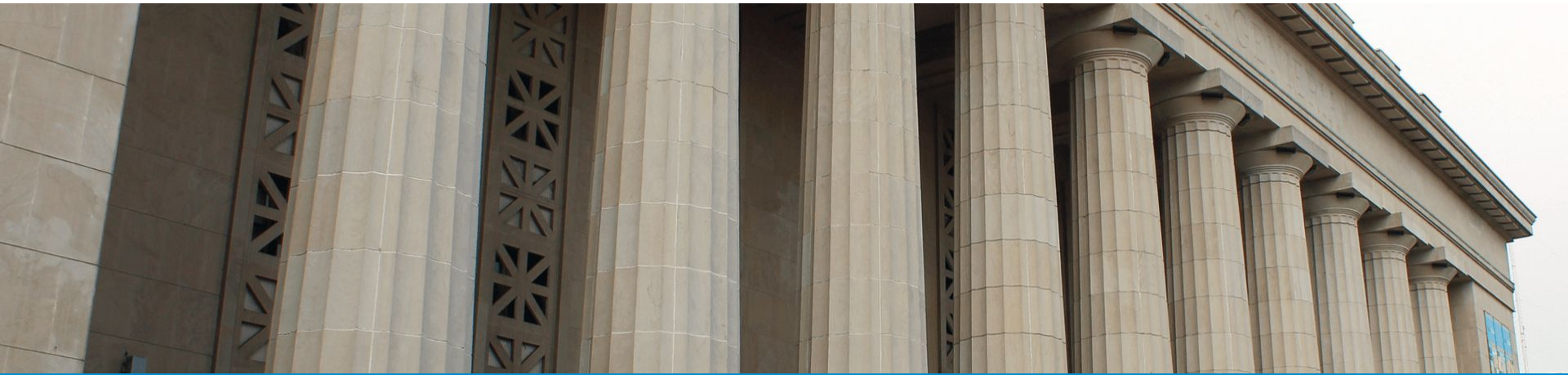
Laboratorio
Capa de Aplicación

86.12/TB067 Redes de Comunicaciones



Requerimientos de conectividad, hardware y software

Para la realización del laboratorio, se necesita una PC conectada a la Internet pública y las aplicaciones listadas a continuación. Verifiquemos si ya se encuentran instaladas y de no estarlo, procedamos a su instalación.

- Wireshark (analizador de protocolo)
  https://www.wireshark.org/download.html

- Python 3.x (lenguaje de programación)
  https://www.python.org/downloads/

- Git (control de versiones)
  https://git-scm.com/download/win

- Editor de texto, a elección. Por ejemplo Notepad++

Instalación y verificación de Wireshark

En el instalación, usar todos los valores preconfigurados (Next/Siguiente).

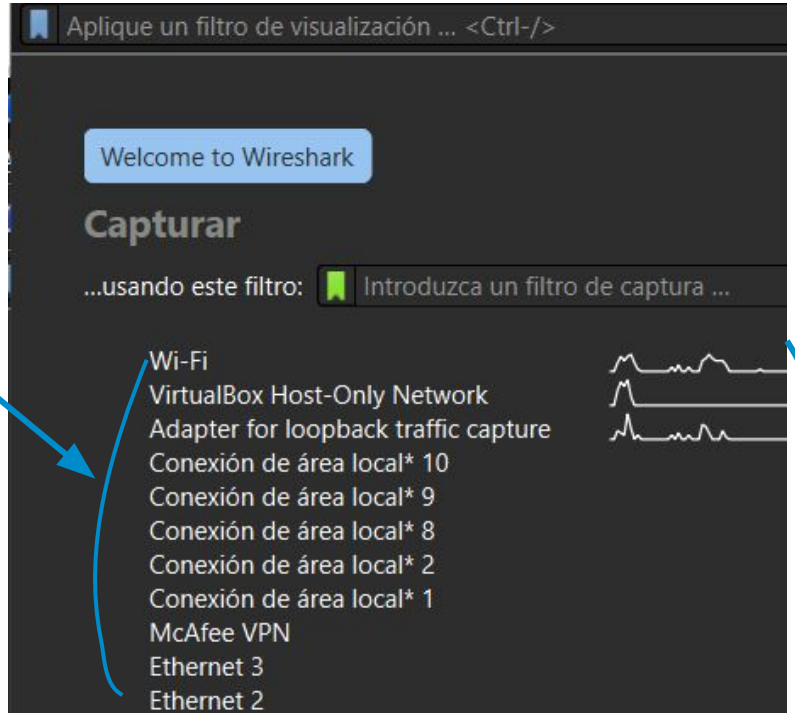
Junto con Wireshark se instala otra aplicación (Npcap), aceptar el acuerdo de licencia (I agree/Acepto) y también aceptar los valores preconfigurados (Next/Siguiente).

