reset-UTS

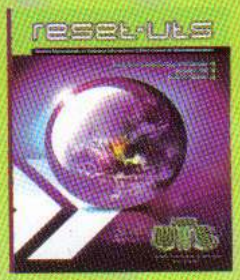
ISSN 1909-258X

Revista Especializada en Sistemas Informáticos y Electrónicos de Telecomunicaciones

Revista de las Unidades Tecnológicas de Santander
Diciembre - 2008
Número 3 - Volumen I

UTS

Unidades Tecnológicas de Santander
Bucaramanga



RESET-UTS. Número 3. Volumen 1. Diciembre de 2008
Unidades Tecnológicas de Santander
Coordinación General de Investigaciones
Centro de Investigaciones en Ciencias Naturales e Ingenierías
Bucaramanga, Colombia, 2008

Apoyo Institucional U.T.S.
Rector: Victor Raúl Castro Neira
Vicerrector: Alfredo Reyes Serpa
Decano Facultad de Ciencias Naturales e Ingenierías: Rafael Osorio Thomas
Coordinador General de Investigaciones: Fabio Alfonso González
Coordinador Centro de Investigaciones en Ciencias Naturales e Ingenierías: Carlos Andrés Guerrero Alarcón

Grupo Editorial RESET-UTS
Director: Emil Javier Guerra Monterroza
Coordinador Académico: Erwin John Saavedra Mercado
Coordinador Editorial: Jorge Gerardo Concha Sánchez

Comité Editorial
Juan David Gutiérrez Torres
Ever Jesús Blanco Meza
Glenn Elmer Hernández Camelo

Comité Científico

Universidad Pontificia Bolivariana, Colombia
Omar Pinzón
opinzon@upbbga.edu.co

Universidad Pedagógica y Tecnológica, Colombia
Julio Enrique Duarte
jduarte@duitama.uptc.edu.co

Universidad Pontificia Bolivariana, Colombia
Cristina Gómez Santamaría
cristina.gomez@upb.edu.co

Universidad Federal del Rio de Janeiro, Brasil
Omar Lengerke Pérez
olengerke@ufrj.br

Universidad Pedagógica y Tecnológica de Colombia
Flavio Fernández
flavio@duitama.uptc.edu.co

Gestión de la Publicación
Marco Fidel Flórez Franco
Antonio Alexi Anteliz Jaimes

Corrección ortográfica y de estilo
María Carolina Jáuregui Paz
Fabio Alfonso González

Diseño de Impresión
Impregráficas

ISSN 1909-258X
RESET-UTS Es una publicación de la Coordinación General de Investigaciones.
Unidades Tecnológicas de Santander.
Calle de los Estudiantes No 9-82
Ciudadela Real de Minas
Teléfono: 6413000 Ext.: 214-220
Bucaramanga- Colombia
Web: http://www.uts.edu.co/reset/
E-mail:reset@uts.edu.co - uts@uts.edu.co

Contenido

MODELO DE SEGURIDAD PARA APLICACIONES WEB QUE UTILIZAN CÓDIGO ABIERTO EN SU IMPLEMENTACIÓN.

Carlos Andrés Guerrero Alarcón

Ingeniero de Sistemas
Magíster en Informática
Grupo de Investigación en Ingeniería del Software GRIIS
Unidades Tecnológicas de Santander
anguerrco@msn.com
Colombia

Luz Elena Gutiérrez López

Ingeniera de Sistemas
Magíster (c) en Ingeniería con énfasis en informática.
Grupo de Investigación en Ingeniería del Software GRIIS
Unidades Tecnológicas de Santander
luzelenagl@yahoo.com
Colombia

RESUMEN. Este artículo presenta parte de los resultados de la investigación denominada "caracterización de sistemas web soportados en arquitecturas de capas que utilizan código abierto para su implementación". Fundamentalmente esta investigación de tipo descriptiva tuvo por objeto identificar las funcionalidades más relevantes que debe tener un sistema de información web desarrollado bajo el paradigma de código abierto. Todo esto, para brindar a los estudiantes y profesionales del área de sistemas instrumento útil y de fácil acceso para la construcción de los desarrollos informáticos a nivel profesional. Uno de los criterios funcionales identificados y definidos en la caracterización fue el aspecto relacionado con la seguridad informática, del cual se obtuvo como resultado en esta investigación un modelo criptográfico que sirve como patrón para ser reutilizado por los estudiantes en sus proyectos de grado.

