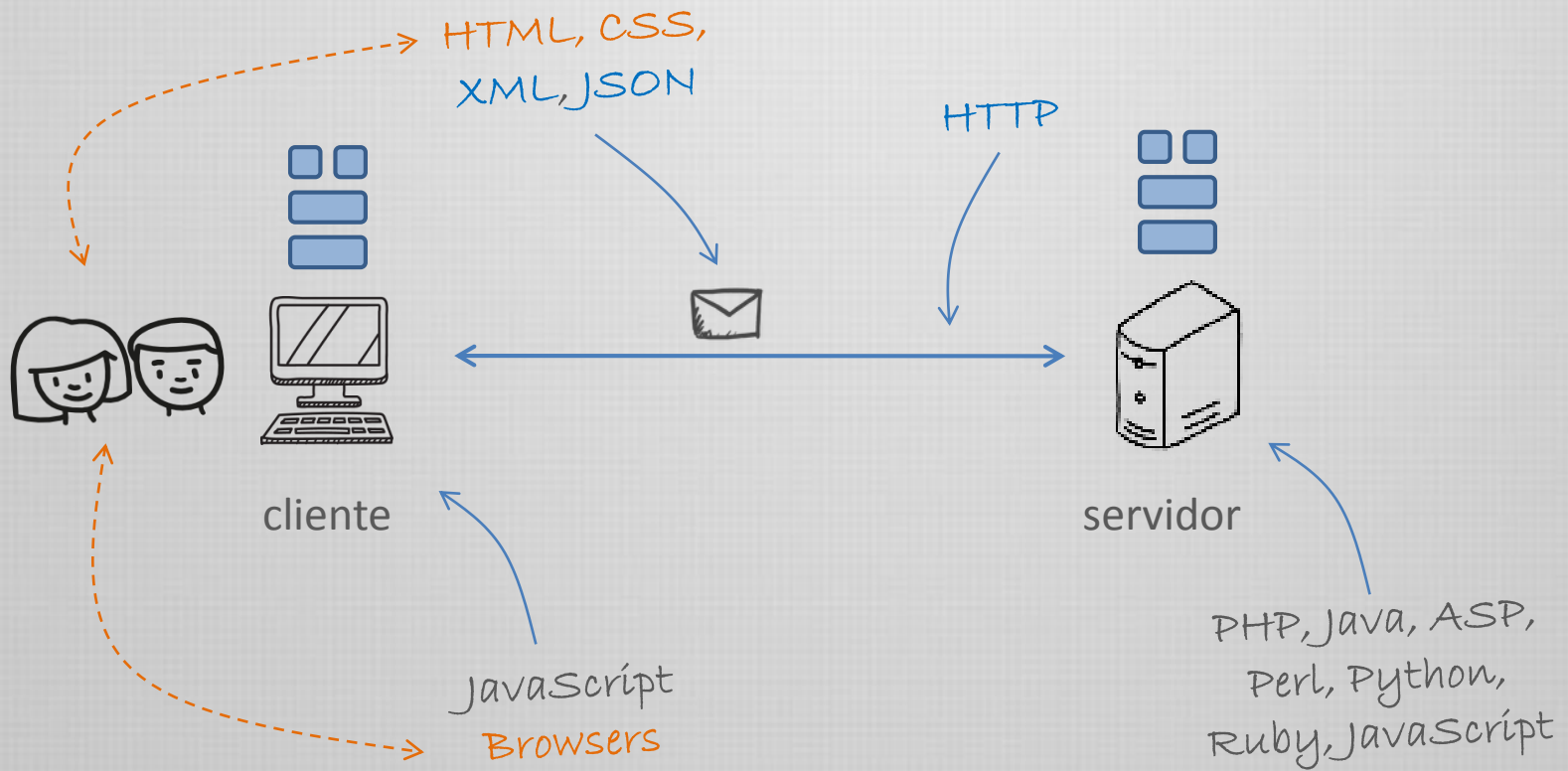


Ingeniería de Aplicaciones Web

Diego C. Martínez

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Escenario de los servicios web



Seguridad en la Web

La seguridad es un aspecto importante de la web.

Requiere especial atención dadas las constantes y crecientes amenazas a la integridad del sitio y de los datos que en él se manipulan.

Es importante conocer los ataques más comunes y la forma de evitarlos desde el lenguaje sobre el que trabajamos.

¿Cuándo un sistema es seguro?

La seguridad no puede probarse, la falta de ella sí.

La seguridad es una medida, no una característica.



Asumir **siempre** que el sistema es inseguro

Top Ten - Vulnerabilidades

Injection

Datos no confiables son enviados como parte de un comando o consulta, causando que el intérprete ejecute comandos o permita accesos sin autorización

Broken Authentication

Implementación incorrecta de los controles de autenticación que causa que un usuario asuma la identidad de otro temporalmente o en forma permanente

Sensitive Data Exposure

Falla en protección adecuada de datos

XML External Entities (XXE)

Abuso de vinculación de entidades externas a XML

Broken Access Control

Falla en control de accesos para los usuarios

Security Misconfiguration

Configuraciones incorrectas o incompletas del sitio

Cross-site scripting

Datos no confiables ingresados via web permiten la ejecución de scripts

Insecure Deserialization

Serialización de objetos que puede manipularse

Components with known vulnerabilities

Uso de componentes defectuosos o vulnerables

Insufficient Logging-Monitoring

Falta de registros que facilitan la indetección

Objetivos de un ataque al sitio

Un ataque a un sitio web puede perseguir varios objetivos

- Robo de información
Datos personales, claves, entorno social, actividades del sitio
- Extorsión
Exigencia de demandas a cambio de información obtenida
- Vandalismo puro
Destrucción sin sentido de datos, estructura del sitio, contenido visual, etc
- Demostración de vulnerabilidades
El ataque tiene como objetivo servir de advertencia a los administradores
- Daño personal o institucional
Deliberadamente se busca perjudicar a un tercero

Seguridad integral

La seguridad debe tenerse en cuenta en

- en el *cliente* (e.g., Apps, Scripts, Firefox, Explorer, Opera, Safari, etc)
- en el *servidor* (e.g., Apache, IIS)
- en las *tecnologías de desarrollo* (e.g., PHP, Java, .NET, Perl).

