Published on *Marco de Desarrollo de la Junta de Andalucía* (http://127.0.0.1/servicios/madeja)

Cifrado

Código: SEG_Cifrado

Una de las principales medidas para asegurar la integridad y la confidencialidad de la información que se transmite a través de la red es la encriptación o codificación de los mensajes, evitando que, aún interceptando nuestra comunicación, no sea posible su entendimiento. Para ello, se resumen diversas situaciones en las que se debe cifrar la información y los algoritmos a utilizar.

El **cifrado de datos** es el proceso por el que una información legible se transforma mediante un algoritmo (llamado ***cifra***) en información ilegible, llamada *criptograma* o *secreto*.

Esta información ilegible se puede enviar a un destinatario con muchos menos riesgos de ser leída por terceras partes. El destinatario puede volver a hacer legible la información (***descifrarla***) introduciendo la **clave del cifrado**.

La seguridad de un buen sistema de cifrado depende enteramente de la clave, y no debe depender del algoritmo de cifrado usado. Es decir, el algoritmo de cifrado a menudo es público, y es conocido por los posibles atacantes, pero si el algoritmo es bueno, esto no debe bastarles para descifrar el mensaje.

Los algoritmos usados en las comunicaciones seguras de Internet son públicos prácticamente siempre, por lo que es necesario centrarse en crear claves suficientemente seguras.

Además, la capacidad computacional de los ordenadores crece constantemente y cada vez son capaces de probar más y más claves por segundo de forma que puedan encontrar la clave simplemente probando una y otra vez.

No debe confundirse la clave del cifrado con las palabras de paso usadas para acceder a algunas aplicaciones: por ejemplo, para acceder a un cliente de correo online es necesaria una contraseña, que es enviada desde la ventana del explorador al servidor para que procese la petición de login. En este caso, la fuerza bruta (probar sucesivamente todas las claves posibles), es inútil, ya que casi todas las aplicaciones tienen limitado el número de intentos. No obstante, esa contraseña que enviamos desde el navegador, se envía cifrada al servidor a través de Internet. Si alguien consiguiera captar la información en la que viaja la contraseña sí podría introducir ese texto cifrado en una aplicación de criptoanálisis e intentar descifrarla y después usarla.

Objetivos

- Asegurar la confidencialidad e integridad de la información

Código	Título	Tipo	Carácter
LIBP-0270	Algoritmos de cifrado	Directriz	Obligatoria
LIBP-0256	Almacenamiento de claves	Directriz	Obligatoria
LIBP-0274	Encriptación de la información sensible	Directriz	Obligatoria
PAUT-0235	Espacio de claves apropiado	Consejo	
PAUT-0202	Generación de tokens seguros	Directriz	Obligatoria
PAUT-0200	Transmisiones de datos	Directriz	Obligatoria

Código	Título	Tipo	Carácter
RECU-0581	Almacenamiento de las contraseñas en PHP	Ejemplo	Obligatorio
RECU-0584	Almacenar datos encriptados en una base de datos o un fichero en PHP	Ejemplo	Obligatorio
RECU-0574	Código vulnerable con contraseña sin encriptar	Ejemplo	Recomendado
RECU-0570	Conexión sin encriptación de la información	Ejemplo	Obligatorio
RECU-0596	Criptología de empleo en el Esquema Nacional de Seguridad	Especificación	Recomendado
RECU-0593	Encriptación de los identificadores de sesión en PHP	Ejemplo	Obligatorio
RECU-0583	Encriptar y desencriptar datos en PHP	Ejemplo	Obligatorio
RECU-0607	Generar números aleatorios fuertes en Java	Ejemplo	Recomendado
RECU-0571	Guía de referencia sobre la arquitectura de criptografía en Java	Referencia	Recomendado
RECU-0554	Introducción al cifrado	Referencia	Recomendado
RECU-0595	Protección de los datos a direfentes niveles	Referencia	Recomendado
RECU-0669	Uso de claves en el código fuente	Ejemplo	No recomendado

Algoritmos de cifrado

- **Área:** Cifrado
- **Tipo de pauta:** Directriz
- **Carácter de la pauta:** Obligatoria

Código: LIBP-0270

Modificar las condiciones del algoritmo de cifrado en función de su tipología

Recientemente se han descubierto que numerosos algoritmos considerados "seguros" son criptográficamente débiles. Esto significa que en lugar de realizar 2^80 operaciones para hacer un ataque fuerza bruta a una clave, puede llevar solamente 2^69 operaciones, algo alcanzable en una máquina normal.

Conforme la criptografía moderna confía en ser computacionalmente costosa de romper, en estándares específicos pueden configurarse tamaños de clave que proporcionen seguridad con la tecnología y el conocimiento actual, y que asegure que el coste de descifrar la clave sea elevado. Por lo tanto, necesitamos asegurarnos que tanto los algoritmos como el tamaño de la clave sean tenidos en cuenta en la selección de un algoritmo.

Pautas

Título	Carácter
Algoritmo simétrico	Obligatoria
Algoritmo asimétrico	Obligatoria
Algoritmos hash	Obligatoria
Algoritmos de cifrado de flujo	No Recomendada

Algoritmo simétrico

Utilizar un tamaño de clave de 128 bits (estándar para SSL) en general

Un tamaño de clave de 128 bits (estándar para SSL) es suficiente para la mayoría de aplicaciones.
Considere 168 o 256 bits para sistemas seguros tales como grandes transacciones financieras.

