas funcionalidades en las cuales alterando el flujo normal del aplicativo puedan ser aprovechadas para beneficio del atacante. Las vulnerabilidades de este tipo pueden ser de las más graves de la aplicación y los más difíciles de identificar pues requieren un profundo conocimiento de la aplicación.
- **Pruebas de validación de datos:** En esta etapa se evalúa si las funcionalidades que reciben datos validan la información de forma correcta y si los datos son depurados de tal manera que no puedan afectar a los intérpretes que los vayan a utilizar.

- **Pruebas de denegación de servicio:** El objetivo fundamental en esta etapa es verificar si la aplicación objeto de estudio puede verse desbordada (ya sean sus funciones de red, su memoria, su espacio de disco, etc.) hasta conseguir que deje de funcionar adecuadamente.
- **Pruebas de servicios web:** Las pruebas de seguridad en esta etapa se centran en evaluar los servicios web, estos suelen utilizar tecnologías como SOAP, WSDL, UDDI; aquí debemos centrarnos en la búsqueda e identificación de las vulnerabilidades en dichas tecnologías.
- **Pruebas de AJAX:** En esta etapa se debe evaluar la seguridad de las llamadas asíncronas hechas con JavaScript. Recoge una serie de pruebas basadas en las secciones anteriores pero enfocadas a esta tecnología.

CAPÍTULO 3

ANÁLISIS DEL SISTEMA WEB DE VENTAS.

3.1 Análisis de arquitectura de aplicación web

En la figura 3.1 se visualiza la arquitectura actual del sistema web de ventas, se puede evidenciar que la aplicación web y la base de datos se encuentran alojadas dentro del mismo equipo. Los clientes internos y externos pueden acceder al sistema de ventas a través de internet o la red interna, adicional existen otras aplicaciones de la organización que utilizan información de la base de datos de este sistema para diferentes procesos e informes empresariales.

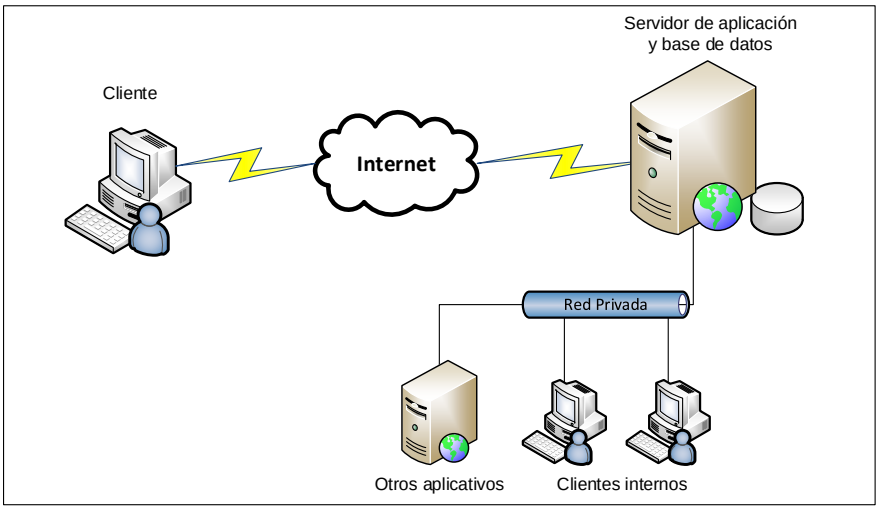


Figura 3.1: Arquitectura del sistema web de ventas

Las tecnologías que utiliza el aplicativo son las siguientes:

- Apache 2
- PHP versión 5.2.7
- MySQL versión 5.1.

