Instituto Politécnico Nacional

IPN

Escuela Superior de Cómputo

ESCOM

Academia de Redes Aplicaciones para comunicaciones en red

6CV2

Práctica 12

“Protocolo DNS”

Alumna:

Navarrete Becerril Sharon Anette

Fecha de entrega: “ 28 - Noviembre - 2024”

Profesor: Ojeda Santillan Rodrigo

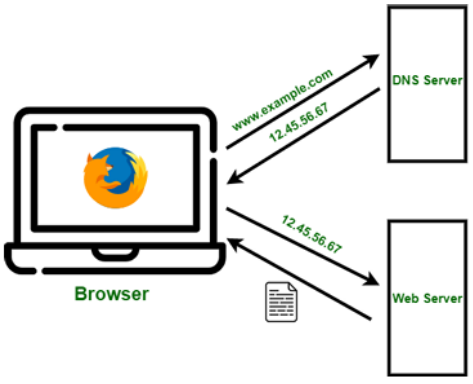
OBJETIVO

Implementar un servidor proxy DNS capaz de actuar como intermediario en las consultas realizadas por clientes, aplicando un sistema de filtrado mediante una lista negra (blacklist). El objetivo es restringir el acceso a dominios no deseados almacenados en la lista negra, de modo que, al solicitar una consulta a un dominio restringido, el servidor devuelva una respuesta de "No Exist Domain" (No existe el dominio), simulando que dicho dominio no está disponible. Con esta práctica, se busca comprender y aplicar los conceptos de bloqueo de consultas en servidores DNS, así como la configuración de listas negras para mejorar la seguridad en la red.

INTRODUCCIÓN

El protocolo DNS (Domain Name System) es una pieza fundamental en la infraestructura de internet, encargado de traducir nombres de dominio comprensibles para los humanos en direcciones IP numéricas que las computadoras utilizan para localizar y conectarse con otros dispositivos en la red. Actúa como una especie de directorio telefónico de internet, permitiendo a los usuarios acceder a sitios web y servicios sin necesidad de recordar secuencias complejas de números. Una característica distintiva del DNS es su estructura jerárquica y distribuida, comenzando en la raíz y extendiéndose a través de dominios de nivel superior (como .com, .org, .net) hasta los dominios específicos que pueden incluir subdominios. Esta organización permite una resolución de nombres eficiente y escalable a nivel global. Además, la distribución del DNS en múltiples servidores alrededor del mundo lo convierte en un sistema tolerante a fallos, asegurando que las solicitudes de consulta puedan procesarse incluso si algún servidor está inactivo. El DNS ofrece diversas funcionalidades que optimizan el rendimiento y la seguridad de la red, como el almacenamiento en caché para acelerar la resolución de nombres y la capacidad de distribuir el tráfico mediante balanceadores DNS. También soporta el DNS dinámico (DDNS) para la asignación flexible de direcciones IP. Más allá de la navegación web, el DNS es esencial en otros servicios de red, como el correo electrónico, donde los registros MX ayudan a localizar servidores de correo.

En la práctica, el DNS es indispensable para la operatividad de internet y redes internas, siendo utilizado por navegadores, aplicaciones móviles y cualquier dispositivo conectado. Además, es clave en configuraciones avanzadas, como redes de entrega de contenido (CDN), que mejoran el rendimiento al dirigir a los usuarios al servidor más cercano. Las organizaciones también lo emplean para gestionar el tráfico interno y reforzar la seguridad, aplicando filtros de contenido y protegiendo contra amenazas mediante servicios DNS seguros.



DESARROLLO

El propósito del código es implementar un servidor proxy DNS en Node.js, que funcione como intermediario entre el cliente y un servidor DNS real. Se utiliza la biblioteca dgram para gestionar las consultas DNS mediante UDP, y mysql2 para conectarse a una base de datos MySQL. Primero, se establece la conexión con MySQL, especificando los detalles del host, usuario, contraseña y base de datos.

