



Instituto Politécnico Nacional

IPN

Escuela Superior de Cómputo

ESCOM

Academia de Redes Aplicaciones para comunicaciones
en red

6CV2

Práctica 13

“Ejemplo Protocolo DNS”

Alumna:

Navarrete Becerril Sharon Anette

Fecha de entrega: “ 25 - Noviembre - 2024”

Profesor: Ojeda Santillan Rodrigo

OBJETIVO

Desarrollar un sistema para manipular y procesar registros DNS, permitiendo la identificación, configuración y presentación de diversos tipos de registros DNS en un formato comprensible, con el fin de facilitar la interpretación de respuestas DNS y su utilidad en el análisis de red y diagnóstico de problemas.

INTRODUCCIÓN

El Sistema de Nombres de Dominio (DNS) es una parte fundamental de la infraestructura de internet, encargada de traducir nombres de dominio en direcciones IP para permitir la conexión entre dispositivos. Cada vez que un usuario intenta acceder a un sitio web, como "www.ejemplo.com", su dispositivo envía una solicitud DNS para obtener la dirección IP correspondiente, estableciendo así la conexión necesaria.

Este ejemplo se centra en el procesamiento de registros DNS, que son las entradas específicas en la base de datos DNS. Existen varios tipos de registros, cada uno con su propio propósito: el tipo "A" asocia un dominio con una dirección IPv4, "CNAME" crea un alias para un dominio, y "MX" se relaciona con los servidores de correo electrónico del dominio. Mediante funciones especializadas, como `imprimir_tipo_registro`, el sistema identifica y muestra en pantalla el tipo de cada registro, siendo una herramienta útil en aplicaciones de clientes DNS y herramientas de diagnóstico de red.

Además, el código incluye procedimientos clave para el flujo de procesamiento de direcciones DNS: configuración del servidor y cliente, manejo de errores, preparación de encabezados y paquetes de consulta, y procesamiento de respuestas. Este flujo, que sigue una estructura de cinco pasos, asegura que las solicitudes DNS se configuren, envíen y procesen correctamente, proporcionando un recurso valioso para el análisis de tráfico y respuesta DNS.

DESARROLLO

El flujo principal del código DNS Proxy sigue una estructura de cinco pasos que aseguran la correcta configuración, solicitud, y procesamiento de consultas DNS.

Primero, se inicia abriendo un socket de red, que actúa como punto de comunicación para enviar y recibir datos entre el cliente y el servidor DNS. Este socket se configura tanto para el cliente (quien realiza la solicitud) como para el servidor DNS (que maneja la consulta), definiendo las direcciones IP y puertos adecuados. Por lo general, se utiliza el protocolo UDP, el estándar para solicitudes DNS, lo cual permite una transmisión rápida y eficiente de datos.

En el siguiente paso, el sistema solicita al usuario la dirección web que desea consultar. Esta dirección se ingresa en el sistema y se utiliza para preparar un paquete DNS que incluye el encabezado de consulta y el dominio solicitado. Este encabezado contiene los detalles específicos de la consulta,

como el tipo de registro DNS (por ejemplo, tipo "A" para una dirección IP), y es crucial que esté en el formato adecuado para que el servidor pueda interpretarlo correctamente.

Una vez que el paquete DNS está listo, se envía al servidor DNS a través del socket configurado, y el sistema entra en un estado de espera hasta recibir la respuesta del servidor. Este servidor puede ser un DNS público, como el de Google (8.8.8.8), o un DNS interno de la red. La respuesta recibida contendrá la dirección IP asociada al dominio solicitado o, en su defecto, un mensaje de error si el dominio no puede resolverse.

Cuando la respuesta es recibida, el código la procesa extrayendo la información relevante, como la dirección IP del dominio, el tipo de registro o el estado de la consulta. Esta información se imprime en la consola, permitiendo al usuario visualizar los resultados. Si el dominio no existe, se muestra un mensaje de error indicando que el dominio no es resoluble.

