



Seguridad en DNS con DNSSEC.

ASIGNATURA:

Redes de información (0056).

ALUMNOS:

- Gorordo, Lucas.
- Maero, Jesica.
- Miranda Martinez, Magali.

FECHA DE PRESENTACIÓN: 11/02/2021



ÍNDICE.

ÍNDICE.	1
INTRODUCCIÓN.	2
DESARROLLO.	3
1- Marco teórico.	3
1.1 ¿Qué es un servidor web?	4
1.2 ¿Qué es un proxy inverso?	4
1.3 ¿Qué es un servidor DNS?	5
1.3.1 DNSSEC.	7
1.3.1.1 ¿Qué es DNSSEC?	7
1.3.1.2 ¿En que se basa DNSSEC?	8
Registros DNSSEC.	8
Función Hash.	11
¿Qué beneficios tiene el uso de DNSSEC?	15
2- Escenario práctico.	17
2.1 Contenedor del cliente.	20
2.2 Contenedor del atacante.	20
2.3 Contenedor del servidor DNS.	20
2.3.1 Base de datos.	20
2.3.2 Claves de seguridad.	22
2.3.3 Firmado de zonas.	23
2.4 Docker compose.	23
3- Verificación de funcionamiento y seguridad	26
CONCLUSIÓN.	33
BIBLIOGRAFÍA.	34



INTRODUCCIÓN.

El objetivo del siguiente trabajo consiste en exponer la importancia de aplicar seguridad a servicios básicos utilizados, hoy en día, en internet. La globalización actual hace que sea fundamental considerar la implementación combinada de varias aplicaciones (como DNS y Web) para proporcionar un servicio integral. ¿Qué significa esto? Que los usuarios puedan navegar por internet y acceder a diferentes recursos sin preocuparse por los protocolos y procesos que corren por detrás, a tal punto, que para ingresar a un sitio web solamente debe conocerse la URL del mismo. Este nivel de abstracción sobre el funcionamiento de internet hace que la mayoría de los usuarios puedan ser presa de diferentes ataques.

Puntualmente, la meta de este trabajo radica en la implementación de servidores, tales como DNS y web, que representan una arquitectura básica en una organización. Estos constituyen los activos que se utilizan para realizar una investigación y desarrollo de seguridad. Cumplir con este objetivo, implica un análisis de las debilidades de los servicios y una inspección general de las herramientas de aplicación que permitan dar soluciones, en este caso, proxy inverso para servidor web y DNSSEC para servidor DNS, dando mayor importancia a éste último, corroborando su eficiencia mediante un tipo de ataque ("man in the middle combinado con modificación de respuesta a solicitudes DNS), de los tantos existentes.

DESARROLLO.

El desarrollo del presente trabajo final consta de tres grandes partes: La primera de ellas, marco teórico, consiste en la presentación de conceptos necesarios para la comprensión de la arquitectura de red, y su funcionamiento, sobre la cual se aplicará DNSSEC que es el tema principal de este proyecto.

Luego, en la segunda etapa, escenario práctico, se plasmarán las configuraciones necesarias para la implementación del escenario propuesto, como así también, las tecnologías utilizadas y las particularidades de cada una de ellas.

Finalmente, en la verificación de funcionamiento y seguridad, se realizarán pruebas de análisis sobre el escenario desarrollado en la instancia anterior para corroborar el correcto desempeño del mismo y el cumplimiento de los objetivos antes mencionados, utilizando los conceptos del marco teórico para validar los resultados.

El fundamento de esta organización radica en una analogía con el proceso de gestión de seguridad que debe llevar adelante un administrador de red. Puntualmente, en las etapas de investigación (marco teórico), implementación (escenario práctico), evaluación (verificación de funcionamiento y seguridad) y actualización. Esta última, forma parte de las conclusiones del trabajo.

1- Marco teórico.

Antes de comenzar con la implementación práctica asociada a la seguridad, es necesario tener presente conceptos básicos que nos permitirán comprender el proyecto propuesto. La arquitectura de red mostrada en la figura 1, comprende dos servidores web (Nginx) , un servidor DNS (bind9), un proxy inverso (traefik), un cliente y un atacante.

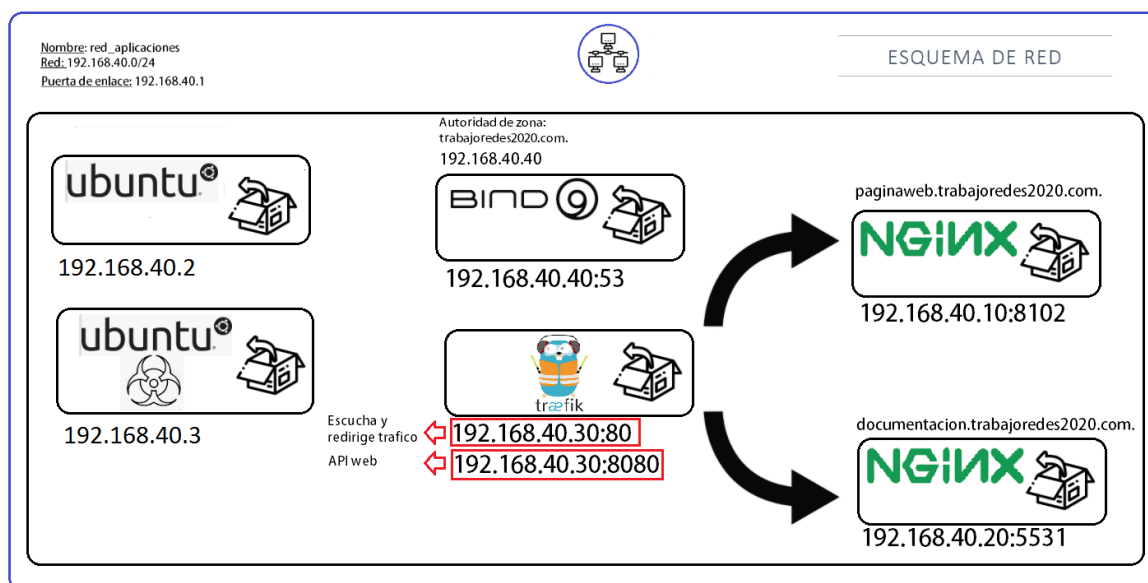


Figura 1: esquema de red.

La tecnología para la implementación es docker. Donde cada uno de los servicios se encuentra en contenedores. La razón de esta modalidad radica en que, debido a la situación actual, no se pudo hacer uso del laboratorio de redes en la universidad nacional de Río

Cuarto. Además, al utilizar una gran cantidad de servicios, la virtualización mediante herramientas como GNS3 se veía limitada.

1.1 ¿Qué es un servidor web?

Un servidor web consiste en un dispositivo de red que contiene un software dedicado a servir documentos web. Para realizar dicha tarea, necesita escuchar y atender solicitudes (peticiones) en un socket utilizando el protocolo de transporte TCP. Los sockets hacen referencia a la combinación de una dirección IP y puerto particular. El beneficio de utilizar este concepto es que permite a un host dentro de una red, ejecutar múltiples aplicaciones y comunicarse todas al mismo tiempo, ya que cada una de ellas posee un puerto particular que la identifica.

En este caso, los servidores implementados contienen las siguientes características:

- 192.168.40.10:8102 , URL: <http://paginaweb.trabajoredes2020.com./index.html>
- 192.168.40.20:5531, URL: <http://documentacion.trabajoredes2020.com./index.html>

Es necesario mencionar que la IANA define lo que se conoce como “puertos bien conocidos”, es decir puertos dedicados a aplicaciones particulares en el rango 0-1023. En el caso de los servidores web, tienen asignados el puerto 80, pero por cuestiones de seguridad, en la implementación se utilizan los puertos efímeros (mayores a 1023).

Para acceder a recursos en internet, el concepto de URL es fundamental. Básicamente, la URL es la forma en la que se especifica cómo se puede encontrar un documento web dentro de internet. Está formada por 3 partes: el protocolo de capa de aplicación (HTTP), el dominio asignado al host que contiene el recurso (paginaweb.trabajoredes2020.com. ó documentacion.trabajoredes2020.com.) y la ruta de directorios junto con el nombre de archivo (/index.html). Respecto a este último elemento, podemos mencionar que es relativo, ya que dependerá del directorio donde tenga autoridad la aplicación web. Estas particularidades están detalladas en los archivos de configuración de Nginx.

Nginx es un servidor web ligero y de alto rendimiento. Es de código abierto, lo que facilita la personalización. Es muy popular y, junto con Apache, gobiernan prácticamente todo el mercado de servidores web. [\(1\)](#)

El acceso a los sitios web generados por Nginx se realiza a través de un proxy inverso.

1.2 ¿Qué es un proxy inverso?

En términos generales, un servidor proxy es una interfaz de comunicación en una red que se hace cargo de las peticiones y las transmite en calidad de representante a un ordenador de destino, es decir, es un *intermediario entre un cliente y un servidor*. En las redes corporativas se recurre a esta estructura para que los dispositivos cliente tengan un acceso controlado a Internet. El servidor configurado como proxy se presenta, en este caso, como la única posibilidad de conexión con la red pública (internet). Se puede hablar, así, de un proxy de reenvío (forward proxy).

Mientras que un forward proxy protege los dispositivos cliente en una red, de las influencias procedentes de Internet, un proxy inverso, por el contrario, trabaja de modo que los proteja en sentido opuesto. Uno o varios servidores web activan un servidor proxy de tales características como componente de seguridad adicional para que pueda hacerse cargo de las solicitudes procedentes de Internet y las transmita a un servidor interno en segundo plano.

[\(2\)](#)

Al disponer de más de un servidor, resulta práctico y seguro acceder a ellos a través de un intermediario. En esta situación, si se dispone de un router en las periferias de la red, debería generarse una zona desmilitarizada donde es necesario la apertura de un único puerto. ¿Por qué? Porque el proxy inverso gestiona todas las solicitudes entrantes desde internet y las redirige de acuerdo al URL solicitado a los servidores correspondientes. Esta metodología proporciona una seguridad adicional, ya que las direcciones IP de los servidores web no son conocidas por agentes externos, solo se conoce la del proxy.

Existen múltiples software que permiten crear proxys inversos, sin embargo, no todos ellos son aptos para ser aplicados dentro de una arquitectura de red dockerizada. Las principales herramientas recomendadas en internet para este tipo de red, son Nginx y Traefik, siendo esta última, la seleccionada para esta implementación. ¿Por qué Traefik? porque cumple una función de proxy reverso rápido y fácil de instalar, con las siguientes características:

- Implementación de HTTPS con LetsEncrypt más simple que con NGINX o Apache.
- Integración con Docker.
- Dashboard para monitorear.
- Admite aplicar balanceo de carga como así también instalar extensiones adicionales (middlewares [\(3\)](#)).

Esta combinación de servicios (servidor web con proxy inverso) permite acceder a los recursos web siempre y cuando, los clientes conozcan la dirección IP del proxy inverso. Sin embargo, en lo cotidiano de la navegación por internet, esto no sucede. La mayoría de las veces los clientes solo conocen la URL, por lo tanto, por medio del navegador acceden a los sitios web. ¿Cómo hacen los navegadores para conocer la dirección IP? tienen integrado un resolver DNS que realiza solicitudes o comúnmente definidas como consultas a un servidor DNS.

1.3 ¿Qué es un servidor DNS?

Es un software que se ejecuta en un host dentro de una determinada red que permite proporcionar servicio de resolución de nombres de dominios (DNS). Esto significa que mediante una consulta con un nombre de dominio entrega una dirección IP asociada. Este tipo de operación se denomina resolución directa. No es la única forma de resolver un recurso accesible en internet. También, se puede obtener el nombre de dominio mediante una consulta que tenga como dato la dirección IP, comúnmente conocida como resolución inversa.

Para realizar ambas tareas, el sistema utiliza una base de datos que contiene información de cada dominio con sus respectivas IPs. Resulta imposible asociar este concepto con lo que ocurre en la realidad, porque el crecimiento abrupto de internet en los últimos años, generó que la cantidad de sitios web, emails y recursos de internet imposibilite centralizar toda esta información en un único servidor. Por este motivo, DNS es un sistema globalmente descentralizado, jerárquico y escalable.

En este contexto, cada servidor DNS autoritativo mantiene una base de datos actualizada con toda aquella información de los dominios de su zona de autoridad, organizada en diferentes Resources Records o RRs. Un Resource Records es una línea en la base de datos que contiene información como dirección IP y dominio, siendo estas las más importantes.

Cuando se habla de autoridad de zona, se hace referencia a que DNS posee una estructura jerárquica en forma de árbol, donde los dominios de nivel superior delegan autoridad sobre diferentes subdominios. La jerarquía comienza en la zona raíz "." siendo el nivel más alto. Aunque normalmente no es mostrado, todo dominio completo termina en un punto final "."

que indica el final del espacio en la zona raíz. Por ejemplo “paginaweb.trabajoredes2020.com” realmente es “paginaweb.trabajoredes2020.com.”, donde el punto final más a la derecha representa la zona raíz. Este dominio completo es lo que se denomina Fully Qualified Domain Name (FQDN).

De esta forma, las etiquetas que constituyen un dominio se construyen de derecha a izquierda siguiendo la jerarquía en forma de árbol que presenta el sistema. Por ejemplo: “paginaweb.trabajoredes2020.com.” se resuelve de la siguiente manera:

- raíz “.”
- dominio de nivel superior TDL “.com”
- subdominio “.trabajoredes2020” (*Autoridad de zona*)
- Recurso “paginaweb”

Esto muestra que la parte más a la izquierda del dominio FQDN suele expresar un nombre de máquina o recurso final, genéricamente conocido como host.

Es necesario comentar que para lograr adquirir la autoridad sobre un determinado dominio, se debe disponer de dos servidores autoritativos. Un servidor denominado Maestro o primario y un segundo servidor, denominado Esclavo o secundario. En los maestros, o primarios, se guardan y administran las versiones definitivas de los registros que son transferidas a servidores autoritativos esclavos, que guardan una copia que es actualizada cada vez que se produce un cambio. Esta actualización se conoce como transferencia de zona. En definitiva, el servidor esclavo o secundario es un respaldo frente a posibles fallas que tenga el primario. Como existen servidores autoritativos también existen servidores no autoritativos (caché). Los servidores DNS cachés almacenan información de consultas (queries) DNS por un determinado tiempo denominado TTL (Time To Live) de cada registro DNS, optimizando el uso de red, reduciendo el tráfico DNS en Internet puesto que almacenan los registros consultados, pudiendo ofrecerlos directamente sin tener que repetir la consulta recursiva. Igualmente, reducen la carga sobre los servidores autoritativos, especialmente los de la zona raíz o root servers.