Volver al índice

Algoritmo asimétrico

Utilizar un tamaño de clave de 1024 bits en general

Tamaños de clave de 1024 bits son suficientes para la mayoría de aplicaciones personales.
1536 bits deberían ser aceptables actualmente para la mayoría de aplicaciones seguras.
2048 bits son recomendables para aplicaciones altamente protegidas.

Volver al índice

Algoritmos hash

Utilizar un algoritmo de hash lo más actual posible

Los algoritmos hash pueden presentar colisiones que permitirían a un atacante proporcionar una contraseña suplente sin pasar por la autenticación. Para mitigar esta situación, habrá que utilizar un algoritmo considerado fuerte actualmente, como pueden ser los algoritmos **SHA-256** ó **SHA-512**.

Los algoritmos **MD-4, MD-5 y SHA-1 son vulnerables** por lo que no deberían ser utilizados.

Volver al índice

Algoritmos de cifrado de flujo

No usar los cifrados de flujo ya que son vulnerables

Los cifrados de flujo, como RC4, son vulnerables debido a una propiedad que poseen este tipo de cifrados:

Si se utiliza la misma clave para “proteger” dos documentos diferentes, el flujo de claves se abandona cuando se realiza la operación XOR con los dos documentos, dejando en texto plano los dos documentos de la operación XOR. Los dos documentos en texto plano se pueden recuperar usando un análisis de frecuencia.

Por lo tanto, no deben usarse cifradores de flujo de esta manera. En su caso, habrá que considerar el uso de **algoritmos simétricos fuertes** como AES.

Volver al índice

Recursos

Área: Desarrollo » Seguridad » Cifrado

Código	Título	Tipo	Carácter
RECU-0596	Criptología de empleo en el Esquema Nacional de Seguridad	Especificación	Recomendado

Confidencialidad | ENS | Integridad | LOPD | Seguridad

Almacenamiento de claves

- **Área:** Cifrado
- **Tipo de pauta:** Directriz
- **Carácter de la pauta:** Obligatoria

Código: LIBP-0256

Se deben tener en cuenta las siguientes recomendaciones para el almacenamiento de claves de forma segura.

La criptografía depende de las claves para asegurar la identidad del usuario, proporcionar confidencialidad e integridad, así como también no-repudio.

Es vital que las claves estén adecuadamente protegidas. Si una clave fuera comprometida, no se podría confiar en ella. Cualquier sistema que haya sido comprometido de alguna manera debería reemplazar todas sus claves de cifrado.

Se deben seguir las siguientes pautas para proteger de manera adecuada las claves:

Pautas

Título	Carácter
Frases de paso	Recomendada
Almacenamiento de claves en el código fuente	No Recomendada
Acceso restringido a las claves	Obligatoria
Información para descifrar los datos	Obligatoria
Almacenamiento de claves en base de datos	Obligatoria

Frases de paso

Almacenar en lugares físicos seguros las frases de paso.

Las frases de paso utilizadas para proteger las claves deberían ser almacenadas en lugares físicos seguros.

En algunos entornos podría ser necesario dividir la frase de paso o contraseña en dos componentes de tal manera que fuesen necesarias dos personas para autorizar el acceso a la clave. Este proceso físico / manual debería ser ligeramente monitorizado y controlado.

Volver al índice

Almacenamiento de claves en el código fuente

No almacenar claves dentro del código fuente o binarios.

Controlar que no se realice el almacenamiento de claves dentro del código fuente o binarios.

Esto no sólo tiene consecuencias si los desarrolladores tienen acceso al código fuente sino que, además, la administración de claves sería casi imposible.

Volver al índice

Acceso restringido a las claves

Restringir el acceso a los ficheros de configuración.

En la mayoría de los entornos web se utilizan ficheros de configuración para almacenar usuarios y claves de las aplicaciones. Debe restringirse el acceso a estos ficheros para evitar que otras aplicaciones, servidores o usuarios mal intencionados puedan acceder a los mismos.

Volver al índice

Información para descifrar los datos

Guardar la información necesaria para descifrar los datos, excepto la clave

En determinados casos, es necesario almacenar datos que serán obtenidos y desencriptados posteriormente desde el servidor web.

Para que esto pueda realizarse, debemos guardar la información necesaria para descifrar los datos (algoritmo, modo de cifrado, inicialización del vector), excepto la clave, junto con la información cifrada.

Volver al índice

Almacenamiento de claves en base de datos

Almacenar las claves cifradas en base de datos

Las contraseñas deben almacenarse siempre cifradas en base de datos, empleando un mecanismo de cifrado irreversible, como por ejemplo una función hash.

Volver al índice

Recursos

Área: Desarrollo » Seguridad » Cifrado

Código	Título	Tipo	Carácter
RECU-0581	Almacenamiento de las contraseñas en PHP	Ejemplo	Obligatorio
RECU-0584	Almacenar datos encriptados en una base de datos o un fichero en PHP	Ejemplo	Obligatorio
RECU-0669	Uso de claves en el código fuente	Ejemplo	No recomendado

Confidencialidad | ENS | LOPD | Seguridad

Encriptación de la información sensible

- **Área:** Cifrado
- **Tipo de pauta:** Directriz
- **Carácter de la pauta:** Obligatoria

Código: LIBP-0274

Encriptar inmediatamente la información sensible.

