|  |  |  |
| --- | --- | --- |
|  |  |  |

Secure Coding – Fase II

Prueba de Black-Box

GRUPO 4

Presentado por Grupo FivePlus

Álvaro López

Guillermo Molina León

Felipe Giraldo

Hernan Tenjo

Ehrlichmann Casas

2014

## TABLA DE CONTENIDO

TABLA DE CONTENIDO 2

1 INTRODUCCION 3

2 CICLO DE VIDA DEL CONOCIMIENTO DE LAS AMENAZAS Y CONTROLES 4

2.1 RECONOCIMIENTO 4

2.2 MAPEO 5

2.3 DESCUBRIMIENTO 6

3 COMPONENTES DE VERIFICACION y PRUEBAS GRUPO 4 (El nuestro) 7

4 Hallazgos Nuestros Aplicación 9

5 Hallazgos Aplicación GRUPO 8 11

6 LECCIONES APRENDIDAS 14

7 Anexos A – Grupo 4 15

8 Anexos B - Grupo 8 18

# INTRODUCCION

El presente documento corresponde al Entregable de la segunda fase para el proyecto del grupo y del grupo No. 8 el cual fue asignado por Martin Ochoa.

El documento se enmarca en el desarrollo inicial de la aplicación la cual solo tiene los requerimientos funcionales y lo básico de seguridad. Para la segunda fase se deberá analizar únicamente las vulnerabilidades de nuestra aplicación y de la aplicación del grupo 8.

El alcance de esta fase de trabajo es reconocimiento, mapeo y descubrimiento de vulnerabilidades de la aplicación**.**

# CICLO DE VIDA DEL CONOCIMIENTO DE LAS AMENAZAS Y CONTROLES

Para iniciar nuestra exploración en los ataques a aplicaciones y sus posibles salvaguradas y contramedidas emplearemos unos pasos los cuales son Conocimiento del objetivo, Mapeo de las posibles vulnerabilidades, Descubrimiento de las mismas (no sea falsos positivos) y por último Explotar dichas vulnerabilidades.



Gráfica tomada del libro The Web Application Hacker´s Handbook – Dafydd Stuttard, Marcus Pinto

## RECONOCIMIENTO

Es la identificacion de tecnologias de un sitio objetivo a evaluar o valorar en fortalezas y/o debilidades técnicas informáticas. Igualmente se realiza un análisis funcional de la apliación, una línea base de componentes y un modelo de flujo de procesos. Podemos tener las siguientes actividades muy globalmente sin tocar los ambientes de la aplicación:

* Seleccionar un objetivo o aplicación web,
* Determinar que técnologia usa el sitio web, portal y/o aplicación
* Sus configuraciones en infraestructura
* Tipos de Usuarios
* Navegar por sus diferentes anillos (Red, DB, App Server, Apliación)
* Verificar como es su autenticación y Autorización

Algunas de las herramientas que podemos utilizar son:

* WHOIS, es provista por LACNIC es un sistema distribuido de consultas de información sobre recursos de Internet que permite obtener información sobre el propietario de un nombre de dominio o una dirección IP en Internet.
* Fierce Domain Scanner, framework de reconocimiento web-base. Muestra el rango de direccionamiento implementado , la IP del dominio, tiempo de escaneo, etc.
* Google Hacking Database (GHDB) / Google Dorks, Busqueda de Información en profundidad mediante el motor de busqueda Google.
* Recon-ng, es un framework web  escrito en Python. Tiene módulos independientes, la interacción de bases de datos. Recon-ng proporciona un potente entorno en el que puede ser de código abierto basado en la Web de reconocimiento llevado a cabo de forma rápida y completamente.
* Dig, es una herramienta para realizar diferentes tipos de consultas a servidores dns, fácilmente integrable con soluciones de scripting.

## MAPEO

En esta fase es de las más importantes del ciclo de vida de un ataque, dado que en esta se relacionan toda la infraestructura y sus posibles vulnerabilidades y servicios utilizados por las aplicaciones. Igualmente se obtienen sus versiones y los posibles enlaces que tiene una aplicación. Se relacionan los formularios, campos y posibles datos que puede aceptar una aplicación.