En el presente artículo se describen los modelos actuales más conocidos en cuanto a la seguridad de aplicaciones Web, dando una breve explicación de las ventajas y desventajas de los mismos. Luego se presenta la construcción del modelo propuesto, así como la forma como se debe instanciar en una aplicación Web de código abierto, ilustrando de forma clara las ventajas de este modelo respecto a los más populares. Por último, este artículo presenta las conclusiones obtenidas del trabajo de investigación en lo relacionado a la parte de seguridad informática, toda vez que en la investigación se analizaron éste y otros criterios funcionales que no son descritos en el presente artículo.

Palabras claves. Sistemas web, seguridad computacional, criptografía, programación orientada a objetos.

ABSTRACT. This article presents part of the results of the denominated investigation "Characterization of supported Web system's in architectures of layers that use open code for their implementation". Fundamentally this investigation of descriptive and exploratory type intended to identify the most important functionalities that must have a web information system developed under the paradigm of open code. All this, to offer to the students and professionals of the system area useful instrument and readily accessible for the construction of the computer science developments to professional level. One of the identified and defined functional criteria in the characterization was the aspect related to the computer science security, from which a cryptographic model was obtained as a result in this investigation which serves as a pattern to be reused by the students in theirs projects.

First of all, the document presents the opportunity found at "Technological Units of Santander" for the accomplishment of this investigation and the methodology used to analyze the most important security aspects for a Web Application, which begins with the protocol for the hypertext transport, continuing with the handling of sessions at level of archives client and identification labels, following with the handling of public and deprived keys, and finally defining the cryptographic model that was developed in this investigation. Finally, this article presents the conclusions obtained from the investigation work related to computer science security, every time in the investigation they analyzed this one and other functional criteria that are not described in the present article.

Keywords: Web System's, Computer Security, Cryptography Object-oriented Programming

INTRODUCCIÓN

Para construir un sistema informático es necesario conceptualizar el problema, especificar requisitos, plantear una posible arquitectura, realizar especificaciones de diseño, para posteriormente dar inicio a un proceso de desarrollo, que de acuerdo a un modelo de ciclo de vida establecido, permita desarrollar el sistema con tiempo y costos adecuados [2]. Aunque lo anterior es una premisa fundamental en cualquier desarrollo software, no es un estándar para el desarrollo de proyectos de pregrado, en donde muchas veces lo más importante es la codificación, restándole importancia a los procesos de planificación y diseño del proyecto.

La fundamentación teórica del proceso de desarrollo software es conocido por gran parte de los estudiantes de sistemas que optan por desarrollar un proyecto de grado, sin embargo la aplicación de todos estos conceptos se vuelve compleja, por cuanto los factores tiempo y costo apremian a la hora de realizar el proyecto. Es entonces, donde la búsqueda de información y marcos de referencia se torna ineficaz, por cuanto existen pocos trabajos que le ayuden a los estudiantes a clasificar y a conceptualizar lo que realmente requiere un sistema informático desde el punto de vista funcional y no funcional.

Por otra parte, el mercado global se ha tornado más exigente en cuanto a los requisitos que un profesional en el área de sistemas debe tener. La adopción de estándares internacionales como ISO 9001:2000 [10] y el modelo de madurez de la capacidad -CMMi- [3] en empresas de desarrollo software son una realidad en Colombia y en el mundo, por lo tanto, es una responsabilidad de las instituciones de educación superior preparar y capacitar a los futuros profesionales.

A nivel técnico y tecnológico los estudiantes tienen diversas fortalezas, dominan y manejan herramientas de desarrollo con fluidez, pero ¿Qué es lo que en realidad construyen o desarrollan?, ¿Son acaso soluciones informáticas que responden a las verdaderas necesidades de un posible cliente, o son el resultado de un trabajo empírico?. A diferencia de la parte técnica, a nivel conceptual existen vacíos que les impiden a los futuros profesionales ingresar al nuevo tipo de empresas de desarrollo que tiene el país, en donde el aseguramiento de la calidad es una exigencia. Por ende, brindar a los estudiantes herramientas informáticas para la solución de problemas es un primer paso, sin embargo dotarlos de conocimientos y modelos con los cuales puedan tomar decisiones acertadas es aun más importante.

Lo anterior plantea una oportunidad para apoyar en el proceso de formación a los estudiantes del área de sistemas. En este trabajo de investigación se realizó una caracterización por funcionalidades de los sistemas Web, encontrándose que uno de los aspectos más relevantes a la hora de realizar una clasificación es la seguridad informática.

Este artículo presenta los resultados de parte de la caracterización realizada, y se enfoca en el planteamiento de un modelo criptográfico para que los estudiantes o personas que desarrollan proyectos software bajo el paradigma de código abierto, lo puedan utilizar.