El servidor UDP se configura para escuchar en el puerto N de la dirección IP X.X.X.X. Cuando recibe una consulta DNS, extrae el dominio solicitado y verifica en la base de datos MySQL si está en una lista negra. Si el dominio está bloqueado, el servidor responde con un código NXDOMAIN (Non-Existent Domain), indicando que el dominio no existe. En caso de que el dominio no esté en la lista negra, la consulta se reenvía a un servidor DNS real, en este caso el DNS público de Google (8.8.8.8). El servidor DNS real procesa la consulta y devuelve una respuesta que el proxy reenvía al cliente. Así, el proxy permite la resolución normal de dominios no bloqueados mientras filtra aquellos que están en la lista negra.

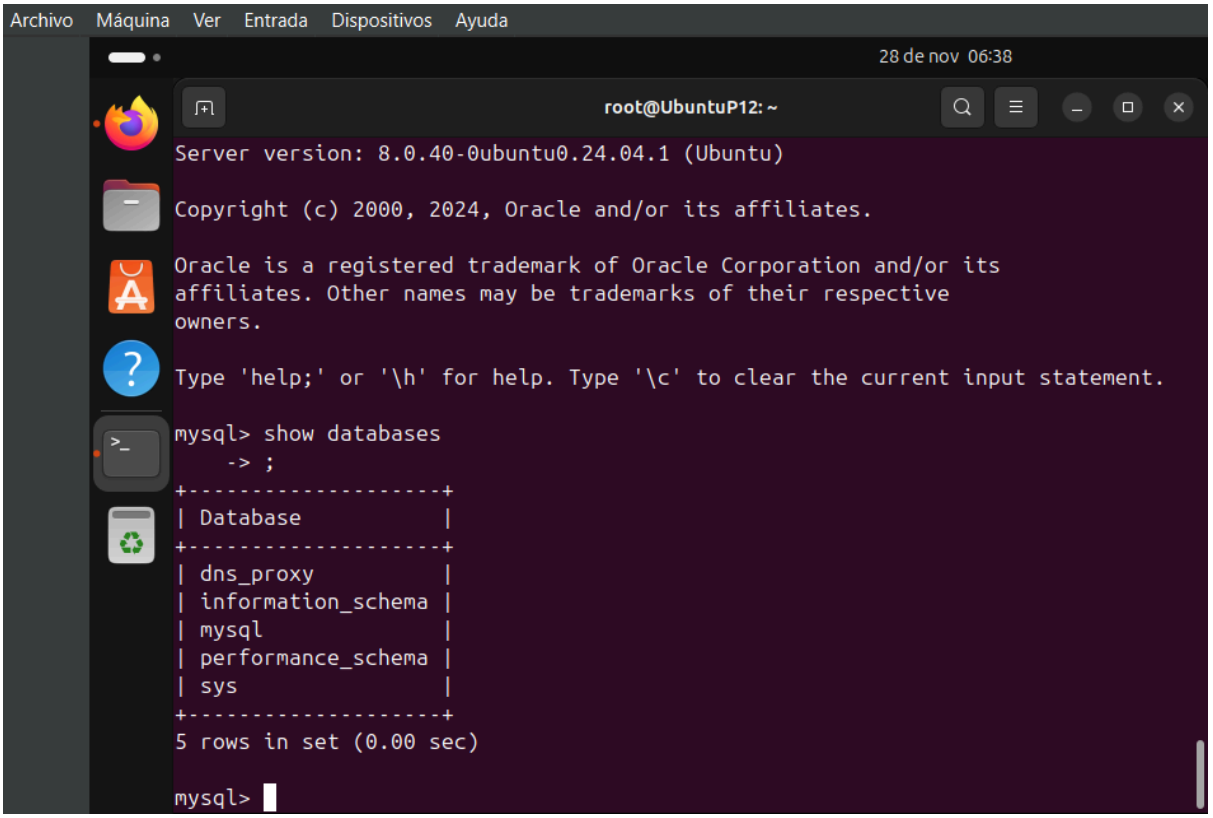
```
const dgram = require('dgram');
const mysql = require('mysql2');
// Configuración de MySQL
const db = mysql.createConnection({
    host: 'localhost',
    user: 'ubuntu',
    password: 'ubuntu',
    database: 'dns_proxy'
});
db.connect((err) => {
    if (err){
        console.error('Error al conectar a MySQL:', err);
        return;
    }
    console.log('Conectado a MySQL');
});

// Crear un servidor UDP para escuchar las consultas DNS
const server = dgram.createSocket('udp4');

// Función para verificar si un dominio está en la lista negra
function isBlacklisted(domain, callback) {
const query = 'SELECT * FROM blacklist WHERE domain = ?';
console.log(`Consultando la lista negra para el dominio: ${domain}`);
db.query(query, [domain], (err, results) => {
    if (err) {
        console.error('Error al consultar la base de datos:', err);
        return callback(false);
    }
    console.log('Resultados de la consulta:', results);
    callback(results.length > 0);
});
}

// Función para manejar las consultas DNS
server.on('message', (msg, rinfo) => {
console.log('Consulta recibida:', msg);
const domain = parseDomainFromQuery(msg);