Hay dos formas por las cuales un software cliente de DNS (resolver) puede realizar consultas a un servidor DNS, ya sea autoritativo o no. Por un lado, en una **consulta recursiva**, el servidor de nombres busca en un servidor de nombres externo la consulta si es que no la tiene registrada en caché y en su base de datos. Es decir, demanda que el servidor lance, a su vez, una consulta para determinar la información solicitada y luego devolvérsela al cliente. Por otro lado, en una **consulta iterativa**, el servidor de nombre debería devolver la información que disponga, además de una lista de servidores adicionales con los que el cliente (resolver) puede comunicarse para completar su consulta. Es decir, el servidor DNS no tiene la responsabilidad de obtener una respuesta.

La principal debilidad en cuanto a seguridad de la que adolece DNS tiene su origen directo en el uso del protocolo UDP para transmitir mensajes. UDP es un protocolo de transporte de red en el que prima la velocidad de la transmisión y sobre el cual se envía y recibe la información sin que se haya establecido previamente una conexión y sin confirmación ni control de entrega/recepción de la misma. Esto posibilita el falseo de direcciones IP (IP spoofing) y la suplantación de mensajes de consulta/respuesta. Aunque en DNS se contempla el uso de TCP para la transmisión de mensajes, en las especificaciones de implementación se recomienda, por motivos de rendimiento, usar UDP en las consultas. Se sugiere limitar el uso de TCP para transacciones de transferencias de zona o para aquellas consultas que superan el tamaño máximo, establecido en 512 bytes en mensajes sobre UDP. Dada la

carencia de control/confirmación en las transmisiones UDP, la responsabilidad final de validar un mensaje recaerá directamente sobre el protocolo DNS.

Paralelamente al problema del uso del protocolo UDP en el transporte de mensajes DNS, se añaden debilidades de diseño en el aspecto de la identificación y validación de los paquetes que favorecen la falsificación de los mismos. Existen múltiples ataques a servidores DNS como lo son: *DNS cache poisoning* (4), *denegación de servicio por amplificación DNS* (5), *Man in the Middle junto con DNS spoofing*, entre otros. En particular, para la implementación práctica se desarrollará el ataque de tipo Man in the Middle dentro de una red LAN con el propósito de interceptar las consultas DNS y sus respuestas, para detectar estas últimas y poder modificarlas.

Como solución a un conjunto de vectores de ataque, se implementa DNSSEC.

1.3.1 DNSSEC.

1.3.1.1 ¿Qué es DNSSEC?

DNSSEC, siglas en inglés de Domain Name System Security Extensions, es considerado un mecanismo eficaz para evitar el spoofing y manipulación de mensajes en el protocolo DNS, y por extensión, proporcionar una vía de protección contra ataques de caché poisoning y similares.

DNSSEC se basa en una infraestructura de criptografía de clave pública PKI y en el uso de firmas digitales para establecer autenticidad de las fuentes y la validez de los mensajes. Aplicado a las queries DNS, asegura la integridad de los mensajes y la autenticidad de la fuente emisora.

La criptografía asimétrica (en inglés asymmetric key cryptography), también llamada criptografía de clave pública (en inglés public key cryptography) o criptografía de dos claves (en inglés two-key cryptography), es el método criptográfico que usa un par de claves para el envío de mensajes. Las dos claves pertenecen a la misma persona que recibirá el mensaje. Una clave es pública y se puede entregar a cualquier persona, la otra clave es privada y el propietario debe guardarla de modo que nadie tenga acceso a ella. Además, los métodos criptográficos garantizan que esa pareja de claves sólo se puede generar una vez, de modo que se puede asumir que no es posible que dos personas hayan obtenido casualmente la misma pareja de claves.

Si una persona que emite un mensaje a un destinatario, usa la llave pública de este último para cifrarlo; una vez cifrado, sólo la clave privada del destinatario podrá descifrar el mensaje, ya que es el único que debería conocerla. Por tanto se logra la confidencialidad del envío del mensaje y es extremadamente difícil que lo descifre alguien, salvo el destinatario. Cualquiera, usando la llave pública del destinatario, puede cifrarle mensajes; los que serán descifrados por el destinatario usando su clave privada.

Por otro lado, si el propietario del par de claves usa su clave privada para cifrar un mensaje, cualquiera puede descifrarlo utilizando la clave pública del primero. En este caso se consigue la identificación y autenticación del remitente, ya que se sabe que solo pudo haber sido él quien empleó su clave privada (salvo que un tercero la haya obtenido). Esta idea es el fundamento de la firma digital, donde jurídicamente existe la presunción de que el firmante es efectivamente el dueño de la clave privada.

El problema de estas comunicaciones es que aunque identifiquemos al emisor, ¿cómo podemos confiar en él? ¿Cómo sabemos que se puede confiar en la clave pública publicada?

En el caso de servidores DNS que utilizan DNSSEC se valida la clave pública de una fuente con una cadena de verificaciones que empieza en un servidor de confianza (como un root server) y bajando por la jerarquía del espacio de nombres DNS verificando sucesivamente la firma de la clave pública de un nodo hijo por su nodo padre. La clave pública de servidores de confianza se denomina “trust anchor”. Este procedimiento permite a un servidor DNS que implementa DNSSEC, utilizar un conjunto especial de registros de recursos RRs, registros específicos de firma (RRSIGs) y registros de clave DNSKEY, que le brindan a un resolvers con capacidades DNSSEC comprobar lo siguiente:

- Autenticidad de origen: Autenticar que los datos recibidos sólo pueden proceder de la zona solicitada
- Integridad: Verificar la integridad de los datos, es decir, que los datos no han sido modificados en el transcurso de la transacción.
- No existencia: Verificar, en el caso de una respuesta de dominio no existente (NXDOMAIN), que, efectivamente el registro no existe en la zona solicitada y no ha sido expresamente eliminado en la intercepción de la transacción.

Sin embargo, el resolver no tiene como única tarea comprobar que la clave pública es realmente auténtica. Una vez realizada esta verificación de clave pública de la fuente, el siguiente paso en DNSSEC es autenticar la respuesta. En este caso, las respuestas incluyen no sólo los registros solicitados, sino además, la firma digital de un conjunto de registros encapsulada en un tipo de registro específico, denominado RRSIG. Entonces, el resolver, usando la clave pública verificada anteriormente, comprueba la validez de la firma y se asegura de que la respuesta es auténtica. En el caso de una respuesta negativa indicando la no existencia de un registro, se adjunta un registro específico denominado NSEC con su firma correspondiente, cuya verificación asegura la validez de la respuesta y que el registro no ha sido eliminado por una manipulación intermedia.

1.3.1.2 ¿En que se basa DNSSEC?

En DNSSEC existen dos procesos principales: firmar y servir, y verificar firma. Estos procesos, se realizan a través de mecanismos basados en criptografía de clave pública. Aunque operativamente no es necesario más que un par de claves pública/privada, es muy común utilizar al menos dos pares para facilitar las tareas de renovación de claves y de re-firmado de zonas. Además, la “separación de claves” es una buena práctica criptográfica, para limitar el alcance de un posible compromiso. De este modo, DNSSEC cuenta con dos pares de claves pública/privada. Un par denominado Key Signing Key (KSK) para el firmado de registros de clave DNSKEY y otro, denominado Zone Signing Key (ZSK) para el firmado de registros (RRsets) .

Registros DNSSEC.

DNSSEC dispone de los siguientes registros específicos para su funcionamiento:

Maero Jesica – Miranda Martinez Magali – Gorordo Lucas

→ *RRSIG* (*Resource Record Signature*). Registro de firma.

Contiene la información de un conjunto de registros DNS del mismo tipo y la firma del mismo (creada con la clave privada).

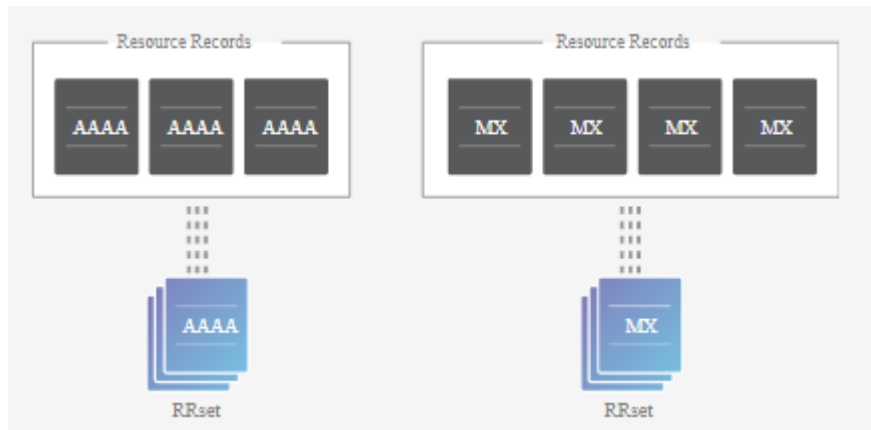


Figura 2: Ejemplos de RRSet.

De hecho, el RRset [registros del mismo tipo (A , AAAA, MX) y con la misma etiqueta (paginaweb.trabajoredes2020.com.)] completo es lo que se firma digitalmente, en lugar de los registros DNS individuales. Por supuesto, eso también significa que debes solicitar y validar todos los registros del mismo tipo y etiqueta que se solicitó para una determinada zona, en lugar de validar solamente uno de ellos.

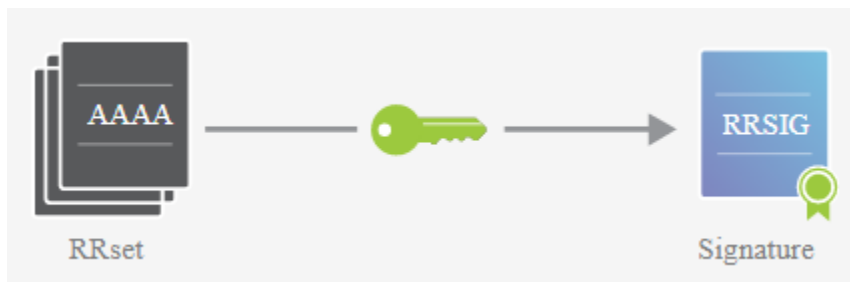


Figura 3: RRSig (RRSet + ZSK privada)..

Cada zona de DNSSEC tiene un par de claves de firma de zona (ZSK, por sus siglas en inglés): la parte *privada* de la clave firma digitalmente cada RRset de la zona, mientras que la parte *pública* verifica la firma.

Para habilitar DNSSEC, un operador de zona crea firmas digitales para cada RRset usando la ZSK privada y la almacena en su servidor de nombres como registros RRSIG. Es como decir: "Estos son mis registros DNS, provienen de mi servidor y deben ser así".

Sin embargo, estos registros RRSIG son inútiles a menos que los solucionadores de DNS tengan una forma de verificar las firmas. El operador de zona también tiene que proveer su ZSK pública añadiéndola a su servidor de nombres en un registro DNSKEY.

→ *DNSKEY*. Firma pública. Es usada para verificar las firmas adjuntas en los registros RRSIG.

Cuando un solucionador DNSSEC solicita un tipo de registro en particular (p. ej., AAAA), el servidor de nombres también devuelve el RRSIG correspondiente. Luego, el solucionador

puede extraer el registro DNSKEY que contiene la ZSK pública del servidor de nombres. Juntos, el RRset, el RRSIG y la ZSK pública pueden validar la respuesta.

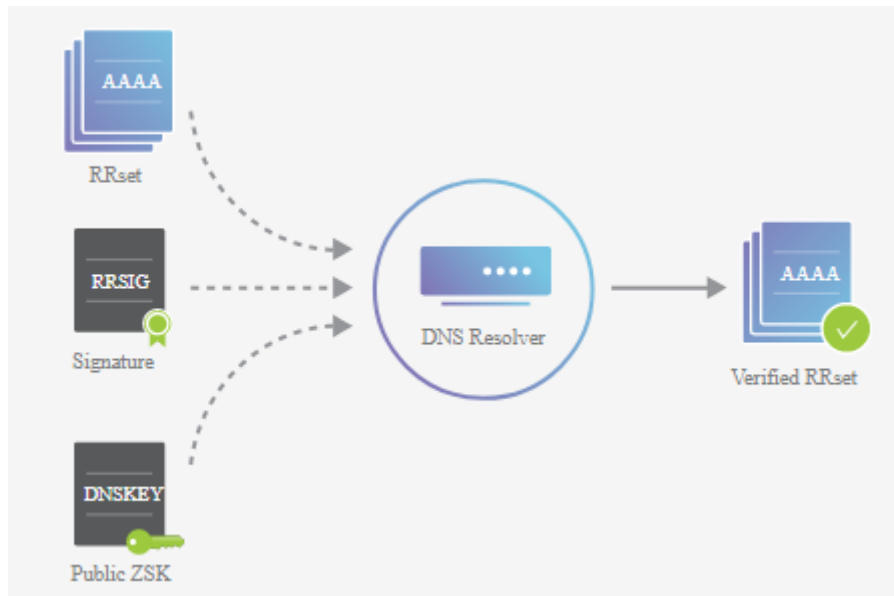


Figura 4: Validación de respuesta..

Si confiamos en la clave de firma de zona del registro DNSKEY, podemos confiar en todos los registros de la zona. Pero, ¿si la clave de firma de zona queda comprometida? **Necesitamos poder validar la ZSK pública.**

Por esta razón surge la **KSK (key-signing key)**. Además de una clave de firma de zona, los servidores de nombres de DNSSEC también tienen una clave de firma de clave (KSK, por sus siglas en inglés). La KSK valida el registro DNSKEY exactamente de la misma manera que nuestra ZSK aseguró el resto de nuestros RRsets : firma la ZSK pública (que se almacena en un registro DNSKEY), creando un RRSIG para la DNSKEY.



Figura 5: ZSK y KSK públicas, firmadas por la KSK privada.