1. MODELOS ACTUALES DE SEGURIDAD EN APLICACIONES WEB

El desarrollo de aplicaciones en los últimos años ha tenido una ligera inclinación hacia los sistemas que tienen orientación hacia la Web, la necesidad de información en línea y en general los continuos cambios tecnológicos y sociales han llevado a los desarrolladores a tener modelos de seguridad cada vez mejores y más sofisticados, buscando siempre la verificación óptima y adecuada de la identidad de un usuario. A continuación se presentan las estrategias utilizadas y los modelos de seguridad que tienen más popularidad en los desarrollos actuales.

1.1 VERIFICACIÓN POR UNIDAD DE CÓDIGO

Una unidad de código es la mínima expresión que puede llegar a tener un sistema a la hora de cuantificar su tamaño (clases, objetos, métodos ó líneas de código), es independiente del tipo de producto a desarrollar y tiene una relación directa con el tipo de métrica que se va a implantar para realizar el seguimiento de los desarrolladores. Este modelo consiste en verificar a nivel de "unidades mínimas de código", si un usuario final puede acceder o no a la funcionalidad requerida. Adicionalmente tiene un nivel de complejidad alto y su rendimiento es bajo, por cuanto si la unidad seleccionada es la menor se deberían realizar verificaciones para cada una de ellas.

Si un objeto ó unidad es utilizado "n" cantidad de veces en una petición Web, entonces será verificada su posible utilización "n" cantidad de veces. En promedio, un sistema Web de complejidad alta puede llegar a tener por petición Web entre mil y dos mil solicitudes, lo cual implicaría una verificación excesiva. A nivel comercial y a gran escala este modelo es poco utilizado pues su núcleo principal se encuentra en el código fuente como tal y no permite hacer una separación por capas entre la seguridad y las reglas del negocio del aplicativo desarrollado.

Entre las ventajas de usar este modelo están:

No.	Ventaja
1	Uso restringido y exclusivo de una unidad de código, pues solo podrá ser utilizada por usuarios con permisos asignados.
2	Útil para desarrollos menores, en donde su modelo relacional sea menor a doce (12) tablas.

Tomado de: Autores

Tabla 1. Ventajas Unidad de código

Entre las desventajas de este modelo están.

No.	Desventaja
1	Poco útil para desarrollos comerciales o a gran escala, pues su complejidad puede llegar a ser inmanejable.
2	El procesamiento en tiempo real de algunos tipos de formatos y documentos constituye su principal debilidad pues el rendimiento es mínimo.

Tomado de: Autores

Tabla 2. Desventajas Unidad de código

Este modelo es utilizado en la actualidad en proyectos de investigación aplicada, particularmente aquellos que trabajan la temática de elicitación y gestión de requisitos, pues permite relacionar requisitos funcionales y posibles actores del sistema con la estructura lógica del código a desarrollar.

1.2 VERIFICACIÓN POR PÁGINAS

La verificación por páginas si permite separar la lógica del negocio de la seguridad del aplicativo, lo cual permite dividir el sistema en capas, siendo su implementación y mantenimiento menos complejo. Este modelo consiste en verificar página por página si un usuario que accede al sistema tiene o no los permisos necesarios para acceder a una funcionalidad. En primera instancia el usuario que desea ingresar al sistema se valida contra una base de datos y una vez identificado se verifica en todas las solicitudes que realice su autenticidad.

Entre las ventajas de usar este modelo están:

No.	Ventaja
1	Buen rendimiento para aplicaciones a gran escala.
2	Control adecuado de los recursos y objetos a utilizar en el sistema.
3	Control y acceso restringido a las funcionalidades de un usuario.

Tomado de: Autores

Tabla 3. Ventajas Verificación páginas

Entre las desventajas de este modelo están:

No.	Desventaja
1	Verificación innecesaria cuando el usuario decide manipular las direcciones Web de manera manual.
2	Si el usuario conoce de antemano los parámetros de una página puede acceder a ella sin restricciones.

Tomado de: Autores

Tabla 4. Desventajas Verificación páginas

La principal debilidad de este modelo radica en el poco control que se tiene sobre el usuario final, por cuanto un usuario con conocimientos básicos de programación Web podría tratar de ingresar de manera ilegal al sistema. Este problema es corregido en este modelo con la aplicación de una verificación en segunda instancia, es decir, que luego de verificar si el usuario tiene permisos para la página, se verifica que los objetos solicitados sean correctos. Aunque existe un doble procesamiento, el rendimiento no disminuye radicalmente y por ende este modelo es muy utilizado en los desarrollos actuales.

Entre las ventajas de usar este modelo están:

No.	Ventaja
1	Aplicaciones comerciales y a gran escala hacen uso de este modelo por su rendimiento y simplicidad.
2	El proceso de verificación se minimiza, toda vez que la información inicializada en una sesión no es manipulada como parámetro en el resto del aplicativo.
3	Encapsulamiento de la información relacionada con la verificación del usuario.

