UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior

Departamento de Ingeniería Telemática

Ingeniería de Telecomunicación

Proyecto Fin de Carrera

Análisis de Seguridad en Redes IPv6

Autor: Carlos García Martín

Tutor UC3M: Daniel Díaz Sánchez

Tutor Telefónica Digital: David Barroso Berrueta

Julio 2012

AGRADECIMIENTOS

Al equipo hacking de Telefónica Digita PDI. Carlos, Fran y Rafa, gracias por vuestra ayuda y vuestros consejos.

A Dani, por su amabilidad e implicación como tutor.

A mis compañeros y amigos, en especial a Diego y Estefanía, por su compañía y ayuda a lo largo de la carrera.

A mi madre, mi hermano y mis abuelos. Todo un ejemplo de trabajo y esfuerzo, es un placer estar a vuestro lado.

A mi tío Félix, nada me hace más feliz que verle sonreír.

A María José, la razón por la que despertarme cada mañana.

RESUMEN

La revolución que ha supuesto Internet en la forma de comunicarnos ha propiciado la aparición de multitud de dispositivos electrónicos que incorporan la capacidad de conectarse a la red. Algunos de los protocolos que se diseñaron en el comienzo de Internet y que han sido utilizados hasta ahora no preveían esta expansión, como en el caso del protocolo IPv4. El agotamiento de direcciones IP es un hecho en la actualidad y es por ello que se ha comenzado a implantar una versión actualizada de este protocolo: IPv6. Sin embargo, el nuevo protocolo IPv6 no sólo produce una mejora el rango de direccionamiento, sino que también introduce nuevos mecanismos de configuración que suponen grandes ventajas respecto a IPv4.

La implantación de IPv6 no sólo trae consigo ventajas respecto a IPv4, también conlleva la aparición nuevos riesgos de seguridad y ataques. Es por ello que es necesario un estudio exhaustivo en el ámbito de la seguridad que ayude a detectar estos riesgos y la forma de tratarlos. De este modo, en el presente documento se realiza un análisis de los nuevos retos de seguridad que plantea la implantación del protocolo IPv6. En primer lugar, se detallan los principales riesgos de seguridad que aparecen con IPv6 a corto, medio y largo plazo. Además, se analizan algunos de los ataques que han aparecido en los pocos años de vida de IPv6. Este análisis se completa con el estudio de técnicas como el *packet crafting* o el *fuzzing* empleadas por los profesionales de la seguridad de red para la comprobación del nivel de seguridad de los distintos equipos que componen las redes de comunicaciones.

En este sentido, estas técnicas se han plasmado en la herramienta Scapyst, desarrollada con el objetivo de facilitar el análisis de seguridad de los equipos habilitados con IPv6 tanto a profesionales dedicados tanto a la seguridad como a la administración de redes. Por último, se utiliza esta herramienta para realizar pruebas de estabilidad sobre las implementaciones del protocolo IPv6 utilizadas en dos de los sistemas operativos de usuario más comunes.

ABSTRACT

The revolution that Internet has meant in the way we communicate has led to the emergence of many electronic devices that incorporate the ability to connect to the network. Some of the protocols that were designed in the beginning of the Internet era and have been used so far did not expect this expansion. That is the case of IPv4. The IP address exhaustion is now a fact and that is why an updated version of this protocol has begun to be implemented: IPv6. However, the new IPv6 protocol not only improves the address range, but also brings new configuration mechanisms that imply major advantages over IPv4.

The implementation of IPv6 not only brings advantages over IPv4, it also means the emergence of new security risks and attacks. That is why we need in-depth study in the field of security to detect these risks and to learn how to treat them. Thus, this paper conducts an analysis of the new security challenges posed by the deployment of IPv6. First, it describes some security risks that come with IPv6 in the short, medium and long term. It also analyzes some of the attacks that have appeared in the few years of life for IPv6. This analysis is complemented with the study of techniques used by professionals employed in the network security, like packet crafting or fuzzing, to verify the security level of the various equipment that make up the networks.

In this sense, these techniques have resulted in Scapyst tool, developed with the aim to facilitate security analysis of IPv6-enabled computers to professionals dedicated to both safety and the administration of networks. Finally, we use this tool to perform stability tests on IPv6 protocol implementations used in two most common user operating systems.

INDICE

INDICE DE FIGURAS

CAPITULO 1

INTRODUCCION Y OBJETIVOS

INTRODUCCION Y OBJETIVOS

1.1 INTRODUCCION

Un aspecto fundamental de los servicios en Internet, ya sea a nivel de aplicación o a nivel de red, es la seguridad de los mismos. Actualmente, tanto empresas como organismos públicos, dedican muchos recursos con el objetivo de mantener la seguridad y privacidad de los usuarios en Internet. Para poder mantener un nivel de seguridad adecuado, es necesario realizar pruebas no solo sobre los dispositivos y aplicaciones que nos van a dar acceso a la información, si no sobre cualquier elemento o protocolo que vaya a manejar la información dentro de Internet.

En este sentido, el protocolo IP es uno de los elementos más importantes (sino el más importante), ya que es el mecanismo principal por el que actualmente podemos hacer uso de Internet. Es por ello que este protocolo y los demás que componen la pila TCP/IP, han sido probados constantemente desde el punto de vista de rendimiento, robustez y compatibilidad. Además de estos aspectos, hay que tener en cuenta uno más: cómo de seguro es el protocolo, su capacidad de soportar posibles ataques de usuarios maliciosos u otras amenazas para ser capaz de salvaguardar nuestra seguridad como usuarios del dispositivo que hace uso del protocolo.

Existen multitud de estudios y herramientas que nos permiten conocer el nivel de seguridad de dispositivos que utilizan un protocolo consolidado como IP. Sin embargo, en la actualidad se presenta un nuevo reto: la introducción del protocolo IPv6, diseñado para sustituir al protocolo IP en un futuro no muy lejano. La aparición de IPv6 es fundamental para el crecimiento y desarrollo de Internet, pero su implantación conlleva nuevos riesgos de seguridad. Por ello, es necesario realizar un análisis exhaustivo de cada uno de los aspectos y características que componen el nuevo protocolo. Para realizar este tipo de pruebas, es necesario contar con herramientas que nos permitan modificar a nuestro antojo los datos que intercambiamos con los dispositivos.

En la actualidad, las herramientas existentes para el análisis de vulnerabilidades de IPv6 son costosas o requieren un conocimiento técnico que muchas veces un administrador de red no posee. Desde el equipo de Hacking Ético de PDI (*Product Development and Investigation*) del grupo Telefónica Digital (dentro del cual se ha desarrollado este proyecto) se detectó la necesidad de implementar una herramienta *Open Source* que permita realizar las pruebas oportunas sobre IPv6. La idea era realizar una herramienta sencilla de utilizar e intuitiva, con una amplia funcionalidad, que permitiera realizar pruebas de seguridad sobre IPv6, no sólo a personas con amplios conocimientos técnicos sino a cualquier persona que estuviera interesada en analizar la seguridad de sus equipos.

Actualmente, el consorcio *IPv6 Forum* ha puesto en marcha el programa denominado "*IPv6 Ready*", donde se definen las especificaciones de prueba para la conformidad deIPv6, con el objetivo de verificar la implementación del protocolo y validar la interoperabilidad de los productos IPv6. Basándonos en estas pruebas, se puede realizar un estudio sobre las vulnerabilidades a que está expuesto un dispositivo debido a una mala implementación o configuración del protocolo IPv6.

Para la realización de estas pruebas, se emplean técnicas de *packet crafting* y *fuzzing*, las cuales se basan en la modificación de paquetes de red para la comprobación del funcionamiento de los distintos elementos de red. Estas técnicas han resultado ser muy efectivas y en la actualidad son utilizadas por empresas como Codenomicon [1] y Symantec [2]. Además, Microsoft incluye estas pruebas como parte de su ciclo de desarrollo software [3].

Este proyecto tratará por tanto de analizar las implicaciones en cuestiones de seguridad que surgen a raíz de la transición de IP a IPv6, haciendo especial hincapié en las técnicas de *fuzzing* aplicadas a IPv6 y basándonos en las pruebas definidas en la iniciativa *"IPv6 Ready*". Además, este estudio se apoyará en el desarrollo de una herramienta, llamada Scapyst, que nos permita realizar pruebas sobre el protocolo IPv6 en distintos escenarios.

1.2 OBJETIVOS

El objetivo general del proyecto es realizar una contribución al ámbito de la seguridad informática, enfocándonos en analizar una tecnología que se encuentra en una fase muy temprana de implantación. De forma más específica y como se ha visto en la introducción, con el presente Proyecto Fin de Carrera se pretende abarcar dos objetivos principales: realizar un estudio sobre nuevas vulnerabilidades que aparecerán con la implantación de IPv6 y desarrollar una herramienta que nos permita analizar dichas vulnerabilidades y realizar pruebas sobre distintas implementaciones de la pila de IPv6.

En un primer momento, se describirá el protocolo IPv6 de forma general, destacando los principales cambios respecto a IPv4. A su vez, se presentarán las tecnologías y herramientas utilizadas para el estudio. Una vez que hemos presentado el contexto en el que se enmarca el proyecto, se pasarán a realizar los objetivos principales, que se presentan con mayor detalle a continuación:

- El primer objetivo será analizar los nuevos retos en cuanto a seguridad que implica la transición de IPv4 a IPv6, qué aspectos se mantienen y qué aspectos cambian. Dentro de este objetivo podemos destacar alguno de los puntos más importantes que lo componen:

    - ✓ Analizar las técnicas de *fuzzing* existentes y extrapolarlo para nuestro caso de estudio (IPv6). Para esta tarea también nos apoyaremos en las pruebas de certificación definidas por el *IPv6 Forum*.
    - ✓ Analizar distintos ataques conocidos sobre IPv6: ver qué vulnerabilidad explotan, cómo actúan y dar posibles soluciones para mitigarlos.
- El segundo objetivo es el desarrollo de una herramienta que nos permita realizar análisis de vulnerabilidades en IPv6. Este paso lo podemos desglosar en los siguientes puntos:

- ✓ Completar la interfaz de usuario (en el momento de comenzar mi estancia en Telefónica Digital, ya había una versión previa de la interfaz).
- ✓ Dotar de funcionalidad a la herramienta para que sea capaz de crear e inyectar paquetes IPv6 de una forma intuitiva y simple.
- ✓ Implementar aspectos que faciliten la interacción con la herramienta al usuario: guardar y cargar paquetes, información sobre el envío…
- ✓ Implementar algunos ataques sobre IPv6 analizados para que el usuario puede ejecutarlos fácilmente.
- ✓ Desarrollar un "modo servidor", de forma que el usuario pueda crear un flujo de paquetes, analizando los paquetes recibidos y pudiendo interactuar con ellos.
- ✓ Implementar las técnicas de *fuzzing* estudiadas para algunos de los principales mensajes sobre IPv6.
- ✓ Prueba y depuración de la herramienta.

Además, podemos establecer un tercer objetivo que complementa el desarrollo de los dos anteriores:

- Realización de pruebas sobre distintas implementaciones de pilas IPv6.

  - ✓ Pruebas sobre Windows 7.
  - ✓ Pruebas sobre Unix.

1.3 FASES DEL DESARROLLO

Como ya se ha comentado, el proyecto se ha desarrollado dentro del equipo de Hacking Ético de PDI, Telefónica Digital. El mismo dio comienzo en Noviembre de 2011 y los pasos seguidos hasta su finalización han sido los siguientes (algunas de las partes se han solapado, ya que la realización de ciertas tareas requería elementos de otras partes del proyecto):

- Desarrollo de la herramienta Scapyst. (Noviembre 2011-Abril 2012)
- Prueba de la herramienta Scapyst. (Abril 2012-Mayo 2012)
- Estudio de vulnerabilidades del protocolo IPv6 y técnicas de fuzzing. (Marzo 2012-Mayo 2012)
- Pruebas sobre distintas implementaciones del protocolo IPv6 (Abril 2012-Junio 2012)
- Análisis de resultados y desarrollo de la memoria. (Junio 2012 – Julio 2012)

1.4 MEDIOS EMPLEADOS

Los medios utilizados podemos dividirlos básicamente en dos grupos:

- Hardware: MacBook Pro Intel Core 2 Duo, procesador de 2.53 GHz, 4 GB de memoria RAM.
- Software: QtCreator, Python, librería Scapy, sistemas operativos virtualizados (Distribución Backtrack), herramienta Scapyst.

1.5 ESTRUCTURA DE LA MEMORIA

La memoria consta de los siguientes capítulos:

- **Introducción:** Se destaca la importancia de la seguridad informática de cara a la privacidad del usuario y se introduce sobre los nuevos retos que aportará la adopción de IPv6 como protocolo de red. Se detallan los objetivos del proyecto y las fases del desarrollo.
- **Estado del Arte:** Descripción general de las tecnologías utilizadas en el presente Proyecto Fin de Carrera. Se describirá de forma general el protocolo IPv6, así como los protocolos y mecanismos más relevantes relacionados con él. Después se hará una descripción de las técnicas de *packet crafting* y *fuzzing* analizando las principales herramientas existentes.

Por último, se analizarán las tecnologías utilizadas en el desarrollo de la herramienta Scapyst.

- **Nuevos riesgos de seguridad en IPv6. Fuzzing IPv6:** En este capítulo se realiza un análisis tanto de nuevos riesgos de seguridad que aparecen con la implantación del protocolo IPv6 como de ataques a equipos que soportan dicho protocolo. Se describen las técnicas de *fuzzing* aplicadas al protocolo IPv6.
- **Herramienta Scapyst:** Se presenta la herramienta para inyección de paquetes IPv6 desarrollada, describiendo cada una de sus funcionalidades.
- **Pruebas de estabilidad:** Descripción de las pruebas realizadas sobre equipos que soportan IPv6. Se detalla el entorno de pruebas utilizado y los resultados obtenidos en la realización de las pruebas.
- **Conclusiones y trabajo futuro:** Se realiza un breve análisis como conclusión final y se marcan las líneas de trabajo futuro para dar continuación al presente Proyecto Fin de Carrera.
- **Presupuesto:** Se detallan los costes asociados a la realización del proyecto.

CAPITULO 2

ESTADO DEL ARTE

ESTADO DEL ARTE

En el presente capítulo se describen las tecnologías que han sido necesarias estudiar para la realización del proyecto con el objetivo de establecer el contexto necesario para el desarrollo de los capítulos posteriores. Se comienza con un repaso del protocolo IPv6, describiendo los nuevos mecanismos que hacen posible la comunicación con el nuevo protocolo. Después, se describen en qué consisten las técnicas de *packet crafting* que se utilizan en el análisis de seguridad del protocolo. Para terminar el capítulos se describen brevemente las tecnologías utilizadas en el desarrollo de la herramienta Scapyst.

2.1 PROTOCOLO IPv6

El *Internet Engineering Task Force* (IETF) es la organización encargada de definir los estándares de los protocolos de internet. IPv4 se definió en [4] y fue el primer protocolo de internet implementado a gran escala. Este protocolo se desarrolló en los años 80 y en un principio se utilizaba para conectar un pequeño conjunto de organizaciones. Cuando se desarrolló IPv4 no se pensó en la gran expansión del uso de internet que se produciría a principios de los años 90, por lo que el protocolo se diseñó con direcciones de 32 bits, algo que en la actualidad es insuficiente debido a que existen multitud de dispositivos que hacen uso de internet. Además del espacio de direcciones, se detectaron otras características mejorables en IPv4 como aspectos de seguridad, priorización de tráfico, etc.

Así es como a principios de los años 90 el IETF se comenzó a plantear el desarrollo de una nueva versión del protocolo de internet. Esto llevo a la aparición del estándar de IPv6 en el RFC 1883 ([5]) en 1995,que luego se reemplazó por el RFC 2460 ([6]) en 1998. Con IPv6 se mejoran algunas características de IPv4, permitiendo nuevas formas de comunicación y dotándolas de mayor seguridad. De las muchas mejoras introducidas en IPv6, las más relevantes son:

- Incremento del espacio de direcciones: de 32 bits en IPv4 a 128 bits en IPv6.

- Simplificación de la cabecera: se aumenta la eficiencia en el reenvío de paquetes.
- Cabeceras de Extensión: se mejora el manejo de opciones con la introducción de las cabeceras de extensión.
- Protocolo SLAAC (*Stateless Adress Autoconfiguration*): los nodos pueden determinar sus propias direcciones IPv6.
- Mejora el uso de comunicaciones *mulitcast*.
- Uso de *Jumbograms*: permite el envío de paquetes con cargas mayores para mejorar la eficiencia.
- Seguridad: facilita el cifrado y la autenticación de las comunicaciones.
- Calidad de servicio: marcado de paquetes y etiqueta de flujo que permite establecer prioridades de tráfico.
- Movilidad: manejo simplificado de nodos itinerantes.

En este capítulo se explicarán las principales características de IPv6, su nuevo formato de cabecera y la nueva versión del protocolo ICMP (ICMPv6). Además, el paso de IPv4 a IPv6 afecta a otros protocolos como DHCP o DNS que también se explicarán brevemente en esta sección.

2.1.1 Cabecera IPv6

La nueva cabecera que se define en el protocolo de internet incluye cambios significativos respecto a la anterior cabecera. En la Figura 2.1 podemos ver el formato de cabecera para IPv6 y en la Figura 2.2 el de IPv4.

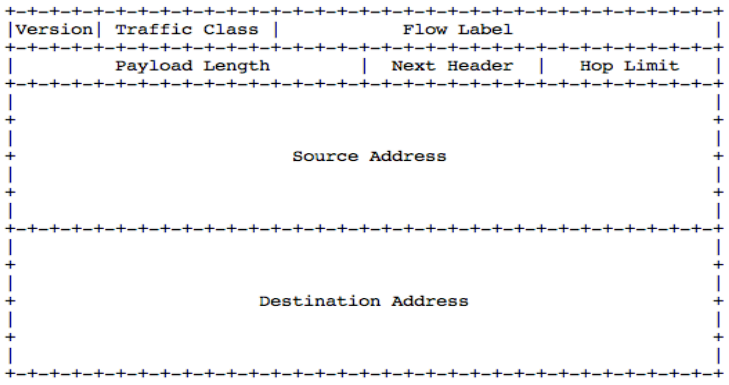

Figura 2.1. Cabecera IPv6 (RFC 2640)

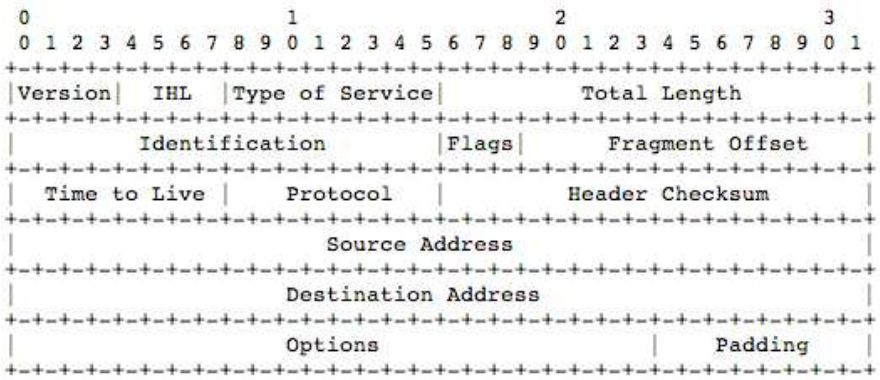

Figura 2.2. Cabecera IPv4 (RFC 791)

En primer lugar podemos notar cómo se han eliminado ciertos campos de la cabecera IPv4. El primero que se elimina es el "IHL" (*Internet Header Length*) debido a que en IPv6 la longitud de la cabecera es fija. En IPv4 se establece una longitud mínima de cabecera de 20 bytes pero existe la posibilidad de añadir opciones con una longitud máxima de 60 bytes. Sin embargo en IPv6, se fija la longitud de la cabecera a 40 bytes, manejando las opciones en cabeceras de extensión (las veremos más adelante en este capítulo), lo que hace que el procesamiento de la cabecera sea más rápido. También se eliminan los campos "*Identification*", "*Flags*" y "*Fragment Offset*", todos ellos relacionados con la fragmentación de paquetes. En IPv6 la fragmentación se maneja mediante una cabecera de extensión por los que estos campos no son necesarios. Por último, se elimina el campo *checksum*. En la nueva cabecera del protocolo de internet no se realiza la comprobación de errores lo que aumenta la rapidez en el procesamiento de los paquetes debido a que los routers no deben comprobar la validez del *checksum* y actualizarlo. Esto no es un inconveniente debido a que en la actualidad tanto la capa de acceso al medio como las capas de transporte realizan comprobación de errores. En relación con esto, es importante destacar que, con IPv4, era opcional incluir *checksum* en el protocolo UDP, con IPv6 es obligatorio.