console.log('Dominio extraído:', domain);
```

```
if (!domain) {
    console.log('Consulta DNS no válida');
    return;
}

isBlacklisted(domain, (blacklisted) => {
    if (blacklisted) {
        console.log(`Dominio bloqueado: ${domain}`);
        sendFakeResponse(msg, rinfo, server);
    } else {
        console.log(`Dominio no bloqueado: ${domain}`);
        forwardQueryToDNS(msg, rinfo, server);
    }
});
});

// Función para analizar el dominio de la consulta DNS
function parseDomainFromQuery(msg) {
const question = msg.slice(12);
let domain = '';
let i = 0;

while (i < question.length) {
    const len = question[i];
    if (len === 0) break;
    domain += question.slice(i + 1, i + 1 + len).toString('utf8') + '.';
    i += len + 1;
}

return domain.slice(0, -1);
}

// Función para enviar una respuesta falsa (IP 0.0.0.0)
function sendFakeResponse(msg, rinfo, server) {
const response = Buffer.alloc(msg.length);
msg.copy(response);
response[2] = 0x81; // Respuesta con error (NXDOMAIN)
response[3] = 0x83;
server.send(response, 0, response.length, rinfo.port, rinfo.address, (err)
=>
{
    if (err) console.error('Error al enviar la respuesta falsa:', err);
});
}

// Función para reenviar la consulta DNS al resolver real
function forwardQueryToDNS(msg, rinfo, server) {
const client = dgram.createSocket('udp4');
client.on('message', (response) => {
    server.send(response, 0, response.length, rinfo.port, rinfo.address,
(err) => {
            if (err) console.error('Error al enviar la respuesta del DNS
real:',
err);
    });
    client.close();
});

client.send(msg, 0, msg.length, 53, '8.8.8.8', (err) => {
```

```
      if (err) console.error('Error al enviar la consulta al DNS real:',
err);
});
}

// Iniciar el servidor DNS Proxy
server.bind(53, '192.168.0.53', () => {
console.log('Servidor DNS Proxy escuchando en el puerto 3306');
});
```