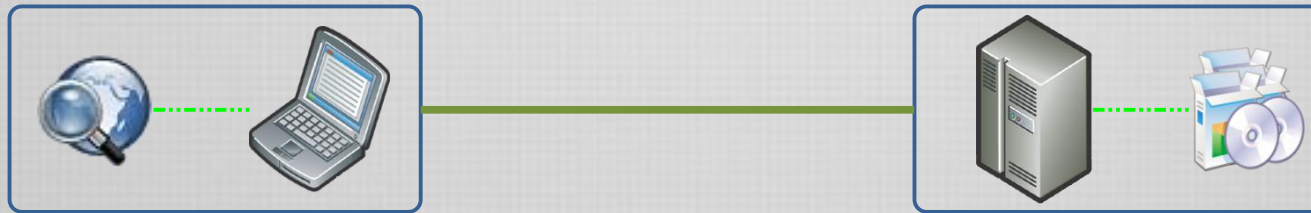


En el *cliente*, es importante
proteger datos personales,
evitar complementos poco seguros
(e.g., plugins no oficiales o poco testeados)
mantener el entorno de ejecución limpio
(e.g., malware, virus, etc)

Seguridad integral

La seguridad debe tenerse en cuenta en

- en el *cliente* (e.g., Apps, Scripts, Firefox, Explorer, Opera, Safari, etc)
- en el *servidor* (e.g., Apache, IIS)
- en las *tecnologías de desarrollo* (e.g., PHP, Java, .NET, Perl).



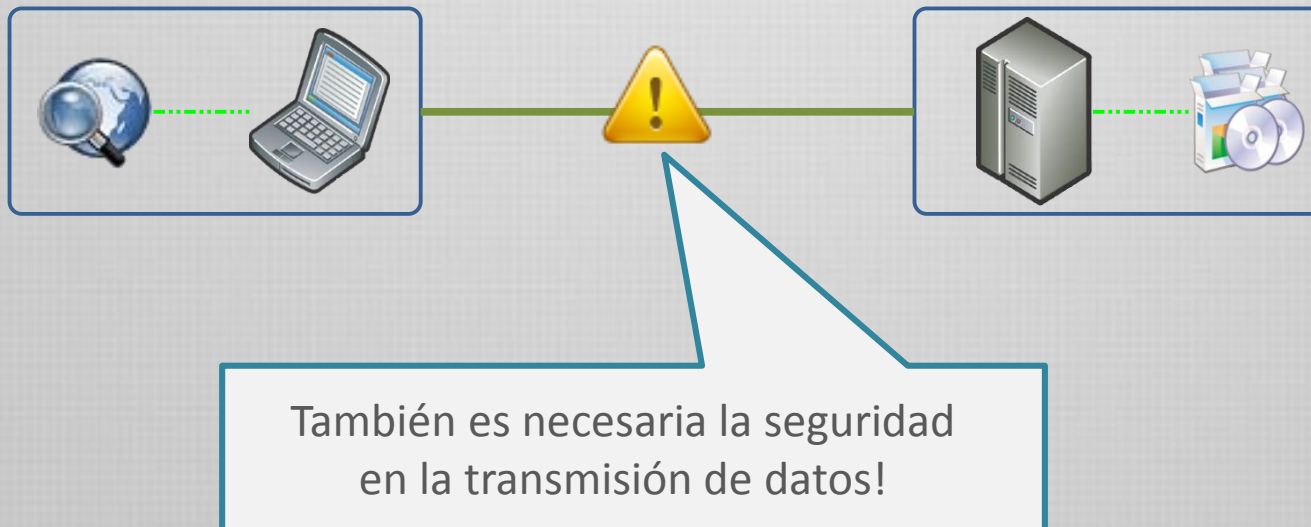
En el *servidor*, es esencial

- conocer la configuración del servidor
- personalizar mensajes de error
- revisar logs diariamente
- no descuidar aplicaciones de test o ejemplos.
- controlar updates y mantener actualizado el software.
- controlar el acceso físico al sector de servidores.

Seguridad integral

La seguridad debe tenerse en cuenta en

- en el *cliente* (e.g., Apps, Scripts, Firefox, Explorer, Opera, Safari, etc)
- en el *servidor* (e.g., Apache, IIS)
- en las *tecnologías de desarrollo* (e.g., PHP, Java, .NET, Perl).



Seguridad integral



En las *tecnologías de desarrollo*, es esencial

- conocer la configuración de la plataforma o framework
- controlar updates y mantener actualizado el framework
- comprender las posibles fallas de seguridad en el código, y evitarlas
- administrar correctamente las versiones de prueba y testeo de nuestros desarrollos (*sandbox*). Eliminar si es innecesaria alguna versión.
- usar correctamente las bases de datos
- encriptar cuando sea necesario (recordar que TODO lo que se envía al cliente es observable: campos “hidden”, cookies, etc)
- ser simple!:

```
$search = (isset($_GET['search']) ? $_GET['search'] : '');
```

- mantenerse actualizado
- trabajar a la defensiva

Principios generales de seguridad web

Diseñar y programar a la defensiva.

La confianza y la subestimación del usuario es el primer enemigo.

Separar servicios adecuadamente

Un servidor web, un servidor de aplicaciones, un servidor de bases de datos, etc

Limitar privilegios de los usuarios

Privilegios en el servidor y privilegios en la aplicación misma

Siempre proteger información sensible

Passwords encriptados, permisos de acceso adecuados, etc

Utilizar estándares

No solo tecnológicos sino también metodológicos

Registrar todo

Utilizar logs de acceso, de transacciones, de monitoreo de usuarios, etc

Testear todo

Utilizar mecanismos manuales y automáticos.

Conocer las vulnerabilidades comunes.

Ataques: Datos sensibles

Parte de los datos manipulados por nuestro sistema será generada por entidades externas, y por lo tanto **no plenamente confiables**.

Usualmente vienen de un formulario, por el URL, por cookies u otras fuentes externas de datos.

Tener en mente la diferencia entre POST y GET, para datos sensibles.

Muchos ataques surgen por *exploración y tanteo* de la forma del URL y sus parámetros (query string)...

Supongamos que iniciamos un proceso para recuperar el password de una cuenta de mail, y detectamos que el URL termina siendo:

`http://mimail.org/reset.php?user=pepe&email=pepe%40mimail.org`

¿cuál será el efecto de invocar manualmente a la siguiente URL?

