Títol: Herramientas de seguridad y máquina virtual para navegación segura

Volum:

Alumne: Luis Miguel Laguna Corripio

Director/Ponent: Manuel Medina Llinàs

Departament: Arquitectura de Computadors

Data:

DADES DEL PROJECTE

Títol del Projecte: Herramientas de seguridad y máquina virtual para navegación segura

Nom de l'estudiant: Luis Miguel Laguna Corripio
Titulació: Enginyeria Tècnica en Informàtica de Sistemes
Crèdits: 22.5
Director/Ponent: Manuel Medina Llinàs
Departament: Arquitectura de Computadors

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President: Jordi Iñigo Griera

Vocal: Robert Lukas Mario Nieuwenhuis

Secretari: Manuel Medina Llinàs

QUALIFICACIÓ

Qualificació numèrica:
Qualificació descriptiva:

Data:

Prefacio

Herramientas de seguridad y máquina virtual para navegación segura es un proyecto que surge de una beca de proyectos de investigación del departamento de Arquitectura de Computadores. Enmarcada ésta en un mayor proyecto de investigación de búsqueda de herramientas antifraude financiero, que tiene como objetivo concebir nuevos algoritmos y herramientas con tal de prevenir, detectar y corregir los posibles ataques a los que pueden estar sometidos diversos canales de banca electrónica.

Mi colaboración en ese proyecto forma parte del segundo eje de investigación, el cual cubre los aspectos de seguridad relacionados con los canales electrónicos. Con el objetivo de garantizar la seguridad de los canales de comunicación: internet fijo, móvil y TDT en las operaciones financieras.

Ya más concretamente, mi trabajo en ése segundo eje de investigación se centra en dos puntos:

- En el no repudio de origen a través de PROXY. Aportando directrices de diseño de portales PROXY para el acceso desde terminales móviles.

- Pruebas de concepto de diferentes canales de acceso. Facilitando un informe de resultados de Pruebas de Concepto en los canales móviles sobre integridad en la identificación de la sesión, usabilidad, identificado de usuario y terminal.

Dado el marco de trabajo de mi beca de investigación, mi proyecto final de carrera paralelo abarcará un estudio de seguridad entre una entidad financiera y un cliente de ésta. Dando un especial interés en la parte de evitar y detectar las potenciales modificaciones del sistema de traducción de nombres de dominio a direcciones ip, en diferentes niveles, sobre los dominios de una entidad financiera.

El reto que plantea el proyecto es hacer investigación en el entorno de la seguridad para proteger de una manera inexistente actualmente los servicios de banca electrónica. Dicho reto habrá de permitir en un futuro la evaluación en tiempo real del comportamiento de un acceso y la detección de fraude en tiempo real basado en la historia del comportamiento anterior y sobre diferentes usuarios.

Índice

1. Introducción....7
   1.1. Objetivos....9
   1.2. Planificación....10
2. Estudio previo y especificación de las necesidades....14
3. Diseño....20
   3.1. Esquema de la red....20
   3.2. Servidores....21
   3.3. Transmisión de datos....22
   3.4. Los entresijos de DNSSEC....25
   3.5. Cliente....35
4. Implementación....38
   4.1. Configuración del firewall+router....39
   4.2. Instalación y configuración del servidor Web....41
   4.3. Instalación y configuración del servidor DNS seguro....44
   4.4. Cliente....50
5. Conclusión....53
6. Futuro del proyecto....54
7. Referencias bibliográficas....55
8. Anexo....56

1. Introducción

El sector financiero es altamente tecnológico y tradicionalmente ha sido pionero a la hora de incorporar las tecnologías de la información y las comunicaciones (TIC) en su negocio. La naturaleza de los productos y servicios financieros, intensivos en el uso de información y conocimiento hacen de las TIC un instrumento ideal, de manera que su desarrollo está fuertemente ligado al sector de las TIC.

Hoy en día la mayoría de entidades financieras de prestigio han impulsado la banca en línea como un canal más de comunicación y de gestión de los clientes, ofreciendo la posibilidad de realizar prácticamente la totalidad de operaciones en línea. En algunas instituciones estas operaciones virtuales ya superan en número a las operaciones presenciales, con el consiguiente ahorro de recursos de personal que eso representa.

La banca en línea respresenta un escenario muy propicio para el intento de realizar fraudes, y ésta es una de las principales barreras para aumentar los usuarios de servicios financieros virtuales. En el entorno de la banca en línea la parte más débil de la cadena de seguridad es el propio usuario, que en muchos casos no conoce los elementos indispensables de seguridad. Aún así, para el sector financiero y para la sociedad en general se hace indispensable potenciar el canal electrónico como un medio para las operaciones financieras ya que presenta múltiples ventajas tanto para clientes como para las entidades.

Los delitos electrónicos están muy vinculados con el desarrollo tecnológico, las organizaciones criminales dedican muchos esfuerzos a actualizarse y a buscar agujeros de seguridad para poder realizar sus ataques, tanto a los usuarios finales, como a los proveedores de servicios financieros. En el argot del sector la situación se define como: “el fraude no se destruye si no se transforma”, es decir si se protege una parte de los canales, resulta que el fraude reaparece en el que ha quedado menos protegido, por tanto el fraude se considera multicanal. De modo que las entidades financieras tienen la obligación de velar por los intereses de sus clientes y por tanto, si ofrecen los servicios en línea, adoptar medidas de seguridad máximas y convenientes tanto para los accesos como para las aplicaciones.

Con ésa máxima, este trabajo pretende ser un estudio de seguridad entre una entidad financiera y un cliente de ésta. Dando un especial interés en la parte de evitar y detectar las potenciales modificaciones del sistema de traducción de nombres de dominio a direcciones ip, en diferentes niveles, sobre los dominios de una entidad financiera.

A continuación se nombran datos referentes al estado actual del fraude en línea:

La compañía McAfee publicó en 2008 el estudio 'Informe de Criminología Virtual' en el que presentó a grandes rasgos los problemas actuales relacionados con los delitos en Internet, según el estudio:

- Países paraíso para los criminales en Internet, algo así como el bullet-proof hosting que ofrecen empresas como la RBN (Russian Business Network) pero a nivel de país.

- Los gobiernos poniendo presión para proteger a los ciudadanos.

- Aprovechamiento de datos de redes sociales para cometer todo tipo de delitos.

- Ataques utilizando nuevas tecnologías: P2P, VoIP... lo cual es comprensible, siendo éstas utilizadas diariamente muchos de nosotros casi el 100% de nuestro tiempo.

- Pérdida de confianza en los servicios en línea a causa de la gran cantidad de riesgo y fraude que existirán.

La española s21sec publicó en 2008 su tercer informe de fraude en línea, en el que presentó las últimas tendencias en estos lares:

- Ingenieria Social: siendo cualquier método bueno para engañar al eslabón más débil: eurocopa, elecciones, declaración de la Renta, loterías, trabajos desde casa, desastres naturales,…

- Spear phishing: utilización de datos personales para hacer los mensajes más creíbles (datos sacados de las redes sociales: amigos, aficiones, correos, fotos, ...)

- Código malicioso cada vez más complejo (entornos virtuales (hypervisor), rootkits, creación a medida, P2P, cifrado).

- Proliferación de ISP a prueba de balas. Como la famosa RBN (Russian Business Network).

- Amenaza de los países emergentes (e.g. China). Tan sólo hay que buscar en un buscador malware+china.

- Irrupción en los dispositivos móviles (iPhone, BlackBerry, Windows Mobile, Symbian).

- Aumento del número de sistemas Microsoft comprometidos debido a la baja penetración de Windows Vista.

- Aumento del número de sistemas MacOSX comprometidos.

- Situación económica mundial.

Y también periódicos, en este caso el país, hace eco del presente crecimiento del fraude online:

http://www.elpais.com/articulo/Pantallas/fraude/Red/dispara/2008/elpepurtv/20080801elpepirtv_1/Tes

ELPAIS.com > Gente y TV

Multimedia

El fraude en la Red se dispara en 2008

R. M. - Madrid - 01/08/2008

Vota ☆☆☆☆☆ Resultado ★★★½☆ 6 votos

Con el verano, llegan los robos. Y las campañas de concienciación antirrobo. Internet no se quiere quedar atrás. Y es que los organismos oficiales y asociaciones que luchan contra este tipo de prácticas en Interner estiman que los intentos de fraude a través de la Red se han multiplicado por ocho en los primeros meses de 2008 con respecto a 2007.

› Una banda rusa infecta las redes de cientos de empresas de EE UU

Instituto Nacional del Consumo

A FONDO

Sede: Madrid (España)

Ver cobertura completa

La noticia en otros webs

- webs en español
- en otros idiomas

Así lo aseguró la directora general de Consumo, Etelvina Andreu, en la presentación de una campaña de prevención que pondrá en marcha a partir de hoy el Instituto Nacional del Consumo (INC).

La intención de Consumo es difundir a los 21 millones de internautas una serie de orientaciones para que nadie sucumba ante reclamos que constituyan engaño. Las recomendaciones están en la página web del INC con fraudes típicos: la supuesta venta de productos caros a precios irrisorios, subastas fraudulentas, diversos engaños que usan como señuelo loterías oficiales y *phishing* o intentos de lograr las claves bancarias mediante correos electrónicos engañosos.

En conclusión, cada vez se utilizan más los servicios de la banca en línea y a la vez que el nivel de operaciones aumente, el fraude irá apareciendo. Se trata de un ámbito en el que es necesario una búsqueda continua para ir un paso adelante de las actividades delictivas. Ya que se trata de un entorno particular, que experimenta un proceso de realimentación: los avances tecnológicos permiten ofrecer nuevos servicios de manera segura, pero al mismo tiempo se generan nuevas herramientas que vulneran los sistemas para realizar delitos. Es por esa razón que este proyecto se centra en la búsqueda y prevención de amenazas de la banca electrónica.

1.1. Objetivos

Hacer un estudio y análisis para mejorar la seguridad en los sistemas de tecnologías de la información entre una entidad financiera y un cliente de ésta. Teniendo en cuenta de que se trataría de entornos de millones de usuarios y dispositivos en tiempo real, se precisa de la máxima seguridad.

Con la banca en línea es el usuario quien ha pasado a gestionar su cuentas y realizar las operaciones, de manera que el reto que tienen las entidades financieras que ofrecen banca en línea es conseguir encontrar un punto medio entre la seguridad y la usabilidad, intentando dar respuesta al problema de la confidencialidad y privacidad.

Para el sector financiero los principales objetivos y factores determinantes que dan cabida al desarrollo de la banca electrónica, son establecer la prevención, detección y corrección de los posibles ataques a los que puede estar sometida la banca electrónica en relación con la autenticación del usuario y la entidad financiera, la privacidad de la información que se intercambia, la integridad de ésta (no modificada al intercambiarse), y el no repudio (no poder negar que aquella transacción ha sido autorizada por el usuario).

Se dará un especial interés en la parte de evitar y detectar las potenciales modificaciones del sistema de traducción de nombres de dominio a direcciones ip, en diferentes niveles, sobre los dominios de una entidad financiera. Evitando así DNS Spoofing.

De forma que posterior al objeto de investigación de este proyecto, se puedan crear herramientas y productos para el sector financiero.

1.2. Planificación

La planificación inicial de horas para el proyecto ha sido de trescientas sesenta horas, dieciocho semanas de veinte horas a dieciséis horas el crédito.

Distribuyendo el proyecto en cuatro fases:

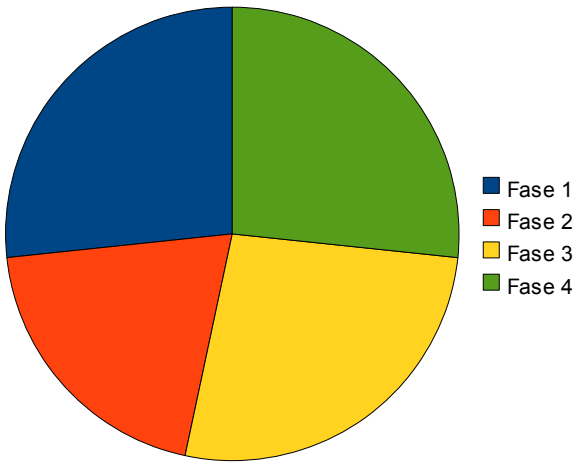
- Fase 1: Estudio previo y especificación de las necesidades.

- Fase 2: Diseño.

- Fase 3: Implementación.

- Fase 4: Documentación y demostración.

La planificación inicial de las fases ha sido la siguiente:



FASE 1	**95 h.**	**26,39%**
FASE 2	**75 h.**	**20,83%**
FASE 3	**95 h.**	**26,39%**
FASE 4	**95 h.**	**26,39%**

Se considera que estas son las etapas o fases principales del proyecto, aunque no sean secuenciales en el tiempo, por el hecho de que se puedan solapar unas con otras. Como en el caso de la fase Documentación con cualquiera de las otras fases.

No se cree oportuno reducir el número de fases, no sería lo correcto pues los objetivos no estarían claramente segmentados. Más fases o etapas alargarían el ciclo de vida, si el modelo de requerimientos del proyecto final de carrera se viera incrementado.

La definición de las etapas es la siguiente:

- Estudio previo y especificación de las necesidades: Estudio previo sobre la temática, definición del problema, análisi del problema y requisitos del proyecto.

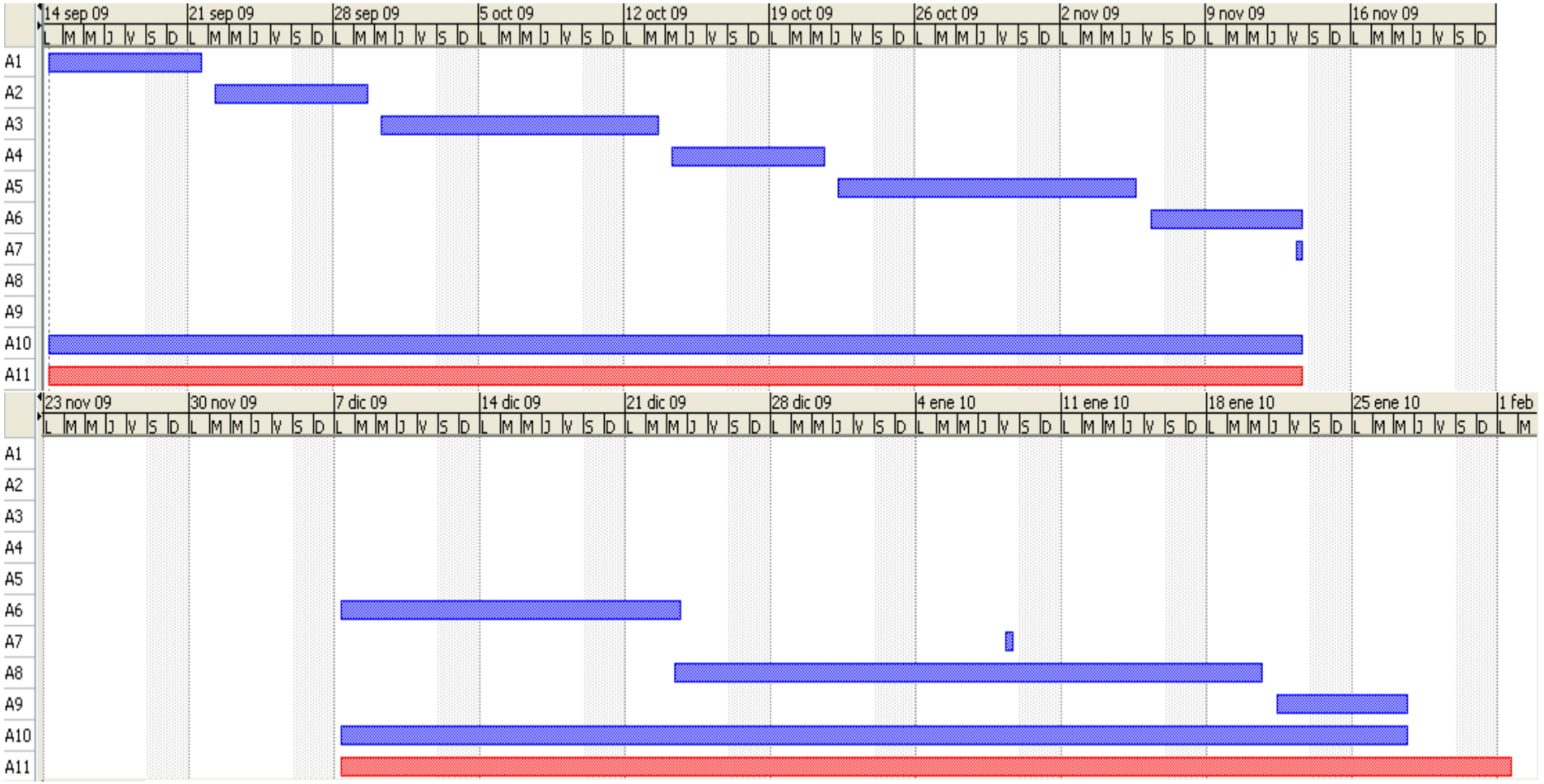
- Diseño: Solución planteada al problema definido y justificación.

- Implementación: Realización material del diseño.

- Documentación y demostración: Realización y distribución de la memoria, demostración de la implementación.

