DGTIC

Jornadas de visibilidad Web unam 2019

Visibilidad Web UNAM

Buenas prácticas de seguridad web

Ing. Angie Aguilar Domínguez
Coordinación de Seguridad de la Información
UNAM-CERT

Agenda

- Funcionamiento de una aplicación web
- Servidor de aplicaciones web
- Manejador de bases de datos
- Autenticación
- Riesgos para aplicaciones web

Funcionamiento de una aplicación web

¿Por qué seguridad web?

- Cualquier sitio puede ser blanco de ataques.
- No importa el tamaño del sitio o la cantidad de usuarios.
- Blanco para uso de recursos.

Elementos necesarios para la seguridad

- No siempre el ataque viene de afuera.

- Conocer el entorno.
  - Software, versiones, fallos conocidos.

- Pensar y actura antes de que algo malo suceda.
  - ¿Si mi sitio no es importante, qué me puede pasar?

Elementos necesarios para la seguridad

- Principio del privilegio menor.
  - Otorgar sólo los privilegios necesarios.

- Simplicidad.
  - Un sistema simple es fácil de configurar, verificar y usar.

Componentes de una aplicación web

- Autenticación
- Autorización
- Almacenamiento (base de datos)
- Control (reglas del negocio, procesamiento, etcétera)

Características de una aplicación web

- No requiere instalación (cliente)
- Escalabilidad: horizontal y vertical
- Gestión de versión y actualizaciones
- Control personalizado de usuarios
- Seguridad centralizada (sandboxing)

Características de una aplicación web

- Siempre disponible
- Independiente de la plataforma
- Interacción entre usuarios (social)
- Distribución con beneficios económicos
- Tendencia: aplicaciones móviles

Protocolo HTTP

- HyperText Transfer Protocol
  - Versiones 1.0 y 1.1
- Transferencia de datos entre sistemas de forma clara y rápida
- Puerto 80
- Arquitectura cliente-servidor:
  - Petición
  - Respuesta
  - Cuerpo y encabezado separados por línea en blanco

Códigos de estado

Código	Tipo	Descripción
1xx	Informativo	La petición se recibe y sigue el proceso
2xx	Éxito	La acción requerida por la petición ha sido recibida, entendida y aceptada
3xx	Redirección	Para completar la petición se han de tomar más acciones
4xx	Error del Cliente	La petición no es sintácticamente correcta y no se puede llevar a cabo
5xx	Error del Servidor	El servidor falla al atender la petición que aparentemente es correcta.

Métodos empleados

- HTTP usa una serie de métodos para realizar acciones en el servidor.
- Algunos son de ayuda al desarrollo de aplicaciones web.
- En caso de no estar bien configurados, pueden ser usados maliciosamente.

Métodos empleados

- OPTIONS
- HEAD
- GET
- POST
- PUT
- DELETE
- TRACE
- CONNECT

Recomendaciones

- Entender el funcionamiento del protocolo HTTP.
- No olvidar que HTTP puede ser interceptado y modificado en el camino.
- Rechazar cualquier acción no permitida explícitamente.

Servidor de aplicaciones web

Servidor de aplicaciones web

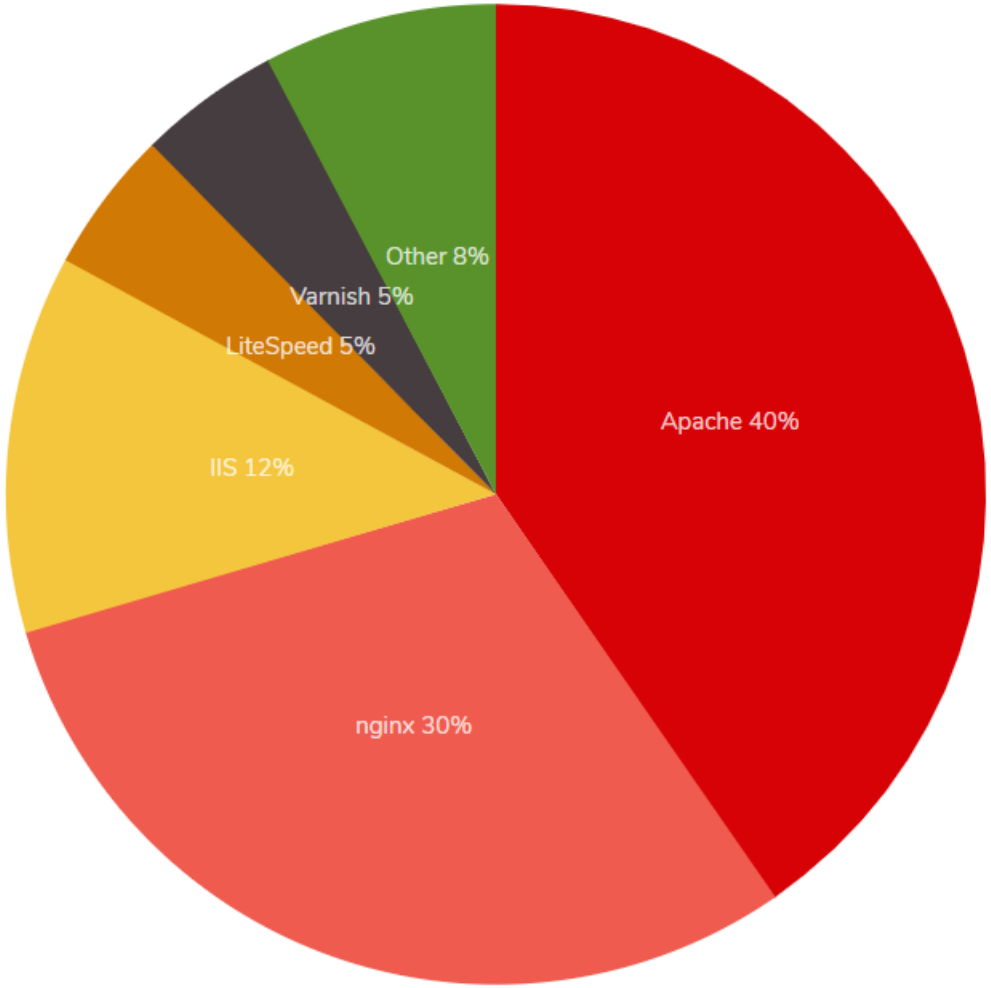
- Tradicionalmente se le conoce como “servidor web” al equipo que ejecuta un programa que procesa las peticiones HTTP:

- Apache HTTPD
- Nginx
- IIS
- Entre otros

Apache HTTPD

- Es el servidor web más utilizado a nivel mundial

- Versión actual: 2.4



Instalación

- Instalación por paquetes:

```
# apt-get install apache2
```

- Ruta por defecto:

/etc/apache2

Sintaxis

- Los comentarios empiezan con #:

```
# Comentario
```

- Incluir otro archivo de configuración:

```
Include /ruta/del/archivo.conf
```

Sintaxis

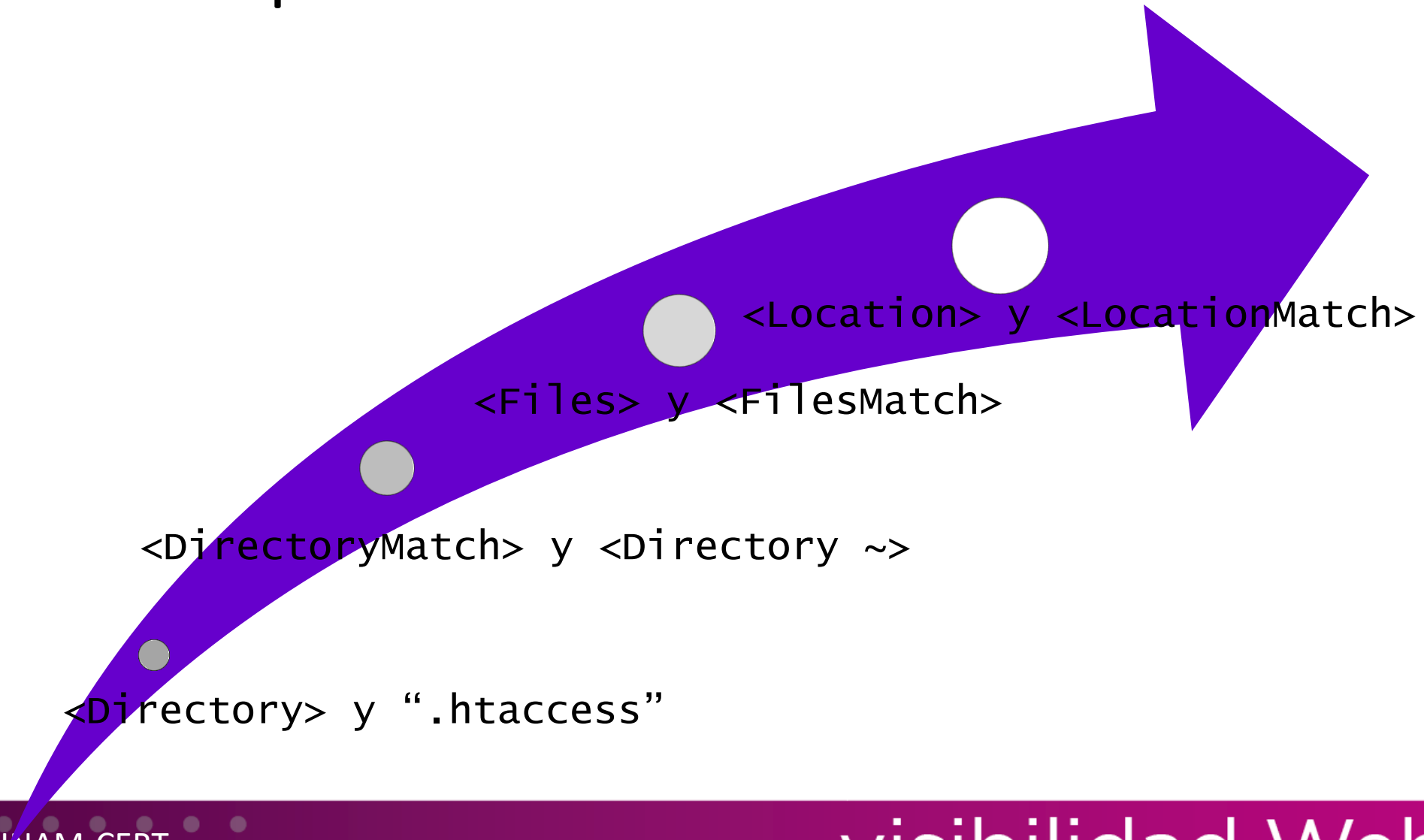
- Una directiva y sus opciones se declaran en una sola línea:

```
Directiva opción opción
```

- Secciones de configuraciones, aplica para una ubicación, ruta, Virtual Host, etc. Puede contener varias directivas:

```
<Directiva>
…
</Directiva>
```

Orden de procesamiento de las directivas



VirtualHost

- Es útil para que un mismo servidor web pueda desplegar varios sitios web.