Es necesario encriptar inmediatamente la información sensible o crítica antes de su almacenamiento o transmisión ya que la falta de cifrado de los datos provoca que no puedan asegurarse las garantías de confidencialidad, integridad y rendición de cuentas que transmite una adecuada aplicación del cifrado. Para mitigar esta situación se tendrán en cuenta las siguientes indicaciones.

Pautas

Título	Carácter
Datos a cifrar	Obligatoria
Algoritmos criptográficos propios	No Recomendada
Límite de confianza para datos sensibles	Obligatoria
Encriptación de las contraseñas	Obligatoria
Encriptación de los identificadores de sesión	Obligatoria

Datos a cifrar

Especificar claramente los datos que deben cifrarse.

Debemos especificar claramente qué datos o recursos son suficientemente valiosos para ser protegidos mediante cifrado: identificaciones de usuario, contraseñas, números de tarjeta, etc.

Volver al índice

Algoritmos criptográficos propios

No desarrollar algoritmos criptográficos propios.

No se deben desarrollar algoritmos criptográficos propios, ya que es probable que estén expuestos a ataques bien conocidos por criptógrafos.

Además, el algoritmo puedes verse comprometido si los atacantes saben cómo funciona, haciéndolo especialmente débil.

Volver al índice

Límite de confianza para datos sensibles

Impedir que los datos sensibles salgan del límite de confianza.

Debemos compartimentar nuestro sistema para que tenga zonas "seguras" donde los límites de confianza puedan ser inequívocamente dibujados, de modo que se impida que los datos sensibles salgan de dicho límite de confianza. Además, debemos tener especial cuidado con la interfaz que maneja el compartimento fuera de la zona segura.

Volver al índice

Encriptación de las contraseñas

Encriptar las contraseñas con claves de, al menos, 128 bits

Una contraseña debe ser encriptada para que no esté al alcance de posibles ataques, si no, es posible que sea vulnerable.

Las contraseñas deben ser encriptadas con claves que tengan, al menos, 128 bits de longitud para asegurar de forma adecuada la privacidad de la misma.

Volver al índice

Encriptación de los identificadores de sesión

Concatenar la clave con otra información antes del cifrado

Es recomendable concatenar la clave con otra información, por ejemplo, el nombre del servidor, del usuario, etc. antes de procesarla.

Es bueno usar una función como **hash()** para almacenar un valor encriptado que no pueda ser revertido a la cadena original. No debe utilizarse hash() directamente en la clave, pues pueden escribirse programas que traten de adivinar esta información por ataques de fuerza bruta.

Volver al índice

Pautas

Área: Desarrollo » Seguridad » Gestión de Sesiones y Usuarios

Código	Título	Tipo	Carácter
LIBP-0311	Gestión de la sesión	Directriz	Obligatoria

Recursos

Área: Desarrollo » Seguridad » Cifrado

Código	Título	Tipo	Carácter
RECU-0574	Código vulnerable con contraseña sin encriptar	Ejemplo	Recomendado
RECU-0581	Almacenamiento de las contraseñas en PHP	Ejemplo	Obligatorio
RECU-0593	Encriptación de los identificadores de sesión en PHP	Ejemplo	Obligatorio
RECU-0570	Conexión sin encriptación de la información	Ejemplo	Obligatorio

Confidencialidad | ENS | Integridad | LOPD | Seguridad

Source URL: http://127.0.0.1/servicios/madeja/contenido/libro-pautas/274

Espacio de claves apropiado

- **Área:** Cifrado
- **Tipo de pauta:** Consejo

Código: PAUT-0235

Garantizar que el tamaño del espacio de claves es el adecuado

El espacio de claves es el rango de valores que puede tomar la clave criptográfica de un algoritmo de cifrado. El espacio de claves es propio de cada algoritmo de cifrado.

Un espacio de claves debe ser lo suficientemente grande (a mayor tamaño, mayor número de claves), como para prevenir ataques de fuerza bruta, teniendo en cuenta que a medida que la capacidad de cálculo y de ancho de banda se incrementen con el tiempo, harán estos tamaños insuficientes y por lo tanto más vulnerables.

Autenticidad | ENS | Seguridad

Generación de tokens seguros

- **Área:** Cifrado
- **Tipo de pauta:** Directriz
- **Carácter de la pauta:** Obligatoria

Código: PAUT-0202

Hacer uso de implementaciones existentes y probadas para la generación de números aleatorios para tokens seguros.

Los servidores Web actuales tratan con un gran número de usuarios. Normalmente se realiza un proceso de identificación sobre las cookies u otros identificadores de sesión para diferenciar entre todos ellos. Si estos identificadores de sesión usan una secuencia predecible, un atacante solo necesita generar un valor de la secuencia para presentar un token de sesión aparentemente válido. Esto puede ocurrir en gran número de sitios; a nivel de red para números de secuencia TCP, o bien a través de la capa de aplicación con las cookies usadas como tokens de autenticación.

La única manera de generar un token de autenticación seguro es asegurándose de que no hay manera de predecir su secuencia, en otras palabras: ***números realmente aleatorios***.

Se puede discutir que los ordenadores no pueden generar números realmente aleatorios, pero utilizando nuevas técnicas tales como la **lectura de los movimientos de ratón** y **pulsaciones de tecla** para mejorar la entropía han incrementado significativamente la aleatoriedad de los generadores de números aleatorios.

Dado el carácter crítico, es recomendable que no intente implementar esto de forma manual y hacer el uso de implementaciones existentes y probadas.

