Laboratorio del Ataque de Kaminsky

1 Descripción del Laboratorio

El objetivo de este laboratorio es que los estudiantes aprendan y experimenten con el ataque remoto al DNS, en particular DNS cache poisoning attack, también llamado el ataque de Kaminsky. DNS (Domain Name System o Sistema de nombre de Dominios) es la guía de teléfono de la Internet; se encarga de traducir los hostnames a direcciones IP (y visce versa). Esta traducción se hace a través de la resolución DNS, esta ocurre detrás de escena. Los ataques DNS manipulan este proceso de resolucion en varias maneras, con la intención de desviar a los usuarios a destinos alternativos, que a menudo son maliciosos. Este laboratorio se focaliza en una técnica particular de ataque al DNS, llamada *DNS Cache Poisoning attack*. En otro laboratorio SEED, hemos diseñado actividades para conducir el mismo ataque pero en un entorno de red loocal, es decir servidor DNS al cual tiene acceso el atacante y la víctima se encuentran en la misma red, donde es posible hacer sniffeo de paquetes. En este laboratorio 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.

Este laboratorio cubre los siguientes tópicos:

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

Lecturas y Videos. Para una cobertura más detallada sobre el protocolo DNS y sus ataques puede consultar:

- Capítulo 18 del libro de SEED, *Computer & Internet Security: A Hands-on Approach*, 2nd Edition, by Wenliang Du. Para más detalles `https://www.handsonsecurity.net`.
- Sección 7 del curso de SEED en Udemy, *Internet Security: A Hands-on Approach*, by Wenliang Du. Para más detalles `https://www.handsonsecurity.net/video.html`.

Entorno de Laboratorio. Este laboratorio ha sido testeado en nuestra imagen pre-compilada de una VM con Ubuntu 20.04, que puede ser descargada del sitio oficial de SEED. Sin embargo, la mayoría de nuestros laboratorios pueden ser realizados en la nube para esto Ud. puede leer nuestra guía que explica como crear una VM de SEED en la nube.

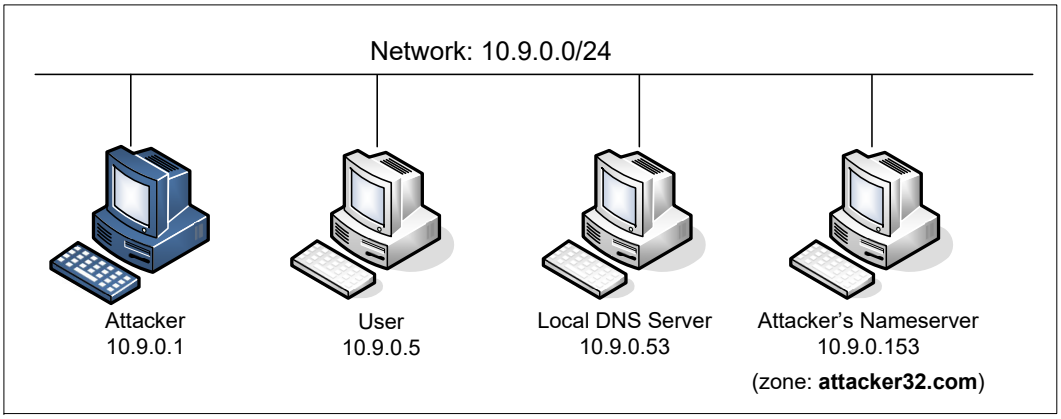

Figure 1: Setup del Entorno

2 Setup del Entorno de Laboratorio (Tarea 1)

El principal objetivo para un ataque de DNS cache poisoning es el servidor de DNS local. Obviamente es ilegal atacar un servidor real, poor lo que necesitaremos crear y configurar nuestro propio servidor DNS para conducir los ataques en nuestros experimentos. El entorno de laboratorio necesita cuatro máquinas por separado: una va a ser la máquina víctima, la segunda será el servidor de DNS local y las dos restantes serán las máquinas de los atacantes. El setup del entorno del laboratorio se ilustra en la Figura 1.

Hemos puesto a todas las máquinas en la misma LAN con el objetivo de facilitar un poco todo. Los estudiantes no pueden explotar este hecho en sus ataques; ellos deberían de considerar a la máquina del atacante como una máquina remota, es decir, el atacante no puede sniffear paquetes en la LAN. Esto es diferente al ataque local del DNS.