Al igual que la ZSK pública, el servidor de nombres pública la KSK pública en otro registro DNSKEY, lo que nos da el RRset de DNSKEY que se muestra en la fig 5. Tanto la KSK pública como la ZSK pública están firmadas por la KSK privada.

Los solucionadores pueden usar la KSK pública para validar la ZSK pública.

La validación de los solucionadores consiste ahora en lo siguiente:

1. Se solicita el RRset deseado, que también devuelve el registro RRSIG correspondiente.
2. Se solicitan los registros DNSKEY que contienen la ZSK pública y la KSK pública, que también devuelve el RRSIG para el RRset de DNSKEY.
3. Se verifica el RRSIG del RRset solicitado con la ZSK pública.
4. Se verifica el RRSIG del RRset de DNSKEY con la KSK pública.

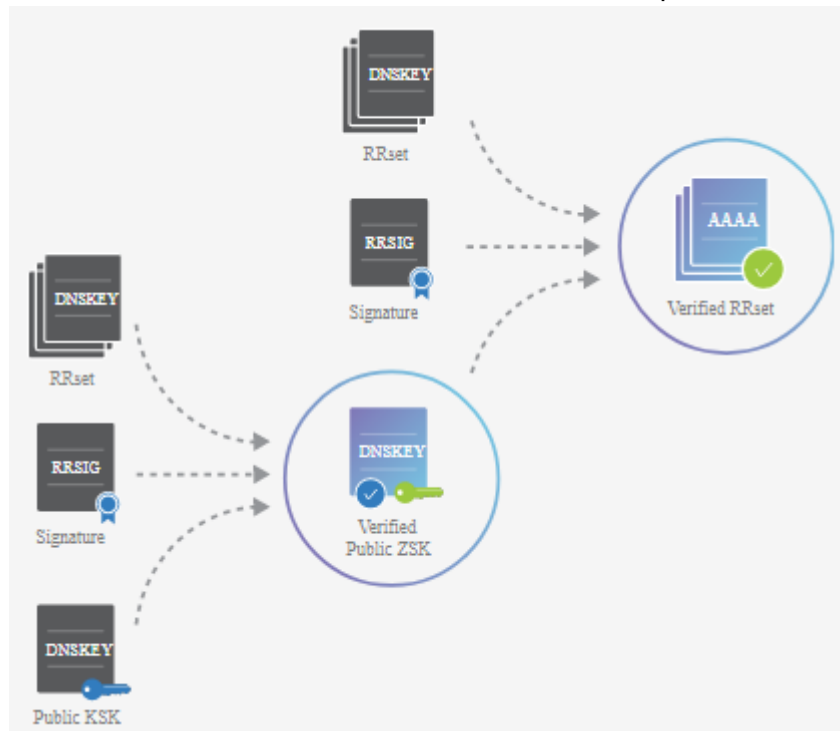


Figura 6: Validación de los solucionadores .

El DNS es un sistema jerárquico, y las zonas no suelen operar de forma independiente. Para complicar más las cosas, la clave de firma de clave (KSK) se firma a sí misma, lo que no proporciona ninguna confianza adicional. Hace falta una manera de conectar la confianza de nuestra zona con su zona principal.

DNSSEC introduce un registro de Delegation Signer (DS) para permitir la transferencia de confianza desde una zona principal a una zona secundaria. Un operador de zona realiza una función hash en el registro DNSKEY que contiene la KSK pública y se lo entrega a la zona principal para publicarlo como registro DS.

Función Hash.

Una función criptográfica hash usualmente conocida como “hash”, es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija. Esto significa que independientemente de la longitud de los datos de entrada, el valor hash de salida tendrá siempre la misma longitud, como se puede observar en la figura 7.

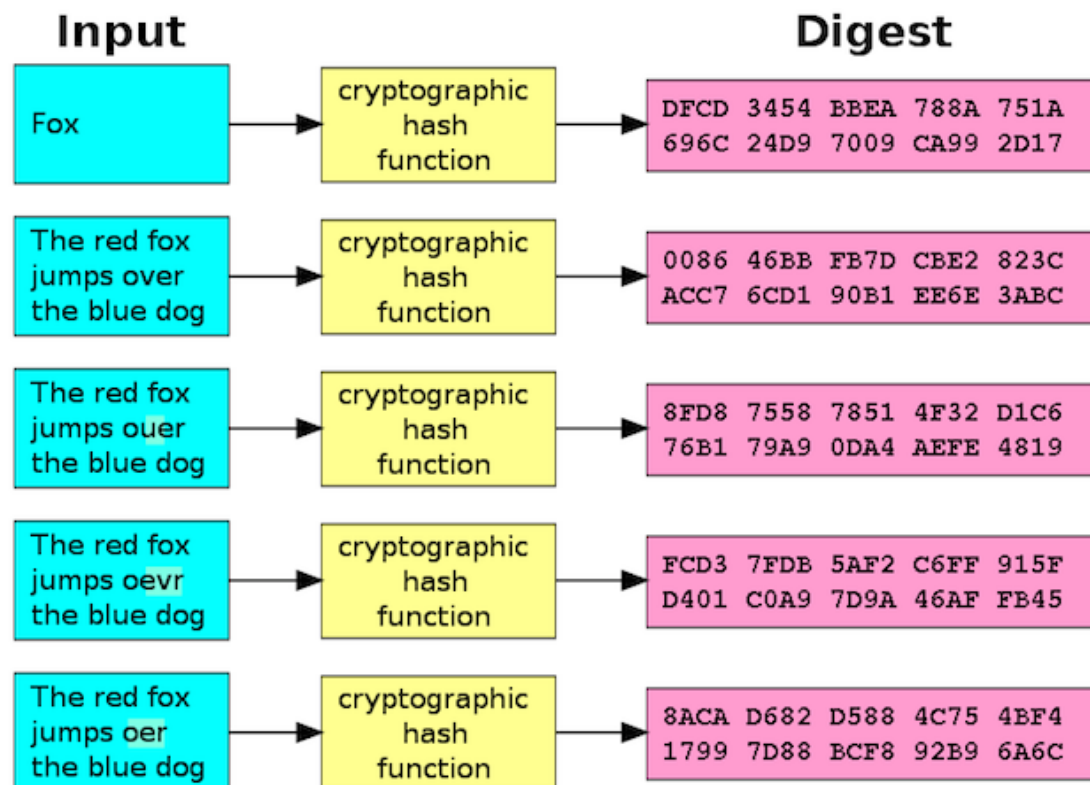


Figura 7: Ejemplos de función "Hash".

Las funciones criptográficas hash se utilizan también para asegurar la "integridad de los mensajes". En pocas palabras, para asegurarse de que algunas comunicaciones o archivos no fueron alterados, se pueden examinar los hash creados antes y después de la transmisión de los datos. Si los dos hash son idénticos, significa que no ha habido ninguna alteración.

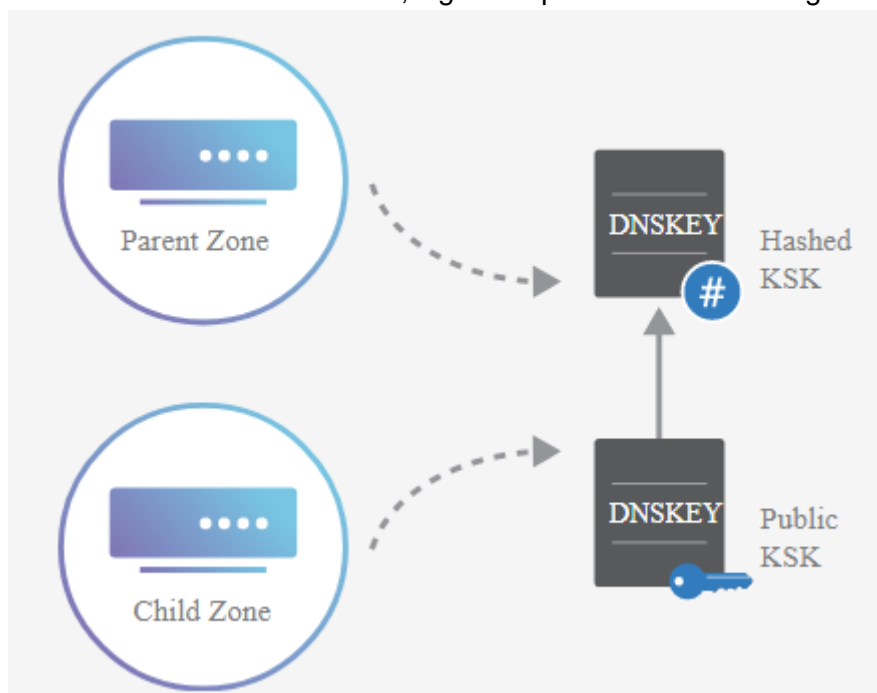


Figura 8: Verificación con función Hash.

→ DS. (Delegation Signer).

Contiene un hash de la clave pública de un nodo hijo. Para tener la certeza de que, en una zona, la clave pública (DNSKEY) y los registros de firma que la misma verifica no han sido manipulados, se genera un hash de la clave pública que se entrega al nodo inmediatamente superior. Este nodo genera el registro DS almacenando ese hash y lo firma con su clave privada, obteniendo el registro RRSIG correspondiente. Esta cadena continúa hacia arriba en la jerarquía hasta llegar al último nodo de confianza en la cadena, típicamente el nodo raíz. Cada vez que un solucionador se dirige a una zona secundaria, la zona principal también proporciona un registro DS. Este registro DS es la forma en que los solucionadores saben que la zona secundaria está habilitada para DNSSEC. Para comprobar la validez de la KSK pública de la zona secundaria, el solucionador realiza una función hash y la compara con el registro DS de la principal. Si coinciden, el solucionador puede asumir que la KSK pública no ha sido alterada, lo que significa que puede confiar en todos los registros de la zona secundaria. Así es como se establece una cadena de confianza en DNSSEC. Debe tenerse en cuenta que cualquier cambio en la KSK también requiere un cambio en el registro DS de la zona principal. El cambio del registro DS es un proceso con varios pasos que puede terminar rompiendo la zona si se realiza de forma incorrecta. Primero, la zona principal tiene que agregar el nuevo registro de DS, luego tiene que esperar hasta que el TTL del registro original de DS caduque antes de quitarlo. Por eso es mucho más fácil cambiar las claves de firma de zona que las claves de firma de clave.

De esta manera expresamos una forma de establecer la confianza dentro de una zona y conectarla a su zona principal, *pero ¿cómo confiamos en el registro DS?* El registro DS está firmado como cualquier otro RRset, lo que significa que tiene un RRSIG correspondiente en la zona principal. Todo el proceso de validación se repite hasta llegar a la KSK pública de la zona principal. Para verificarlo, tenemos que ir al registro DS principal y a continuación ascendemos en la cadena de confianza.

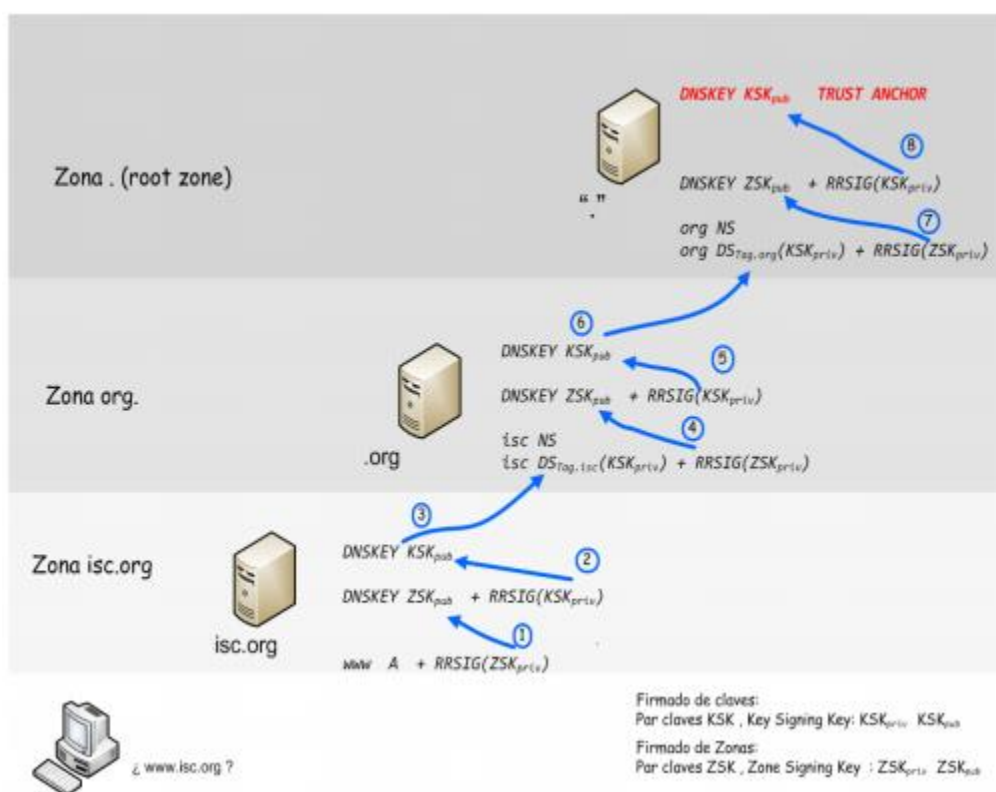


Figura 9: Validación de claves, siguiendo la cadena de confianza.

Sin embargo, cuando finalmente llegamos a la zona raíz del DNS, tenemos un problema: no hay ningún registro DS principal para validar. **Aquí es donde podemos ver un lado humano del Internet global.**

En la denominada ceremonia de firma de raíz, personas seleccionadas de todo el mundo se reúnen para firmar el RRset de DNSKEY raíz de manera pública y muy auditada. La ceremonia produce un registro RRSIG que se puede utilizar para verificar la KSK y ZSK públicas del servidor de nombre raíz. En lugar de confiar en la KSK pública por el registro DS de la zona principal, **asumimos que es válida porque confiamos en los procedimientos de seguridad para acceder a la KSK privada.**

La capacidad de establecer confianza entre la zona principal y la secundaria es una parte integral de DNSSEC. Si alguna parte de la cadena se rompe, no podemos confiar en los registros que solicitamos, porque un intermediario podría alterar los registros y dirigirnos a cualquier dirección IP que desee.