Una vez visto los campos que no son necesarios en la nueva cabecera IPv6, vemos cuáles se han introducido o modificado:

- *Version*: se mantiene el campo y se cambia su valor a 6 para identificar la cabecera IPv6.

- *Traffic Class*: sustituye el campo *Type of Service* de IPv4. Permite establecer prioridades de tráfico y facilita el manejo de ciertos tipos de paquetes como los de las comunicaciones en tiempo real.
- *Flow Labe*l: indica que un paquete pertenece a un flujo de tráfico determinado, esto es, que debe tratarse de igual a todos los paquetes procedentes de la misma fuente y con el mismo *Flow Label*.
- *Payload Length*: contiene la longitud de la carga del paquete, excluyendo la cabecera. En el cálculo también se incluyen cabeceras de extensión. Dado que el campo tiene un tamaño de 2 bytes, el tamaño de la carga está limitado a 64 KB. IPv6 permite el envío de paquetes de mayor tamaño mediante el uso de cabeceras de extensión (*Jumbograms*) que veremos después en este capítulo.
- *Next Header*: contiene un indicador sobre la cabecera que sigue a continuación, sustituyendo el campo *Protocol* de la cabecera de IPv4. Al no existir cabeceras de extensión, en IPv4 este campo indicaba el protocolo que seguía a IP pero en IPv6 este campo puede indicar también una cabecera de extensión. El listado de valores que puede tomar este campo lo podemos encontrar en [7].
- *Hop Limit*: es el campo análogo al TTL (*Time to Live*) en IPv4. El objetivo inicial del TTL en IPv4 era actuar como un contador de tiempo, de forma que el paquete no pudiera sobrepasar este tiempo en la red. Al final, los routers acabaron simplificando el proceso y simplemente decrementaban el valor del TTL. Así es como en IPv6 este campo ha pasado a denominarse *Hop Limit*, representando el número máximo de nodos que puede atravesar el paquete. Cuando este campo llega a 0, el paquete debe ser descartado.
- *Source Address, Destination Address*: contienen las direcciones origen y destino del paquete. Este es, casi con todo seguridad, el cambio más importante respecto de IPv4 dado que las direcciones han aumentado su tamaño hasta los 128 bits en IPv6.

2.1.2 Direccionamiento en IPv6

Como ya se comentó en el apartado anterior, la principal ventaja del nuevo protocolo IPv6 es el aumento del espacio de direcciones. En la actualidad la población mundial ha sobrepasado los 7000 millones de habitantes. Con el tamaño de direcciones disponible en IPv4 tendríamos 2^{32} direcciones, o lo que es lo mismo 4.294.967.296 direcciones. Aunque una gran parte de la población mundial aún no dispone de acceso a internet, la aparición de dispositivos de nueva generación capaces de conectarse a internet ha hecho que el agotamiento de direcciones IPv4 se acelere. Durante años se ha intentado atenuar este efecto empleando técnicas como NATs, lo que sin embargo no ha impedido que en Febrero de 2011 la IANA (*Internet Assigned Numbers Authority*) haya asignado los últimos bloques de direcciones libres a los RIRs (*Regional Internet Registers*). Aunque esto no quiere decir que se vayan a agotar en un corto plazo, las organizaciones han dado un paso al frente para comenzar a implantar el nuevo protocolo de internet. Con IPv6 el espacio de direcciones se incrementa hasta 2^{128} direcciones, lo que equivale a 340.282.366.920.938.463.463.374.607.431.768.211.456 direcciones. Como vemos, el problema de las direcciones se soluciona.

En IPv6, las direcciones se clasifican en tres categorías:

- *Unicast*: identifica un único interfaz de red.
- *Multicast*: identifica un grupo de interfaces IPv6. El paquete enviado a la dirección *multicast* será entregado a todos los interfaces de red pertenecientes a dicho grupo.
- *Anycast*: identifica un grupo de interfaces IPv6. En este caso, el paquete enviado a la dirección *anycast* será entregado a uno de los interfaces del grupo, normalmente al de menor coste según la métrica del protocolo de encaminamiento utilizado.

Notación.

Debido a la gran longitud de las direcciones IPv6 (128 bits, 16 bytes), estas se separan en 8 bloques de 16 bits separados por dos puntos. Un ejemplo sería:

FE80:0000:0000:0226:0000:BBFF:FE04:1DC2

Para facilitar la lectura de las direcciones, se toman algunas reglas:

- Los ceros a la izquierda de cada bloque de 16 bits se pueden eliminar. La dirección indicada anteriormente quedaría:

  FE80:0:0:226:0:BBFF:FE04:1DC2

- Se pueden reemplazar grupos de ceros consecutivos por "::". Esta regla sólo puede aplicarse una vez en la dirección IPv6 ya que de lo contrario no podría determinarse cuántos ceros han sido reemplazados. De nuevo, para el ejemplo anterior, podrían ser direcciones válidas las siguientes:

  FE80::226:0:BBFF:FE04:1DC2
  FE80:0:0:226::BBFF:FE04:1DC2

La notación para la representación de la longitud del prefijo es la misma que en IPv4: Dirección-IPv6/Longitud-prefijo. Por ejemplo, las siguientes son representaciones válidas del prefijo 12AB00000000CD30 (60 bits):

12AB:0000:0000:CD30:0000:0000:0000:0000/60
12AB::CD30:0:0:0:0/60
12AB:0:0:CD30::/60

Prefijos IPv6.

Como ya hemos visto, existen distintos tipos de direcciones IPv6, que por lo general podemos diferenciar por el prefijo. El IETF ha definido los prefijos que podemos ver en la Figura 2.3 para los distintos tipos de direcciones IPv6.

Address type	Binary prefix	IPv6 notation
Unspecified	00...0 (128 bits)	::/128
Loopback	00...1 (128 bits)	::1/128
Multicast	11111111	FF00::/8
Link-local unicast	1111111010	FE80::/10
Site-local unicast	1111111011	FEC0::/10
Global unicast	(everything else)	

Figura 2.3. Prefijos IPv6 (RFC 3513)

(Las direcciones Anycast no tienen prefijo asignado, lo toman del espacio de direcciones de las direcciones global unicast. Sintácticamente son indistinguibles).

Las direcciones *Site-Local* fueron definidas en las primeras especificaciones pero posteriormente se eliminaron [8], sustituyéndose por las direcciones *Unique-Local Unicast* con prefijo FC00::/7, definidas en [9].

En [10] podemos encontrar en detalle la definición de cada tipo de dirección, que a continuación se resume:

Direcciones Global Unicast.

Las direcciones *Global Unicast* pueden ser enrutadas por todo internet y son globalmente únicas. En la Figura 2.4 podemos ver el formato de este tipo de direcciones:

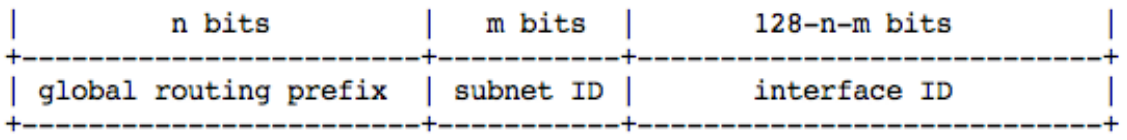

Figura 2.4. Estructura Direcciones Global Unicast (RFC 4291)

Normalmente el identificador de interfaz tendrá 64 bits en formato EUI-64 donde el identificador se construye a partir de la dirección MAC de nuestra tarjeta de red (existen algunas excepciones como por ejemplo direcciones IPv4 encapsuladas en direcciones IPv6).

La IANA es el organismo que se encarga de asignar las direcciones y el único prefijo que, de momento, se ha asignado a estas direcciones es el 2000::/3.

Direcciones *Link-Local* y *Unique-Local Unicast*.

Las direcciones *Link-Local Unicast* están pensadas para utilizarse en un enlace local, por lo que no pueden utilizarse fuera de este ámbito. Suelen usarse para mecanismos como la autoconfiguración de direcciones, *Neighbour Discovery*, o cuando no existen routers.

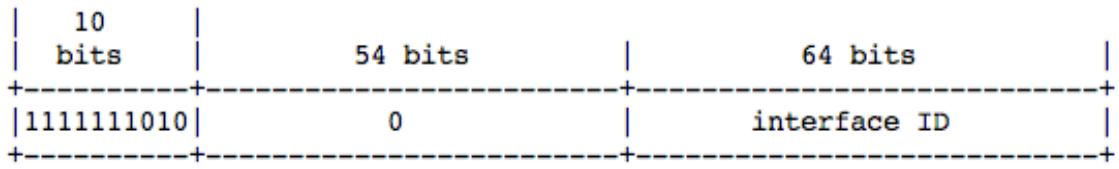

Figura 2.5. Estructura Direcciones Link-Local (RFC 4291)

Las direcciones Unique-Local Unicast, o ULA (*Unique-Local Address*), tienen el formato de la Figura 2.6:

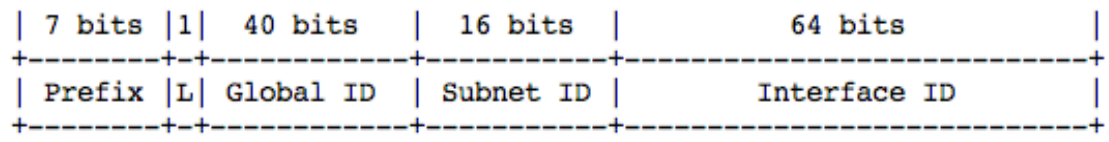

Figura 2.6. Estructura Direcciones Unique-Local (RFC 4291)

Dónde:

- *Prefix*: prefijo asignado a las direcciones ULA, FC00::/7.
- L: indica si Global ID está localmente asignado
- *Global ID*: identificador para crear un prefijo global único, aleatoriamente asignado.
- *Subnet ID:* subred dentro del *site*.
- *Interface ID*: 64 bits en formato EUI-64.

Este tipo de direcciones tienen un ámbito local y son enrutables sólo dentro de un ámbito corporativo. A pesar de su uso local, la dirección se pretende que sea única y global para evitar conflictos si llegasen a enrutarse hacia Internet.

Direcciones *Unicast* Especiales.

- Dirección no especificada: se representa como “0:0:0:0:0:0:0:0” (o simplemente “::”). Indica la ausencia de una dirección válida y puede usarse, por ejemplo, en el proceso de arranque de ciertas máquinas.
- Dirección de *loopback*: tiene el mismo significado que en IPv4, se utiliza para enviar un paquete sin que salga a la red, únicamente atraviesa la pila de la máquina. Se representa por “0:0:0:0:0:0:0:1” (“::1").

- Direcciones IPv6 con direcciones IPv4 mapeadas: se utilizan para representar direcciones IPv4 dentro de direcciones IPv6, de forma que un nodo utilizando IPv6 pueda comunicarse con un nodo que utiliza IPv4. Estas direcciones se forman poniendo 80 bits de la dirección a '0', 16 bits '1' y los 32 restantes contienen la dirección IPv4, pudiéndose escribir estos últimos en decimal, por ejemplo: "::ffff:192.168.89.9" (o en formato IPv6 "::ffff:c0a8:5909").
- Otros tipos de direcciones IPv6 utilizados para mecanismos de transición como 6to4 o Teredo. Para ver en detalle el resto de direcciones especiales en IPv6 se puede consultar [11].

Direcciones Multicast

Como se ha dicho, las direcciones *multicast* identifican un grupo de nodos y tienen la estructura que vemos a continuación:

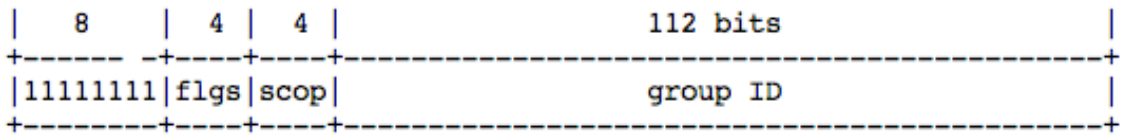

Figura 2.7. Estructura Direcciones Multicast (RFC 4291)

El primer byte identifica la dirección como *multicast* (FF), los siguiente 4 bits se usan como *Flags*: el primero se reserva para uso futuro, el segundo determina si la dirección tiene un punto de encuentro (punto de distribución para un flujo *multicast* específico), el tercero nos dice si la dirección contiene información de prefijo y la cuarta indica si se trata de una dirección *multicast* reservada o es temporal. El campo "*Scope*" delimita el ámbito de la dirección *multicast*, pudiendo ser los ya vistos (Figura 2.8): interfaz, local o global o algún otro configurado por el administrador de red.

0	reserved
1	Interface-Local scope
2	Link-Local scope
3	reserved
4	Admin-Local scope
5	Site-Local scope
6	(unassigned)
7	(unassigned)
8	Organization-Local scope
9	(unassigned)
A	(unassigned)
B	(unassigned)
C	(unassigned)
D	(unassigned)
E	Global scope
F	reserved

Figura 2. 8. Ámbitos Multicast (RFC 4291)

Los últimos 112 bits contienen el identificador de grupo que, como ya se ha dicho, puede determinar una dirección conocida o bien una dirección *multicast* temporal. Las direcciones *multicast* conocidas son asignadas por el IANA, y a continuación podemos ver las más relevantes (para consultar la asignación hasta el momento, se remite a [12]:

Address(s)	Description
FF01:0:0:0:0:0:0:1	All Nodes Address
FF01:0:0:0:0:0:0:2	All Routers Address

Figura 2.9. Direcciones Multicast, ámbito Interfaz (IANA)

Address(s)	Description
FF02:0:0:0:0:0:0:1	All Nodes Address
FF02:0:0:0:0:0:0:2	All Routers Address

Figura 2.10. Direcciones Multicast, ámbito Enlace Local (IANA)

Por último, se definen las direcciones *multicast solicited-node.* Cada nodo debe tener una dirección de este tipo por cada dirección *unicast* o *anycast* configurada. Se utilizan para el mecanismo *Neighbor Discovery* que más tarde veremos y se forman de la siguiente forma: se toma el prefijo FF02:0:0:0:0:1:FF00::/104 y se añaden los 24 bits de menor orden de la dirección IPv6 correspondiente. Por ejemplo, para la dirección *link-local*

FE80::202:0:fbcc:1234 la dirección *multicast* que le correspondería sería FF02::1:FFcc:1234.

Direcciones IPv6 requeridas.

Cada nodo debe tener configuradas, al menos, las siguientes direcciones IPv6 para identificarse:

- Dirección *link-local* para cada interfaz.
- Alguna dirección *unicast* o *anycast* asignada.
- Dirección de *loopback*.
- Dirección "*all-nodes*".
- Dirección *multicast solicited-node* por cada dirección *unicast* o *anycast* configurada.
- Dirección *multicast* para cada grupo al que pertenezca el host.

Además de estas, un router necesita añadir las siguientes:

- Dirección *anycast* de la subred para las interfaces que tiene configuradas para actuar como router en el enlace.
- Dirección *anycast* con las que el router ha sido configurado.
- Dirección *multicast all-routers.*
- Direcciones *multicast* para los grupos a los que el router pertenece.

2.1.3 Cabeceras de Extensión

Otro importante cambio que se introduce en el protocolo IPv6 es la inclusión de las cabeceras de extensión. En IPv4 las opciones se anexaban a la propia cabecera del protocolo, lo que hacía que esta tuviera una longitud variable. En IPv6, las opciones se añaden en forma de cabeceras de extensión, pudiendo incluirse ninguna, una o varias en un mismo paquete. El método empleado para determinar la presencia de cabeceras de extensión es mediante el campo *"Next Header"* presente tanto en la cabecera IPv6, como en la estructura de las cabeceras de extensión (Figura 2.11). Esta nueva forma de incluir opciones hace que los paquetes sean más simples y que el procesamiento de éstos sea mucho más rápido

ya que en muchas no es necesario analizar el paquete mediante software. Además esta arquitectura define una forma flexible en el desarrollo de nuevas opciones.

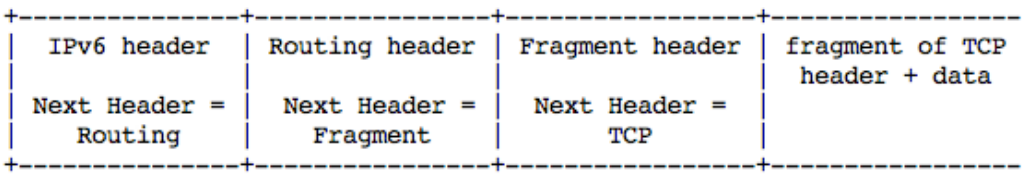

Figura 2.11. Esquema Cabeceras de Extensión (RFC 2640)

De las posibles cabeceras de extensión, todas salvo una (Hop-By-Hop) son analizadas sólo en el destino lo que disminuye la carga de procesamiento en los nodos intermedios. A continuación pasamos a describir las cabeceras de extensión, poniendo mayor énfasis en las cuatro primeras ya que son las más comunes.

***Hop-By-Hop Header*.**

La cabecera Hop-By-Hop porta opciones que deben ser examinadas por cada nodo que atraviesa el paquete. La estructura de esta cabecera se muestra en la Figura 2.12:

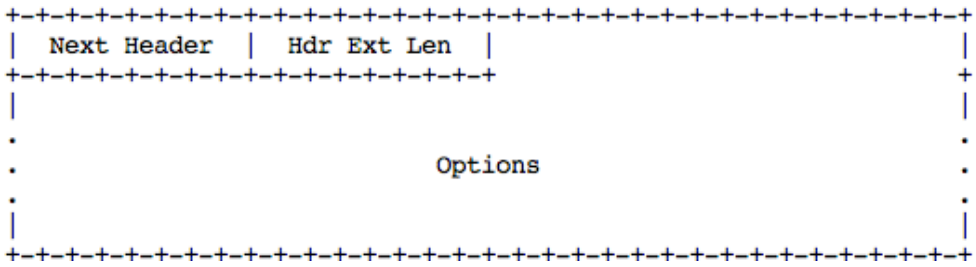

Figura 2.12. Cabecera Hop-By-Hop (RFC 2640)

El campo *"Next Header"* indica el tipo de protocolo que sigue a la cabecera de extensión y el campo *"Hdr Ext Len"* marca la longitud de la cabecera en unidades de 8 bytes (sin contar los primeros 8 bytes).

Las opciones tienen formato TLV (*Type-Length-Value*), como se muestra en la Figura 2.13:

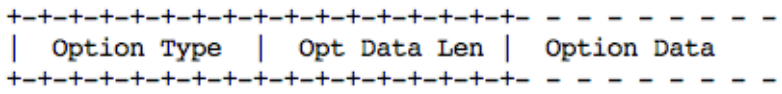

Figura 2.13. Formato de Opciones Cabeceras de Extensión (RFC 2640)

Puede haber una o más opciones en una misma cabecera de extensión. El campo *"Option Type"* indica el tipo de opción, o en caso de que no se reconozca la opción, indica la acción a realizar por el nodo. En este último caso, los dos primeros bits determinan qué hacer con el paquete:

- '00': omitir y continuar con el procesamiento del paquete.
- '01': descartar el paquete.
- '10': descartar y enviar un mensaje *ICMP Parameter Problem Code 2* al destino.
- '11': descartar y enviar un mensaje *ICMP Parameter Problem Code 2* al destino sólo si el destino no es una dirección *multicast*.

El tercer bit indica si la opción puede cambiar durante el trayecto del paquete.

Todos los tipos de opciones asignados por el momento se pueden consultar en [7], siendo los más importantes:

- *Jumbograms*: permiten el envío de paquetes de gran tamaño, extendiendo la longitud máxima permitida de la carga en IPv6 (65.535 bytes) hasta los 4.294.697.295 bytes.
- *Router Alert:* indica que el paquete contiene información importante que debe ser procesada al reenviar el paquete. Suele usarse para protocolos como RSVP (*ResourceReservation Protocol)* y MLD (*Multicast Listener Discovery*).

***Routing Header*.**

La cabecera de *Routing* se utiliza para indicar que un paquete debe atravesar uno o más nodos intermedios antes de entregarse al destino. La Figura 2.14 muestra el formato de esta cabecera:

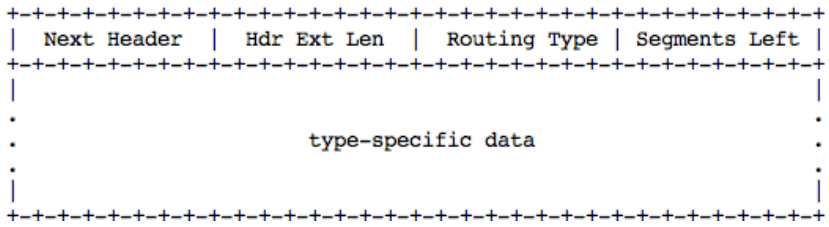

Figura 2.14. Cabecera de Routing(RFC 2640)

Se añaden dos campos a los ya vistos en la cabecera *Hop-By-Hop*: *"Routing Type" y "Segments Left".* El primero indica el tipo de cabecera de *Routing*, siendo los únicos definidos hasta ahora los tipos 0 y 2. El segundo contiene el número de nodos restantes hasta alcanzar el destino final.