Finalmente, se establece la lógica completa del código para garantizar un flujo de trabajo continuo y robusto. Esto puede incluir configuraciones adicionales como filtros de bloqueo (listas negras) para ciertos dominios o ajustes para manipular las respuestas y redirigir al usuario en función de políticas de red específicas. Además, se integran funciones de manejo de errores para permitir que el sistema se recupere de fallas en la red, asegurando que el servidor DNS proxy continúe operando sin interrupciones.

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <ctype.h>

const unsigned char INDICADORES[2] = {0x01, 0x00};
const unsigned char ID_TRANS[2] = {0x02, 0x0a}; /* ID de la transaccion */
const unsigned char PETICIONES[2] = {0x00, 0x01}; /* Contador de Peticiones (1)*/
const unsigned char CERO[1] = {0x00};
```

```

const unsigned char SOLICITUD[1] = {0x00};      /* Solicitud al DNS Host */
const unsigned char CODIGO_SOLICITUD[4] = {0x00, 0x00, 0x00, 0x00}; /*Codigo de
operacion de solicitud al DNS */
const unsigned char RECURSION[1] = {0x01};

const unsigned char REG_HOST[2] = {0x00, 0x01}; /* Registro Host */
const unsigned char CLASE_INT[2] = {0x00, 0x01}; /* Clase Internet */

struct sockaddr_in configurar_servidor(struct sockaddr_in);
struct sockaddr_in configurar_cliente(struct sockaddr_in);
void manejo_errores(int, int);
void configurar_encabezado(unsigned char[12]);
void pedir_direccion(unsigned char[50]);
int configurar_buffer(unsigned char[12], unsigned char[50], unsigned char[512]);
void respuesta_dns(unsigned char[512]);
void imprimir_tipo_registro(int);
void imprimir_clase();
void imprimir_tiempo(unsigned char[512], int);
int es_apuntador(unsigned char buffer[512], int p);
int imprimir_n_caracteres(unsigned char buffer[512], int num, int apuntador);
int ciclo_impresion(unsigned char buffer[512], int pointer);
int guardar_apuntador(unsigned char buffer[512], int point, int apuntador);
int imprimir_ip(unsigned char buffer[512], int pointer);
int recorrer_trama(unsigned char buffer[512], int apuntador);
int imprimir_ipv6(unsigned char buffer[512], int apuntador);

void main()
{
    int sockfd;          /* Socket */

    int longFinal; /* Almacenaremos la longitud final de la solicitud */

```

```

    int tam;          /* Variable que usaremos para guardar posibles errores -1 */

    unsigned char buffer[512]; /* Buffer para almacenar paquetes recibidos */

    unsigned char encabezado[48]; /* Encabezado DNS */

unsigned char direccion[50]; /* Arreglo donde almacenaremos la direccion Web*/

    unsigned char direc[50]="www.youtube.com";

unsigned char di[50]="youtube.com";

struct sockaddr_in cliente; /* Estructura que usaremos para configurar el
cliente*/

struct sockaddr_in servidor; /* Estructura que usaremos para configurar el
servidor */

    memset(buffer, 0, 512); /* Limpiamos la cadena para evitar posibles errores
*/memset(direccion, 0, 50); /* Limpiamos la cadena para evitar posibles errores*/

    /* Abrimos el socket */ sockfd = socket(AF_INET, SOCK_DGRAM, 0);

manejo_errores(0, sockfd); /*Verificamos que no haya errores al abrir el socket */


    /* Configuramos el cliente, familia, puerto, etc. */

    cliente = configurar_cliente(cliente);


    /* Asignamos direccion al socket */

    tam = bind(sockfd, (struct sockaddr *)&cliente, sizeof(cliente));

    manejo_errores(1, tam); /* Verificamos que no haya errores en el bind */


    /* Configuramos el servidor, familia, puerto, IP, etc. */

    servidor = configurar_servidor(servidor);