`http://mimail.org/reset.php?user=juan&email=pepe%40mimail.org`

Esto se denomina *URL attack*.

BASIC-AUTH

Es el mecanismo básico de autenticación en la web (*RFC 2617*)

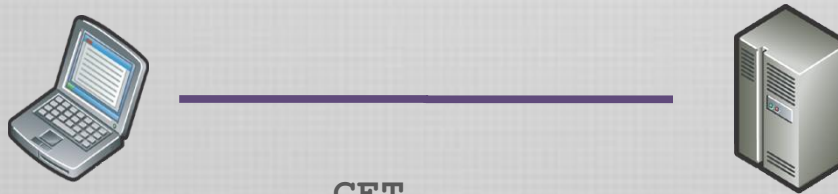
- El nombre de usuario y el password son concatenados y codificados en Base64

Usuario = *homero*
Password = *mipassword*

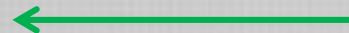
homero:mipassword
aG9tZXJvOnRvZGRhcGVzdGE=

- Se almacena en un header HTTP

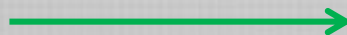
Authorization: Basic aG9tZXJvOnRvZGRhcGVzdGE=



401 - Authorization Requested



GET + user:pass



Los datos pueden
decodificarse

HTTPS

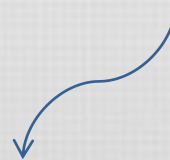
La codificación Base64 no es encriptación.

Es mejor complementado con HTTPS



*Hypertext Transfer
Protocol Secure*

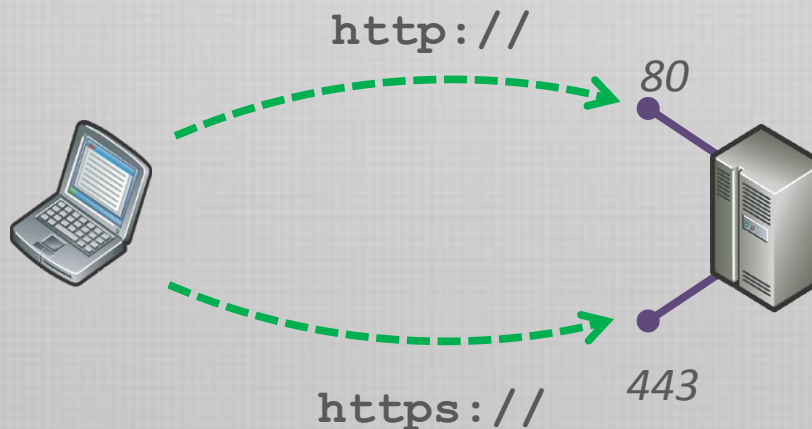
Todos los requerimientos y respuestas HTTP son codificados antes de ser enviados a la red.



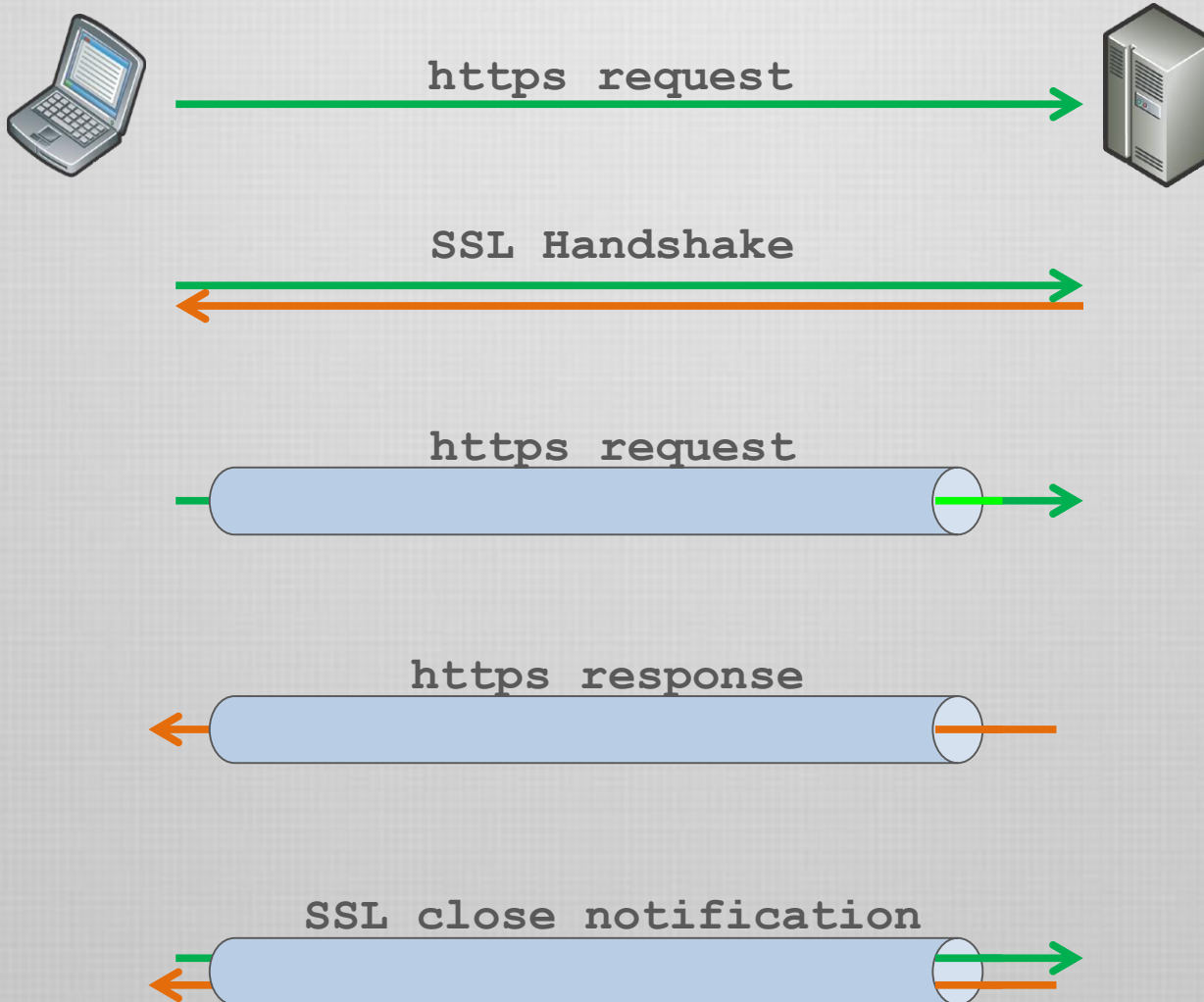
capa de seguridad criptográfica a nivel de transporte