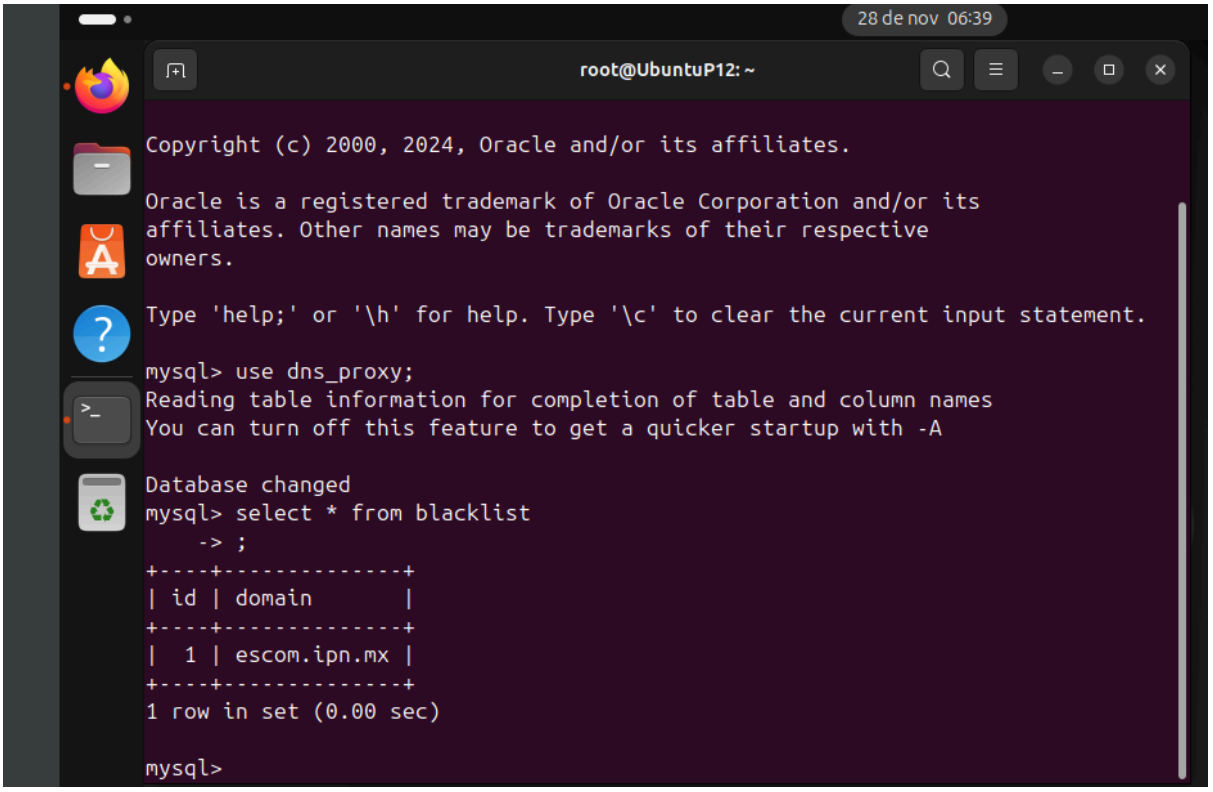
Una vez que se tenga la lógica se debe de crear la base de datos, en este caso utilice MySQL Workbench pero se puede utilizar Postgre, Oracle, etc.

```
CREATE DATABASE dns_proxy;
USE dns_proxy;
CREATE TABLE blacklist ( id INT AUTO_INCREMENT PRIMARY KEY,
domain VARCHAR(255) NOT NULL
);
```