Finalizada la instalación, ejecutar Wireshark.

La vista debería ser similar a la siguiente captura:

Instalación y verificación de Wireshark

Listado de todas las interfaces de la PC en particular



Actividad de red en algunas interfaces

Instalación y verificación de Python

Desde consola o línea de comando, puede verificarse si Python está instalado, tipeando

```
python --version
```

De estar instalado se imprimirá la versión, por ejemplo:

```
Python 3.12.3
```

De lo contrario habrá que instalarlo tal como se describe a continuación.

Instalación y verificación de Python

Para instalar Python, descargar el instalador y ejecutarlo. Marcar la opción Add python.exe to PATH y luego Install Now para que use la configuración preestablecida

Instalación y verificación de Python (Ubuntu/Debian)

Para instalar Python en Ubuntu,

```
$ sudo apt-get update
$ sudo apt-get install python3.12
```

Instalación de Git

Para instalar Git, primero verifiquemos si está instalado previamente. Desde línea de comando, ejecutar

```
git -- version
```

Si devuelve versión, es porque está instalado, de lo contrario, descargar el instalador desde https://git-scm.com/downloads (Standalone installer 64 bits)
Ejecutar el instalador usando todos los valores sugeridos (Next/Siguiente)

Instalación de Git (Ubuntu/Debian)

Para instalar Git en Ubuntu o Debian, en caso en que no esté instalado:

```
sudo apt update
sudo apt install git
```

Creación de ambiente de trabajo para el laboratorio

Abrir una ventana de “Símbolo de sistema” (en adelante nos referiremos simplemente a “cmd”).

Crearemos una variable conteniendo un directorio en donde almacenar todos los archivos que usaremos. Aquí usamos el directorio “C:\labo-apps” a modo de ejemplo, pero puede elegirse otro.

```
set DIR_LABO=”C:\labo-apps”
```

Crear un directorio en donde guardaremos los archivos de trabajo e ingresar al directorio

```
mkdir %DIR_LABO%
cd %DIR_LABO%
```

Servidor HTTP “casero”

Ya tenemos el software necesario y espacio de trabajo. Estamos listos para realizar las actividades del laboratorio.

- Todos hemos oído hablar de servidores web o “web servers”. En esta primera práctica vamos a implementar y ejecutar uno en la PC.

- No es un curso de programación, ni siquiera de programación en redes.

  El propósito: aprender que las aplicaciones de red pueden desarrollarse, ejecutarse y probarse en una PC de escritorio. Y no solo “clientes” sino también “servidores”.

Servidor HTTP “casero”

Ingresar a Chat GPT y solicitar lo siguiente:

escribir un servidor web simple en Python que escuche en todas las interfaces y devuelva “Hola, Laboratorio!!!” seguido de una afectuosa bienvenida

Luego de unos segundos de generar texto, el primer cuadro de código debería contener una aplicación funcional. El código generado puede diferir levemente entre distintas PCs, pero en general debería ser adecuado para nuestro propósito.
Copiar el código usando el botón “Copiar código”.