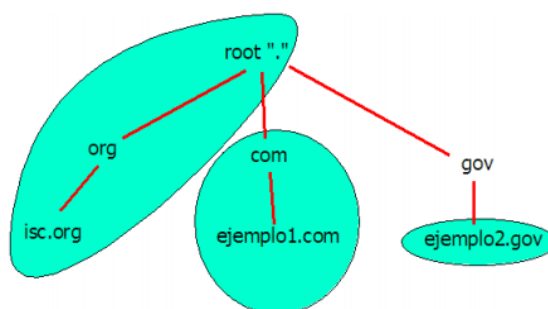


Figura 10: Cadena de confianza.

Trust anchor instalada en el resolver	Estado de la respuesta DNSSEC para consultas dirigidas a		
	www.isc.org	www.ejemplo1.com	www.ejemplo2.gov
ninguna	inseguro	inseguro	Inseguro
root	seguro	inseguro	Inseguro
.com	inseguro	seguro	Inseguro
ejemplo2.gov	inseguro	inseguro	Seguro

Figura 11: Ejemplo de anclaje de confianza.

Como se puede apreciar en las figuras 10 y 11, no es un detalle menor la configuración que tenga nuestro resolver respecto a las claves públicas KSK que considera seguras.

→ NSEC (Next Secure). Si se solicita al DNS, la dirección IP de un dominio que no existe, devuelve una respuesta vacía, no hay forma de decir de manera explícita, "lo siento, la zona que solicitaste no existe". Esto es problemático si se quiere autenticar la respuesta, ya que no hay ningún mensaje para firmar. DNSSEC lo soluciona añadiendo los tipos de registro NSEC y NSEC3. **Ambos permiten la negación de existencia autenticada.** Cuando se

solicita un registro no existente, se devuelve este tipo de registro y su correspondiente firma (RRSIG) para demostrar al resolver la no existencia del mismo. NSEC construye para el dominio sobre el que se aplica DNSSEC, un listado en orden canónico (estructurado alfabéticamente) del siguiente dominio autoritativo o punto delegado (NS) , junto con los tipos de registros presentes en los mismos. Se acompaña a cada uno de los elementos del listado, con su correspondiente registro RRSIG. Junto a su registro de firma (RRSIG), constituyen el método de verificación.

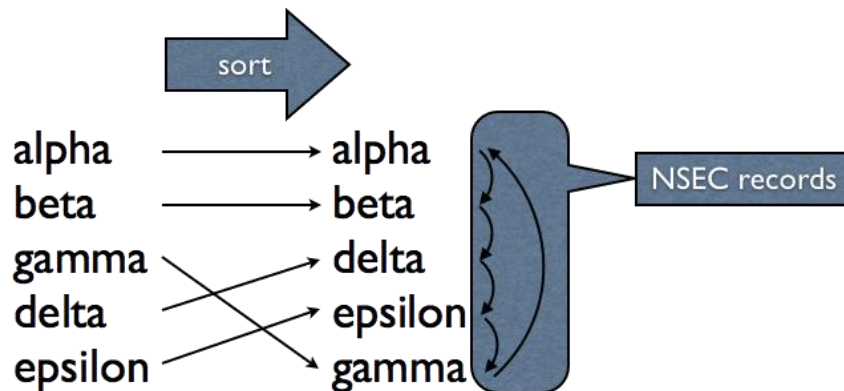


Figura 12: Ejemplo de funcionamiento de NSEC .

Si se realiza una solicitud de "charlie" para el ejemplo de la fig 12. El servidor responde con un estado NXDOMAIN y, además, proporciona un registro NSEC:

beta.example.net. NSEC delta.example.net. A RRSIG NSEC

Esto afirma que no hay un nombre válido entre "beta" y "delta". Además, establece que los únicos datos disponibles para "beta" son un registro A, RRSIG y NSEC. Dado que "charlie" se clasifica entre "beta" y "delta", esto es una prueba de que "charlie" no existe.

Este método ha sido sustituido por la versión NSEC3, debido a que NSEC por su comportamiento posibilita obtener información de zonas (enumeración) solicitando registros inexistentes.

¿Qué beneficios tiene el uso de DNSSEC?

Como se comentó anteriormente, la utilización de DNSSEC aporta una capa extra de autenticidad sobre los datos enviados mediante DNS sin extensiones de seguridad, evitando ataques de suplantación, tal como redirigir a la víctima a sitios maliciosos, como los de tipo phishing. También aumenta la protección frente a ataques de observación e interceptación del tráfico, ya que la cadena de custodia que aporta DNSSEC lo evita.

A continuación, en la figura 13 se muestra a un ciberdelincuente atacando un servidor DNS que utiliza DNSSEC. El resultado muestra el bloqueo de la página web falsa, lo que constituye una barrera importante para mitigar ataques de ingeniería social.

DNSSEC

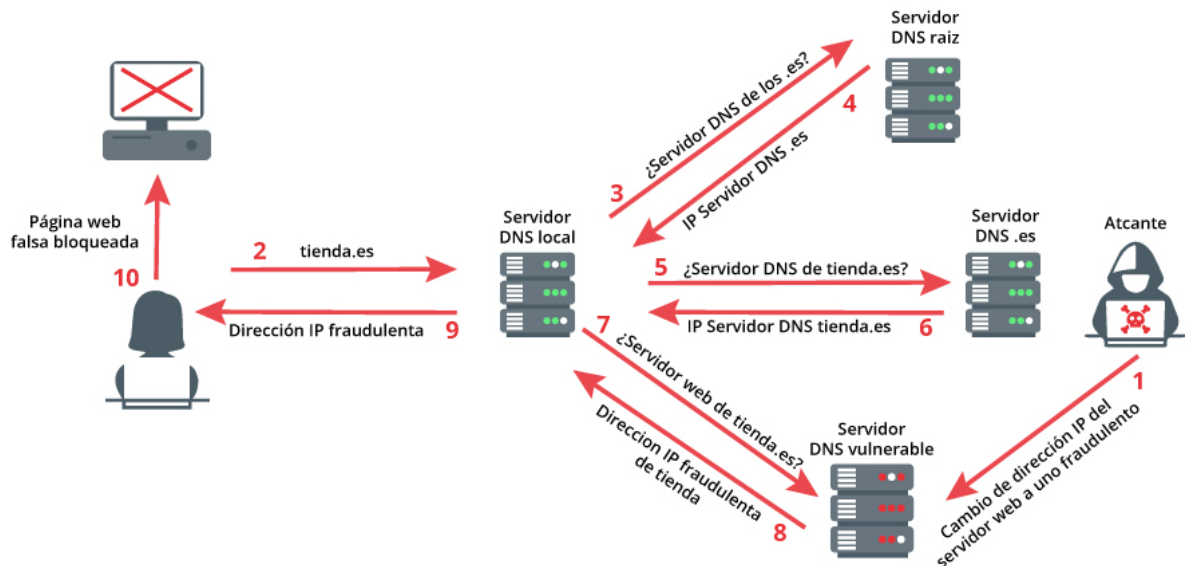


Figura 13: Ataque DNS cache poisoning.

Estos ataques (DNS Cache Poisoning) constituyen una amenaza para todo servidor DNS presente en internet. Sin embargo, existen metodologías dentro de una red LAN que tienen como finalidad interceptar el tráfico para dirigir al usuario a un sitio fraudulento. Esta modalidad se denomina Man in the Middle. El ataque Man in The Middle, o en español Hombre en el Medio, como se observa en la figura 14, consiste en la introducción de un atacante en la comunicación entre dos host para que todo el tráfico pase por él y poder así, interceptar y visualizar los datos.

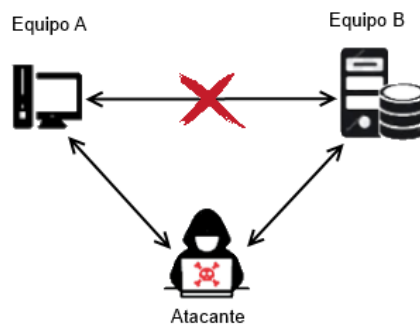


Figura 14: Ataque "Man in the middle".

Si bien este fue el objetivo principal por el cual surgió esta modalidad de ataque, ya no se utiliza porque la mayoría de las comunicaciones están encriptadas. En la actualidad, se combinan estos mecanismos con otros que permiten la modificación de los datos transmitidos para generar ataques más complejos. Uno de ellos se centra en falsificar las resoluciones DNS utilizando mecanismos de man in the middle (ARP Spoofing) como se muestra a continuación:

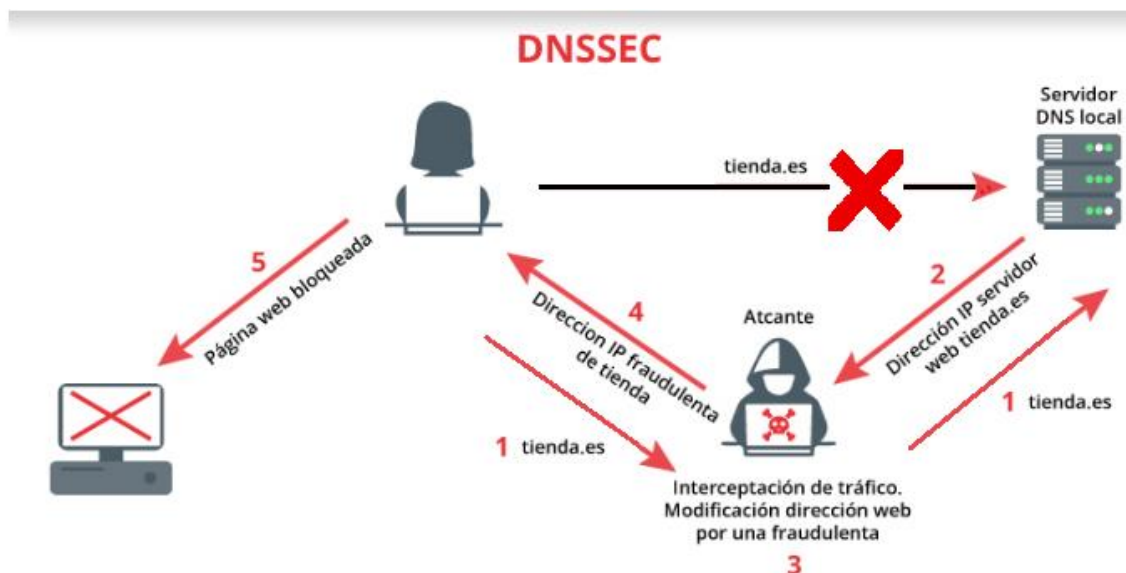


Figura 15: Ataque man in the middle con modificación de resolución DNS.

La implementación práctica presente en este trabajo se centra en este último escenario. ¿Por qué? porque con firewall, servidores proxy y zonas desmilitarizadas (DMZ), el equipamiento en seguridad de las grandes empresas para protegerse de los riesgos procedentes de Internet crece constantemente y, sin embargo, los ataques no siempre provienen del exterior. El talón de Aquiles de la cadena de seguridad es, con frecuencia, la red local o LAN. Si un atacante ha conseguido infiltrarse en la red interna, tiene generalmente todas las puertas abiertas para interceptar el tráfico de datos y manipularlo a su conveniencia. Los hackers se benefician, en este caso, de la alta vulnerabilidad que presenta el protocolo ARP (del inglés Address Resolution Protocol), usado en redes ethernet basadas en IPv4 para resolver direcciones IP en direcciones MAC, situando a los administradores, hasta hoy, ante un importante problema para la seguridad.

Las tablas ARP, que contienen las direcciones y sus equivalencias, se pueden manipular fácilmente mediante paquetes de datos falsificados. Se puede hablar en este caso de ARP spoofing o envenenamiento de tablas ARP. El ataque hace referencia a la construcción de tramas de solicitud y respuesta ARP falseadas, de forma que en una red local se puede forzar a una determinada máquina a que envíe los paquetes a un host atacante en lugar de hacerlo a su destino legítimo.

2- Escenario práctico.

Teniendo en cuenta la arquitectura de red que se presenta en la figura 16, es necesario mencionar las bondades que presenta la tecnología Docker [\(6\)](#) para realizar modelos de aplicaciones, justificando así su implementación en este proyecto.

Un conjunto de aplicaciones que conforman una arquitectura de red, no solo incluyen las aplicaciones como tal (servidor web, proxy inverso, DNS), sino que tenemos un motor, un conjunto de librerías y el kernel o software para cada una de ellas. Docker permite agrupar para cada aplicación específica, todo esto en un contenedor y almacenarlo en nuestro disco duro. La gran ventaja es que se pueden transportar todas las aplicaciones y ejecutarlas donde

se desee, sin preocuparse por el software o si se tienen todos los componentes de las aplicaciones para iniciarlas.

Desde el punto de vista del desarrollo, esta tecnología facilita el trabajo en grupo, ya que al disponer de una gran abstracción de acuerdo al entorno donde se aplica, los desarrolladores pueden centrarse en el desempeño, funcionamiento y tareas de cada una de las aplicaciones, sin perder tiempo en instalar librerías y solucionar cuestiones de compatibilidad por no disponer del mismo entorno de trabajo.

Como se mencionó al comienzo del informe, y se puede observar en la imagen 16, la arquitectura de red está conformada por los siguientes elementos: dos servidores web (Nginx), un servidor DNS (bind9), un proxy inverso (traefik), un cliente (ubuntu 18.09) y un atacante (ubuntu 18.09), donde cada uno de ellos es un **contenedor** (7).