La cabecera de *Routing* tipo 0, fue la primera que se introdujo y permitía indicar al paquete la ruta que debía seguir antes de llegar al destino. Esto conllevaba muchas implicaciones de seguridad por lo que se declaró obsoleta [13].

El tipo 2 es utilizado para mecanismos de movilidad en IPv6, permitiendo a los paquetes ser enrutados directamente desde el nodo móvil correspondiente [14].

Fragment Header.

En IPv6 la fragmentación se introduce también como cabecera de extensión. Como se ha dicho, las cabeceras de extensión (salvo *Hop-By-Hop)* se procesan en el destino por lo que los nodos intermedios no pueden fragmentar. Esta es otra novedad que se introduce en IPv6: los paquetes se fragmentan únicamente en el origen y se reensamblan en el destino. Este esquema libera de procesamiento a los nodos intermedios y hace que el campo "*don't fragment*" de IPv4 no sea necesario.

El formato de la cabecera es el siguiente:

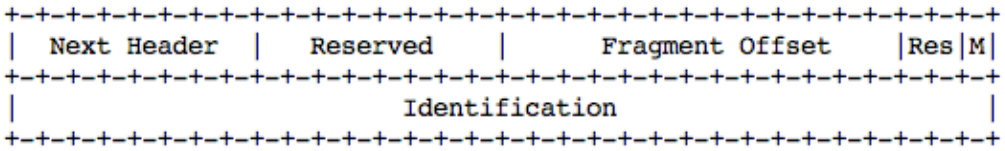

Figura 2.15. Cabecera de Fragmentación (RFC 2640)

El campo *"Fragment Offset"* es el offset, en unidades de 1 byte, de los datos que contiene el paquete en relación con el comienzo de la parte fragmentable del paquete original. "M" indica si existen más fragmentos ('1') o es el último ('0'). Por último el campo *"Identification"* marca el paquete con un identificador que deberá ser igual para todos los fragmentos del paquete original para que puedan reensamblarse en destino. Los otros dos campos se reservan para uso futuro.

El esquema de fragmentado puede verse en la Figura 2.16:

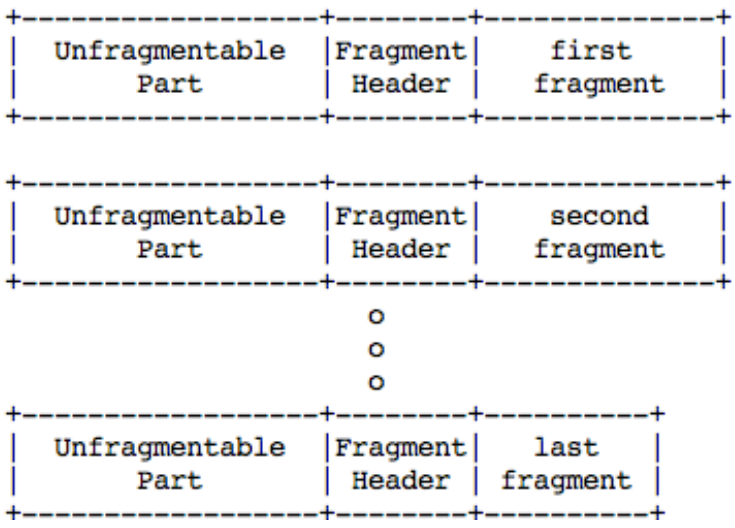

Figura 2.16. Esquema de fragmentación (RFC 2640)

La parte no fragmentable estaría formada por la cabecera IPv6 y la cabecera *Hop-By-Hop,* en caso de que hubiera, ya que esta debe ser procesada por cada nodo.

Destination Option Header.

Esta cabecera lleva información que sólo es procesada en el destino. La estructura es idéntica a la cabecera *Hop-By-Hop* (Figura 2.12) así como las posibles opciones que se pueden incluir.

Esta cabecera es la única que puede aparecer dos veces en un mismo paquete, en el caso en que se añada la cabecera de *Routing.* En este caso, una cabecera *Destination Option* se insertaría antes de la cabecera de *Routing* de tal

forma que las direcciones presentes en ésta la analizarían, y otra se podría insertar después de forma que sólo la analizara el destino final.

Otras cabeceras.

Actualmente existen otras cabeceras de extensión definidas, sin embargo su uso no está extendido. La cabecera *Authentication Header*[15] permite autenticar la comunicación, la cabecera *Encapsulating Security Payload Header*[16] maneja la encriptación de la información que se transporta y ambas son la base del protocolo IPSEC. Estas dos cabeceras son de gran importancia ya que harían que multitud de ataques existentes no pudieran llevarse a cabo, sin embargo, su utilización es escasa. Por último, también está definida la cabecera *Mobility Header*[14], utilizada para funciones de movilidad en dispositivos móviles.

Orden de las cabeceras.

En [6] se recomienda que las cabeceras de extensión se coloquen en el siguiente orden:

1. IPv6.
2. *Hop-By-Hop.*
3. *Destinantion Option.*
4. *Routing*
5. *Fragment.*
6. *Authentication.*
7. *Encapsulating Security Payload.*
8. *Destination Option.*
9. Capa superior.

Se recomienda, aunque no es obligatorio, que las cabeceras sigan este orden y que cada cabecera aparezca como máximo una vez en el paquete, exceptuando la cabecera *Destination Option* que puede aparecer dos veces en el caso de que se incluya la cabecera de *Routing.*

2.1.4 ICMPv6

ICMPv6 es la nueva versión del protocolo ICMP (*Internet Control Message Protocol*), que era el protocolo encargado del control y notificación de errores en IPv4. En esta nueva versión del protocolo se añaden un mayor número de funcionalidades, ya que en ICMPv6 se combinan protocolos de IPv4 como ARP o IGMP.

El protocolo ICMPv6 está definido en [17] y agrupa distintas funciones compuestas por mensajes que pueden clasificarse de la siguiente forma:

- Mensajes de error:
  - *Destination Unreachable.*
  - *Packet Too Big*.
  - *Time Exceeded*.
  - *Parameter Problem*.
- Mensajes de información:
  - *Echo Request.*
  - *Echo Reply.*
- *Neighbor Discovery Protocol:*
  - *Router Solicitation.*
  - *Router Advertisement.*
  - *Neighbor Solicitation.*
  - *Neighbor Advertisement.*
  - *Redirect.*
- *Multicast Router Discovery:*
  - *Multicast Router Adevertisment.*
  - *Multicast Router Solicitation.*
  - *Multicast Router Termination.*
- *Mobile ICMPv6:*
  - *Home Agent Address Discovery Request Message*
  - *Home Agent Address Discovery Reply Message*
  - *Mobile Prefix Solicitation*
  - *Mobile Prefix Advertisement*

- *Secure Neighbor Discovery:*
    - *Certification Path Solicitation.*
    - *Certification Path Advertisement.*
- Otros mensajes:
    - *Router Renumbering.*
    - *Node Information Query.*
    - *Node Information Response.*
    - *Inverse Neighbor Discovery Solicitation.*
    - *Inverser Neighbor Discovery Advertisement.*

En esta sección se comentarán los tres primeros grupos por ser los más comunes, para una información más detallada de cada tipo de mensaje pueden consultarse [11] y [18].

El formato general de cabecera se muestra en la Figura 2.17:

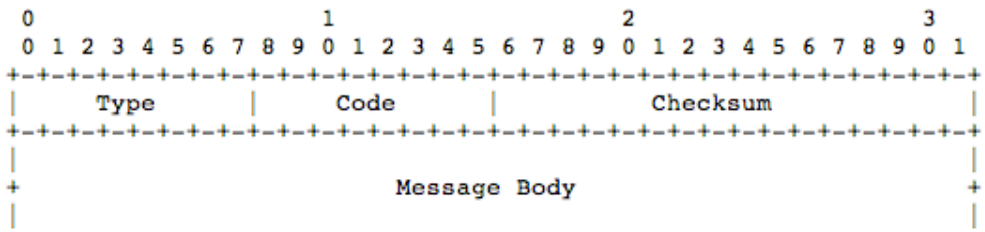

Figura 2.17. Cabecera ICMPv6 (RFC 4443)

De forma general, cada mensaje ICMPv6 tendrá un campo "*Type*" que indicará el tipo de mensaje ICMPv6, un campo *"Code"* que dará información específica dependiente del tipo de mensaje, un *"Checksum"* para detectar si se han corrompido datos en la cabecera ICMPv6 e IPv6.

Mensajes de error

- *Destination Unreachable:*

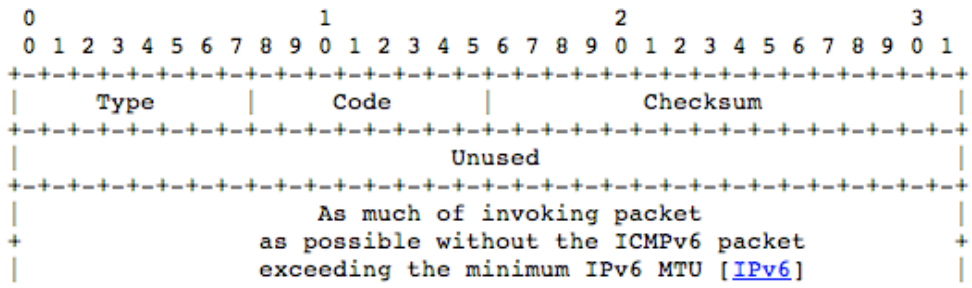

Figura 2.18. Cabecera Destination Unreachable (RFC 4443)

Este mensaje se genera cuando un paquete no puede ser entregado al destino. El campo *"Type"* toma el valor 1 y el campo "*Code"* da información más detallada sobre por qué no se entregó el paquete [7][17].

- *Packet Too Big.*

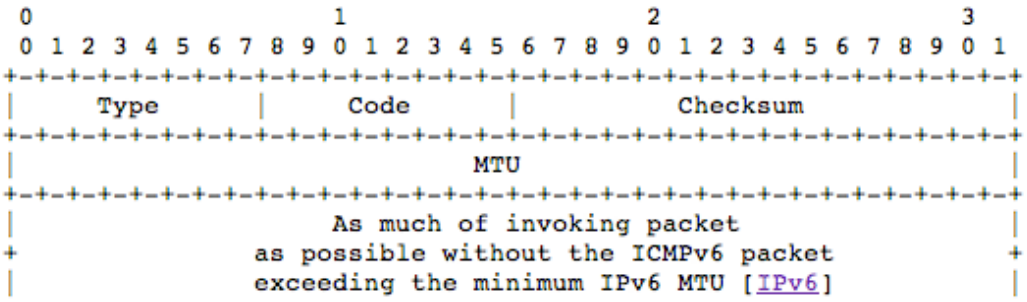

Figura 2.19. Cabecera Packet Too Big (RFC 4443)

Cuando un router no puede reenviar el paquete debido a que su tamaño es mayor que la MTU del enlace de salida, genera un mensaje *Packet Too Big* y lo envía al origen. El campo "*Type"* toma el valor 2, el campo "*Code"* no se utiliza y se pone a 0. El parámetro más importante en este caso es el que se indica en el campo "MTU". Cuando un host recibe este mensaje, debe informar a las capas superiores y establecer un nuevo tamaño máximo de paquete.

Este tipo de mensajes son la base de un nuevo mecanismo utilizado en IPv6 denominado *"Path MTU Discovery".* Debido a que IPv6 no soporta la fragmentación en nodos intermedios, aparece el concepto de MTU por trayecto. Este parámetro marca el tamaño máximo de paquete que se puede enviar en una comunicación entre dos host. El establecimiento de la MTU sigue el siguiente proceso: el host origen establece inicialmente como MTU la del enlace utilizado en el primer salto hacia el destino y si en el trayecto del paquete se excede la MTU de alguno de los enlaces, se envía un mensaje *Packet Too Big* al origen indicándole la MTU del enlace. Cuando el host origen recibe el mensaje, actualiza la MTU del trayecto y vuelve a enviar el paquete. Este proceso se realiza hasta que el paquete llegue al destino, disminuyendo su MTU cuanto sea necesario para enviar el paquete pero sin bajar de 1280 bytes, valor que se establece como MTU mínima en IPv6. Posteriormente el host puede enviar paquetes mayores para detectar si se ha incrementado la MTU del trayecto.

- *Time Exceeded*

Estos mensajes tienen el campo "*Type"* con valor 3 y pueden ser generados en dos casos. El primer caso ("*Code"* igual a '0') se da cuando un paquete ha agotado el número de saltos permitidos (el campo "*hop limit"* de la cabecera IPv6 llega a 0), evitando así que se generen bucles. El segundo caso ("*Code"* igual a '1'), indica que se ha excedido el tiempo permitido para realizar el reensamblado de un paquete fragmentado, debido a que uno de los fragmentos no ha llegado al destino.

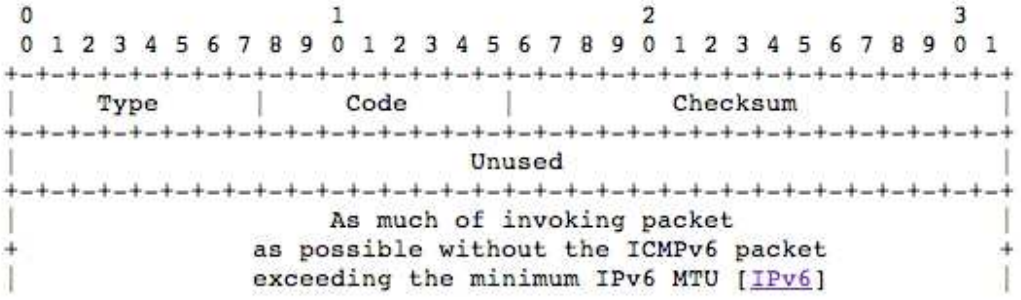

Figura 2.20. Cabecera Time Exceeded (RFC 4443)

- *Parameter Problem.*

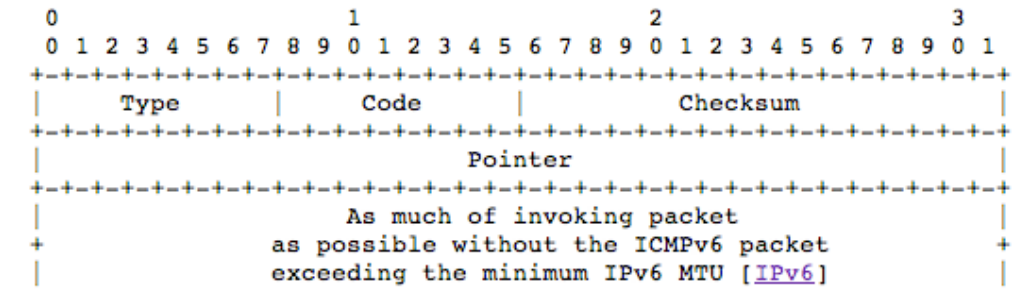

Figura 2.21. Cabecera Parameter Problem (RFC 4443)

Se genera cuando el paquete no ha podido ser procesado por un nodo por algún problema en la identificación de un campo en la cabecera IPv6 o en las cabeceras de extensión. El campo *"Type"* toma el valor 4, el campo *"Code"* puede tener valor 0 (campo erróneo encontrado en la cabecera), 1 (campo "*next header*" no reconocido) o 2 (cabecera de extensión no reconocida). *"Pointer"* indica la posición (byte) donde se ha producido el error.

Mensajes de información

- *Echo Request y Echo Reply.*

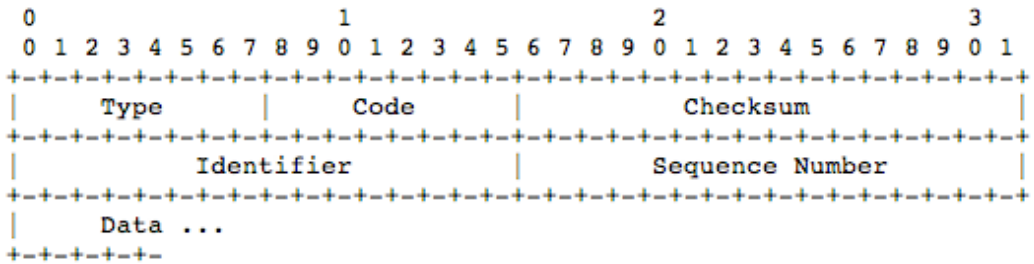

Figura 2.22. Cabecera Echo Request/Reply (RFC 4443)

Se utilizan para verificar la actividad de un nodo de internet. El nodo que necesita saber otro nodo está activo, envía un mensaje *Echo Request* y en caso de que el destinatario esté activo contestará con un mensaje *Echo Reply*. El campo *"Type"* tiene el valor 128 para *Echo Request* y 129 para *Echo Reply*. El campo "*Code*"

se establece a 0 y los campos "*Identifier*" y "*Sequence Number*" deben ser arbitrarios y se utilizan para marcar los mensajes y saber así qué respuesta corresponde a qué petición.

Neighbor Discovery

El protocolo *Neighbor Discovery* [19] representa una de las mejoras más significativas de IPv6. Su funcionalidad viene dada por un conjunto de mensajes ICMPv6 y realiza las siguientes funciones:

- Descubrimiento de routers en el enlace.
- Determinación de prefijos de red y otros parámetros de configuración.
- Protocolo SLAAC (*Stateless Address Autoconfiguration).*
- Determinación de direcciones de capa dos de los nodos que se encuentran en el mismo enlace.
- Detección de vecinos inalcanzables.
- Detección de direcciones duplicadas.
- Redirección.

Estas funciones se realizan a través de cinco mensajes, cuyas principales características se describen a continuación:

- *Router Solicitation* y *Router Advertisement*:

Los host generan mensajes *Router Solicitation* para pedir a los routers el envío de mensajes *Router Advertisement.*

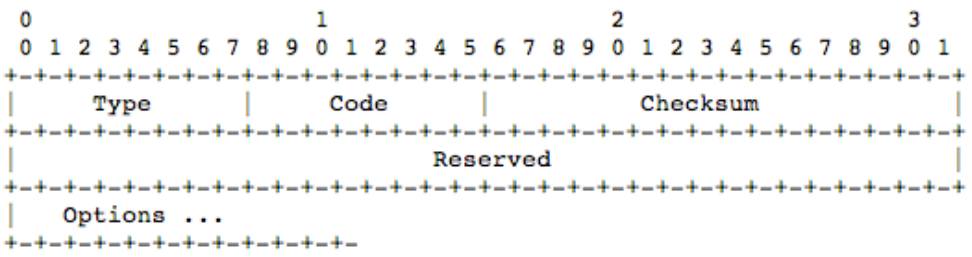

Figura 2.23. Cabecera Router Solicitation (RFC 4861)

El campo *"Type"* para *Router Solicitation* toma el valor 133 y el campo *"Code"* no se utiliza por lo que se pone a 0. El formato de opciones se describirá más

adelante pero la única válida en este mensaje es *"Source Link-layer Address"* que incluye la dirección de la capa de enlace del host que envía el mensaje. Además este tipo de mensaje suele incluir como dirección IPv6 destino la dirección *multicast* FF02::2, por lo que el mensaje se envía a todos los routers del enlace.