- Alternativas:
  - Utilizar una dirección IP por cada sitio.
  - Utilizar varios nombres de dominio.

VirtualHost basado en IP

- Se define un sitio por cada dirección IP que tenga asignada el servidor:

```
<VirtualHost 132.248.x.y:80>
     ...
</VirtualHost>

<VirtualHost 132.247.x.y:80>
     ...
</VirtualHost>
```

VirtualHost basado en nombre de dominio

- Se utiliza un nombre de dominio para cada VirtualHost y opcionalmente la dirección IP:

```
<VirtualHost _default_:80>
      ...
</VirtualHost>

<VirtualHost *:80>
  ServerName www.cert.org.mx
      ...
</VirtualHost>
```

Ubicación de VirtualHost

- Ubicación en sistema de archivos:

`/etc/apache2/sites-available`

- Al habilitarse se enlazan simbólicamente hacia:

`/etc/apache2/sites-enabled`

VirtualHost por defecto

- Dos VirtualHost disponibles por defecto:
  - 000-default
  - default-ssl

- Sólo se encuentra habilitado:
  - 000-default

Comandos útiles

- Listar los VirtualHost habilitados:

```
# apache2ctl -S
```

- Habilitar un VirtualHost:

```
# a2ensite sitio
```

- Deshabilitar un VirtualHost:

```
# a2dissite sitio
```

Configuración mínima recomendada

- VirtualHost
- ServerName
- LogLevel
- ErrorLog y CustomLog
- DocumentRoot , <Directory>
- Options
- AllowOverride
- Order
- Allow

TraceEnable

- Permite controlar la disponibilidad del método TRACE de HTTP en el servidor.
- Si se encuentra habilitado, puede hacer el servidor susceptible a *Cross Site Tracing* (XST).
- Viene deshabilitado de forma predeterminada en versiones recientes de Apache HTTPD.

ServerSignature

- Habilitado de forma predeterminada.
- Muestra en las páginas de error la versión de Apache HTTPD y de los módulos cargados.
- Ayuda a descubrir las vulnerabilidades conocidas.
- Se recomienda deshabilitarlo.

ServerTokens

- Habilitado de forma prestablecida.
- Permite divulgar la versión de Apache HTTPD y los módulos en las cabeceras del protocolo.
- No es un fallo de seguridad, pero conocer la versión hace más fácil obtener la lista de fallos existentes.

ErrorDocument

- Permite mostrar un mensaje o página de error personalizado.
- Se requiere una por cada código de error en el que se desee un mensaje personalizado.
- Por ejemplo:

```
ErrorDocument 500 http://www.unam.mx/
```

<Limit>

- Permite restringir los métodos HTTP que se pueden utilizar en una ubicación, a excepción del método TRACE.

```
<Location />
  <Limit OPTIONS HEAD GET POST>
    Order Allow,Deny
    Allow From ALL
  </Limit>
</Location>
```

Headers

- Configuración de la respuesta HTTP.
- Debe de estar habilitado el módulo `mod_headers`.
- Por ejemplo:

```
Header edit Set-Cookie ^(.*)$ $1;httponly
```

mod_ssl

- Apache HTTP usa el módulo mod_ssl como interfaz para OpenSSL.

- Proporciona un canal seguro utilizando el protocolo SSL y TLS.

Generación de certificado SSL

- Para habilitar el soporte de SSL en Apache es necesario contar con un certificado digital.

- Primero se debe generar una llave:

```
# openssl genrsa -out apache.key 4096
```

Entrada: Contraseña para la llave (ninguna)

Salida: Llave privada apache.key

Generación de certificado SSL

- Generar el archivo de solicitud de firma de Certificado (.csr):

```
# openssl req -new -days 365
-key apache.key -out apache.csr
```

Entrada: Llave privada “apache.key”
Datos de la solicitud
Salida: Archivo de solicitud de firma “apache.csr”

Generación de certificado SSL

- Generar el certificado:

```
# openssl x509 -req -days 365
      -signkey apache.key
      -in apache.csr -out apache.crt
```

Entrada: Archivo de llave privada “apache.key”
Archivo de solicitud de firma “apache.csr”
Salida: Certificado digital “apache.crt”

Generación de certificado SSL

- Establecer los permisos adecuados para el certificado y la llave:

```
# chown -cR root:ssl-cert /etc/ssl/certs/apache.crt
# chown -cR root:ssl-cert /etc/ssl/private/apache.key
# chmod -c 644 /etc/ssl/certs/apache.crt
# chmod -c 640 /etc/ssl/private/apache.key
```

Generación de certificado SSL

- Editar el archivo “default-ssl”:

```
# cp default-ssl sitiopost-ssl.conf
```

- Incluir el certificado en el archivo de configuración:

```
SSLCertificateFile      /etc/ssl/certs/apache.crt
SSLCertificateKeyFile   /etc/ssl/private/apache.key
```

Generación de certificado SSL

- Activar el módulo `ssl` con:

```
# a2enmod ssl
```

- Después debe activarse el nuevo sitio por medio de la siguiente sentencia:

```
# a2ensite default-ssl.conf
```

Generación de certificado SSL

- Reiniciar el servicio:

```
# /etc/init.d/apache2 restart
```

- En un navegador de Internet, verificar:

```
https://localhost
```

Let’s Encrypt

- Requiere ciertas condiciones para su implementación:
  - Registro DNS existente (nombre de dominio asignado).
  - IP visible públicamente.
  - Instalación del cliente de Let’s Encrypt (certbot).
  - Extra: configuración del firewall para permitir la conexión entre el servidor de Let’s Encrypt y el servidor web.