2.1 Setup del Contenedor y sus Comandos

Para empezar a preparar el contenedor, deberá descargarse el archivo `Labsetup.zip` ubicado en el laboratorio correspondiente dentro del sitio web oficial y copiarlo dentro de la Máquina Virtual prevista por SEED. Una vez descargado deberá descomprimirlo y entrar dentro del directorio `Labsetup` donde encontrará el archivo `docker-compose.yml` que servirá para setear el entorno de laboratorio. Para una información más detallada sobre el archivo `Dockerfile` y otros archivos relacionados, puede encontrarla dentro del Manual de Usuario del laboratorio en uso, en el sitio web oficial de SEED.

Si esta es su primera experiencia haciendo el setup del laboratorio usando contenedores es recomendable que lea el manual anteriormente mencionado.

A continuación, se muestran los comandos más usados en Docker y Compose. Debido a que estos comandos serán usados con mucha frecuencia, hemos creados un conjunto de alias para los mismos, ubicados en del archivo `.bashrc` dentro de la Máquina Virtual provista por SEED (Ubuntu 20.04)

```
$ docker-compose build  # Build the container image
$ docker-compose up     # Start the container
$ docker-compose down   # Shut down the container

// Aliases for the Compose commands above
$ dcbuild       # Alias for: docker-compose build
$ dcup          # Alias for: docker-compose up
$ dcdown        # Alias for: docker-compose down
```

Dado que todos los contenedores estarán corriendo en un segundo plano. Necesitamos correr comandos para interactuar con los mismos, una de las operaciones fundamentales es obtener una shell en el contenedor. Para este propósito usaremos "`docker ps`" para encontrar el ID del contenedor deseado y ingresaremos "`docker exec`" para correr una shell en ese contenedor. Hemos creado un alias para ello dentro del archivo `.bashrc`

```
$ dockps        // Alias for: docker ps --format "{{.ID}}  {{.Names}}"
$ docksh <id>   // Alias for: docker exec -it <id> /bin/bash

// The following example shows how to get a shell inside hostC
$ dockps
b1004832e275  hostA-10.9.0.5
0af4ea7a3e2e  hostB-10.9.0.6
9652715c8e0a  hostC-10.9.0.7

$ docksh 96
root@9652715c8e0a:/#

// Note: If a docker command requires a container ID, you do not need to
//       type the entire ID string. Typing the first few characters will
//       be sufficient, as long as they are unique among all the containers.
```

En caso de problemas configurando el entorno, por favor consulte la sección “Common Problems” en el manual ofrecido por SEED.

2.2 Sobre el Contenedor del Atacante

Para este laboratorio podemos usar tanto una Máquina Virtual como un contenedor como máquina de ataque. Si observa el archivo Docker Compose, verá que el contenedor de ataque está configurado de forma diferente al resto de los contenedores.

- *Directorio Compartido.* Cuando usemos el contenedor del atacante para realizar los ataques, necesitamos poner el código de ataque dentro del contenedor. La edición del código es más conveniente dentro de la Máquina Virtual que dentro del contenedor, ya que podemos usar nuestro editor de texto preferido. Para que la Máquina Virtual y el contenedor puedan compartir archivos, hemos creado un directorio compartido entre ambos para esto hemos usado `volumes` de Docker. Dentro del archivo de Docker Composer, encontrará que se ha agregado esta entrada en algunos de los contenedores. Esta entrada indica que se montará el diretorio `./volumes` en la Máquina Host (es decir nuestra Máquina Virtual) y se podrá usar dentro del contenedor. Escribiremos nuestro código dentro del directorio `./volumes` (en la Máquina Virtual) y este podrá ser usado en el contenedor.

  ```
  volumes:
         - ./volumes:/volumes
  ```

- *Host mode.* En este laboratio, el atacante va a necesitar sniffear los paquetes, pero correr el programa de sniffing dentro del contenedor del atacante tiene sus inconvenientes, ya que el contenedor está atachado a un switch virtual y sólo podrá ver su propio tráfico y no el del resto de los contenedores. Para solucionar este problema, usaremos el modo `host` para el contenedor del atacante. Esto permite que el contenedor del atacante vea el tráfico de toda la red. La siguiente entrada es usada para el contenedor del atacante:

```
network_mode: host
```

Cuando un contenedor está en modo `host`, este puede ver todas las interfaces de red de los hosts que la componen, inclusive tiene la misma dirección IP como si fuera el host principal. Básicamente es ponerlo en el mismo espacio de red como si fuera la Máquina Virtual de Host. Sin embargo, el contenedor sigue siendo una máquina diferente.