Estas etapas dan lugar a una serie de procesos y actividades que se resumen y gestionan en el siguiente diagrama de Gantt que muestra la durada de cada una de las actividades que se han realizado para finalizar el proyecto.

Como en todos los proyectos, la planificación, duración y herramientas como el diagrama de Gantt pueden acabar resultando algo diferentes a como inicialmente se había planteado, por el hecho de la simple naturaleza del proyecto en el que intervienen personas.


14 sep 09
21 sep 09
28 sep 09
5 oct 09
12 oct 09
19 oct 09
26 oct 09
2 nov 09
9 nov 09
16 nov 09
23 nov 09
30 nov 09
7 dic 09
14 dic 09
21 dic 09
28 dic 09
4 ene 10
11 ene 10
18 ene 10
25 ene 10
1 feb
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11

Actividades

- A1: Definición del proyecto: Se nos pide asegurar cliente - entidad

- A2: Vistazo al panorama actual de seguridad

- A3: Vistazo al panorama actual de seguridad en banca electrónica

- A4: Estudio de las tecnologías actualmente en uso en banca electrónica
    - DNS
    - SSL
    - Comunicaciones móviles
    - Seguridad nivel aplicación

- A5: Estudio de soluciones a problemas existentes
    - DNS
    - Soluciones fijas
    - Soluciones móvil

- A6: Solución y Diseño de la red
    - Servidor DNS
    - Servidor WEB
    - Router
    - Cliente

- A7: Reunión Tutor

- A8: Implementación de la red
    - Servidor DNS
    - Servidor WEB
    - Router
    - Cliente

- A9: Conclusiones

- A10: Gestión del proyecto

- A11: Redacción de la Memoria

2. Estudio previo y especificación de las necesidades

Partiendo de la idea original:

Figura 1

Tan simple como usuario – internet – entidad, uno se cuestiona cómo sería el escenario más seguro posible. Probablemente sería un escenario en el que no existiera usuario, ni internet, ni entidad. Pero entonces no tendríamos nada, y ya sabiendo pues, que la seguridad al cien por cien no existe, aún así decidimos confiar en los sistemas y en la gente que hay detrás, aunque para éso se haya de tomar medidas, tanto nosotros, como la entidad y el canal.

Un sistema bastante seguro sería aquel que tuviera en cuenta tanto la seguridad a nivel de datos, como de aplicación, red, los sistemas y canales en general, aparte de la seguridad entorno al factor humano.

Una solución correspondiente podría ser la siguiente:

- A nivel de sistema, si se tuvieran los recursos, optar por un sistema de *full virtualization* con *hypervisor* de tipo I, corriendo sobre el *hardware* algún sistema *lightweight*, parece correcto, si se tiene en cuenta que el sistema que corra sea lo más seguro posible. Hay diversos estudios de seguridad hechos en relación a asegurar hipervisores. Una buena opción sería el *hypervisor* Xen, mejorando su seguridad a través desagrupación del *domain0*, dividiéndolo en partes pequeñas y seguras, convirtiéndolo en una TCB (Trusted Computing Base). Investigando encontré un proyecto muy interesante, relacionado con el tema de hipervisores seguros que me gustaría recalcar, http://www.opentc.net/, en el que incluyen un TPM (Trusted Platform Module), para darle diversos usos criptográficos. Aún así, esta opción añade mucha complejidad, por ello lo de si *se tuvieran los recursos*. Y la complejidad a menudo incrementa los errores humanos, algo nada bueno para la seguridad en las tecnologías de la información.

- A nivel de aplicación, como serían máquinas dedicadas, se trataría de que corrieran pocos servicios en cada una y con labores específicas, así podríamos por ejemplo encontrar un servidor DNS, un servidor web, y demás. De manera que al ser pocos servicios y distribuidos en varias máquinas, se podrían controlar mejor. Si un servicio se viera comprometido, el estar en máquinas diferentes favorecería a que los demás no tuvieran por qué verse en esa situación.

- A nivel de transporte, que las comunicaciones estuvieran encriptadas y que formaran un tunel.

- A nivel red, un buen filtrado de paquetes y herramientas de monitorización.

Es obvio que se podría dibujar escenarios más seguros a éste, pero también se ha de tener en cuenta los recursos existentes en cuanto a costes.

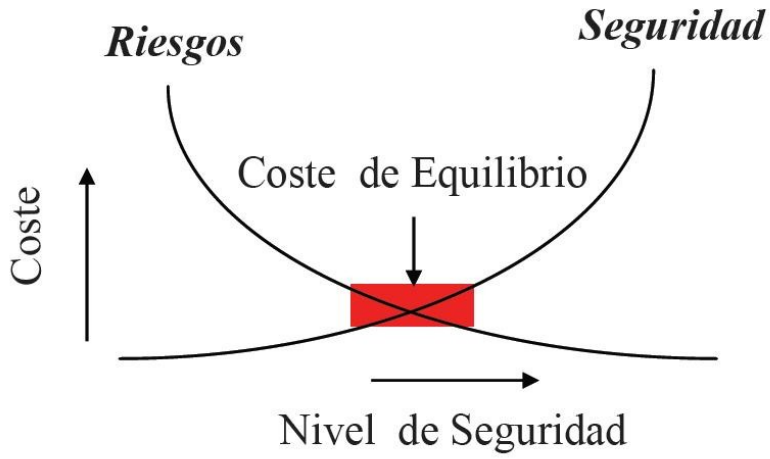


Figura 2

En cualquier caso, y yendo a las cosas que nos preocupan, la autenticación, la privacidad, la integridad y el no repudio, para cubrir éstas, se presentará en la sección 3 de esta memoria el diseño de la solución pertinente.

En el diseño se hará uso de las últimas tecnologías existentes hoy en día para cubrir los aspectos de seguridad relacionados con la seguridad de los canales de comunicación entre cliente y entidad financiera. Dando un especial interés en la parte de evitar y detectar las potenciales modificaciones del sistema de traducción de nombres de dominio a direcciones ip, en diferentes niveles, sobre los dominios de una entidad financiera. Evitando así DNS Spoofing.

Pongamos pues, que la entidad financiera establece cuatro líneas abiertas, o subcanales a través de los cuales los usuarios puedan hacer sus operaciones:

> Una Línea sería a través de SMS, la cual no sería objeto de estudio, considerando así la seguridad aportada por los operadores y sobre el SMSCenter suficiente.
>
> Una Línea Wap, la cual dado el avance de la tecnología y de los dispositivos móviles viene estando un tanto obsoleta, cualquiera de sus versiones, debido a que la tecnología ha saltado a la siguiente generación, pero aún así la analizaremos.
>
> Una Línea Java, que sería como una Línea usual, pero capaz de adaptarse convenientemente al tamaño de pantalla de dispositivos móviles y modificar sus CSS.
>
> Una Línea usual, la habitual de los usuarios comunes de internet.

Así pues, para proteger estas líneas se hará uso de tecnologías tales como:

Para la transmisión de datos, hay varias opciones que son claras candidatas que permiten considerar los objetivos suficientemente cumplidos, IPSec, SSL/TLS y WTLS.

IPSec (Internet Protocol Security) proporciona un conjunto de protocolos de seguridad creados por IETF, una organización que promueve y desarrolla estándares de internet, para garantizar la seguridad de las comunicaciones en Internet mediante la encriptación, autenticación, confidencialidad, integridad de los flujos de datos, protección antirepetición y protección contra el análisis de flujo de tráfico a la capa de red.

Incluye los siguientes protocolos para establecer las siguientes funciones:

> IKE y IKEv2 (Internet Key Exchange) establece una SA (Security Association) manejando la negociación de protocolos y algoritmos para una mútua autenticación entre agentes al principio de la sesión mediante la generación de claves de cifrado y autenticación para ser usadas durante la sesión.
>
> AH (Authentication Header) proporciona integridad, verificación de la modificación de paquetes (*connectionless integrity*), autenticación de los datos de origen de datagramas IP y proporciona protección contra ataques de repetición.
>
> ESP (Encapsulating Security Payload) proporciona confidencialidad, autenticación de origen de los datos, integridad, verificación de la modificación de paquetes (*connectionless integrity)*, un servicio contra la repetición (una forma de integridad de secuencia parcial), y confidencialidad de flujo de tráfico limitado.

SSL (Secure Sockets Layer) y **TLS** (Transport Layer Security), apartir de aquí aplicaré el término SSL a ambos protocolos a menos que el contexto indique lo contrario.

SSL proporciona autenticación y privacidad de la información para la tunelación entre extremos sobre Internet mediante el uso de criptografía en la capa de aplicación. Habitualmente sólo el servidor es autenticado, es decir, se garantiza su identidad, mientras que el cliente se mantiene sin autenticar. La autenticación mutua requiere un despliegue de infraestructura de claves públicas (o PKI) para los clientes. Los protocolos permiten a las aplicaciones cliente - servidor comunicarse de una forma diseñada para prevenir capturas de tráfico, la falsificación de la identidad del remitente y mantener la integridad del mensaje.

WTLS (Wireless Transport Layer Security) es un protocolo de seguridad que parte de la pila WAP (Wireless Application Protocol). Se situa entre la capa WTP y la WDP en la pila de comunicaciones WAP. WTLS es un protocolo derivado de TLS. Utiliza una semántica parecida, adaptada para un bajo ancho de banda de dispositivos móviles.

Los principales cambios son:

> Estructuras de datos comprimidos: En donde posiblemente los paquetes sean reducidos de tamaño por el uso de los campos de *bits*, descartando redundancia y truncando algunos elementos criptográficos.
>
> Nuevo formato de certificado: WTLS define un formato de certificado comprimido. El cual sigue ampliamente la estructura X.509 v3 de certificados, pero usa estructuras de datos más pequeñas.
>
> Diseño basado en paquetes: TLS está diseñado para uso sobre flujo de datos. WTLS adapta la idea para una red basada en paquetes. Una gran parte del diseño está basado en el hecho de que es posible usar un paquete de red como el de SMS como datos de transporte.

Pero WTLS fué substituido en el estándard de WAP 2.0 por la especificación End-to-end Transport Layer Security.

Para la autenticación de los usuarios, en el caso de la banca online, usualmente el usuario se autentifica mediante un identificador de usuario y un PIN. Y cuando quiere efectuar una transacción entonces se le solicita un segundo PIN, obtenido desde otro tipo de soporte. De hecho, ese segundo PIN quiere a la vez conseguir el objetivo de no repudio. Estas medidas estan reforzadas por otras como: la limitación en las oportunidades de introducir erróneamente alguno de los PINs, el cierre de sesión al cabo de pocos minutos de inactividad, etc. Estos sistemas todavía presentan bastantes oportunidades para el fraude en caso de que alguno de estos elementos cayera en manos de alguien malintencionado. Y aún hay ciertos navegadores que no permiten normalmente la inclusión de certficados de usuario para la autenticación.

La integridad de los datos también parece bastante bien resuelta actualmente, ya que ésta queda garantizada al mismo nivel que la privacidad por el hecho de que las comunicaciones hagan servir SSL, WTLS o SMS. En estos dos últimos casos se requiere de confianza en el operador móvil.

La autenticidad del servidor también ofrece todavía bastantes oportunidades para mejorar la seguridad. Los riesgos más frecuentes vienen dados por el hecho de que las URLs suelen ser demasiado largas y el usuario no está familiarizado o no pone atención a las indicaciones de conexión segura.

También a día de hoy, los principales ISPs no proporcionan un mecanismo seguro para validar la autenticidad de los servidores DNS. Teniendo en cuenta que el DNS conforma uno de los primeros pasos de toda conexión, supone un riesgo importante dejar en segundo plano la seguridad de éstos.

Es evidente que con SSL no podemos asegurar que nos conectamos allí donde toca, pero existen algunas opciones como TSIG, **DNSSEC** o DNSCurve para disipar este tipo de dudas.

Ahora, gracias a estas tecnologías, cuando un cliente de la red pide una dirección IP, se garantiza el grado genuino de la respuesta. Por supuesto, el salto entre el cliente y el primer servidor no está salvaguardado y teóricamente podría ser manipulado (véase figura 3). Aquellos responsables de la seguridad de una red tendrán que decidir cómo valorar este posible riesgo.

TSIG (Transaction SIGnature) es un protocolo de red definido en el RFC-2845. Usado principalmente por DNS para establecer autenticación en las actualizaciones a base de datos DNS dinámicas, aunque también puede ser usado entre servidores para las peticiones habituales. TSIG usa clave secreta compartida y una función de hash unidireccional para establecer identificación entre extremos de una conexión mediante criptografía.

DNSCurve usa criptografía de curvas elípticas para añadir una capa de seguridad en enlace sobre el estándar DNS de manera simple y escalable. Añade protección de clave pública a nivel enlace a paquetes DNS. No se sabe mucho sobre este protocolo desde que se inventó en Septiembre de 2008, al ser propuesto por Daniel J. Bernstein. No es tan conocido. DNSCurve, en concepto, es similar a TSIG al asegurar las comunicaciones con nombres de servidores y bastante diferente a DNSSEC, al asegurar éste los propios registros DNS.

DNSSEC (Domain Name System Security Extensions) es un conjunto de especificaciones de IETF para garantizar cierto tipo de información proporcionada por el DNS que dan de resultado un conjunto de registros y modificaciones de protocolo para establecer autenticación en el DNS. Añadiendo firmas de claves públicas a los registros DNS. Y definiendo así los registros de recursos de clave pública (DNSKEY), delegación de firma (DS), registro de firma digital (RRSIG) y negación de la existencia autenticada (NSEC).

Estas especificaciones siendo un conjunto de extensiones DNS que se envían a los clientes DNS, permiten:

> Autenticación de origen de datos DNS, el no repudio o irrenunciabilidad, permitiendo probar la participación del origen de la comunicación. De manera que el emisor no puede negar un envío porque el destinatario tiene pruebas de ello.
>
> La integridad de los datos, pero no la disponibilidad o la confidencialidad, refiriéndose a la corrección y completitud, permitiendo comprobar que no se ha producido manipulación alguna en los datos originales. Dicha comprobación se obtiene adjuntando al mismo otro conjunto de datos de comprobación de la integridad. Una función hash que genere una huella digital asociada a los datos es un mecanismo que aporta esta característica.
>
> Autenticación de la negación de la existencia, para proporcionar protección contra enumeración de zonas, siendo el nombre del propietario original el que viene en forma de hash criptográfico utilizado al comienzo como una sola etiqueta para el nombre de cada zona.

En resumen, la resolución DNSSEC comprueba si la consulta corresponde a una zona asegurada con DNSSEC. La respuesta es positiva cuando el destino se encuentra en zona segura. Los nodos superiores de estas estructuras son los SEP (Secure Entry Points). El SEP designa como región segura la jerarquía de la zona por debajo de él. La adición de estos SEPs a la configuración del servidor de DNS con DNSSEC es responsabilidad del administrador de la organización.

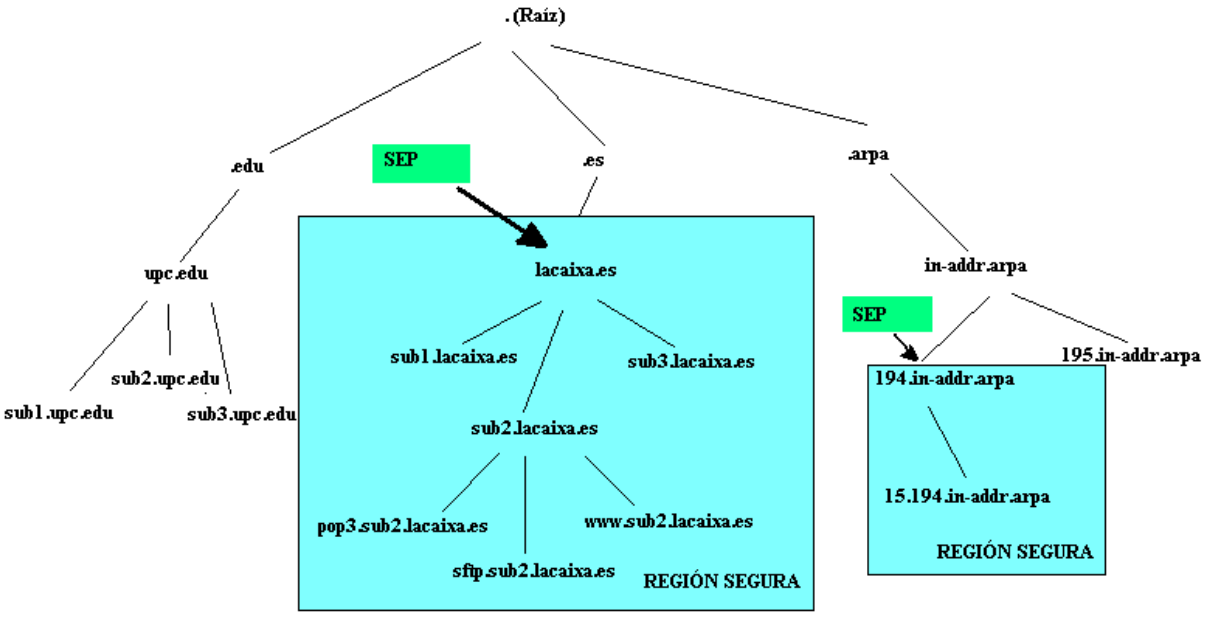


Figura 3

Por tanto, la lista de SEPs es el equivalente en la práctica a los proveedores de certificados CA de los navegadores web. Para ello se puede hacer uso de un DLV, un repositorio para SEPs.