Tomado de: Autores

Tabla 5. Ventajas sobrecarga

Entre las desventajas de este modelo están:

No.	Desventaja
1	Uso excesivo de sesiones en un desarrollo Web, lo cual redunda en detrimento del servidor en el cual está alojado el sistema.
2	Se requiere una verificación adicional cuando el usuario manipula los parámetros directamente.
3	Se requiere de una técnica de cifrado adicional para almacenar el contenido de las variables de Sesión en el cliente.

Tomado de: Autores

Tabla 6. Desventajas sobrecarga

1.3 SOBRECARGA DE SESIONES

Una mejora radical en el concepto de seguridad para aplicaciones Web la ofrece el manejo de sesiones. Este modelo permite verificar la identidad de un usuario cuando ingresa al sistema y genera un identificador único mientras el usuario utiliza la aplicación. Si bien es cierto este concepto optimiza el proceso de verificación cuando se procesa la identidad de un usuario, es insuficiente cuando él manipula los parámetros de la aplicación. Este problema se corrige asignando no una, sino varias variables de sesión durante el uso del aplicativo, sin embargo esta solución conlleva a otro problema, el cual es conocido como sobrecarga de sesiones.

2. MODELO CRIPTOGRÁFICO PROPUESTO.

Antes de plantear el modelo hay que tener en cuenta las siguientes consideraciones:

- En los desarrollos Web es importante la autenticación [4], sin embargo a la hora de interactuar con el sistema el factor preponderante es la autorización, toda vez que esta limita el alcance de un usuario en el sistema.
- Todo desarrollador que desee usar este modelo de cifrado, debe tener en cuenta que la información que se gestione en dichos sistemas debe ser importante solo para quien la visualiza, pues no hace uso de protocolos de seguridad para la transmisión de datos.
- La estrategia para el uso de sesiones es un factor clave en este modelo de cifrado, para esto se hace uso del "URL Rewriting" para identificar al usuario en cada una de las peticiones que realiza al servidor, sin embargo el uso de archivos en el cliente no es descartado

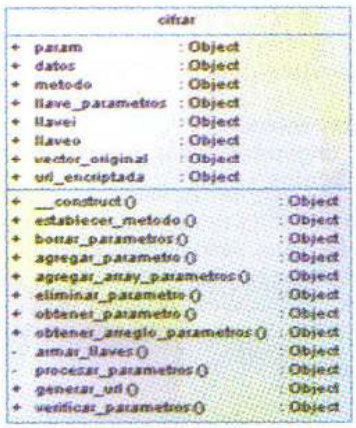
y por el contrario se deja como una alternativa para aumentar el nivel de seguridad.

2.1 ESQUEMA CONCEPTUAL

La idea de este modelo de cifrado es que los estudiantes conozcan a fondo una forma de cifrar información en sus desarrollos Web. Ahora bien, este esquema se soporta en la posibilidad que tiene el desarrollador Web para ocultar información al usuario final, y mostrar de cierta forma solo lo que sea estrictamente relevante.

Adicionalmente al paradigma de programación Web, el desarrollador debe utilizar el paradigma orientado a objetos para reutilizar este modelo, por ende en primera instancia se tiene una clase que puede ser definida como interfaz o como una clase con relaciones simples con el resto del modelo.

La clase denominada "cifrar" gestiona en sus propiedades los parámetros que utiliza la aplicación Web para cada petición, así como las respectivas llaves públicas y privadas para asegurar la seguridad del sistema. La representación conceptual se puede apreciar en la Figura 1.



Tomado de: Autores

Figura 1. Clase cifrar.

Los métodos asociados a la clase, así como el modelo matemático en el cual se soporta el cifrado serán referenciados en el siguiente apartado.

A grandes rasgos el funcionamiento conceptual es el siguiente:

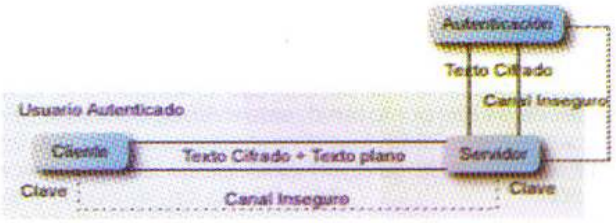
1. El usuario se dispone a entrar al sistema Web, para ello hace uso de una interfaz que cifra su clave y la envía al servidor.
2. El servidor autentica al usuario, si es válido lo deja continuar y crea las variables globales necesarias para su funcionamiento.
3. Una vez en el sistema, el usuario solo puede hacer uso de las opciones predefinidas en el sistema para su rol específico. Todo comportamiento ilícito del usuario será verificado por el modelo de cifrado por lo tanto, la autorización a los datos está garantizada.
4. Toda petición al servidor, ya sea por medio seguro o inseguro será validada por el modelo de cifrado, con lo cual se garantiza que un usuario no autorizado no pueda ingresar al sistema.