3.2 Identificación de riesgos

Se realizó un análisis de la arquitectura y se hicieron “pruebas de penetración” manuales en el sistema web y se logró identificar los siguientes riesgos:

- Dado que el servidor de aplicaciones puede ser accedido de forma externa y si fuese víctima de un atacante, la base de datos también se vería comprometida.
- Si este equipo es víctima de un ataque de desborde, por ejemplo denegación de servicios, los demás aplicativos de la organización se verían afectados ya que utilizan la base de datos que se encuentra en el servidor vulnerado.
- Se puede aplicar inyección sql sobre el aplicativo el cual es una deficiencia muy crítica.
- Se pueden listar los archivos de un directorio del aplicativo web pudiendo descargase archivos que podrían tener información sensible de los clientes.
- Cuando se producen errores se puede visualizar información sensible del sistema.

3.3 Análisis de vulnerabilidades mediante revisión de código fuente

En esta sección vamos a evaluar el grado de seguridad del código fuente de la aplicación web de ventas, el objetivo es buscar vulnerabilidades que podrían ser explotadas por los atacantes, realizaremos una revisión de las siguientes características: Autenticación, Autorización, Gestión de Cookies, Validación de datos de entrada, Gestión de errores, Cifrado, Gestión de Sesiones.

A continuación detallamos las principales vulnerabilidades encontradas:

1. El control de sesiones no se maneja de forma apropiada, cualquier usuario que envíe por GET o POST la variable “alxuser” puede ingresar al sistema y puede saltar el mecanismo de autenticación. La función “extract” de PHP convierte en variables todos los valores dentro del arreglo que se le envía como parámetro. En la figura 3.2 se puede apreciar la sección del código mencionada.

```
ini_set('session.gc_maxlifetime', 36000);
session_start();
extract($_SESSION);
extract($_POST);
extract($_GET);
$bd = "[redacted]" . strtolower($clave_pais) ;


if (trim($alxuser) == "") {
    $ruta = "http://" . $_SERVER['HTTP_HOST']
     "/[redacted]/tmklogin.php?entrada=Es%20nece
    header("Location: " . $ruta);
    exit;
}
```

Figura 3.2: Control de sesiones deficiente

2. Los controles de autorización son deficientes ya que las URL’s de la aplicación pueden ser accedidas por cualquier usuario, en la figura 3.3 se puede verificar que no existen validaciones de privilegios en el aplicativo, si conocen la URL podrán ingresar a sobre la página y trabajar en ella.

3. La validación de datos de entrada en los formularios de la aplicación web es deficiente, es propenso a varios tipos de ataques de inyección de código. En la figura 3.3 se puede visualizar que no se realizan depuraciones ni validaciones de los datos de entrada.

```
<?
  include ("[redacted]alxprim.php");
  include ("[redacted]confalix.php");
  include_once ("[redacted]funciones.php");

  $conexion = conecta_sql2($ip, $bd,$usuario, $pwd );    // Password

  // Agregar el cliente
  if( $acc == 'a' ) {
     $consulta = "INSERT INTO clientes ( nombre, ejecutivo, ejetel1, ejetel2, ejedir1,
                  VALUES ( '$cnombre', '$cejecutivo', '$ctel1', '$ctel2', '$cdir1', '$c
     mysql_query($consulta,$conexion)
     or die( "No se pudo agregar el cliente <br> $consulta <br><br>" . mysql_error() );
  }
```

Figura 3.3: Deficiencias en validaciones y SQL

4. Varias páginas son vulnerables a ataques Cross Site Scripting (XSS). Esta vulnerabilidad puede permitir al atacante enviar código malicioso (JavaScript, VBScript, ActiveX, HTML) para engañar al usuario y obtener datos de él. En la figura 3.4 se puede apreciar un ejemplo muy sencillo de inyección de código JavaScript.