SSL = Secure Sockets Layer

TLS = Transport Layer Security



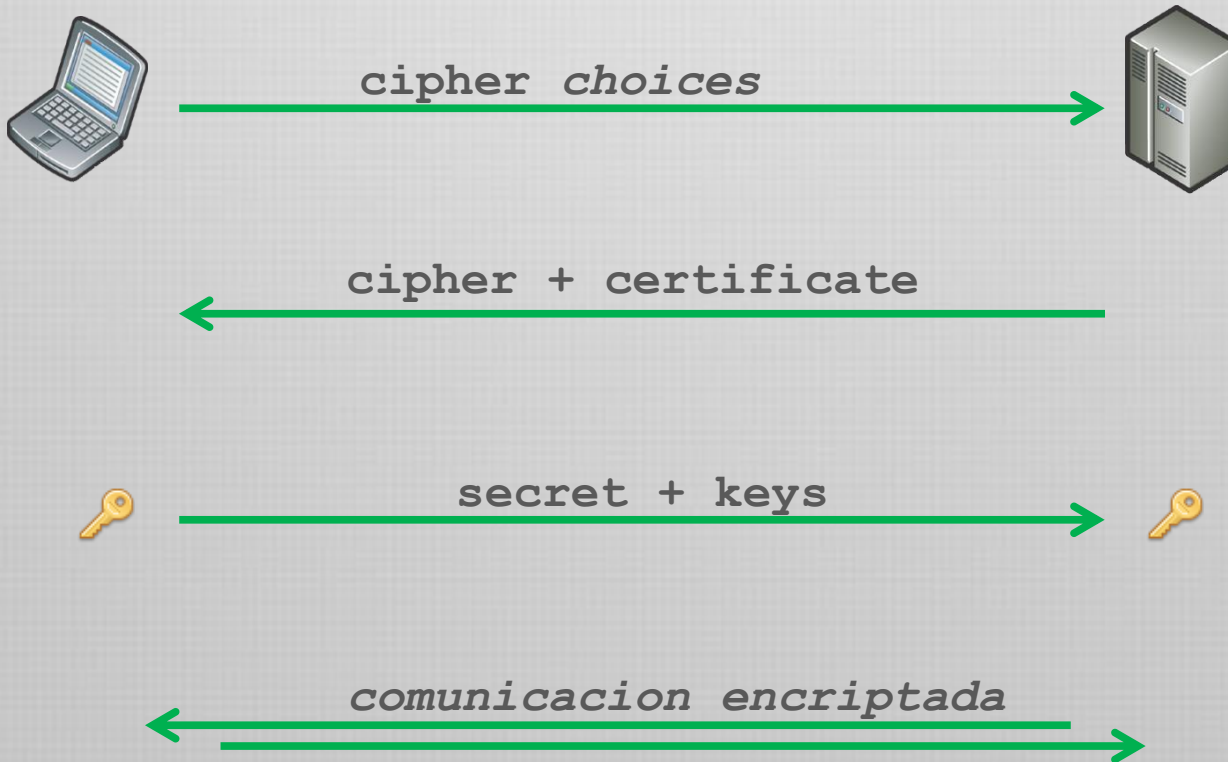
HTTPS - transacción



SSL Handshake

SSL Handshake

- 1) Intercambiar # version de protocolo
- 2) Seleccionar un cifrador conocido por ambos
- 3) Autenticar las identidades de ambos lados
- 4) Generar claves temporarias para encriptar el canal



SSL

El servidor puede demandar certificación al cliente, pero se usa rara vez.

Una intranet puede proveer certificados a sus empleados.

HTTPS requiere siempre certificación del servidor.

*La certificación contiene información sobre el servidor,
expedida por una entidad confiable:*

Nombre

Fecha de expiración

Certificador

Firma digital de la entidad certificada

El navegador automáticamente recupera el certificado de una conexión HTTPS

Si el servidor no tiene certificado, la conexión falla.

Una vez recuperado el certificado, verifica la entidad certificadora.

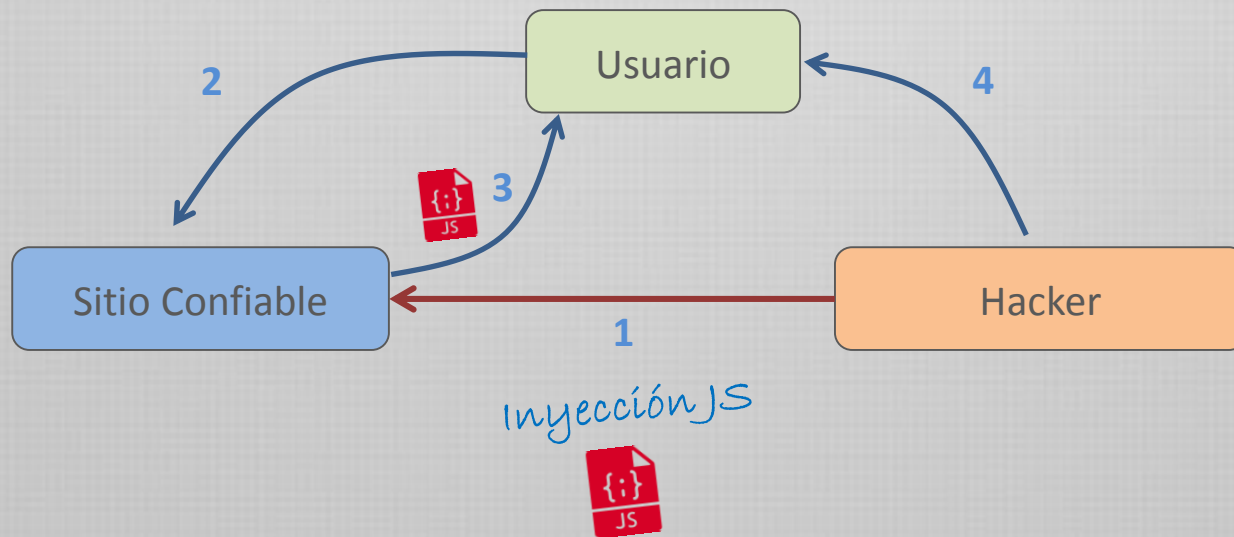
Ataques: Cross-site scripting

Cross-site scripting o XSS, es uno de los ataques más comunes.