Los mensajes *Router Advertisement* se envían por los routers periódicamente, cuyo caso la dirección destino IPv6 será FF02::1 (todos los nodos del enlace), o en respuesta de un mensaje *Router Solicitation*, que tendrá como dirección IPv6 destino la dirección origen del mensaje *Router Solicitation.*

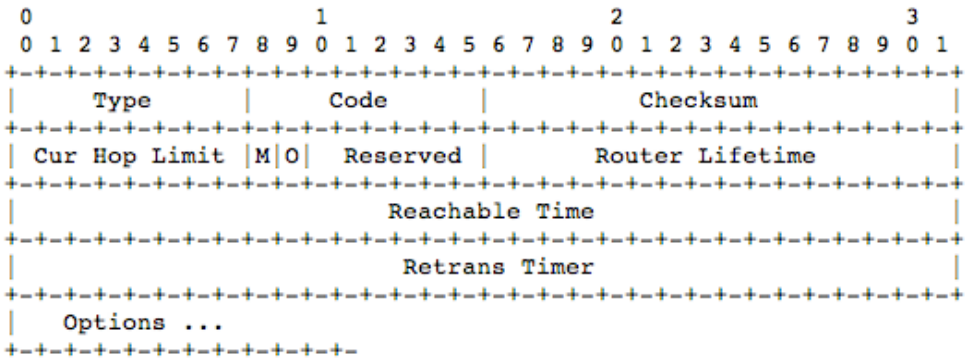

Figura 2.24. Cabecera Router Advertisement (RFC 4861)

En este caso, *"Type"* toma el valor 134 y *"Code"* el valor "0". El campo *"Cur Hop Limit"*indica a los host el valor que deben utilizar para el campo *"Hop Limit"* de la cabecera IPv6 (si el valor es 0 significa no especificado). Después tenemos el flag "M", que cuando toma el valor '1' indica que las direcciones están disponibles mediante DHCPv6. El flag "O", cuando su valor es '1', indica que existe información de configuración disponible mediante DHCPv6. El campo *"Router Lifetime"*indica el tiempo en milisegundos que el host debe mantener al router como router por defecto (en caso de que el host lo utilice como tal). *"Reachable Time"*marca el tiempo en milisegundos que un nodo puede asumir que un vecino es alcanzable después de recibir la confirmación de que lo es. *"Retrans Timer"*es el tiempo en milisegundos que debe pasar entre envíos de mensajes *Neighbor Solicitation*. Por último, se pueden incluir una serie de opciones de entre las siguientes:

- *Source Link-layer Address*: incluye la dirección de la capa de enlace de la interfaz desde la que se envía el mensaje.
- MTU: parámetro enviado en enlaces con valores MTU variables para que los host lo configuren.
- *Prefix Information*: indica los prefijos disponibles en el enlace o para ser utilizados en el protocolo SLAAC.

- *Neighbor Solicitation y Neighbor Advertisement:*

Un nodo envía un mensaje *Neighbor Solicitation* para preguntar a otro nodo por su dirección de capa de enlace, mientras a la vez anuncia su propia dirección de capa de enlace. Además, este mensaje se emplea en los mecanismos de detección de vecinos inalcanzables y detección de direcciones duplicadas. La dirección IPv6 destino será *multicast* cuando el nodo trate de averiguar las direcciones de otros nodos o *unicast* cuando trata de verificar si un nodo es alcanzable.

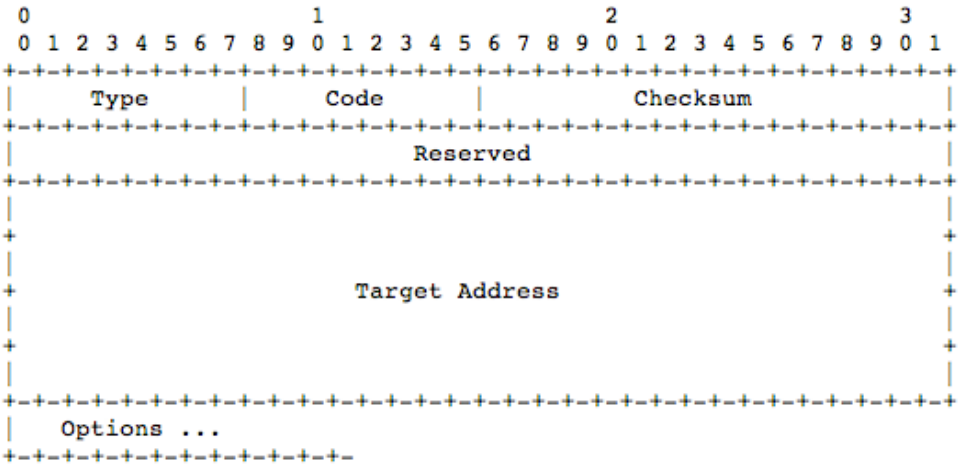

Figura 2.25. Cabecera Neighbor Solicitation (RFC 4861)

El campo "*Type*" se fija a 135, el campo *"Code"* a 0 (sin uso) y "*Target Address"* lleva la dirección IPv6 del nodo del que se quiere saber la dirección de la capa de enlace (no puede ser multicast). Por último, en opciones sólo se permite *"Source Link-layer Address*".

Los mensajes *Neighbor Advertisement* se generan en respuesta de *Neighbor Solicitation* o automáticamente si se desea propagar la información más rápido.

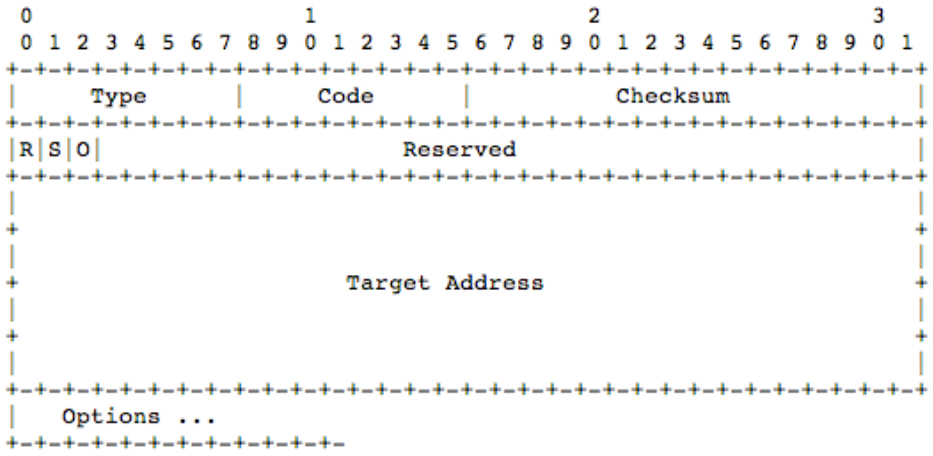

Figura 2.26. Cabecera Neighbor Advertisement (RFC 4861)

"Type" toma el valor 136 y *"Code"* el valor 0 (sin uso). El flag "R" indica si el mensaje lo ha enviado un router y se utiliza para detectar cuándo un router pasa a ser un host. "S" indica que el mensaje se ha originado en respuesta de un mensaje *Neighbor Solicitation* y "O" informa de que la información actual debe sobrescribir (si existe) la entrada caché correspondiente. El campo "Target Address" puede tener dos significados: si la información responde a un mensaje, este campo contiene la dirección IPv6 del nodo que envió la solicitud. Si la información se envía de forma automática, contiene la dirección IPv6 del nodo que ha cambiado de dirección de capa de enlace. Como opciones, únicamente puede contener la opción *"Target Link-layer Address"* que incluye la dirección de capa de enlace del nodo que genera el mensaje.

Es importante destacar el hecho de que mediante estos dos mensajes se realiza el proceso de resolución de direcciones de capa de enlace,algo que en IPv4 se llevaba a cabo mediante el protocolo ARP.

- *Redirect:*

Estos mensajes son enviados por los routers para informar a un host de un mejor "salto" hacia un destino.

El campo "*Type*" toma el valor 137, *"Code"* el valor 0 (sin uso). *"Target Address"* contiene la dirección IPv6 del nuevo "salto" al que debe dirigirse el host para llegar al destino especificado en el campo "*Destination Address".*Las opciones posibles en este tipo de mensajes son: "*Target Link-layer Address*" donde se incluye la dirección de capa de enlace del nodo que será el siguiente "salto" y *"Redirected Header"*.

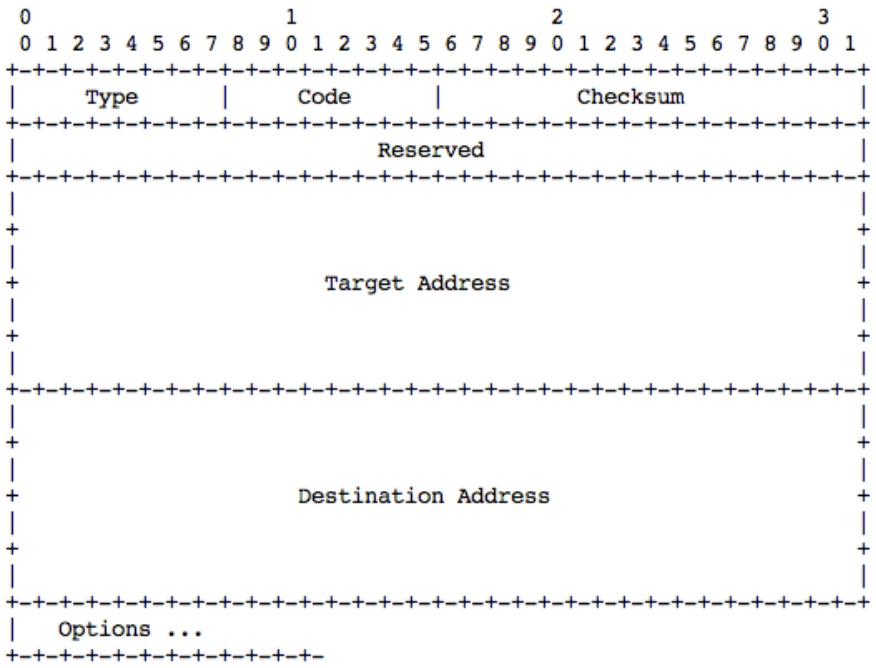

Figura 2.27. Cabecera Redirect (RFC 4861)

- Formato de opciones:

Las opciones que se pueden incluir en cada uno de los mensajes del protocolo *Neighbor Discovery* tienen formato TLV (el formato específico de cada opción puede consultarse en [17]):

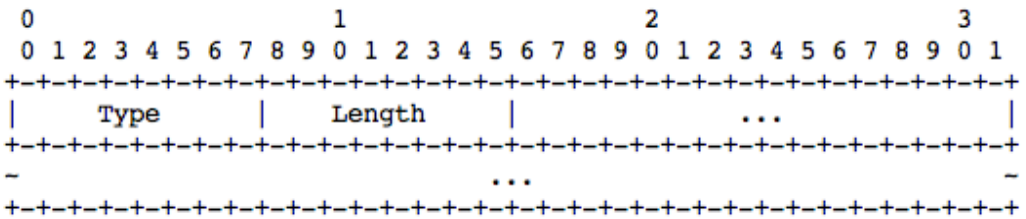

Figura 2.28. Opciones Neighbor Discovery (RFC 4861)

- Mecanismo de autoconfiguración de direcciones(SLAAC):

El mecanismo de autoconfiguración de direcciones es uno de las principales mejoras introducidas en el protocolo IPv6. Está especificado en [20], donde se estandariza el proceso mediante el cual un host puede obtener por si mismo las direcciones y parámetros de red necesarios para comunicarse en una red IPv6.

En un primer momento el host debe configurarse una dirección *link-local*. Para ello, primero se construye una dirección a partir de su dirección MAC, después comprueba que efectivamente esa dirección no está siendo utilizada por ningún otro host en la red local (mediante mensajes *Neighbor Solicitation* y *Neighbor Advertisement*). Si la dirección ya se encuentra asignada, hay dos opciones: detener el proceso y realizar una configuración manual de la dirección o construir otra dirección *link-local* a partir de otro identificador, volviendo a comprobar que dicha dirección no está ya asignada.

Una vez configurada la dirección *link-local* y comprobada que es única en el enlace, el host ya tiene conectividad mediante IPv6. El siguiente paso es obtener las direcciones de los routers presentes en la red (si es que hay). Para ello, el host puede esperar a recibir mensajes *Router Advertisement* periódicos o enviar un mensaje *Router Solicitation* a una dirección *multicast* y esperar la respuesta. En estos mensajes *Router Advertisement* se incluye información de los prefijos de la red mediante los host pueden configurarse direcciones globales (al igual que con las direcciones *link-local,* debe comprobarse que son únicas) y parámetros como MTU, tiempo de vida, etc.

En IPv4 era necesario un servidor DHCP para realizar este proceso, en IPv6 se elimina la dependencia de éste aunque el proceso también puede realizarse con DHCPv6 (*Stateful Autoconfiguration*).

2.1.5 DHCPv6

El protocolo DHCP (*Dynamic Host Configuration Protocol*) se utiliza para configuración de direcciones e información adicional de red en IPv4. Ahora en IPv6, el mecanismo de autoconfiguración elimina la necesidad de utilizar DHCP para la configuración de hosts. Aun así, la utilización de servidores DHCP sigue siendo de gran utilidad en muchos casos. Debido a ello, IPv6 viene acompañado de DHCPv6 [21], una versión adaptada del antiguo protocolo DHCP. La configuración de hosts mediante DHCPv6 se denomina *Stateful Autoconfiguration.* Algunos casos en los que puede ser conveniente utilizar DHCPv6 son: asignación dinámica de servidores DNS, configurar una dirección IPv6 que no provenga de la dirección MAC, utilización de un esquema específico de direcciones IPv6, etc. De cualquier forma, lo más conveniente es utilizar de forma simultánea la configuración mediante DHCPv6 y mediante el protocolo SLAAC. Por ejemplo, DHCPv6 para parámetros sobre servidores DNS y SLAAC para configuración de direcciones IPv6.

La configuración de host se realiza mediante el intercambio de mensajes entre el cliente y el servidor (o el *Relay Agent)*. Los mensajes que pueden intercambiar son:

- *Solicit (1):* utilizados por los clientes para localizar servidores DHCPv6.
- *Advertise (2):* enviado por el servidor en respuesta a un mensaje *Solicit,* indicando que está disponible para proporcionar un servicio DHCPv6.
- *Requets (3):* los envía el cliente para pedir parámetros de configuración a un determinado servidor.
- *Confirm (4):* enviado por el cliente a cualquier servidor DHCPv6 disponible para confirmar que la dirección asignada sigue siendo válida en el enlace.
- *Renew (5):* el cliente envía este mensaje al servidor que anteriormente le sirvió los parámetros de configuración y direcciones, para que extienda los tiempos de validez o actualizar dichos parámetros y direcciones.
- *Rebind (6):* enviado a cualquier servidor DHCPv6 por el cliente después de que este no reciba respuesta a un mensaje *Renew.* Se vuelve a pedir la extensión de tiempos de validez y/o actualización de parámetros y direcciones.

- *Reply (7):* el servidor envía mensajes *Reply* en respuesta a mensajes *Solicit, Request, Renew, Rebind* o *Confirm*. El mensaje puede contener parámetros de configuración y direcciones o una confirmación/denegación de la dirección del cliente, dependiendo del mensaje que llegara al servidor.
- *Release (8):* mediante este mensaje el cliente indica al servidor que a partir de ahora no utilizará las direcciones que le fueron asignadas.
- *Decline (9):* el cliente indica al servidor que las direcciones que éste le ha asignado están en uso en el enlace en el que el cliente está conectado.
- *Reconfigure (10):* enviado por el servidor para indicar al cliente que tiene información de configuración nueva o actualizada.
- *Information-Request (11):* de esta forma el cliente pide parámetros de configuración al servidor DCHPv6 sin que se asigne ninguna dirección al cliente.
- *Relay-Forw (12):* mensaje reenviado por un *Relay Agent* desde el clientehacia el servidor DHCPv6.
- *Relay-Repl (13):* enviado por el servido a un Relay Agent conteniendo el mensaje que éste debe reenviar al cliente.

En DHCPv6 todos los mensajes intercambiados entre cliente y servidor tienen el formato que se muestra en la Figura 2.29:

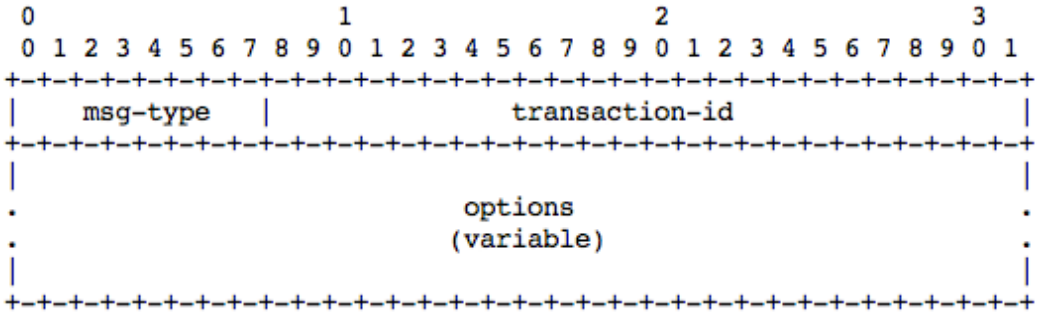

Figura 2.29. Cabecera DHCPv6 (RFC 3315)

El campo "*msg-type*" identifica el tipo de mensaje DHCPv6 (listados anteriormente) y *"transaction-id"* contiene el identificador para la operación en curso. El campo "options" lleva las opciones en formato TLV. Las principales opciones disponibles se resumen a continuación:

- *Client Identifier*: Usada para transmitir el identificador DUID (DHCP Unique Identifier) para validar al cliente frente al servidor.
- *Server Identifier*: Usada para transmitir el identificador DUID para validar al servidor frente al cliente.
- *Identity Association for Non-temporary Addresses (IA_NA):* indica el identificador de la asociación de direcciones, parámetros y tiempo hasta que el cliente vuelva a hacer una petición para extender el tiempo de validez.
- *Identity Association for Temporary Address (IA_TA):*indica el identificador, direcciones (temporales) y parámetros de la asociación.
- *Identity Association Address:* se utiliza para indicar las direcciones asociadas con IA_NA o IA_TA.
- *Option Request:* especifica una lista de opciones en un mensaje entre cliente y servidor.
- *Preference:*es enviada por el servidor para indicar al cliente que le seleccione como servidor DHCPv6.
- *Elapsed Time:* contiene el tiempo correspondiente al momento en que el cliente comenzó la transacción DHCPv6.
- *Relay Message:* contiene el mensaje original en un mensaje *Relay-Forw* o *Relay-Repl.*
- *Authentication:* contiene información para la autenticación de la identidad y de los mensajes DHCPv6.
- *Server Unicast:* la utiliza el servidor para indicar al cliente que puede usar dirección *unicast* para la comunicación.
- *Status Code:* indica el resultado de una transacción DHCPv6.
- *Rapid Commit:* utilizada para avisar que la asignación de dirección debe realizarse en dos mensajes.
- *User Class:* la utiliza el cliente para identificar el tipo de categoría del propio usuario o aplicación.

En la actualidad existen más opciones definidas en diferentes estándares, como por ejemplo opciones relacionadas con SIP [22], gestión de prefijos [23] o configuración DNS [24].

2.1.6 Cambios en otros protocolos

Capa de Transporte

En la capa de transporte el único cambio debido a la utilización del protocolo IPv6 se produce en la computación del *checksum*. Debido a que la cabecera IPv6 no incluye suma de comprobación, es importante que esta capa realice dicha función. El cambio se produce en los campos que se incluyen en dicha comprobación, ya que ahora hay que adaptarlo a las nuevas longitudes de las direcciones IPv6. Los campos que se tienen en cuenta a la hora de calcular el *checksum* son: dirección origen, dirección destino, longitud del paquete de capa superior y campo *"Next Header*". Por último, es importante resaltar que con IPv4 la inclusión de *checksum* en el protocolo UDP era opcional, con IPv6 es obligatorio.

DNS

Este protocolo se ha modificado mediante la inclusión de dos nuevos registros: AAAA y A6. El primero es una generalización del registro A del protocolo DNS para su uso con IPv6 por lo que no se produce un gran cambio [25]. El uso de registros A6 es más complejo [26], por lo que durante la implantación de IPv6 se recomienda utilizar registros AAAA, mientras la implantación de los registros A6 se pone a punto [27].

2.2 MALFORMACION DE PAQUETES

La malformación de paquetes o *packet crafting* se refiere a la técnica de generar paquetes alterando alguno de sus parámetros con el objetivo de analizar el comportamiento de los equipos que componen la red (*firewalls*, routers, pilas de protocolos…) frente a un comportamiento inesperado. Por ejemplo: un administrador de red puede utilizar esta técnica para comprobar si un *firewall* aplica correctamente los filtros que se le han configurado, un auditor de seguridad

puede utilizarla en busca de vulnerabilidades y un usuario malintencionado para obtener información privilegiada. En este proyecto utilizaremos el *packet crafting* con el objetivo de comprobar la estabilidad de diferentes implementaciones del protocolo IPv6.

Se puede dividir el proceso en varias fases:

- Creación del paquete.
- Alteración del paquete.
- Envío del paquete.
- Decodificación e interpretación del tráfico generado.

Aunque existen herramientas específicas para cada una de las partes, de forma general las tres primeras fases suelen realizarse con la misma herramienta, el generador de paquetes. Para la última, se utiliza un analizador de paquetes o analizador de tráfico.

2.2.1 Malformación de paquetes: herramientas

En este punto se comentan una serie de herramientas (todas gratuitas) comúnmente utilizadas en pruebas de seguridad:

Hping[1]

Se trata de un analizador/ensamblador de paquetes TCP/IP en modo consola. Soporta los protocolos TCP, UDP, ICMP e IP en bruto. Puede utilizarse para pruebas de firewall, escaneo de puertos, pruebas sobre protocolos, *fingerprinting,* etc.

