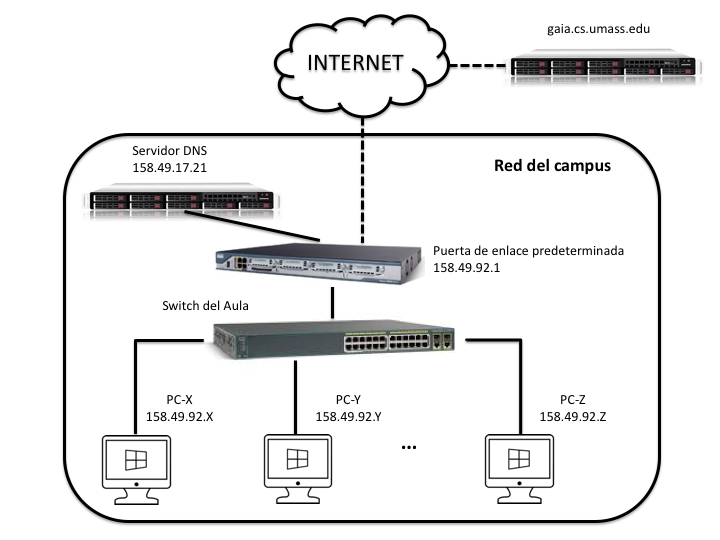
**AL04-1. Investigación de protocolos de capa de Aplicación con Wireshark: HTTP**

*Actividad basada en la traducción del material suplementario del libro Redes de computadores: un enfoque de arriba hacia abajo, 7ª ed, J.F. Kurose y K.W. Ross.*

*“Dime y lo olvido. Muéstrame y recuerdo. Implícame y entiendo“.* Proverbio chino

**Diagrama de topología**

****

**Objetivos**

**Tarea 1. La interacción básica HTTP GET/respuesta.**

**Tarea 2. La interacción HTTP GET CONDICIONAL/respuesta.**

**Tarea 3. Recuperación de documentos Web “largos”.**

**Tarea 4. Los documentos HTML con objetos incrustados.**

**Tarea 5. Autenticación HTTP.**

**Información básica**

El **protocolo de transferencia de hipertexto** (en inglés: **HyperText Transfer Protocol o HTTP**) es el **protocolo de comunicación de la capa de Aplicación** que permite las transferencias de información en la World Wide Web. HTTP fue desarrollado por el World Wide Web Consortium y la Internet Engineering Task Force. El más importante de ellos es el RFC 2616 que especifica la actual versión “1.1”. HTTP define la sintaxis y semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. El cliente o "agente de usuario" (user agent) realiza una petición enviando un mensaje con cierto formato al servidor y el servidor web envía un mensaje de respuesta que contiene habitualmente el código HTML de la página Web solicitada o cualquier otro recurso accesible. Ejemplos de cliente son los navegadores web

El **protocolo de transferencia de hipertexto** **seguro** (en inglés: **HyperText Transfer Protocol Secure**) es más conocido por sus siglas **HTTPS**. Es un **protocolo de capa de Aplicación** basado en el protocolo HTTP y destinado a la transferencia segura de información mediante el cifrado de los datos durante la transferencia Web entre cliente y servidor. HTTPS no es un protocolo independiente, pero requiere el uso de HTTP sobre una **capa de conexión segura cifrada (en inglés: Secure Sockets Layer o SSL)** o una **conexión con seguridad de la capa de transporte (en inglés: Transport Layer Secure o TLS)**.

Recursos necesarios

1 PC con Wireshark conectado a Internet.

# **Tarea 1. La interacción básica HTTP GET/respuesta**

Antes de comenzar, asegúrate de **vaciar la** **caché del navegador web (se recomienda usar Google Chrome)** para que el recurso solicitado (en este caso una página HTML) sea realmente descargado desde el servidor Web**.** Los navegadores Web más utilizados disponen de una caché web de objetos y por lo tanto realizan un **GET condicional** al recuperar un recurso previamente descargado (aprenderás más sobre esto en la **Tarea 2**).