Aprovechan la **ausencia de ciertos controles de validación** y depuración de datos ingresados por el usuario, para ejecutar código de un tercer sitio.

El atacante **inyecta código JavaScript** en el navegador.

Este script accede a información nuestra, puede obtener cookies, historiales, etc.



Ataques: Robo de sesiones

La obtención del identificador de sesión de otro usuario es sumamente riesgosa.

Posibilita un robo de identidad frente al servidor.

Existen varias formas:

- **Predicción.**

Básicamente, adivinar el identificador. No es muy viable, ya que son números al azar y muy grandes.

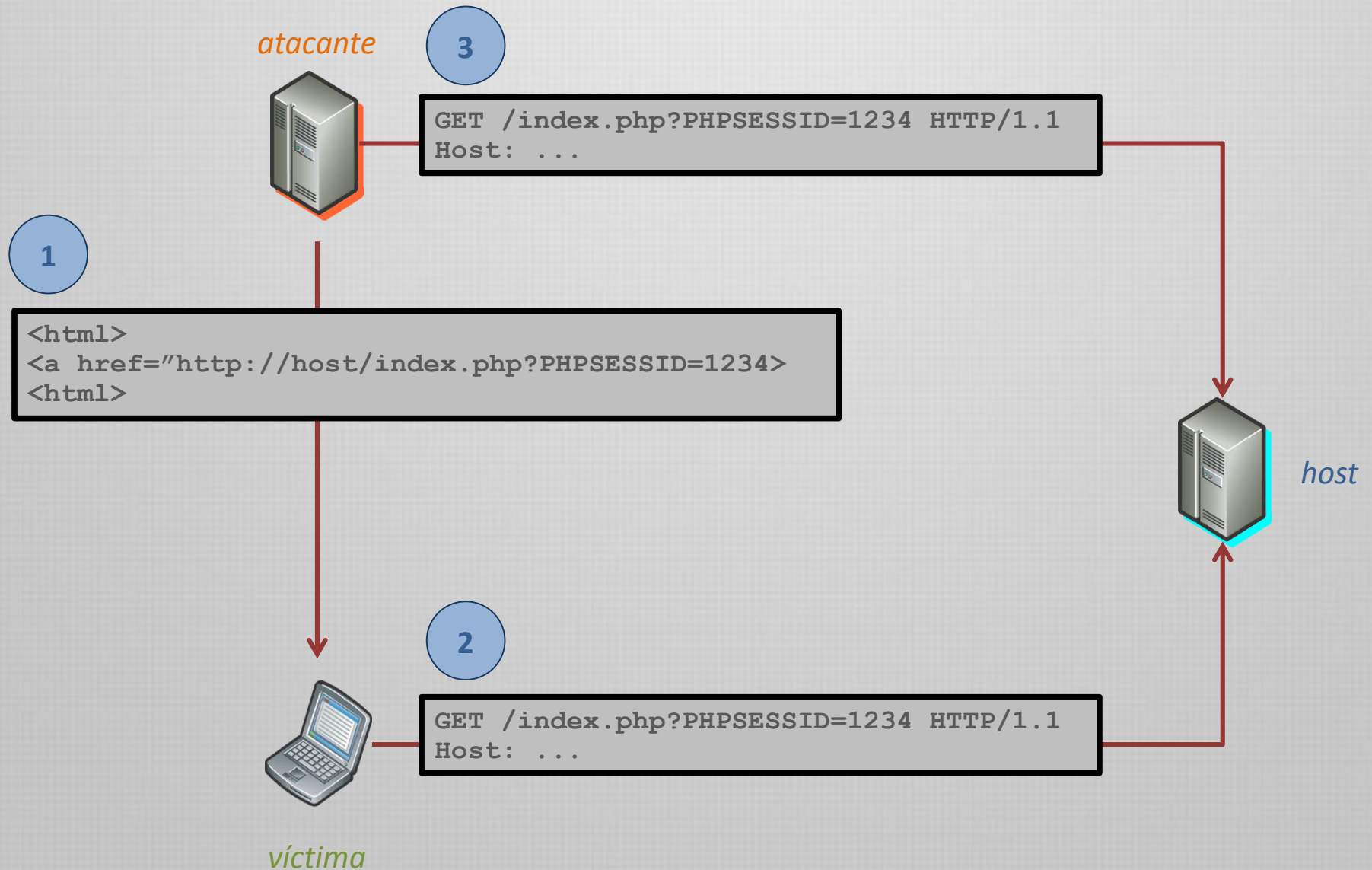
- **Captura.**

El más común. Pueden obtenerse desde las cookies, o desde las variables GET.

- **Fixation.**

Session fixation es la obtención de un número válido de sesión para ser usado con otros propósitos.

Session fixation



Ataques: Bases de datos

La utilización de base de datos debe ser cuidadosa en varios aspectos.

- Evitar exponer información de acceso

Usualmente el nombre de usuario y el password para acceder a la base de datos, son almacenados en un archivo (config.inc).

Si es accesible via web, exponemos los datos anteriores fácilmente.

- Usar encriptación cuando sea necesario

Aún tomando recaudos, las tablas pueden ser consultadas por diferentes usuarios con diferentes permisos (por ejemplo root).

Evitar exponer información sensible que pueda ser leída fácilmente.

- Depurar las consultas SQL para evitar inyecciones de código

Algunos scripts no realizan los controles suficientes al hacer consultas a la base de datos. Esto permite algunas inyecciones de código malignas...