2.3 Resumen de la Configuración del DNS

Todos los contenedores han sido configurados para este laboratorio. Por lo que haremos un resumen de ellos, de esta manera los estudiantes estarán al tanto de estas configuraciones. Para explicaciones más detalladas sobre las configuraciones puede consultar el manual.

Servidor de DNS local. Como servidor de DNS local usaremos el software BIND 9. BIND 9 carga su configuración de un archivo llamado `/etc/bind/named.conf`. Este es el principal archivo de configuración y usualmente contiene varias entradas de `"include"`, por medio de este include puede cargar las configuraciones de diferentes archivos. Uno de los archivos usado por ese include es llamado `/etc/bind/named.conf.options`. Es en este archivo donde se establece la configuración actual.

- *Simplificación.* Los servidores DNS hoy en día randomizan el número de puerto de origen en sus consultas DNS; esto hace que los ataques sean mucho más difíciles. Desafortunadamente, muchos servidores DNS siguen usando número de puertos de origen predecibles. Para simplificar este laboratorio, hemos fijado el número de puerto de origen a `33333` dentro del archivo de configuración.
- *Desactivando DNSSEC.* DNSSEC fue introducido como mecanismo de protección en contra de ataques de spoofing en los servidores DNS. Para mostrar como funciona este ataque, hemos desactivado esta protección en el archivo de configuración.
- *DNS caché.* Durante el ataque, necesitaremos inspeccionar la caché DNS en el servidor de DNS local. Los siguientes dos comando sirven para este propósito. El primer comando hace un dump 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
```

- *Forwardeo de la zona `attacker32.com`.* Una zona de forwardeo es agregada en el servidor de DNS local, por lo que si alguien quiere consultar el dominio `attacker32.com`, la consulta será forwardeada al nameserver de este dominio, que será hosteado en el contenedor del atacante. La entrada para esta zona se ubcica dentro del archivo `named.conf`.

```
zone "attacker32.com" {
    type forward;
    forwarders {
        10.9.0.153;
    };
};
```

Máquina del Usuario. El contenedor del usuario cuya dirección IP es `10.9.0.5` está configurada 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.

Nameserver del Atacante. Dentro de la máquina del atacante, se hostean 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`:

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

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

2.4 Testeando la Configuración DNS

Desde el contenedor del usuario, ejecutaremos una serie de comandos para asegurarnos que la configuración de nuestro laboratorio sea la adecuada. En su informe de laboratorio, por favor documente los resultados de estas pruebas.

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 de venir del archivo de zona (`attacker32.com.zone`) que se configuró en nameserver del atacante. Si esto no es lo que ud. obtiene, la configuración que se hizo ha sido errónea. Por favor describa su observación en el informe del laboratorio.