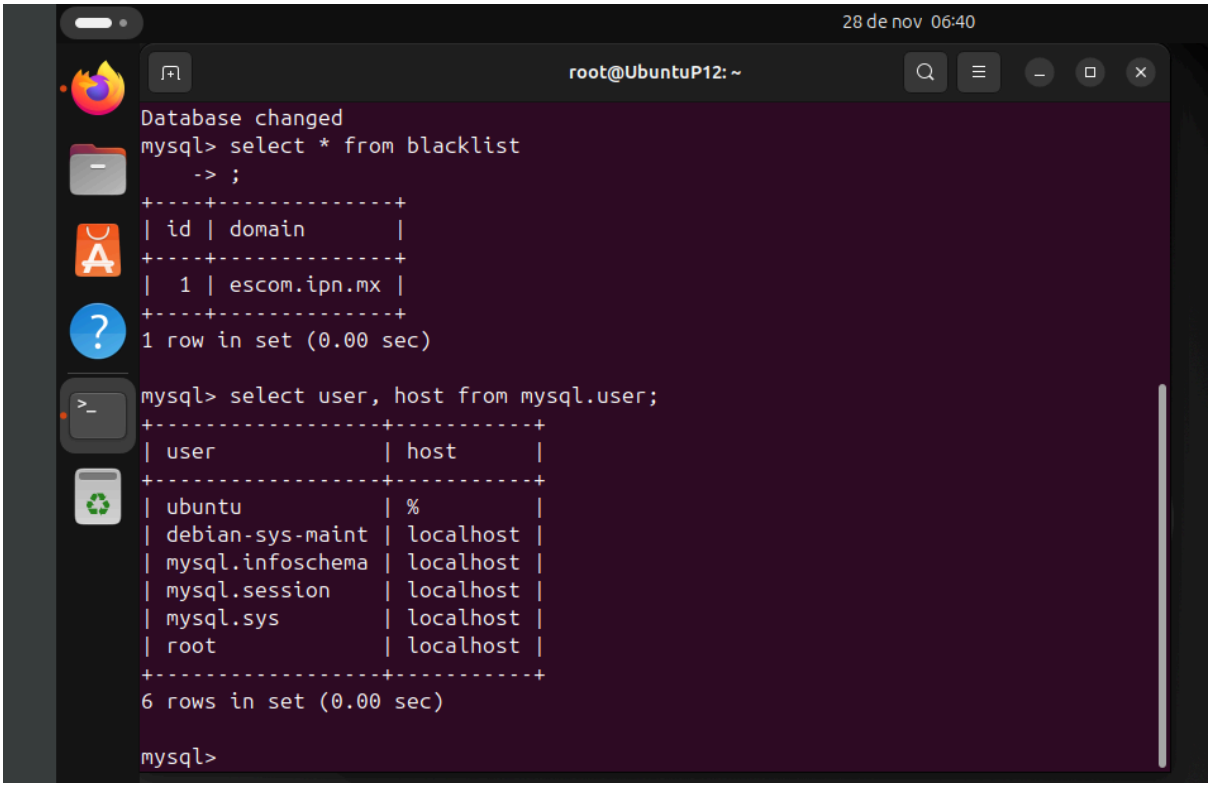


Se puede agregar un dominio para fines prácticos en este caso se va a intentar que se bloquee el sitio escom.ipn.mx (la página de la escuela).

```
INSERT INTO blacklist (domain) VALUES ('escom.ipn.mx');
```