Los métodos asociados a la clase, así como el modelo matemático en el cual se soporta el cifrado serán referenciados en el siguiente apartado.

A grandes rasgos el funcionamiento conceptual es el siguiente:

1. El usuario se dispone a entrar al sistema Web, para ello hace uso de una interfaz que cifra su clave y la envía al servidor.
2. El servidor autentica al usuario, si es válido lo deja continuar y crea las variables globales necesarias para su funcionamiento.
3. Una vez en el sistema, el usuario solo puede hacer uso de las opciones predefinidas en el sistema para su rol específico. Todo comportamiento ilícito del usuario será verificado por el modelo de cifrado por lo tanto, la autorización a los datos está garantizada.
4. Toda petición al servidor, ya sea por medio seguro o inseguro será validada por el modelo de cifrado, con lo cual se garantiza que un usuario no autorizado no pueda ingresar al sistema.

La Figura 2 presenta el modelo conceptual referenciado.



Tomado de: Autores

Figura 2. Esquema conceptual de cifrado.

2.2 IMPLEMENTACIÓN DEL MODELO

El modelo de cifrado usa el algoritmo denominado HMAC, el cual se encuentra referenciado en detalle en el documento RFC 2104 [8]. También se hace uso del cifrado con MD5 descrito en el documento RFC-1321 [9]. Para reutilizar la clase y sus métodos, el desarrollador Web debe incluirla en su modelo de objetos, para ello puede integrarla como una clase a nivel de interfaz, o en su defecto como una clase independiente que pueda interactuar con el resto del modelo de objetos a través de relaciones débiles. Una vez integrada o reutilizada la clase, se procede a realizar la instanciación de la misma.

Dado que el modelo de cifrado es para ser utilizado en aplicaciones Web de código abierto, es de aclarar que se utilizó el lenguaje de secuencias de servidor denominado Preprocesador de Hipertexto en su versión cinco (5) -PHP5- [5]. Para el constructor se utiliza la palabra reservada "construct" y tiene por defecto el método md5 para el cifrado de datos.

Otros métodos pueden ser utilizados, como por ejemplo "sh1", sin embargo estos dependen del lenguaje de programación por lo tanto, se toma como referencia el md5

dado que es un estándar mundial. A continuación se presenta el constructor de la clase "cifrar".

```
function __construct($metodo="md5"){
    $this->metodo=$metodo;
    $this->llave_parametros=LLAVE_URL;
    $this->borrar_parametros();
}
```

El constructor inicializa tres propiedades privadas, el primero es el método, el cual es MD5, el segundo es la llave privada al sistema, siendo una constante que el desarrollador puede cambiar a voluntad y en cualquier momento. Se debe tener en cuenta que si este cambio se realiza en tiempo de ejecución, la totalidad de los usuarios serán expulsados del sistema, dado que esta llave es utilizada para cifrar y descifrar en cada petición que se realice al servidor. La llave utilizada es de 64 bits de longitud.

Finalmente, el constructor inicializa la cadena cifrada en vacío, de igual forma, elimina todo posible parámetro o argumento de la aplicación Web, esto se realiza para evitar que algún dato anterior haga parte de la nueva cadena cifrada.

A continuación se presenta el método privado que construye las llaves necesarias para el método de cifrado. Como primera medida se verifican los dos métodos de cifrado, el MD5 y el SH1, queda a criterio de los desarrolladores utilizar nuevos métodos. Como se expresó anteriormente la llave debe ser de 64 bits por lo tanto, este método verifica la misma empacando los valores para el caso en que sobrepase los 64 bits, o rellenando la llave en caso de no cumplir con esta condición.

```
private function armar_llaves(){
 if(!in_array($this->metodo,array("sh1","md5")))
 {
    exit("El metodo ".$this->metodo." NO es  soportado");
 }
 $metodo=$this->metodo;
 $llave=$this->llave_parametros;
 if(strlen($llave)>64){
   $llave=pack("H32",$metodo($llave));
 }elseif(strlen($llave)<64){
    $llave=str_pad($llave,64,chr(0));
 }

      $this->llavei=substr($llave,0,64) ^
str_repeat(chr(0x36),64);
      $this->llaveo=substr($llave,0,64) ^
str_repeat(chr(0x5c),64);
}
```

El paradigma de la programación Web obliga a los desarrolladores a brindar especial atención al manejo de variables locales y globales. Estas variables deben ser procesadas como parámetros de tal manera, que no queden por fuera de la cadena de cifrado ninguno de los parámetros que se involucran en una solicitud que se realiza a un servidor.