```
$ dig ns.attacker32.com
```

Obtener la dirección IP de `www.example.com`. Dos nameservers se encuentran hosteando el dominio `example.com`, uno es el nameserver oficial y el otro es el contenedor del atacante. Consultaremos ambos nameservers y veremos que respuesta obtenemos de ellos. Por favor ejecute los siguientes comando (desde la máquina de usuario) y describa su observación.

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

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

Obviamente, nadie va a consultar `ns.attacker32.com` para obtener la dirección IP de `www.example.com`; siempre se consultará el nameserver oficial del dominio `example.com`. El objetivo del ataque de DNS

cache poisoning es hacer que la víctima consulte `ns.attacker32.com` para obtener la dirección IP de `www.example.com`. Así, si nuestro es exitoso, si corremos el primer comando `dig` sin usar la opción `@`, deberíamos de obtener el resultado falsificado por el atacante, en vez del resultado autentico del nameserver legítimo del dominio `www.example.com`.

3 Las tareas de ataque

El principal objetivo de los ataques DNS sobre un usuario es poder redireccionar al usuario hacia una máquina B cuando el usuario intente acceder a una máquina A usando el hostname de A. Por ejemplo, cuando un usuario trata de acceder al online banking, los atacantes pueden redireccionarlo a un sitio web malicioso que luce casi igual al sitio oficial del banco, en consecuencia el usuario puede ser engañado y así los atacantes pueden obtener sus credenciales bancarias.

En esta tarea, vamos a usar el nombre de dominio `www.example.com` como nuestro objetivo para hacer el ataque. Cabe señalar que el dominio `www.example.com` está reservado solamente para el uso experimental dentro del contexto de este laboratorio y no para el mundo real. La dirección IP real de `www.example.com` es `93.184.216.34` y su nameserver es controlado por la Internet Corporation for Assigned Names and Numbers (ICANN). Cuando el usuario ejecuta el comando `dig` sobre este dominio o usa el navegador para visitarlo, la máquina del usuario envia una consulta DNS a su servidor de DNS local que consultará la dirección IP usando el nameserver de `example.com`.

La meta del ataque es ejecutar un ataque de DNS cache poisoning attack en el servidor de DNS local, de tal forma que cuando el usuario ejecute el comando `dig` para obtener la dirección IP de `www.example.com`, el servidor de DNS local terminará consultando el nameserver del atacante `ns.attacker32.com` y así obtendrá la dirección IP definida por atacante. Como resultado final el usuario será dirigido al sitio web del atacante en lugar del sitio original `www.example.com`.

3.1 Como funciona el Ataque de Kaminsky

En esta tarea, el atacante envía una consulta de petición DNS al servidor DNS de la víctima (`Apollo`), activando una consulta DNS desde `Apollo`. La consulta puede viajar a través de uno de los servidores DNS raíz, el servidor DNS `.COM` y el resultado final vendrá desde el servidor DNS de `example.com`. Esto se ilustra en la Figura 2. En caso que la información del nameserver para el dominio `example.com` esté cacheada por `Apollo`, la consulta no irá al servidor raíz o al servidor `.COM`; esto se ilustra en la Figura 3. En este laboratorio, la situación representadda en la Figura 3 es la más común, por lo que usaremos esta figura como base para describir el mecanismo de ataque.