**Cómo vaciar la caché para varios navegadores Web:**

* **Firefox**, clic en el botón **Menú > Preferencias > Privacidad > Contenido web en caché > Limpiar ahora**.
* **Microsoft Edge**, clic en el icono **Hub** (esquina superior derecha) **> Historial > Borrar todo el historial > Elige los tipos de datos que quieres quitar del equipo > Borrar**.
* **Google Chrome**, en la barra de herramientas del navegador, haz clic en **Personaliza y controla Google Chrome> Más herramientas > Borrar datos de navegación > clic en casilla Archivos e imágenes almacenados en caché > Borrar datos de navegación.**

Vamos a comenzar nuestra exploración de HTTP mediante la descarga de un archivo HTML muy simple, corto y que no contiene objetos incrustados. Haz lo siguiente:

* 1. Abre una consola de comandos **CMD** y averigua la **dirección IP** de tu **PC** para la **interfaz Internet** mediante el comando **ipconfig /all**.
  2. Usa esa misma consola para descubrir la **dirección IP** del **servidor** **gaia.cs.umass.edu** de la siguiente forma:

**nslookup gaia.cs.umass.edu**

* 1. Inicia el navegador Web y vacía la caché.
  2. Pon en marcha la captura de paquetes de Wireshark.
  3. Introduce el siguiente filtro de visualización para mostrar sólo los mensajes HTTP dirigidos o provenientes del servidor en la ventana **Lista de paquetes**. Sustituye <dirección\_ip\_servidor> por la obtenida en el punto b.:

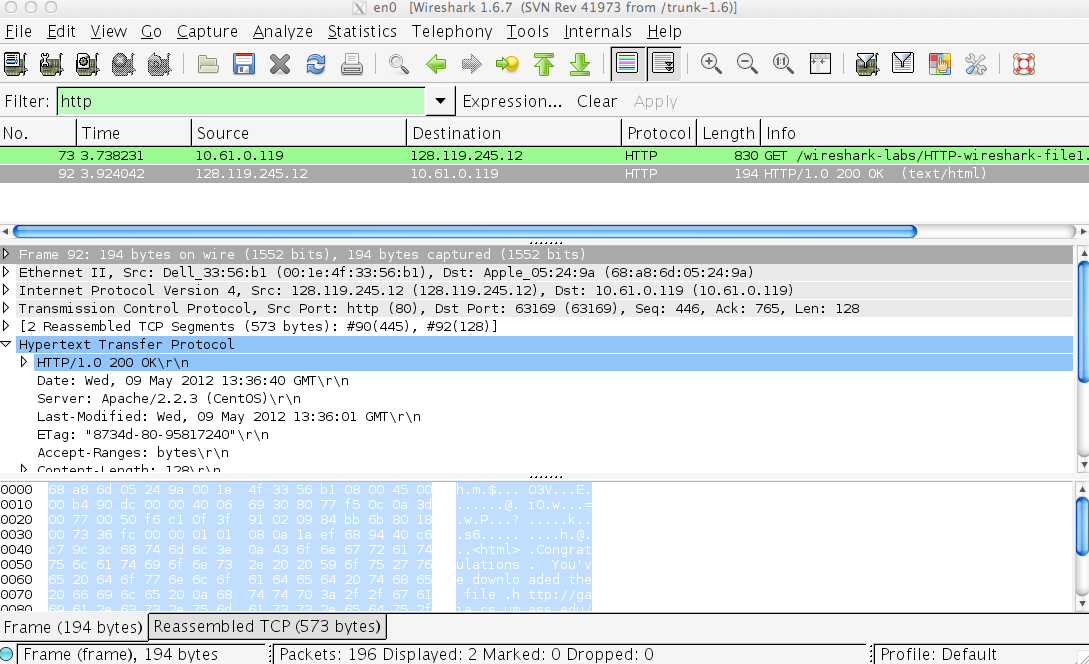
**http&&ip.addr==<dirección\_ip\_servidor>**

* 1. Introduce la siguiente dirección en el navegador Web:

**http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html**

* 1. Detén la captura de paquetes.
  2. Guarda la captura realizada como **AL04\_tarea1.pcap**.