Recursos

Área: Desarrollo » Seguridad » Cifrado

Código	Título	Tipo	Carácter
RECU-0571	Guía de referencia sobre la arquitectura de criptografía en Java	Referencia	Recomendado
RECU-0607	Generar números aleatorios fuertes en Java	Ejemplo	Recomendado

Autenticidad | ENS | Seguridad

Source URL: http://127.0.0.1/servicios/madeja/contenido/pauta/202

Transmisiones de datos

- **Área:** Cifrado
- **Tipo de pauta:** Directriz
- **Carácter de la pauta:** Obligatoria

Código: PAUT-0200

Cifrar los datos cuando se vayan a transmitir a través de redes de telecomunicaciones.

Debemos cifrar lo más cerca posible del origen de datos para que el nivel de confianza que se tiene de la información sea mayor. Esto significa, aplicado a la transmisión de datos sensibles, que debemos asegurar que el cifrado ocurre antes de que transmitamos los datos a cualquier red no confiable.

Recursos

Área: Desarrollo » Seguridad » Cifrado

Código	Título	Tipo	Carácter
RECU-0583	Encriptar y desencriptar datos en PHP	Ejemplo	Obligatorio
RECU-0595	Protección de los datos a direfentes niveles	Referencia	Recomendado

ENS | Integridad | LOPD | Seguridad

Almacenamiento de las contraseñas en PHP

- **Área:** Cifrado
- **Carácter del recurso:** Obligatorio
- **Tecnologías:** PHP

Código: RECU-0581
Tipo de recurso: Ejemplo

Descripción

Este ejemplo muestra cómo alamcenar las claves en una base de datos utilizando el algoritmo SHA-512 en PHP.

Ejemplos

Es necesario tener bajo control las contraseñas de los usuarios para que puedan acceder a su sitio web.

Nunca se deben almacenar contraseñas en texto plano en una base de datos. En su lugar, almacene el hash de la contraseña, y haga uso de un salto para obtener mejores resultados:

```
<?php
/ $password contains the password. /

$salt = 'SHIFLETT';
$password_hash = hash('sha256', $salt . hash('sha256', $password . $salt));

/* Store password hash.*/
?>
```

Cuando se quiere determinar cuando un usuario ha introducido una contraseña correcta, hash provee la contraseña usando la misma técnica y comparando los hashes.

```
<?php

$salt = 'SHIFLETT';
$password_hash = hash('sha256', $salt . hash('sha256',$_POST['password'] . $salt));
/* Compare password hashes. */
?>
```

Si los valores hash son idénticos, entonces se asegura que las contraseñas son también idénticas. Usando esta técnica, no es posible recordar a los usuarios cuáles son sus contraseñas. Cuando un usuario se olvida de su contraseña, en lugar de dejar crear uno nuevo, almacena el hash de la contraseña nueva en la base de datos. Por supuesto, se debe ser muy cuidadoso para identificar al usuario correctamente ya que estos mecanismos son blanco frecuente de ataques y una fuente común de vulnerabilidades de seguridad.

Pautas

Área: Desarrollo » Seguridad » Cifrado

Código	Título	Tipo	Carácter
LIBP-0256	Almacenamiento de claves	Directriz	Obligatoria
LIBP-0274	Encriptación de la información sensible	Directriz	Obligatoria

ENS | LOPD

Source URL: http://127.0.0.1/servicios/madeja/contenido/recurso/581

Almacenar datos encriptados en una base de datos o un fichero en PHP

- **Área:** Cifrado
- **Carácter del recurso:** Obligatorio
- **Tecnologías:** PHP

Código: RECU-0584
Tipo de recurso: Ejemplo

Descripción

El ejemplo muestra un fragmento de código para desencriptar datos cifrados, recuperando la clave del usuario y usando los datos almacenados para descifrarlos.

Ejemplos

```
<?php

/* Encrypt the data. */
$algorithm = MCRYPT_BLOWFISH;
$mode = MCRYPT_MODE_CBC;
$iv = mcrypt_create_iv(mcrypt_get_iv_size($algorithm, $mode), MCRYPT_DEV_URANDOM);
$ciphertext = mcrypt_encrypt($algorithm, $_POST['key'], $_POST['data'], $mode, $iv);

/* Store the encrypted data. */
$st = $db->prepare('INSERT
            INTO noc_list (algorithm, mode, iv, data)
            VALUES (?, ?, ?, ?)');
$st->execute(array($algorithm, $mode, $iv, $ciphertext));

?>
```

Para desencriptar los datos, se recupera la clave del usuario y la usa con los datos almacenados

```
<?php
$row = $db->query('SELECT *
                   FROM noc_list
                   WHERE id = 27')->fetch();
$plaintext = mcrypt_decrypt($row->algorithm,
                            $_POST['key'],
                            $row['data'],
                            $row['mode'],
                            $row['iv']);
?>
```

Pautas

Área: Desarrollo » Seguridad » Cifrado

Código	Título	Tipo	Carácter
LIBP-0256	Almacenamiento de claves	Directriz	Obligatoria

ENS | LOPD

Source URL: http://127.0.0.1/servicios/madeja/contenido/recurso/584

Código vulnerable con contraseña sin encriptar

- **Área:** Cifrado
- **Carácter del recurso:** Recomendado
- **Tecnologías:** Java

Código: RECU-0574
Tipo de recurso: Ejemplo

Descripción

Este es un ejemplo de código vulnerable que se ejecutará con éxito aunque cualquier persona con acceso a **config.properties** podría leer el valor de la contraseña y determinar fácilmente que el valor se ha codificado de base 64.