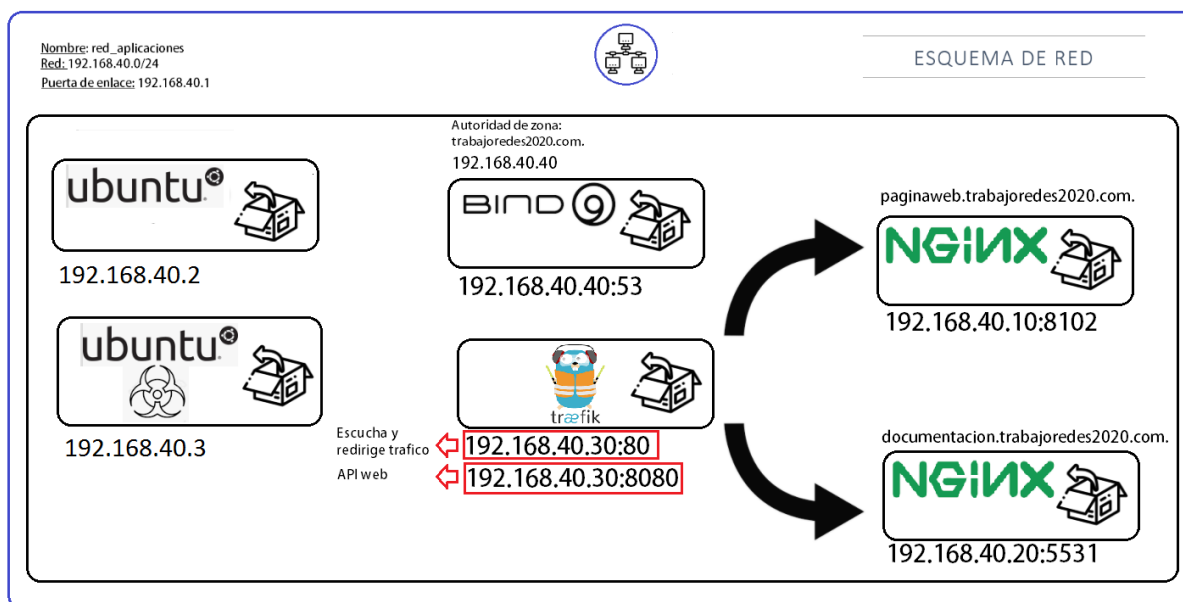


Figura 16: Arquitectura de red implementada.

Algunos de los contenedores fueron implementados mediante modificaciones de las imágenes base disponibles en el sitio oficial de docker, denominado “docker hub” y otros resultan ser, simplemente, implementaciones de imágenes prefabricadas con parámetros y volúmenes particulares.

En el primer caso, cuando se trata de una modificación sobre una imagen base, es necesario la construcción de un archivo denominado Dockerfile. Un Dockerfile es un archivo de texto plano que contiene una serie de instrucciones necesarias para crear una imagen que, posteriormente, se convertirá en una sola aplicación utilizada para un determinado propósito. Este concepto puede observarse en la figura 17.

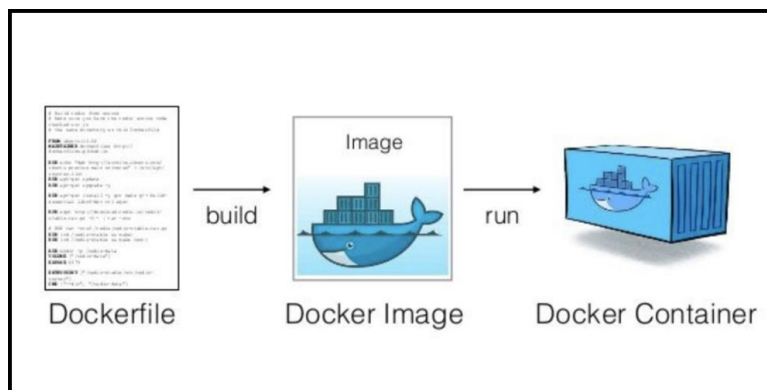


Figura 17: Generación de un contenedor por modificaciones de imagen base.

Los contenedores que utilizan estos conceptos son: **Atacante**, **Usuario** y **DNS**. Algunos de ellos, poseen un archivo dockerfile más extenso que otros, porque sus funcionalidades así lo requieren. Por lo tanto, a modo de presentación, se expone a continuación, en la figura 18, una captura del archivo dockerfile correspondiente al contenedor del usuario:

```
1 FROM ubuntu:18.04
2
3 RUN apt-get update
4
5 RUN apt install net-tools
6
7 RUN apt install -y dnsutils
8
9 RUN apt install -y curl
10
11 RUN apt-get install -y iputils-ping
12
13 RUN apt-get install -y nano
14
15 CMD ["/bin/bash", "-c", "while ;; do sleep 10; done"]
```

Figura 18: dockerfile del contenedor del usuario.

Sin entrar en profundo detalle, es fácil identificar que cada línea del archivo de configuración se basa en un comando y un argumento. Para este caso, se utilizan : **FROM**, **RUN** y **CMD**. Los comandos FROM, tienen como argumento las imágenes base utilizadas de docker hub. Luego, se utiliza el comando RUN para ejecutar instrucciones sobre la imagen base del contenedor. Es necesario saber, que si se utiliza una imagen base con un sistema operativo “alpine” en el comando FROM, las siguientes instrucciones indicadas con el comando RUN, deben obedecer a dicho sistema operativo. **Esta es la manera de instalar dependencias necesarias para las aplicaciones del proyecto**. Finalmente, el comando CMD, permite ejecutar una instrucción antes de terminar la construcción del contenedor. En este caso, se genera un bucle infinito while para que el contenedor se mantenga vivo luego del proceso de construcción.

2.1 Contenedor del cliente.

El contenedor del cliente requiere las siguientes herramientas dentro del sistema operativo ubuntu 18.04: *net-tools* (herramientas de redes para visualizar direcciones IP , tablas de ruteo), *iputils-ping* (verificar conectividad a nivel de red dentro de la arquitectura dockerizada), *curl* (herramienta para generar solicitudes HTTP a servidores web), *dnsutils* (herramienta para generar consultas DNS). En definitiva, se elaboró un contenedor con un sistema operativo y un conjunto de herramientas para hacer pruebas de seguridad orientadas a DNS.

2.2 Contenedor del atacante.

En el caso del atacante, a diferencia del usuario, se instalan todas aquellas dependencias necesarias para llevar a cabo el ataque. Entre ellas se puede mencionar: *iptables* (configuración de firewall), *scapy* (herramienta de python para la manipulación de paquetes), *nfqueue* (herramienta que permite enviar paquetes por colas separadas y específicas), entre otras. Además, utilizando el concepto de volúmenes, se incorporan dentro del contenedor dos script python (*ARP_Spoof.py* y *DNS_Spoof.py*). El primero de ellos permite realizar envenenamiento ARP especificando las IP de los extremos (*target* y *gateway*) para constituir un ataque Man in the Middle [\(8\)](#) y el segundo permite modificar todas aquellas respuestas a solicitudes DNS que coincidan con el diccionario definido en el script [\(9\)](#).

2.3 Contenedor del servidor DNS.

Para el caso del contenedor que representa el servidor DNS, hay que mencionar algunas particularidades. Si bien el archivo dockerfile sigue la misma lógica que el resto, se ejecutan una mayor cantidad de comandos y más complejos, porque la construcción de este último implica la automatización de las siguientes tareas:

- Instalación de bind9 junto con las dependencias necesarias para realizar firmado en línea mediante DNSSEC.
- Creación de claves ZSK y KSK en una carpeta específica dentro del contenedor.
- Creación de una copia de la KSK pública en una carpeta específica dentro del contenedor, para utilizarla como anclaje de confianza en las pruebas de seguridad.
- Gestión de los archivos de configuración y bases de datos de la zona para obtener un servidor DNS autoritativo. Utilizando volúmenes, es posible incorporar dentro de un contenedor los siguientes archivos: *named.conf.local*, *named.conf.options*, *trabajoredes2020.com-db* (base de datos para resolución directa) y *192.168.40-db* (base de datos para resolución inversa)

2.3.1 Base de datos.

Respecto al último punto, debe hacerse mucho hincapié, porque dichos archivos contienen la configuración necesaria para dar de alta DNSSEC mediante firmado en línea. En primer lugar, hay que habilitar DNSSEC en el archivo de *named.conf.options* mediante las siguientes líneas de configuración:


```
//Lineas de DNSSEC
//=====

    dnssec-enable yes;
    dnssec-validation auto;

//=====

};
```

Figura 19: Habilitación de DNSSEC.

Luego, se procede a la creación de las claves de seguridad para el firmado de la zona. Sin embargo, no debe restarse importancia a la base de datos presente en el servidor. Esto quiere decir, que para aplicar DNSSEC es necesario un servidor DNS funcional, que resuelva los nombres de dominio correspondientes a la autoridad de zona.

```
5 ;Configuracion de la base de datos de bind para resolver nombres
6 ;
7 ;
8 $TTL 604800
9 @ IN SOA trabajoredes2020.com. magajsluc.trabajoredes2020.com. (
10     4 ; Serial - recordar que por cada cambio en este archivo hay que incrementarlo
11     604800 ; Refresh
12     86400 ; Retry
13     2419200 ; Expire
14     604800 ) ; Negative Cache TTL
15 ;
16 ; Registros NS
17 @ IN NS trabajoredes2020.com.
18 ;
19 ; Registro A para NS declarado
20 ;
21 @ IN A 192.168.40.40
22 ;
23 ;Registro A para servidores web
24 ;
25 paginaweb IN A 192.168.40.30
26 ;
27 documentacion IN CNAME paginaweb.trabajoredes2020.com.
```

Figura 20: Base de datos del servidor DNS.

Como se puede apreciar en la imagen anterior (20), la base de datos está compuesta por 5 Resources Records. El registro del tipo **SOA**, especifica que el servidor DNS tiene autoridad sobre el dominio “trabajoredes2020.com.”. Presenta algunos campos adicionales a diferencia del resto de registros, entre los cuales se puede mencionar:

- **correo electrónico** (*magajsluc.trabajoredes2020.com.= magajsluc@trabajoredes2020.com.*).
- **número de serie** (*Serial*): parámetro útil que indica la versión del archivo.
- **Tiempo de actualización** (*Refresh*): Tiempo que espera un servidor de nombres secundario para ver si el archivo ha cambiado, y por lo tanto pedir una transferencia de zona.
- **Tiempo de reintento** (*Retry*): Tiempo que espera un servidor de nombres secundario para iniciar una nueva transferencia de zona en el caso de que falle este procedimiento.

- **Tiempo de vida (TTL):** Indica el tiempo máximo de validez de la información presente en la base de datos, tras el cual deberá refrescarse o actualizarse (para comprobar que no haya cambiado).

El registro **NS** indica cuales son los servidores DNS que atienden las solicitudes sobre los recursos del dominio “trabajoredes2020.com.”. Esta información se encuentra entre las líneas 15 y 22 del archivo “trabajoredes2020.com-db”, indicando que el servidor de nombres se encuentra en la dirección IP 192.168.40.40.

Por último, se observan los registros del tipo **A** o de direcciones y los registros **CNAME**. Los primeros, vinculan un dominio con la dirección IP física de un host en el que se alojan los servicios de ese dominio y los segundos, enlazan el nombre de los dominios, con diferentes alias, es decir, con otros nombres que también redirigen al mismo contenido. ¿Por qué utilizar CNAME? porque el servidor proxy inverso resuelve las consultas HTTP para ambos servidores web en la dirección IP 192.168.40.30 independientemente del dominio en cuestión, lo que hace que sea totalmente innecesario definir dos registros del tipo A hacia la misma dirección IP. Es más sencillo definir un registro del tipo A y otro del tipo CNAME que apunte al primero, de esta forma al cambiar la dirección IP del servidor proxy inverso, solo hay que modificar un RR en la base de datos.

2.3.2 Claves de seguridad.

BIND 9 proporciona una utilidad denominada “dnssec-keygen” que simplifica la generación de claves para su uso en DNSSEC. Los comandos que se implementan para la creación de las claves ZSK y KSK son los siguiente:

```
→ dnssec-keygen -a [RSASHA256 | NSEC3RSASHA1 | ECDSAP256SHA256] -b 1024  
trabajoredes2020.com
```

```
→ dnssec-keygen -a [RSASHA256 | NSEC3RSASHA1 | ECDSAP256SHA256] -b 2048 -fk  
trabajoredes2020.com
```

Donde el parámetro “a” determina el algoritmo de firma utilizado, el parámetro “b” especifica el tamaño de la clave en bits y el último argumento, el nombre de la zona a firmar, que se obtiene del archivo *named.conf.local*. Sin embargo, los comandos aplicados para la creación de las claves ZSK y KSK se encuentra en el dockerfile y tiene un parámetro adicional “K”, que indica el directorio donde se desea almacenar las claves generadas, como se observa en la figura 21.

```
RUN dnssec-keygen -a RSASHA256 -b 1024 -K /etc/bind/keys/trabajoredes2020.com trabajoredes2020.com  
RUN dnssec-keygen -a RSASHA256 -b 2048 -K /etc/bind/keys/trabajoredes2020.com -fk trabajoredes2020.com
```

Figura 21: Generación de claves.

¿Por qué se utiliza una carpeta específica para guardar las claves? Porque se recomienda como buenas prácticas para facilitar el proceso de firmado de la zona, ya que debe indicarse su ubicación en los archivos de configuración. Una vez generada las claves, se procede a firmar.

2.3.3 Firmado de zonas.

El firmado de la zona puede ser manual o automático. Para este trabajo, se implementó un firmado de zona automático, denominado *inline signing*.

BIND 9.9 introdujo la funcionalidad inline signing mediante la cual, el software crea una versión interna de la zona que se firma en tiempo real en memoria y que es la que se sirve de cara a las consultas DNS, manteniendo intacta en el sistema de ficheros, la versión de la zona sin firmar, que no es servida directamente. Las modificaciones de la zona se realizan sobre el fichero sin firmar, lo que simplifica su administración.

Esta funcionalidad, se habilita definiendo el parámetro "*inline-signing yes*" en el fichero de configuración de la aplicación bind9. La opción de inline signing, suele ir ligada a la gestión de la zona DNSSEC de forma automática (parámetro "*auto-dnssec maintain*"), aunque conceptualmente no representen lo mismo.

Para llevar a cabo el proceso de firmado automático de la zona encargada de la resolución directa (no se firmó la zona de resolución inversa), mediante inline signing, se implementa en el archivo "named.conf.local" la siguiente configuración :

```
zone "trabajoredes2020.com" {
    type master;
    file "/etc/bind/zones/trabajoredes2020.com-db";
    //Lineas nuevas, configuracion DNSSEC
    //=====
    key-directory "/etc/bind/keys/trabajoredes2020.com";
    auto-dnssec maintain;
    inline-signing yes;
    //=====
```

Figura 22: Configuración para firmado de zonas.

Luego, lo único que debemos hacer es aplicar un pequeño cambio en los archivos de base de datos para que la aplicación BIND detecte los cambios y firme la zona. Por lo general, suele modificarse el serial mediante un incremento en una unidad del valor actual. Luego, se indica al software bind9, que debe volver a leer y aplicar la nueva configuración mediante "rndc reload <zona>". Tras ello, BIND se encargará de mantener la zona firmada, tanto para los nuevos registros como para la gestión y renovación de las claves.