Let’s Encrypt

- Es importante contemplar:
  - Versión de sistema operativo y servidor de aplicaciones web vs versión del cliente de Let’s Encrypt.
  - El certificado tiene una validez de 3 meses.
  - Puede establecerse un proceso de renovación automática, mediante una tarea programada.

Vulnerabilidades conocidas de HTTPS

- Drown Attack – Marzo 2016
  - SSLv2

- Weak DH & The Logjam Attack – Mayo 2015
  - TLS – Llaves DH menores a 1024 bits
- Freak Attack – Marzo 2015
  - Apple's SecureTransport and OpenSSL

- POODLE – Octubre 2014
  - SSLv3

- Heartbleed – Abril 2014
  - OpenSSL

- Weak Diffie-Hellman and the Logjam Attack

SSLProtocol

- Esta directiva sirve para indicar que protocolos utilizar, la configuración predeterminada es:

```
SSLProtocol all -SSLv2
```

- Se recomienda usar:

```
SSLProtocol all –SSLv2 –SSLv3 -TLSv1 –TLSv1.1
```

SSLCipherSuite

- Sirve para listar que *ciphers* puede negociar y utilizar un cliente web. La configuración predeterminada es:

```
SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5
```

- Se recomienda usar:

```
SSLCipherSuite HIGH:!MEDIUM:!aNULL:!NULL:!MD5

SSLCipherSuite "ECDH+aRSA+AES256 ECDH+aRSA+AES128 AES256-SHA”
```

Manejador de bases de datos

Manejador de bases de datos

- Interacción con aplicaciones web:
  - Gran cantidad de almacenamiento.
  - Interacción adecuada con lenguajes de programación.
  - Uso actual ampliamente difundido: bases de datos de tipo relacional.

Sistema Manejador de Bases de Datos Relacionales

- Del inglés: Relational Data Base Management System (RDBMS).
- Software que controla la organización, almacenamiento y recuperación de los datos.
- Acepta solicitudes de la aplicación y ordena al sistema operativo transferir los datos apropiados.

Seguridad en bases de datos

- Objetivo: lograr una mayor seguridad de la información manejada.

- Para el caso de los RDBMS, consiste en ajustar la configuración predeterminada.

- Se busca implementar:
  - Integridad de los datos.
  - Confidencialidad de los datos.
  - Disponibilidad de los datos.

Usuarios

- En algunos manejadores, los usuarios y roles son lo mismo.
- Es necesario especificar roles de acceso hacia la base de datos.
- Emplear las diferentes opciones disponibles de acuerdo al manejador dependiendo de las necesidades.
- Funcionar con el principio del privilegio menor.

Privilegios

- Los diferentes privilegios que se les pueden asignar a un rol a un objeto de base de datos son los siguientes:

Parámetro	Descripción
SELECT	Acceso a todas las columnas de una tabla/vista específica.
INSERT	Inserta datos en todas las columnas de una tabla específica.
UPDATE	Actualiza todas las columnas de una tabal específica.
DELETE	Elimina filas de una tabla específica.
TRUNCATE	Elimina filas de una tabla específica, además que resetea los auto-incrementales a 0.
REFERENCES	Referencia a otra tabla existente (foreign key).

Privilegios

Parámetro	Descripción
TRIGGER	Crear un trigger en la tabla indicada.
EXECUTE	Ejecutar una función y el uso de operadores que implementa.
USAGE	Usar un lenguaje procedural, esquema, secuencia, empaquetador de datos externos, servidores.
CREATE	Crear nuevos esquemas, o nuevos objetos dentro de.
CONNECT	Conexión a la base de datos específica.
TEMPORARY	Creación de tablas temporales dentro de la base de datos.
ALL	Todos los privilegios anteriores.

Usuarios y privilegios

- Se pueden dar de alta distintos roles y otorgarles privilegios.
- DBA no debería ser DBO principal de ninguna base de datos.
- Ejemplo: 2 usuarios para cada base de datos, el DBO y otro con menor privilegio.
- Negar el acceso a rutinas del sistema y al sistema de archivos, (sólo para el DBA).

Usuarios y privilegios

- Ejemplo:
  - Tabla de usuarios

Usuario	Select	Update	Delete	Insert
DBA	X	X	X	X
usuario1	X			X
usuario2		X	X	
usuario3	X			

PostgreSQL

- SMBDR de código abierto y gratuito.
- Compatible con distintos sistemas operativos y lenguajes de programación.
- Puede manejar gran cantidad de datos.
- Permite implementar gestión de usuarios y privilegios.

Archivos de configuración

- Ruta de instalación por defecto:

/etc/postgresql/version/main

- Archivo principal de configuración:
  - postgresql.conf
- Archivo de configuración de acceso al manejador:
  - pg_hba.conf
- Archivo de configuración de mapeo de usuarios:
  - pg_ident.conf

Usuarios y privilegios

- Considerar 2 elementos:
  - Archivo de configuración de accesos (pg_hba.conf).
  - Alta de usuarios dentro del manejador (SQL).