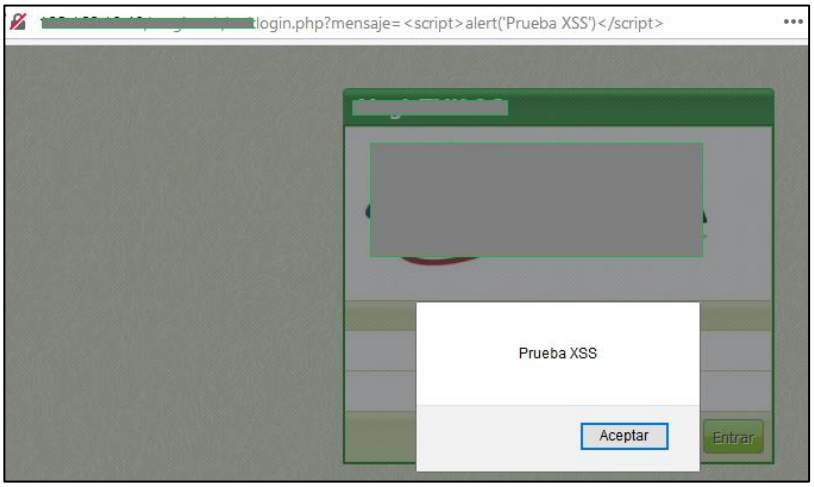


Figura 3.4: Ataque XSS

5. No existe parametrización en las consultas de base de datos por lo que esta aplicación es blanco fácil para ataques de Inyección SQL. En la figura 3.5 se puede apreciar como ingresando una “comilla simple” en el formulario de entrada la consulta SQL retorna un error.

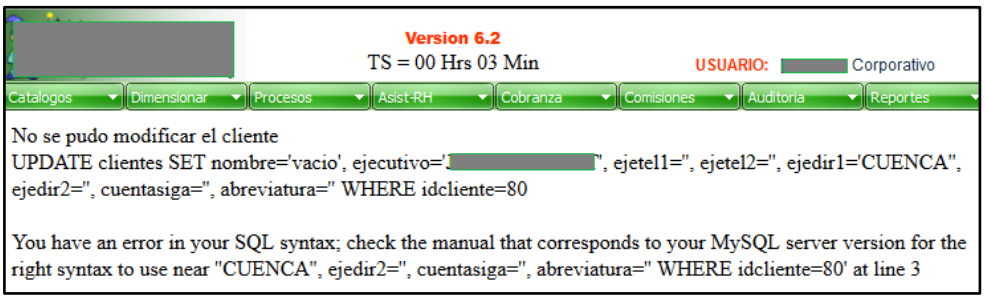


Figura 3.5: Ataque Inyección SQL

6. No existen restricciones en los accesos a las carpetas del aplicativo es posible listar los directorios y archivos fuentes que se encuentran dentro del sistema. En la figura 3.6 se puede evidenciar lo mencionado.

Figura 3.6: Configuraciones deficientes

7. Existen archivos .php que ejecutan procesos de actualización y regularización de datos, estos archivos se los utiliza para ejecutarlos a través de tareas programadas por lo cual no es necesario un proceso de autenticación para poder ejecutarlos, desde la URL también podrían ser ejecutados por cualquier usuario. En la figura 3.7 se puede evidenciar lo mencionado.

```php
<?php

$databaseHostDB = '[redacted]';
$databaseUserDB = '[redacted]';
$databasePassDB = '[redacted]';
$databaseBaseDB = 'call_center_pro';
$databasePortDB = '1433';


$conElastix = mysqli_connect($databaseHostDB, $databaseUserDB, $databasePassDB);

if (!mysqli_select_db($conElastix, $databaseBaseDB)) {
    echo " [:(] Error al seleccionar la base de datos BD: {$databaseBaseDB}, serv: $databaseH
    exit();
}

echo "paso conexM<br>";

$sql = "SELECT MAX(id_elastix) as id_elastix FROM $databaseBaseDB.elastix_call_progress_log";
echo $sql."<br>";
$rss = mysqli_query($conElastix, $sql);
while ($datas = mysqli_fetch_array($rss)) {
    if ($datas['id_elastix'] > 0) {
        $ultimo_reg = $datas['id_elastix'];
    }
}
```