2.4 Docker compose.

El soporte detrás de todos los contenedores, (independientemente de la forma en que son contruidos y las variables que manejan) está en el archivo "docker-compose.yml". Esta tecnología, complementaria a Docker, es una herramienta que permite simplificar el uso de Docker. A partir de archivos YAML es más sencillo crear contenedores, conectarlos, habilitar puertos, volúmenes, etc.

En vez de utilizar Docker vía una serie inmemorable de comandos bash y scripts, Docker Compose permite mediante archivos YAML, instruir al Docker Engine a realizar tareas de forma programada. Y esta es la clave, la facilidad para dar una serie de instrucciones, y luego repetirlas en diferentes ambientes. Al utilizar dicha herramienta, el único comando que debe ejecutarse para poner en marcha todo el proyecto y construir cada uno de los contenedores de forma automatizada es "docker-compose up".

A grandes rasgos, el archivo docker-compose.yml contiene las siguientes secciones:

- **versión:** Indica la versión base de compose que se está utilizando para construir cada contenedor. ¿Qué cambia elegir una versión u otra? La variedad y forma de ejecutar los comandos para construir los contenedores.
- **services:** Especifica los servicios que estarán presentes en cada contenedor. En este trabajo, la sección se compone de : *proxy_reverse*, *web_pages*, *web_doc*, *dns_bind9*, *atacante*, *usuario*. Cada uno de ellos, tiene configurado varios parámetros, entre los que se encuentran:
 - **build:** especifica la ubicación del archivo Dockerfile para el contenedor en cuestión.
 - **image:** es la alternativa al comando **build**. La presencia de esta instrucción indica la utilización de una imagen base de docker hub. Los contenedores que utilizan esta directiva son *web_pages* y *web_doc*, utilizando nginx:1.19 y *proxy_reverse*, utilizando traefik:v2.1.
 - **command:** indica que luego de la construcción del servicio, se deben ejecutar uno o más comandos particulares dentro del contenedor. Para el caso del contenedor *proxy_reverse*, se especifica la utilización de docker como engine, la utilización de API web, se deshabilita el descubrimiento automático de contenedores. ¿Cómo se conocen las instrucciones admitidas por este comando?. Mediante la documentación de la imagen en docker hub ([10](#)).
 - **ports:** Establece que se van a exponer los puertos del contenedor a determinados puertos del host. Esto permite que los servicios y contenedores sean accesibles desde fuera de la estructura dockerizada.
 - **expose:** Esta instrucción indica los puertos en los que un contenedor escucha conexiones sin exponerlos al host. Los puertos definidos mediante esta instrucción sólo son accesibles para la red dockerizada.
 - **volumes:** Establece que un directorio del sistema operativo estará presente de una dirección particular dentro del contenedor. Esto facilita realizar operaciones y cambios en el contenedor, sin tener que ingresar constantemente o trabajar dentro del mismo.
 - **network:** Establece que se utilizará una estructura de red interna asociada al grupo de contenedores donde cada uno tendrá una dirección ip diferente.

```
dns_bind9:
  build: bind9
  expose:
    - "53"
  volumes:
    - ./bind9/config/named.conf.local:/etc/bind/named.conf.local
    - ./bind9/config/named.conf.options:/etc/bind/named.conf.options
    - ./bind9/config/trabajoredes2020.com-db:/etc/bind/zones/trabajoredes2020.com-db
    - ./bind9/config/192.168.40-db:/etc/bind/zones/192.168.40-db
    - ./bind9/config/bind9:/etc/default/bind9
  networks:
    red_aplicaciones:
      ipv4_address: 192.168.40.40
  container_name: DNS

atacante:
  build: Atacante
  privileged: true
  volumes:
    - ./Atacante/script/ARP_Spoof.py:/Ataque/ARP_Spoof.py
    - ./Atacante/script/DNS_Spoof.py:/Ataque/DNS_Spoof.py
  networks:
    red_aplicaciones:
      ipv4_address: 192.168.40.3
  container_name: Atacante

usuario:
  build: Usuario
  networks:
    red_aplicaciones:
      ipv4_address: 192.168.40.2
```

Figura 23: archivo Docker-compose.yml.

- **network:** Esta sección dentro del archivo “docker-compose.yml” permite definir la red a crear, así como también el driver a utilizar. Para más detalles, en la página oficial de docker se encuentra la documentación completa [\(11\)](#).

```
networks:
  red_aplicaciones:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.40.0/24
```

Figura 24: Sección network (docker-compose.yml)..

3- Verificación de funcionamiento y seguridad

Una vez lograda la puesta en marcha de la infraestructura dockerizada mediante el comando “docker-compose up -d”, se procede a realizar pruebas para comprobar, en primer instancia, si funciona la configuración del servidor DNS con DNSSEC.

```
Creating Proxy          ... done
Creating Atacante       ... done
Creating Servidor_NGINX_WebDoc ... done
Creating Usuario        ... done
Creating Servidor_NGINX_WebPage ... done
Creating DNS            ... done
```

Figura 25: Verificación de puesta en marcha de contenedores.

Para cumplir con este objetivo, debe iniciarse el servicio de bind9 desde el interior del contenedor y ejecutar una serie de pasos que están especificados en el repositorio de github [\(12\)](#).

A continuación, se muestran los resultados de comandos de verificación de firmado DNSSEC para el software bind9.

```
root@ba2451df5f74:/etc/bind# rndc reconfig
root@ba2451df5f74:/etc/bind# ls /etc/bind/zones
192.168.40-db      trabajoredes2020.com-db.jbk      trabajoredes2020.com-db.signed.jnl
trabajoredes2020.com-db  trabajoredes2020.com-db.signed
root@ba2451df5f74:/etc/bind# rndc zonestatus trabajoredes2020.com
name: trabajoredes2020.com
type: master
files: /etc/bind/zones/trabajoredes2020.com-db
serial: 5
signed serial: 7
nodes: 3
last loaded: Tue, 02 Feb 2021 18:47:53 GMT
secure: yes
inline signing: yes
key maintenance: automatic
next key event: Tue, 02 Feb 2021 19:48:55 GMT
next resign node: trabajoredes2020.com/A
next resign time: Thu, 25 Feb 2021 06:03:42 GMT
dynamic: no
reconfigurable via modzone: no
root@ba2451df5f74:/etc/bind# rndc signing -list trabajoredes2020.com
Done signing with key 18262/RSASHA256
Done signing with key 24978/RSASHA256
```

Figura 26: Verificación de firmado.

Como se puede comprobar en la imagen, la zona está firmada (*secure*) por medio del firmado en línea (*inline signing*) y el proceso de actualizar las claves se realiza de manera automática (*key maintenance*). La gestión de claves y su renovación es un tema importante dentro de DNSSEC y existen muchas variantes y formas de hacerlo. Para más detalles, consultar la sección siete, definida como “operación”, en el siguiente documento [\(13\)](#).

Para lograr visualizar las firmas correspondientes a los registros de la base de datos se utiliza el comando: `named-checkzone -D -f raw trabajoredes2020.com trabajoredes2020.com-db.signed`. en la carpeta `/etc/bind/zones` dentro del contenedor bind.

Posteriormente, se comprueba el firmado de la zona realizando consultas DNS con la herramienta *dig* desde el contenedor del usuario.


```
root@84950d899c30:/# dig @192.168.40.40 +sigchase +trusted-key=./respaldoKeys/key paginaweb.trabajoredes2020.com A +multiline
;; RRset to chase:
paginaweb.trabajoredes2020.com. 604800 IN A 192.168.40.30

;; RRSIG of the RRset to chase:
paginaweb.trabajoredes2020.com. 604800 IN RRSIG A 8 3 604800 (
  20210305194941 20210203190743 19728 trabajoredes2020.com.
  BFHpcmZNDI3dHIQp0DKFrUm7DYGLjFS6+3f5QvW0cPKS
  y/FCNnoyhyjL/263Lnuj/npaA9WmwLrX80WmJyFjedf0
  J76HWEUZIIdrFdHA0uAeBps5GbtfhIgbfxtfEMaJ7p1nJ
  l1Lau8xbuDLQMbYXaQHmgjh+syaz9ZfByHfGupBo= )

launch a query to find a RRset of type DNSKEY for zone: trabajoredes2020.com.

;; DNSKEYset that signs the RRset to chase:
trabajoredes2020.com. 604800 IN DNSKEY 256 3 8 (
  AwEAAcvY0j3g5m/b7zSD5bsXvIDP1eJ60y9wtCbwDQli
  fFixZKd89JBnubbsDGhqoukPy3WivsohALbB+kDkxHjy
  cbTDG4o/cloSXf9oFMV0dm1K8+tc3GGALCd0KISIJ1Bx
  78FL55HD7sRRtiIY02z9w005D0uMQlVpZno9DkLNTmml
  ) ; ZSK; alg = RSASHA256 ; key id = 19728
604800 IN DNSKEY 257 3 8 (
  AwEAAEIpN0AWFm57VaGqWq9BSV0df05ZCH68RD+mLVyV
  B4Ei8eUXYpRuj8qCWLJemDIqvlbgdAokv/c14E2zoW31
  AW9HYkuW0sVXMT0ZpEBLXZv1tu+w3ZjMVyKYeLo3nKx4
  0XPq+b37RNMIEiaQMPZS27Tmn9Rlu99lhXv81Mw3Qm/I
  MNzAcn7vWwFdrDT69sUs/80jcfmbdXWF0m2o45A8Udz
  Vio21seLAKltPj+V+TNQ6VUWzjy5HvB66QWdRHBmV1HF
  zBvn9rmfPUe5JWFR7H0Cw/fVwciL3CMY4Y0n4nGLZbCx
  7poCSIs4ypd5y0Lrep70a3iQGWztYN5KIv0D0vk=
  ) ; KSK; alg = RSASHA256 ; key id = 35182

;; RRSIG of the DNSKEYset that signs the RRset to chase:
trabajoredes2020.com. 604800 IN RRSIG DNSKEY 8 2 604800 (
  20210305200743 20210203190743 19728 trabajoredes2020.com.
  HNggt1feftWwo0Nv45KIS3JaGXX3C2GESDq2dUm960L0
  b0yEy92Bj4377PZgC13C00VethNwnQ7Ft4+MHm5Lb1b
  PdBrZQMhNCmWU4mCvIYMP7PHpW3BrBk1THCGW1Yxql
  3utnqQ46P4VDHmWTLmL2xIaMkBgY/U3dP263BU= )
604800 IN RRSIG DNSKEY 8 2 604800 (
  20210305200743 20210203190743 35182 trabajoredes2020.com.
  AEcFEKoP+StvLy6oWJgPiCcBwj5t5B8WjyWPaupT0x54
  LZSGTkrbv5e0e0QKxKB110KXjEwxIh/h657oBA/ckKFA
  WMkZPqdbRg0wVymQ2KB0yGvmjh4/c02ius3Hn0Vzpxr
  BA7WV9EkaQCd0aAZGgnyd4q5kmPHw2KK0k/Bf/kI7pv2
  g21W/0B12D21UyJpXY4eWMT0hyMzCPVx0pHpMpr3Mws
  yTqbu47rmFs2mp0/dB7mYXkiZiZEdVzmDDruxC9LjFoR
  6A5U+DKevLB/DL0FWCYejw7m7NTmVH8U2EKASi9nejsJ
  1D0lWK23y7GvSgydFkkfH32VxyQbWefLFQ== )
```

Figura 27: Consulta DNS.

Como se puede observar, el servidor DNS autoritativo responde al usuario con registros RRset y registros RRSIG para el subdominio “paginaweb”, así como también, con las claves DNSKEY y sus correspondientes RRSIG. Esto indica que la tecnología DNSSEC está implementada en el servidor.

Pero, ¿cómo verificamos que es segura la zona y las claves pertenecen realmente a un servidor DNS autorizado?. Para verificar la veracidad de las claves proporcionadas por el servidor DNS, se utiliza la herramienta *delv* (14). *delv* es una herramienta de línea de comandos para probar las firmas DNSSEC. Lo que hace es enviar a un servidor de nombres especificado, todas las consultas necesarias para obtener y validar los datos solicitados, para lograr verificar la cadena de confianza.

```
root@53472003ae4d:/# delv @192.168.40.40 paginaweb.trabajoredes2020.com.
;; broken trust chain resolving 'trabajoredes2020.com/DNSKEY/IN': 192.168.40.40#53
;; broken trust chain resolving 'paginaweb.trabajoredes2020.com/A/IN': 192.168.40.40#53
;; resolution failed: broken trust chain
```

Figura 28: Ruptura de cadena de confianza.

La figura , muestra que se ha roto la cadena de confianza, ¿Que significa esto? Como no se realizó una función hash sobre la clave KSK pública y tampoco se informó a la región padre, no existe un registro DS firmado por el dominio “.com.” en su servidor, para el dominio “trabajoredes2020.com.”. Por lo tanto, nadie certifica realmente que el dominio existe y es valido en internet. Esta situación, expone la irregularidad del servidor DNS dockerizado. Para dar solución a la problemática, habría que adquirir autoridad legal sobre el subdominio “trabajoredes2020”, que depende del dominio “.com”. Sin embargo, a fines prácticos, una solución más sencilla consiste en configurar un anclaje de confianza para el dominio “trabajoredes2020.com.” en el resolver que realiza las solicitudes. ¿Es esto posible? Si, ya que en primer instancia, al no especificar anclaje de confianza, el resolver se vale de las claves seguras de la raíz para verificar el resto de dominios utilizando DNSSEC. Si se configura una clave KSK pública en la cual confiar, la verificación sería válida.

Dentro del contenedor de bind9, se ha creado en la raíz, un directorio “respaldoKeys” que contiene un archivo definido como “key”, con los datos de la KSK pública. Este archivo en su interior, debe tener un formato similar al que se muestra en la imagen:

```
trusted-keys {
trabajoredes2020.com. 257 3 5 "AQPWA4BRyjB3eqYNy/oykeGcSXjl+HQK9CciAxJfMcS1vEuwz9c
+QG7s EJnQuH5B9i5o/ja+DVitY3jpXNa12mEn";
};
```

Figura 29: Anclaje de confianza.