Ataques: Bases de datos

```
<?php
$username=$_POST['username'];
$password=$_POST['pass'];
$email=$_POST['email'];

$sql = "INSERT INTO users (reg_username,reg_password, reg_email)
        VALUES (      '$username',
                      '$password',
                      '$email')";

?>
```

\$sql

```
INSERT INTO users (reg_username,reg_password, reg_email)
VALUES (      'homero',
            'homero123',
            'pepe@snpp.com')
```

Ataques: Bases de datos

```
<?php
$username=$_POST['username'];
$pass=$_POST['pass'];
$email=$_POST['email'];

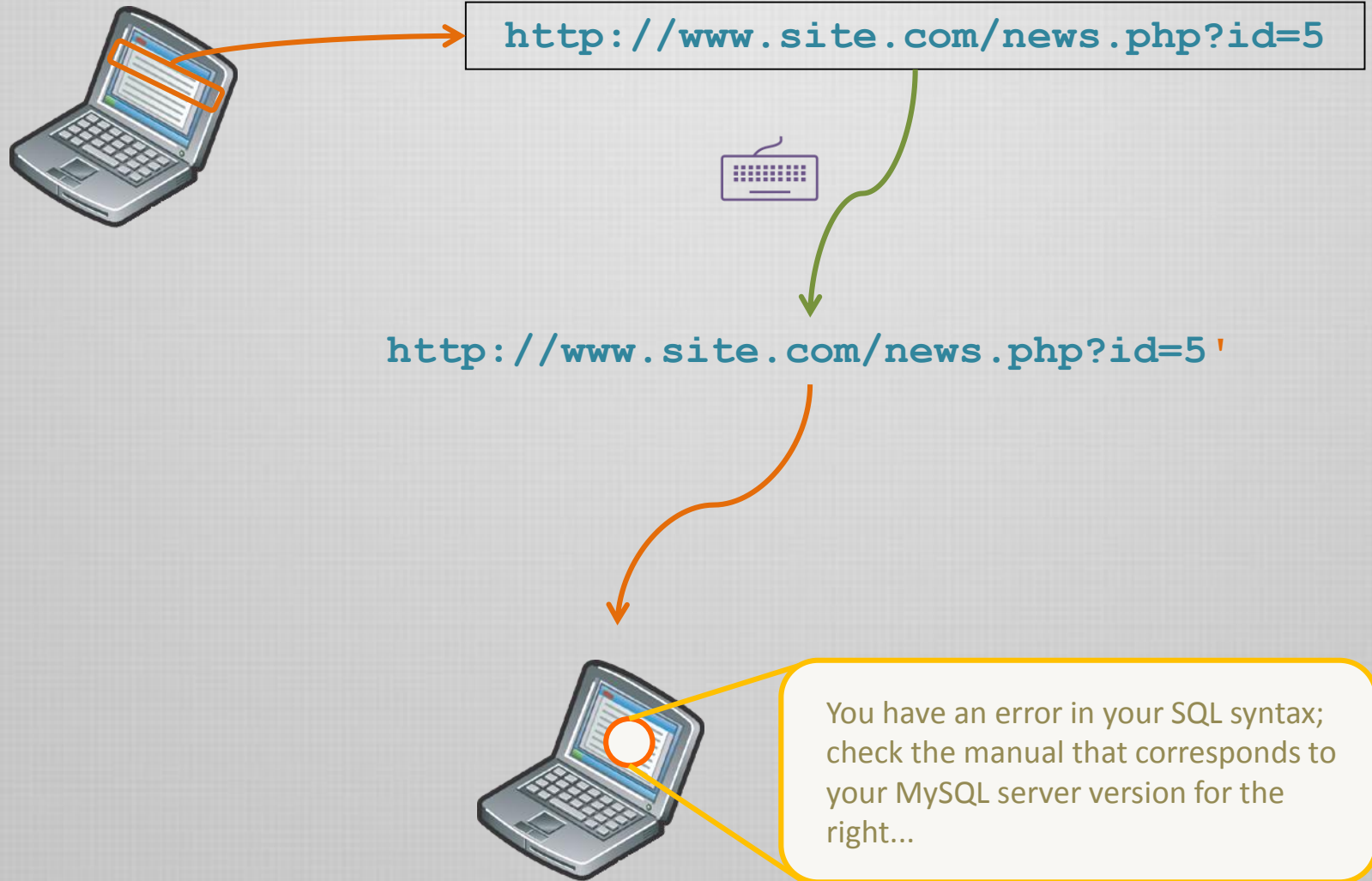
$sql = "INSERT INTO users (reg_username,reg_password, reg_email)
        VALUES (      '$username',
                    '$pass',
                    '$email')";

?>
```

\$sql

```
INSERT INTO users (reg_username,reg_password, reg_email)
VALUES (      'usuariohacker', 'hackpass', ''), ('homero',
        'homero123',
        'pepe@snpp.com')
```

Ejemplo de test



Ejemplo de test

Descubrir cierta información es relativamente fácil...

`http://www.site.com/news.php?id=5 order by 1` 

`http://www.site.com/news.php?id=5 order by 2` 

`http://www.site.com/news.php?id=5 order by 3` 

`http://www.site.com/news.php?id=5 order by 4` 

La tabla tiene 3 columnas

`http://www.site.com/news.php?id=5 union all select 1,2,3` 

Funciona UNION ALL

Reemplazamos uno de los números por otros valores...

`http://www.site.com/news.php?id=5 union all select 1,@@version,3`

Versión de MySQL

Controles en el cliente

No confiar en los controles realizados en el cliente, pues pueden ser forzados.

```
Cookie: rol=guest administrator
```