El ejemplo de la figura 1 muestra los dos paquetes que fueron capturados con dos mensajes HTTP: el mensaje con la petición GET desde el navegador del cliente al servidor Web gaia.cs.umass.edu y el mensaje de respuesta correspondiente. La ventana de contenido de paquetes muestra los detalles del mensaje seleccionado en la ventana de lista de paquete de arriba (en este caso el mensaje de respuesta **HTTP/1.0 200 OK**). Recordemos, que los datos enviados desde la **capa de Aplicación** de los hosts serán encapsulados sucesivamente según descienden por las capas del modelo hasta la capa Física. Cada capa incorpora a la PDU de la capa anterior información de control según el protocolo acordado en cada una de ellas. En este ejemplo, el mensaje HTTP se llevó dentro de un segmento TCP, que se llevó dentro de un datagrama IP, que se llevó dentro de una trama Ethernet, y Wireshark también mostrará esta información. En esta actividad sólo nos interesa conocer con detalle la información relativa al protocolo HTTP de la capa de Aplicación. Para el resto de protocolos, la información debe estar colapsada para no interferir en la interpretación de los datos.

**Figura 1:** Ventana con la captura de página Web solicitada por su navegador.

**Nota:** Debe ignorarse cualquier HTTP GET y la respuesta para el archivo **favicon.ico**. Si ves una referencia a este archivo, es que el navegador pide automáticamente al servidor si tiene un archivo de icono para mostrarlo al lado de la URL que aparece en el navegador.

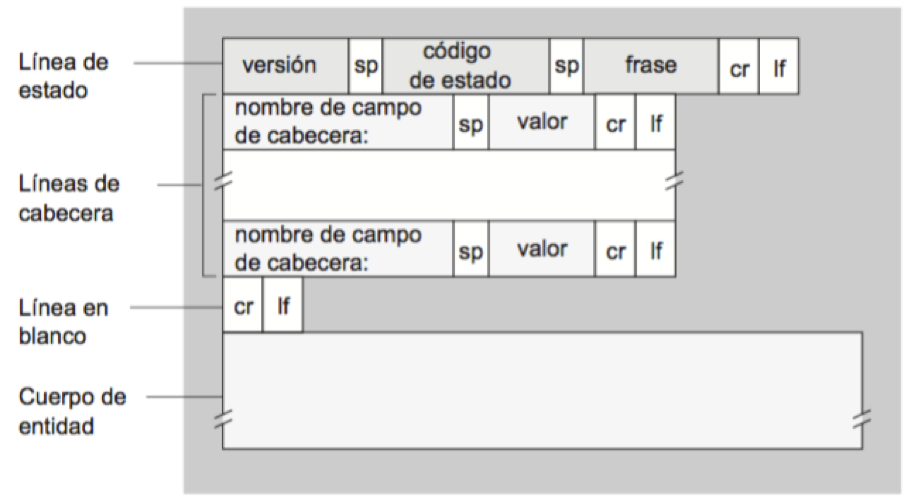
1. ¿Cuál es la dirección IP de tu PC? ¿Y la del servidor gaia.cs.umass.edu? Anótelas a partir de la información de los dos comandos ejecutados más arriba.

Observa la información contenida en los mensajes HTTP GET y respuesta para responder a las siguientes preguntas:

1. ¿Está ejecutando el navegador la versión HTTP/1.0 o HTTP/1.1? \_\_\_\_\_

¿Qué versión de HTTP ejecuta el servidor? \_\_\_\_\_\_ ¿Cuáles son los números de paquete en los que puedes observar esta información? \_\_\_\_

1. ¿Qué idiomas (si los hay) dice su navegador que puede aceptar para las respuestas del servidor? *Observar el valor para el encabezado* ***Accept-Language.***



**Figura 2:** Formato general de un mensaje de respuesta HTTP.

1. ¿Cuál es la línea de estado devuelto desde el servidor a su navegador en la respuesta HTTP? Identifica la versión el protocolo, el código de estado y la frase asociada (véase figura 2).
2. ¿Cuál es la fecha de la última actualización del código HTML del archivo recuperado desde el servidor? *Observar el valor para el encabezado* ***Last-Modified.***

**Nota**: La respuesta a esta pregunta puede sorprenderte porque puedes encontrar que el documento que acabas de recuperar fue modificado por última vez un minuto antes de que fuese descargado. Esto se debe a que el servidor fija automáticamente como última modificación del archivo la fecha/hora actual, y lo está haciendo una vez por minuto. Por lo tanto, si se espera un minuto entre accesos, aparecerá como que el archivo ha sido recientemente modificado, y por tanto su navegador descargará una “nueva” copia del documento.