Una vez formateada la clave dentro de la instancia “trusted-keys”, como se muestra anteriormente, se copia al contenedor del usuario. Luego, se ejecuta el siguiente comando como se observa en la imagen 30:

```
root@53472003ae4d:/# delv @192.168.40.40 -a respaldoKeys/key +root=trabajoredes2020.com. paginaweb.trabajoredes2020.com.
; fully validated
paginaweb.trabajoredes2020.com. 604800 IN A      192.168.40.30
paginaweb.trabajoredes2020.com. 604800 IN RRSIG A 8 3 604800 20210304180342 20210202174753 24978 trabajo
redes2020.com. p7VZWToEiWxXW0aobS8iz41bDmr3NJmKgj4oeUKuFWtErXW9Yoadty8 tqsw2fc4L5mU/UZLEJDUEYfkhdb2Ve9A
JUZWAC7WsbRi3UCCmyCoD+x M7uizZDldndUKs9Czeuv9IpfxrSJEFTXUb+mh5tdxE0owMI/w2lDdG9 Zfs=
```

Figura 30: Verificación de firmas.

Como resultado, el comando *delv* junto con las especificaciones asociadas a la clave KSK pública, permiten verificar las firmas utilizadas por el dominio “trabajoredes2020.com.”. Luego, se puede simular el comportamiento de un navegador mediante la herramienta *curl*, para realizar las consultas HTTP y obtener el contenido de los servidores web. El comando debe tener la siguiente forma: “*curl paginaweb.trabajoredes2020.com. -x (socket)*”. Donde socket hace referencia a la dirección IP del proxy inverso y el puerto 80.

HTTP (protocolo de transferencia de hipertexto) es un protocolo de la capa de aplicación, que sigue el modelo cliente/servidor, y funciona bajo el paradigma de petición y respuesta. El cliente es quien inicia la conexión (three ways handshake) hacia el servidor, utilizando el protocolo de transporte TCP, en el puerto 80, para solicitar recursos mediante diferentes métodos como get, post, head, options, entre otros. Al ejecutar el comando que se mencionó anteriormente, el protocolo HTTP genera la siguiente petición: GET /index.html http/1.1. Las respuestas por parte del servidor están compuestas por una cabecera y un cuerpo. En la cabecera se incluye un código de estado (estado de la respuesta) referenciando éxito, redirección, o errores, tanto del servidor como del cliente, para una solicitud del usuario, así como también parámetros adicionales que incluyen la versión del protocolo HTTP que se esta

utilizando, la longitud de la misma, fecha de petición y creación, etc. En el cuerpo, o body, se incluye el código HTML en texto plano.

¿Qué es HTML?

HTML es un lenguaje estándar para documentos web, el cual utiliza marcas o tags, que no necesita ser compilado.

En la imagen 31 se puede observar lo antes mencionado:

```
root@06177b4d198b:/# curl paginaweb.trabajoredes2020.com. -x 192.168.40.30:80 -i
HTTP/1.1 200 OK
Accept-Ranges: bytes
Content-Length: 131
Content-Type: text/html
Date: Thu, 04 Feb 2021 20:24:33 GMT
Etag: "5ff9f444-83"
Last-Modified: Sat, 09 Jan 2021 18:21:56 GMT
Server: nginx/1.19.6

<!DOCTYPE html>
<html>
<head>
<title>pagina web</title>
</head>
<body>
<h1>Pagina web simulando servicio http</h1>
</body>
</html>
```

Figura 31: Consulta HTTP.

Al combinar ambos comandos (*delv* y *curl*), se logran resoluciones DNS con DNSSEC, verificando la integridad y autenticidad de la respuesta, indicado mediante *fully validated*, y luego, mediante la dirección IP obtenida, se genera la conexión entre el cliente y el servidor para obtener el código HTML.

```
root@53472003ae4d:/# delv @192.168.40.40 -a respaldoKeys/key +root=trabajoredes2020.com. paginaweb.trabajoredes2020.com.
; fully validated
paginaweb.trabajoredes2020.com. 604800 IN A 192.168.40.30
paginaweb.trabajoredes2020.com. 604800 IN RRSIG A 8 3 604800 20210304180342 20210202174753 24978 trabajo
redes2020.com. p7VZWtoEiWxXW0aobS8iz41bDmr3NJmkKgJ4oeUKuFwtErXW9Yeatdy8 tqSW2fc4L5mU/UZLEJDuEYfkhdB2Ve9A
UuZWAc7WsbRLi3UCCmyCoD+x M7uizZDlndUKs9Czeuv9IpfxrSJEFKTxUb+mh5tdxE0oWMI/w2lDdG9 Zfs=
root@53472003ae4d:/# curl paginaweb.trabajoredes2020.com. -x 192.168.40.30:80
<!DOCTYPE html>
<html>
<head>
<title>pagina web</title>
</head>
<body>
<h1>Pagina web simulando servicio http</h1>
</body>
</html>
```

Figura 32: Utilización de comandos *delv* y *curl*.

Superada la etapa de implementación y verificación del funcionamiento, se procede a ejecutar pruebas de seguridad para verificar la robustez de DNSSEC frente a ataques del estilo “man in the middle” sobre el servidor DNS. A continuación, se muestran las resoluciones DNS para un usuario, en presencia de un atacante, el cual modifica las respuestas DNS para “paginaweb.trabajoredes2020.com.” y “documentacion.trabajoredes2020.com.”, donde la dirección IP en lugar de ser 192.168.40.30, se modifica a 192.168.40.3.

Dentro del contenedor del atacante, deben ejecutarse las siguientes líneas de comando:

Maero Jesica – Miranda Martinez Magali – Gorordo Lucas

→ python ARP_Spoof.py & (ejecucion en segundo plano)

→ python DNS_Spoof.py

La ejecución combinada de ambos script entrega información interesante que se puede observar en la imagen 33:

```
[+] Sent to 192.168.40.2 : 192.168.40.40 is-at 02:42:c0:a8:28:03
[+] Sent to 192.168.40.40 : 192.168.40.2 is-at 02:42:c0:a8:28:03
[+] Sent to 192.168.40.2 : 192.168.40.40 is-at 02:42:c0:a8:28:03
[+] Sent to 192.168.40.40 : 192.168.40.2 is-at 02:42:c0:a8:28:03
[+] Sent to 192.168.40.2 : 192.168.40.40 is-at 02:42:c0:a8:28:03
[+] Sent to 192.168.40.40 : 192.168.40.2 is-at 02:42:c0:a8:28:03
[+] Sent to 192.168.40.2 : 192.168.40.40 is-at 02:42:c0:a8:28:03
[+] Sent to 192.168.40.40 : 192.168.40.2 is-at 02:42:c0:a8:28:03
```

Figura 33: Ejecución del ataque (dos script).

Básicamente, lo que puede observarse en la imagen anterior, es que el atacante está realizando envenenamiento de las tablas ARP, tanto para el usuario, como para el servidor DNS. De esta manera, logra interceptar el tráfico entre ambos. Una forma más sencilla de entender la imagen anterior es:

→ Mensaje a: Host 192.168.40.2 (usuario), diciendo: “la MAC asociada al host 192.168.40.40 (servidor DNS) es MAC del atacante.”

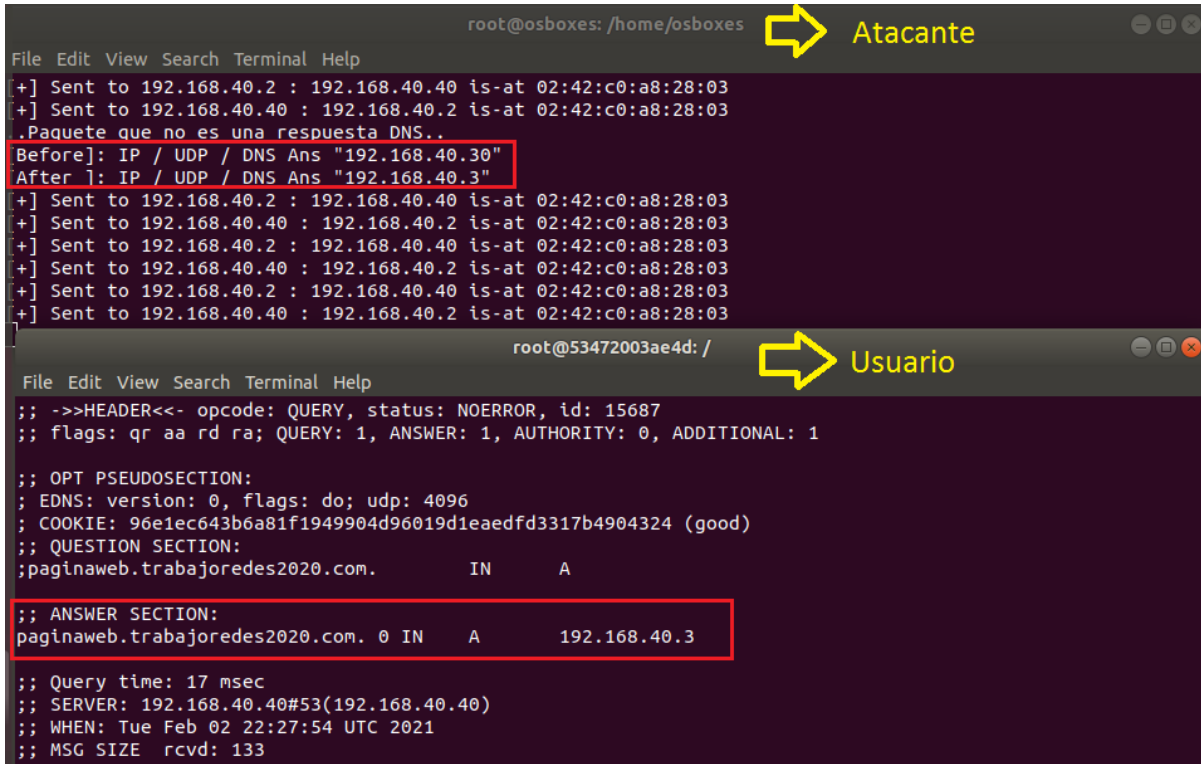
→ Mensaje a: Host 192.168.40.40 (Servidor DNS), diciendo: “la MAC asociada al host 192.168.40.2 (usuario) es MAC del atacante”.

En la siguiente imagen (34), se corrobora el ataque inicial, al verificar las tablas ARP de ambos.

root@53472003ae4d:/# arp						Tabla ARP envenenada (Usuario)
Address	HWtype	HWaddress	Flags	Mask	Iface	
Proxy.tpfinaldnssecsecu	ether	02:42:c0:a8:28:1e	C		eth0	
192.168.40.1	ether	02:42:ce:3b:fc:07	C		eth0	
Atacante.tpfinaldnssecs	ether	02:42:c0:a8:28:03	C		eth0	
DNS.tpfinaldnssecsecuri	ether	02:42:c0:a8:28:03	C		eth0	
root@ba2451df5f74:/# arp						Tabla ARP envenenada (Servidor DNS)
Address	HWtype	HWaddress	Flags	Mask	Iface	
Usuario.tpfinaldnssecse	ether	02:42:c0:a8:28:03	C		eth0	
Atacante.tpfinaldnssecs	ether	02:42:c0:a8:28:03	C		eth0	
192.168.40.1	ether	02:42:ce:3b:fc:07	C		eth0	

Figura 34: Envenenamiento de tablas ARP.

Luego, cuando el usuario realiza una consulta DNS para el dominio “paginaweb.trabajoredes2020.com.”, se encuentra que la dirección IP es 192.168.40.3.



```
root@osboxes: /home/osboxes ➡ Atacante
File Edit View Search Terminal Help
+] Sent to 192.168.40.2 : 192.168.40.40 is-at 02:42:c0:a8:28:03
+] Sent to 192.168.40.40 : 192.168.40.2 is-at 02:42:c0:a8:28:03
.Paquete que no es una respuesta DNS..
Before]: IP / UDP / DNS Ans "192.168.40.30"
After ]: IP / UDP / DNS Ans "192.168.40.3"
+] Sent to 192.168.40.2 : 192.168.40.40 is-at 02:42:c0:a8:28:03
+] Sent to 192.168.40.40 : 192.168.40.2 is-at 02:42:c0:a8:28:03
+] Sent to 192.168.40.2 : 192.168.40.40 is-at 02:42:c0:a8:28:03
+] Sent to 192.168.40.40 : 192.168.40.2 is-at 02:42:c0:a8:28:03
+] Sent to 192.168.40.2 : 192.168.40.40 is-at 02:42:c0:a8:28:03
+] Sent to 192.168.40.40 : 192.168.40.2 is-at 02:42:c0:a8:28:03

root@53472003ae4d: / ➡ Usuario
File Edit View Search Terminal Help
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 15687
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; COOKIE: 96e1ec643b6a81f1949904d96019d1eaedfd3317b4904324 (good)
;; QUESTION SECTION:
;paginaweb.trabajoredes2020.com.      IN      A

;; ANSWER SECTION:
paginaweb.trabajoredes2020.com. 0 IN      A      192.168.40.3

;; Query time: 17 msec
;; SERVER: 192.168.40.40#53(192.168.40.40)
;; WHEN: Tue Feb 02 22:27:54 UTC 2021
;; MSG SIZE rcvd: 133
```

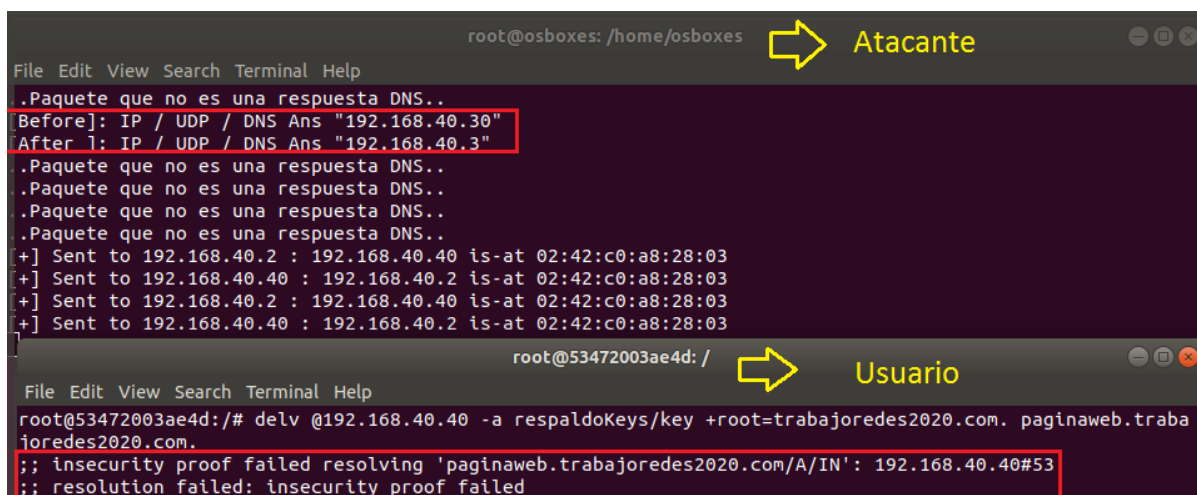
Figura 35: Ataque sin utilizar DNSSEC.