Servidor HTTP “casero”

Usando un editor de texto, pegar el portapapeles en un archivo nuevo. Grabar el archivo en el directorio de trabajo con nombre “webserver1.py”. (*)

Desde la ventana “cmd” en el directorio de trabajo donde se grabó el script ejecutar:

```
python webserver1.py
```

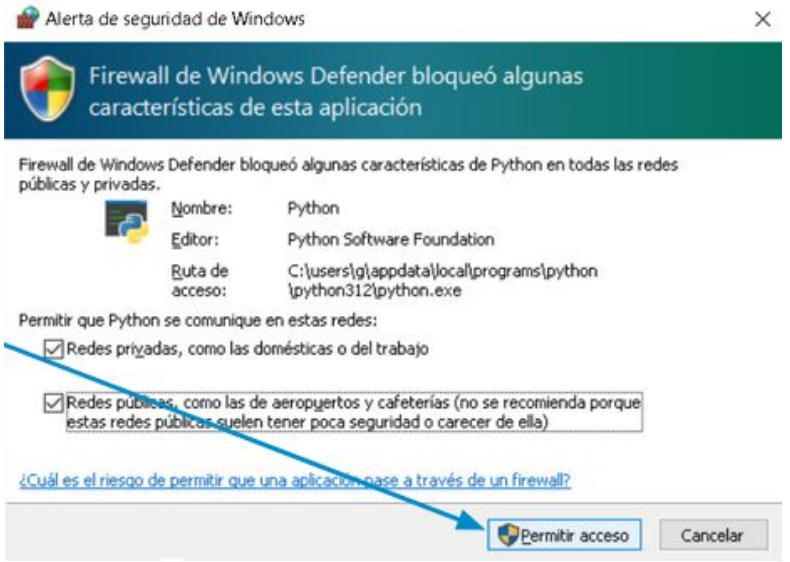
Dependiendo de la configuración de cada PC, es posible que en ese momento se abra una ventana de alerta del Firewall. Veamos de qué se trata:

(*) Es posible que haya que evitar los signos no ASCII de los textos, como “¡” y “ñ”

Firewall

Un firewall es una aplicación destinada a cuidar la ciberseguridad de las redes y sistemas.

En este caso, nos alerta que una aplicación busca crear un **servidor** (desde el punto de vista de TCP/IP) lo cual podría interpretarse como una brecha de seguridad proveniente de software malicioso.



Sin embargo, como el script que “creamos” se trata de un software conocido por nosotros podemos hacer click en Permitir acceso.

Si en cambio hiciéramos clic en Cancelar, la aplicación quedaría bloqueada para acceder a la función de red para la que fue diseñada (servir páginas web).

Ejecución del primer servidor

Una vez que servidor web esté en ejecución, desde la misma PC ingresar a un browser (Mozilla, Chrome, Edge, Ópera, etcétera) y tipear http://localhost:8000/ (u 8080, dependiendo del código generado)

Magia, armamos nuestro propio servidor web!
Realicemos a continuación algunas pruebas…

Algunas pruebas sobre este primer servidor

Algunas pruebas y discusión:

- Qué ocurre si en vez de localhost ponemos como direcciones 127.0.0.1, 127.1.2.3, o la dirección IP de interfaz (Wi-Fi o Ethernet)?
  Para averiguar la(s) dirección8(es) IP, desde “cmd” tipear
  ipconfig (Windows)
  ip address (Linux)

- Qué ocurre si omitimos poner los dos puntos y el 8000? Y si cambiamos 8000 por otro número?
- Qué pasa si escribimos https://localhost:8000/ (con “s”)

Usando Wireshark, analicemos las diferentes transacciones.

Servidor HTTP “casero”

Veamos qué ocurre si queremos ejecutar una segunda instancia de nuestro servidor web. Para ello, sin interrumpir el funcionamiento del servidor en ejecución, abramos otro “cmd” y ejecutemos el script de la misma forma que hicimos anteriormente.

```
python webserver1.py
```

Qué sucede? Leer y analizar lo que el script imprime como resultado.

Servidor HTTP “casero” sobre puerto estándar

Ahora queremos que el servidor web funcione en el puerto HTTP estándar, que es el número 80, en vez el puerto 8000 (o el que haya usado Chat GPT).
Si revisamos el script Python, identifiquemos el puerto y reemplacémoslo por el valor nuevo.
Grabar el archivo con el nombre webserver2.py.
Ejecutar la nueva versión mediante

```
python webserver2.py
```

(Nuevamente, podría ocurrir la alerta del Firewall, ya que estamos usando otro puerto)

Si estamos trabajando en Linux, podríamos necesitar ejecutar el script con sudo.