Los parámetros a procesar pueden ser gestionados por los dos métodos más populares del protocolo de transmisión de hipertexto, como son GET y POST. Un dato interesante en el manejo de los parámetros que viajan por las aplicaciones Web, es que no tienen un orden específico en cada una de sus peticiones, por lo tanto identificar cual parámetro llega primero y cual le sigue es muy importante en el modelo de cifrado.

En seguridad informática, y en particular en las técnicas de cifrado, el orden de los elementos es relevante por lo tanto, como solución a este inconveniente en el modelo se utiliza el método "ksort" para ordenar indistintamente los parámetros que llegan a través de POST o de GET.

A continuación se presenta el método que permite realizar el procesamiento de los parámetros, nótese que este método incluye la función privada "armar llaves", tomando como base los parámetros que se reciben en la página Web.

```
private function procesar_parametros(){
$vector_parametros=array();
  foreach($this->param as $key => $value){
    $this->param[$key]=urlencode($value);
  }
  ksort($this->param);
  foreach($this->param as $key => $valor){
   $datos=$key . $valor;
   $vector_parametros[]=$key."=".$valor;
  }
  $metodo=$this->metodo;
  $this->vector_original=$vector_parametros;
  $this->datos=$datos;
  $this->armar_llaves();
  $this->llavei;
  $this->llaveo;
    $empaquetar_llavei=pack("H32",$metodo($this->llavei .
$this->datos));
      $this->url_encriptada=$metodo($this->llaveo .
$empaquetar_llavei);
}
```

Ahora bien, hasta este momento se ha presentado la gestión de parámetros y la forma como se cifran los mismos. A continuación se presenta el método en cuestión que permite armar la cadena URL cifrada, la cual se entrega tanto al cliente como al servidor para el procesamiento de los datos.

```
function
generar_url($borrar_parametros=true,$ampersand=true) {
  if(!is_bool($borrar_parametros)){
    throw new Exception("Debe pasar un booleano que indica
si se borran o no los parámetros después de generar la
cadena de URL.");
  }$this->procesar_parametros();
  if ($ampersand){
            $cadena_url=implode("&",$this-
>vector_original)."&url_k=".$this->url_encriptada;
  }else{
      $cadena_url=implode("&amp;",$this-
>vector_original)."&amp;url_k=".$this->url_encriptada;
  }
  if ($borrar_parametros){
    $this->borrar_parametros();
  }return $cadena_url;
}
```

Actualmente el método anterior, está soportado en el estándar ISO para el manejo de caracteres [6], lo cual indica la utilización de caracteres ASCII, sin embargo, los desarrolladores Web pueden adecuar este método para incluir aplicaciones del tipo UTF8.

Una vez cifrada la información y una vez preparados los parámetros a pasar por la URL, se debe verificar en cada página del aplicativo Web la consistencia de los datos y la autorización de los usuarios. Para la verificación de los parámetros que se direccionan a las páginas se utiliza el siguiente método:

```
function verificar_parametros($querystring){
 if ($querystring=="") return true;
 $temp = explode("&",$querystring);
 for ($i=0; $i<count($temp);$i++){
    $art = explode("=",$temp[$i],2);
    $this->param[$art[0]]=urldecode($art[1]);
 }
 $cadena_llave=$this->param["url_k"];
 $this->eliminar_parametro("url_k");
 $this->procesar_parametros();
 $this->url_encriptada;
  if ($cadena_llave!= $this->url_encriptada){
     return false;
  }else{
     return true;
  }
}
```

El método verificar parámetros retorna un valor booleano "falso" en caso que un usuario autorizado haya manipulado la información de manera intencional, o en su defecto si ha tratado de utilizar funcionalidades no asignadas, esto se hace, confrontando la petición realizada por el cliente contra los datos cifrados en ese envío, por lo anterior es que este método debe ser utilizado en la totalidad del aplicativo Web realizado.

La implementación del modelo criptográfico propuesto se verificó y validó con la construcción de un componente para la gestión de usuarios a través de la Web. A continuación se presentan los resultados de la validación del modelo propuesto.

2.3 VALIDACIÓN DEL MODELO PROPUESTO

Para la validación del modelo se construyó un componente Web, denominado usuarios, en el cual se presenta la forma como se visualiza el uso de la clase cifrar en el paso de parámetros por la URL del navegador.