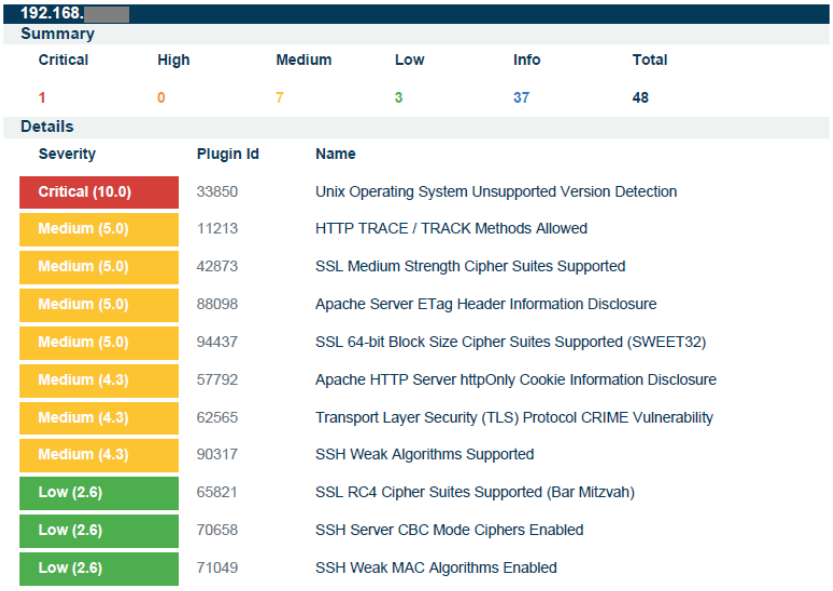
Figura 3.7: Archivos sin proceso de autenticación

8. En el archivo de configuración del sistema se pudo evidenciar que el usuario que utilizan para conectarse a la BD es el root el cual tiene

privilegios para todas las bases de datos que se encuentran en el servidor incluso la de mysql.

3.4 Análisis de vulnerabilidades dinámico con Nessus

Se procedió a realizar un análisis de vulnerabilidades con la herramienta Nessus versión 7.0.2, en la figura 3.8 se listan las principales deficiencias encontradas.



192.168.

Summary

Critical	High	Medium	Low	Info	Total
1	0	7	3	37	48

Details

Severity	Plugin Id	Name
Critical (10.0)	33850	Unix Operating System Unsupported Version Detection
Medium (5.0)	11213	HTTP TRACE / TRACK Methods Allowed
Medium (5.0)	42873	SSL Medium Strength Cipher Suites Supported
Medium (5.0)	88098	Apache Server ETag Header Information Disclosure
Medium (5.0)	94437	SSL 64-bit Block Size Cipher Suites Supported (SWEET32)
Medium (4.3)	57792	Apache HTTP Server httpOnly Cookie Information Disclosure
Medium (4.3)	62565	Transport Layer Security (TLS) Protocol CRIME Vulnerability
Medium (4.3)	90317	SSH Weak Algorithms Supported
Low (2.6)	65821	SSL RC4 Cipher Suites Supported (Bar Mitzvah)
Low (2.6)	70658	SSH Server CBC Mode Ciphers Enabled
Low (2.6)	71049	SSH Weak MAC Algorithms Enabled

Figura 3.8: Informe de vulnerabilidades con Nessus

En la imagen se puede apreciar que la falencia más critica que nos reporta Nessus es: “Unix Operating System Unsupported Version Detection” el cual indica que esta versión del sistema operativo ya no tiene soporte lo cual puede ser perjudicial si se presentan nuevas falencias de seguridad y dada la ausencia de soporte sobre esta versión del SO estas no puedan ser parchadas.