En este caso, como el puerto es el estándar, no es necesario indicarlo en la petición. Es decir, se puede escribir simplemente http://localhost o usar la dirección IP de una interfaz real.

Servidor HTTP “casero”

Observar que ahora a la izquierda figura la leyenda “No es seguro”, mientras que si usamos una dirección IP de “loopback” no aparece ese mensaje.

En unos minutos buscaremos experimentar con la versión segura del protocolo HTTP

Servidor HTTP “casero” y “público”

Todo servidor web que se precie puede consultarse desde cualquier browser conectado a la red. Con lo cual, podríamos probar conectarnos al servidor web de otras PC!
Tengamos en cuenta que la dirección IP de loopback tiene significado local. Es decir, para poder conectarnos a servidores en otras PCs, deberemos usar direcciones IP de interfaces de red reales (Wi-Fi o Ethernet).

Para simplificar las pruebas, de modo de poder ver quién desde qué cliente se consulta, modifiquemos levemente nuestros servidores, generando una nueva versión a través de Chat GPT. Solicitar lo siguiente:

Escribir un servidor web simple en Python usando http.server. Debe devolver la IP del cliente y la dirección IP del servidor. Debe vincularse a una interfaz de red física y escuchar en el puerto 80

Servidor HTTP “casero” y “público”

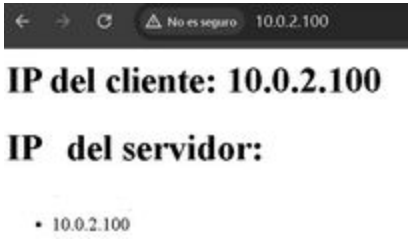
Al igual que antes, copiar el código en un nuevo archivo.

Averiguar la dirección IP de la interfaz física con **ipconfig** (Windows) o **ip address** (Linux). En el script generado, completar con la dirección IP obtenida (seguramente hay un comentario en las primeras líneas del script). Grabar el archivo como webserver3.py

Cerrar la ejecución de los servidores anteriores (cerrando ventana/consola, o Control-C). Abrir una nueva ventana nueva y ejecutar: `python webserver3.py`

Compartir la dirección IP del servidor con los usuarios de las otras estaciones de trabajo. ¿Cómo? Hablando entre ustedes! (¿Cómo se resuelve esto en Internet?)

Desde los browsers, consultar a los diferentes servidores levantados: el propio y los ajenos!



Los clientes web de celular también funcionan! 🤯

Servidor HTTPS "casero" y "seguro"

La próxima experiencia es crear un servidor web seguro. El pedido a Chat GPT es

Escribir un servidor web seguro en Python usando http.server y la clase SSLContext del módulo ssl. Debe devolver las direcciones IP del cliente y del servidor. Debe escuchar una interfaz física en el puerto 443

Guardar el código generado en un nuevo archivo webserver4.py dentro del directorio de trabajo.

Tal como seguramente indique el texto generado, en esta oportunidad es necesario generar una clave privada y un certificado (otra vez, cuestiones de ciberseguridad). Esa información también debe grabarse en el mismo directorio de trabajo.

Servidor HTTPS “casero” y “seguro” (cont.)

Usaremos la aplicación openssl para generar un archivo conteniendo la clave privada y el certificado. Desde “cmd” entrar al directorio donde debería estar instalado openssl:

```
cd “C:\Program Files\Git\mingw64\bin”
```

Ejecutar lo siguiente para generar la clave y el certificado.

```
openssl req -new -x509 -keyout %DIR_LABO%\server.out -out
%DIR_LABO%\server.pem -days 365 -nodes -config "C:\Program
Files\Git\usr\ssl\openssl.cnf"
```