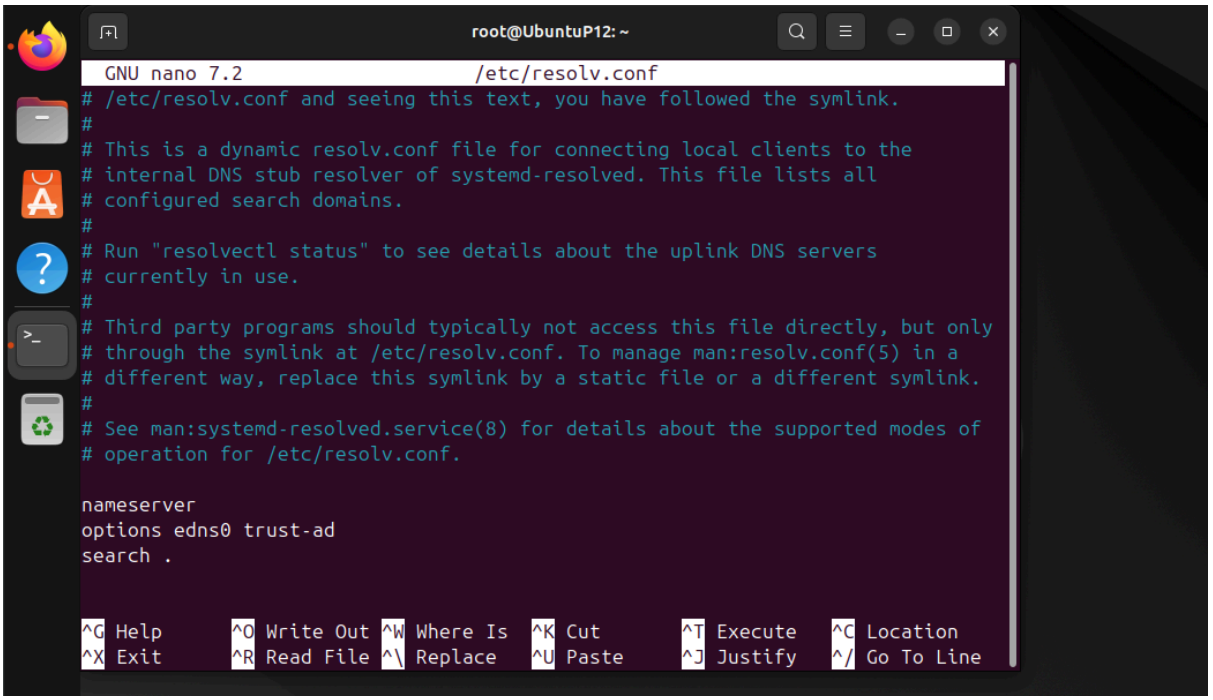


Se debe de configurar para que se puede acceder de forma remota desde cualquier IP concediendo privilegios.

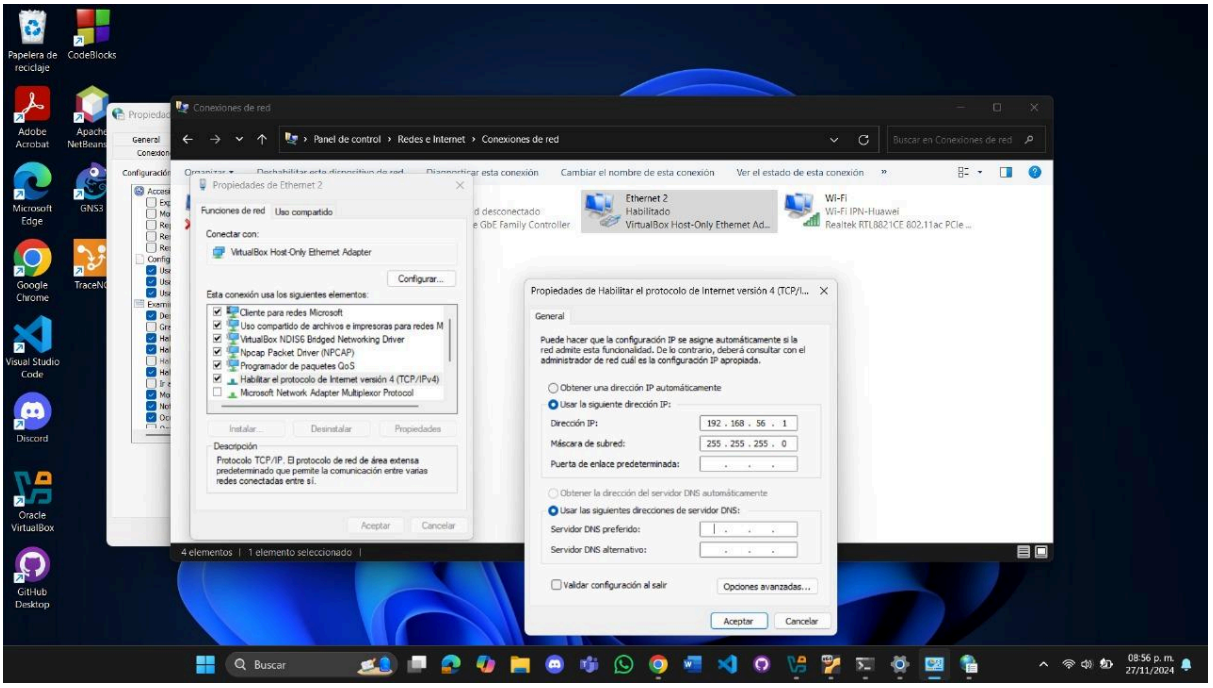


Ahora se debe de configurar algunos archivos en Ubuntu en este caso se debe de establecer la IP a la que va a apuntar el servidor DNS ya que se encuentra uno por defecto:

```
sudo nano /etc/resolv.conf
```