    /* Le pedimos al usuario que nos de la direccion Web */

    pedir_direccion(direccion);


if(strcmp(direccion,direc)==0){

printf("BIENVENIDO A TU DNS");

direc[strlen(direc) - 1] = '\0';

```

```

/* Configuramos el encabezado con id de transaccion, identificadores,
peticiones, etc */

    configurar_encabezado(encabezado);

/* Configuramos el buffer a enviar, colocamos el encabezado, entradas de
solicitud, etc. recibimos la longitud final de la solicitud */

    longFinal = configurar_buffer(encabezado,di, buffer);

/* Enviamos la solicitud al DNS */
tam = sendto(sockfd, buffer, longFinal, 0, (struct sockaddr *)&servidor,
sizeof(servidor));

    manejo_errores(3, tam); /* Verificamos que no hubo errores al enviar */
    perror("\nExitó al enviar la solicitud");

    int len = sizeof(servidor); /* Es un apuntador a la estructura servidor */
    /* Recibimos la respuesta del DNS */
tam = recvfrom(sockfd, buffer, 512, 0, (struct sockaddr *)&servidor, &len);

    manejo_errores(2, tam); /* Verificamos que no hubo errores al recibir */
    perror("\nExitó al recibir la respuesta");

/* Identificamos los datos recibidos en la respuesta del DNS */
respuesta_dns(buffer);

/* Cerramos el socket */    close(sockfd);

/* Metemos un salto de líneas por pura estética */    printf("\n"); }

/* Configuramos el encabezado con id de transaccion, identificadores, peticiones,
etc */

    configurar_encabezado(encabezado);

/* Configuramos el buffer a enviar, colocamos el encabezado, entradas de
solicitud, etc. recibimos la longitud final de la solicitud */

    longFinal = configurar_buffer(encabezado, direccion, buffer);

```

```

        /* Enviamos la solicitud al DNS */ tam = sendto(sockfd, buffer, longFinal,
0, (struct sockaddr *)&servidor, sizeof(servidor));

        manejo_errores(3, tam); /* Verificamos que no hubo errores al enviar */

        perror("\nExito al enviar la solicitud");

        int len = sizeof(servidor); /* Es un apuntador a la estructura servidor */

        /* Recibimos la respuesta del DNS */
tam = recvfrom(sockfd, buffer, 512, 0, (struct sockaddr *)&servidor, &len);

        manejo_errores(2, tam); /* Verificamos que no hubo errores al recibir */

        perror("\nExito al recibir la respuesta");


        /* Identificamos los datos recibidos en la respuesta del DNS */

        respuesta_dns(buffer);

        /* Cerramos el socket */ close(sockfd);

        /* Metemos un salto de lines por pura estetica */ printf("\n");
}

/* Configuramos el servidor */
struct sockaddr_in configurar_servidor(struct sockaddr_in servidor)
{
    servidor.sin_family = AF_INET;          /* Familia: AF_INET */

    servidor.sin_port = htons(53);          /* Puerto del DNS */

    servidor.sin_addr.s_addr = inet_addr("148.204.103.2"); /* IP del Poli */

    //servidor.sin_addr.s_addr = inet_addr("8.8.4.4"); //IP DE GOOGLE

    return servidor; }

/* Configuramos el cliente */
struct sockaddr_in configurar_cliente(struct sockaddr_in cliente)
{ cliente.sin_family = AF_INET; /* Familia: AF_INET */

    cliente.sin_port = htons(0); /* Colocamos puerto efimero asi */

    cliente.sin_addr.s_addr = INADDR_ANY;

    return cliente; }

```

```

/* Manejamos los errores -1 en tam */
void manejo_errores(int codigo, int num)
{
    switch (codigo)
    {
        case 0:
            if (num == -1)
            {
                perror("\nError al abrir el socket");
                exit(0); }

            perror("\nExito al abrir el socket");
            break;

        case 1:
            if (num == -1)
            {
                perror("\nError en el bind");
                exit(0); }

            perror("\nExito en el bind");
            break;

        case 2:
            if (num == -1)
            {
                perror("\nError al recibir");
                exit(0); }

            break;

        case 3:
            if (num == -1)
            {
                perror("\nError al enviar");
                exit(0);
            }
            break;
    }
}