Encontramos entre muchas herramientas tales como:

* Nmap, herramienta de escaneo de puertos y servicios. Tiene un motor de scripting como escáner de vulnerabilidades que además puede encontrar otras debilidades en la configuración.
* SqlMap, variante de Nmap pero únicamente para inyecciones hacia las bases de datos.
* Burpsuite y Zep Attack Proxy – ZAP. Son proxys de seguridad utilizados para pruebas de seguridad y ataques a sitios web específicamente. Pueden interceptar y modificar el trafico que va hacia y/o desde una aplicación y su navegador. Igualmente tienen una araña con reconocimiento de aplicaciones, contenido y funcionalidad. Igualmente tienen muchas otras funcionalidades y módulos para descubrir y explotar vulnerabilidades.
* Nikto, es un escáner de vulnerabilidades. Se encarga de efectuar diferentes tipos de actividades tales como, detección de malas configuraciones y vulnerabilidades en el servidor objetivo, detección de ficheros en instalaciones por defecto, listado de la estructura del servidor, versiones y fechas de actualizaciones de servidores, tests de vulnerabilidades XSS, ataques de fuerza bruta por diccionario, reportes en formatos txt, csv, html, etc.
* W3AF (Web Application Attack and Audit Framework) es una herramienta que permite detectar vulnerabilidades web y explotarlas. Básicamente trabaja con 4 módulos; Configuración del análisis, se indica el objetivo y se seleccionan los plugins o escáneres que se desean utilizar; en Logs se puede ver el estado del proceso; en Resultados las vulnerabilidades detectadas con lujo de detalles (SQL Injection, Cross Site Scripting, Full Path Disclosure, File Inclusion, etc); y finalmente desde la pestaña Exploit se pueden explotar estos fallos

## DESCUBRIMIENTO

Verificar si existen vulnerabilidades en toda la infraestructura, apliaciones, campos, etc. Verificar si las vulnerabiliades que presentan las herramientas automáticas no son falsos positivos.

Algunas herramientas utilizadas son:

* Skipfish, escáner de seguridad para aplicaciones web. Esta aplicación está orientada a ser eficiente, veloz y con baja incidencia de falsos positivos. Genera un mapa interactivo para la página web que queramos visitar; para realizar esto se apoya en un rastreo recursivo del sitio en internet y en un amplio diccionario basado en sondas.
* W3af, Web application attack and audit framework es un escáner de vulnerabilidades en aplicaciones web que provee información relativa a vulnerabilidades comunes encontradas en las aplicaciones web actuales
* CEWL, Es una herramienta generadora de diccionarios de palabras clave que puede ser utilizada por programas como cain & Abel, John the riper y demás para realizar ataques de fuerza bruta. La herramienta busca en un sitio web y en links relacionados desde esta aplicación por palabras únicas que sirvan para crear un diccionario.
* Dirbuster, Aplicación diseñada para obtener por fuerza bruta los directorios y archivos de un sitio web.
* ZAP y/o Burpsuite.

# COMPONENTES DE VERIFICACION y PRUEBAS GRUPO 4 (El nuestro)

La siguiente información fue utilizada del documento “OWASP Secure Coding Practices Quick Reference Guide – 2011”, The OWASP Foundation.

Se presentan algunos de los controles propuestos en el documento.