Si un atacante consigue acceder a esta información, podría utilizarla para irrumpir en el sistema.

Ejemplos

```
…
Properties prop = new Properties();
prop.load(new FileInputStream("config.properties"));
String password = Base64.decode(prop.getProperty("password"));
DriverManager.getConnection(url, usr, password);
…
```

Pautas

Área: Desarrollo » Seguridad » Cifrado

Código	Título	Tipo	Carácter
LIBP-0274	Encriptación de la información sensible	Directriz	Obligatoria

ENS | LOPD

Source URL: http://127.0.0.1/servicios/madeja/contenido/recurso/574

Conexión sin encriptación de la información

- **Área:** Cifrado
- **Carácter del recurso:** Obligatorio
- **Tecnologías:** Java

Código: RECU-0570
Tipo de recurso: Ejemplo

Descripción

En este ejemplo se establece una conexión con una URL, para comunicar información sensible, sin encriptar dicha conexión. Aunque la conexión se realiza correctamente, es posible que todos los datos sensibles enviados o recibidos desde el servidor sean leídos por actores no deseados.

Ejemplos

```
try {
    URL u = new URL("http://www.secret.example.org/");
    HttpURLConnection hu = (HttpURLConnection) u.openConnection();
    hu.setRequestMethod("PUT");
    hu.connect();
    OutputStream os = hu.getOutputStream();
    hu.disconnect();
} catch (IOException e) {
    //...
}
```

Pautas

Área: Desarrollo » Seguridad » Cifrado

Código	Título	Tipo	Carácter
LIBP-0274	Encriptación de la información sensible	Directriz	Obligatoria

Área: Desarrollo » Seguridad » Control de Acceso y Autenticación

Código	Título	Tipo	Carácter
LIBP-0269	Canal de comunicación	Directriz	Obligatoria

ENS | LOPD

Source URL: http://127.0.0.1/servicios/madeja/contenido/recurso/570

Criptología de empleo en el Esquema Nacional de Seguridad

- **Área:** Cifrado
- **Carácter del recurso:** Recomendado

Código: RECU-0596
Tipo de recurso: Especificación

Introducción

Es especifican en este recurso el conjunto de técnicas y algoritmos de cifrado de la información contemplados en el Esquema Nacional de Seguridad

Enlace oficial

Guía Criptología

Pautas

Área: Desarrollo » Seguridad » Cifrado

Código	Título	Tipo	Carácter
LIBP-0270	Algoritmos de cifrado	Directriz	Obligatoria

ENS | LOPD

Source URL: http://127.0.0.1/servicios/madeja/contenido/recurso/596

Encriptación de los identificadores de sesión en PHP

- **Área:** Cifrado
- **Carácter del recurso:** Obligatorio

Código: RECU-0593
Tipo de recurso: Ejemplo

Introducción

No se deben almacenar las claves de identificación de sesión como texto plano, es inseguro aunque se almacenen en el servidor.

Es bueno usar una función como **hash()** para almacenar un valor encriptado que no pueda ser revertido a la cadena original. No debe utilizarse hash() directamente en la clave, pues pueden escribirse programas que traten de adivinar esta información por ataques de fuerza bruta.

Es recomendable concatenar la clave con otra información, por ejemplo, el nombre del servidor, del usuario, etc. antes de procesarla.

Descripción

A continuación puede consultar un ejemplo de encriptación basado en SHA256:

Ejemplos

```
?php
/*
 * Transparent SHA-256 Implementation for PHP 4 and PHP 5
 *
 * Author: Perry McGee (pmcgee@nanolink.ca)
 * Website: http://www.nanolink.ca/pub/sha256
 *
 * Copyright (C) 2006,2007,2008,2009 Nanolink Solutions
 *
 * Created: Feb 11, 2006
 *
 *    This library is free software; you can redistribute it and/or
 *    modify it under the terms of the GNU Lesser General Public
 *    License as published by the Free Software Foundation; either
 *    version 2.1 of the License, or (at your option) any later version.
 *
 *    This library is distributed in the hope that it will be useful,
 *    but WITHOUT ANY WARRANTY; without even the implied warranty of
 *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 *    Lesser General Public License for more details.

 *    You should have received a copy of the GNU Lesser General Public
 *    License along with this library; if not, write to the Free Software
 *    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 *    or see <http://www.gnu.org/licenses/>.
 *
 *  Include:
```

Pautas

Área: Desarrollo » Seguridad » Cifrado

Código	Título	Tipo	Carácter
LIBP-0274	Encriptación de la información sensible	Directriz	Obligatoria

ENS | Seguridad

Encriptar y desencriptar datos en PHP

- **Área:** Cifrado
- **Carácter del recurso:** Obligatorio
- **Tecnologías:** PHP

Código: RECU-0583
Tipo de recurso: Ejemplo

Descripción

Este ejemplo nos enseña cómo podemos encriptar y desencriptar datos en PHP.