/* Configuramos el encabezado */
void configurar_encabezado(unsigned char encabezado[12])
{
    memset(encabezado, 0, 12); /* Colocamos todo en ceros */

    memcpy(encabezado + 0, ID_TRANS, 2); /* Colocamos el ID de transaccion

```

```

en el encabezado */

memcpy(encabezado + 2, INDICADORES, 2); /* Colocamos los indicadores para una
solicitud */

memcpy(encabezado + 4, PETICIONES, 2); /* Colocamos el contador de peticiones en
1 */

}

int configurar_buffer(unsigned char encabezado[12], unsigned char direccion[50],
unsigned char buffer[512])
{
    int apuntador = 12;          /* Creamos un contador apuntado iniciado en 12
que es la posicon actual del buffer */

    unsigned char longCadena[2]; /* Almacenaremos el valor devuelto por strlen */
    memcpy(buffer + 0, encabezado, 12); /* Colocamos el encabezado en el buffer */

    /* Separamos la cadena por cada . en ella */

    unsigned char *token = strtok(direccion, ".");

    /* Si queremos que la función siga dividiendo nuestra cadena; en las
siguientes llamadas a la misma no debemos
pasarle la cadena (sino NULL), y también los delimitadores. */

    if (token != NULL)

    {
        while (token != NULL)

        {
            *(short *)longCadena = htons(strlen(token)); /* Pasamos la longitud
de la cadena en hexadecimal */

            memcpy(buffer + apuntador, longCadena + 1, 1); /* Colocamos la
longitud de la cadena en el buffer */

            apuntador++; /* Incrementamos en 1 el apuntador */

            memcpy(buffer + apuntador, token, strlen(token)); /* Colocamos la
cadena en el buffer */

            apuntador += strlen(token); /* Incrementamos el
apuntador con la longitud de la cadena */

```

```

/* Sólo en la primera pasamos la cadena; en las siguientes pasamos NULL */
    token = strtok(NULL, ".");          }    }

    memcpy(buffer + apuntador, CERO, 1); /* Colocamos cero en el buffer
marcando
el final de la direccion web */

    apuntador++;                        /* Incrementamos en 1 el apuntador */

    memcpy(buffer + apuntador, REG_HOST, 2); /* Colocamos que es registro host
en el buffer */

    apuntador += 2;                      /* Incrementamos el apuntador en 2*/

    memcpy(buffer + apuntador, CLASE_INT, 2); /* Colocamos la clase internet
en el buffer */

    apuntador += 2;                      /* Incrementamos el apuntador en 2*/

    /* Devolvemos el apuntador */ return apuntador;
}

/* Solicitamos al usuario la direccion web */
void pedir_direccion(unsigned char direccion[50])
{
    printf("\nDireccion Web: ");
    fgets(direccion, 50, stdin);
    direccion[strlen(direccion) - 1] = '\0'; /* Eliminamos el salto de linea \n */

    /*Comenzamos a idenificar y separar los datos que vienene en la trama recibida*/
void respuesta_dns(unsigned char buffer[512])
{ /* Iniciamos un apuntador en la posicon donde comienza la respuesta del DNS */
    int apuntador = 12 + strlen(buffer + 12) + 4 + 1 + 1;

    /* Mostramos los n registros de recursos */
    printf("\nRespuesta RRs: %d\n", buffer[7]);
    printf("-----");

    /* Ciclamos por el numero de respuestas */
    for (int i = 0; i < (int)buffer[7]; i++)

```

```

    {   apuntador = recorrer_trama(buffer, apuntador);   }

/* Mostramos los n registros de autoridad */
printf("\nAutoridades RRs: %d\n", buffer[9]);
printf("-----");

/* Ciclamos por el numero de respuestas */
for (int i = 0; i < (int)buffer[9]; i++)