3.5 Informe de resultados

Como resultado de los pentesting realizados a través de los diferentes métodos de análisis de vulnerabilidades se identifica que el sistema web de ventas y el equipo físico tienen múltiples falencias de seguridad, y varias de ellas son criticas ya que pueden conllevar desde la divulgación de datos sensibles de un cliente hasta la pérdida total de la información del negocio. Los resultados que se obtuvieron son mostrados a continuación:

- Arquitectura del aplicativo web poco eficiente dado la naturaleza de sus funciones
- Versión del Sistema Operativo sin soporte
- Manejo ineficiente en el control de sesiones del aplicativo web
- Manejo ineficiente en el control de privilegios (autorización) del aplicativo web
- Validaciones de entradas de datos ineficientes dentro del aplicativo web, es propenso a ataques de inyección de código
- Consultas y transacciones SQL inseguras y no parametrizadas propensas a ataques de inyección SQL.
- Mala configuración del sistema de archivos del aplicativo web
- Usuario de bases de datos con privilegios y permisos inadecuados
- Archivos .php de ejecución en batch con permisos inadecuados

CAPÍTULO 4

ANÁLISIS DE RESULTADOS.

4.1 Evaluación de puntos críticos

De las vulnerabilidades encontradas luego del pentesting es importante identificar cuáles de ellas representan un mayor riesgo para nuestro sistema. A continuación explicaremos 4 vulnerabilidades que se consideran de potencial riesgo para el aplicativo y por ende para la organización:

- **Versión del Sistema Operativo sin soporte:** se considera critica esta deficiencia ya que significa que el SO no recibirá más parches de actualización y seguridad por ser considerado obsoleto, esto puede ser peligroso en la actualidad debido a que cada día aparecen nuevas amenazas y virus los cuales se solucionan mediante esta vía de actualización.
- **Arquitectura del aplicativo web poco eficiente:** se considera crítico ya que un solo equipo recibe toda la carga operativa (aplicación y base de datos) y puede atentar contra la disponibilidad de la información, esto podría perjudicar al resto de aplicaciones que utilizan la base de datos del sistema de ventas para emitir informes de análisis y toma de decisiones.
- **Validaciones de entradas de datos ineficientes:** esta vulnerabilidad es considerada una de las más críticas para el aplicativo, el usuario puede

hacer inyección de código malicioso para robar sesiones, redirigir a páginas comprometidas, atentar contra la base de datos, etc.

- **Consultas y transacciones SQL inseguras:** esta vulnerabilidad es la más crítica encontrada en el aplicativo web, un usuario puede incrustar código malicioso para obtener información sensible de la base de datos, puede borrar la BD del sistema e inclusive puede borrar la del propio motor de MySQL debido al usuario con que se conecta “root” y este tiene todos los privilegios.

4.2 Informe de mejoras implementadas

Posterior al análisis de vulnerabilidades y las amenazas más críticas del sistema se implementó una serie de modificaciones para mejorar la seguridad de la aplicación y esta sea menos vulnerable a los ataques que describimos en apartados anteriores. A continuación se detalla la lista con las principales mejoras realizadas:

- Se implementó POO para la conexión a la base de datos parametrizando todas las consultas y transacciones SQL, esto evita los ataques de Inyección SQL.
- Se implementó un objeto “Usuario” que contiene información de perfil y privilegios sobre el sistema, esto se hizo con el objetivo de mejorar el control de Sesión y Autorización.
- En cada página del aplicativo se agregó la validación de privilegios sobre las acciones que el usuario puede realizar.
- En cada página del aplicativo se agregó la validación de sesión autenticada, esto evita accesos no autorizados al sistema.
- Se borraron las funciones “extract” para las variables de Session, Get y Post, esto con el objetivo de evitar accesos no autorizados y escalamiento de privilegios.
- Se agregó un archivo “.htdocs” el cual evita que los usuarios puedan listar los directorios dentro del aplicativo web.

- En cada archivo PHP que es utilizado para ejecutarse como tarea programada se agregó el comando “php_sapi_name()” el cual impide que sea ejecutado desde un browser.
- Se agregaron validaciones y depuraciones de los datos que se ingresan a través de formularios web para evitar ataques de inyección de código malicioso.
- Se optimizó la arquitectura de la aplicación web, existe 1 equipo exclusivamente para la base de datos y otro equipo que trabaja como servidor de aplicaciones.
- Se crearon usuarios de base de datos exclusivos para cada aplicativo de la empresa, estos usuarios tienen los privilegios mínimos necesarios.
- Se implementó un archivo enrutador a través del cual pasan todas las peticiones del sistema, este archivo es el que controla la gestión de sesiones.