Volver al directorio de trabajo

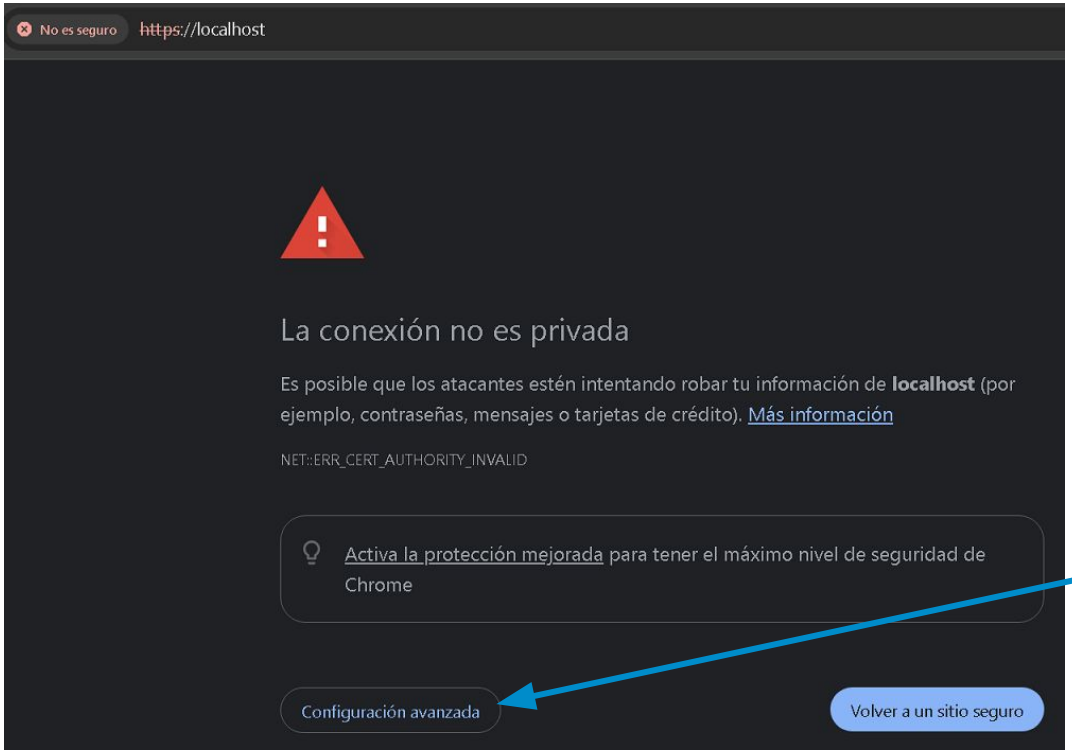
```
cd %DIR_LABO%
```

Verificar que el archivo server.pem se haya generado y ejecutar el webserver:

```
python webserver4.py
```

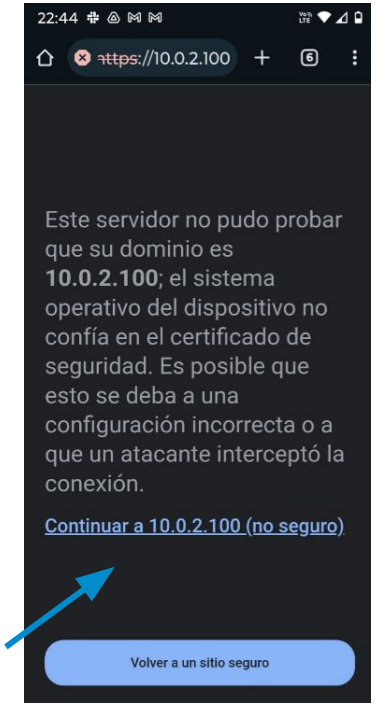
Servidor HTTPS "casero" y "seguro"

Consultar la página "segura" en cualquier browser usando **https://localhost** o usando la IP de interfaz. Qué ocurre?