    {   apuntador = recorrer_trama(buffer, apuntador);   }

/* Mostramos los n registros adicionales */
printf("\nAdicionales RRs: %d\n", buffer[11]);
printf("-----");

/* Ciclamos por el numero de respuestas */
for (int i = 0; i < (int)buffer[11]; i++)

    {   apuntador = recorrer_trama(buffer, apuntador);   }   }

/* Imprimimos el tipo de registro */
void imprimir_tipo_registro(int tipo)
{
    switch (tipo)
    {
        case 1:

            /* Registro Host */

            printf("\nRegistro: Host");

            break;

        case 2:

            /* Registro (A) servidor de nombres */

            printf("\nRegistro: (A) servidor de nombres");

            break;

        case 5:

```

```

        /* Registro alias (CNAME) */

        printf("\nRegistro: alias (CNAME)");

        break;

        case 0:

        /* Registro de busqueda inversa */
printf("\nRegistro: de busqueda inversa");

        break;

        case 28:

        /* Tiene IPv6 */

        printf("\nRegistro: AAAA (IPv6)");

        break;

        default:

        break;          }          }

/* Imprimimos el tipo d clase (INTERNET XDXXDXD) */
void imprimir_clase()
{ printf("\nClase: Internet");}

/* Imprimimos el tiempo de vida */
void imprimir_tiempo(unsigned char buffer[512], int p)
{ // printf("\nTiempo de vida: %d s", buffer[p + 3]);

printf("\nTiempo de vida: %u%u%u%u s", buffer[p], buffer[p + 1], buffer[p + 2],
buffer[p + 3]);

        // printf("\nTiempo de vida: %d s", t);    }

/* Verificamos que sea apuntador */
int es_apuntador(unsigned char buffer[512], int point)
{
        if (buffer[point] == 192)

        return 1;

        else

        return 0;          }

```

```

/* Imprimimos n caracteres */

int imprimir_n_caracteres(unsigned char buffer[512], int num, int apuntador)
{
    for (int i = 0; i < num; i++)
    {
        apuntador++;
        printf("%c", buffer[apuntador]);
    }

    return apuntador + 1;
}

/* Hacemos un ciclo para imprimir los datos */

int ciclo_impresion(unsigned char buffer[512], int pointer)
{
    int p;

    for (int i = 0; i < strlen(buffer + 12); i++)
    {
        /* Mandamos como parametros el buffer, el tam a imprimir, y el apuntador */
        pointer = imprimir_n_caracteres(buffer, (int)buffer[pointer], pointer);

        if ((int)buffer[pointer] == 0)
        {
            break;
        }

        else if (es_apuntador(buffer, pointer) == 1)
        {
            printf(".");

            /* Incrementamos en un para quedar en el byte donde esta la posicion*/
            pointer++;

            p = guardar_apuntador(buffer, p, pointer);

            p = ciclo_impresion(buffer, p);

            break;
        }

        else

```

```

    {

        /* Imprimimos el '.' */

        printf(".");

    }

    return pointer;
}

/* Guardamos el apuntador en una variable */
int guardar_apuntador(unsigned char buffer[512], int point, int apuntador)
{

    point = (int)buffer[apuntador]; // Guardamos el apuntador

    return point;
}

/* Imprimimos la direccion IP */
int imprimir_ip(unsigned char buffer[512], int pointer)
{

    for (int i = 0; i < 4; i++)

    {

        printf("%d", buffer[pointer]);

        pointer++;

        if (es_apuntador(buffer, pointer) != 1)

        {

            // Ponemos el .

            printf(".");    }    }

    return pointer - 1;
}

/* Recorremos la trama recibida en busca de los datos */
int recorrer_trama(unsigned char buffer[512], int apuntador)
{

```

```

    int point; /* La usaremos para auxiliarnos con los punteros */

    /* Mostramos el nombre */

    printf("\nNombre: ");