4.3 Definición de políticas y estándares de programación segura

Para mitigar potenciales fallos de seguridad en desarrollos futuros se definieron políticas y estándares de programación, todo el personal (programador) de sistemas es capacitado para que siga estas buenas prácticas al momento de codificar soluciones de software. A continuación se detallan los principales lineamientos que deben considerar:

- Se deben validar todos los datos de entrada tanto del lado del cliente como del lado del servidor y se deben depurar todos los caracteres que puedan afectar al intérprete web y de base de datos.
- Se deben validar que los usuarios solo puedan realizar acciones de acuerdo a su nivel de privilegios y su perfil de acceso.
- Se debe controlar que cualquier archivo que es utilizado para tareas programadas no puedan ser ejecutados desde un navegador.

- Se debe controlar que no se pueda realizar ninguna acción en el sistema sin antes haberse autenticado, no se pueden saltar el mecanismo de ingreso.
- Para realizar cualquier consulta hacia la base de datos se debe utilizar el objeto de conexión “MySqlConnection.php” que ofrece el sistema.
- No se puede liberar ningún cambio que no haya sido versionado bajo SVN y que no haya pasado por el proceso de pruebas de verificación del área de validación de software.
- No se deben utilizar plugins que no sean de fuentes confiables.
- Se deben utilizar las variables reservadas $_SESSION, $_GET y $_POST de PHP para la captura de datos del request, no se debe aplicar ningún comando de extracción sobre estas variables.

CONCLUSIONES Y RECOMENDACIONES

1. Es vital que las organizaciones hagan conciencia sobre la importancia de la seguridad de la información y no menospreciar el valor que este tiene, un análisis de vulnerabilidades de forma continua puede prevenir desastres informáticos y pérdidas invaluables.
2. Actualmente existen varias herramientas de pentesting inclusive Open Source, sin embargo, aún con la existencia de estos softwares las organizaciones continúan con un desconocimiento de cuándo deben ser utilizadas ignorando que pueden ser blanco de un ataque informático.
3. Es recomendable crear políticas y estándares de programación segura desde el inicio del desarrollo de una aplicación, el coste de modificación sobre un sistema ya finalizado es muy alto.
4. OWASP Top 10 y OWAS Testing guide son estándares muy útiles que pocas organizaciones conocen y que ayudan a mejorar el desarrollo aplicaciones para hacerlas más seguras.
5. Es importante evaluar la arquitectura de las aplicaciones considerando la carga operativa que tienen, una mala arquitectura provoca un funcionamiento inadecuado e inseguro y puede generar pérdidas en el negocio

BIBLIOGRAFÍA

[1] Alfonso Alfonso, Triangulo CIA [Online]. Disponible en: http://zonaasegurada.blogspot.com/2012/06/que-es-el-triangulo-cia.html

[2] Instituto Nacional de Tecnologías Educativas y Formación del Profesorado, Naturaleza de las amenazas [Online]. Disponible en: http://recursostic.educacion.es/observatorio/web/fr/software/software-general/1040-introduccion-a-la-seguridad-informatica?start=5

[3] TecnoXXI Seguridad Informática, Tipos de amenaza [Online]. Disponible en: https://www.tecnoxxi.com/blog/seguridad-informatica/amenazas-informaticas/

[4] G3orgx, Seguridad Informática: Tipos de ataque [Online]. Disponible en: https://blog.yottahack.net/hacking/2017/04/19/seguridad-informatica-tipos-de-ataques.html

[5] K. Graves, “Diferentes tipos de tecnologías del hacking”, CEH Certified Ethical Hacker, Abril 2010

[6] The OWASP Foundation, OWASP Top 10 [Online]. Disponible en: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[7] The OWASP Foundation, OWASP Top Testing guide [Online]. Disponible en: https://www.owasp.org/index.php/OWASP_Testing_Guide_v3_Table_of_Contents