1. ¿Cuántos bytes de contenido están siendo devueltos a su navegador? *Observar el valor para el encabezado* ***File Date.***

# **Tarea 2. La interacción HTTP GET CONDICIONAL/respuesta**

A continuación, comprobarás cómo cliente y servidor HTTP se comunican para que el navegador Web haga uso de su memoria caché interna. Para ello, se emplea el **método GET CONDICIONAL** que evita descargas innecesarias desde el servidor, disminuye la sobrecarga de datos en la red y reduce el consumo de ancho de banda.

Ahora haz la captura:

* 1. Inicia el navegador Web con la caché vacía tal y como se explicó en la tarea anterior.
  2. Pon en marcha la captura de paquetes de Wireshark.
  3. Vuelve a introducir **http&&ip.addr==<dirección\_ip\_servidor>** como filtro de visualización para que sólo los mensajes HTTP dirigidos o provenientes del servidor se muestren en la ventana Lista de paquetes.
  4. Introduce la siguiente URL en el navegador:

**http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html**

* 1. Vuelve a cargar la misma dirección URL en el navegador (puedes recargarla pulsando la tecla **F5**)
  2. Detén la captura de paquetes.
  3. Guarda la captura realizada como **AL04\_tarea2.pcap**.

Responde las siguientes preguntas:

1. Inspecciona el contenido de la **primera solicitud** HTTP GET desde el navegador al servidor, ¿encuentras una línea de cabecera **If-Modified-Since:** en el HTTP GET? \_\_\_\_\_\_\_\_\_\_\_

¿Cuál es el número de paquete correspondiente? \_\_\_\_\_\_\_\_\_\_\_

1. Inspecciona el contenido de la **respuesta del servidor**. ¿El servidor ha devuelto de manera explícita o cifrada el contenido del archivo? \_\_\_\_\_\_\_\_\_\_\_\_

¿Cuál es el número de paquete dónde puede encontrarse esta información? \_\_\_\_\_\_\_\_\_\_\_\_

1. Ahora inspecciona el contenido de la **segunda petición** HTTP GET desde el navegador al servidor, ¿encuentras una línea de cabecera **If-Modified-Since:** en el HTTP GET? Si es así, ¿qué información completa esa cabecera? Anótala más abajo.

¿Cuál es el número de paquete correspondiente? \_\_\_\_\_\_\_\_\_\_\_

1. ¿Cuál es el código de estado HTTP y la frase devuelta por el servidor en **respuesta a esta segunda HTTP GET**?

¿El servidor ha vuelto a enviar el contenido del archivo? \_\_\_\_\_\_\_\_\_\_\_\_\_ y caso afirmativo ¿de manera explícita o cifrada? \_\_\_\_\_\_\_\_\_\_\_\_

¿Cuál es el número de paquete correspondiente? \_\_\_\_\_\_\_\_\_\_\_

# **Tarea 3. Recuperación de documentos Web “largos”**

En los ejemplos anteriores, los documentos recuperados han sido archivos HTML simples de tamaño pequeño. Vamos a ver lo que sucede cuando descargamos un archivo HTML de mayor tamaño que no puede devolverse en un sólo mensaje de respuesta. Haz lo siguiente:

* 1. Inicia el navegador Web con la caché vacía tal y como se explicó en la tarea 1.
  2. Pon en marcha la captura de paquetes de Wireshark.
  3. Introduce el siguiente filtro de visualización para mostrar los mensajes para los protocolos HTTP o TCP dirigidos o provenientes del servidor en la ventana **Lista de paquetes**.

**(tcp||http)&&ip.addr==<dirección\_ip\_servidor>**

* 1. Introduce la siguiente URL en el navegador. El navegador debe mostrar la Declaración de Derechos de los Estados Unidos de Norte América:

**http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html**

* 1. Detén la captura de paquetes.
  2. Guarda la captura realizada como **AL04\_tarea3.pcap**.