Nping[2]

Nping es una herramienta de código abierto para la generación de paquetes de red, análisis de la respuesta y la medición de tiempo de respuesta. Nping puede generar paquetes de red para una amplia gama de protocolos, permitiendo a los

[1] www.hping.org
[2] nmap.org/nping

usuarios un control total sobre las cabeceras de los protocolos. Mientras Nping puede ser utilizado como una utilidad de ping simple para detectar hosts activos, también puede ser utilizado como un generador de paquetes para las pruebas de estabilidad de la pila, envenenamiento ARP o ataques de denegación de servicio. Incluye Nmap, una de las herramientas más utilizadas en las auditorías de seguridad de red, para la realización de escaneos de puertos.

Netdude[1]

Netdude es un *framework* para la inspección, análisis y manipulación de archivos de tcpdump. Cubre la necesidad de un conjunto de herramientas que permitan una fácil inspección, modificación y creación de archivos pcap/tcpdump.

Tcpreplay[2]

Tcpreplay es un conjunto de herramientas que permiten utilizar el tráfico capturado previamente en formato libpcap para la realización de pruebas sobre dispositivos de red. Permite clasificar el tráfico, como cliente o servidor, modificación de las cabeceras de las capas 2, 3 y 4 y reinyectarel tráfico de vuelta a la red.

Scapy[3]

Scapy es una utilidad escrita en Python que ofrece la funcionalidad de crear y manipular paquetes, escanear, funciones de *sniffer*, *fingerprinting*, creación gráficos, etc. Además, podemos crear utilidades escritas en Python usando Scapy. Puede trabajar mediante línea de comandos, es integrable en Python, programable, versátil y flexible.

Yersinia[4]

Yersinia es una herramienta diseñada para explotar algunas vulnerabilidades en los diferentes protocolos de red. Pretende ser un *framework*para el análisis de sistemas y redes. Implementa ataques sobre

[1] netdude.sourceforge.net
[2] tcpreplay.synfin.net
[3] www.secdev.org/projects/scapy/
[4] www.yersinia.net

protocolos de bajo nivel como STP (Spanning Tree Protocol), CDP (Cisco Discovery Protocol), DHCP y muchos más.

Nemesis[1]

Nemesis es una herramienta de *packet crafting* basada en línea de comandos. Némesis, es adecuado para realizar pruebas sobre sistemas de detección de intrusos, firewalls, pilas, etc.

Wireshark[2]

Aunque no realiza manipulación de paquetes, forma parte de la última etapa del *packet crafting*, la de decodificación e interpretación de la información. Se trata es el analizador de protocolos más utilizado. Permite la captura interactiva del tráfico que pasa por la red. Permite la visualización de cada paquete y de todos sus campos y valores. Además permite guardar y cargar las capturas.

2.2.2 *Fuzzing*

Las técnicas de *fuzzing* para pruebas de elementos de red se engloban dentro de las técnicas de *packet crafting*. Se conoce por *fuzzing* "las técnicas de caja negra donde el sistema bajo análisis es "atacado" con entradas y estructuras de datos inesperados a través de interfaces externas" [28]. Por lo tanto, el propósito es detectar errores en la implementación mediante el envío de datos o mensajes modificados y la inspección del comportamiento del sistema. También puede darse el caso de que el sistema falle al recibir datos en principio válidos pero que no han sido manejados correctamente en la implementación.

A menudo, los casos de prueba a tener en cuenta al analizar un sistema son demasiados por lo que es necesaria la introducción de un grado de automatización. En este sentido, podemos encontrar dos formas de automatizar las pruebas de *fuzzing*: automatización basada en mutación y automatización basada en modelos. El *fuzzing* basado en mutación toma muestras de entrada reales y crea los casos de prueba a través de la modificación aleatoria de estas entradas. En las pruebas de

[1] nemesis.sourceforge.net
[2] www.wireshark.org

fuzzing basadas en modelos se utilizan especificaciones sobre el sistema o protocolo bajo test para crear los casos de prueba.

Algunos parámetros importantes a la hora de realizar pruebas de *fuzzing* son [29]:

- Superficie de ataque analizada: expresa la capacidad del *fuzzer* de cubrir la mayor parte de los casos en los que el sistema puede ser vulnerable. La idea de las técnicas de *fuzzing* es analizar la mayor superficie de ataque posible.
- Profundidad: nivel que el *fuzzer* es capaz de alcanzar en el sistema analizado. Normalmente, ciertos niveles (ciertas partes de código) sólo se ejecutan en el sistema al introducir una secuencia de datos específica.
- Eficiencia: existen muchos factores que pueden contribuir a la eficiencia de las pruebas de *fuzzing*, pero esencialmente se puede considerar que un test de *fuzzing* es eficiente cuando se maximizan la superficie de ataque analizada y la profundidad, minimizando el coste computacional.

El modelo *Security Development Lifecycle* (SDL)[3] es utilizado por Microsoft con el objetivo de analizar la seguridad de los sistemas desarrollados y reducir los costes debido a mantenimiento de los productos software. En él se definen una serie de técnicas que se deben aplicar durante el desarrollo de software. En la fase de verificación, se incluye el *fuzzing* como parte de las pruebas a realizar y explicando que la aplicación de esta técnica es "una forma efectiva de encontrar fallos potenciales de seguridad".

2.2.3: *Fuzzing*: herramientas

Las técnicas de *fuzzing* no son exclusivas en protocolos de red, también son ampliamente utilizadas para la detección de fallos en el desarrollo de software. Dado que este proyecto está centrado en el análisis de un protocolo de red, únicamente comentaremos las herramientas que se centran en éstos:

Codenomicon[1] DEFENSICS

Se trata de un software comercial con soporte para más de 200 protocolos, capaz de utilizar casos de prueba pre-definidos. Además incluye una librería de vulnerabilidades y la habilidad de generar casos de prueba a través de capturas de red pcap. Incluye soporte para IPv6.

beStorm[2] y Mu Dynamics[3]

Ambos son herramientas comerciales con la capacidad de generación de tráfico mediante el uso de módulos de protocolos. Mu Dynamics está basado en XML, generando tráfico basado en las especificaciones XML. beStorm incluye (en su versión de pago) soporte para IPv6.

SPIKE[4]

Se trata de un *framework*que permite al usuario definir un protocolo mediante la utilización de plantillas. Además incluye una API para la generación de tráfico a partir de estas plantillas. Muy popular debido a la simplicidad con que representa internamente los datos.

Scapy

Aunque Scapy, como vimos antes, es una librería que nos permite la manipulación total de paquetes, también incluye métodos que nos permiten aplicar técnicas de *fuzzing* (básicas). La gran ventaja es que el usuario puede decidir en qué parte del protocolo aplicar el fuzzing por lo que es muy flexible. Sin embargo, es necesario que el usuario tenga ciertos conocimientos de programación.

[1] www.codenomicon.com
[2] www.beyondsecurity.com
[3] www.mudynamics.com
[4] www.immunitysec.com

Sulley[1]

Fuzzer evolucionado desarrollado en Python que toma características de los anteriores, fácil de utilizar y que permite el reenvío de paquetes a partir de archivos pcap.

The Art of Fuzzing[2]

Este *fuzzer* es el único de los vistos hasta ahora que está basado en mutación (a partir de capturas pcap). El proyecto fue abandonado en 2009.

2.3 PYTHON, SCAPY &QT

A continuación se presentan las principales tecnologías utilizadas para desarrollar la herramienta. Python se ha utilizado como lenguaje de programación, Scapy como librería para la manipulación, envió y tratamiento de los paquetes y Qt para la realización de la interfaz de usuario.

2.3.1 Python

Python es un lenguaje de scripting multiplataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, por lo que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

En los últimos años el lenguaje se ha hecho muy popular, gracias a varias razones como:

- La cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero.

[1] github.com/OpenRCE/sulley
[2] sourceforge.net/projects/taof/

- La sencillez y velocidad con la que se crean los programas.
- La cantidad de plataformas en las que podemos desarrollar, como Unix, Windows, OS/2, Mac, Amiga y otros.
- Además, Python es gratuito, incluso para propósitos empresariales.

Algunas de las características más relevantes de Python son:

- Propósito general: se pueden crear todo tipo de programas.
- Multiplataforma: hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.
- Interpretado: quiere decir que no se debe compilar el código antes de su ejecución. En realidad sí que se realiza una compilación, pero esta se realiza de manera transparente para el programador. En ciertos casos, cuando se ejecuta por primera vez un código, se producen unos bytecodes que se guardan en el sistema y que sirven para acelerar la compilación implícita que realiza el intérprete cada vez que se ejecuta el mismo código.
- Permite definición dinámica de variables.
- Gestión de memoria mediante conteo de referencias (transparente al usuario).
- Interactivo: Python dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que puede ayudarnos a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.
- Orientado a Objetos: la programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables.
- Funciones y librerías: dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen muchas librerías que podemos importar en los programas para tratar temas específicos.
- Sintaxis clara: por último, destacar que Python tiene una sintaxis muy visual, gracias a una notación identada. En muchos lenguajes, para separar

porciones de código, se utilizan elementos como las llaves o palabras clave. Para separar las porciones de código en Python se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.

2.3.2 Scapy

Scapy es un utilidad escrita en Python que puede ser utilizadas para la creación de *scripts* escritos en Python y utilizando Scapy como librería. Según su propia descripción es una "*aplicación interactiva capaz de manejar y manipular paquetes con multitud de protocolos. Permite capturar de diferentes interfaces de red, comprobar en tiempo real parámetros, crear nuevos protocolos, etc. Gracias a sus capacidades es posible realizar de forma manual y automática tareas como escaneos, tracerouting o pruebas de red. Además nos da la posibilidad de crear paquetes mal formados, inyectar tramas 802.11 inválidas o combinar técnicas de forma automática*" [30].

Todo esto nos proporciona una base potente para realizar pruebas sobre sistemas de red. Scapy permite monitorizar la red como lo hace Wireshark o Tcpdump, implementar pruebas de protocolo, añadir nuevos protocolos sobre los que implementa de forma nativa, implementar pruebas de *fuzzing* sobre los protocolos implementados, etc. Sin embargo, Scapy sacrifica una interfaz intuitiva por una capacidad increíble de ser utilizada como librería dentro de scripts escritos en Python. Como vimos en el apartado anterior, existen multitud de herramientas para usos específicos, sin embargo Scapy tiene un enfoque más general y flexible ya que permite la creación, alteración y envío del paquete. Además permite la decodificación y manipulación del tráfico recibido. Por lo tanto, vemos que Scapy abarca todas las fases o funciones del *Packet Crafting.*

Es difícil percibir la potencia que ofrece Scapy sin ver la forma en la que nos permite realizar las operaciones antes comentadas. Por ello, a continuación se hará un breve repaso sobre cómo utilizar esta herramienta en su aspecto más básico.

Lo primero que nos ofrece Scapy es una lista de comandos básicos informativos:

- ls(): nos muestra la lista de protocolos que podemos utilizar.
- lsc(): muestra la lista de comandos o funciones disponibles.
- conf: configuración de Scapy.
- help(func): muestra la ayuda de la función que pasemos como parámetro.

También podemos utilizar la función ls(), para obtener información sobre el protocolo que deseemos, como por ejemplo IPv6:

Figura 2.30. Ayuda IPv6 Scapy

En este caso Scapy nos muestra los campo que componen la cabecera del protocolo, el tipo de dato por el que se representa cada campo y el valor que se le asigna por defecto.Scapy provee protocolos comunes como Ethernet, IP, TCP, DNS, ICMP o DNS, pero también otros como Radius o STP.

El siguiente paso es ver la forma en la que se crean paquetes con Scapy. Para ello, basta con apilar las capas que queramos incluir en el paquete en el orden deseado, de cualquiera de las siguientes formas:

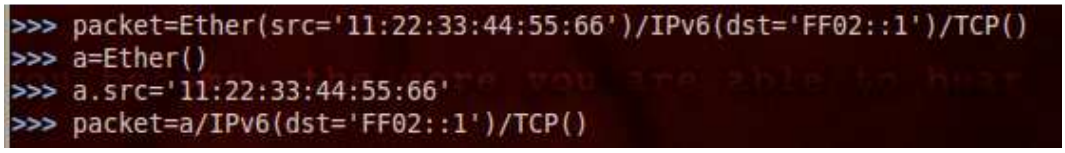

Figura 2.31. Creación de paquetes con Scapy

Como vemos, para crear una capa únicamente tenemos que poner su nombre y entre los paréntesis inicializar los campos que deseemos. Los campos que no se inicialicen tomarán su valor por defecto, excepto ciertos atributos como por ejemplo el campo *checksum* o el capo *next header* que se inicializarán con el valor correcto que corresponda en cada caso.

Una vez creado el paquete (o la capa) podemos ver su estructura con el comando show() o show2() (show2 nos muestra el valor que tomarán campos especiales como el *checksum* o *next header,* cuyo valor se asigna en función de otras capas*)*, que nos muestra el paquete en detalle:

Figura 2.32. Ejemplo show2()

Una vez creado el paquete, el siguiente paso es enviarlo. Para ello, Scapy provee dos funciones: sendp() y send(). La primera envía el paquete a nivel de capa de enlace y el segundo a nivel de red. Estas funciones permiten especificar la interfaz por la que enviar el paquete, el número de paquetes, etc.

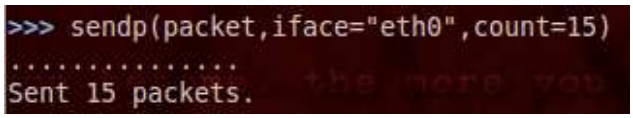

Figura 2.33. Envío de paquetes con Scapy

Otra opción muy útil que nos permite Scapy es escribir o leer archivos pcap mediante las funciones wrpcap() y rdpcap().

Además de enviar, existen funciones específicas para enviar un paquete y recibir su respuesta a nivel de capa de enlace (srp1, srp, srploop) o de capa de red (sr, sr1, srloop).Por ejemplo, esto podemos utilizarlo para enviar un "ping" y comprobar que se recibe la respuesta:

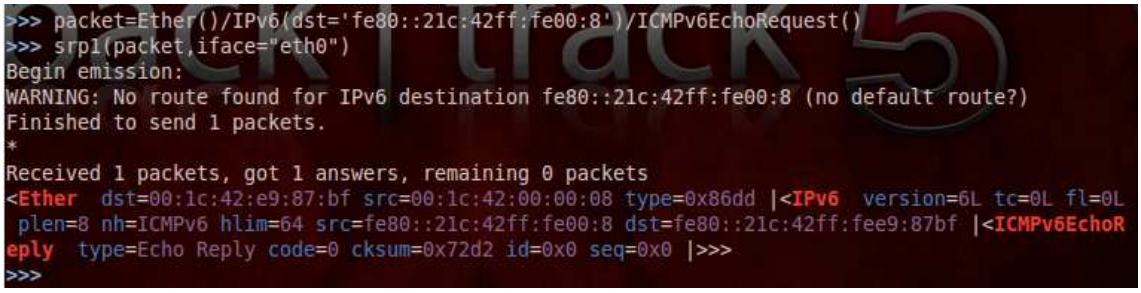

Figura 2.34. Ping utilizando srp1

Como vemos, scapy no sólo envía paquetes, si no que también es capaz de capturarlos y ofrecérnoslos para poder interactuar con ellos.Para la captura de paquetes sin necesidad de envío, tenemos la función sniff(), que nos devuelve una lista con los paquetes y a la cual podemos aplicar filtros para fijar la captura en un determinado tipo de paquetes. Por ejemplo, si queremos captar dos paquetes que contengan como capa de transporte TCP:

Figura 2.35. Ejemplo de captura de paquetes con Scapy

Esto es una muy breve introducción a lo que puede ofrecer Scapy, pero sólo con estas nociones ya disponemos de una herramienta muy poderosa para hacer todo tipo de pruebas de red.

2.3.3 Qt

En este apartado se resumirá brevemente la tecnología utilizada para el desarrollo de la interfaz gráfica de la herramienta Scapyst:Qt, PyQt y QtCreator.

Qt

Qt es una biblioteca multiplataforma utilizada generalmente para el desarrollo de aplicaciones con interfaz gráfica de usuario (GUI), aunque también para el desarrollo de programas sin interfaz. La compañía Trolltech[1] (actualmente QT Software, actualmente propiedad de Nokia) es la encargada de su desarrollo. Utiliza el lenguaje de programación C++ de forma nativa, aunque existen múltiples

[1]http://qt.nokia.com/

bindings (adaptación de una biblioteca para ser utilizada en un lenguaje de programación distinto de aquel en el que ha sido escrita) para otros lenguajes:

- Python (PyQt)
- Java (Qt Jambi)
- Ruby (QtRuby)
- Perl (PerlQt)
- PHP (PHP-Qt)

PyQt

PyQt es un conjunto de herramientas derivado de Qt y utilizado para la creación de aplicaciones GUI mediante Python. PyQt es Software Libre bajo licencia GNU/GPL y multiplataforma. Está estructurado como un conjunto de módulos:

- QtGui: manejo de componentes gráficos.
- QtCore: núcleo de Qt con funciones relacionadas con el tiempo, directorios, tipos de datos, etc.
- QtNetwork: conjunto de clases para trabajar en entornos de red.
- QtXml: permite el manejo de archivo XML.
- QtSvg: habilita la visualización de imágenes SVG.
- QtOpenGL: renderizado 2D y 3D utilizando OpenGL.
- QtSQL: permite trabajar con bases de datos.
- QtMultimedia: funcionalidad multimedia de bajo nivel.

QtCreator

Se trata de una potente herramienta de la que dispone Qt, la cual permite de forma simple, rápida y eficiente el diseño de interfaces gráficas de usuario. Permite construir un GUI de forma gráfica e intuitiva sin la necesidad de escribir líneas de código. Además, posee: un conjunto muy amplio de componentes para añadir en las interfaces, un inspector de objetos para la exploración de los componentes

utilizados, un editor de propiedades que nos permite modificar las características de los objetos presentes en la interfaz, etc.

Una vez creada la interfaz, la herramienta “pyuic” permite la conversión del código (originalmente en C++) a Python. Mediante este archivo ya podemos realizar modificaciones en la interfaz, gestión de eventos y demás funciones mediante Python.

CAPITULO 3

RIESGOS DE SEGURIDAD EN IPv6. *FUZZING* IPv6

RIESGOS DE SEGURIDAD EN IPv6. *FUZZING IPv6*

La implantación de una nueva tecnología siempre conlleva un riesgo de seguridad, por lo que es necesario un análisis exhaustivo con el objetivo de encontrar posibles vulnerabilidades. Con IPv6, esto es aún más necesario debido a que se trata del protocolo sobre el que se apoyan otros protocolos tan importantes como TCP, UDP, HTTP o SIP.

En este caso, la implantación del protocolo IPv6 puede llevar a la aparición de dos tipos de vulnerabilidades:

- Fallo en la especificación del protocolo: la vulnerabilidad viene dada por un aspecto de la especificación que puede ser explotado para fines maliciosos.
- Fallo en la implementación del protocolo: son fallos que aparecen debido a un mal desarrollo del software que permite la utilización del protocolo, comúnmente denominado "pila del protocolo".

Los fallos debidos a una mala implementación del protocolo son menos comunes debido a que este software sufre un exhaustivo proceso de pruebas.. Estos fallos pueden tener una gran repercusión y altos costes de reparación asociados cuando suceden en equipos con una gran carga de tráfico que conecten usuarios a la red, debido a que su reparación puede suponer la interrupción del servicio para los usuarios. Este tipo de vulnerabilidades son las que comúnmente se puede detectar mediante *fuzzing*.

La utilización de los mecanismos (en la especificación) del propio protocolo para la realización de ataques, significa que debe hacerse una revisión del estándar que define el protocolo, buscar la forma de eliminar los fallos (si es que la hay) y una vez subsanados los errores, proponer un nuevo estándar. Esto conlleva que las implementaciones del protocolo deben ser actualizadas de nuevo. Es por ello que los equipos suelen protegerse de muchas de estas vulnerabilidades mediante mecanismos de protección como *Firewalls*.

En el presente capítulo describiremos por un lado los nuevos riesgos que aparecen debido a la introducción del protocolo IPv6, después detallaremos ataques que han pueden realizarse en IPv6 y por último plantearemos el esquema para realizar *fuzzing* sobre la pila IPv6.