|  |  |
| --- | --- |
| COMPONENTE | DESCRIPCION |
| VALIDACION DE ENTRADAS Y SALIDAS | Verificar que las entradas de datos de la aplicación sean confiables, donde se utilice rutinas o API . Codificar los datos a un set de caracteres común antes de validar ([Canonicalización](#Canonicalizar)).   * Validar datos redireccionados * Validar tipos de datos no esperados * Validar rangos de datos * Validar longitud de datos. * Validar toda entrada con una whitelist * Sanitice todas las salidas de datos. |
| ADMINISTRACION DE AUTENTICACION Y CONTRASENAS | * Autenticar todas los recursos del sistema a menos que sean de carácter público. * Utilizar sistemas conocidos * Cambiar contraseñas por defecto y de usuarios privilegiados o del sistema |
| ADMINISTRACION DE SESIONES | * Utilizar los controles del servidor o del framework para la administración de sesiones. La aplicación solo debe reconocer estos identificadores como válidos * Logout * Las sesiones que no sean persistentes * Utilice sesiones OTP por cada recurso |
| CONTROL DE ACCESO | * Utilice o cree un sistema de autorización y control de acceso fiable y probado. * Separar lógica privilegiada de otro código de la aplicación. * Restringir acceso a ficheros u otros recursos, incluyendo aquellos fuera del control directo de la aplicación, únicamente a usuarios autorizados |
| CIFRAMIENTOS | * Utilice estandares conocidos y actualizados * Proteja las claves o frases |
| MANEJO DE ERRORES Y LOGS | * Todo error debe ser capturado y manipulado * Sanitice las salidas * Configurar las cookies con el atributo HttpOnly |
| PROTECCION DE DATOS | * Principio del menos privilegiado * Manejo de información cuando ya no se utilice en cualquiera de sus estados (almacenada, procesamiento y transmisión). |
| COMUNICACIONES CIFRADAS | * Utilizar mecanismos seguros, confiables y probados. * Utilice las versiones sugeridas por el proveedor |
| CONFIGURACION | * Tenga un inventario de los activos de su sistema * Aplique Hardening a cada activo * Privilegios de los directorios |
| BASE DE DATOS | * Utilice validación de las entradas y codificación de las salidas y asegúrese de manejar los meta caracteres. * Utilice procedimientos almacenados para abstraer el acceso a los datos y elimine los permisos de las tablas en la base de datos * Cifre la información sensible * Cambie las contraseñas de los usuarios privilegiados |
| MANEJO DE ARCHIVOS | * Sanitice las entradas * No utilizar directamente información provista por el usuario en ninguna operación dinámica * Evite o restrinja la transferencia de archivos que puedan ser interpretados por el servidor web (por ejemplo: asp, php, jsp, etc). * Eliminar los permisos de ejecución a los archivos transferidos * Asegúrese que los archivos y recursos de la aplicación sean de solo lectura |
| MANEJO DE MEMORIA | * Utilice controles en la entrada y la salida de información no confiable. * Revise dos veces que el largo de los buffer sean los requeridos y especificados. * Cuando utilice primitivas que requieran el número de bytes a copiar, como ser strncpy(), sea consciente que si el tamaño del buffer destino es igual al tamaño del buffer origen, el destino podría quedar son el NULL-byte requerido del final. * Verifique los límites de los buffers si se llama a las funciones dentro de un loop y asegúrese de no correr riesgo de escribir fuera del espacio reservado. |