Debido a la estructura de cadena jerárquica en que hasta ahora se han resuelto nombres con el sistema de DNS, DNSSEC sólo podrá garantizar la integridad en aquellos niveles en los que actúe. Una solución completa requeriría, así pues, la adopción a escala global de DNSSEC. De lo contrario se crean lo que se denominan islas de seguridad.

Ofreciendo tantos beneficios, y desde 1999 que fué creado, cómo es que no se llevó a nivel mundial?

Pues resulta que el despliegue de DNSSEC específicamente se ha visto obstaculizado por diversas dificultades, cómo:

- La elaboración de una norma compatible hacia atrás que pueda escalar el tamaño de Internet.

- Impedir la enumeración de zonas.

- El despliegue a nivel mundial de aplicaciones DNSSEC en una amplia variedad de servidores DNS y la resolución por parte de los clientes.

- Los mútliples estándares que han salido.

- El desacuerdo entre los actores clave sobre quién debería poseer los *top-level domains*.

- La superación de la percepción de la complejidad de DNSSEC y su despliegue.

- Los costes y su gestión.

Algunos de estos problemas están en proceso de ser resueltos, y los despliegues en diversos ámbitos han comenzado a tener lugar, como en los dominios .gov y .mil, en el año 2009.

3. Diseño

Esta sección establecida con el fin de hacer un proceso previo de configuración mental "prefiguración" en la búsqueda de una solución final a las necesidades anteriormente expuestas, nos mostrará los objetos a implementar en una simulación que estableceremos previa a la solución final.

3.1. Esquema de la red

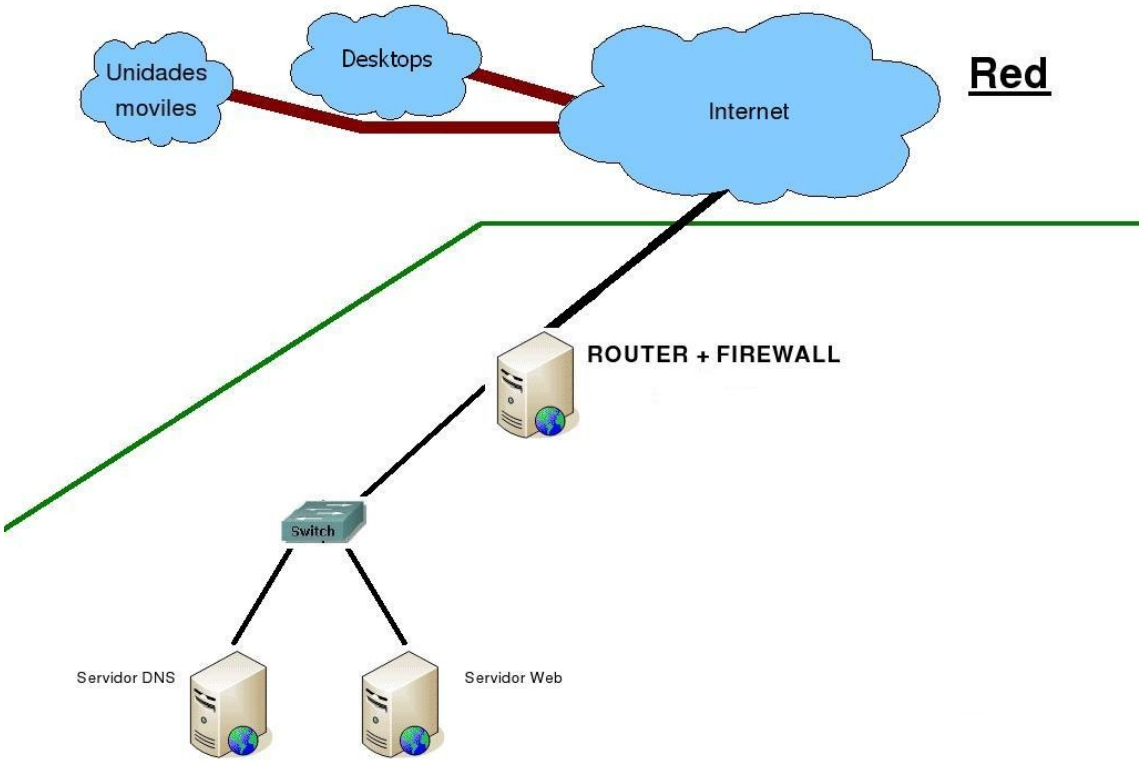


Figura 4

Porqué este esquema de red?

Con este esquema de red pretendemos obtener la seguridad necesaria y a su vez separar lo que son servicios internos de los servicios externos.

Como podéis ver en el esquema tenemos:

- Un servidor DNS y un servidor Web.

- Un *router* que también hará de *firewall*.

- Una red interna que estará formada por el *Switch*, el servidor DNS, el Web y el *Router*.

- Como conexiones de entrada desde el exterior o Internet tenemos Unidades móviles y *Desktops*.

Esquema IP de red

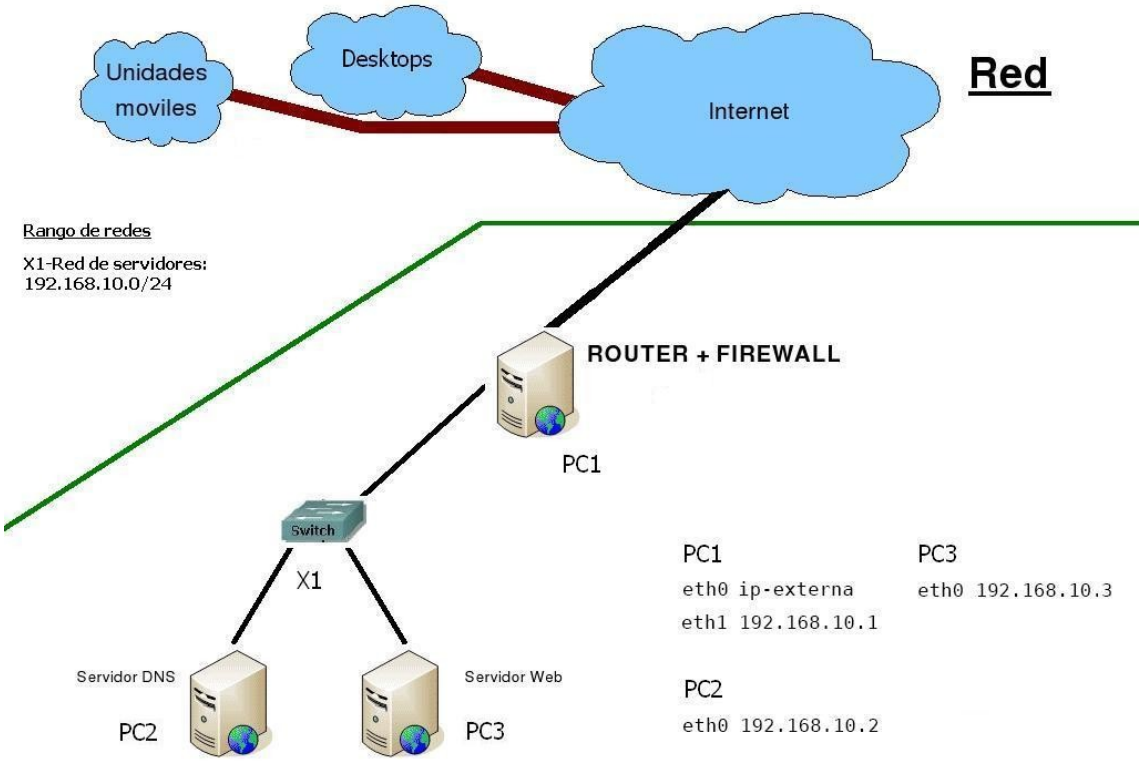


Figura 5

Red interna formada por el *Switch*, el servidor DNS, el Web y el *Router*.

3.2. Servidores

DNS
Con sistema operativo Debian GNU/Linux 5.0 de instalación mínima.

Y servidor DNS BIND 9 con soporte para Secure Extensions.

Web
Con sistema operativo Debian GNU/Linux 5.0 de instalación mínima.

Y servidor Web Apache2.

Router
Con sistema operativo Debian GNU/Linux 5.0 de instalación mínima.

Firewall+Router con reglas *iptables*, que se encargará de distribuir las conexiones y permitirá ciertas conexiones externas hacia la red local.

3.3. Transmisión de datos

Como ya sabemos, como conexiones de entrada desde el exterior o Internet tenemos Unidades móviles y *Desktops*. Para proteger la transmisión de datos de estas conexiones, en un principio de protocolo HTTP, hay dos opciones que son claras candidatas, IPSec y SSL. Si se quisiera implantar una VPN se podría elegir cualquiera de las dos opciones para la implantación de una VPN segura, formando una red privada entre el cliente y el servidor de la entidad, en función de sus niveles de seguridad y de las necesidades de acceso.

Aunque lo bueno sería, tal y como varios organismos de estandarización (OMA, JCP, GSMA, 3GPP) hacen, que están trabajando para que fabricantes de móviles adopten mecanismos que permitan, por ejemplo, incluir la autenticación de usuario con PKI y además aprovechando un Secure Element como la SIM o algún token externo, basándose en aprovechar principalmente la infraestructura del operador móvil. De manera que la autenticación sea menos invasiva para el usuario. Sin embargo, sería algo difícil por el continuo cambio de propietario y adquisición de nuevos móviles, sustracción, etc. Además está el tema de la usabilidad de darse de alta y baja de la entidad financiera, tener en cuenta la clave en cada conexión, al tratarse de millones de usuarios debería haber una gran labor de concienciación social detrás que respaldara estas medidas. Y teniendo consciencia de cómo son las personas, que hoy en día se cambia antes de móvil que de contraseñas de navegación de internet, la implantación de una VPN al final no parece tan buena idea.

En realidad IPSec y SSL a menudo son vistos como opciones excluyentes, pero pueden ir juntos y ser complementarios. Cada uno tiene sus propias ventajas y limitaciones; cada uno funciona de manera más eficaz en escenarios concretos. Pero que no sea homogéneo el acceso remoto de diferentes tipos de usuarios, no implica que deba elegirse uno u otro exclusivamente. Utilizar ambos es lo ideal.

Es por éso, que lo primero que se pregunta uno es qué nivel de control de acceso se desea. La decisión de dónde y cuándo instalar uno u otro protocolo estará en función de los niveles de acceso y del tipo de usuario al que se desea otorgar dicho acceso.

Si se quiere conceder el acceso a todas las aplicaciones y recursos dentro de un segmento de la red a todos los usuarios en una VPN, IPSec será la opción adecuada. Algunas IPSec VPN utilizan LDAP para proporcionar mayor control de acceso granular basado en decenas o miles de perfiles de usuarios. Y si se precisa asignar un alto nivel de encriptación y autenticación, IPSec asegura que los clientes se encuadren dentro de la política de seguridad corporativa.

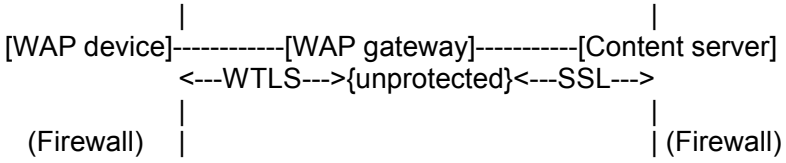
Pero si se quiere controlar el acceso a aplicaciones específicas, SSL sería la elección ideal, porque asegura las comunicaciones desde el usuario al servidor de la aplicación, en vez de desde el usuario al punto de entrada de la zona asegurada.

Si lo que se desea es alta seguridad controlada y extremo a extremo SSL sobre IPSec.

En este caso nos bastará con SSL y WTLS. De hecho, para el entorno WAP, SSL únicamente se utiliza entre el servidor web y el gateway WAP. Entre el gateway WAP y el dispositivo WAP, se utiliza WTLS.

A pesar de que SSL y WTLS proporcionan seguridad suficiente para la mayoría de aplicaciones, existen potenciales problemas de seguridad dónde se encuentran los dos protocolos, en el gateway WAP.

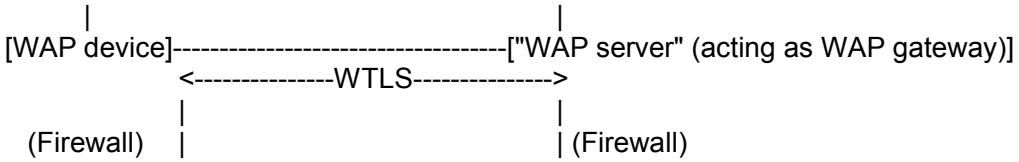
SSL no es directamente compatible con WTLS, así que el gateway WAP debe desencriptar el flujo de datos protegidos por SSL provinientes del servidor web y reencriptarlos usando WTLS antes de pasar los datos al dispositivo WAP. Dentro de la memoria del gateway WAP los datos suelen estar desprotegidos.



Así es, imaginemos que la entidad financiera hace público un servicio de WAP. Cuando los datos dejen la seguridad de sus sistemas y su red, estarán razonablemente bien protegidos por el SSL. Entonces, cuando entren el gateway WAP, el cual suele ser de terceros, como por ejemplo las operadoras de telefonía móvil, los datos se desencriptan. Lo cual no significa que los operadores de telefonía representen un gran riesgo de seguridad, pero se ha de ser consciente en que compañía se confía. Ya que esta podría ser cualquier compañía del mundo, y estando de vacaciones en algún país en el que la seguridad no ocupe un espacio importante, lo más probable es que la red del operador sea vulnerable a ataques y por lo tanto los datos también.

Es por ese motivo que todos los actores principales WAP están desarrollando soluciones a este problema, pero por ahora esas soluciones crean otros problemas. Una medida sería los llamados “servidores WAP”, o servidores web con capacidad de gateway WAP, que proporcionan seguridad extremo a extremo de manera que el flujo de datos abandona el servidor de contenido (el servidor WAP) ya encriptado con WTLS.

De esta manera:



Aunque sacando fuera de la cadena el gateway WAP del operador de telefonía móvil, de modo que el usuario tuviera que reconfigurar su dispositivo WAP para apuntar al “servidor WAP”, que sería el gateway WAP para esa sesión. Pero este gateway WAP solo aportaría acceso a ese servicio, y cuando el usuario quisiera acceder a cualquier otro *site* WAP, tendría que reconfigurar su móvil otra vez. Algunos dispositivos que utilicen WAP son fácilmente configurables por los usuarios si leen el manual, pero otros dispositivos no lo son tanto.

Aunque muchos de los operadores de telefonía móvil no ofrecen el servicio de WAP si la conexión no se hace exclusiva a su gateway WAP. Se han buscado soluciones a esta cuestión, pero las últimas versiones del protocolo WAP ya ofrece por la especificación End-to-end Transport Layer Security descrita en el documento WAP-187-TransportE2ESec-20010628-a una seguridad más que suficiente, de modo que no se plantee tanto riesgo y salga más a cuenta otros objetivos de ataque que no un gateway WAP.

En cuanto a la autenticidad de los servidores DNS para la resolución de nombres - @ip en la transmisión de datos, por qué escoger DNSSEC y no TSIG o DNSCurve?

Con TSIG no se hizo la previsión de la distribución de claves secretas. Se espera que sea el administrador quién configure estáticamente los nombres de servidores y clientes utilizando algún mecanismo en desuso como un *sneaker-net* hasta la llegada de un mecanismo seguro automatizado de distribución de claves. Además de que típicamente no es usado para todas las peticiones, principalmente para las actualizaciones de registros DNS. Y sólo sirve para la autenticación.

DNSCurve, diseñado conservando los paquetes de tamaño pequeño para la autenticación y encryptación. Es un protocolo no muy conocido a día de hoy, seguramente debido a que no salió hace mucho tiempo. Pero en el futuro es posible que juegue grandes bazas. En la página web del proyecto se puede encontrar en tablas una comparación con DNSSec:
http://dnscurve.org/dnssec.html
(Véase también tablas adjuntas en el Anexo)

DNSCurve y DNSSEC tienen metas de seguridad complementarias. Si ambos fueran ampliamente desplegados entonces cada uno proporcionaría algo de seguridad que el otro no proporciona.

DNSSEC sin embargo, ofrece las garantías de seguridad suficientes, al tiempo que mantiene compatibilidad hacia atrás y puede coexistir con DNS sin extensiones de seguridad, es aplicable a la mayoría de servidores DNS y está cada vez más extendido. Aunque no es una solución sencilla de implementar, se puede encontrar un compromiso entre simplicidad y seguridad.

Pero veamos cuáles son estas garantías de seguridad:

3.4. Los entresijos de DNSSEC

Cómo ya se ha dicho DNSSEC ofrece un conjunto de registros y modificaciones de protocolo para garantizar la información proporcionada por el DNS. DNSSEC usa criptografía de clave pública para firmar y autenticar conjunto de registros de recurso.