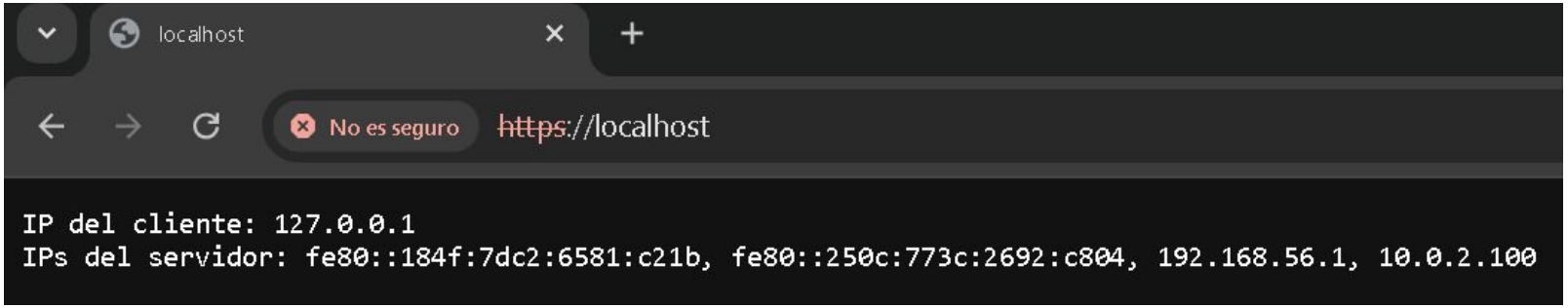
No era que habíamos "desarrollado" un servidor "seguro"??

En Configuración avanzada, aceptar el "riesgo"



Servidor HTTPS “casero” y “seguro”

Una vez que le indicamos al browser aceptar la excepción, se muestra la página web



Servidor HTTP plano (público)

En la actualidad la gran mayoría de los servicios web han migrado a versiones con encriptación y autenticación. En este ejercicio vamos a utilizar un sitio HTTP tradicional, de una dependencia de esta casa de estudios.

- Capturar usando Wireshark la descarga de la página http://www.idiomas.filo.uba.ar/ para analizar su contenido:
- ¿Qué versión de HTTP usa esta descarga?
- ¿Hay conexiones HTTP persistentes?
- ¿Cuánto tiempo demanda descargar la página completa?
- ¿Cuántas conexiones TCP/IP se establecieron?
- Sabiendo que la IP de esa dirección es 157.92.88.10 ¿Qué ocurre en el navegador si en vez de consultar por la URL, se consulta por la IP? Es decir: http://157.92.88.10/

Desafío HTTP: el cliente y las descargas repetidas.

El protocolo HTTP brinda la posibilidad de que navegadores o los servidores proxy detecten si los archivos fueron modificados respecto a un determinado momento. De ese modo, se puede evitar descargas innecesarias, ahorrando ancho de banda.

El desafío que planteamos es que revisen o investiguen cómo ese mecanismo ahora mismo. En vez de navegador, usar telnet, empleando los encabezados necesarios, de modo de demostrar la situación de evitar una descarga ya hecha en el pasado.
Se puede notas de clase, el libro, ChatGPT, cualquier recurso. Como servidor web a consultar, usar por supuesto un servidor sin encriptación, por ejemplo http://www.santafe.gov.ar/

Resolución de nombres de dominio usando servidores root

Esta sección del laboratorio busca indagar acerca del servicio de nombres de dominio (DNS)

- Objetivo: consultar a un servidor de dominio root por la dirección IP correspondiente a www.fi.uba.ar
- Los servidores root están distribuidos en todo el mundo. La zona root está conformada por 13 servidores: a.root-servers.net, b.root-servers.net, c.root-servers.net, …, m.root-servers.net (cada cual vinculado a una IPv4 y a una IPv6).
- La lista completa puede encontrarse en https://www.iana.org/domains/root/servers.
- Por ejemplo, tomemos el servidor g.root-servers.net cuya dirección IPv4 es 192.112.36.4
- Utilizaremos la aplicación nslookup, desde una sesión de consola de texto (“cmd”)

Resolución de nombres de dominio usando servidores root (cont)

- La sintaxis es :
  `nslookup dominio_a_consultar servidor_de_dominio`

- Entonces, tomando como servidor de dominio 192.112.36.4 la consulta que realizaremos es la que figura abajo. Antes de ejecutarla, abramos la aplicación Wireshark e iniciemos la captura en la interfaz correcta.