# Hallazgos Nuestros Aplicación

|  |  |  |
| --- | --- | --- |
| COMPONENTE | RIESGO | EJEMPLO DEBILIDAD |
| VALIDACION DE ENTRADAS Y SALIDAS | Medio | * Valida caracteres especiales para inyección * No Valida longitud de campos. En ningún campo. * No Valida tipo de datos al efectuar una transaccion permite datos no númericos. * Las salidas son muy específicas y permite recolectar información sensible. Ej. Formulario de crear cuenta o olvido de contraseña el error dice que falta o si el usuario existe pero hay un campo repetido o ya existente. * No se validan las entradas del archivo en batch /localhost/loginsys/loadBatchFile.php |
| ADMINISTRACION DE AUTENTICACION Y CONTRASENAS | ALTO | * No tiene políticas de contraseñas, se puede tener una contraseña debil. * Las contraseñas son debiles, utilizamos el sitio web gratis   <http://www.hashkiller.co.uk/md5-decrypter.aspx> y colocamos el hash de la tabla users  7e04da88cbb8cc933c7b89fbfe121cca MD5 : Felipe |
| ADMINISTRACION DE SESIONES |  | * Herramientas como ZAP y BURP SUITE nos determinan y probamos que las cookies quedan abiertas sino se realiza logout. * No se maneja HTTPonly * No utiliza OTP para cada página * Las cookies pueden ser accedidas <http://localhost/loginsys/index.php?do=index>, Parameters: PHPSESSID utilizando BurpSuite o ZAP |
| CONTROL DE ACCESO | ALTO | * No se tiene control de acceso, si el usuario ingresa a la aplicación ya puede ir a cualquier aplicación con solo referenciar la URL * Si no termina la sesion y cierra el browser, otro usuario puede abrir la aplicación y continuar con la sesión del anterior usuario. * Se puede realizar path transversal |
| CIFRAMIENTOS | ALTO | * No se utiliza SSL/TSL en la transmisión de la información. * Las bases de datos no utilizan ningún campo cifrado. |
| MANEJO DE ERRORES Y LOGS | BAJO | * Se tiene manejo de errores |
| PROTECCION DE DATOS | MEDIO | * No se utiliza el principio del menos privilegiado porque se tiene roles genericos |
| COMUNICACIONES CIFRADAS | ALTO | No se tiene SSL/TSL |
| CONFIGURACION | ALTO | * Se tienen los usuarios por defecto en la infraestructura Mysql/Root, Samurai * Se debe asegurar el Apache dado que permite realizar path transversal, por ejemplo:   <http://localhost/loginsys/uploads/>  <http://localhost/loginsys/assets/?C=S%3BO%3DA>  <http://localhost/loginsys/assets/js/global.js> |
| MANEJO DE ARCHIVOS | ALTO | * Se puede enviar cualquier archivo respetando el formato o mascara de los campos, pero un usuario puede colocar una inyección en uno de los campos, por ejemplo en el módulo transferencias /localhost/loginsys/loadBatchFile.php * Se puede colocar un campo con cualquier longitud y formato en el archivo. * Los permisos de los directorios estan abiertos. |
| MANEJO DE MEMORIA | MEDIO | Se tiene controles para el desbordamiento de memoria. En la carga masiva de transacciones se realizaron las pruebas. |

# Hallazgos Aplicación GRUPO 8

Se realizaron pruebas automáticas y manuales para la aplicación del grupo 8. Se resumen en la siguiente tabla y en el Anexo B se dejan evidencias en más detalle.

|  |  |  |
| --- | --- | --- |
| COMPONENTE | RIESGO | EJEMPLO DEBILIDAD |
| VALIDACION DE ENTRADAS Y SALIDAS | Medio | * Valida caracteres especiales para inyección * No Valida longitud de campos. En ningún campo. * El formato de registro, parecen no realizar filtrado de las entradas. Es posible crear nombres de usuario como shiiv’ah sin ningún problema. * En el registro de clientes, se insertó en el campo del correo un java script Alert, que se ejecutó al momento de listar los usuarios (XSS). |
| ADMINISTRACION DE AUTENTICACION Y CONTRASENAS | ALTO | * No tiene políticas de contraseñas, se puede tener una contraseña débil. * No utiliza CAPTCHAS * La aplicación no bloquea a un posible atacante después de varios intentos fallidos. Esto lo hace susceptible a ataques de fuerza bruta. * Al poner como correo el siguiente código, se logró registrar un usuario que al ser cargada la página de activación se intenta autorizar automáticamente permitiendo el acceso a cualquier usuario:   <script>  window.onload=function(){  var xmlHttp = new XMLHttpRequest();  xmlHttp.open( "GET", "[http://localhost/clientes.php?activar=11](http://localhost/clientes.php?activar=11" \t "_blank)", false);  xmlHttp.send(null);  xmlHttp.responseText;  };  </script> |
| ADMINISTRACION DE SESIONES |  | * No se maneja HTTPonly * No utiliza OTP para cada página (susceptible a CSRF). |
| CONTROL DE ACCESO | ALTO | * Es posible acceder a la página principal de la aplicación sin estar logueado. Sin embargo no muestra información. * Si no termina la sesión y cierra el browser, otro usuario puede abrir la aplicación y continuar con la sesión del anterior usuario. * En la carpeta /exe/ se pueden revisar los archivos cargados para transacciones masivas. Esto permite conocer información de cuentas de clientes y tokens utilizados. Esto abre un riesgo de SQL-i, brute forcing y XSS |
| CIFRAMIENTOS | ALTO | * No se utiliza SSL/TSL en la transmisión de la información. |
| MANEJO DE ERRORES Y LOGS | BAJO | * No se centralizan los errores en términos comunes, es decir no dar demasiada información que puede ser utilizada por externos. |
| PROTECCION DE DATOS | MEDIO | * No se utiliza el principio del menos privilegiado porque se tiene roles genéricos * Es posible realizar transacciones de transferencia con la cuenta de origen siendo similar a la de destino. * La aplicación no valida si un usuario se registra varias veces con el mismo correo. Esto conlleva a colusión de información y perdida de confidencialidad dado que coloco el mismo correo del admin y podemos pedir que refresque la contraseña. |
| COMUNICACIONES CIFRADAS | ALTO | No se tiene SSL/TSL |
| CONFIGURACION | ALTO | * Se tienen los usuarios por defecto en la infraestructura. Como Samurai y MySql. * Realizando una verificación simple de caja blanca, se encontró la utilización de mysql\_real\_escape\_string. Esta librería se encuentra obsoleta para las nuevas versiones de php, lo que genera un gran riesgo su utilización. |
| MANEJO DE ARCHIVOS | ALTO | * En la carpeta clientes, el archivo index permite revisar la información de clientes por aprobar, sin estar logueado.. * Se puede colocar un campo con cualquier longitud y formato en el archivo. |
| MANEJO DE MEMORIA |  | * No se obtuvieron hallazgos. |