3.1 RIESGOS EN IPv6

Debido a que en el protocolo IPv6 se introducen cambios respecto a IPv4, aparecen nuevos aspectos a tratar en cuestión de seguridad. Con IPv6 seguimos teniendo muchos de los problemas de seguridad que existían con IPv4. Por ejemplo, aunque IPv6 permite la utilización de IPSEC, éste es raramente empleado debido a dos situaciones: en equipos de red el manejo de algoritmos y claves aumenta la carga de procesamiento de la CPU y en equipos de usuario la necesidad de manejar certificados para la distribución de claves unida al desconocimiento general de estos mecanismos. Sin el uso de IPSEC, pueden seguir empleándose técnicas como *IP spoofing* (suplantación de identidad) o *sniffing*. A continuación, hacemos una clasificación temporal de los riesgos potenciales de IPv6, describiendo los más destacados.

3.1.1 Riesgos a corto plazo

Una de las principales técnicas que ayudan a la implantación de IPv6 es el mecanismo de tunelación de tráfico IPv6 en IPv4, ya que hace posible la transición de un protocolo a otro. Sin embargo, este mecanismo puede resultar en una pérdida del control del tráfico en la red. En relación con esto, es muy común encontrar que las configuraciones de seguridad para IPv6 no están activadas por defecto, ignorándose los paquetes IPv6. De esta forma, IPv6 se convierte en una "puerta trasera" que puede ser utilizada por hackers o usuarios maliciosos. Por ello, es importante establecer políticas de seguridad para IPv6 de la misma forma que se hace para IPv4 y teniendo en cuenta los nuevos aspectos del nuevo protocolo.

Por otro lado, pueden surgir comportamientos imprevistos en los equipos de red en el tratamiento del tráfico IPv6. El rendimiento de los equipos se ha optimizado para IPv4 con el paso de los años, sin embargo, para IPv6 pueden surgir problemas de procesamiento de tráfico así como de rendimiento. Además la falta de formación de los administradores de red puede conllevar a que se apliquen configuraciones de seguridad no adecuadas.

Por último, el nuevo esquema de conectividad extremo a extremo que plantea IPv6 hace que desaparezca el uso generalizado de NATs, por lo que los equipos finales (y por tanto sus vulnerabilidades) quedan expuestos a ataques desde cualquier punto de la red (a no ser que se utilicen mecanismos como *firewalls)*.

3.1.2 Riesgos a medio plazo

Como riesgos a medio plazo se pueden identificar dos focos principales. El primero es la aparición de problemas con equipos y servicios de red que utilicen métodos de control basados en *Blacklists* o *Whitelists*. Por ejemplo, los mecanismos aplicados en IPv4 para el control de SPAM (bloque de direcciones IP) son difícilmente aplicables en IPv6 debido a que cada máquina puede autoconfigurarse su propia dirección. Una posible solución sería el filtrado por prefijo pero esto conlleva una serie de problemas que hace que su aplicación no sea inmediata.

El segundo punto es el problema para mantener la privacidad en accesos IPv6, debido a la desaparición de NATs. Por lo tanto, habrá que analizar la forma de ofrecer el nivel y servicios de privacidad de IPv4 en IPv6. Además, las funciones de movilidad que ofrece IPv6 abren la puerta a mecanismos de rastreo, por lo que sería necesario un análisis de implicaciones técnicas y legales.

3.1.3 Riesgos a largo plazo

Como principal riesgo a largo plazo se puede destacar la aparición de nuevas herramientas para ataques de tipo DDoS (*Distributed Denial of Service*) debido al aumento exponencial del número equipos electrónicos con dirección

IPv6. El protocolo IPv6 está ideado para que en un futuro todos los equipos que lo necesiten, pueda utilizar una dirección IPv6 pública para conectarse a Internet. El amplio direccionamiento disponible, y la eliminación de barreras para la conectividad punto a punto (NAT), facilitarán que se tienda a un nuevo esquema de conectividad más flexible.

3.2 ATAQUES EN IPv6

En las siguientes páginas se describen ataques a la seguridad en redes IPv6 que se deben tener en cuenta debido a las nuevas características del protocolo: cabeceras de extensión, ICMPv6 y DHCPv6. Para un mayor detalle de los ataques y mecanismos de mitigación de éstos puede consultarse [31], a continuación se muestran las principales implicaciones.

3.1.1 Escaneos en redes IPv6

De forma general, la primera fase de cualquier ataque es la identificación de la estructura de la red, esto es, nodos que están activos, puertos, servidores DHCP, etc. Con IPv4, escanear una red es bastante simple debido a que la mayoría de LANs utilizan una máscara de subred de 24 bits, con lo que no suele haber más de 254 hosts por red. Este reducido número de host hace posible un escaneo mediante TCP/UDP para identificar los nodos activos en la red.

Debido al aumento del espacio de direcciones en IPv6, no es posible emplear esta técnica debido a que llevaría mucho tiempo (2^{64} direcciones disponibles generalmente). Pero esto no quiere decir que no se realicen escaneos de red en IPv6, existen otras técnicas. Hay veces que los administradores de red asignan direcciones IPv6 secuencialmente a partir de la dirección de red. Esto hace que se reduzca el espacio efectivo de direcciones y facilita enormemente el escaneo de host, por lo que no es una práctica recomendada. Igualmente, se recomienda no utilizar ningún tipo de regla en la asignación de direcciones IPv6 (por ejemplo tomar los 8 últimos bits de la dirección IPv4). Vemos por lo tanto que

los principales riesgos provienen de configuraciones inseguras. De hecho hay estudios que señalan que un porcentaje bastante significativo de equipos no aplican una configuración adecuada con IPv6 [32].

Otro caso es en el que las direcciones se asignan a partir de la dirección de la capa de enlace (direcciones *link-local*). En este caso, el espacio de direcciones se ve reducido a 48 bits, y en el caso de que el atacante conozca el fabricante de la tarjeta Ethernet de la víctima, quedaría reducido a 24 bits (ya que las direcciones *link-local* se forman a partir de la dirección MAC y ésta incluye una parte característica que identifica al fabricante). Es por ello que se recomienda utilizar estas direcciones únicamente para la configuración inicial, utilizando la asignación mediante DHCPv6 posteriormente [33].

Además, IPv6 brinda una potente característica para el reconocimiento de nodos: direcciones *Multicast*. Por ejemplo, podría enviarse un paquete *Ping* a la dirección FF02::1 (todos los nodos de la red local) y observar qué equipos contestan a estos mensajes. Para realizar estos ataques, se supone que el atacante tiene acceso a la red local pero aún así, es recomendable no emitir respuestas a peticiones cuya dirección destino es una dirección *Multicast*.

3.1.2 Ataques derivados de Cabeceras de Extensión

La especificación del protocolo IPv6 indica un orden recomendado para las cabeceras de extensión y que ninguna (salvo alguna excepción) puede aparecer más de una vez en el paquete. Mediante la utilización de técnicas de *Packet Crafting* es fácil crear paquetes que no sigan estas recomendaciones. Por ejemplo, podrían crearse paquetes con una larga lista de cabeceras de extensión con el objetivo de crear un ataque DoS (*Denial of Service*), producir un aumento en el uso de la CPU, recursos, etc. Además, si el software no trata correctamente estos casos, el paquete puede producir un error del sistema o un comportamiento no deseado.

En los siguientes subapartados se comenta un poco más en detalle los riesgos que puede causar cada cabecera de extensión.

Cabecera *Hop-By-Hop* y *Destination Options*

Tanto la cabecera *Hop-By-Hop* como *Destination Options* deben aparecer una sola vez en el paquete, por lo que cada nodo debe comprobar que se cumple esta condición. Sin embargo, no hay restricción en el número ni en el orden de las opciones que pueden aparecer en la cabecera[1]. Esto puede ser causa de problemas debido a que la inclusión de un gran número de opciones puede aumentar la carga de procesamiento del nodo.

Las opciones de *Padding* en la cabecera de extensión se utilizan para la alineación de los paquetes en múltiplos de 1 byte, pero pueden utilizarse también con fines maliciosos. Por ejemplo, estas opciones pueden utilizarse para que contengan información en un *Covert Channel*. Según la especificación, un paquete que contenga la opción PadN no puede tener *payload.* Sin embargo, si enviamos un paquete de este tipo con cierta información en el *payload* la víctima puede que no procese dicho *payload* y no detecte el error. Otro proceso en la máquina de la víctima puede estar atento a esta información y por tanto disponer así de un canal de comunicación encubierto.

Otra de las opciones que tiene implicaciones en la seguridad es la opción *Router Alert*. Esta opción indica a los routers que deben analizar el contenido de la cabecera. Un usuario malicioso podría enviar un gran número de paquetes haciendo que el consumo de recursos aumente significativamente.

La forma de mitigar estos ataques es incluir reglas en el ACL (*Access Control List*) de los routers con el objetivo de no procesar estas opciones y limitar este tipo de paquetes.

Cabecera de *Routing*

La cabecera de *Routing* contiene dos tipos válidos hasta el momento: tipo 0 (RH0) y tipo 2 (RH2). La cabecera RH0 ha quedado obsoleta por cuestiones de seguridad pero es conveniente revisar este caso como ejemplo de las implicaciones que puede tener una mala especificación del protocolo.

[1] Aunque ambas cabeceras tienen propósitos distintos, su estructura es la misma.

Esta cabecera tiene una función similar a la que tenía la opción *source routing* en IPv4. Puede utilizarse para reencaminar el tráfico a través de uno o varios hosts intermedios. Esta funcionalidad puede utilizarse con varios fines maliciosos.

El primero es el de dirigir el tráfico por ciertos caminos con el objetivo de atravesar equipos que no detectan este tipo de tráfico haciendo posible un ataque *man-in-the-middle.*

El segundo uso que un atacante puede dar a estas cabeceras nace del hecho de que la especificación no incluía ninguna prohibición acerca de las direcciones por las que se quiere reenviar el tráfico. De esta forma, podría utilizarse para amplificar la carga de tráfico incluyendo las mismas direcciones varias veces en la cabecera de *Routing.* Esto a su vez podría llegar a provocar una denegación de servicio de la víctima (a través del sistema intermedio, normalmente ajeno al atacante). Este proceso se puede ver en la Figura 3.1:

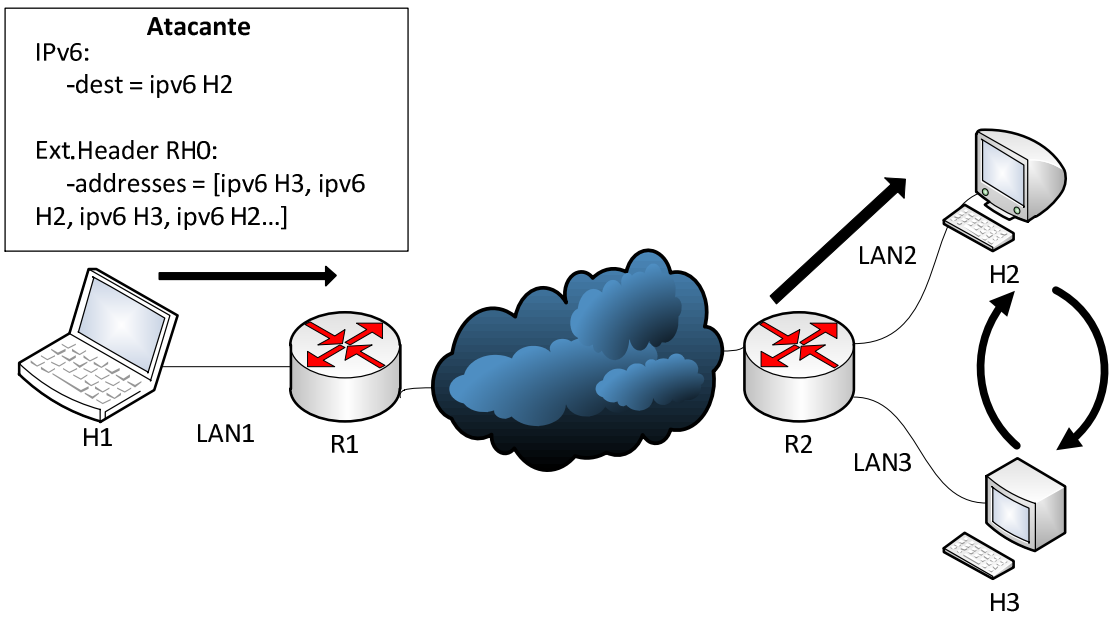

Figura 3.1. Ataque mediante cabecera RH0

Debido a estos inconvenientes, la cabecera de RH0 quedó obsoleta, por lo que cualquier ningún nodo debería procesar los paquetes que la contengan.

Cabecera *Fragmentation*

La MTU mínima que se define en IPv6 es de 1280 bytes por lo que los paquetes de menor tamaño pueden ser sospechosos. Sin embargo, existe un caso donde si se permite el uso de fragmentos de menor tamaño en IPv6: cuando es el último fragmento del paquete. En este caso un atacante podría utilizar estos fragmentos para ocultar la información maliciosa del ataque debido a que algunos *Firewalls* sólo comprueban la parte no fragmentable, o sólo comprueban el primer paquete. De esta forma, la fragmentación puede utilizarse para ofuscar información que no se desea que sea analizada por el *Firewall.*

3.1.3 Ataques derivados de ICMPv6

Con IPv4 el protocolo ICMP no es esencial para la comunicación, pero en IPv6 los mecanismos de configuración más importantes se realizan a través de ICMPv6. Por este motivo el protocolo ICMPv6 es un objetivo clave para los atacantes, especialmente en entornos de red local. Los ataques basados en este protocolo se pueden clasificar por el mecanismo que utilizan: SLAAC, *Neighbor Discovery,* detección de direcciones duplicadas y redirección.

***Satateless Address Autoconfiguration* (SLAAC)**

Como vimos en el Capítulo 2, mediante este mecanismo un host puede configurarse de forma autónoma para comunicarse en una red. Aunque esto ofrece grandes beneficios y es uno de las principales mejoras introducidas en IPv6, también conlleva importantes riesgos de seguridad que han derivado en la aparición de varios tipos de ataques.

Con este modo de funcionamiento, los routers anuncian su dirección mediante mensajes *Router Advertisement* (RA). Así, el router manda periódicamente mensajes RA para que los host puedan configurarse su propia dirección y tabla de *routing* a partir de la información que transporta. Dado que no se emplea ningún mecanismo de autenticación, un atacante podría capturar estos mensajes, inspeccionar la información y enviar él mismo los mensajes modificados

indicando una dirección de capa de enlace falsa. Esto causaría una denegación de servicio ya que el resto de hosts enviarían la información a una dirección falsa. En la Figura 3.2 se puede ver el esquema de este ataque, dónde el atacante anuncia la dirección MAC falsa (XX:XX:XX:XX:XX:Xe):

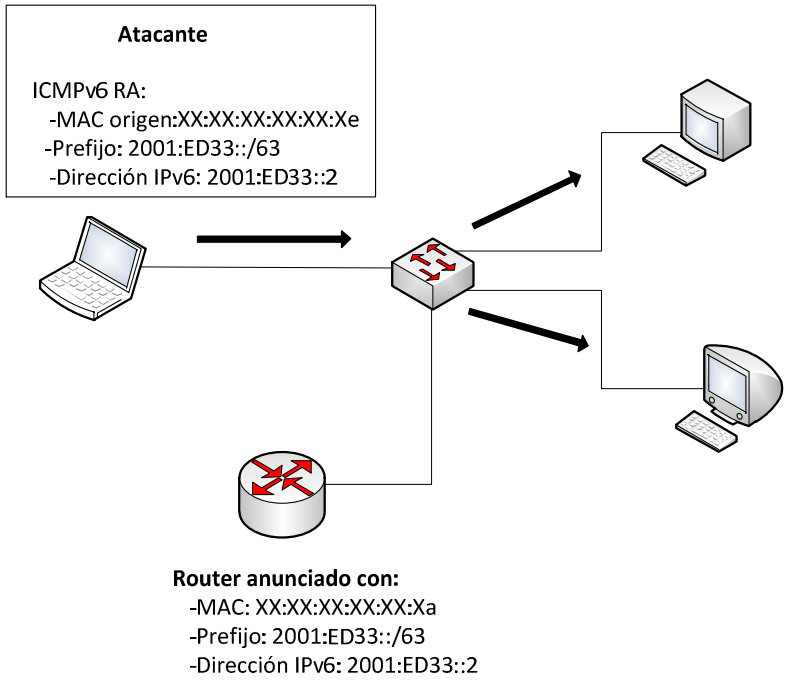

Figura 3.2. Ataque mediante RA falso

Otro ataque se trata de hacer *flooding* (inundación) de mensajes RA con direcciones aleatorias para que los equipos realicen una gran cantidad de operaciones necesarias para la configuración de direcciones. Esto hace aumentar el consumo de CPU al 100%, inutilizando por completo el equipo [34].

Neighbor Discovery

El protocolo ARP para descubrimiento de vecinos utilizado en IPv4 recae ahora en ICMPv6 a través de los mensajes *Neighbor Advertisement* (NA) y *Neighbor Solicitation* (NS), englobados en el protocolo *Neighbor Discovery* (NDP). Como ocurría con SLAAC, NDP tampoco utiliza autenticación por lo que es fácil suplantar la identidad de otro usuario. Un primer ataque sería similar al comentado sobre

SLAAC, pero en este caso enviando mensajes NA con la dirección IPv6 de la víctima y una dirección de capa de enlace falsa. De esta forma se crea una denegación de servicio sobre la víctima dado que los mensajes se enviarán a una dirección de capa de enlace que no corresponde con la real.

En un segundo ataque sobre NDP, un usuario malicioso podría hacer *spoofing* de mensajes NA y NS. Si el atacante obtuviera la dirección MAC del router, podría suplantar su identidad haciendo un ataque MITM (*Man in the Middle,* Figura 3.3) a cualquier usuario de la red local. Mediante este procedimiento el atacante podría realizar cualquier acción sobre el tráfico de la víctima.

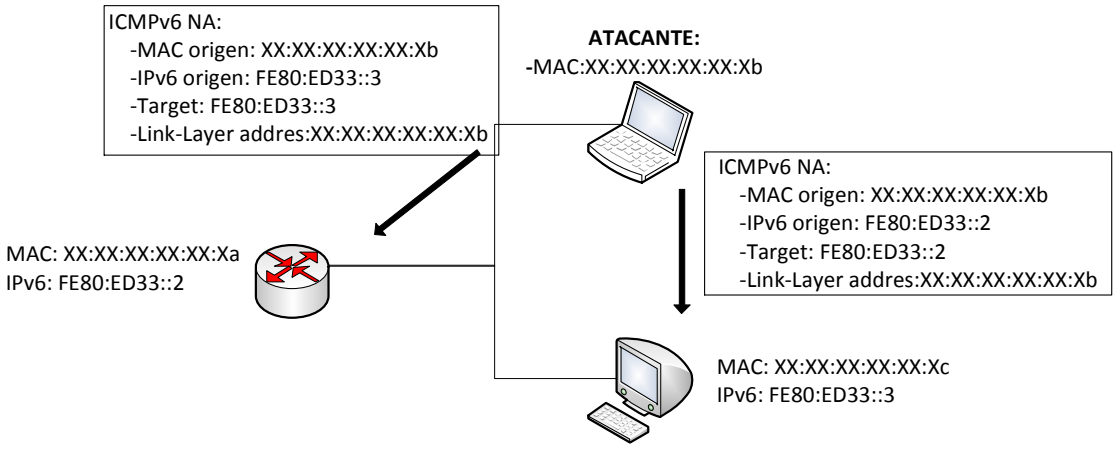

Figura 3.3. Ataque man in the middle

Otro ataque que podría llevarse a cabo mediante NDP, trata de explotar el alto espacio de direccionamiento que soporta IPv6. Una configuración de red típica permite tiene un prefijo de 64 bits, lo que nos da opción de configurar 2^{64} direcciones IPv6. Los routers deben contener una tabla actualizada con cada dirección configurada en la red, necesaria para reenviar el tráfico. Sin embargo, el tamaño de estas tablas no es suficiente para almacenar todo el rango de direcciones, lo que permite a un atacante enviar un *flooding* de mensajes NA con direcciones aleatorias de forma que se llene la tabla NDP. Así, ningún usuario un solo host ocuparía todos los recursos, denegando el acceso a la red a nuevos

usuarios (DoS). El comportamiento de los equipos ante esta situación anómala puede ser inesperado, pudiendo afectar a otras interfaces de red [35].

Detección de direcciones duplicadas

El mecanismo DAD (*Duplicate Address Detection*) forma parte del protocolo SLAAC y permite que los hosts comprueben si una dirección IPv6 autoconfigurada está siendo utilizada por otro host, mediante mensajes NS. De nuevo no se utiliza autenticación y un atacante podría modificar estos mensajes para crear una denegación de servicio sobre un usuario. Simplemente tendría que responder a los mensajes NS utilizados en el mecanismos DAD, indicando que la dirección está ocupada. Si esto se hace para todos los mensajes, el host no podría configurarse nunca una dirección IPv6.