Se trabaja con cuatro (4) secuencias de comandos programados en PHP5:

1. clase_cifrar.php: Contiene la clase y los métodos del modelo propuesto y presentados en la sección 2.2.
2. clase_usuarios.php: Contiene los atributos y métodos que gestionan la información de este componente.
3. usuario_editar.php: Presenta el formulario de edición de datos de un usuario en particular, además permite visualizar el cifrado a través de la dirección URL.
4. usuario_listado.php: Presenta el listado de los usuarios del sistema.

2.3.1 Autorización de eliminación de usuarios.

Un caso relevante en la autorización de permisos es el borrado de datos, no todos los usuarios pueden eliminar un dato del sistema, por tanto, este es un ejemplo ideal para ejemplificar el cómo se debe usar el modelo de cifrado propuesto dentro del código que lista los usuarios (clase_usuarios.php).

```
$Url-
>agregar_parametro("codigousuario",$codigo_usuario);
$cadena = $Url->generar_url(false);
$Url->agregar_parametro("accion","eliminarusuario");
$bandera = $Url->generar_url();

$html ="<tr>";
$html.="<td>{$records->fields["nombreusuario"]}</td>";
$html.="<td>{$records->fields["nombres"]}{$records-
>fields["apellidos"]}</td>";
$html.="<td>";
$html.="<a href=\"usuario_editar.php?{$cadena}\"
title=\"Editar usuario\">";
$html.="<img src=\"../imagenes/editar.gif\"></a>";
$html.="<a href=\"javascript:eliminar_usuario('{$bandera}')\"
title=\"Eliminar usuario\">";
$html.="<img src=\"../imagenes/eliminar.gif\"></a>";
$html.="</td>";
$html.="<td align=\"center\"><input type=\"checkbox\"
onclick=\"javascript:cambiar_color(this);\"";
$html.="name=\"codigousuario[]\" id=\"codigousuario\"
value=\"{$codigo_usuario}\"></td>";
$Html.="</tr>";
```

Como se puede observar la utilización de este esquema de seguridad implica la adición de cuatro líneas de código, lo cual es bastante simple. La Figura 3 presenta la forma como se visualiza el código en un navegador. En la sección Dirección URL del explorador de Internet se puede observar el paso de parámetros con el modelo de cifrado, además al hacer clic en el icono de eliminar usuarios, valida la existencia de permisos y por ende permite la ejecución de dicha orden.

Tomado de: Autores

Figura 3. Página usuario_listado.php.

2.3.2 Autorización de edición de usuarios.

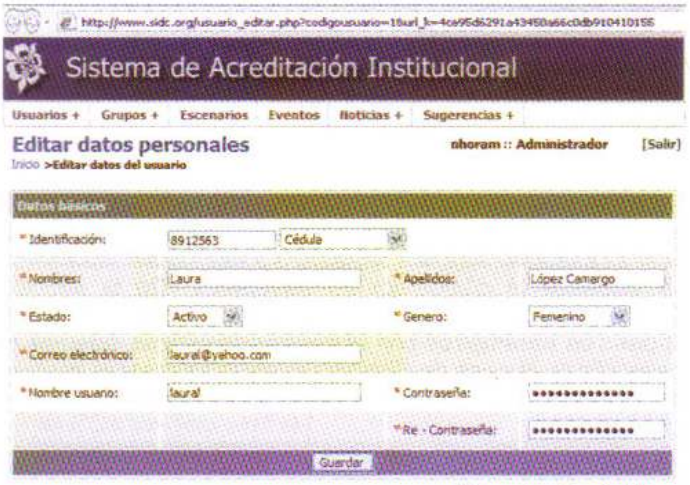
Otra funcionalidad del modelo de cifrado se puede apreciar a la hora de editar información, aunque la información viaje por un medio seguro, se garantiza que el usuario que hace la petición tiene permisos para ejecutar una solicitud. La Figura 4 presenta el formulario respectivo que se construyó para la validación de los datos de un usuario particular.

En el navegador se visualiza la siguiente dirección: usuario_editar.php?codigousuario=1&url_k=4ce95d6291a43450a66c0db910410155, donde:

- codigousuario=1: indica el código del usuario al cual se le deben actualizar los datos.
- url_k= 4ce95d6291a43450a66c0db910410155: es la información cifrada para este usuario de tal manera, que no pueda ser manipulada por ningún usuario intruso.