El resultado de generar una consulta HTTP al supuesto “servidor” que se encuentra en la dirección IP 192.168.40.3 mediante la herramienta *curl*, entrega el siguiente mensaje: *“Failed to conect to 192.168.40.3 port 80: connection refused”* . Significa que el atacante está generando la **denegación de servicio** al usuario. Sin embargo, podría montar un servidor web falso como mecanismo de phishing para robar datos al usuario, por lo tanto, la denegación del servicio no supone la peor amenaza en estas situaciones.

El término Phishing es utilizado para referirse a uno de los métodos más utilizados por delincuentes cibernéticos para estafar y obtener información confidencial de forma fraudulenta como puede ser una contraseña o información detallada sobre tarjetas de crédito u otra información bancaria de la víctima.

El estafador, conocido como phisher, se vale de técnicas de ingeniería social, haciéndose pasar por una persona o empresa de confianza en una aparente comunicación oficial electrónica, por lo general un correo electrónico, o algún sistema de mensajería instantánea, redes sociales SMS/MMS, a raíz de un malware o incluso utilizando también llamadas telefónicas.[\(15\)](#)

Para contrarrestar al atacante, el resolver debería utilizar DNSSEC como mecanismo para autenticar las respuestas del servidor DNS y comprobar que nadie las ha modificado. De esta manera, el usuario debe ejecutar el siguiente comando como se muestra en la imagen 36 presentada a continuación:



```

root@osboxes: /home/osboxes ➡ Atacante
File Edit View Search Terminal Help
..Paquete que no es una respuesta DNS..
Before]: IP / UDP / DNS Ans "192.168.40.30"
After ]: IP / UDP / DNS Ans "192.168.40.3"
..Paquete que no es una respuesta DNS..
..Paquete que no es una respuesta DNS..
..Paquete que no es una respuesta DNS..
..Paquete que no es una respuesta DNS..
+] Sent to 192.168.40.2 : 192.168.40.40 is-at 02:42:c0:a8:28:03
+] Sent to 192.168.40.40 : 192.168.40.2 is-at 02:42:c0:a8:28:03
+] Sent to 192.168.40.2 : 192.168.40.40 is-at 02:42:c0:a8:28:03
+] Sent to 192.168.40.40 : 192.168.40.2 is-at 02:42:c0:a8:28:03

root@53472003ae4d: / ➡ Usuario
File Edit View Search Terminal Help
root@53472003ae4d:/# delv @192.168.40.40 -a respaldoKeys/key +root=trabajoredes2020.com. paginaweb.trabajoredes2020.com.
;; insecurity proof failed resolving 'paginaweb.trabajoredes2020.com/A/IN': 192.168.40.40#53
;; resolution failed: insecurity proof failed

```

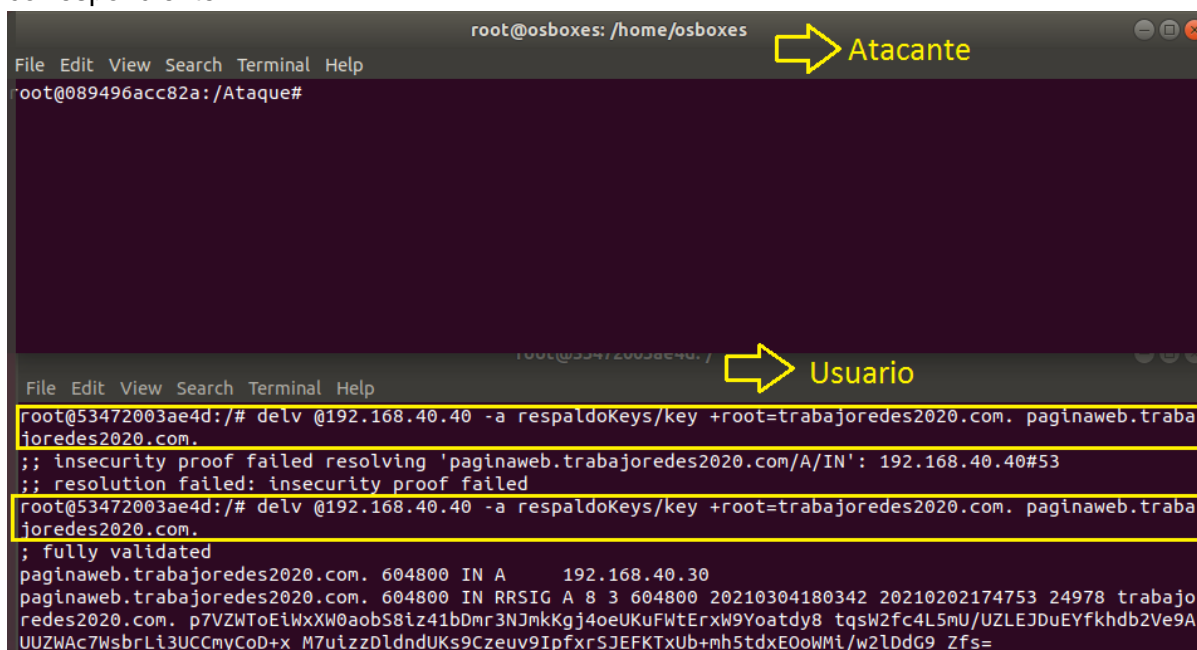
Figura 36: Ataque utilizando DNSSEC.

La respuesta a este comando, como se puede observar en la figura 36, nos indica que la resolución DNS es insegura, por lo tanto, el resolver no entrega la dirección IP falsa al usuario, como si lo hacía en consultas sin DNSSEC.

¿Se evita la denegación del servicio?

No, no se evita la denegación del servicio. Mientras el atacante esté presente no se puede recuperar la dirección IP verdadera del servidor web, pero si se puede evitar ataques de phishing, lo que garantiza que los clientes no se conecten a servidores web falsos.

Por último, en la imagen 37, se puede observar que si el atacante cesa sus acciones, el resolver DNS comprueba la integridad de los mensajes y entrega la dirección IP del servidor correspondiente.



```

root@osboxes: /home/osboxes ➡ Atacante
File Edit View Search Terminal Help
root@089496acc82a:/Ataque#

root@53472003ae4d: / ➡ Usuario
File Edit View Search Terminal Help
root@53472003ae4d:/# delv @192.168.40.40 -a respaldoKeys/key +root=trabajoredes2020.com. paginaweb.trabajoredes2020.com.
;; insecurity proof failed resolving 'paginaweb.trabajoredes2020.com/A/IN': 192.168.40.40#53
;; resolution failed: insecurity proof failed
root@53472003ae4d:/# delv @192.168.40.40 -a respaldoKeys/key +root=trabajoredes2020.com. paginaweb.trabajoredes2020.com.
; fully validated
paginaweb.trabajoredes2020.com. 604800 IN A      192.168.40.30
paginaweb.trabajoredes2020.com. 604800 IN RRSIG A 8 3 604800 20210304180342 20210202174753 24978 trabajo
redes2020.com. p7VZWToEiWxXW0aobS8iz41bDmr3NJmkKgJ4oeUKuFwtErXW9Yeatdy8 tqsw2fc4L5mU/UZLEJDuEYfkhdb2Ve9A
UWZwac7WsbriLi3UCCmyC0D+x M7uizzDldndUKs9Czeuv9IpfXrSJEFKTXUb+mhStdxE0oWMI/w2lddG9 Zfs=

```

Figura 37: Solicitud con ataque y frenando el mismo.

CONCLUSIÓN.

Luego de un arduo trabajo de investigación e implementación de servidores DNS y web, y sus correspondientes herramientas de seguridad, haciendo hincapié en DNSSEC, se pudo observar la importancia de aplicar seguridad en redes porque, actualmente, es muy fácil poder llevar adelante actividades maliciosas sin que la víctima pueda advertir las mismas, en caso de ausencia de esta herramienta.

Es importante destacar que el objetivo del proyecto se pudo cumplir debido a que pudimos analizar e investigar herramientas de seguridad, en particular DNSSEC, comprendiendo su funcionamiento y observando la diferencia en un escenario con dicha tecnología y en ausencia de la misma, frente a un ataque.

Si bien durante el desarrollo del proyecto se implementó una metodología específica de ataque, existen múltiples amenazas a la seguridad de los servidores, fiándose de las vulnerabilidades de los mismos. Por esto, la seguridad debe ser considerada como un proceso integral y de continua evolución. En este análisis, las herramientas que existen hoy en día para generar los ataques son cada vez más fáciles de conseguir e implementar, aunque su complejidad es mayor. También, existen los conocidos como “zero day exploits” que consisten en ciberataque que se produce el mismo día en que se descubre una vulnerabilidad en el software. En ese punto, se aprovecha la vulnerabilidad antes de que el creador del software ponga a disposición una solución [\(16\)](#). Esto último, es un asunto muy importante para la seguridad, que debe llevar mucha atención por parte del administrador de red.

Esto significa que el tema abordado en este trabajo es una temática de todos los días dentro de una organización porque no solo se debe aplicar a ataques ya conocidos, con herramientas como DNSSEC, sin perder de vista aquellos ataques latentes a suceder que aprovechan vulnerabilidades de la red, para los cuales pueden aplicarse las mismas herramientas o acudir a nuevas.

Como trabajo futuro, se podría plantear el estudio de HSM (*Hardware Security Module*) ya que la seguridad del sistema criptográfico debe recaer en la seguridad de la clave. En este sentido, una de las alternativas más recomendables para realizar la gestión de las claves es el uso de un módulo de seguridad de hardware.

Otra línea de acción a futuro radica en el análisis de vectores ataques distintos al planteado, y sus correspondientes soluciones de seguridad. También, analizar ataques a diferentes servidores, como por ejemplo web, y verificar la efectividad de un proxy inverso como herramienta de seguridad para ataques DOS.

BIBLIOGRAFÍA.

(1)

Deyimar A. (1 de noviembre de 2019). Qué usar – Nginx vs Apache. Hostinger.

<https://www.hostinger.com.ar/tutoriales/que-usar-nginx-vs-apache/>

(2)

(17 de marzo de 2020). ¿Qué es un servidor proxy inverso?. Ionos.

<https://www.ionos.es/digitalguide/servidores/know-how/que-es-un-servidor-proxy-inverso/>

(3)

Middlewares. Traefik. <https://doc.traefik.io/traefik/middlewares/overview/>

(4)

¿Qué es el envenenamiento de caché de DNS?. Cloudflare.

<https://www.cloudflare.com/es-la/learning/dns/dns-cache-poisoning/>

(5)

(5 de diciembre de 2016). Denegación de servicio - DNS amplificado. Fwhibbit.

<https://fwhibbit.es/denegacion-de-servicio-dns-amplificado>

(6)

¿Por qué debería usar docker en mi proyecto de desarrollo?. Apiumhub.

<https://apiumhub.com/es/tech-blog-barcelona/usar-docker/#:~:text=Docker%20nos%20permite%20tener%20control,almacenarlo%20en%20nuestro%20disco%20duro.>

(7)

Rodriguez, T. (10 de septiembre de 2019). De Docker a Kubernetes: entendiendo qué son los contenedores y por qué es una de las mayores revoluciones de la industria del desarrollo. Xataka.

<https://www.xataka.com/otros/docker-a-kubernetes-entendiendo-que-contenedores-que-mayores-revoluciones-industria-desarrollo>

(8)

Rockikz, A. (Noviembre de 2020). How to Build an ARP Spoofer in Python using Scapy. thepythoncode. <https://www.thepythoncode.com/article/building-arp-spoofing-using-scapy>

(9)

Rockikz, A. (Noviembre de 2020). How to Make a DNS Spoof attack using Scapy in Python. thepythoncode. <https://www.thepythoncode.com/article/make-dns-spoof-python>

(10)

Traefik. DockerHub. https://hub.docker.com/_/traefik

(11)

Use bridge networks. Docker docs. <https://docs.docker.com/network/bridge/>



(12)

Gorordo, L. Maero, J. Miranda Martinez, M. (29 de diciembre de 2020). bind9-DNSSEC. GitHub. <https://github.com/Gorordo96/bind9-DNSSEC>

(13)

(Octubre 2018). Guía de implantación y buenas prácticas de DNSSEC. Incibe.

https://www.google.com/url?q=https://www.incibe-cert.es/sites/default/files/contenidos/guias/doc/incibe_guia_de_implantacion_y_buenas_practicas_de_dnssec.pdf&sa=D&source=editors&ust=1612369280019000&usq=AOvVaw3l1mcqylmbEKMkDycMU9f6

(14)

Weber, J. (19 de diciembre de 2019). Dive into delv: DNSSEC Validation. Weberblog. <https://weberblog.net/dive-into-delv-dnssec-validation/>

(15)

Rivero, M. (03 de noviembre de 2008) ¿Qué es el Phishing?. Info Spyware. <https://www.infospyware.com/articulos/que-es-el-phishing/>

(16)

(5 de diciembre de 2017). ¿Qué es un exploit de día cero?. Kaspersky. <https://latam.kaspersky.com/resource-center/definitions/zero-day-exploit>

López Padilla, A. (9 de abril de 2014). Guía de seguridad en servicios DNS. Incibe.

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjY5MSkpNruAhXKHLkGHRIPCiwQFjAAegQIAhAC&url=https%3A%2F%2Fwww.incibe.es%2Fextfrontinteco%2Fimg%2Ffile%2Fintecocert%2FManualesGuias%2Fguia_de_seguridad_en_servicios_dns.pdf&usq=AOvVaw1czeCI23vBaKyUPq8abAgP