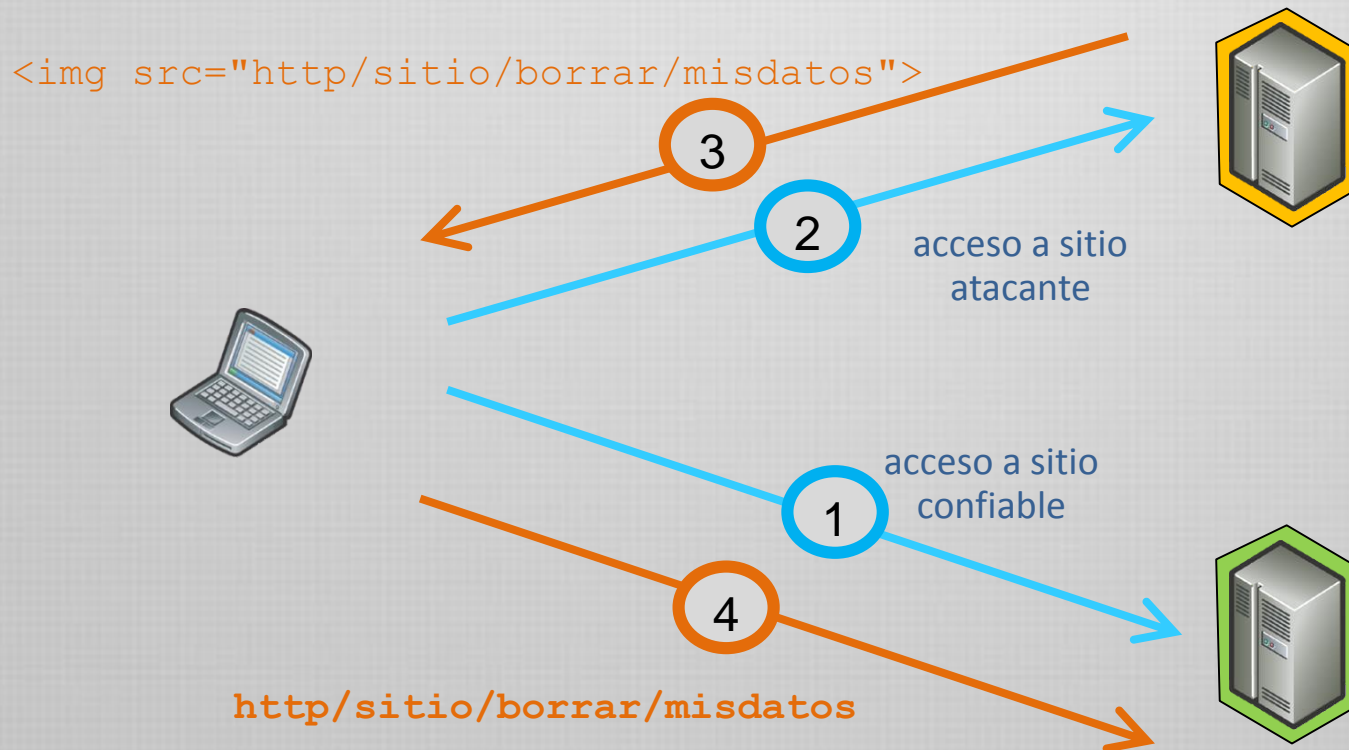
```
<input type="hidden" name="rol" value="guest">  
administrator
```

```
function autorizar(){  
    // controles de acceso return true;  
    ...  
}
```

Se pueden duplicar controles en clientes y servidores

CSRF

Cross-Site Request Forgery es un ataque en el cual se fuerza al cliente a enviar comandos no autorizados a un sitio confiable.



CSRF - contramedidas

Preferir POST antes que GET

Aunque también pueden falsificarse posts

```
<form id="hack" action="http://sitio.com"method="post">
  <input type="hidden" name="id" value="123">
  <input type="hidden" name="accion" value="baja">
</form>
<script>
  var formulario = document.getElementById('hack');
  formulario.submit();
</script>
```

Controlar el valor de Referer

El header Referer menciona la página originadora del request.

Usar tokens en los requests

Cada formulario o link posee un token generado por el servidor.

Controlar la validez del token

```
<input name="token" type="hidden" value="qwesd234lkdn1qk34"
```

Control de datos

Existen librerías y soporte general para el control y sanitización de datos

PHP Filters

- Validación de datos: controla si los datos poseen ciertas calificaciones
- Sanitization: corrige los datos, alterando caracteres no deseables.

```
if (filter_var($email_a, FILTER_VALIDATE_EMAIL)) {  
    // direccion valida  
}
```

```
if (filter_var($ip_a, FILTER_VALIDATE_IP)) {  
    // ip valido  
}
```


Control de datos

Existen librerías y soporte general para el control y sanitización de datos

PHP ESAPI

Librería desarrollada por OWASP: *Open Web Application Security Project*
Gratuita y open source.

```
$ESAPI = new ESAPI(dirname(__FILE__)."/../ESAPI.xml");  
  
$validator = ESAPI::getValidator();  
  
if($validator->isValidInput("test","hello@world.com",  
                           "Email", 100, false)){  
    //entrada valida  
}
```

Autenticación

"Wrong password. Try again"

Este mensaje indica que el nombre de usuario **existe**.
En un ataque de fuerza bruta puede ser un indicador de éxito.



Obligar al usuario a elegir passwords complejos

- *Ofrecer ayuda visual de complejidad*
- *Controlar en el cliente, replicar en el servidor*



Controlar intentos fallidos de acceso

- *Bloquear cuenta luego de determinados intentos.*
- *Ofrecer mecanismos de recuperación de claves.*
- *Informar del último acceso (IP y fecha)*

Forceful browsing

Forceful browsing es un ataque en el que se busca acceder recursos no referenciados por la aplicación en el flujo de la sesión

Predictable Resource Location, File Enumeration, Directory Enumeration, Resource Enumeration

`www.sitio.com/users/calendar.php/user1/20140627`

usuario

fecha

`www.sitio.com/users/calendar.php/user23/20140613`

Otras URL forzadas, muy comunes:

`www.sitio.com /system/
/password/
/logs/
/admin/
/test/`

*Existen herramientas
automatizadas que pueden testear
URL habitualmente vulnerables*

Buscan un 200-OK

HTTP Response Splitting

Este ataque consiste en la **inserción deliberada de contenido** en el *header* del mensaje HTTP.

Debe incluirse un **CRLF** (*Carrier Return – Line Feed*) en el *header*.

El contenido siguiente es en realidad interpretado como el cuerpo del mensaje.

Ocurre usualmente bajo un mensaje de respuesta de redireccionamiento (3xx)

```
HTTP/1.1 302 Moved Temporarily
Date: Fri, 31 Dec 1999 23:59:59 GMT
Location: http://abc/index.php?a=1
Content-Type: text/html
Content-Length: 1354
```

CRLF

```
<html>
<body>...</body>
</html>
```

```
HTTP/1.1 302 Moved Temporarily
Date: Fri, 31 Dec 1999 23:59:59 GMT
Location: http://abc/index.php?a=1
Content-Length: 0
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 23
```

```
<html><body>DANGER</body></html>
```

```
Content-Type: text/html
Content-Length: 1354
```


CRLF

```
<html>
<body>...</body>
</html>
```

XML Security

SSL y la encriptación en capa transporte es sólo una parte de la seguridad.
La aplicación, la red, los datos, deben ser protegidos.