  `nslookup www.fi.uba.ar 192.112.36.4`

- Analizar el resultado devuelto por la aplicación y la captura en Wireshark.
  (un filtro recomendado para aplicar en Wireshark es dns && ip.addr==192.112.36.4)

Resolución de nombres de dominio usando servidores root (cont)

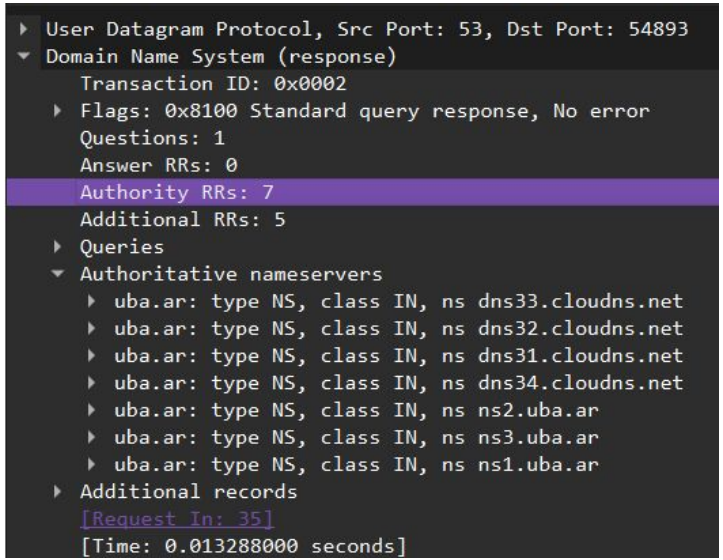
- Como conclusión, podemos ver que los servidores root solamente devuelven registros de TLD (top level domain). En este caso, al consultarle por un dominio terminado en “ar”, nos devolvió todos los servidores TLD que pueden resolver dominios terminados en .ar.

- Repitamos esta consulta, pero usando alguno de los servidores devueltos, por ejemplo c.dns.ar = 200.108.148.50, no sin antes iniciar captura Wireshark, para el análisis.

Resolución de nombres de dominio usando servidores root (cont)

- Análogo a lo que pasó antes, estos servidores TLD solamente devuelven registros de otros servidores de nombre capaces de resolver nombres terminados en “uba.ar”.

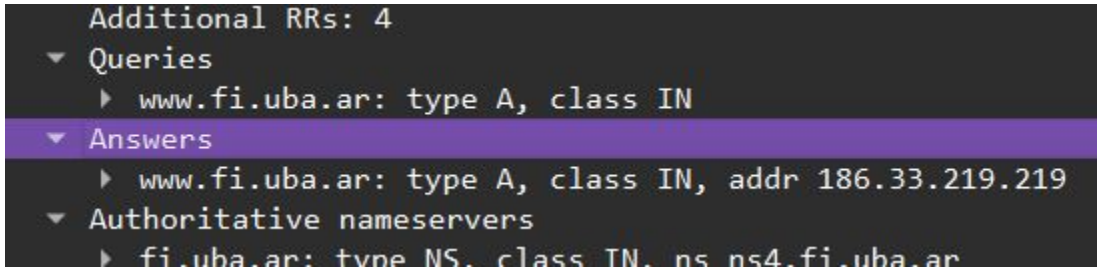
- Si vemos la respuesta Wireshark, los servidores listados son “autoritativos”, esto es, son oficialmente responsables de la información DNS para el dominio consultado.



Resolución de nombres de dominio usando servidores root (cont)

- Iteramos una vez más para obtener finalmente la dirección IP de dominio que buscamos (www.fi.uba.ar), tomando algún servidor devuelto en la consulta anterior, por ejemplo ns1.uba.ar=157.92.1.1. Recordar iniciar captura Wireshark para el análisis
- nslookup www.fi.uba.ar 157.92.1.1

- Observar que finalmente tenemos un registro A, diciéndonos que la IP buscada es 186.33.219.219



Gracias