Mientras `Apollo` espera por la respuesta DNS del nameserver de `example.com`, el atacante puede enviar respuestas falsificadas/spoofeadas a `Apollo`, fingiendo que las respuesta provienen del nameserver de `example.com`. Si la respuesta falsificada llega primero, será aceptada por `Apollo`. Podemos decir que el ataque fue exitoso.

Si ud. ha hecho el laboratorio de Ataque de DNS local, debería de saber que esos ataques asumen que el atacante y el servidor DNS están en la misma LAN, es decir el atacante puede observar el mensaje de una consulta DNS. Cuando el atacante y el servidor DNS no se encuentran en la misma LAN, el ataque de cache poisoning se vuelve un poco más difícil. La dificultad es causada por el hecho que el ID de transacción en el paquete de respuesta DNS debe de coincidir con el del paquete de la consulta. Dado que el ID de transacción en la consulta es generado de forma aleatoria, sin poder observar el paquete de consulta, no es fácil para un atacante saber el ID correcto.

Obviamente, el atacante puede adivinar el ID de transacción, dado que el tamaño de este ID es de sólo 16 bits, si el atacante puede falsificar/spoofear K respuestas dentro de la ventana de ataque (es decir antes que la

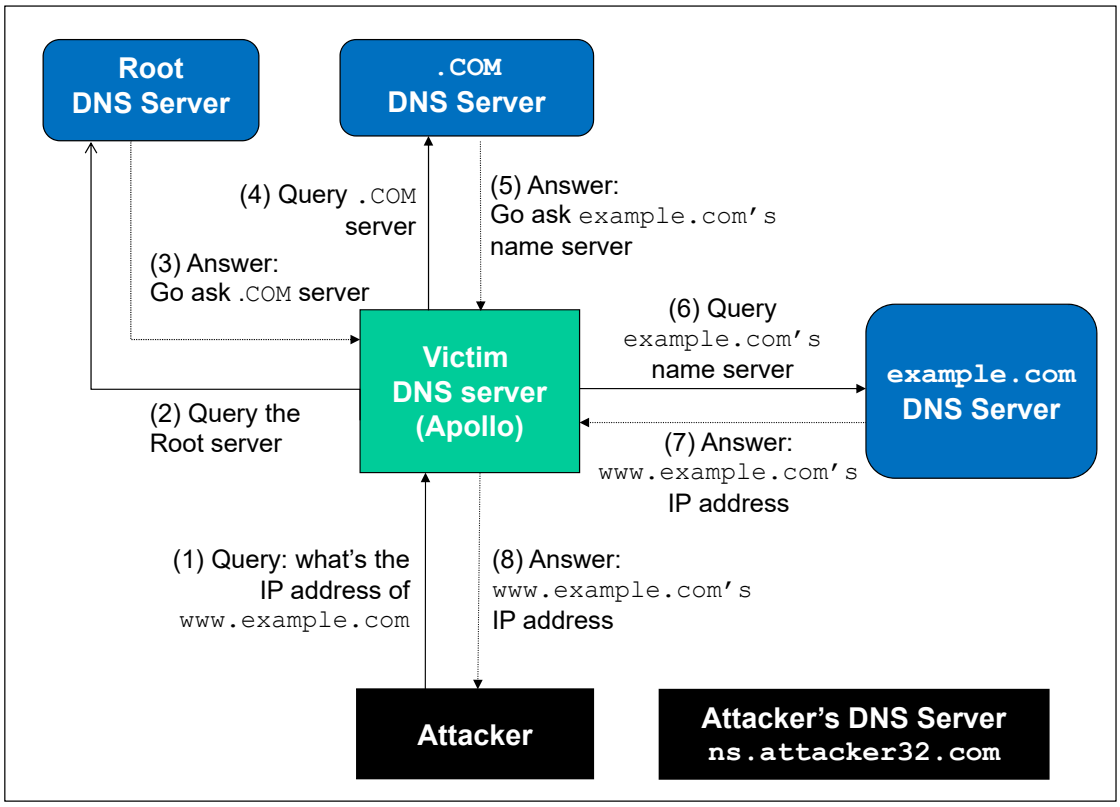

Figure 2: El proceso de consulta DNS

respuesta legítima llegue a destino), la probabilidad de suceso de K es de 2^{16}. Enviar cientos de respuestas falsas no es poco práctico, por lo que no serán necesarios muchos intentos antes de que el atacante pueda tener éxito.

Sin embargo, el hipotético ataque que se planteó anteriormente ha pasado por alto el efecto de la caché. En realidad, si el atacante no es lo suficientemente afortunado en su trabajo de adivinar el ID correcto antes de que el paquete de la respuesta original llegue a su destino, la información correcta será cacheada por un período de tiempo en el servidor DNS. Este efecto de almacenamiento en la caché, hace imposible que el atacante pueda falsificar otra respuesta con respecto al mismo nombre, dado que el servidor DNS no enviará otra consulta DNS para este nombre hasta que expire la caché para este nombre. Para falsificar la respuesta sobre el mismo nombre, el atacante tiene que esperar por otra consulta DNS sobre este nombre, lo que significa que debe de esperar que la caché expire. El tiempo de expiración puede estar en el orden de la horas o de los días.

El Ataque de Kaminsky. Dan Kaminsky ideó una elegante técnica para evadir este efecto del almacenamiento en la caché [**?**]. Con el ataque de Kaminsky, los atacantes tienen la posibilidad de atacar continuamente un servidor DNS en un nombre de dominio específico sin la necesidad de esperar que la caché expire, por lo que los ataques pueden ser exitosos en un tiempo relativamente corto. Los detalles del ataque son descriptos en [**?**, **?**]. En esta tarea, usaremos este ataque. Los siguientes pasos que hacen referencia a la Figura 3 que describe el ataque.

1. El atacante consulta el servidor DNS `Apollo` por un nombre no existente en `example.com` por ejemplo `twysw.example.com`, donde `twysw` es un nombre aleatorio.