Ejemplos

Para el manejo de la encriptación de la información es recomendable utilizar la extensión de PHP es **mcrypt**, que es compatible con un número de diferentes algoritmos criptográficos. Para ver cuáles son compatibles con su plataforma, utiliza la función mcrypt_list_algorithms ():

```
<? Php
echo '<pre>. print_r (mcrypt_list_algorithms (), TRUE). '</ Pre> ";
?>
```

El encriptado y desencriptado de datos se realiza mediante el uso de las funciones mcrypt_encrypt( ) y mcrypt_decrypt( ), respectivamente. Estas funciones aceptan cinco argumentos, el primero de ellos es acerca del algoritmo a usar, el segundo es la clave . El tercer argumento incluye los datos para encriptar o desencriptar en función de la funcionalidad implementada. El cuarto argumento indica el modo de encriptado o desencriptado y finalmente el quinto argumento esta dedicado al vector de inicializacion

```
<?php

mcrypt_encrypt($algorithm, $key, $cleartext, $mode, $iv);

mcrypt_decrypt($algorithm, $key, $ciphertext, $mode, $iv);
?>
```

un ejemplo real sería el siguiente:

```
<?php

$algorithm = MCRYPT_BLOWFISH;
$key = 'That golden key that opens the palace of eternity.';
$data = 'The chicken escapes at dawn. Send help with Mr. Blue.';
$mode = MCRYPT_MODE_CBC;

$iv = mcrypt_create_iv(mcrypt_get_iv_size($algorithm, $mode),
                       MCRYPT_DEV_URANDOM);

$encrypted_data = mcrypt_encrypt($algorithm, $key, $data, $mode, $iv);
$plain_text = base64_encode($encrypted_data);
echo $plain_text . "\n";

$encrypted_data = base64_decode($plain_text);
$decoded = mcrypt_decrypt($algorithm, $key, $encrypted_data, $mode, $iv);
echo $decoded . "\n";
?>
```

y su salida sería:

```
NNB9WnuCYjyd3Y7vUh7XDfWFCWnQY0BsMehHNmBHbGOdJ3cM+yghABb/XyrJ+w3xz9tms74/a70=
The chicken escapes at dawn. Send help with Mr. Blue.
```

Pautas

Área: Desarrollo » Seguridad » Cifrado

Código	Título	Tipo	Carácter
PAUT-0200	Transmisiones de datos	Directriz	Obligatoria

ENS | LOPD

Source URL: http://127.0.0.1/servicios/madeja/contenido/recurso/583

Generar números aleatorios fuertes en Java

- **Área:** Cifrado
- **Carácter del recurso:** Recomendado

Código: RECU-0607
Tipo de recurso: Ejemplo

Introducción

Los generadores de números pseudoaleatorios (PRNGs) se basan en el uso de algoritmos matemáticos deterministas para producir una secuencia de números con buenas propiedades estadísticas, pero los números presentados no son genuinamente al azar.

PRNG suele comenzar con un valor de inicialización aritmética. El algoritmo usa esta semilla para generar un valor de salida y una nueva semilla, que se utiliza para generar el valor siguiente, y así sucesivamente.

Descripción

La API de Java proporciona un PRNG, la clase **java.util.Random**. Este PRNG es portátil y repetible por lo que, si dos instancias de azar se crean utilizando la misma semilla, se pueden generan secuencias idénticas de números en todas las implementaciones de Java.

A veces la misma semilla se reutiliza en la inicialización de la aplicación o después de cada reinicio del sistema. En otras ocasiones, la hora actual obtenida del reloj del sistema es utilizada para obtener la semilla. Un adversario puede aprender el valor de la semilla mediante la realización de algunos reconocimientos sobre el blanco vulnerable y proceder a construir una tabla de consulta para la estimación de valores futuros de la semilla.

Ejemplos

Un ejemplo de código vulnerable es el siguiente: si se utiliza la misma semilla se obtiene la misma secuencia de números como resultado, las cifras no son "al azar". Este ejemplo de código utiliza la clase insegura **java.util.Random**.

```
import java.util.Random;
// …
Random number = new Random(123L);
//…
for (int i = 0; i < 20; i++) {
  // Generate another random integer in the range [0, 20]
  int n = number.nextInt(21);
  System.out.println(n);
}
```

Es recomendable utilizar la clase **java.security.SecureRandom** que produce números aleatorios de calidad, como en el ejemplo siguiente:

```
import java.security.SecureRandom;
import java.security.NoSuchAlgorithmException;
// …
public static void main (String args[]) {
   try {
     SecureRandom number = SecureRandom.getInstance("SHA1PRNG");
     // Generate 20 integers 0..20
     for (int i = 0; i < 20; i++) {
       System.out.println(number.nextInt(21));
     }
   } catch (NoSuchAlgorithmException nsae) {
     // Forward to handler
   }
}
```

Pautas

Código	Título	Tipo	Carácter
PAUT-0202	Generación de tokens seguros	Directriz	Obligatoria

ENS | Seguridad

Source URL: http://127.0.0.1/servicios/madeja/contenido/recurso/607

Guía de referencia sobre la arquitectura de criptografía en Java

- **Área:** Cifrado
- **Carácter del recurso:** Recomendado
- **Tecnologías:** Java

Código: RECU-0571
Tipo de recurso: Referencia

Descripción

Enlace a la guía de referencia de la arquitectura criptográfica en Java.