- Abrir el archivo principal de configuración de PostgreSQL:

```
~# vim /etc/postgresql/9.1/main/pg_hba.conf
```

PSQL

- Cliente que se instala por defecto al realizar la instalación de código fuente.
- Permite la conexión a las bases de datos.
- Cuenta con ayuda para su manejo.
- Indica de forma sencilla la acción realizada.

PSQL

- Cuenta con ayuda sobre:
  - Uso de SQL.
  - Uso del mismo cliente.
- Directivas propias del cliente.
- Cursor de acuerdo a:
  - Base de datos en uso.
  - Usuario conectado (DBA, otros).
  - Indica cierre de sentencia.

Comandos en PSQL

- Establecer conexión a una base de datos en específico:
  `psql -d nombre_base_de_datos`
- Listar las bases de datos existentes: `\l`
- Listar los privilegios existentes: `\z`
- Listar el detalle de las tablas: `\d nombre_tabla`
- Salir del cliente: `\q`

Alta de usuarios y privilegios

- Cambiar la contraseña del usuario postgres:

```
# su - postgres
~$ psql

=# ALTER USER postgres ENCRYPTED PASSWORD ‘XXXXX';
=# \q
```

Alta de usuarios y privilegios

- Cambiar:

```
local      all         postgres          peer
```

- Por:

```
local      all         postgres          md5
```

- Reiniciar el servicio:

```
~$ exit
~# service postgresql restart
```

Alta de usuarios y privilegios

- Editar el archivo de configuración pg_hba.conf (como usuario root) cambiando:

```
#Database administrative login Unix domain socket
local    all        postgres       peer
```

- Por:

```
#Database administrative login Unix domain socket
local    all        postgres       md5
```

Alta de usuarios y privilegios

- Reiniciar el servicio PostresSQL para aplicar los cambios:

```
~# service postgresql restart
```

- Ahora cada vez que el usuario postgres intente conectarse con psql en modo local, deberá ingresar la contraseña establecida previamente.

Alta de usuarios y privilegios

- Es recomendable establecer sólo los privilegios necesarios para cada usuario.
- Por ejemplo:

Usuario	Lectura	Escritura	Actualizar	Borrar
Dbo (DBO)	X	X	X	X
user1	X	X		
user2	X			

Usuarios y privilegios: recomendaciones

- Sólo cuentas de usuario indispensables
- Principio del privilegio menor
- Privilegios sobre grupos de usuarios

Recomendaciones de seguridad para el manejador

- Conexiones sólo a servidores y usuarios autorizados.
  - Desactivar todas las conexiones que no se estén empleando.
- Establecer roles y privilegios de acceso.
- Limitar el acceso a las rutinas del sistema y al sistema de archivos.

Recomendaciones de seguridad para el manejador

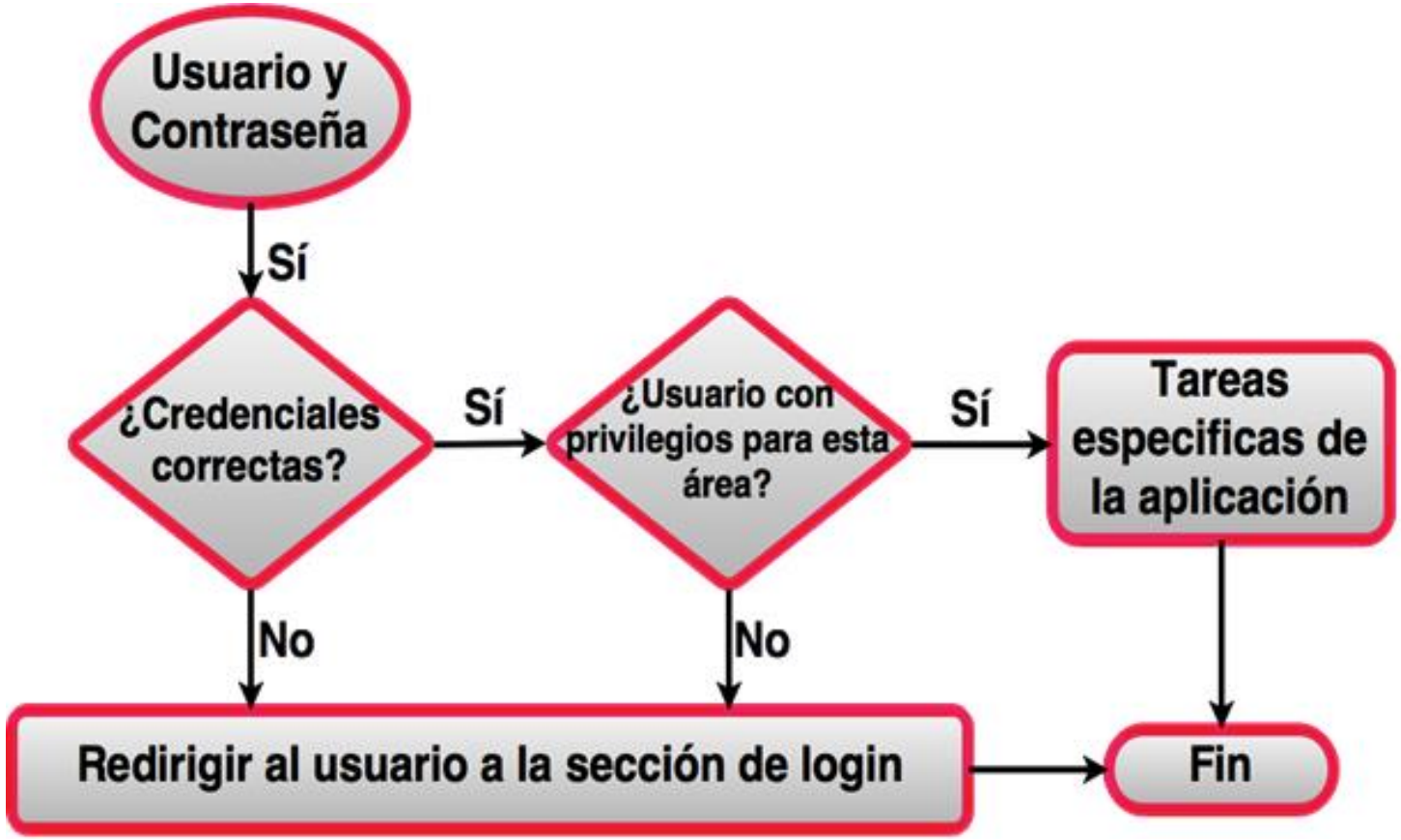
- Utilizar un mecanismo de autenticación que soliciten contraseña a los usuarios.
- Limitar conexiones únicamente de direcciones IP permitidas.
- De ser posible, implementar SSL para las conexiones al manejador.