# LECCIONES APRENDIDAS

1. La seguridad de las aplicaciones normalmente reposa en tener una autenticación, control de acceso y logs, pero en verdad existen muchas vulnerabilidades las cuales los desarrolladores no tienen consciencia de su existencia y pueden generar falsa sensación de seguridad.
2. Se debe aprender a utilizar las herramientas básicas para las pruebas de vulnerabilidades y poder complementarlas con una verificación manual.
3. Existen muchas metodologías de pruebas de vulnerabilidades lo cual las organizaciones deberán al menos saber y utilizar alguna de ellas.
4. Sensibilizar a los desarrolladores y al gobierno de la seguridad en las pruebas de código.
5. Se debe poder garantizar al menos que el top 10 de OWASP sea revisado en cada aplicación en desarrollo y pruebas.
6. Utilice las aplicaciones y/o herramientas de explotación que se complementen entre ellas y no probar de todas, esto último evita generar falsos positivos.

# Anexos A – Grupo 4

Lo primero que realizamos es un mapa de todo el servidor para poder saber de todos los servidores, aplicaciones y versiones utilizados en la infraestructura del servidor.

Con nmap se observa que se tiene un Linux 2.6.X, una base de datos Mysql 5.x y puertos 80 y 443 abiertos.

Los resultados fuerón:

Starting Nmap 6.25 ( http://nmap.org ) at 2014-06-21 18:42 COT

Nmap scan report for localhost (127.0.0.1)

Host is up (0.000055s latency).

Not shown: 992 closed ports

PORT STATE SERVICE VERSION

**22/tcp** open ssh OpenSSH 5.9p1 Debian 5ubuntu1.1 (Ubuntu Linux; protocol 2.0)

| ssh-hostkey: 1024 bb:16:71:8f:d7:d0:ec:66:74:62:75:4a:26:50:37:4c (DSA)

| 2048 60:91:1d:16:d9:4a:0c:01:07:9d:4b:5f:bc:ac:6f:dc (RSA)

|\_256 f6:4a:07:a2:07:d7:5f:d2:b1:b4:13:a5:90:e6:51:d8 (ECDSA)

**80/tcp** open http Apache httpd 2.2.22 ((Ubuntu))

|\_http-methods: POST OPTIONS GET HEAD

|\_http-title: Site doesn't have a title (text/html).