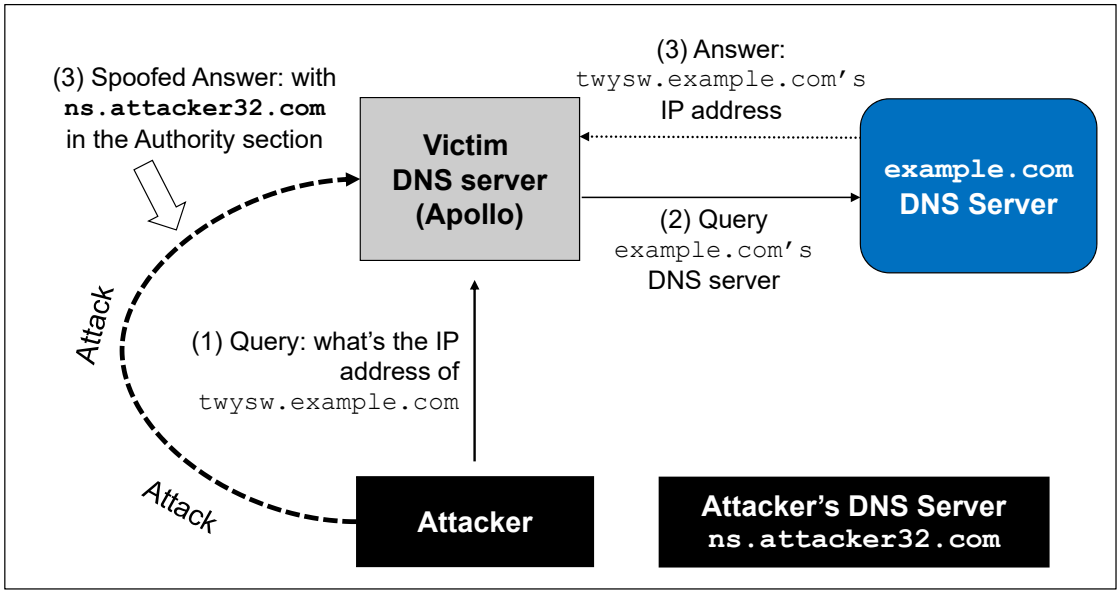

Figure 3: El Ataque Kaminsky

2. Dado que este mapeo no está disponible en la cache del DNS de `Apollo`, `Apollo` envía una consulta DNS al nameserver del dominio `example.com`.

3. Mientras que `Apollo` espera por la respuesta, el atacante inunda `Apollo` con un flujo de respuestas DNS falsificadas, en cada una de estas respuestas se prueba un ID de transacción diferente esperando que alguna sea la correcta. En la respuesta, el atacante no solo proporciona una resolución de IP para `twysw.example.com`, el atacante también proporciona un registro de "Authoritative Nameservers", indicando a `ns.attacker32.com` como nameserver para el dominio `example.com`. Si la respuesta falsificada supera a las respuestas reales y el ID de transacción coincide con el de la consulta, `Apollo` va a aceptarla y cacheará la respuesta falsificada por lo que la caché DNS de `Apollo` está envenenada (poisoned)

4. Incluso si la respuesta DNS que se falsificó/spoofeo falla (por ejemplo el ID de transacción no coincide o llega tarde), no importa, porque la próxima vez, el atacante consultará un nombre diferente, por lo que `Apollo` deberá de enviar otra consulta dándole al atacante otra chance de hacer un ataque de este tipo. Definitivamente este ataque evade el efecto del almacenamiento en la caché.

5. Si el ataque funciona, en la caché DNS de `Apollo`, el nameserver para `example.com` será reemplazado con el que proporciona el atacante `ns.attacker32.com`. Para demostrar que el ataque funciona, los estudiantes deben de mostrar que tal registro se encuentra dentro de la caché DNS de `Apollo`.

Descripción de la Tarea. La implementación del ataque de Kaminsky es algo desafiante, por lo que la hemos separado en varias sub-tareas. En la Tarea 2, construíremos petición DNS para un hostname aleatorio usando el dominio `example.com`. En la Tarea 3, construíremos spoofearemos las respuestas provenientes del nameserver en `example.com`. En la Tarea 4, pondremos todo junto para lanzar el ataque de Kaminsky y finalmente en la Tarea 5, verificaremos el impacto del ataque.

3.2 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 envié consultas, de esta forma tendrán las chances necesarias para empezar a falsificar respuestas DNS. Dado que los atacantes necesitan intentar varias veces antes de que el ataque sea exitoso, es mejor automatizar el proceso usando un programa.

Los estudiantes deben escribir un programa para enviar consultas DNS al servidor DNS objetivo (es decir, el servidor de DNS local de nuestro setup). La tarea de los estudiantes es escribir este programa y demostrar (usando Wireshark) que sus consultas hacen que el servidor DNS objetivo esté enviando las consultas DNS correspondientes. Los requerimientos de performance para esta tarea no son altos, por lo que lo estudiantes pueden usar C o Python (con Scapy) para desarrollar el código del programa. A continuación se muestra un fragmento de código (los +++ son placeholders; los estudiantes deben de reemplazarlos con los valores pertinentes):

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

ip  = IP(dst='+++', src='+++')
udp = UDP(dport=+++, sport=+++, chksum=0)
request = ip/udp/dns
```