Autenticación

Autenticación

- Permite detectar y comprobar la identidad de un usuario.

Autenticación de un sistema web



Autenticación basic

- Usada para restringir un recurso web
- La cabecera contiene la palabra Basic (esquema a usar)
- En su uso, el navegador muestra una ventana de login
- Fuerza bruta trivial para obtener las contraseñas
- Se recomienda cifrar el tráfico

Autenticación basic

Client Request: HTTP 1.1 GET /restringido.html

Server Reply:
“HTTP/1.1 401 Authorization Required”
WWW-Authenticate: Basic realm=Camelot

Client Request: HTTP 1.1 GET /restringido.html
Authorization: Basic dGVzdDpwYXNzd29yZA==

Autenticación digest

- Similar a Basic

- Aplica hash md5 a la cadena que combina los siguientes elementos y es enviado en la cabecera:

  usuario:realm:contraseña:nonce

- Con Digest no se envía la contraseña: se puede considerar un reemplazo un poco más seguro que Basic

Autenticación digest

Client Request: HTTP 1.1 GET /restringido.html

Server Reply:
WWW-Authenticate: Digest realm="camelot", nonce="Ny8yLzIwMDIgMzoyNjoyNCBQTQ", opaque="0000000000000000", stale=false, algorithm=MD5, qop="auth"

Client Request:
Authorization: Digest username="Arthur", realm="camelot", qop="auth", algorithm="MD5", uri="/restringido.html", nonce="Ny8yLzIwMDIgMzoyNjoyNCBQTQ", nc=00000001, cnonce="c51b5139556f939768f770dab8e5277a", opaque="0000000000000000", response="afa30c6445a14e2817a423ca4a143792"

Diferencia entre basic y digest

Basic

- Usuario y contraseña codificados en Base64
- Considerado el método menos seguro de todos
- Puede ser fácilmente deshecho
- La cabecera de autenticación WWW es enviada con la palabra *Basic* y las credenciales del usuario en claro
- Las credenciales del usuario persisten en el cliente además de que no hay cierre de sesión

Digest

- Utiliza funciones *hash* MD5 en el usuario y contraseña
- Diseñada para ser más segura que la autenticación *Basic*
- Es complicado obtener las credenciales del usuario
- La cabecera de autenticación WWW es enviada con la palabra *Digest*, un *realm* entre otros elementos
- Las credenciales del usuario persisten en el cliente además de que no hay cierre de sesión

Formularios

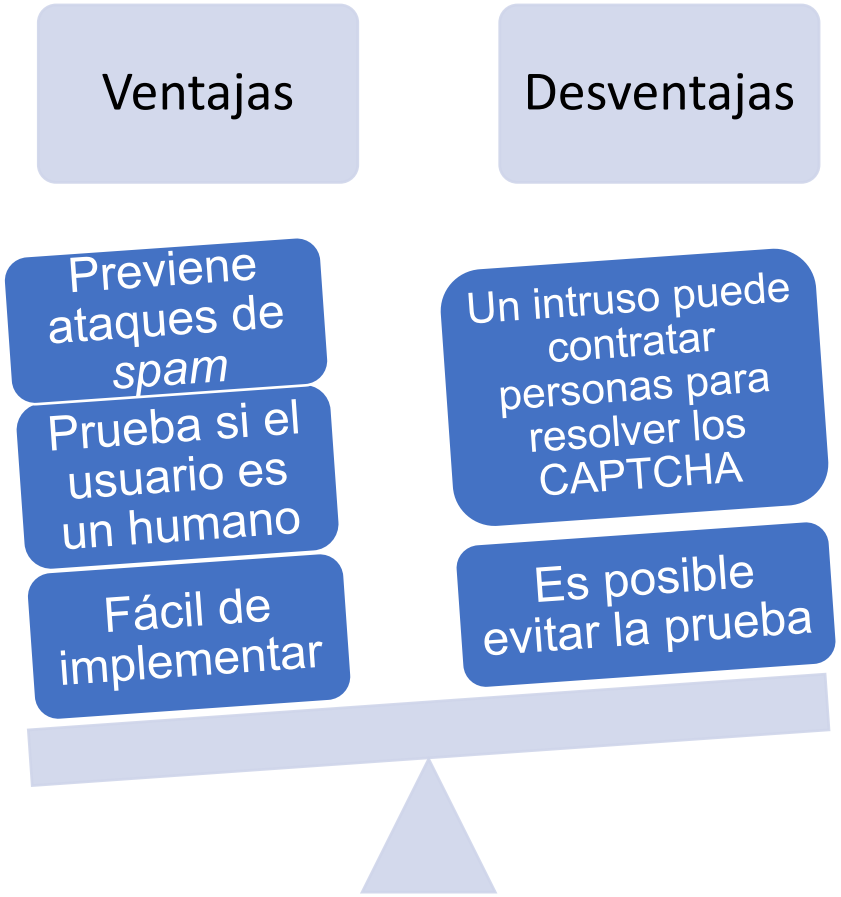
- Amigable con el usuario
- Hace uso de usuario y contraseña.
- Los datos se procesan por medio de lenguaje de programación.