Después de eso se debe de configurar de lado de Windows el servidor DNS por preferencia y esto es desde el apartado de redes en el panel de control y se selecciona la opción de cambiar configuración del adaptador: Se selecciona la opción de IPV4 y se presiona Propiedades:

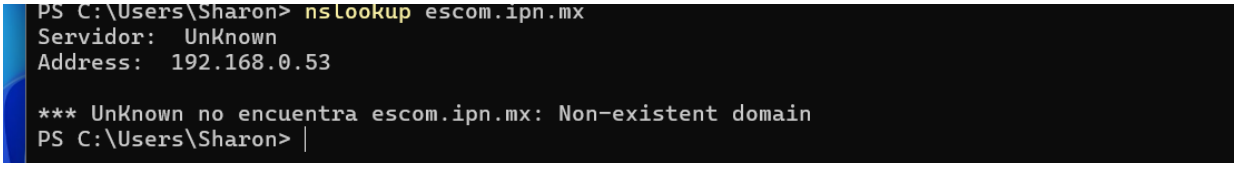


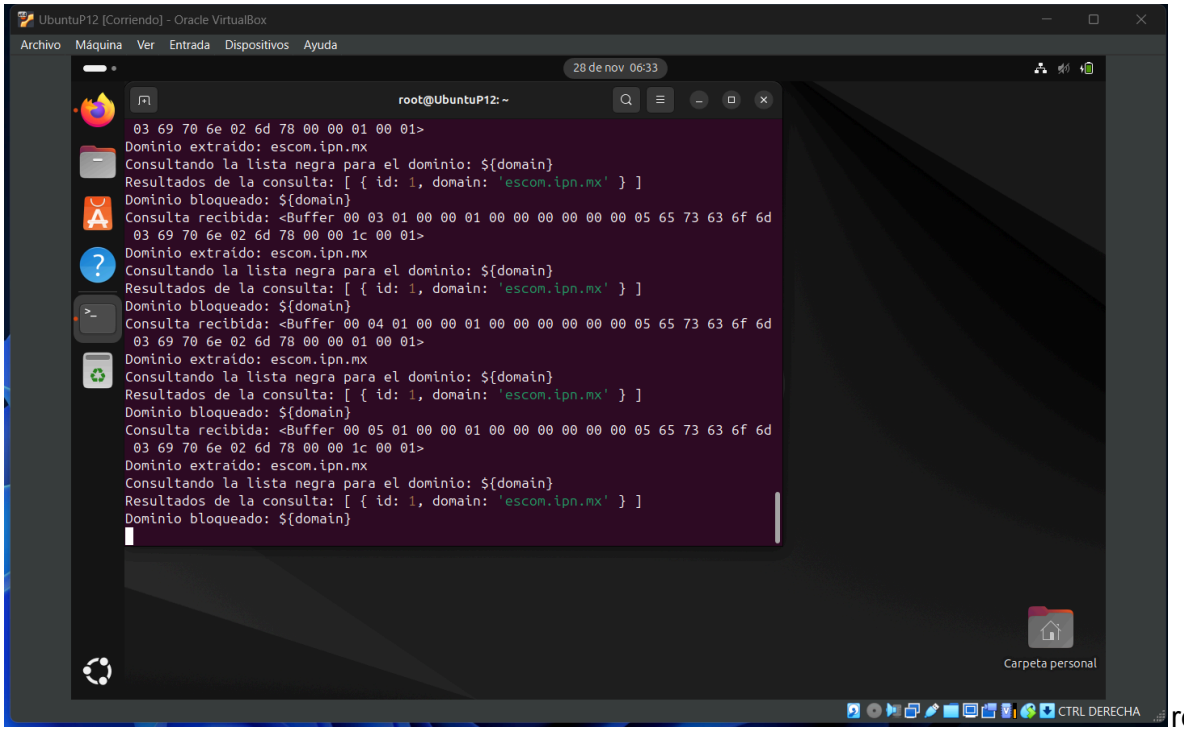
Se debe de poner la IP donde se esta ejecutando el servidor DNS, debe ser la misma que se puso como servername en Ubuntu. Ahora se ejecuta el servidor en Ubuntu con node servidor_proxy.js