3.3 Tarea 3: Spoofeando respuestas DNS.

En esta tarea, necesitamos spoofear las respuestas DNS en el ataque de Kaminsky. Dado que nuestro objetivo es `example.com`, necesitamos spoofear las respuesta del nameserver para este dominio. Los estudiantes primero necesitarán descubrir la dirección IP de los nameservers originales de `example.com` (cabe señalar que existen multiples nameservers para este dominio).

Los estudiantes pueden usar Scapy para implementar esta tarea. El siguiente fragmento de código, construye un paquete que representa una respuesta DNS que incluye la sección Question, una sección Answer y una sección NS. En este código deejemplo, usarmos +++ como placeholders; los estudiantes deben de reemplazar estos placeholders con los valores correctos que son necesarios para el ataqu de Kaminsky. Los estudiantes deben de explicar porque escogieron esos valores.

```
name   = '+++'
domain = '+++'
ns     = '+++'

Qdsec  = DNSQR(qname=name)
Anssec = DNSRR(rrname=name,   type='A',  rdata='1.2.3.4', ttl=259200)
NSsec  = DNSRR(rrname=domain, type='NS', rdata=ns, ttl=259200)
dns    = DNS(id=0xAAAA, aa=1, rd=1, qr=1,
             qdcount=1, ancount=1, nscount=1, arcount=0,
             qd=Qdsec, an=Anssec, ns=NSsec)

ip    = IP(dst='+++', src='+++')
udp   = UDP(dport=+++, sport=+++, chksum=0)
reply = ip/udp/dns
```

Dado que esta respuesta por sí sóla no será capaz de conducir a un ataque exitoso, para demostrar lo hecho en esta tarea, los estudiantes deberán de usar Wireshark para capturar las respuestas DNS spoofeadas

y mostrar que los paquetes spoofeados son válidos.

3.4 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 entre en el número correcto de transacción y llegue antes que la respuesta original. La velocidad es esencial en este asunto: mientras más paquetes enviemos, más posibilidades tendremos de que el ataque sea exitoso. Si usamos Scapy para enviar respuestas DNS spoofeadas como hicimos en las tareas anteriores, la taza de exito será baja. Los estudiantes pueden usar C, pero construir paquetes DNS en C es algo no trivial. Vamos a introducir un enfoque híbrido usando Scapy y C (vea el libro de SEED para más detalles)

Con este enfoque híbrido, primero usaremos Scapy para generar un paquete DNS que sirva como planilla, esta será guardada en un archivo. Luego, cargaremos esta planilla dentro de un programa en C, y le haremos unos pequeños cambios a algunos de sus campos y por último enviaremos este paquete. Hemos incluído un código base en C dentro de `Labsetup/Files/attack.c`. Los estudiantes pueden realizar cambios en las áreas que están marcadas. Para una explicación mas detallada consulte la sección Guías.

Chequear la caché 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 caché DNS y se encarga de buscar si la palabra `attacker` se encuentra dentro de la caché (en nuestro ataque hemos usado `attacker32.com` como el dominio del atacante, si los estudiantes desean usar un nombre diferente, deberían de buscar por una palabra difernte)

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

3.5 Tarea 5: Verificación 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, vaya a la máquina de usuario y corra los siguientes dos comandos usando `dig`. En las respuesta, la direcciones IP para `www.example.com` deberían de ser las mismas para ambos comandos, y debería de ser lo que haya incluído en el archivo de la zona del nameserver del atacante.

```
// Ask the local DNS server to do the query
$ dig www.example.com

// Directly query the attacker32 nameserver
$ dig @ns.attacker32.com www.example.com
```

Por favor incluya su observación (capturas de pantalla) en el informe del laboratorio y explique porque ud. piensa que su ataque fue exitoso. En particular, cuando corra el primer comando `dig` use Wireshark para capturar el tráfico de red y señalar que paquetes son disparados por este comando `dig`. Use el trackeo del paquete para probar que su ataque es exitoso. Note que los resultados en el servidor de DNS local pueden estar cacheados después de que correr por primera vez el comando `dig`. Esto puede influenciar en los resultados si ud. corre el comando `dig` antes de usar Wireshark. Puede limpiar la caché usando "`sudo rndc flush`" en el servidor de DNS local, pero esto requerirá que ud. lance nuevamente el ataque.

4 Guías

Para implementar el ataque de Kaminsky, podemos usar Scapy para hacer spoofing de paquetes. Desafortunadamente la velocidad de Python es demasiado lenta; el número de paquetes generados por segundo son insuficientes para que el ataque tenga éxito. Es mejor usa un programa hecho en C. Esto puede ser algo desafiante para mucho de los estudiantes, porque construir un paquete DNS usando C no es tan fácil. He desarrollado un método híbrido y lo hemos usado en mis clases. Usando este enfoque, el tiempo que tardan los estudiantes en desarrollar el programa se ve reducido de manera significativa, esto es beneficioso ya que pueden concentrarse mucho más tiempo en el ataque en sí.