Aunque en realidad, a nivel *externo* de la red de la entidad, la seguridad adicional la proporciona el cliente con capacidad para DNSSEC al validar las firmas de los registros de recurso. Si una respuesta no está debidamente firmada, se ignora. Se trata de un procedimiento muy seguro, ya que nunca se insta al usuario a aceptar una respuesta potencialmente comprometida. De todos modos, el usuario debe acostumbrarse a las respuestas NXDOMAIN, que significan que el dominio no existe, aunque estas respuestas dependen de la configuración del servidor.

En contraste con los certificados web, que cuando navegamos en la misma situación se abre una ventana y el usuario puede decidir qué hacer con el certificado no válido; por desgracia, muchos usuarios simplemente ignoran el aviso.

De DNSSEC existen plugins o parches para navegadores, como puede ser Firefox, que comprueban las resoluciones DNS hechas dentro del navegador y suele aparecer en él una pantalla que muestra al usuario que se realizan muchas búsquedas de DNS con el fin de mostrar la página actual. En ella también se suelen resumir las búsquedas en función de si fueron validadas, de confianza debido a la política o no, debido a errores de validación de DNSSEC, con tal de mantener al usuario informado de que está visitando un sitio con extensiones de seguridad. También suele haber una pantalla de configuración que permite desactivar la validación de DNSSEC en tiempo de ejecución.

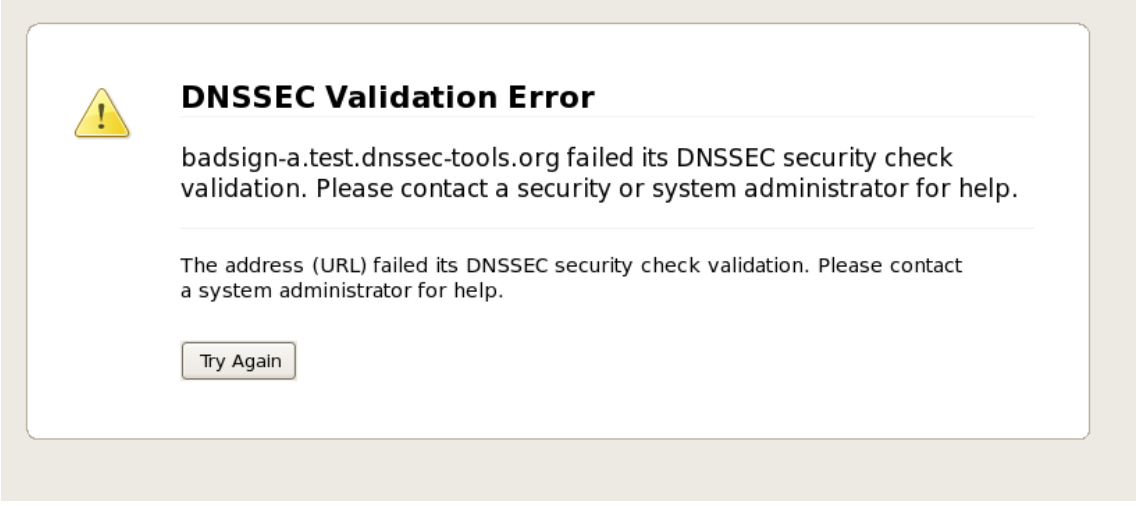


Figura 6

Figura 7

Cuando la respuesta no proviene de una zona segura, el cliente recurre a los métodos tradicionales para la resolución, devolviendo la respuesta oportuna al cliente que hace la petición. Los administradores encargados de la seguridad deben tener en cuenta que, si están usando DNSSEC, el usuario no podrá distinguir entre las respuestas que están autenticadas mediante DNSSEC y las que no lo están, a no ser que tengan un alguna herramienta.

Y a nivel *interno de la red de la entidad*, a diferencia de a nivel *externo* en el que la estructura PKI no se ve aprovechada, visto que no se ha desplegado DNSSEC a nivel global, la seguridad se ve realmente cumplida al cumplirse la cadena de confianza y evitar ataques de hombre en el medio, tal y como *externamente* existe la potencial posibilidad, al poder un atacante intentar crear una firma o una cadena de confianza autofirmada e intentar hacerse pasar por la entidad. Aún así, a nivel *interno*, se deben de tomar otras medidas de seguridad para evitar la copia de firmas o certificados que compongan la cadena de confianza por un atacante.

En términos generales, tal y como se ha visto en el estudio previo de esta memoria, en DNSSEC se asegura de que se tenga un solo SEP apuntando a la raíz jerárquica de los niveles a asegurar. Una cadena de confianza enlaza la clave de la firma con el resto de zonas que están por debajo en la jerarquía, en niveles inferiores, cosa que permite a los clientes de DNSSEC validar las firmas.

En el Internet de hoy día se podrían intercalar otras regiones seguras independientes. Por lo que mientras estas regiones no crezcan juntas, clientes y administradores tendrán que gestionar como SEPs, múltiples claves de confianza. Que podrían ser guardadas en un DLV, un repositorio de SEPs.

Veamos ahora en que consisten los registros:

Cómo ya sabemos se llama Registro de Recurso (RR) a cada unidad de información del DNS. Cada registro tiene un tipo asociado que describe el dato que contiene y una clase que especifica el tipo de red al que se aplica. Algunos de ellos encontraremos que tienen un tiempo de vida (o TTL), esto es así para definir cuanto tiempo un servidor DNS quiere que se guarde la información de un RR en caché. El tiempo especificado es en segundos. Y un valor común suele ser 86400 segundos, lo cual es 24 horas. Por ejemplo, eso querría decir que si un RR se cambia, los servidores DNS de alrededor del mundo podrían mostrar el valor anterior durante un máximo de 24 horas después del cambio.

Así con DNSSEC se pasan a definir los RRs de clave pública (DNSKEY), delegación de firma (DS), registro de firma digital (RRSIG), y negación de la existencia autenticada (NSEC).

DNSKEY
Es un registro que contiene claves criptográficas. El valor de su tipo es 48. La clase es independiente. Y no requiere de un TTL especial.

Las claves criptográficas que contiene son usadas para firmar registros asociados a un dominio en el archivo de zona. Estas claves pueden ser de tipo KSK (Key Signing Key) o ZSK (Zone Signing Key). El protocolo de validación de DNSSEC no distingue entre los diferentes tipos.

En la práctica, se usa ambos y se distinguen con el flag llamado SEP (Secure Entry Point), es decir, el flag que indica que se trata del primer servidor de una red asegurada con DNSSEC. El *bit* 15 del campo Flags de DNSKEY corresponde a SEP y está descrito en el RFC-3757. SEP será un número impar (un uno) cuando DNSKEY contenga una clave destinada al uso del llamado punto de entrada seguro. Este flag únicamente está destinado a ser una pista para las condiciones de uso del registro en cuanto a la firma de zonas o depuración software. Cuando sea impar se asimilará que se trata de KSK y par cuando se trate de ZSK. El SEP, que indica que se trata del primer servidor de la red asegurada, indicaría también el vértice de la isla de seguridad.

Pero además de que se tenga el *bit* SEP establecido (con valor uno), también se necesitará el flag Zone Key establecido (a uno) con el fin de ser capaz de generar firmas legalmente. De no ser así, estando el flag Zone Key no establecido, no debería usarse para la verificación de RRSIGs que cubran conjuntos de RRs (RRsets). El *bit* 7 del campo Flags es el que corresponde al flag Zone Key. Si el *bit* 7 tiene valor uno, entonces el registro DNSKEY contiene una clave de zona DNS y el nombre del propietario del registro debería ser el nombre de la zona. Si el *bit* 7 tiene valor cero, entonces DNSKEY contiene otro tipo de clave pública de DNS y no debe ser usada para la verificación de RRSIGs que cubran RRsets.

Como decía, las claves pueden ser de tipo KSK o ZSK, y estos tipos vienen definidos en el RFC-4641. Las KSKs se utilizan sólo para firmar las claves contenidas dentro de una zona. Son almacenadas en DNSKEY y son usadas en el proceso de autenticación de DNSSEC descrito en RFC-4035. Debido a que son usadas para firmar menos datos, su vida útil criptográfica puede ser bastante duradera, antes de que se necesite crear unas nueva. También pueden ser más duraderas debido a que las firmas ya producidas a través de su uso sólo se adjuntarán a un único RRset dentro de una zona, el RRset DNSKEY. Las KSKs son las claves previstas a ser configuradas para su uso por un agente de validación de resolución de Trust Anchors.

Un agente de validación de resolución es un simple agente de resolución DNS que sabe como llevar a cabo la validación DNSSEC de los datos que recibe. Cuando se configura con un Trust Anchor, puede asegurar a su cliente que los datos que recibe no han sido afectados por amenazas contra las que DNSSEC fué diseñado para proteger.

Qué son los Trust Anchors?

Para entender que son los Trust Anchors o anclas de confianza, lo primero es entender el concepto de origen de confianza.

El origen de confianza se basa en que para validar el primer certificado de una cadena de certificados se necesita conocer una autoridad de certificación y su clave pública. A partir de esa clave pública se van validando los sucesivos certificados de la cadena. La validez del proceso descansa sobre la confianza en el primer certificado. Es frecuente que el primer certificado sea autofirmado.

Una ancla de confianza viene definida de diversas maneras según los estándares:

- Ancla de confianza:
Es una clave pública y el nombre de una autoridad de certificación que es usado para validar el primer certificado en una secuencia de certificados. La clave pública de la ancla de confianza es usada para verificar la firma en un certificado expedido por una autoridad certificadora de anclas de confianza. La seguridad del proceso de validación depende de la autenticidad e integridad de la ancla de confianza. Las anclas de confianza son a menudo distribuidas como certificados autofirmados.
[SP800-57:2007] Recommendation for Key Management - Part 1: General, NIST Special Publication 800-57, March 2007

- Ancla de confianza:
Información de confianza, la cual incluye un algoritmo de clave pública, un valor de clave pública, un nombre de emisor, y opcionalmente, otros parámetros.
NOTA 1: Otros parámetros pueden incluir, pero no están limitados a un período de validez.
NOTA 2: Una ancla de confianza puede venir dada en forma de certificado autofirmado.
[19790:2006] ISO/IEC 19790:2006, Information technology -- Security techniques -- Security requirements for cryptographic modules. 2006.

- Ancla de confianza:
Se trata de un conjunto de la siguiente información adicional a la clave pública: identificador de algoritmo, parámetros de clave pública (si se aplican), nombre distinguido del titular de la clave privada asociada (es decir, la autoridad de certificación sujeto) y facultativamente un período de validez. La ancla de confianza puede presentarse en la forma de un certificado autofirmado. Un sistema que utiliza un certificado puede confiar en una ancla de confianza y puede aplicarla para validar certificados en los trayectos de certificación.
[X.509:2005] ITU-T X.509, ISI/IEC 9594-8, Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks. 08/2005.

- Ancla de confianza:
Un RR DNSKEY o un hash RR DS de un RR DNSKEY. Un agente de validación consciente de la seguridad de resolución usa esta clave pública o hash como punto de entrada para construir una cadena de autenticación a una respuesta DNS firmada. En general, un agente de validación de resolución habrá de obtener los valores iniciales de sus anclas de confianza mediante algunos seguros o medios de confianza fuera del protocolo DNS.
La presencia de anclas de confianza también implica que el agente de resolución debería esperar la zona la cual la ancla de confianza apunta a ser firmada.
[RFC-4033] DNS Security Introduction and Requirements. 03/2005

Nos quedaremos con esta última definición que es la que tiene más relación con DNSSEC, vendría a decir pues, que una ancla de confianza sería un DNSKEY, normalmente una KSK que esté situada en un agente de validación de resolución, de modo que se pueda validar criptográficamente los resultados a una petición hacia atrás a una clave pública conocida, la ancla de confianza.

Por otra parte las ZSKs son las claves utilizadas para firmar cada RRset de una zona. De modo que una zona firma sus RRsets autoritativos usando una clave privada y almacena la correspondiente clave pública en un RR DNSKEY. Un agente de resolución puede entonces usar la clave pública para validar firmas que cubran los RRsets en la zona, y ésta para autenticarlas. Las ZSK necesitan ser cambiadas con mayor frecuencia debido a que son usadas para firmar más datos.

De modo que en DNSKEY tenemos las KSK que firman el conjunto de claves de una zona y también están las claves públicas correspondientes a cada una de las ZSKs.

DNSKEY no está previsto como registro para almacenar claves públicas arbitrarias y no debe ser usado para almacenar certificados o claves públicas que no sean directamente relacionadas a la infraestructura DNS. De hecho, aparte de proteger direcciones IP, DNSSEC puede proteger otra información cómo certificados criptográficos de propósito general almacenados en RRs CERT. El RFC-4398 describe como distribuir estos certificados, incluyendo los que se utilizan para e-mail, haciendo posible el uso de DNSSEC como infraestructura de clave pública mundial para e-mail.

El siguiente RR DNSKEY almacena una clave de zona para example.com.:

```
example.com. 86400 IN DNSKEY 256 3 5 ( AQPSKmynfzW4kyBv015MUG2DeIQ3
                                       Cbl+BBZH4b/0PY1kxkmvHjcZc8no
                                       kfzj31GajIQKY+5CptLr3buXA10h
                                       WqTkF7H6RfoRqXQeogmMHfpftf6z
                                       Mv1LyBUgia7za6ZEzOJBOztyvhjL
                                       742iU/TpPSEDhm2SNKLijfUppn1U
                                       aNvv4w== )
```

El primer parámetro:
example.com. 86400 IN DNSKEY 256 3 5
indica el nombre del propietario.

El segundo parámetro:
example.com. **86400** IN DNSKEY 256 3 5
indica el TTL.

El tercero:
example.com. 86400 **IN** DNSKEY 256 3 5
la clase.

El siguiente:
example.com. 86400 IN **DNSKEY** 256 3 5
el tipo de RR.

example.com. 86400 IN DNSKEY **256** 3 5
El 256 indica que el *bit* de Zone Key, el que pertenece al *bit* 7, en el campo de Flags tiene valor uno.

example.com. 86400 IN DNSKEY 256 **3** 5
El valor 3 es un valor fijo del protocolo.

example.com. 86400 IN DNSKEY 256 3 **5**
El valor 5 indica el algoritmo de clave pública. En este caso el 5 corresponde al RSA/SHA1 (el formato del campo de clave pública RSA/SHA1 viene definido en el RFC-3110).

El texto restante es una codificación en Base64 de la clave pública:

```
example.com. 86400 IN DNSKEY 256 3 5( AQPSKmynfzW4kyBv015MUG2DeIQ3
                                      Cbl+BBZH4b/0PY1kxkmvHjcZc8no
                                      kfzj31GajIQKY+5CptLr3buXA10h
                                      WqTkF7H6RfoRqXQeogmMHfpftf6z
                                      Mv1LyBUgia7za6ZEzOJBOztyvhjL
                                      742iU/TpPSEDhm2SNKLijfUppn1U
                                      aNvv4w== )
```

RRSIG
Es un registro de firma digital que contiene el resultado de la firma de un conjunto de RRs asociados a un dominio, habiéndose usado para ello un RR DNSKEY. El valor de su tipo es 46. La clase es independiente, aunque ha de tener la misma clase que el RRset que cubra. El valor de TTL ha de corresponderse con el TTL del RRset que cubre.

Las firmas digitales almacenadas en el registro son usadas en el proceso de autenticación DNSSEC descrito en el RFC-4035. Un validador puede usar este RR para autenticar RRsets de una zona. RRSIG sólo debe ser usado para firmas digitales y garantizar las operaciones DNS. Este registro no debería firmarse, ya que la firma por si sola no le añadiría ningún valor y podría crear un bucle infinito en el proceso de firma.

Un RRSIG contiene la firma para un RRset con un nombre, una clase y un tipo especiales. El registro especifica un intervalo de validez para una firma y usa el Algoritmo, el Nombre del firmante, y el Key Tag para identificar un RR DNSKEY que contenga una clave pública que un validador pueda usar para verificar la firma.

El Key Tag se usa como ayuda para que la selección del RR DNSKEY sea más eficiente, pero no es un identificador único. Es posible que dos RRs DNSKEY diferentes tengan el mismo nombre de propietario, el mismo tipo de algoritmo y el mismo Key Tag. Una implementación que use únicamente el Key Tag para seleccionar el RR DNSKEY podría seleccionar una clave pública no esperada. Así, el Key Tag sirve para limitar los posibles candidatos. Es por eso que en la mayoría de casos se usa una combinación de nombre de propietario, algoritmo y Key Tag. Sobre el cálculo del Key Tag, no es preciso entrar en ciertos detalles del diseño de DNSSEC, por lo que para más información consúltese el Apéndice B del RFC-4034. El RR DS tiene un Key Tag y ocurre lo mismo como veremos más adelante.