**443**/tcp open http Apache httpd 2.2.22 ((Ubuntu))

|\_http-methods: POST OPTIONS GET HEAD

**631**/tcp open ipp CUPS 1.5

| http-methods: GET HEAD OPTIONS POST PUT

| Potentially risky methods: PUT

|\_See http://nmap.org/nsedoc/scripts/http-methods.html

| http-robots.txt: 1 disallowed entry

**3306**/tcp open mysql MySQL 5.5.32-0ubuntu0.12.04.1

| mysql-info: Protocol: 10

| Version: 5.5.32-0ubuntu0.12.04.1

| Thread ID: 1661

| Some Capabilities: Long Passwords, Connect with DB, Compress, ODBC, Transactions, Secure Connection

| Status: Autocommit

|\_Salt: @!$.[\_ME

Running: Linux 2.6.X|3.X

OS CPE: cpe:/o:linux:linux\_kernel:2.6 cpe:/o:linux:linux\_kernel:3

CSRF

Riesgo: Alto

* Si se invoca la pantalla http://localhost/loginsys/main.php?do=batchTransfers y no verifica que se tenga una sesión activa. Lo cual se puede cargar el archivo que se quiera de transacciones.
* Se carga y se envia dando a conocer el archivo PHP que lo ejecuta: /localhost/loginsys/loadBatchFile.php

El archivo tiene este formato:

1234, 78901,2000,'TOKEN-123456789 ',1

1234,78901,2000,'TOKEN-123456789',1

87654,78901,3000,'TOKEN-234567890',1

3456,78901,4000,'TOKEN-345678901',1

se puede vulnerar y la función en C no la valida, la función valida el token lo cual inyectamos un sql injention como parametro.

Nuevo Archivo:

1234333333333222222222222222,78901,1000,' or 1=1 or '',1

1234,78901,2000,'TOKEN-123456789',1

87654,78901,3000,'TOKEN-234567890',1

3456,78901,4000,'TOKEN-345678901',1

Directory Browsing o Path Transversal

Riesgo: Medio

Podemos realizar Path Transversal y navegar por el directorio del sistema

<http://localhost/loginsys/uploads/>

<http://localhost/loginsys/assets/?C=S%3BO%3DA>

<http://localhost/loginsys/assets/js/global.js>

XSS - Cookie set without HttpOnly flag

Riesgo: Medio

session hijacking, las cookies pueden ser accedidas desde un javascript externo y ser trasmitida a otro lugar,

http://localhost/loginsys/index.php?do=index

Parameters: PHPSESSID

Malas Configuraciones

* No se utiliza SSL para ninguna transacción.
* Los usuarios privilegiados siguen con las contraseñas por defecto. Root, Mysql.
* En la pantalla de reseteo de contraseña se puede empezar a deducir los usuarios dado que el mensaje de error es exacto. Por ejemplo si el problema es que no concuerda el correo lo dice, luego deducimos que el usuario si existe.
* El ingreso de los usuarios no tiene CAPTCHA luego se hace susceptible a ataques de fuerza bruta.
* No tiene control de contraseñas, se puede de cualquier longitud y sin ninguna política
* Se salta el control de acceso, se puede invocar páginas directamente sin autenticación.
* La sesión se mantiene activa así se haya cerrado el browser.
* Acepta caracteres no numéricos en el campo de valor del módulo de transacciones.

# Anexos B - Grupo 8

Hallazgos en forma manual

* Injection (SQL, Shell, LDAP, XPath, Hibernate)

Las entradas al parecer están protegidas de injections, no se logró inyectar ningún campo.

* Broken Authentication and Session Management

Al poner como correo el siguiente código, se logró registrar un usuario que al ser cargada la página de activación se intenta autorizar automáticamente permitiendo el acceso a cualquier usuario. Sin embargo al parecer la función de autorización se ejecuta posteriormente al envió del correo y al generarse un error en el envió del correo no se logra autorizar el nuevo usuario, pero puede ser cuestión de tiempo para encontrar una combinación correcta.