En la ventana Lista de paquetes deberías ver el mensaje HTTP GET, seguida de una respuesta a esta solicitud formada de múltiples segmentos TCP. Esta respuesta en paquetes múltiples requiere de una breve explicación. Recuerda que el **mensaje de respuesta HTTP** consta de una línea de estado, seguido de líneas de cabecera, seguida de una línea en blanco y por último el **cuerpo de entidad (**véase figura 2). El **cuerpo de entidad** de la respuesta debería contener todo el archivo HTML solicitado. Sin embargo, el archivo HTML tiene un tamaño aproximado de **4500 bytes** es demasiado grande para entrar en un sólo segmento TCP (con un tamaño máximo por segmento o **MSS = 1460 bytes**). Por este motivo, el contenido de la respuesta HTTP ha sido dividido (segmentado) por TCP y cada parte del mensaje HTML está contenida dentro de un segmento TCP separado.

Responde las siguientes preguntas:

1. ¿Cuántos mensajes HTTP GET de solicitud envió su navegador para descargar la página? \_\_\_\_\_

¿Cuál es número de paquete correspondiente? \_\_\_\_\_\_\_

1. ¿Cuál es el código de estado y la frase asociada en la respuesta HTTP?

¿Cuál es número de paquete correspondiente? \_\_\_\_\_\_\_\_\_\_

1. ¿Cuántos segmentos TCP son necesarios para “llevar” toda la respuesta HTTP? \_\_\_\_\_\_\_\_\_\_

*Observar la línea justo encima de Hypertext Transfer Protocol en la ventana Detalle del paquete.*

¿Cuáles son los números de paquetes correspondientes? \_\_\_\_\_\_\_\_\_\_\_\_

# **Tarea 4. Los documentos HTML con objetos incrustados**

Ahora que hemos visto cómo Wireshark muestra el tráfico de paquetes capturados para archivos HTML “largos”, podemos ver lo que sucede cuando el navegador descarga un archivo con objetos incrustados, es decir, un archivo que incluye otros objetos (como imágenes o animaciones) que se almacenan en otro servidor (es).

Haz lo siguiente:

* 1. Inicia el navegador Web con la caché vacía tal y como se explicó en la tarea 1.
  2. Poner en marcha la captura de paquetes de Wireshark.
  3. Vuelve a introducir **http&&ip.addr==<dirección\_ip\_servidor>** como filtro de visualización para que sólo los mensajes HTTP dirigidos o provenientes del servidor se muestren en la ventana Lista de paquetes.
  4. Introduce la siguiente URL en el navegador:

**http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html**

**Nota:** El navegador debe mostrar un archivo HTML con dos imágenes. Estas dos imágenes están *vinculadas* dentro del archivo HTML base. Es decir, las imágenes no están contenidas en el propio código HTML; pero sí los hipervínculos o direcciones (URLs) a las mismas. Por tanto, el navegador también tendrá que recuperar esas imágenes a partir de esos hipervínculos. El logo de la editorial se recupera desde el sitio Web **gaia.cs.umass.edu**. La imagen de la portada se almacena en el servidor **manic.cs.umass.edu**.

* 1. Detén la captura de paquetes.
  2. Guarda la captura realizada como **AL04\_tarea4.pcap**.

Responde las siguientes preguntas:

1. ¿Cuántos mensajes HTTP GET de solicitud envió el navegador? \_\_\_\_\_\_

¿Cuáles son los números de paquete correspondientes? \_\_\_\_\_\_\_\_

1. Observa el código HTML de la página devuelto en la primera respuesta. Localiza las siguientes dos líneas:

**<img src=http://gaia.cs.umass.edu/pearson.png WIDTH=”70” HEIGHT=”41”>**

**...**

**<img src=http://manic.cs.umass.edu/pearson.png WIDTH=”70” HEIGHT=”41”>**

¿Cuáles son las direcciones de Internet (IP) asociadas a esos dos servidores? *Mira cómo obtener esta información en la tarea 1:*

¿Corresponden a servidores distintos o son el mismo? \_\_\_\_\_\_\_\_\_

¿Puedes deducir la respuesta anterior de la captura en Wireshark? \_\_\_\_

1. ¿Cuántos mensajes de respuesta se reciben? \_\_\_\_\_ Observa y anota el contenido de las cabeceras **Content-type** en los mensajes de respuesta:

# **Tarea 5. Autenticación HTTP**

Por último, vamos a visitar un sitio Web protegido con contraseña y examinar la secuencia de mensajes HTTP intercambiados. Cuando aparezca el formulario de autenticación, el nombre de usuario es **wireshark-students** y la contraseña es **network**. Haz lo siguiente:

* 1. Inicia el navegador Web con la caché vacía tal y como se explicó en la tarea 1.
  2. Pon en marcha la captura de paquetes de Wireshark.
  3. Vuelve a introducir **http&&ip.addr==<dirección\_ip\_servidor>** como filtro de visualización para que sólo los mensajes HTTP dirigidos o provenientes del servidor se muestren en la ventana Lista de paquetes.
  4. Introduce la siguiente URL en el navegador:

**http://gaia.cs.umass.edu/wireshark-labs/protected\_pages/HTTP-wireshark-file5.html**

* 1. Detén la captura de paquetes.
  2. Guarda la captura realizada como **AL04\_tarea5.pcap**.

Ahora vamos a examinar la captura de Wireshark y responde las siguientes preguntas:

1. ¿Cuál es la respuesta del servidor (código de estado y frase asociada) a la solicitud inicial HTTP GET desde el navegador?

¿Cuál es número de paquete correspondiente? \_\_\_\_\_\_\_

1. Cuando el navegador envía la solicitud GET por segunda vez, ¿qué nuevo campo de cabecera incluye este mensaje? Copia y pégalo a continuación:

¿Cuál es número de paquete correspondiente en la captura? \_\_\_\_\_\_\_

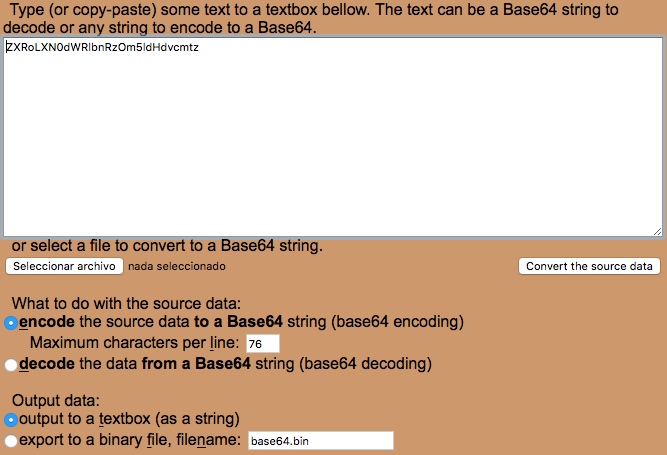
Con el uso de la cabecera **Authorization: Basic XXXXXXXXXXXX** en el mensaje GET HTTP, pudiese parecer que el nombre de usuario y la contraseña se envían cifrados. Sin embargo, simplemente están **codificados** en un formato conocido como **Base64** y, por tanto ¡el nombre de usuario y la contraseña **no están cifrados**!

1. Localiza y observa la información de autenticación enviada por el usuario dentro del segunda petición GET en los diferentes paneles de Wireshark, ¿cuál es la cadena de datos codificada en Base64 enviada al servidor?

**Nota:** Para copiar el valor del campo Authorization directamente desde Wireshark, haz clic con el botón derecho del ratón sobre ese encabezado y selecciona la opción **Copy > Value**

1. Para comprobar lo anterior, puedes ir al sitio Web http://www.motobit.com/util/base64-decoder-encoder.asp y decodificarla mediante los siguientes pasos (véase figura 3):
   1. Pega el valor del encabezado en el campo del formulario disponible en la página. ¡¡OJO!! Borra la palabra “Basic” que aparece al principio de la cadena.
   2. Selecciona la opción “**decode** the data **from a Base64** string (base64 decoding)”.
   3. Haz clic en el botón **Convert the source data**.
   4. El nombre de usuario y contraseña decodificados aparecerán en código ASCII separados por un signo “:”.

**Nota:** Dado que cualquiera puede descargar una herramienta como Wireshark, husmear los paquetes (y no sólo los propios) que pasan por su adaptador de red y traducir de Base64 a ASCII, debería quedar claro que introducir credenciales en sitios Web no es seguro si no se toman medidas adicionales (como el uso de certificados SSL de servidor para HTTPS).



**Figura 3:** Decodificación de una cadena de texto de Base64 a ASCII.