    /* Guardamos la posicion que tiene el apuntador en la variable point */

    point = guardar_apuntador(buffer, point, apuntador);

    if ((int)buffer[12])

    { /* code */ }

    point = ciclo_impresion(buffer, point);

/* Incrementamos en +2 para quedar en el segundo byte del tipo de registro */

    apuntador += 2;

    imprimir_tipo_registro((int)buffer[apuntador]);

    /* Guardamos esta posicon para usarla mas adelante */

    int posRegistro = apuntador;

/* Incrementemos en +2 para imprimir la clase internet + 1 para quedar en el
primer byte del tiempo de vida */

    imprimir_clase();

    apuntador += 3;

    imprimir_tiempo(buffer, apuntador);

/* Incrementamos en +4 para quedar en el primer byte del tam de los datos + 1
para quedar en el segundo byte */

    apuntador += 5;

    printf("\nTam de los datos = %d", buffer[apuntador]);

/* Incrementamos en 1 para quedar en el primer byte, donde esta el tam de
cadenas o la IP */

    apuntador++;

/* Si es registro Host, entonces cambiamos el proceso de impresion un poco */

```

```

    if ((int)buffer[posRegistro] == 1)

    {
        printf("\nDireccion: ");

        apuntador = imprimir_ip(buffer, apuntador); }

    /* Si es IPv6, entonces cambiamos el proceso de impresion un poco */
    else if ((int)buffer[posRegistro] == 28)

    { printf("\nDireccion: ");

        apuntador = imprimir_ipv6(buffer, apuntador);    }

    else

    {
/* Si es registro Server, entonces cambiamos el proceso de impresion un poco */

        if ((int)buffer[posRegistro] == 2)

            printf("\nServidor: ");

        else

            printf("\nCNAME: ");

        int bandera = 0;

if (es_apuntador(buffer, apuntador) == 1)

    {

        apuntador++;

        point = guardar_apuntador(buffer, point, apuntador);

        point = ciclo_impresion(buffer, point);

        bandera = 1;

    }

    for (int i = 0; i < (int)buffer[apuntador - 1]; i++)

    {

        if (bandera == 1)

            break;

```

```

/* Mandamos como parametros el buffer, el tam a imprimir, y el apuntador */
apuntador = imprimir_n_caracteres(buffer, (int)buffer[apuntador],
apuntador);

    if ((int)buffer[apuntador] == 0)
    {
        /* Si entramos aqui es porque es fin de la cadena */
        break;
    }

    else if (es_apuntador(buffer, apuntador) == 1)
    {
        printf(".");
/* Incrementamos en un para quedar en el byte donde esta la posicion */
        apuntador++;
        point = guardar_apuntador(buffer, point, apuntador);
        point = ciclo_impresion(buffer, point);
        break;
    }
    else
    {
        /* Imprimimos el '.' */
        printf(".");
    } } }

/* Metemos un salto de linea para separar datos */
printf("\n");

/* Finalmente, incrementamos en +2 para quedar en el segundo byte, donde

```

```
estara el apuntador */

    apuntador += 2;

    return apuntador;
}

/* Imprimimos la IPv6 */
int imprimir_ipv6(unsigned char buffer[512], int apuntador)
{ for (int i = 1; i <= 16; i++){
if (i > 6 && (int)buffer[apuntador] == 0)
{ apuntador++; continue; }
printf("%x", buffer[apuntador]);
apuntador++;
if (i <= 6 && (int)buffer[apuntador] == 0)
printf("0");
if ((i) % 2 == 0)
{ if (i == 16)
break;
printf(":"); }
return apuntador - 1;
}
```

Capturas de funcionamiento:

```
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@kali: ~
File  Actions  Edit  View  Help

(root@kali)-[~]
# exit

(kali@kali)-[~]
$ ls
a.out      cliente_telnet2.py  ClientHandler.java  dns13  Documents  ejemplohttp15  Music  Postman  Public  servidor_http14.js  servidor_telnet.py
Cliente18.java  cliente_telnet.py  Desktop            dns13.c  Downloads  ejemplohttp15.c  Pictures  postman.tar.gz  Servidor18.java  servidor_telnet2.py  Templates