El siguiente RR RRSIG almacena una clave de zona para example.com.:

```
host.example.com. 86400 IN RRSIG A 5 3 86400 20030322173103 (
                        20030220173103 2642 example.com.
                        oJB1W6WNGv+ldvQ3WDG0MQkg5IEhjRip8WTr
                        PYGv07h108dUKGMeDPKijVCHX3DDKdfb+v6o
                        B9wfuh3DTJXUAfI/M0zmO/zz8bW0RznI8O3t
                        GNazPwQKkRN20XPXV6nwwfoXmJQbsLNrLfkG
                        J5D6fwFm8nN+6pBzeDQfsS3Ap3o= )
```

Los primeros cuatro campos especifican como siempre, el nombre del propietario, el TTL, la clase y el tipo de RR.

host.example.com. 86400 IN RRSIG **A** 5 3 86400 20030322173103
La A indica el campo Type Covered. Éste identifica el tipo de RRset que está cubierto por RRSIG.

host.example.com. 86400 IN RRSIG A **5** 3 86400 20030322173103
El valor 5 indica el algoritmo usado (RSA/SHA1) para crear la firma.

host.example.com. 86400 IN RRSIG A 5 **3** 86400 20030322173103
El valor 3 del campo Labels es el mismo valor, al número de etiquetas en el nombre de propietario del RRset, sin contar la etiqueta null root y sin contar la eitqueta de más a la izquierda si es un comodín (*).

host.example.com. 86400 IN RRSIG A 5 3 **86400** 20030322173103
El valor 86400 es el TTL Original para el RRset A cubierto.

host.example.com. 86400 IN RRSIG A 5 3 86400 **20030322173103**
20030322173103 la fecha de expiración.

Dentro de las llaves el contenido sería:
20030220173103 fecha inicial, **2642** es el Key Tag y **example.com.** es el nombre del firmante que es igual al nombre de la zona que contiene el RRset. El texto restante es una codificación en Base64 de la clave.

Nótese que la firma puede ser autenticada usando un RR DNSKEY de la zona example.com cuyo algoritmo es 5 y cuyo Key Tag es 2642. El algoritmo, el nombre del firmante y el Key Tag identifican el RR DNSKEY de la zona indicada. También se ha de tener en cuenta que el RRset NS que aparece en el nombre de la zona indicada debería ser firmado, pero los RRsets NS que aparecen en los puntos de delegación (es decir, los RRsets NS en la zona padre que delega el nombre a los nombres de servidores de las zonas hijas) no deberían ser firmados. Debería de haber un RRSIG para cada RRset utilizando al menos un RR DNSKEY para cada algoritmo en el RRset DNSKEY de la zona indicada. Este RRset DNSKEY debería ser firmado por cada algoritmo que aparezca en el RRset DS localizado en el padre de la delegación (si lo hay).

NSEC
El registro lista dos cosas separadas: el siguiente nombre de propietario (en orden canónico de la zona), o siguiente entrada del archivo de zona, que contenga datos autoritativos o un punto de delegación RRset NS, y el conjunto de tipos de RR presentes en el nombre del propietario de NSEC tal y como se describe en el RFC-3845. Cuando llega al último registro del archivo de zona, NSEC apunta a la primera entrada. De este modo se asegura que nadie pueda borrar entradas, puesto que el registro NSEC está firmado también con RRSIG. El valor de su tipo es 47. La clase es independiente. El valor del TTL debería ser el mismo que el TTL mínimo de SOA, tal y como se describe en el RFC-2308.

El conjunto completo de RRs NSEC en una zona indica qué RRsets autoritativos existen en una zona y también forma una cadena de nombres de propietarios autoritativos en la zona. Esta información es usada para proporcionar denegación de existencia autenticada para datos DNS, como se describe en el RFC-4035. NSEC se ha convertido en uno de los mayores problemas para la implementación generalizada de DNSSEC, debido a que posibilita el *zonewalking*, es decir, la consulta de todos los registros de un archivo de zona de forma sucesiva.

El siguiente RR NSEC identifica los RRsets asociados a alfa.example.com. e identifica el siguiente nombre autoritativo después de alfa.example.com.

```
alfa.example.com. 86400 IN NSEC host.example.com. (
                              A MX RRSIG NSEC TYPE1234 )
```

Los primeros cuatro campos especifican como siempre, el nombre, el TTL, la clase y el tipo de RR. La entrada host.example.com. es el siguiente nombre autoritativo después de alfa.example.com en orden canónico. Los nemotécnicos A, MX, RRSIG, NSEC y TYPE1234 indican que hay los RRsets A, MX, RRSIG, NSEC y TYPE1234 asociados al nombre alfa.example.com. Estos nemotécnicos constituyen el campo Type Bit Maps.

La sección RDATA, el resto a partir del tipo del RR NSEC de encima, sería codificada de la siguiente manera:

```
0x04 'h'  'o'  's'  't'
0x07 'e'  'x'  'a'  'm'  'p'  'l'  'e'
0x03 'c'  'o'  'm'  0x00
0x00 0x06 0x40 0x01 0x00 0x00 0x00 0x03
0x04 0x1b 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x20
```

Pudiéndose autenticar este registro, podría ser usado para probar que beta.example.com no existe, o probar que no hay un registro AAAA asociado a alfa.example.com.

Cada nombre de propietario en la zona que tenga datos autoritativos o un RRset NS que haga de punto de delegación debe tener un RR NSEC. Un registro NSEC (y su RRset RRSIG asociado) no debería ser el único RRset con ese nombre. Es decir, el proceso de firma no debe crear RRs NSEC o RRSIG para nodos con nombres de propietario que no sean el nombre de propietario de algún RRset antes de que la zona sea firmada. La razón principal es por la coherencia del espacio de nombres entre las versiones firmadas y no firmadas de una misma zona y la intención de reducir el riesgo de respuestas incoherentes en servidores recursivos ajenos.

El campo de Type Bit Maps de cada NSEC en una zona firmada debe indicar la presencia de ambos, el registro NSEC en sí mismo y el correspodiente RRSIG.

La diferencia entre el conjunto de nombres de propietarios que requiere el registro RRSIG y el conjunto de nombres de propietario que requiere el registro NSEC es sutil y digno de destacar. El registro RRSIG está presente en el nombre de propietario de todos los RRsets. El registro NSEC está presente en el nombre de propietario de todos los nombres para los cuales la zona firmada es autoritativa y también en el nombre de propietario de las delegaciones desde la zona firmada hasta sus hijos. Ni NSEC ni RRSIG están presentes (en la zona padre) en los nombres de propietario de *RRsets de enlace*[1]. Nótese sin embargo, que esta distinción es para la mayor parte visible sólo durante el proceso de firma de zona, ya que los RRsets NSEC son datos autoritativos y están por lo tanto firmados. De este modo, ningún nombre de propietario que tenga un RRset NSEC tendrá RRs RRSIG, al igual que en la zona firmada.

El *bitmap* para el registro NSEC en un punto de delegación requiere una especial atención. Los *bits* correspondientes a la delegación RRset NS y cualquier RRsets para los cuales la zona padre tenga datos autoritativos deben estar establecidos; los *bits* correspondientes a cualquier RRset que no sea el NS para los cuales el padre no sea autoritativo no deben estar establecidos.

DS

Es el registro destinado a la delegación de firma. Establece cadenas de autenticación entre zonas DNS. El valor de su tipo es 43. La clase es independiente. No se requiere de un valor en particular para el TTL. El DS referencia a un RR DNSKEY y es usado en el proceso de autenticación. Un DS referencia a un RR DNSKEY mediante el almacenamiento del Key Tag, el número de algoritmo y un resumen del RR DNSKEY.

1 Un RR de enlace en una delegación es un RR usado para localizar los servidores DNS autoritativos para la zona delegada. Estos registros son usados para enlazar y juntar zonas y proporcionar una delegación efectiva y una ruta de referencia para que otros servidores DNS la sigan a la hora de la resolución de nombres.

Nótese que mientras el resumen debería ser suficiente para identificar la clave pública, almacenando el Key Tag y el algoritmo de clave se ayuda a hacer el proceso de identificación más eficiente. Mediante la autenticación del DS, un agente de resolución puede autenticar el RR DNSKEY al que apunta DS. El proceso de autenticación de clave está descrito en el RFC-4035.

El DS y su correspondiente RR DNSKEY tienen el mismo nombre de propietario, pero están almacenados en sitios diferentes. El DS sólo aparece en la parte superior (los padres) de una delegación y los datos son autoritativos en la zona padre.

Si por ejemplo, el DS para “example.com” se almacena en la zona “com” (la zona padre) más que en la zona “example.com" (zona hija). Y el correspondiente RR DNSKEY se almancena en la zona "example.com" (zona hija). Ésto simplifica la administración de zonas DNS y la firma de zonas, pero introduce unos requerimientos de procesado de respuesta especiales para el DS; lo cual se describe en el RFC-4035.

El siguiente ejemplo muestra un RR DNSKEY y su correspondiente RR DS:

```
dskey.example.com. 86400 IN DNSKEY 256 3 5 ( AQOeiiR0GOMYkDshWoSKz9Xz
                                fwJr1AYtsmx3TGkJaNXVbfi/
                                2pHm822aJ5iI9BMzNXxeYCmZ
                                DRD99WYwYqUSdjMmmAphXdvx
                                egXd/M5+X7OrzKBaMbCVdFLU
                                Uh6DhweJBjEVv5f2wwjM9Xzc
                                nOf+EPbtG9DMBmADjFDc2w/r
                                ljwvFw==
                                ) ;  key id = 60485

dskey.example.com. 86400 IN DS 60485 5 1 ( 2BB183AF5F22588179A53B0A
                                98631FAD1A292118 )
```

Los primeros cuatro campos especifican como siempre, el nombre, el TTL, la clase y el tipo de RR. El valor 60485 es el Key Tag para el correspondiente “dskey.example.com.” RR DNSKEY y el valor 5 indica el algoritmo usado por el “dskey.example.com.” RR DNSKEY. El valor 1 es el algoritmo usado para construir el resumen, y el resto del texto RDATA es el resumen en hexadecimal.

Nótese que un RRset DS debería estar presente en un punto de delegación cuando la zona hija esté firmada. El RRset DS debería contener múltiples registros, cada uno referenciando a una clave pública en una zona hija usada para verificar los RR RRSIGs en esa zona. Todos los RRsets DS en una zona deben estar firmados y los RRsets DS no deberían aparecer en el *corte de zona*[2] en el lado hijo.

2 El árbol DNS está dividido en “zonas”, las cuales son una colección de dominios que son tratados como una unidad para ciertos propósitos administrativos. Las zonas están limitadas por cortes de zona. Cada corte de zona separa una zona hija (por debajo del corte) de una zona padre (por encima del corte). El nombre de dominio aparece en la parte superior de la zona (justo debajo del corte que separa la zona de sus padres) es llamado origen de zona. El nombre de la zona es el mismo que el nombre del dominio en la zona origen. Cada zona comprende aquel subconjunto del árbol DNS que está en o por debajo de la zona origen, y que está más allá de los cortes que separan la zona de sus hijos (si los hay). La existencia de un corte de zona está indicado en la zona padre por la existencia de un registro NS especificando el origen de una zona hija. Una zona hija no contiene ninguna referencia explícita hacía su padre.

Un RR DS debería apuntar a un RR DNSKEY que esté presente en el DNSKEY RRset del hijo, y el DNSKEY RRset del hijo debería estar firmado por la correspondiente clave privada. Los RRs DS que fallen al encontrar estas condiciones no serán útiles para la validación, pero es debido al RR DS y sus correspondientes RR DNSKEY que están situados en zonas diferentes y debido a que el DNS es simplemente poco consistente al poderse producir desajustes temporales.

El TTL de un RRset DS debería coincidir con el TTL de un RRset NS de la delegación, es decir, el RRset NS de la misma zona que contenga el RRset DS.

La construcción de un RR DS requiere de conocimientos del RR DNSKEY correspodiente en la zona hija, lo cual implica una comunicación entre él, la zona hija y padre. Sobre esta comunicación no es preciso entrar en ciertos detalles del diseño de DNS.

Sobre protocolo

DNSSEC requiere de cambios en el protocolo DNS, entre ellos, un cambio en la definición del RR CNAME tal y como se describe en el RFC-1035. Este cambio principalmente permite a los registros RRSIG y NSEC tener el mismo nombre de propietario que el RR CNAME de una zona firmada. Ya que cada RRset autoritativo de una zona debe ser protegido por una firma digital, el RR RRSIG requiere tener el mismo nombre de propietario que el RR CNAME. Y puesto que cada nombre autoritativo de una zona debe ser parte de una cadena NSEC, el RR NSEC requiere tener el mismo nombre de propietario que el RR CNAME. Así se forma el cambio en la especificación DNS tradicional descrita en el RFC-1034, la cual afirma que si un CNAME es presentado para un nombre, éste es el único tipo permitido para ese nombre. Un RRset KEY con ese nombre en fines de actualización dinámica segura también está permitido, tal y como se describe en el RFC-3007.

Otro cambio en el protocolo resultante de los nuevos registros sería sobre el RR RRSIG, el valor de TTL del cual ha de corresponderse con el TTL del RRset que cubre. Esto es una excepción a la regla descrita en el RFC-2181, para valores de TTL de RRs individuales dentro de un RRset, que dice que “RRs RRSIG individuales con un mismo nombre de propietario tendrán diferentes valores de TTL si los RRsets que cubren tienen diferentes valores TTL”. El RR RRSIG a diferencia de los otros tipos de registros, no forma RRsets.

Como ya sabemos, algunos registros encontraremos que tienen un tiempo de vida (o TTL). Si un servidor esclavo hace una consulta de la caché de nombres para el mismo registro antes de que el TTL expire, el servidor cache simplificará la respuesta con el RR ya en caché, más que recuperarla del servidor autoritativo otra vez. Eso es lo que se entiende por envenenamiento de caché. Los servidores de nombres también deberían tener un TTL para NXDOMAIN; pero son generalmente cortos en durada, tres horas como mucho. TTLs cortos, podría causar cargas más pesadas en servidores autoritativos, pero podrían ser útiles al cambiar la dirección de servicios críticos como servidores web o registros MX, y por tanto son rebajados por el administrador DNS antes de que se traslade un servicio para minimizar las perturbaciones.

También encontramos que dos nuevos tipos de RR, NSEC y DS, cambiarían de lugar. Podrían ser emplazados en el lado padre de un corte de zona, es decir, en el punto de delegación. En donde RRs NSEC podrían ser requeridos en el nombre del propietario. Y también en donde un RR DS podría estar presente si la zona que está siendo delegada está firmada y trata de disponer de una cadena de autenticación a la zona padre. Estos RRsets serían autoritativos para el padre cuando aparezcan en el lado padre de un corte de zona. Todo esto sería una excepción a la prohibición general de inserción de datos en la zona padre de un corte de zona, una excepción a la especificación DNS original del RFC-1034, la cual establece que únicamente los RRsets NS pueden aparecer en el lado padre de un corte de zona.

Recordemos que el RR NSEC es uno de los mayores obstáculos para la difusión global de DNSSEC, ya que algunos expertos opinan que pueden derivarse problemas de protección de datos al poder un atacante simplemente consultando una cadena obtener todos los registros de una zona, lo que se conoce como *zonewalking*. El borrador de NSEC3 ofrece detalles sobre una solución potencial al mismo haciendo uso de técnicas de cifrado. Los escépticos se preguntan si merece la pena proteger nombres de DNS públicos; aunque ven el inconveniente de que personas no autorizadas puedan listar zonas automáticamente, objetan que deben ser otras las medidas a tomar, más en línea con la protección de datos y la confiabilidad. Incluyen ACLs (Access Control Lists) y autenticación de clientes, pero sin extenderse a los registros de DNS de libre acceso. Al final la decisión vendrá dada por las políticas de seguridad de cada compañía. Pero la mayoría de las zonas privadas no se ven afectadas por el problema de la protección de datos NSEC, ya que sólo contienen registros www, mail y otros registros públicos.

También se dice que otro inconveniente a la hora de introducir DNSSEC es que los procesos criptográficos de un servidor de nombres DNSSEC generan el doble de carga en la infraestructura que un servidor normal. Y si los sistemas no están preparados podría darse aumento de tiempo de respuesta, cuellos de botella y la consiguiente denegación de servicio.

Aparte de cambios en el protocolo, también nos aporta conceptos nuevos, como el de isla de seguridad, ya mencionado alguna vez anteriormente en esta memoria. Isla de seguridad sería el término usado para describir una zona delegada y firmada que no tenga una cadena de autenticación desde la delegación padre. Es decir, no hay RR DS que contenga un hash de un RR DNSKEY para la isla en su zona padre de delegación. Una isla de seguridad debe proporcionar cadenas de autenticación a cualquier zona hija delegada. Las respuestas de la isla de seguridad o sus descendiente pueden ser sólo autenticadas si sus claves pueden ser autenticadas por algunos medios de confianza fuera del protocolo DNS. Si se previese que una zona firmada fuera a ser utilizada de forma diferente a una isla de seguridad, la zona debería contener al menos un RR DNSKEY para actuar como SEP dentro de la zona. Este punto podría entonces ser usado como objetivo de una delegación de seguridad mediante un RR DS correspondiente en la zona padre.

3.5. Cliente

Sin duda parte importante de la comunicación entre usuario y entidad financiera. A menudo no se tiene muy en cuenta por parte de empresas o agentes que ofrezcan un servicio, que la comunicación según la esquematización procesal se basa en emisor - canal – receptor, con lo que muchas veces se toman medidas para asegurar el emisor, el canal y no el receptor. Por temas de usabilidad, suele pasarse por alto muchos de los sistemas de seguridad que se podrían implantar en el receptor, delegando al usuario tal responsabilidad. El cliente que se implantará en este proyecto, por tanto, intentará responder a los requeriemientos de usabilidad y de seguridad oportunos. Con el uso de una máquina virtual, se intentará que el usuario navegue de forma segura por la web financiera.