<script>

window.onload=function(){

var xmlHttp = new XMLHttpRequest();

xmlHttp.open( "GET", "[http://localhost/clientes.php?activar=11](http://localhost/clientes.php?activar=11" \t "_blank)", false);

xmlHttp.send(null);

xmlHttp.responseText;

};

</script>

* Cross Site Scripting (XSS)

En el registro de clientes, se insertó en el campo del correo un java script Alert, que se ejecutó al momento de listar los usuarios.

* Insecure direct object references

Se pudo observar que cuando se realiza la activación de un usuario se invoca la url

"[http://localhost/clientes.php?activar=11](http://localhost/clientes.php?activar=11" \t "_blank)" lo que podría explotarse.

* Security Misconfiguration

No se evidencia el uso de https para ninguna de las operaciones en el sitio, lo que podría ser explotado por un sniffer.

* Sensitive Data Exposure

Junto con XSS y Framing, se puede poner en peligro la exposición de información de la sesión.

* Missing Function Level Access Control

No se cuenta con mecanismos que eviten los ataques de fuerza bruta  (p.e. captchas o contadores de intentos fallidos para bloquear cuentas)

* Cross Site request forgery (CSRF)

La aplicación permite hacer framing, lo que combinado con clickjacking podría comprometer la seguridad al hacer que el usuario haga clicks en una máscara que oculte las verdaderas funcionalidades ejecutadas.

* Using known vulnerable components

Al realizar pruebas de SQL injection no se pudo ejecutar el procedimiento, por lo que se asume que se está usando algún tipo de librería que permite escapar los caracteres peligrosos.

Realizando una verificación simple de caja blanca, se encontró la utilización de mysql\_real\_escape\_string. Esta librería se encuentra obsoleta para las nuevas versiones de php, lo que genera un gran riesgo su utilización.

Hallazgos con herramientas automáticas

|  |  |
| --- | --- |
| **Ejemplo** | **Nivel de riesgo** |
| Es posible navegar en medio de las carpetas asociadas a la aplicación. Se hizo la prueba con las carpetas /images/, /exe/, /include/ entre otras. Es perfectamente posible descargar datos de dichas carpetas. | Medio |
| Es posible conocer el id de sesión de la aplicación, habilitando el riesgo de CSRF y Xss | Medio |
| No se utiliza la directiva HTTPOnly |  |
| En el formato de registro, parecen no realizar filtrado de las entradas. Es posible crear nombres de usuario como shiiv’ah sin ningún problema. Esto puede acarrear problemas de SQL-i | Medio |
| Es posible acceder a la página principal de la aplicación sin estar logueado. Sin embargo no muestra información. | Alto |
| Es posible realizar transacciones de transferencia con la cuenta de origen siendo similar a la de destino. | Medio |
| Su política de contraseñas es baja, lo cual lo hace susceptible a ataques de fuerza bruta | Medio |
| La aplicación no bloquea a un posible atacante después de varios intentos fallidos. Esto lo hace susceptible a ataques de fuerza bruta. | Alto |
| La aplicación no valida si un usuario se registra varias veces con el mismo correo. | Medio |
| La aplicación no verifica la clave ingresada al momento de registro, lo cual puede inducir a errores. | Bajo |
| Utilizando la URL /index.php?username=&password=&Submit=Ingresar, el sistema parece admitir que el usuario esta registrado dentro de la aplicación. Esto acarrea un riesgo de ataques CSRF | Medio |
| En la carpeta clientes, el archivo index permite revisar la información de clientes por aprobar, sin estar logueado. | Medio |
| En la carpeta /exe/ se pueden revisar los archivos cargados para transacciones masivas. Esto permite conocer información de cuentas de clientes y tokens utilizados. Esto abre un riesgo de SQL-i, brute forcing y XSS | Alto |
| Hay carpetas ocultas en la aplicación, como la de Login, que permite ver mayor información del usuario. Si se investiga un poco más a fondo, puede revelar datos sensibles. | Bajo |
| el comando /server-status proporciona información del servidor, que puede ser utilizada para realizar explots conocidos. | medio |