Enlaces externos

- Guía de referencia de JCA (en inglés)

Pautas

Área: Desarrollo » Seguridad » Cifrado

Código	Título	Tipo	Carácter
PAUT-0202	Generación de tokens seguros	Directriz	Obligatoria

ENS | LOPD

Introducción al cifrado

- **Área:** Cifrado
- **Carácter del recurso:** Recomendado

Código: RECU-0554
Tipo de recurso: Referencia

Descripción

Tipos de cifrado

Se pueden realizar al menos tres tipos de cifrado:

Cifrado simétrico

El emisor cifra el mensaje con una clave, y esa misma clave deberá ser la utilizada para descifrarlo. Estos algoritmos son rápidos y permiten cifrar y descifrar eficientemente con claves relativamente grandes. El problema que tienen es la seguridad de la clave: El emisor cifra el mensaje con la clave. Manda el mensaje cifrado, de forma que nadie puede descifrarlo sin esa clave. Ahora bien, el receptor ha de conocer la clave, ¿cómo se transmite sin comprometer su seguridad? Algunos algoritmos de este tipo son:

- Blowfish
- IDEA
- DES

Cifrado asimétrico

Existen dos claves, una pública y una privada, y se puede usar en dos direcciones.

a) El emisor cifra el mensaje con la clave pública A, que es la que puede conocer cualquiera. Sin embargo para descifrarlo hace falta la clave B, que sólo tiene el receptor, ya que es privada. Con esto se garantiza confidencialidad, cualquiera podría cifrar, pero sólo quien tenga la clave privada podrá descifrar.

b) El emisor cifra el mensaje con la clave privada B, que sólo él conoce. Ahora cualquiera puede descifrarlo con la clave privada A, pero una vez descifrado con esa clave A, la naturaleza del algoritmo estará garantizando que se ha cifrado con la clave B, por lo que la utilización del algoritmo en este sentido se usa para asegurar la autenticidad, y no para ocultar información. Cualquiera tendrá acceso a la información, pero podrá saber a ciencia cierta de dónde procede. El cifrado asimétrico en esta dirección se usa para certificar la autenticidad de las firmas digitales.

El funcionamiento de estos algoritmos, basados en factorización de números primos, permite que el cifrado se calcule con relativa sencillez, pero haga falta más procesamiento para descifrarlo. No obstante, aunque este algoritmo garantiza la seguridad de la clave privada, ya que sólo la tiene el receptor, es más lento y hace que los mensajes cifrados tengan un volumen mayor. Ejemplos de este cifrado son:

- DSA
- RSA
- Diffie-Hellman

Cifrado híbrido

A menudo las conexiones seguras de Internet se sirven de una mezcla de los dos tipos de cifrado anteriores. Aprovechan la ligereza de uno y la fortaleza del otro. Lo que suelen hacer protocolos de comunicación seguros como HTTPS (basado en la capa SSL), es cifrar los mensajes usando un algoritmo simétrico, de forma que se hace un cifrado y descifrado rápido de los mismos, además de tener menos volumen los mensajes cifrados. Como hay que transmitir la clave del cifrado de alguna manera, ésta se cifra con un algoritmo asimétrico. Con esto se consigue que la parte más voluminosa de la información, que es el mensaje, vaya cifrada con un algoritmo seguro pero ligero, y la parte menos voluminosa, que es la clave, vaya cifrada con el algoritmo más pesado, y que garantiza que sólo podrá ser descifrada en el destino.

De esta manera, en el destino primero ha de descifrar la clave del cifrado simétrico, con su clave privada, y una vez tenida esta clave, descifrar el mensaje. Esto garantiza una conexión segura.

Aplicaciones del cifrado

El cifrado de datos se usa en numerosas aplicaciones cotidianas. Algunas de las más habituales son:

SSL

SSL, del inglés Secure Socket Layer, es un protocolo criptográfico que proporciona comunicaciones seguras en Internet. Esta seguridad es en forma de privacidad y autenticación: por un lado autentica el servidor de la comunicación (mediante certificados), y por otra parte selecciona un algoritmo de cifrado, permite el intercambio de claves de forma segura entre

cliente y servidor, y cifra la información con cifrado simétrico.

Esta capa de seguridad se puede aplicar en diversos ámbitos:

- HTTPS: Protocolo http seguro
- FTP: protocolo de intercambio de ficheros. Puede utilizar SSL para ser seguro.
- SMTP: protocolo de correo. Puede utilizar SSL para ser seguro.

Firma digital

La firma digital es el método criptográfico que permite asociar la identidad de una persona o máquina a un documento como autor del mismo. Para incorporar las firmas digitales, primero se calculan los datos de la firma, que se obtienen de aplicar cierto algoritmo matemático. Esos datos se cifran con alguno de los algoritmos descritos anteriormente y finalmente la firma cifrada es incorporada al documento. El receptor del documento deberá tener medios tecnológicos tanto para extraer la firma cifrada como para descifrarla, por lo que también deberá tener la clave.

VPN

La red privada virtual, en inglés Virtual Private Network (VPN), es una red con las características de una LAN, pero está extendida sobre una red pública como Internet; esto es, tiene el control y la seguridad que ofrece una red LAN pero topológicamente tiene un ámbito descontrolado e inseguro como es Internet. Para que estas redes sean seguras se usan técnicas de tunneling que consisten en crear un "túnel" seguro dentro de la red insegura, por el que circulan los datos de la VPN cifrados. Es por esto que las redes privadas virtuales es uno de los usos más frecuentes de cifrado.

Para garantizar la seguridad de la red VPN y las características que debe cumplir, se usan protocolos de comunicación segura como IPSec, que es el estándar de facto, aunque también se usan otros como SSL o PPTP.