En el estudio previo se habla de máquinas virtuales *full virtualization* con *hypervisor* de tipo I. Actualmente se vislumbran, bien configuradas, posiblemente como una respuesta correcta antirootkits. Pero dichas máquinas virtuales no ofrecen salida a los requerimientos de usabilidad para nuestro caso, puesto su complejidad de instalación, su complejidad de uso y la gran reestructuración de los sistemas del usuario que representaría.

En cuanto a máquinas virtuales *full virtualization* con *hypervisor* de tipo II, ofrecen un impacto menor sobre el usuario y el sistema, pero tal y como vemos en la figura 8, *malware* a muy bajo nivel podría interceptar llamadas a *hardware*.

La virtualización permite a los sistemas operativos invitados y al *software* correr en el *hardware* anfitrión aislados unos de otros. Las máquinas virtuales son perfectamente conscientes de que están compartiendo el *hardware*, y el *hypervisor* es responsable de interceptar las acciones del *software* invitado para mantener esa ilusión. Las acciones que requieren un tratamiento especial por parte del *hypervisor* son pocas y la mayoría de acciones se ejecutan directamente sobre el *hardware*. De esta manera, la virtualización es eficiente, normalmente un 85-95% de la velocidad del sistema nativo, pero también como consecuencia:

- El *hardware* real tiene que tener una arquitectura idéntica a la del sistema invitado. Por ejemplo, VMware, Xen o HyperV pueden crear múltiples PCs virtuales para ejecutar Windows XP "dentro" de un PC real x86. Sin embargo, no se puede en cualquier tipo de *hardware*, por ejemplo, en los procesadores CELL.

- Es vulnerable a fallos de seguridad, ya que la ejecución sucede en la CPU real. *Software* malintencionado podría aprovechar esto para tomar el control de la máquina física real, por ejemplo, los ataques de *Blue Pill* o de SMM (System Management Mode).

Figura 8

En cambio un emulador simula el *hardware* de un ordenador en *software*. Por lo tanto, no hay conexión inherente al *hardware* subyacente, mientras se ejecuta el emulador que hará creer al sistema invitado que se ejecuta en un PC x86. Los emuladores también pueden evitar problemas de seguridad a bajo nivel, ya que se separan del *hardware* subyacente – un sistema invitado malicioso sólo puede corromperse a sí mismo.

Por tanto un emulador, que cumpla un PC virtual, parece una buena solución. Hay un proyecto que me gustaría destacar, el JPC http://www-jpc.physics.ox.ac.uk/ , que se basa en la máquina virtual java, teniendo el siguiente diagrama:

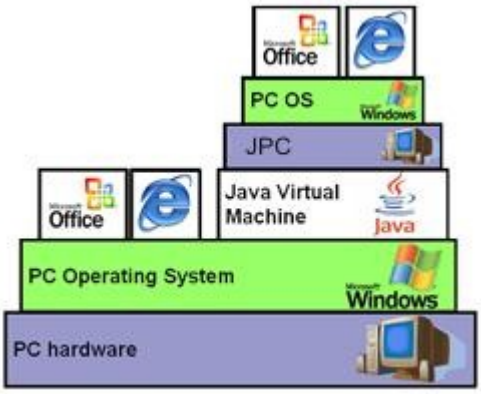


Figura 9

Es una opción interesante, al ser una aplicación basada en Java, es portable a muchos sistemas. Otra ventaja es que ofrece la posibilidad de mostrarse por applets en un navegador web, tal vez pudiendo ser útil para *Cloud Computing*. La versión de la aplicación que permite sistemas operativos *lite o lightweight* con interfaz gráfica, es un poco inestable, pero no está lejos el día en que se pueda instalar por ejemplo, Damn Small Linux (DSL) con múltiples funciones de red.

Pero a día de hoy, existen otras soluciones que se han venido utilizando y mucho más estables, tales como Bochs o Qemu. De modo que se pueda instalar un sistema operativo *lightweight*, como un Linux con el software mínimo. Estaríamos hablando de crear una metadistro. Una metadistro es una distribución hecha a medida por un grupo determinado de usuarios con un objetivo concreto. Aunque se dice que una metadistro no es lo mismo que *live* CD, por el hecho de que la metadistro tiene como objetivo final que sea instalada, ya sea durante el arranque, como desde el escritorio por medio de un instalador. El caso es que también puede funcionar en modo *live* CD. También se le llama metadistros al conjunto de herramientas o a la plataforma para crear o personalizar distribuciones del sistema operativo GNU/Linux a un entorno de trabajo concreto.

De modo que se podría crear una metadistro con un usuario navegador, que las conexiones únicamente apuntasen a la ip fija del servidor DNS de la entidad, un *window manager lightweight* con el que no se pueda acceder al terminal, ni hacer nada mas, solo haya acceso al navegador web con un plugin DNSSEC instalado, y como pagina inicial la de la entidad financiera.

Al únicamente apuntar de forma directa al DNS de la entidad financiera, se aprovecharía mejor la infraestructura DNSSEC de forma externa, puesto que a nivel global no lo está implantada. El plugin del navegador nos avisaría con un simple aviso, como antes, pero ya con la seguridad de que apuntamos al servidor DNS correcto.

De ese modo, se trataría de cubrir los requisitos de seguridad y usabilidad software y que la entidad financiera distribuiría al dar de alta a los usuarios en la entidad. Y se debería procurar también proveer algún sistema de actualización del sistema.

Para los clientes de las unidades móviles cómo *PDAs* o *Smartphones* un problema aparente es la portabilidad. Es por eso que se piensa en una máquina virtual Java (JVM), para incorporar el cliente DNSSEC para Java y un navegador Web Java. Como podría ser JDICPlus Browser o Lobo Browser. Pero surge la problemática de que hoy en día las PDAs o Smartphones hacen uso de máquinas virtuales Java Micro Edition, o anteriormente Java 2 Micro Edition (J2ME) y lo que correspondería sería una máquina virtual que entendiera Java SE, anteriormente llamado J2SE. Pero una vez más, el mercado nos ofrece varias opciones cómo Mysaifu JVM o Eve JVM de Ewesoft.

4. Implementación

Para llevar a cabo la simulación y después grabarla en vídeo, decidí usar máquinas virtuales *full virtualization* de tipo II para la red de la entidad financiera. Haciendo uso de VirtualBox 3.1.2 PUEL (Personal Use and Evaluation License) para implementar las máquinas virtuales y rdesktop para las grabaciones.

Porqué VirtualBox?
Me pareció correcto debido a su portabilidad a otros sistemas y por la gran variedad de sistemas *guest* que acepta.

Porqué PUEL?
VirtualBox PUEL ofrece más prestaciones de configuración que OSE (Open Source Edition), sobretodo si por comodidad en algún momento dado nos interesa utilizar los dispositivos USB, aunque no tendríamos la libre distribución del producto, pero para ello se podría usar la OSE sin problemas.

En versiones anteriores de Virtualbox a la utilizada en la implementación, la configuración de la red en modo *bridge* resultaba un tanto compleja y cambiante según versiones. Dicho modo de configuración, nos permitiría realizar la simulación de la entidad, como si tuvieramos varias máquinas reales dedicadas conectadas.

Aunque la primera opción fué usar los recursos que tenía disponible para llevar a cabo la instalación de la red, haciendo uso de tres máquinas antiguas con las siguientes características:

pc01	pc02	pc03
Intel Celeron 600MHz 128MB RAM 10GB HDD	Intel Pentium III 666.4MHz 256MB RAM 10GB HDD	AMD 500MHz 256MB RAM 8GB HDD

A las que instalaría los siguientes sistemas:

pc01	pc02	pc03
Xubuntu 8.10 rdsktop Virtualbox 3.1.2 Guest OS: Debian GNU/Linux 5.0	Xubuntu 8.10 rdsktop Virtualbox 3.1.2 Guest OS: Debian GNU/Linux 5.0	Ubuntu 8.04 rdsktop Virtualbox 3.1.2 Guest OS: Debian GNU/Linux 5.0

El tiempo de instalación junto con la aparente lentitud en el procesamiento de tareas cuotidianas, a pesar de instalar los sistemas mínimos, me hizo desestimar la idea.

Por lo que finalmente se optó por integrar las tres máquinas virtuales en un solo pc más moderno, con sistema operativo host Debian GNU/Linux 5.0 y sistemas operativos *guest* Debian GNU/Linux 5.0 también, pero en su versión más *lightweight* y sin modo gráfico.

Una vez instalados los sistemas, necesitaríamos instalar y configurar el *firewall+router*, el servidor Web y el servidor DNS.

Las interfícies *ethernet* de cada máquina están configuradas de la siguiente manera:

Máquina PC01 (*firewall+router*, nombre: fwall y usuario: pc01)

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
        address 192.168.10.1
        netmask 255.255.255.0
        broadcast 192.168.10.255
```

Máquina PC02 (servidor DNS, nombre: dnsserv y usuario: pc02)

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.10.2
        netmask 255.255.255.0
        broadcast 192.168.10.255
        gateway 192.168.10.1
```

Máquina PC03 (servidor web, nombre: webserv y usuario: pc03)

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.10.3
        netmask 255.255.255.0
        broadcast 192.168.10.255
        gateway 192.168.10.1
```

4.1. Configuración del firewall+router

El *firewall+router* está implementado a partir de reglas de iptables teniéndose en cosideración varias medidas de seguridad. Para ello se creó un *script* que haría de servicio de *firewall* que se iniciase nada más arrancar el sistema. Esto se hizo de la siguiente manera:

```
$>sudo ln -s /etc/init.d/firewall.sh /etc/rc2.d/S89firewall.sh
```

A la hora de utilizar iptables se ha de asegurar que reglas anteriores no interfieran con las nuevas, para ello si lo que queremos es implantar un *firewall* nuevo tendremos que borrar las reglas previas establecidas, por si las hubiera:

```
# borramos las reglas previas establecidas

$IPTABLES -F            # Borrado de reglas
$IPTABLES -X            # Borrado de reglas creadas por el usuario
$IPTABLES -Z            # Quitamos los contadores de reglas
$IPTABLES -t filter -F  # Borramos las reglas filter
$IPTABLES -t nat -F     # Borramos las reglas nat
$IPTABLES -t mangle -F        # Borramos las reglas mangle, si las hay
```

Después asignar políticas DROP, de este modo si no filtramos algún paquete se verá desechado:

```
# politicas

$IPTABLES -t filter -P INPUT DROP
$IPTABLES -t filter -P OUTPUT DROP
$IPTABLES -t filter -P FORWARD DROP
```

Lo siguiente será permitir ciertas llamadas icmp que podrían ser útiles:

```
# permitimos icmp

$IPTABLES -t filter -A INPUT -i $INET_IFACE -p icmp --icmp-type 0 -j ACCEPT
$IPTABLES -t filter -A INPUT -i $INET_IFACE -p icmp --icmp-type 3 -j ACCEPT
$IPTABLES -t filter -A INPUT -i $INET_IFACE -p icmp --icmp-type 5 -j ACCEPT
$IPTABLES -t filter -A INPUT -i $INET_IFACE -p icmp --icmp-type 11 -j ACCEPT
$IPTABLES -t filter -A OUTPUT -o $INET_IFACE -p icmp -j ACCEPT
```

Permitimos las resoluciones DNS, con tal de poder solucionar direcciones @ip:

```
# permitimos dns para INET_IFACE

$IPTABLES -t filter -A INPUT -i $INET_IFACE -s $ISP_DNS1 -p udp --sport 53 -j ACCEPT
$IPTABLES -t filter -A OUTPUT -o $INET_IFACE -d $ISP_DNS1 -p udp --dport 53 -j ACCEPT
```

Permitimos las conexiones dchp con tal de conectarnos a nuestro ISP:

```
# permitimos dchp para INET_IFACE

$IPTABLES -t filter -A INPUT -i $INET_IFACE -p udp --sport 67 --dport 68 -j ACCEPT
$IPTABLES -t filter -A OUTPUT -o $INET_IFACE -p udp --dport 67 -j ACCEPT
```

Activamos el *bit* de *forward* del sistema:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Hacemos el *forwarding* que queramos:

```
# forwarding servidor web y cualquier otro servicio establecido

$IPTABLES -t nat -A PREROUTING -i $INET_IFACE -s ! $LAN_IP_RANGE -d $INET_IP -p tcp --dport 443 -j DNAT --to-destination $SWEB:443
$IPTABLES -t nat -A PREROUTING -i $INET_IFACE -s ! $LAN_IP_RANGE -d $INET_IP -p udp --dport 53 -j DNAT --to-destination $SDNS:53
```

llega

$IPTABLES -t filter -A FORWARD -i $INET_IFACE -s ! $LAN_IP_RANGE -d $LAN_IP_RANGE -o $LAN_IFACE -j ACCEPT

y se va

$IPTABLES -t filter -A FORWARD -i $LAN_IFACE -s $LAN_IP_RANGE -o $INET_IFACE -j ACCEPT

$IPTABLES -t nat -A POSTROUTING -s $LAN_IP_RANGE -o $INET_IFACE -j MASQUERADE

Permitimos conexiones LAN:

aceptamos cualquier conexion de LAN en LAN_IFACE

$IPTABLES -t filter -A INPUT -i $LAN_IFACE -d $LAN_IP_RANGE -j ACCEPT
$IPTABLES -t filter -A OUTPUT -o $LAN_IFACE -d $LAN_IP_RANGE -j ACCEPT

Se podría hacer un *firewall stateful* para más seguridad. Un *firewall* con *stateful packet inspection* (SPI) o *stateful inspection* es un *firewall* que sigue la pista de los estados de las comunicaciones que lo atraviesan, como trazas TCP o comunicaciones UDP. El *firewall* está programado para distinguir paquetes legítimos para diferentes tipos de conexiones. Únicamente los paquetes que concuerden con un estado de una conexión conocida por el *firewall* serán permitidos; otros serán rechazados.

Como ejemplo:

$IPTABLES -t filter -A FORWARD -i $INET_IFACE -s ! $LAN_IP_RANGE -m state --state NEW,RELATED,ESTABLISHED -o $LAN_IFACE -j ACCEPT

$IPTABLES -t filter -A FORWARD -i $LAN_IFACE -s $LAN_IP_RANGE -m state --state NEW,RELATED,ESTABLISHED -o $INET_IFACE -j ACCEPT

Si quisiéramos listar las reglas para comprobar que funcionan:

$>sudo iptables -L -v

También, instalé para monitorizar la red la herramienta iptraf de la siguiente manera:

$>sudo apt-get install iptraf

La herramienta conviene desinstalarla al finalizar las pruebas, puesto que esta herramienta situa la interfície de red en modo promiscuo. Y podría ser usada por alguien no legítimo en nuestra contra. Se procedería a su desinstalación de la siguiente manera:

$>sudo apt-get purge iptraf

4.2. Instalación y configuración del servidor Web

El servidor Web a instalar es un Apache2. Aunque la instalación de Debian GNU/Linux 5.0 ya ofrece la posibilidad de instalar directamente un Apache2, opción escogida por el autor de este proyecto, si se quisiera instalar manualmente sería algo tan fácil como lo siguiente:

$>sudo apt-get install apache2

Después podríamos habilitar el módulo de SSL de la siguiente manera:

$>sudo ln -s /etc/apache2/mods-available/ssl.conf /etc/apache2/mods-enabled/
ssl.conf
$>sudo ln -s /etc/apache2/mods-available/ssl.load /etc/apache2/mods-enabled/
ssl.load

$>sudo ln -s /etc/apache2/sites-available/default-ssl /etc/apache2/sites-enabled/
001-default
$>sudo unlink /etc/apache2/sites-enabled/000-default

Y después agregar certificados autofirmados. Para crearlos se debe installar el *software* openssl, que en este caso ya venía instalado, pero si se tuviera que hacer sería:

$>sudo apt-get install openssl

Después en nuestro *home* mismo, inicializamos números de serie de certificados:
echo '01' > serial
touch ssl.srl

Creamos las carpetas donde irán los certificados:
$>sudo mkdir ssl.key
$>sudo mkdir ssl.csr
$>sudo mkdir ssl.crt

Y ya podremos empezar a crear los certificados. Lo primero que crearemos en esta simulación, será un certificado de Entidad Certificadora Raíz y después un certificado de servidor que dependa de la Entidad Certificadora Raíz.

CERTIFICADO RAÍZ, VeriS1gn Trust Network
(VeriS1gn, Inc.)

Generamos la petición de certificado:
$>sudo openssl req -new -extensions v3_ca -keyout ssl.key/ca_key.pem -out ssl.csr/ca_cert-req.pem

Enter PEM pass phrase: qapoll2vpjpz8wjg
Verifying – Enter PEM pass phrase: qapoll2vpjpz8wjg
Country Name [AU]: US
State or Province Name (full Name) [Some-State]: United States
Locality Name (eg, city) []: Whasington D.C.
Organization Name (eg, company) [Internet Widgits Pty Ltd]: "VeriS1gn, Inc."
Organizational Unit Name (eg, section) []: Class 3 Public Primary Certification Authority
Common Name (eg, YOUR name) []: "VeriS1gn, Inc."
Email Address []: questions@veris1gn.com

A challenge password []: yyip3ilx
An optional company name []: .

Generamos el certificado de la EC a partir de la petición pendiente:
$>sudo openssl req -new -x509 -in ssl.csr/ca_cert-req.pem -out ssl.crt/ca_cert.crt -days 1460 -key ssl.key/ca_key.pem

Enter pass phrase for ssl.key/ca_key.pem: qapoll2vpjpz8wjg
Country Name (2 letter code) [AU]: US
State or Province Name (full Name) [Some-State]: United States
Locality Name (eg, city) []: Whasington D.C.
Organization Name (eg, company) [Internet Widgits Pty Ltd]: www.veris1gn.com/CPS Incorp.by Ref. LIABILITY LTD.(c)97 VeriS1gn
Organizational Unit Name (eg, section) []: VeriS1gn International Server CA - Class 3
Common Name (eg, YOUR name) []: "VeriS1gn, Inc." VeriS1gn Trust Network
Email Address []: questions@veris1gn.com

CERTIFICADO CDS dependiente de (VeriS1gn, Inc.)

Generamos la petición del certificado:

$>sudo openssl req -new -keyout ssl.key/cds_key.pem -out ssl.csr/cds_cert-req.pem

Enter PEM pass phrase: dbofsh3izmxa9 eu
Verifying – Enter PEM pass phrase: dbofsh3izmxa9 eu
Country Name (2 letter code) [AU]: SP
State or Province Name (full name) [Some-State]: Spain
Locality Name (eg, city) []: Barcelona
Organization Name (eg, company) [Internet Widgits Pty Ltd]: Ca1xa d.Estalvis i Pensions de Barcelona
Organizational Unit Name (eg, section) []: laca1xa
Common Name (eg, YOUR name) []: laca1xa
Email Address []: contacto@laca1xa

A challenge password []: 5j5ul5ge
An optional company name []: .

Generamos el certificado y lo firmamos con la EC raíz:

$>sudo openssl x509 -req -in ssl.csr/cds_cert-req.pem -out ssl.crt/cds_cert.crt -days 1460 -CA ssl.crt/ca_cert.crt -CAkey ssl.key/ca_key.pem -CAcreateserial
Signature ok
subject=/C=SP/ST=Spain/L=Barcelona/O=Ca1xa d.Estalvis i Pensions de Barcelona/OU=laca1xa/CN=laca1xa/emailAddress=contacto@laca1xa
Getting CA Private Key
Enter pass phrase for ssl.key/ca_key.pem: qapoll2vpjpz8wjg

Entonces configuramos Apache2 de la siguiente manera:

cd /etc/apache2/sites-available
$>sudo vi default-ssl

Comentamos las lineas SSLCertificateFile y SSLCertificateKeyFile. Y añadimos:

SSLCertificateFile /home/pc03/ssl.crt/cds_cert.crt
SSLCertificateKeyFile /home/pc03/ssl.key/cds_key.pem

Como ya vienen comentadas SSLCACertificatePATH y SSLCACertificateFile, solo tendremos que añadir:

SSLCACertificatePATH /home/pc03/ssl.crt/
SSLCACertificateFile /home/pc03/ssl.crt/ca_cert.crt

Lo que ocurre es que al iniciar Apache2 nos pide el PassPhrase del certificado, si queremos evitar esto deberemos hacer lo siguiente:

cd /home/pc03/ssl.key/
$>sudo cp cds_key.pem cds_key.key
$>sudo openssl rsa -in cds_key.key -out cds_new_key.key

Y entonces substituir en el archivo de configuración la linea:

SSLCertificateKeyFile /home/pc03/ssl.key/cds_key.pem

por:

SSLCertificateKeyFile /home/pc03/ssl.key/cds_new_key.key

Y crear una web en el directorio que se especifica en la configuración del servidor en el archivo httpd.conf. En los archivos de configuración del servidor se indican diversas opciones a tener en cuenta de cara a la seguridad. Pero no es materia de este proyecto entrar en todos los entresijos de seguridad en servidores web Apache2.

También, instalé para monitorizar las conexiones hacia el servidor la herramienta iptraf. A recordar que la herramienta conviene desinstalarla al finalizar las pruebas.

4.3. Instalación y configuración del servidor DNS seguro

El servidor DNS a instalar es un BIND 9, en este caso aunque la instalación de Debian GNU/Linux 5.0 también ofrece la posibilidad de instalar directamente un DNS, escogí instalarlo manualmente, sin ningún motivo en especial, de la forma siguiente:

$>sudo apt-get install bind9

Con tal de hacer pruebas localmente instalé un cliente de DNS, el *host*:

$>sudo apt-get install host

También instalé para monitorizar las conexiones hacia el servidor la herramienta iptraf. Recordar que la herramienta conviene desinstalarla al finalizar las pruebas.

Bind9 por defecto ya viene con soporte para Security Extensions. Solo habría que escribir en el archivo de configuración las opciones:

```
options {
        ...
        dnssec-enable yes;
        dnssec-validation yes;
};
```

Substituir los archivos de zonas por los archivos firmados y crear el SEP.

Pero ésto no es bien bien así. Al estar trabajando con un servidor DNS autoritativo, bind9 no puede validarse así mismo y hacerle saber al cliente el resultado. Siempre dará la respuesta de que es un servidor autoritativo tal y como vemos con la herramienta dig que hemos instalado en una máquina externa auxiliar no presente en este proyecto:

```
$> dig +dnssec @81.184.43.199 laca1xa.

; <<>> DiG 9.4.2-P2 <<>> +dnssec @81.184.43.199 laca1xa.
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8206
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;laca1xa.                        IN      A

;; ANSWER SECTION:
laca1xa.                604800IN        A       81.184.43.199
laca1xa.                604800IN        RRSIG A 5 1 604800 20100224121957
20100125121957 58710 laca1xa.
fCK/Yd1LIyU8Asn9sFPp2hXGcQEYqiqaEkze2SnpguFlXzgEm0/Lfhdv
Ud8y+jwleG+oQ/c4iMP+DCaqAR2paA==

;; AUTHORITY SECTION:
laca1xa.                604800IN        NS      laca1xa.
laca1xa.                604800IN        RRSIG NS 5 1 604800 20100224121957
20100125121957 58710 laca1xa.
Kv/2BfJfgblj4456hdIH+6x/57EUpwkkgsyAakx3wAHkbzGaZVdM/bIV
Ptkt5eluB/qVpTEdENA0ZP+yfqiMsg==

;; Query time: 8 msec
;; SERVER: 81.184.43.199#53(81.184.43.199)
;; WHEN: Mon Jan 25 21:08:23 2010
;; MSG SIZE  rcvd: 272
```

Se puede apreciar en los *flags* la etiqueta **aa** que nos indica que se trata de un servidor autoritativo. De este modo un cliente, como un *plugin* de firefox, no sabría si estamos ante un servidor en el que se haya validado dnssec. En cuyo caso los *flags* nos mostrarían la etiqueta **ad**.

Todo esto sugiere un replanteamiento del diseño de la red de la siguiente manera:

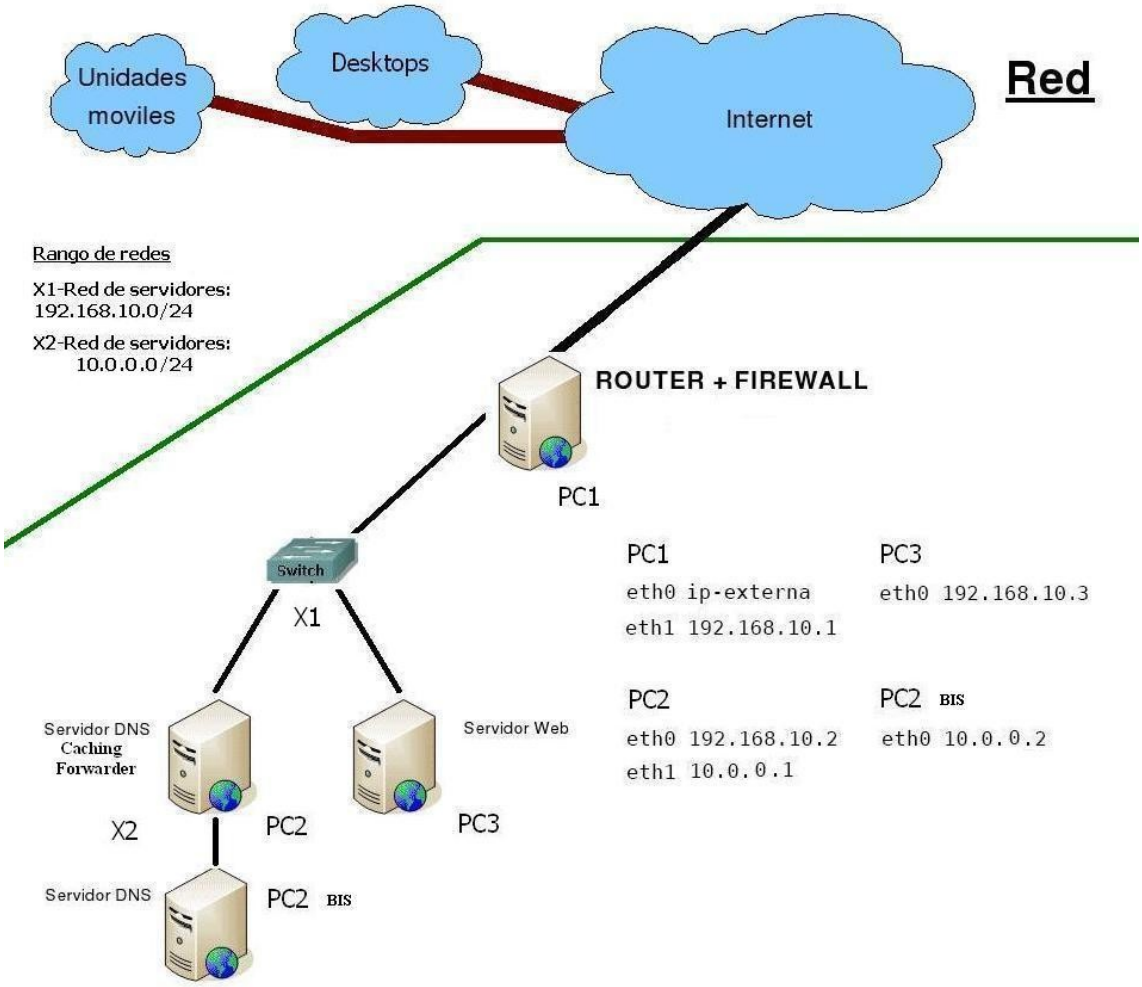


Figura 10

Como se puede ver en el esquema tenemos las mismas máquinas, pero se ha añadido una que hace de *caching forwarder validator*, de modo que ahora es posible obtener una buena respuesta por parte del servidor. Para ello se copió una imagen de disco de VirtualBox para crear otra máquina idéntica, pero se produjo el siguiente error:

Código Resultado
NS_ERROR_INVALID_ARG (0x80070057)
Componente:
VirtualBox
Interface:
IVirtualBox {339abca2-f47a-4302-87f5-7bc324e6bbde}

errorvb

Esto es debido a que las imágenes de disco en VirtualBox llevan un UUID que las identifica, de modo que al añadir otra, el gestor comprueba que no está repetida para evitar que una máquina virtual arranque unas veces con uno y otras veces con otro disco. Por ello las imágenes de VirtualBox se deben copiar con un comando especial que les genere un nuevo UUID. El comando sería el siguiente:

$>sudo VBoxManage clonehd "/home/miusuario/.VirtualBox/HardDisks/dnsserv.vdi" "/home/miusuario/.VirtualBox/HardDisks/dnsfwd.vdi"

$>cd /home/miusuario/.VirtualBox/Machines
$>sudo cp -R dnsserv dnsfwd

$>sudo chown -R miusuario:miusuario dnsfwd

De este modo clonamos la máquina PC2 original con tal de ahorrar tiempo, en vez de crear una desde cero. Una vez hecho esto, encontramos que no van del todo bien las interfaces de red, puesto que Debian guarda la MAC asignada por VirtualBox. Lo que hay que hacer es en el archivo /etc/udev/rules.d/70-persistent-net.rules substituir la MAC, en este caso:
08:00:27:81:4b:56
por:
08:00:27:81:3C:EA

También mirar que en el archivo /etc/udev/persistent-net-generator.rules haya una regla para ignorar generar reglas persistentes para interfaces Vmware y añadir después de esas reglas la siguiente:

ignore VirtualBox virtual interfaces
ATTR{address}=="08:00:27:*", GOTO="persistent_net_generator_end"

Finalmente borrar el archivo /etc/udev/z25_persistent-net.rules. No cambiar el archivo /etc/network/interfaces. E importante también es no olvidarse reiniciar el sistema.

Una vez ya funciona la copia, cambiamos de nombre la máquina que pasará a llamarse dnsfwd, y si se considera se podrá cambiar usuario y contraseña y alguna opción más que sea interesante.

Desinstalamos bind9:

$>sudo apt-get purge bind9

Para después intalar unbound que hará de servidor DNS *caching forwarder validator* porque un servidor bind9 no puede hacer de *caching forwarder validator*, o hace de *caching* o hace de *forwarder*.

Instalamos unbound:

$>sudo apt-get install unbound

Después de la firma de zonas para la implementación de dnssec procederemos a la configuración de unbound.

A la hora de la implementación de dnssec, hay que tener en cuenta que hay dos servidores involucrados. El bind9 del PC2 (BIS) y el unbound del PC2. Pero el primer paso será la generación de claves en el PC2 (BIS) del bind9. Con bind9 ya vienen todas las herramientas de criptografía que nos harán falta para la implementación de dnssec.

A la hora de generar claves y firmar las zonas nos surge un inconveniente, falta de entropía. La entropía, para que nos entendamos, es lo que da aleatoriedad a aquellos recursos que así lo requieren en un sistema. En este caso el proceso de firma criptográfica requiere de esta aleatoriedad. En el caso de Debian, la entropía se genera a raíz de interrupciones de ratón, teclado, dma, inputs - outputs. El problema por tanto, si estuvieramos en una máquina con X Window System (sistema de ventanas X) que dotase de interfaz gráfica sería fácilmente resuelto. Pero este no es el caso y solo disponemos de línea de comandos, por lo que deberemos generar por medio de la línea de comandos lo que haga falta. Así que primero en la carpeta de las zonas, generaremos la clave de zona de la siguiente manera:

$>sudo dnssec-keygen -a RSASHA1 -b 512 -n ZONE laca1xa. &

Ejecutando el comando con el & a final de la línea, lo que haremos será que pase a segundo plano y así podremos tener disponible la línea de comandos y de ese modo poder operar en caso de que se cuelgue el proceso por falta de entropía.

Para consultar el estado de la entropía, el siguiente comando:

$>cat /proc/sys/kernel/random/entropy_avail

Al arrancar la máquina, la entropía suele estar sobre unos 3000 puntos. Si vamos ejecutando el cat mientras se ejecuta en segundo plano la firma, y vemos que está entre 0 y 100 puntos, significará que falta entropía. Al ejecutar ese comando, puesto que tecleamos con el teclado de la máquina, ya estamos generando un poco de entropía. También una buena manera es hacer que la máquina trabaje un poco como con el comando:

$>sudo find -iname "*"

Y que busque ficheros en el disco duro.

Después, lo segundo que haremos será crear la KSK para el SEP:

$>sudo dnssec-keygen -f KSK -a RSASHA1 -b 512 -n ZONE laca1xa. &

Y en cuanto a la entropía operaremos de la misma manera que antes.

Una vez esto, lo tercero será cargar las claves en la zonas, así que con los privilegios necesarios, añadiremos las claves al final de los ficheros de zona de la siguiente manera:

```
$>echo "" >> db.43.184.81
$>echo "\$include Klaca1xa.+005+58710.key ; ZSK" >> db.43.184.81
$>echo "" >> db.43.184.81

$>echo "" >> db.laca1xa
$>echo "\$include Klaca1xa.+005+58710.key ; ZSK" >> db.laca1xa
$>echo "" >> db.laca1xa
```

y la KSK también:

```
$>echo "\$include Klaca1xa.+005+53773.key ; KSK" >> db.43.184.81
$>echo "" >> db.43.184.81

$>echo "\$include Klaca1xa.+005+53773.key ; KSK" >> db.laca1xa
$>echo "" >> db.laca1xa
```

Una vez echo esto firmamos las zonas de la siguiente manera:

$>sudo dnssec-signzone -o laca1xa -k Klaca1xa.+005+53773 db.43.184.81 Klaca1xa.+005+58710.key

$>sudo dnssec-signzone -o laca1xa -k Klaca1xa.+005+53773 db.laca1xa Klaca1xa.+005+58710.key

Y en cuanto a la entropía operaremos de la misma manera que antes.

Una vez tenemos las zonas firmadas, substituiremos en named.conf.local, las zonas por las zonas ".signed" generadas.

Añadir en named.conf.options la opción:

```
options {
        ...
        dnssec-enable yes;
};
```

Y no poner la opción:

```
options {
        ...
        dnssec-validation yes;
};
```

Puesto que el que hará de *validator* será el unbound.

Ahora que ya tenemos las zonas firmadas y el bind configurado, procederemos a la configuración de unbound. La configuración del unbound está en el archivo unbound.conf. Pero antes deberemos copiar del PC2 (BIS) del bind9 una serie de claves que necesitará unbound para validar.

Del PC2 (BIS) copiamos la primera clave del archivo dsset-laca1xa. y la pegamos en el archivo /etc/unbound/anchors/laca1xa.anchor del PC2.

Del PC2 (BIS) copiamos la segunda clave del archivo dsset-laca1xa. y la pegamos en el archivo /etc/unbound/anchors/laca1xa2.anchor del PC2.

Y del PC2 (BIS) copiamos la clave keyset-laca1xa. en el archivo /etc/unbound/anchors/trusted-keys del PC2.

Y ahora sí se puede configurar unbound, en el archivo /etc/unbound/unbound.conf, tendremos que configurarlo de la siguiente manera, entre otras opciones:

```
# specify the interfaces to answer queries from by ip-address.
# The default is to listen to localhost (127.0.0.1 and ::1).
# specify 0.0.0.0 and ::0 to bind to all available interfaces.
# specify every interface on a new 'interface:' labelled line.
# The listen interfaces are not changed on reload, only on restart.
# interface: 192.0.2.153
# interface: 192.0.2.154
# interface: 2001:DB8::5
interface: 192.168.10.2
```

```
        # specify the interfaces to send outgoing queries to authoritative
        # server from by ip-address. If none, the default (all) interface
        # is used. Specify every interface on a 'outgoing-interface:' line.
        # outgoing-interface: 192.0.2.153
        # outgoing-interface: 2001:DB8::5
        # outgoing-interface: 2001:DB8::6
        outgoing-interface: 10.0.0.1

        # File with trusted keys for validation. Specify more than one file
        # with several entries, one file per entry.
        # Zone file format, with DS and DNSKEY entries.
         trust-anchor-file: "/etc/unbound/anchors/laca1xa.anchor"
         trust-anchor-file: "/etc/unbound/anchors/laca1xa_2.anchor"

        # File with trusted keys for validation. Specify more than one file
        # with several entries, one file per entry. Like trust-anchor-file
        # but has a different file format. Format is BIND-9 style format,
        # the trusted-keys { name flag proto algo "key"; }; clauses are read.
         trusted-keys-file: "/etc/unbound/anchors/trusted-keys"

# Forward zones
# Create entries like below, to make all queries for 'example.com' and
# 'example.org' go to the given list of servers. These servers have to handle
# recursion to other nameservers. List zero or more nameservers by hostname
# or by ipaddress. Use an entry with name "." to forward all queries.
# forward-zone:
#         name: "example.com"
#         forward-addr: 192.0.2.68
#         forward-addr: 192.0.2.73@5355  # forward to port 5355.
# forward-zone:
#         name: "example.org"
#         forward-host: fwd.example.com
forward-zone:
        name: "laca1xa."
        forward-addr: 10.0.0.2
```

4.4. Cliente

Para el cliente montamos el emulador o máquina virtual Qemu en un *pendrive* USB de manera que podríamos instalarlo para diferentes plataformas, gracias a su gran portabilidad. Recordar además que Qemu está bajo licencia GNU GPL, pero que diversas partes del *software* tienen diferentes licencias.

Se puede descargar desde la web oficial:
http://www.qemu.org/download.html

Y no instalaremos Kqemu, el acelerador de Qemu, ni ejecutaremos el modo de usuario y virtual en modo de código 8086 directamente sobre la CPU del computador, porque lo que haría sería convertirse en una máquina *full virtualization* con *hypervisor* de tipo II y ya en el diseño se ha explicado porque no queríamos eso.

Crearemos entonces un archivo lanzador que llame al programa con las opciones o parámetros que nos interesen. Este lanzador lo que hará será llamar a Qemu para que ejecute una imagen ISO de una metadistro que creemos.

Como opciones para crear una metadistro teníamos varias distribuciones en las que basarnos, las principales eran DSL (basada en Debian), Slax (basada en Slackware) y Debian.

A pesar de que DSL era la más *lightweight*, DSL junto con Debian requerían más herramientas y más complejidad para crear una metadistro a partir de ellas. Por eso hice uso de Slax. Aún así, crear una metadistro resulta un trabajo laborioso y largo, y en cuanto surge una modificación, se ha de recompilar todo.

Para crear mi metadistro de Slax, descargué desde la página oficial una imagen ISO de Slax solo con el módulo de *core*. Slax es una distribución que se puede crear totalmente a partir de módulos. Una vez con la ISO de Slax con el módulo de *core,* con el programa MySlax Creator, que descargué desde su web oficial http://sites.google.com/site/myslaxcreator/download añadí a la ISO un módulo del navegador Firefox ya preparado, un módulo de Xorg también preparado e intercambié el *core* por uno preparado.

Para crear o modificar módulos, hice uso de las herramientas proporcionadas por el paquete linux-live que se puede descargar en la página web oficial:
http://www.linux-live.org/

Para personalizar un módulo Firefox, lo que hice fué por un lado descargar una imagen ISO de Slax con el *core*, el entorno gráfico y el Firefox. Ejecuté la imagen e hice los cambios pertinentes en el Firefox, como instalar el *add-on* para DNSSEC drill, que se puede descargar en la siguiente página web oficial:
http://nlnetlabs.nl/projects/drill/drill_extension.html

La extensión o *add-on* drill requiere de una librería llamada ldns, la cual se puede descargar también de:
http://nlnetlabs.nl/projects/ldns/

También aproveché esta librería para incluir en ella las claves para que el cliente valide las conexiónes al servidor DNS que a su vez, éste se validará a sí mismo. Entonces creé un fichero llamado /milibdns/usr/local/laca1xa.key y copié del servidor DNS (PC2 (BIS)) la clave keyset-laca1xa. o lo que es lo mismo, del archivo de zona firmada db.laca1xa.signed el DNSKEY 257. Y después copié del archivo de zona firmada db.laca1xa.signed el DNSKEY 256.

Por último configuré la página de inicio del navegador Firefox como la página de la entidad bancaria en cuestión.

Una vez tenía ese Firefox personalizado, necesitaba saber que carpetas y archivos habían cambiado en referente a los archivos originales del módulo. Es por eso que aparte decompilé el módulo original con las herramientas proporcionadas por el paquete linux-live, comparé que ficheros y carpetas habían cambiado con el Firefox personalizado. Y cuando los averigüé, creé un nuevo módulo a partir de los archivos base y los modificados y nuevos.

Lo mismo hice con el módulo Xorg. El módulo Xorg trae por defecto un gestor de ventanas, el Fluxbox, el cual es un gestor *lightweight* bastante configurable y que serviría perfectamente. Así que se pasó a configurar los menús del gestor para que no se puediera acceder al terminal, ni hacer nada más, solo acceder al navegador web.

En cuanto al módulo de *core*, solo añadí un script o servicio con tal de que cada vez que se iniciase el sistema y / o resolviese por dchp la ip que se nos asignase, no se sobreescribiese también la dirección DNS indicada en el archivo /etc/resolv.conf de la máquina en cada negociación dchp. Puesto que nos interesa tener la dirección DNS del servidor de la entidad financiera.

5. Conclusión

El trabajo nos ha aportado una visión de futuro para la seguridad en los canales de comunicaciones entre una entidad financiera y su cliente, de modo que damos un paso adelante en búsqueda de sistemas y herramientas antifraude, sobretodo en cuanto a protección en el lado servidor.

Pero se podrían ampliar los sistemas de seguridad en el lado servidor y modificar la estructura de red, para ampliar la seguridad con sistemas no vistos por ahora en este trabajo, y también en el lado cliente. La solución final correspondería a la de un escenario que tuviera en cuenta tanto la seguridad a nivel de datos, como de aplicación, red, los sistemas y canales en general, aparte de la seguridad entorno al factor humano.

La solución que se plantea es la siguiente:

- A nivel de sistema, optar por un sistema de *full virtualization* con *hypervisor*, corriendo sobre el *hardware* algún sistema *lightweight*, parece correcto, si se tiene en cuenta que el sistema que corra sea lo más seguro posible. El sistema *lightweight* a su vez podría ofrecer virtualización a nivel de sistema operativo (jaulas). En este caso sólo habría un sistema operativo en el que se virtualizasen aplicaciones, generalmente servidores, de manera que las aplicaciones que corrieran en el entorno *guest* lo verían como un sistema autónomo.

- A nivel de aplicación, como serían máquinas dedicadas, se trataría de que corrieran pocos servicios en cada una y con labores específicas, así encontraríamos por ejemplo, un servidor DNS con extensiones de seguridad, un servidor proxy, algún servidor proxy invertido web con *firewall* analizador http para la prevención y detección de ciertas intrusiones, un servidor web y cualquier servicio necesario. De manera que al ser pocos servicios y distribuidos en varias máquinas, se podrían controlar mejor. Si un servicio se viera comprometido, el estar en máquinas diferentes favorecería a que los demás no tuvieran por qué verse en esa situación. En el lado cliente, se podría implantar una máquina virtual portal proxy, que virtualizase un navegador y diversos módulos de seguridad, que estuvieran conectados a la red bancaria. Y también vendría bien un sistema de actualizaciones del software.

- A nivel de transporte, que las comunicaciones estuvieran encriptadas con sistemas criptográficos de dos vías o reversibles. Empleando algoritmos de cifrado simétrico, que se intercambien usando algoritmos de clave pública.

- A nivel red, un buen filtrado de paquetes y herramientas de monitorización que permitieran la evaluación en tiempo real del comportamiento de un acceso mediante cualquier canal, como son las herramientas de prevención y detección de intrusos. O también herramientas que se basen en la historia del comportamiento anterior de diferentes usuarios en diferentes canales. Si es necesario llevando a cabo un sistema de privilegios basado en listas de control de acceso o ACLs.

- A nivel humano, decidimos confiar bastante en los sistemas y en la gente que hay detrás, aunque para éso se haya de tomar medidas, tanto nosotros, como la entidad y el canal.
Como charlas para empleados, charlas para clientes, difusión audiovisual, cursos de administración básica y cursos avanzados.

6. Futuro del proyecto

Este es un proyecto el cual ofrece muchas posibilidades de ampliación, mejora y evolución a nuevas tecnologías. Se podría ampliar para hacer frente, prevenir, detectar y corregir los posibles ataques que puede estar sometida la banca electrónica del futuro. Sería bueno, que dichas posibilidades permitieran la evaluación en tiempo real del comportamiento de un acceso mediante cualquier canal y la detección del fraude en tiempo real, basado en la historia del comportamiento anterior de diferentes usuarios accediendo a diferentes canales.

Sobre casos de estudio, en cuanto a la seguridad para dispositivos móviles, encontramos temas relacionados con los identificadores IMSI, IMEI, BT id. Identificadores únicos que nos permitirían autenticar al usuario con el dispositivo móvil. La idea sería utilizar estos identificadores como refuerzo a la autenticidad del cliente. Refuerzo, puesto que hoy en día con el continuo cambio de propietario y adquisición de nuevos móviles, sustracción, etc, sería difícil asemejar el identificador a una contraseña con la que autenticarse. No sólo darse de alta y baja de la entidad financiera de cara al usuario, si no que hoy en día las personas cambian antes de móvil que de contraseña para sus navegaciones. Y también está el tema de si sería posible la captura de estos identificadores. Se podría diseñar un software específico para los móviles de los clientes de la entidad, que enviásen este identificador de manera unequívoca y antifraude. En algunos países son las operadoras telefónicas las que adjuntan estos identificadores a las peticiones de los clientes.

En un futuro, también se pretende que con la llegada de la TDT y el estándard MPH, muchas de estas estrategias de seguridad puedan aplicarse a este nuevo canal. El principal problema del canal de TDT nace en la verificación del usuario y en la seguridad del canal de interacción.

La TDT al ser un dispositivo de consumo doméstico multipersonal, tiene la dificultad añadida respecto otros canales, que la verificación del usuario, así como la posibilidad de suplantación de personalidad, es más difícil de controlar. Además los dispositivos terminales TDT no disponen normalmente de periféricos específicos que faciliten la tarea y si tienen alguno, éste es muy simple. Todo hace que sea más difícil la puesta en marcha de mecanismos antifraude bancario, que hará falta resolver.

Se tendrá que tener en cuenta la propia seguridad del canal de comunicación interactivo de la TDT, que como cualquier otro canal, es susceptible a ser manipulado para usos fraudulentos.

También se podrían tratar temas como la integración de tecnologías biométricas, tanto en estaciones fijas como en dispositivos móviles, para las aplicaciones financieras y su aceptación en la sociedad con tal de establecer la máxima seguridad para la aplicación. Estudiando la utilización de biometria multimodal (combinación de varias biometrias) como solución a la prevención de la suplantación de identidad de usuarios. Y sistemas de alerta frente a posibles situaciones de peligro. Así como la problemática de la autenticación en tiempos razonables con tasas de fiabilidad aceptables. En este sentido la utilización de biometria multimodal, puede hacer que mejore el tiempo de reconocimiento cuando se trate de grandes bases de datos.

7. Referencias bibliográficas

Fuentes primarias
- RFC-1034: Domain Names - Concepts and facilities

- RFC-1035: Domain Names - Implementation and specification

- RFC-2181: Clarifications to the DNS Specification

- RFC-4033: DNS Security Introduction and Requirements

- RFC-4034: Resource Records for the DNS Security Extensions

- RFC-4035: Protocol Modifications for the DNS Security Extensions

- RFC-4641: DNSSEC Operational Practices

- RFC-5155: DNS Security (DNSSEC) Hashed Authenticated Denial of Existence

- Wikipedia: www.wikipedia.org

Fuentes secundarias
- RFC-2308: Negative Caching of DNS Queries (DNS NCACHE)

- RFC-2845: Secret Key Transaction Authentication for DNS (TSIG)

- RFC-3007: Secure Domain Name System (DNS) Dynamic Update

- RFC-3110: RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS)

- RFC-3757: Domain Name System KEY (DNSKEY) Resource Record (RR) Secure Entry Point (SEP) Flag

- RFC-3845: DNS Security (DNSSEC) NextSECure (NSEC) RDATA Format

- RFC-4398: Storing Certificates in the Domain Name System (DNS)

- RFC-5155: DNS Security (DNSSEC) Hashed Authenticated Denial of Existence

- CERT de Centro Criptológico Nacional de Centro Nacional de Inteligencia: www.ccn-cert.cni.es

8. Anexo

Tablas comparativas DNSSEC y DNSCurve:

Security

Type of security	DNSSEC	DNSCurve
Confidentiality against network sniffers	RFC 4033: "Due to a deliberate design choice, DNSSEC does not provide confidentiality."	→ DNSCurve encrypts requests and responses, adding some confidentiality.
Confidentiality against active attackers	DNSSEC reduces existing confidentiality by publishing the complete list of "secured" DNS records. This publication is integrated into the DNSSEC protocol; it is independent of classic "zone transfers" and cannot be disabled by administrators. The "NSEC3" variant of DNSSEC attempts to reduce this exposure but is almost always breakable.	→ DNSCurve does not publish lists of DNS records.
Integrity despite forged network packets	DNSSEC uses public-key cryptography, normally 1024-bit RSA, to detect forgeries. 1024-bit RSA is already breakable by large companies and botnets but has not yet been broken by academic teams.	→ DNSCurve uses public-key cryptography, specifically 255-bit elliptic-curve cryptography, to detect forgeries. 255-bit elliptic-curve cryptography is billions of times harder to break than 1024-bit RSA by current cryptanalytic algorithms.
Integrity despite corruption of parent computers	DNSSEC does not protect nytimes.com or google.com against an attacker controlling the .com computers.	DNSCurve does not protect nytimes.com or google.com against an attacker controlling the .com computers.
Integrity despite corruption of one's own computers	→ DNSSEC *does* protect nytimes.com DNS data against an attacker controlling the nytimes.com DNS servers, if the nytimes.com administrator generates keys and signatures on a separate computer that has not been compromised.	DNSCurve does not protect nytimes.com DNS data against an attacker controlling the nytimes.com DNS servers.
Availability	RFC 4033: "DNSSEC does not protect against denial of service attacks."	→ DNSCurve adds some protection against dos attacks.

Administration

Component	DNSSEC	DNSCurve
DNS cache software (e.g., dnscache or PowerDNS Recursor or Unbound or BIND)	Administrator upgrades to a cache that supports DNSSEC.	Administrator upgrades to a cache that supports DNSCurve.
DNS server software (e.g., tinydns or PowerDNS Server or NSD or BIND)	Administrator upgrades to a server that supports DNSSEC.	Administrator upgrades to a server that supports DNSCurve, *or* installs a DNSCurve forwarder alongside any existing DNS server.
DNS database software (often site-developed)	Administrator upgrades to database software that supports DNSSEC records.	→ No action required.
Public-key format	Software generates a DNSKEY record that encodes a DNSSEC public key.	Software generates a new server name that encodes a DNSCurve public key.
Public-key distribution by parent	Parent extends web interface, database, registrar-registry protocol, etc. to accept DNSKEY records.	→ No action required.
Public-key distribution by child	Administrator supplies new DNSKEY record to parent through new interface.	Administrator supplies new server name through existing interface.
Initial signatures	Administrator generates signatures on DNS records, stores the signatures, and integrates the signatures into the DNS database.	→ No action required. Cryptographic protection is applied automatically by the DNS server.
Subsequent signatures	Administrator must generate new signatures every month or else domain drops off Internet ("DNSSEC suicide").	→ No action required.
Signatures on updates	Administrator generates new signatures after adding or changing DNS records.	→ No action required.