Formularios

Captcha

- Complemento para la autenticación por medio de formularios.
- Ayuda a identificar si el usuario es humano o se encuentra automatizado.
- Evitar que robots (spambots) accedan a ciertos servicios.

Captcha


Ventajas
Desventajas
Previene ataques de *spam*
Prueba si el usuario es un humano
Fácil de implementar
Un intruso puede contratar personas para resolver los CAPTCHA
Es posible evitar la prueba

reCAPTCHA

- *reCAPTCHA* es un servicio gratuito de *captcha* por parte de Google.
- Utilizado para proteger sitios web contra abusos de spam o bots.
- Busca mantenerse robusto en cuanto a seguridad y funcionalidad.
- Puede resultar molesto para el usuario.

Implementación de reCAPTCHA

- Ingresar a:
  - https://www.google.com/recaptcha/intro/v3.html
  - Dar clic en el botón Admin Console, pedirá ingresar con una cuenta de google.
  - Se pueden registrar cuantos dominios se desee.
  - Para concluir dar clic en el botón de “+”.
  - Seleccionar *reCAPTCHA V2* y llenar el formulario.
  - Se generará un par de llaves.

Implementación de reCAPTCHA

- Para implementar el captcha, es necesario copiar las dos líneas de código que vienen en seguida de las claves.

- La primera cargará el estilo y el bloque de reCAPTCHA.

- La segunda se utilizara como bloque de referencia para que la clave del sitio sea encontrada por el servicio.

Implementación de reCAPTCHA

- Descargar la biblioteca reCAPTCHA disponible en el repositorio oficial de Google (se hara uso de la carpeta src)

- Por ejemplo:

```
git clone https://github.com/google/recaptcha.git
```

- Utilizar la biblioteca reCAPTCHA para realizar la validación del captcha con el servidor de Google.

Implementación de reCAPTCHA

- Ejemplo de código en PHP:

```
<?php
  …
        require_once ”recaptcha/src/autoload.php";
        $secret = "llave-secreta";
        $response = null;
        $reCaptcha = new \ReCaptcha\ReCaptcha($secret);
  …
?>
```

Implementación de reCAPTCHA

- Incluir en la sección HTML del sitio, la siguiente línea:

```
<script src="https://google.com/recaptcha/api.js"
async defer></script>
```

- Incluir dentro la sección del formulario:

```
<div class="g-recaptcha" data-site-key="llave-
publica"></div>
```

Implementación de reCAPTCHA

Ejemplo de código en PHP:

```
<?php
…
$captcha = $_POST['g-recaptcha-response'];
$ip = $_SERVER['REMOTE_ADDR'];
$llavesecreta = ”llave-secteta";
$response=file_get_contents("https://www.google.com/recaptcha/api/siteverify?secret=".
$llavesecreta."&response=".$captcha."&remoteip=".$ip);
$respuesta = json_decode($response,true);
…
?>
```

Recomendaciones de seguridad

- Se debe seleccionar el método de autenticación de acuerdo a las reglas del negocio.
- También es bueno documentarse sobre otros métodos de autenticación.
- Considerar que cada forma de autenticar tiene ventajas y desventajas (operativas y de seguridad).

Recomendaciones de seguridad

- La implementación de captcha puede ser con una biblioteca de terceros, pero hay que informarse sobre:
  - Actualizaciones y mejoras.
  - Fallos de seguridad conocidos.
  - Ventajas y desventajas en su implementación.

Riesgos para aplicaciones web

OWASP

- OWASP, Open Web Application Security Project, es un grupo de seguridad y profesionales de aplicaciones web.

- Su sitio oficial es:

- http://www.owasp.org

OWASP

- Promueve el mejor uso de los recursos y la implementación de buenas prácticas, para esto:
- Posee una base de datos en Internet de vulnerabilidades.
- Además cuenta con guías gratuitas para el desarrollo.
- Algunas herramientas disponibles: WebGoat, OWASP ZAP.

Top 10 de OWASP

- En 2017 OWASP publicó un top 10 de las vulnerabilidades mas comunes en las aplicaciones web. Sólo se mencionarán 5.

- El objetivo principal es educar y difundir sobre las vulnerabilidades más importantes.

- Esta publicación provee de técnicas básicas sobre cómo protegerse y orientación sobre qué acciones tomar.

1. Inyección

- Se produce cuando datos que no son de confianza son enviados como parte de un comando o consulta a un intérprete.

- Los datos maliciosos pueden engañar al interprete para que ejecute comandos no deseados o de acceso a datos sin autorización.