Cifrado de archivos

También existen aplicaciones que permiten el cifrado, no ya de una información que se va a enviar, si no de un archivo, que puede que se vaya a enviar, pero puede que simplemente quiera guardarse cifrado para que sólo puedan leerlo quienes tengan una clave. Esto también es útil para almacenar información confidencial de una organización. En caso de sustracción de la información no servirá de nada si no se tiene una clave para descifrarla.

Cifrado de disco duro

Tener todo el sistema de archivos cifrado permite que cada vez que se guarde un archivo ya lo haga cifrado por defecto y que todo lo contenido en el disco duro esté cifrado. Esto hace que haya procesos ligeramente más lentos, ya que cada vez que se guarda, por ejemplo ha de cifrarlo.

Mal uso del cifrado

Tener un fichero o una información cifrada no garantiza absolutamente su integridad o su fiabilidad.

Incluso con los algoritmos más seguros, se pueden presentar una serie de problemas:

- Tener un fichero cifrado en un disco duro, y que el disco duro se estropee, por lo que la información se pierda.
- La persona que sabe la clave, la olvida, o bien por deslealtad a la compañía, la filtra.
- La clave se almacena el mismo sitio que el fichero cifrado, por lo que si alguien accede al fichero cifrado también podrá acceder a la clave para descifrarlo.
- La información cifrada está corrupta o no es válida, por lo que aunque el cifrado y descifrado sean correctos, la información obtenida por el destinatario seguirá corrupta o inválida.
- Cualquier otro problema derivado de una mala gestión de la información cifrada o las claves.

Para evitar estos problemas hay que seguir buenas prácticas como tener backups de la información, o garantizar la seguridad de las claves.

Pautas

Área: Desarrollo » Seguridad » Servicios Web

Código	Título	Tipo	Carácter
LIBP-0308	Confidencialidad e Integridad	Directriz	Recomendada

ENS | LOPD

Protección de los datos a direfentes niveles

- **Área:** Cifrado
- **Carácter del recurso:** Recomendado

Código: RECU-0595
Tipo de recurso: Referencia

Descripción

A la hora de transmitir los datos tenemos la posibilidad de cifrarlos o, en caso de no ser posible, protegerlos a diferentes niveles. La elección del sitio correcto puede conllevar requisitos de recursos y seguridad.

Características

Los distintos niveles donde se puede aplicar la protección de los datos son los siguientes:

- Aplicación: a este nivel, la aplicación actual realiza el cifrado u otra función de criptográfica. Esto es lo más deseable, pero puede dar lugar a más presión adicional en recursos y crear una complejidad inmanejable. El cifrado debería realizarse típicamente a través de una API tal como el kit de herramientas OpenSSL (www.openssl.com) o a través de funciones de cifrado proporcionadas por el sistema operativo.
- Protocolo: en esta capa, el protocolo proporciona el servicio de cifrado. Más comúnmente, esto se ve con HTTPS, usando cifrado SSL para proteger tráfico Web sensible. La aplicación ya no necesita implementar conectividad segura. Sin embargo, esto no significa que la aplicación se libere de ello. SSL requiere una atención especial cuando se usa en autenticación mutua (en la parte cliente), existen dos claves de sesión diferentes, una para cada dirección. Debería verificarse cada una antes de transmitir datos sensibles. Los atacantes pueden beneficiarse del uso de SSL para ocultar peticiones maliciosas (ataques de inyección por ejemplo).
- Red: por debajo de la capa de protocolo, podemos usar tecnologías tales como Redes Privadas Virtuales (VPN) para proteger los datos. Esto tiene muchas posibilidades, siendo la más popular IPsec (Seguridad del Protocolo de Internet v6), típicamente implementado como un 'túnel' protegido entre dos routers gateway. Ni la aplicación ni el protocolo necesita disponer de cifrado , todo el tráfico es interceptado independientemente.

Pautas

Área: Desarrollo » Seguridad » Cifrado

Código	Título	Tipo	Carácter
PAUT-0200	Transmisiones de datos	Directriz	Obligatoria

ENS | LOPD

Uso de claves en el código fuente

- **Área:** Cifrado
- **Carácter del recurso:** No recomendado
- **Tecnologías:** Java

Código: RECU-0669
Tipo de recurso: Ejemplo

Descripción

El uso de una clave en el código de cifrado aumenta significativamente la posibilidad de que los datos cifrados pueden ser reclamados. Si se usan claves criptográficas no modificables, es casi seguro que los usuarios malintencionados podrán acceder a través de la cuenta en cuestión.

Un ejemplo de mal código sería el siguiente: se intenta comprobar una contraseña usando una clave en el código criptográfico. La clave de cifrado se encuentra dentro de un valor de cadena en el código que se compara con la contraseña y se devuelve un valor verdadero o falso para la comprobar si la contraseña es equivalente a la clave criptográfica no modificable

Ejemplos

```
public boolean VerifyAdmin(String password) {
   if (password.equals("68af404b513073584c4b6f22b6c63e6b")) {
      log.info("Entering Diagnostic Mode...");
      return true;
   }
   log.error("Incorrect Password!");
   return false;
}
```

Pautas

Área: Desarrollo » Seguridad » Cifrado

Código	Título	Tipo	Carácter
LIBP-0256	Almacenamiento de claves	Directriz	Obligatoria

Seguridad

Source URL: http://127.0.0.1/servicios/madeja/contenido/recurso/669