La idea es unir las fuerzas de Scapy y C: Scapy es mucho mas conveniente en la creación de paquetes DNS que C. Hemos usado Scapy para crear paquetes DNS spoofeados y guardarlos en un archivo, para después cargar estos paquetes dentro de un programa en C. Aunque necesitamos enviar mucho paquetes DNs durante el ataque de Kaminsky, esos paquetes son bastantes similares, exepto por algunos pocos campos. Podemos usar a Scapy para generar paquetes que sirvan como base para encontrar los offsets en donde esos cambios deben de realizarse (Ejemplo: el campo de ID de Transacción) y hacer directamente el cambio sobre el. Esto será mucho más fácil que crear el paquete DNS en C. Despues de que los cambios se han realizado podemos usar raw sockets para enviar esos paquetes. Los detalles de este método híbrido son provistos en el capítulo de Packet Sniffing and Spoofing en el libro de SEED [**?**]. El siguiente programa usa Scapy para crear un paquete de respuesta DNs bastante simple y lo guarda en un archivo.

Listing 1: `generate_dns_reply.py`

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

# Construct the DNS header and payload
name   = 'twysw.example.com'
Qdsec  = DNSQR(qname=name)
Anssec = DNSRR(rrname=name, type='A', rdata='1.1.2.2', ttl=259200)
dns    = DNS(id=0xAAAA, aa=1, rd=0, qr=1,
             qdcount=1, ancount=1, nscount=0, arcount=0,
             qd=Qdsec, an=Anssec)

# Construct the IP, UDP headers, and the entire packet
ip  = IP(dst='10.0.2.7', src='1.2.3.4', chksum=0)
udp = UDP(dport=33333, sport=53, chksum=0)
pkt = ip/udp/dns

# Save the packet to a file
with open('ip.bin', 'wb') as f:
  f.write(bytes(pkt))
```

En el programa en C, cargamos este paquete desde un archivo llamado `ip.bin` y lo usamos como nuestro paquete planilla (template), a partir del mismo crearemos varios paquetes similares y inundaremos con estos paquetes spoofeados de respuesta al servidor de DNS local. Por cada respuesta, debemos de cambiar tres lugares: el ID de transacción y el nombre `twysw` que aparece en dos lugares (la sección Question y la sección Answer). El ID de transacción siempre se encuentra en el mismo lugar (offset `28` a partir del comienzo de nuestro paquete IP), pero el offset para el nombre `twysw` depende de la longitud del nombre de dominio. Podemos usar un editor binario como `bless` para visualizar el archivo binario `ip.bin` y encontrar los dos offsets donde se encuentra la cadena `twysw`. En nuestro paquete se encuentran en los offsets `41` y `64`.

El siguiente fragmento de código muestra hacer los cambios en esos campos. Cambiamos el nombre en nuestra respuesta a `bbbbb.example.com` y enviamos una respuesta DNS spoofeada con el ID de transacción que es `1000`. En este código la variable `ip` apunta al comienzo del paquete IP.

```
// Modify the name in the question field (offset=41)
memcpy(ip+41, "bbbbb" , 5);

// Modify the name in the answer field (offset=64)
memcpy(ip+64, "bbbbb" , 5);

// Modify the transaction ID field (offset=28)
unsigned short id = 1000;
unsigned short id_net_order = htons(id);
memcpy(ip+28, &id_net_order, 2);
```

Generar Nombres Aleatorios. En el ataque de Kaminsky, necesitamos generar nombres aleatorios. Hay muchas formas de hacerlo. El siguiente fragmento de código muestra como generar un nombre aleatorio de 5 caracteres.

```
char a[26]="abcdefghijklmnopqrstuvwxyz";

// Generate a random name of length 5
char name[6];
name[5] = 0;
for (int k=0; k<5; k++)
   name[k] = a[rand() % 26];
```

5 Informe del Laboratorio

Debe enviar un informe de laboratorio detallado, con capturas de pantalla, para describir lo que ha hecho y lo que ha observado. También debe proporcionar una explicación a las observaciones que sean interesantes o sorprendentes. Enumere también los fragmentos de código más importantes seguidos de una explicación. No recibirán créditos aquellos fragmentos de códigos que no sean explicados.

Agradecimientos

Este documento ha sido traducido al Español por Facundo Fontana