(kali@kali)-[~]
$ gcc dns13.c -o dns13

(kali@kali)-[~]
$ sudo su -
(root@kali)-[~]
```

```
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@kali: /home/kali
File  Actions  Edit  View  Help

(root@kali)-[/home/kali]
# ./dns13

Exito al abrir el socket: Success
Exito en el bind: Success
Direccion Web: www.escom.ipn.mx
Exito al enviar la solicitud: Success
Exito al recibir la respuesta: Success
Respuesta RRs: 2
-----
Nombre: www.escom.ipn.mx
Registro: alias (CNAME)
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 2
CNAME: escom.ipn.mx

Nombre: escom.ipn.mx
Registro: Host
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 4
Direccion: 148.204.58.225

Autoridades RRs: 3
-----
Nombre: escom.ipn.mx
Registro: (A) servidor de nombres
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 7
Servidor: dns3.ipn.mx

Nombre: escom.ipn.mx
Registro: (A) servidor de nombres
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 7
```

```
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@kali: /home/kali

File  Actions  Edit  View  Help
Autoridades RRs: 3

Nombre: escom.ipn.mx
Registro: (A) servidor de nombres
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 7
Servidor: dns3.ipn.mx

Nombre: escom.ipn.mx
Registro: (A) servidor de nombres
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 7
Servidor: dns1.ipn.mx

Nombre: escom.ipn.mx
Registro: (A) servidor de nombres
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 7
Servidor: dns2.ipn.mx

Adicionales RRs: 3

Nombre: dns1.ipn.mx
Registro: Host
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 4
Direccion: 148.204.103.2

Nombre: dns2.ipn.mx
Registro: Host
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 4
Direccion: 148.204.198.2

Nombre: dns3.ipn.mx
Registro: Host
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 4
```

```
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
root@kali: /home/kali

File  Actions  Edit  View  Help
Registro: (A) servidor de nombres
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 7
Servidor: dns3.ipn.mx

Nombre: escom.ipn.mx
Registro: (A) servidor de nombres
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 7
Servidor: dns1.ipn.mx

Nombre: escom.ipn.mx
Registro: (A) servidor de nombres
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 7
Servidor: dns2.ipn.mx

Adicionales RRs: 3

Nombre: dns1.ipn.mx
Registro: Host
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 4
Direccion: 148.204.103.2

Nombre: dns2.ipn.mx
Registro: Host
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 4
Direccion: 148.204.198.2

Nombre: dns3.ipn.mx
Registro: Host
Clase: Internet
Tiempo de vida: 001416 s
Tam de los datos = 4
Direccion: 148.204.235.2.
```

CONCLUSIONES

Navarrete Becerril Sharon Anette:

La integración de un servidor proxy DNS con funciones de filtrado y manejo de errores permite construir una solución resiliente que puede operar de manera continua, recuperándose de fallos en la red y manteniendo un alto nivel de disponibilidad y funcionalidad para los usuarios.

BIBLIOGRAFÍA

Stevens, W. Richard. Programación en red con Unix: La API de Sockets. Addison-Wesley Professional, 2003.

Forouzan, Behrouz A. Comunicaciones de Datos y Redes. McGraw-Hill, 2012. Tanenbaum, Andrew S., y David J. Wetherall. Redes de Computadoras. Pearson, 2010.

Comer, Douglas E. Internetworking con TCP/IP Volumen Uno. Prentice Hall, 2006. Kurose, James F., y Keith W. Ross. Redes de Computadoras: Un Enfoque Descendente. Pearson, 2017.

Beazley, David, y Brian K. Jones. Python Cookbook: Recetas para Dominar Python 3. O'Reilly Media, 2013. Begg, A. Sockets TCP/IP en C: Guía Práctica para Programadores. Morgan Kaufmann, 2000