2.4 COMPARACIÓN DE MODELOS EXISTENTES VS MODELO PROPUESTO

Los modelos anteriormente expuestos se analizaron desde tres perspectivas: Cantidad mínima de procesamiento por solicitud -1-, control sobre el usuario final a nivel de parámetros -2- y rendimiento -3-. A continuación se presenta la comparación entre los modelos existentes y el modelo propuesto de acuerdo a las perspectivas mencionadas:



Tomado de: Autores

Figura 4. Página usuario_editar.php

Modelo	-1-	-2-	-3-
Verificación por unidad de código	×	×	×
Verificación por páginas	×	×	✓
Sobrecarga de sesiones	✓	×	✓
Modelo criptográfico	✓	✓	✓

Tomado de: Autores

Tabla 7. Comparación de modelos

Como se puede apreciar en la tabla anterior, el modelo propuesto cumple con las tres perspectivas analizadas, brindando rendimiento, control sobre la forma como los usuarios manipulan los parámetros y tomando la fortaleza de las sesiones para autenticar las solicitudes que se realicen al aplicativo.

2.5 APORTES DEL MODELO

Luego de conocer la estructura y forma de uso del modelo criptográfico planteado anteriormente, se pueden enunciar algunas de las ventajas o aportes del mismo:

- El rendimiento del modelo de seguridad expuesto es alto pues, solo realizar una verificación completa durante toda la sesión de un usuario en el sistema, las verificaciones parciales por componentes se realizan contra una cadena de datos codificada en MD5 o en su defecto en SH1.
- El tamaño del sistema no es relevante para la utilización del modelo, es decir, no importa si la aplicación cuenta con gran cantidad de componentes.
- No es necesario incluir librerías y demás archivos, solo se requiere una instancia de la clase cifrar y la ejecución de un método para acceder a las funcionalidades del modelo.
- Con este modelo el sistema solo recibirá mensajes del software, pues el usuario final no puede modificar los parámetros que se gestionan por la dirección URL.

3. CONCLUSIONES

La reutilización de código implica un nivel de complejidad alto en la realización de nuevos desarrollos Web. Durante la fase de verificación, se comprobó que los estudiantes confunden el concepto de reutilización con los conceptos de copiar y pegar bloques de código de otras aplicaciones.

El paradigma de desarrollo Web, permite utilizar casi cualquier sistema de seguridad informático, siempre y cuando dicho sistema permita gestionar las variables locales y globales en cada una de las peticiones que hace un cliente al servidor.

Cuando se está desarrollando una aplicación Web, el desarrollador tiene en cuenta los requerimientos del cliente, sin embargo en muchas ocasiones la seguridad del sistema no viene especificada en los requisitos, por lo tanto los desarrolladores no entregan la importancia debida a este componente, lo que influye en que los sistemas en muchas ocasiones sean inseguros.

La construcción de nuevo conocimiento acerca de un modelo matemático no es del todo necesaria para realizar el cifrado de datos, lo realmente importante, es conocer a fondo un modelo criptográfico que permita construir o plantear nuevas estrategias o formas de implementar sistemas de seguridad.

4. REFERENCIAS

[1] AMADOR Durán Toro y Beatriz Bernárdez Jiménez. Metodología para la Elicitación de Requisitos de Sistemas Software. Ministerio de Educación y Ciencia de España. Proyecto "MENHIR" de la CICYT TIC 97–0593–C05–01. Sevilla, Octubre 2000

[2] BRAUDE, Eric. Ingeniería del Software Una perspectiva orientada a objetos. Alfa omega, 1ª Edición, 2003.

[3] CHRISSIS, Mary; KONRAD, Mike y SHRUM Sandy. CMMI Guidelines for Process Integration and Product Improvement. Addison Wesley, 2003.

[4] FERNÁNDEZ Lanvin Daniel. Autentificación y Seguridad en Aplicaciones Web. Universidad de Oviedo. [Citado 10 Noviembre de 2008]: http://www.di.uniovi.es/~dflanvin/docencia/dasdi/teoria/Transparencias/09.%20Autentificacion.pdf

[5] GUTIÉRREZ, Abraham y BRAVO, Ginés. PHP 5 a través de ejemplos. Alfaomega, 1ª Edición, 2005.

[6] ISO 2709:1996 Information and documentation -- Format for information exchange. [Citado 10 Noviembre de 2008]: http://www.hispafuentes.com/hf-doc/man/man7/unicode.7.html

[7] MICROSOFT. [Citado 10 Noviembre de 2008]: http://www.microsoft.com/spanish/msdn/articulos/archivo/030502/voices/xmlwssec.asp

[8] NETWORK Working Group, HMAC: Keyed-Hashing for Message Authentication. http://www.ietf.org/rfc/rfc2104.txt. Enero 2007

[9] NETWORK Working Group, The MD5 Message-Digest Algorithm. http://www.ietf.org/rfc/rfc1321.txt Febrero 2007

[10] NORMA ISO 9001:2000. [Citado 10 Noviembre de 2008]: http://www.homoqualitas.com/castella/infos/iso90002000/portada.htm