1. Inyección

- Consecuencias:
  - Pérdida y/o corrupción de datos.
  - Negación de acceso.

- Acciones:
  - Implementar código seguro: validación de entradas, incluso las provenientes de otros sistemas.
  - Uso de herramientas para realizar la búsqueda de vulnerabilidades a ataques de inyección.
  - Es altamente recomendable mantener datos no confiables separados de comandos y consultas.
  - Uso de API seguras (no intérpretes).

2. Pérdida de autenticación

- Son las vulnerabilidades relacionadas con la pérdida de autenticación y gestión de sesiones.

- Son críticas en la seguridad de las aplicaciones, ya que permiten a un atacante suplantar la información de un determinado usuario:
  - Pudiendo llegar a obtener una cuenta de administración que le permita sabotear los controles de autorización y registro de la aplicación.

2. Pérdida de autenticación

- Consecuencias:
  - Acceso no autorizado a cualquier tipo de información que se encuentre almacenada en el servidor.
  - Acceso a servicios que han sido comprometidos.

- Acciones:
  - Considerar una buena autenticación de los usuarios.
  - Proteger los datos de sesión (ID, contraseña, token).
  - Realizar seguimiento robusto de sesiones (se recomienda modificar el token de sesión cada cierto tiempo).
  - Evitar vulnerabilidades del tipo XSS, ya que pueden provocar el secuestro de datos de sesión.

3. Exposición de datos sensibles

- Los fallos de configuración, validación de acceso, envío de datos en texto plano entre cliente y servidor, así como almacenamiento incorrecto de respaldos comprometen datos que deberían estar protegidos.

- Esto incluye:
  - Información Personal Sensible (PII) como registros de salud, datos personales, credenciales y tarjetas de crédito.
  - Archivos de configuración del servidor y la aplicación.
  - Respaldos de la base de datos o el sitio completo: usuarios y contraseñas incluidos.

3. Exposición de datos sensibles

- Consecuencias:
  - Robo de información sensible -> violación de leyes vigentes.
  - Accesos no deseados a la aplicación y/o servidor.

- Acciones:
  - No almacenar datos sensibles innecesariamente -> cifrar (en la medida de lo posible).
  - Cifrar el canal de comunicación entre cliente y servidor, con otras aplicaciones y/o manejador de bases de datos.
  - Verificación de la aplicación completa en búsqueda de exposición accidental o intencional de datos sensibles.

4. Entidades externas XML (XXE)

- Explotación de procesadores XML vulnerables al cargar o incluir contenido hostil en un documento XML, explotando código vulnerable, dependencias o integraciones.

4. Entidades externas XML (XXE)

- Consecuencias:
  - Extracción de datos y ejecución de solicitudes remotas desde el servidor.
  - Escaneo de sistemas internos y realización de ataques de denegación de servicio y otros tipos de ataques.

- Acciones:
  - Usar formatos de datos más sencillos (JSON) y evitar serializar datos confidenciales.
  - Actualizar los procesadores y bibliotecas XML.
  - Implementar validación con listas blancas, filtrado y sanitización.
  - Verificar que al cargar archivos XML o XSL se valide el XML, usando validación del esquema XML o similar.

5. Pérdida de control de acceso

- Fallo que conduce a la divulgación, modificación o destrucción de información no autorizada, a realizar una función de negocio fuera de los límites del usuario.

5. Pérdida de control de acceso

- Consecuencias:
  - Cambio de ID de usuario, permitiendo el acceso o edición de cuentas ajenas.
  - Pasar por alto las comprobaciones de control de acceso modificando la URL.
  - Elevación de privilegios: actuar como un usuario sin iniciar sesión o actuar como administrador siendo usuario estándar.

- Acciones:
  - Restringir el acceso a todo: salvo recursos públicos.
  - Implementar mecanismos de control de acceso una vez y replicarlo en toda la aplicación.
  - Control de acceso de acuerdo al perfil del usuario (usuarios y privilegios).
  - Incluir pruebas de control de acceso a la aplicación.

Otros puntos de seguridad

- Configuración de seguridad incorrecta.
- Cross-Site Scripting.
- Deserialización insegura.
- Uso de componentes con vulnerabilidades conocidas.
- Registro y monitoreo insuficientes.

WebGoat

- Desarrollo realizado por OWASP en formato de laboratorio.

- Desarrollado con:
  - Java
  - MySQL
  - Apache Tomcat

- Proporciona un entorno realista de una aplicación web.

WebGoat

- Vulnerable al Top 10 de OWASP.
- Recomendación: no implementar en entorno productivo.
- Lecciones:
  - Explicación del elemento.
  - Ejercicios por nivel de complejidad (incluido un plan de lo que debe realizarse).
  - Descripción de los posibles escenarios de mitigación.

WebGoat

- Algunos ejercicios:

- Cross-Site Scripting
  - Almacenado
  - Reflejado
  - Basado en DOM

- SQL injection
  - Tipo 1
  - Tipo 2

Finalmente

- Generación y prueba de respaldos de forma periódica.
- Control de versiones sobre fuente del sitio.
- Documentación de las implementaciones.
- Uso de herramientas para la detección de intrusos (IDS, IPS).

Finalmente

- Uso de alguna herramienta de monitoreo del servidor: logwatch, logcheck.
- Suscripción a las listas de seguridad de los servicios empleados.
- Protección adicional: WAF.

GRACIAS

Coordinación de Seguridad de la Información
UNAM-CERT
Ing. Angie Aguilar Domínguez