Redirect

La redirección en IPv6 permite a un router indicar a un host de que existe una mejor ruta hacia otro host. El proceso es como sigue: un host HA tiene configurada una ruta por defecto hacia el router R2 y en la red existe el otro router R1. En algún momento R2 detecta que existe una mejor ruta de salida para HA que pasa por R1. Cuando HA necesita enviar un paquete, lo hace por R2 y este le informa de que la ahora la mejor ruta es a través de R1 mediante un mensaje ICMPv6 *redirect.* HA cambia su tabla de rutas.

En este caso si existe un mecanismo de protección: una copia del mensaje que causa la redirección se debe incluir en el mensaje ICMPv6 *redirect.* De esta forma se evita que un atacante envíe mensajes falsos. Sin embargo, esta restricción se puede bordear fácilmente. Si el atacante envía un *Ping* a la víctima, éste sabe que la respuesta va a ser un mensaje ICMPv6 *Echo Reply* Por lo tanto, al atacante le basta con incluir este mensaje en el mensaje ICMPv6 *Redirect* para falsearlo.

***Secure Neighbor Discovery* (SEND)**

Como acabamos de ver el protocolo NDP aparte de tener un destacado papel en IPv6, supone una gran superficie de ataque, siendo vulnerable a varios ataques. Es por esto que se han definido una serie de especificaciones que proporcionan mecanismos de seguridad al protocolo NDP [36].

3.1.4 Ataques derivados de DHCPv6

DHCPv6 se utiliza como complemento del mecanismo SLAAC para dotar no sólo de direcciones IPv6, sino de prefijos de red a los routers, servidores DNS y demás información. El ataque más evidente que puede sufrir este protocolo es un *flooding* mediante mensajes *SOLICIT* donde el atacante pretende agotar el *pool* de direcciones o prefijos que el servidor DHCPv6 tiene disponible. Esto puede causar una denegación de servicio por agotamiento del *pool* o por un elevado consumo de recursos del servidor DHCPv6.

3.3 *FUZZING* IPv6

En el Capítulo 2, denominamos *fuzzing* a un tipo de caso de pruebas que nos permite descubrir fallos de implementación software mediante el envío de entradas inesperadas y la monitorización del equipo bajo pruebas. En el caso de IPv6 (así como de otros nuevos protocolos), es de especial importancia detectar el mayor número de fallos posibles antes de su implantación real en los equipos. Esto es debido a que la dificultad de actualización de algunos de estos equipos hace que los fallos encontrados después de la implantación de los dispositivos sean críticos y muy costosos [37]. Una vez vista la importancia de la utilización de técnicas de *fuzzing* en el ciclo de desarrollo de cualquier producto software, nos centramos en cómo aplicar estas técnicas para el caso que nos ocupa: el protocolo IPv6.

La pruebas de *fuzzing* que se aplican al protocolo IPv6 se denominan de "caja negra", debido a que, en principio, no tenemos acceso al código fuente de la

pila del protocolo. En este tipo de pruebas sólo se cuenta con una especificación de cómo el sistema debería comportarse ante la recepción de datos de todo tipo (tanto válidos como inválidos), sin saber cómo el sistema llega hasta el estado final en que proporciona la salida.

Como ya se comentó, las pruebas de *fuzzing* pueden clasificarse en dos categorías: basadas en mutación y basadas en modelos. En nuestro caso las pruebas de *fuzzing* que se implementarán serán basadas en modelos, debido a que no se parte de tráfico previo capturado y a que se definirán un conjunto de casos de pruebas acotado, con el objetivo de evitar procesamiento innecesario el equipo bajo pruebas y tiempo de ejecución de las pruebas. Además podemos enmarcar nuestras pruebas de *fuzzing* dentro de las denominadas "basadas en bloques", en oposición a las pruebas totalmente aleatorias. El protocolo IPv6 está bien definido en bloques (campos) con distintas funcionalidades y cada uno con una longitud determinada. Es por ello que a la hora de determinar los valores que se utilizarán como entradas, resulta más eficiente tratar los datos como un conjunto de bloques y no como una serie de bits. Por ejemplo, si tratamos un mensaje ICMPv6 como un conjunto de bits y modificamos aleatoriamente algunos de ellos, es muy probable que el campo *"checksum"* resulte no ser válido con lo que el sistema bajo pruebas únicamente procesará este campo y descartará el paquete.

Una vez definidas las características que tendrán las técnicas de *fuzzing* que aplicaremos al protocolo IPv6, podemos desarrollar los pasos que forman el proceso de aplicación de dichas técnicas, cuyo diagrama se detalla en la Figura 3.4. Lo primero es realizar un estudio sobre el protocolo sobre el que se desean aplicar las técnicas de *fuzzing*, ya que el conocimiento del protocolo nos facilitará la generación de casos de pruebas efectivos y la supervisión del equipo que se está analizando. Una vez que se conoce el protocolo con el que vamos a trabajar, se definen los casos de prueba que se ejecutarán. Estos casos deben ser lo suficientemente amplios para analizar la mayor superficie de ataque posible, pero a la vez deben ser efectivos, evitando ejecutar casos de pruebas que a priori pueden considerarse innecesarios. Una vez que tenemos creados los casos de prueba, procedemos por un lado a guardar el estado inicial del sistema bajo pruebas y del sistema atacante (para poder replicar el ataque en el futuro) y por

otro lado procedemos a ejecutar los casos de prueba. De forma paralela, se supervisa el equipo en busca de posibles fallos o comportamientos no apropiados. En caso de que se encuentra un fallo, este debe ser analizado, detectando que acción lo produjo y comprobando las consecuencias que este fallo tiene en el equipo bajo pruebas.

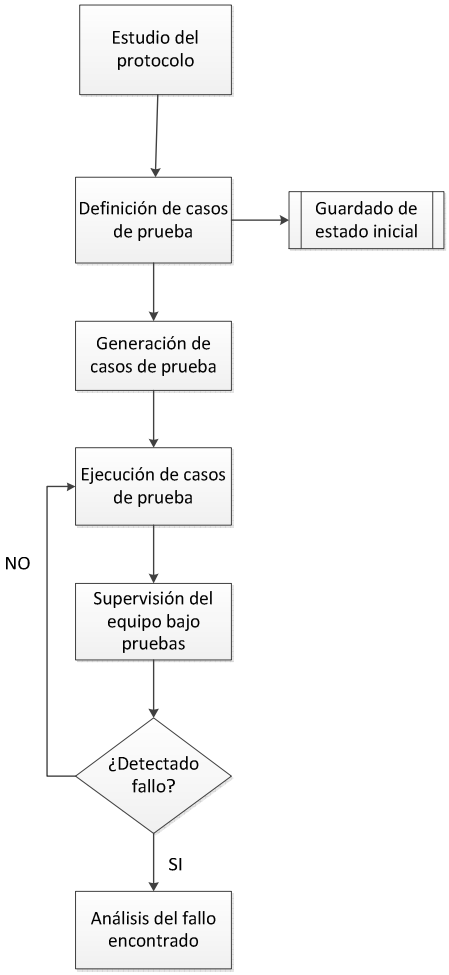

Figura 3.4. Diagrama de fuzzing

Por último, es importante recalcar dos aspectos. El primero es que la monitorización puede estar también automatizada si el propio *fuzzer* actúa como servidor, analizando las respuestas del equipo en pruebas. Sin embargo, para IPv6 vemos más recomendable realizar la supervisión del equipo de forma manual, debido a que cada equipo puede tener distintas implementaciones del protocolo o distintas políticas de filtrado, por lo que es difícil implementar una supervisión automática que sea general para todos los equipos. El segundo aspecto a comentar

es que es muy importante guardar el estado inicial del *fuzzer*. Aunque se definan unos casos de pruebas, las técnicas de *fuzzing* siempre incluyen (en menor o mayor medida) un cierto grado de aleatorización. Es por ello que si queremos replicar los test ejecutados, debemos almacenar el estado inicial del generador de números aleatorios (la "semilla") para posteriormente poder volver al mismo estado en caso de ser necesario.

3.3.1 Casos de prueba. IPv6 Ready.

El siguiente paso después del análisis del protocolo IPv6 es la definición de los casos de prueba. Es aquí donde se decide las estrategias que se van a aplicar para la detección eficiente de vulnerabilidades. Algunas de estas estrategias pueden ser: borrar campos, insertar campos, modificar el valor de uno o más campos, borrar capas, insertar capas, modificar el orden de las capas.

En nuestro caso, nosotros aplicaremos las técnicas de *fuzzing* a la cabecera IPv6, a la cabecera de extensión *Hop-By-Hop*, a la cabecera de extensión *Destination Options* y las opciones de fragmentación. Se aplicarán distintos métodos de *fuzzing* dependiendo de la parte que estemos tratando, ya que aunque modifiquemos el paquete, éste debe mantener cierta integridad para que el equipo lo procese. Por ejemplo, en la cabecera IPv6 únicamente modificaremos los algunos campos (de forma pseudo-aleatoria) ya que de otros no podemos (por ejemplo el *checksum*). Con las cabeceras de extensión podemos realizar más cambios: modificar campos, variar el número de opciones, variar el número de cabeceras que se incluyen, cambiar el orden de las cabeceras, etc. A continuación se muestran cada una de los métodos que se aplicarán para modificar el paquete IPv6:

- Cabecera IPv6
    - Modificación del valor de los campos
- Cabeceras de Extensión
    - Modificación del valor de los campos.
    - Modificación del número y el orden de las opciones.
    - Modificación del número y el orden de las cabeceras.

- Fragmentación:
    - Modificación del tamaño de los fragmentos.
    - Modificación del orden de envío de los fragmentos.
    - Modificación del valor de los campos de la cabecera de fragmentación.

Hay que resaltar que cada una de estas opciones no se aplican por separado, si no que se ejecutan de forma conjunta y pseudo-aleatoria. De esta forma, mediante la combinación de estas estrategias se busca forzar a la máquina a un estado de error. Como podemos ver, la mayor parte de las técnicas que aplicaremos consisten en modificar el valor de los campos. Esto podría llevarse a cabo mediante la utilización de todos los valores posibles que pudiera tomar el campo en cuestión. En este caso estaríamos hablando de un ataque por fuerza bruta, algo que en este contexto resulta ineficiente debido al amplio número de campos que tenemos y la gran longitud que pueden tener. Por ello, se hace necesario escoger un número acotado de valores para cada campo, limitando así la superficie de ataque pero aumentando enormemente la efectividad del *fuzzer.*

Por lo tanto, únicamente nos queda definir de qué forma modificaremos los valores u orden de los campos o cabeceras en el paquete, para lo que nos apoyaremos en [38]. En este documento, redactado por el "*IPv6 Forum", se* definen un conjunto de casos de prueba a aplicar sobre los equipos que implementen el protocolo IPv6. De esta forma, se define un marco de interoperabilidad entre equipo, a la vez que se certifica a los equipos que superen satisfactoriamente las pruebas. El documento está organizado en cinco secciones, donde cada una se centra una funcionalidad del protocolo IPv6. Nosotros utilizaremos la primera sección, donde se definen las pruebas para la cabecera IPv6, las cabeceras de extensión y la fragmentación. Así, nos basaremos en las pruebas definidas en este documento para crear los casos de pruebas a aplicar. Por ejemplo, el primer caso que se define en [38] es la comprobación de que el equipo bajo pruebas maneja correctamente el campo "*Version"* de la cabecera IPv6. Esto lo hace enviando un paquete con dicho campo modificado (valores de "*Version"* 0, 5 ,7 y 15) y observando el comportamiento del equipo. Este sería nuestro primer caso para el *fuzzer.*

Hay que destacar que ciertas pruebas se mantendrán y otras se extenderán, dependiendo del cada campo en particular. Además, como ya se comentó, las pruebas no se realizarán por separado y de forma secuencial, si no que se introducirá un parámetro aleatorio que defina el número de modificaciones que se introducen en el paquete y la forma de introducirlas. De esta forma, por un lado evitamos el desperdicio de recursos que supone realizar los casos de pruebas de forma totalmente aleatoria y por otro ganamos en tiempo automatizando las pruebas y analizado una extensa superficie de ataque.

CAPITULO 4

HERRAMIENTA SCAPYST

HERRAMIENTA SCAPYST

Hasta ahora se ha visto el apartado teórico sobre el nuevo protocolo IPv6 y los riesgos que conlleva su implantación. Este capítulo tiene una orientación más práctica ya que en él se presenta la herramienta Scapyst. Esta utilidad nos permite analizar riesgos de seguridad mediante el uso de las técnicas y descripciones vistas en los apartados anteriores: *packet crafting*, ataques IPv6 y *fuzzing.* La herramienta se sirve de la librería Scapy para crear una interfaz intuitiva dónde crear y enviar paquetes IPv6 a medida, con libertad para modificar cualquier parámetro. Además ofrece distintas funcionalidades como la posibilidad de ejecutar ataques, hacer *fuzzing,* manejar sesiones PPP o ejecutar un flujo de paquetes en modo servidor. Todo esto se detalla en los apartados siguientes, donde se describen cada una de las posibilidades que ofrece la herramienta Scapyst. En el Capítulo 6 se comentan posibles funcionalidades adicionales que no han entrado como parte de este proyecto y que se podrían añadir a la herramienta en el futuro.

4.1 PANTALLA GENERAL

La pantalla general engloba todos los menús de los que está compuesta la herramienta e incluye las funcionalidades de creación y envío de paquetes, así como de visualización de la información. Como se puede ver en la Figura 4.1, la interfaz consta de tres bloques principales: uno para la selección de las capas que formarán el paquete, otro para su visualización y modificación de los campos de cada capa y otro con las opciones de envío. Además, consta de un menú donde podemos activar el modo servidor, ejecutar ataques o *fuzzing*, así como abrir otras herramientas integradas. A continuación se detallan las funcionalidades que ofrece la pantalla general, describiendo el modo de operación y mostrando la forma de uso. A su vez, se mostrarán ejemplos para poner de manifiesto con qué facilidad se pueden crear paquetes IPv6 personalizados, algo esencial para el uso de Scapyst para personas no especializadas.

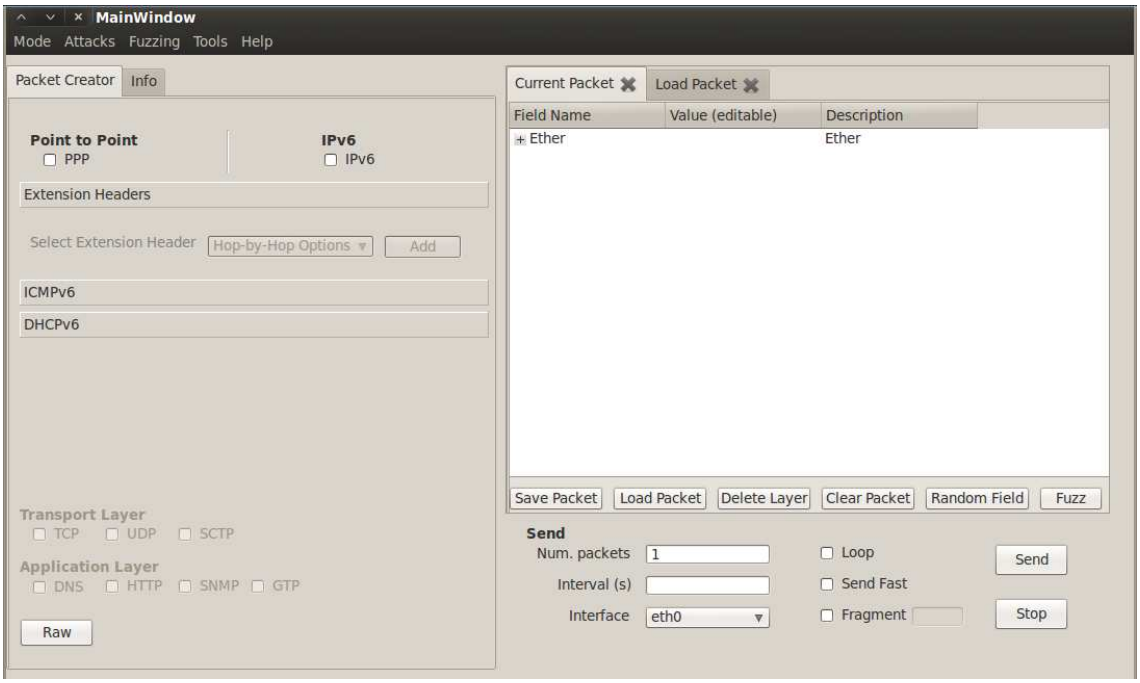

Figura 4.1. Pantalla General Scapyst

4.1.1 Construcción del paquete

La principal funcionalidad que nos ofrece Scapyst es la creación de paquetes IPv6 a medida, de forma gráfica e intuitiva. Como se comentó anteriormente, esto se realiza haciendo uso de dos paneles de la pantalla general: el de selección de capas y el de modificación de los campos de dichas capas.

Las capas a añadir en el paquete se eligen simplemente seleccionando los *checkbox* correspondientes a las capas que deseemos. Hay que tener en cuenta que existen ciertas restricciones de cara a la construcción del paquete. Por ejemplo, si se añade un mensaje ICMPv6 no podemos añadir capa de transporte o aplicación. En la Figura 4.2 se muestra un ejemplo de cómo se formaría un mensaje ICMPv6 *Echo Request*:

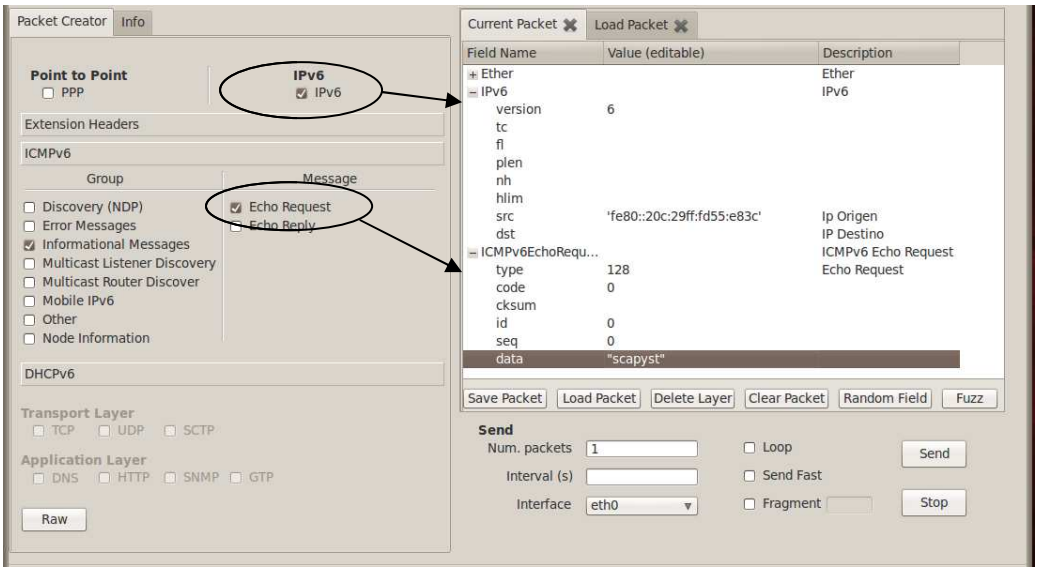

Figura 4.2. Creación de Mensaje ICMv6

Las capas de las que dispone Scapyst para formar el paquete son:

- Ethernet: se añade por defecto y no se puede borrar, todos los paquetes por lo tanto la contienen.
- PPP: esta capa es útil cuando se trabaja con sesiones PPP. Scapyst dispone además de una herramienta para detectar sesiones PPP que más adelante comentaremos.
- IPv6: añade la cabecera IPv6 al paquete. Cuando se añade, se habilitan las capas de nivel superior.
- Cabeceras de Extensión: no se han puesto restricciones a la hora de añadir cabeceras de extensión, ni en el número ni en el orden, debido a que para realizar pruebas de seguridad resulta de utilidad no poner demasiadas restricciones.

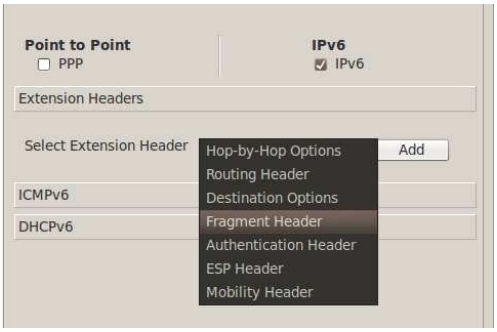

Figura 4.3. Selección de Cabeceras de Extensión

- ICMPv6: contiene los distintos mensajes ICMPv6, clasificados en 8 grupos: NDP, Error, Información, MLD, MRD, Mobile IPv6, NI y otros. Seleccionando uno de los grupos, aparecen los mensajes disponibles en dicho grupo.
- DHCPv6: cada uno de los posibles mensajes DHCPv6 (Figura 4.4).