También si en CMD o Poweshell se puede comprobar con el siguiente comando:

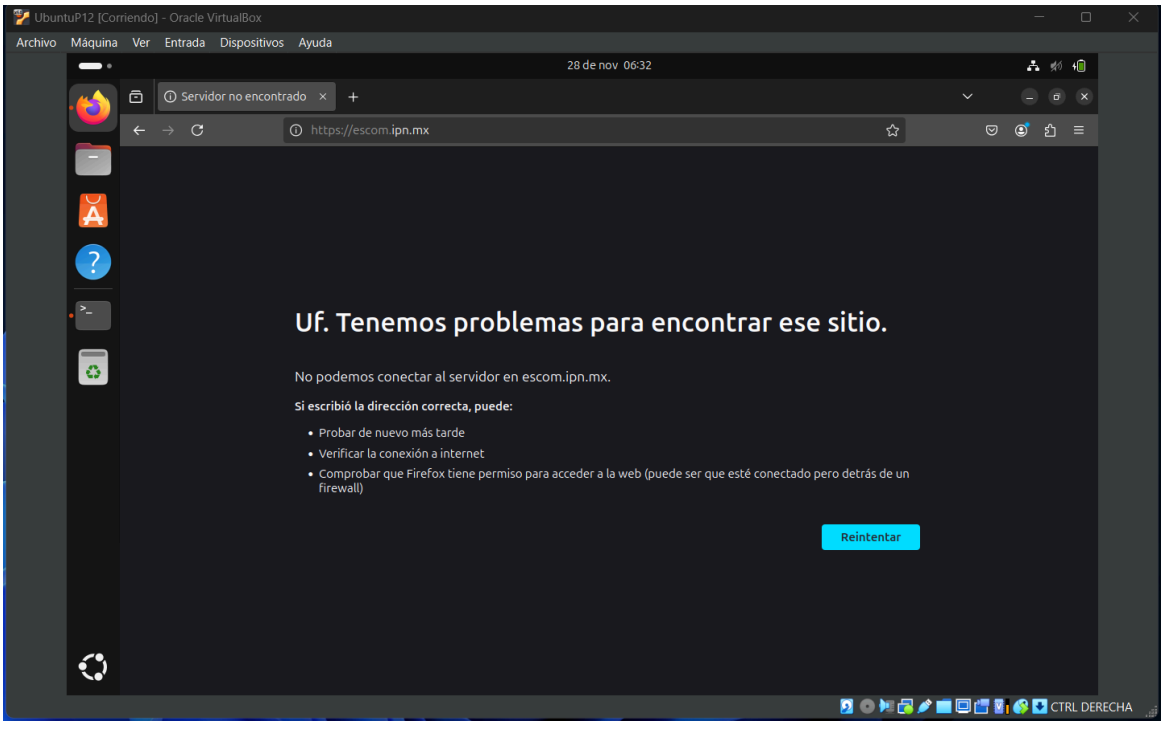
```
nslookup escom.ipn.mx
```

Capturas de funcionamiento:





re



CONCLUSIONES

Navarrete Becerril Sharon Anette:

La configuración remota de la base de datos y del servidor DNS para trabajar en conjunto demuestra la importancia de la interoperabilidad y la flexibilidad en el manejo de herramientas y servicios de red. Esta práctica resalta cómo diferentes tecnologías pueden integrarse para alcanzar objetivos de filtrado y gestión del tráfico de manera eficiente.

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