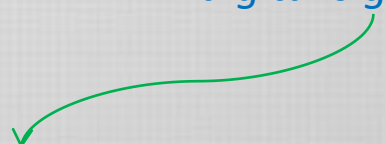
En los servicios web es importante considerar la seguridad cuando se utiliza XML.
Principalmente para proteger la integridad de los datos.



*el dato no ha sido alterado de
su estado original*



digital signatures



```
$sha1hash = sha1_file(archivo.xml);  
$md5hash = md5_file('archivo.xml');
```

Adecuado para la manipulación local.

Si el XML es transferido, la otra parte debería conocer el hash.
No pueden enviarse juntos!

HMAC

HMAC calcula un hash con una clave secreta.

- *Se provee al receptor de la misma clave.*
- *Se puede enviar el documento y el hash al mismo tiempo.*
- *Un interceptor sin la clave secreta no podrá intervenir el mensaje.*

```
if (isset($_POST['xml doc']) && isset($_POST['hmac'])) {  
    $xmldata = base64_decode($_POST['xml doc']);  
    $hmac_shalhash = bin2hex(mhash(MHASH_SHA1, $xmldata, $secret_key));  
  
    if ($hmac_shalhash == $_POST['hmac']) {  
        $dom = new DOMDocument();  
        $dom->loadXML($xmldata);  
        print $dom->saveXML();  
    } else {  
        print 'Datos alterados!!!';  
    }  
}
```

XML encriptado

Si los datos son críticos, deben transportarse encriptados.

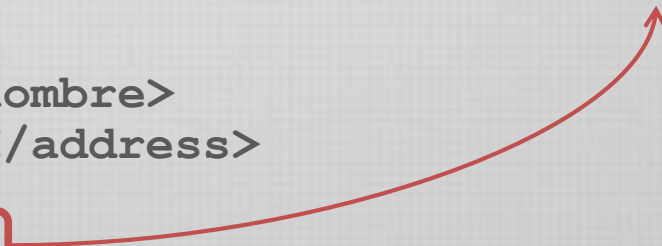
```
<?xml version="1.0"?>
<orden>
  <items>
    <item>
      <id>111</id>
      <cantidad>2</cantidad>
      <precio>9.99</precio>
    </item>
  </items>
<cliente>
  <nombre>Cosme Fulanito</nombre>
  <address>Fake Street 123</address>
</cliente>
<tarjetacredito>
  <numero>1234 1234 123 1234</numero>
  <ccv>123</ccv>
  <exp>0107</exp>
</tarjetacredito>
</orden>
```

XML encriptado

Si los datos son críticos, deben transportarse encriptados.

```
<?xml version="1.0"?>
<orden>
  <items>
    <item>
      <id>111</id>
      <cantidad>2</cantidad>
      <precio>9.99</precio>
    </item>
  </items>
<cliente>
  <nombre>Cosme Fulanito</nombre>
  <address>Fake Street 123</address>
</cliente>
<encrypted iv="5F161c4xjwA=">
Jhm3UYs90vxaOkD6OWfKsaO/zm3G0aCNft/9/57qzmODhz51WC3fL8dxuPze
x1E9aNworn1dn7YFT2bP+WjHUP/qzv0pIQh9vVQ48TlO18Z/Qeh4ffyfVThC
Vpt4esauyhalLSOeqJaE2/GW5sOnEEgqM7p9iHj4
</encrypted>
</orden>
```

vector de inicialización



Canonical XML

La especificación *Canonical XML* (<http://www.w3.org/TR/xml-c14n>) establece cuándo dos documentos son idénticos.

- *El documento debe estar codificado en UTF-8.*
- *Los saltos de línea se normalizan a #xA antes de parsear.*
- *Los valores de los atributos son normalizados.*
- *Los caracteres se reemplazan por entity references*
- *Secciones CDATA son reemplazadas con su contenido.*
- *La declaración XML y el DTD son removidos.*
- *Los elementos vacíos (empty elements) se reemplazan por pares de tags.*
- *Espacios en blanco fuera del documento son normalizados*
- *Se conservan los espacios en blanco en los contenidos*
- *Se aplican comillas dobles a los atributos*
- *Se remueven declaraciones de espacios de nombre superfluos.*
- *Se agregan atributos default a cada elemento*
- *Se aplica un orden lexicográfico a las declaraciones de espacios de nombres y atributos de cada elemento.*

*Dos documentos son idénticos si la representación canónica es la **misma**.*

XML Signature

Las **firmas XML** pueden verificar la integridad de los datos y su fuente de origen.
(<http://www.w3.org/TR/xmlsig-core/>)

La forma más común es utilizando pares de claves públicas y privadas.

Las firmas permiten verificar:

- *la autenticación del originador del documento.*
- *la integridad, al comprobar que no ha sido alterado.*

Existen tres formas de firmas:

- **Enveloped Signatures**

Ubicadas dentro del documento que se está firmando.

```
<mydocument>
  <tag1>contenido1</tag1>
  <tag2>contenido2</tag2>
  <Signature>
    <!-- Firma -->
  </Signature>
</mydocument>
```


XML Signature

- Enveloping Signatures

El dato que se firma *está incluido* en el XML.

```
<mydocument>
  <tag1>contenido1</tag1>
  <Signature>
    <Object Id="mydata">
      <tag2>datos firmados</tag2>
    </Object>
  </Signature>
</mydocument>
```

- Detached Signatures

El dato que se firma *está incluido* en el XML.

```
<mydocument>
  <tag1>contenido1</tag1>
  <Signature>
    <!-- ref al elemento Object -->
  </Signature>
  <Object Id="mydata">
    <tag2>datos firmados</tag2>
  </Object>
</mydocument>
```

XML Signature – *enveloping signature*

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
      Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
    <Reference URI="#object">
      <DigestMethod
        Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
      <DigestValue>nTZuluErIxkl4DgMsBO/E5TiLRA=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>OUubDO2l6XUIODuLSjKAtjYlaTk=</SignatureValue>
  <Object Id="object">Hello World!</Object>
</Signature>
```