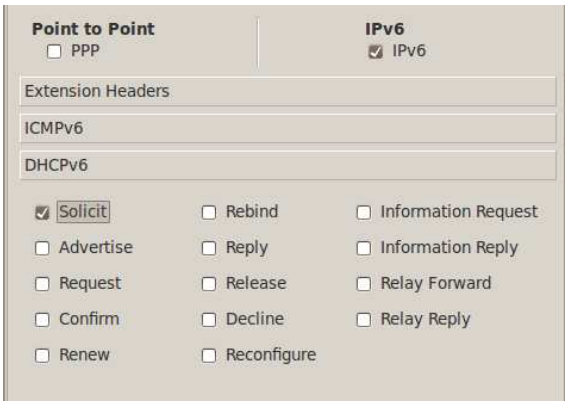

Figura 4.4. Mensajes DHCPv6 disponibles

- Capa de Transporte: en la capa de transporte se puede elegir entra TCP, UDP o SCTP.
- Capa de Aplicación: los protocolos disponibles en la capa de aplicación son HTTP, DNS, SNMP y GTP.
- *Payload*: mediante la opción "*Raw*" aparece una ventana secundaria donde podemos añadir un payload o datos en crudo como se muestra a continuación:

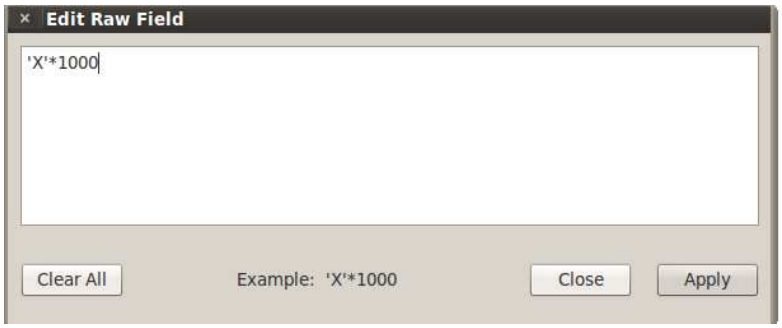

Figura 4.5. Payload

Una vez visto cómo se forma la estructura del paquete con Scapyst, pasamos a ver cómo podemos modificar los valores de los campos de cada una de las capas que forman el paquete IPv6. Al añadir las capas éstas se pueden visualizar en el

panel "*Current Packet*", dónde encontramos tres columnas. La primera muestra el nombre de la capa o el nombre del campo, la segunda el valor del campo y la tercera una breve descripción sobre el campo de la capa. Además es aquí donde se pueden modificar los valores como se muestra en la Figura 4.6:

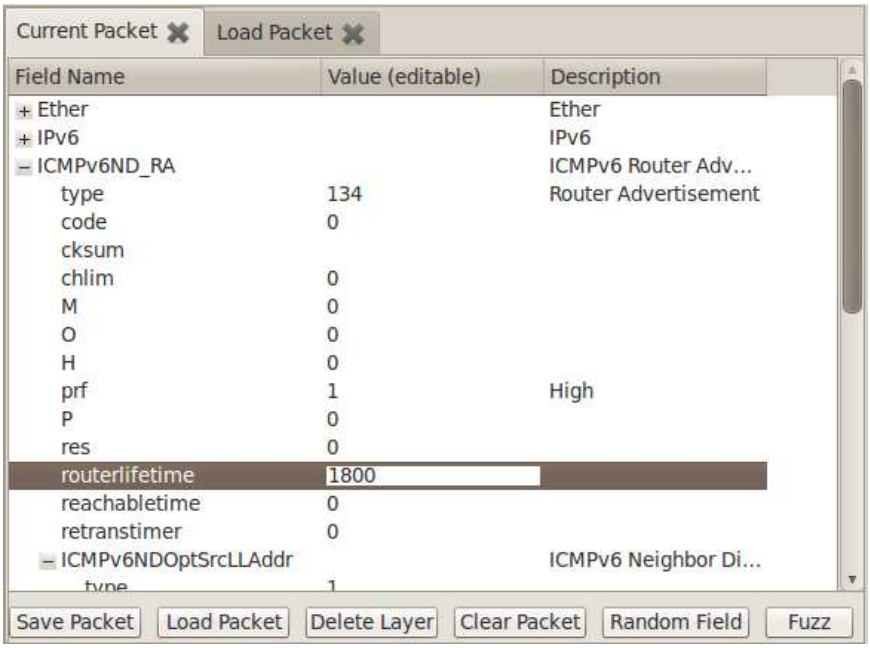

Figura 4.6. Modificación del valor del campo "RouterLifeTime"

Como se puede observar en la figura, una capa puede estar formada por subcapas, que no son más que campos agrupados. En este caso, el mensaje *Neighbor Discovery Router Advertisement* contiene el subgrupo *Source Link Layer Address.* También podemos ver que el valor de los campos puede estar en blanco, lo que quiere decir que el valor que tomará es el que pone Scapy por defecto (campos como el "*checksum"* o "*next header"* se computan correctamente). Aparte de la modificación básica de campos, existen ciertos casos donde la mecánica es algo diferente. Un ejemplo se puede encontrar en varios campos de la capa DNS (Figura 4.7) o sobre las opciones de la cabecera de extensión *Hop-By-Hop* (Figura 4.8), donde el valor se establece mediante un nuevo formulario que se muestra al pulsar sobre el valor del campo en cuestión:

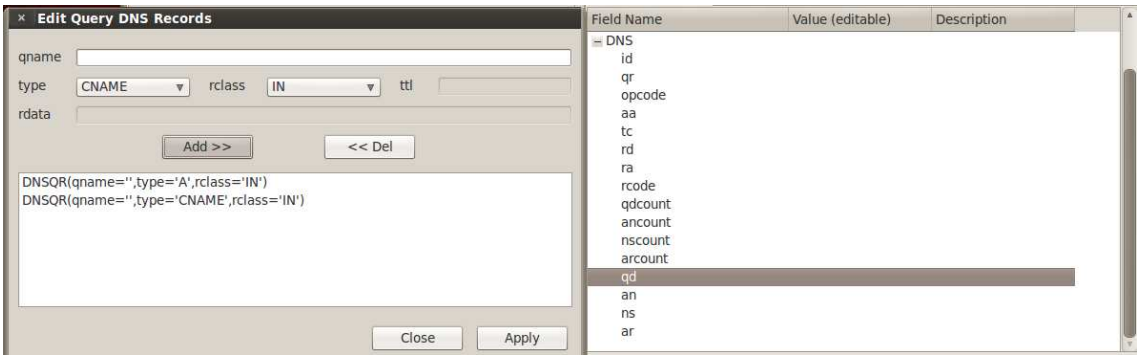

Figura 4.7. Modificación de Query's DNS

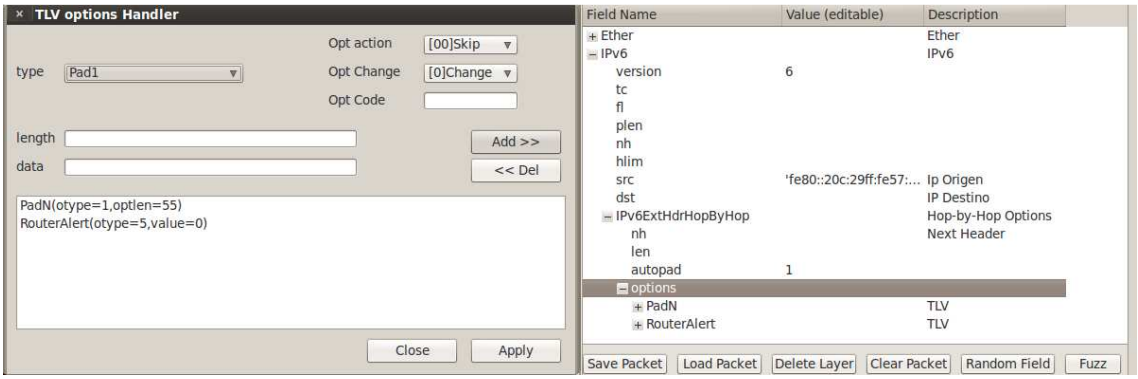

Figura 4.8. Modificación de opciones Hop-By-Hop

Otra opción para modificar el valor de los campos es utilizar algunos de los botones disponibles en la parte inferior del panel "*Current Packet*" (ver Figura 4.2). Podemos borrar capas añadidas mediante el botón "*Delete Layer*" podemos borrar una capa (también deseleccionando el *checkbox* de la capa), mediante "*Clear Packet*" eliminamos todas las capas (excepto *Ethernet*) y mediante los botones "*Random Field*" (solo habilitado para direcciones MAC e IP) y "*Fuzz*" se permite poner un valor aleatorio o realizar *fuzzing* (sobre los campos con del capa con valor en blanco) respectivamente:

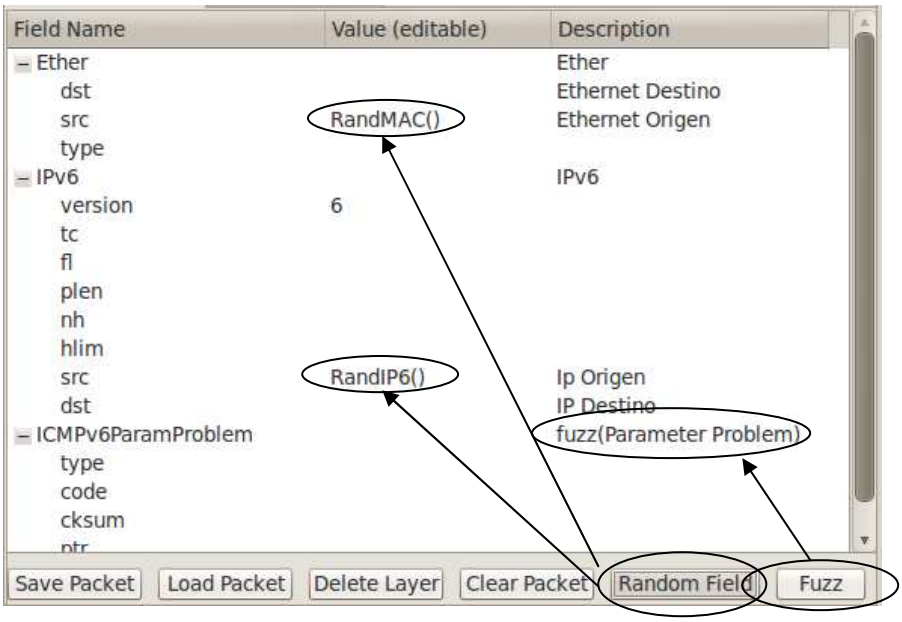

Figura 4.9. Opciones Random Field y Fuzz

Las otras dos opciones de la lista sirven para guardar o cargar paquetes y las comentaremos más adelante.

4.1.2 Opciones de envío

Cuando el paquete está creado, éste está listo para ser enviado. La opciones de envío se encuentran debajo del panel "*Current Packet*" y nos permiten elegir el número de paquetes a enviar, el intervalo de tiempo entre el envío de paquetes, la interfaz por la que queremos enviarlo, enviar en modo *loop*, fragmentar o enviar en modo rápido. Todo esto se puede observar a continuación:

Figura 4.10. Opciones de envío del paquete

Existen dos restricciones a la hora de utilizar estas opciones. La primera es que la opción de envío rápido no permite el envío de varios paquetes (se realiza a través de la herramienta tcpreplay). La segunda es que al marcar la opción

fragment la cabecera de extensión de fragmentación debe estar presente, por lo que si no se encuentra añadida al paquete se mostrará un mensaje indicándolo.

Por último, en las Figura 4.11 y 4.12 se muestra un ejemplo de envío de 5 paquetes (ICMPv6 *Echo Request"*) con un intervalo de 1 segundo y a una dirección destino aleatoria y su captura en Wireshark (obviamente no se recibe respuesta porque las direcciones origen y destino no son adecuadas):

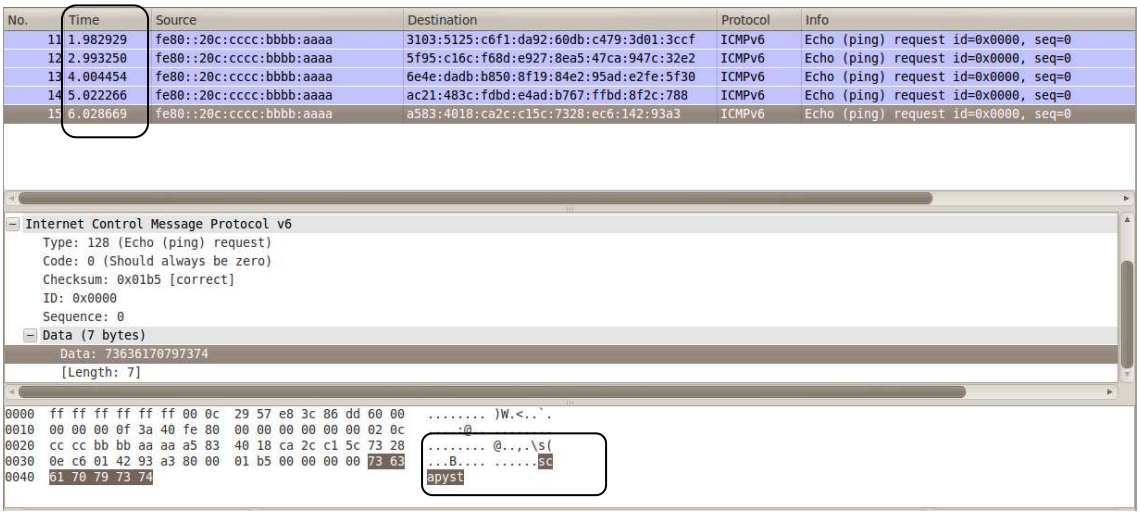

Figura 4.11. Ejemplo de envío (captura)

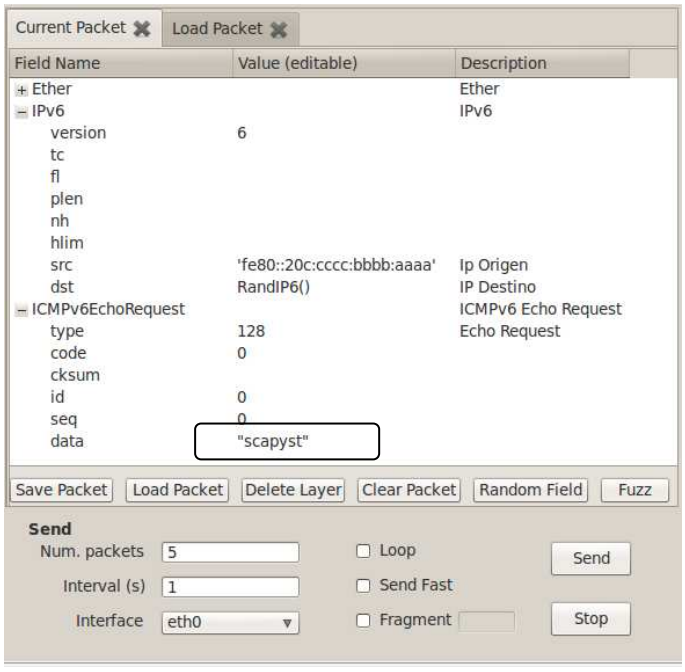

Figura 4.12. Ejemplo de envío (Scapyst)

4.1.3 Información

La pestaña situada al lado de la pestaña *"Packet Creator"* muestra información de varios tipos: muestra el comando Scapy que crearía el paquete actual, muestra información de Debug cuando se carga o envía un paquete y por último enseña información sobre las interfaces de red disponibles y otros datos utilizados. Todo ello se puede observar, para el ejemplo del apartado anterior, en la Figura 4.13:

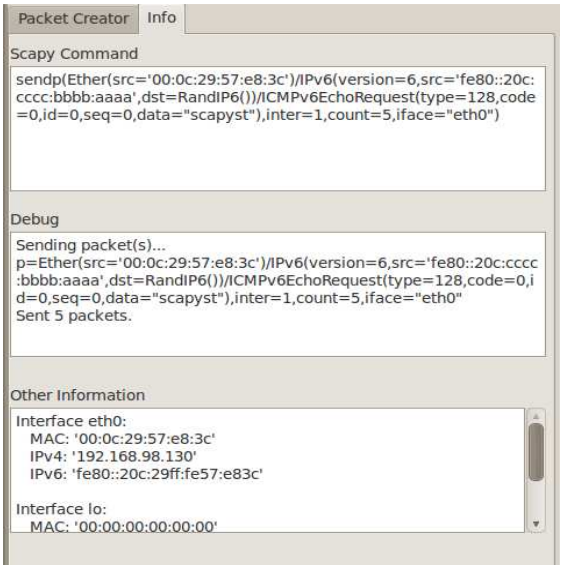

Figura 4.13. Cuadros de información

4.1.4 Guardar y Cargar Paquetes

La última funcionalidad que ofrece la herramienta Scapyst en la pantalla general es la de guardar y cargar paquetes. Los paquetes se pueden guardar mediante el botón *Save Packet* y se guardan en formato "pcap" de forma que se pueden utilizar posteriormente con herramientas como Snort, Wireshark, Nmap. De la misma, Scapyst permite cargar paquetes en este formato mediante la opción *Load Packet* o mediante el panel situado en la pestaña "*Load Packet"* situada al lado de la pestaña "*Current Packet*". Como se muestra en la Figura 52 Mediante la segunda opción tenemos la posibilidad de visualizar los paquetes contenidos en un directorio y realizar una pre-visualización para posteriormente cargar el paquete que se desee:

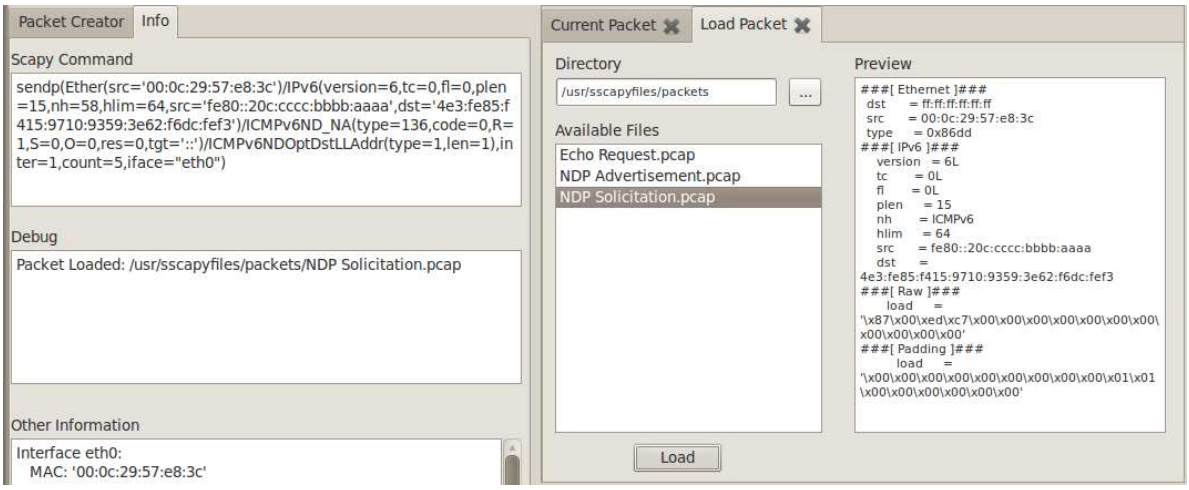

Figura 4.14. Carga de paquetes

4.2 MODO SERVIDOR

Fuera de la pantalla general podemos encontrar una barra de menú con distintas opciones. La primera de ellas es el modo servidor. Activando esta opción se pueden crear flujos de paquetes, no sólo a enviar sino que se pueden crear un paquete y esperar a que se reciba para después actuar de uno u otro modo. Por lo tanto el modo servidor ofrece dos interesantes modos de funcionamiento:

- Envío de una lista de paquetes: Nos permite construir uno a uno los paquetes y enviarlos secuencialmente e incluyendo opciones como *delays,* bucles, definición de puntos de control, etc.
- Interactuar con las máquinas destino: mediante la definición de paquetes de entrada, el programa puede esperar a recibir estos paquetes para seguir enviando. Incluso se pueden utilizar en los paquetes enviados valores de campos de los paquetes recibidos mediante la definición de variables.

El modo servidor se habilita cuando se pulsa en la opción *Server Mode* de la barra de menú. Al activarse, la pantalla general se amplía y la parte derecha de la interfaz del programa aparece como se muestra en la Figura 4.15:

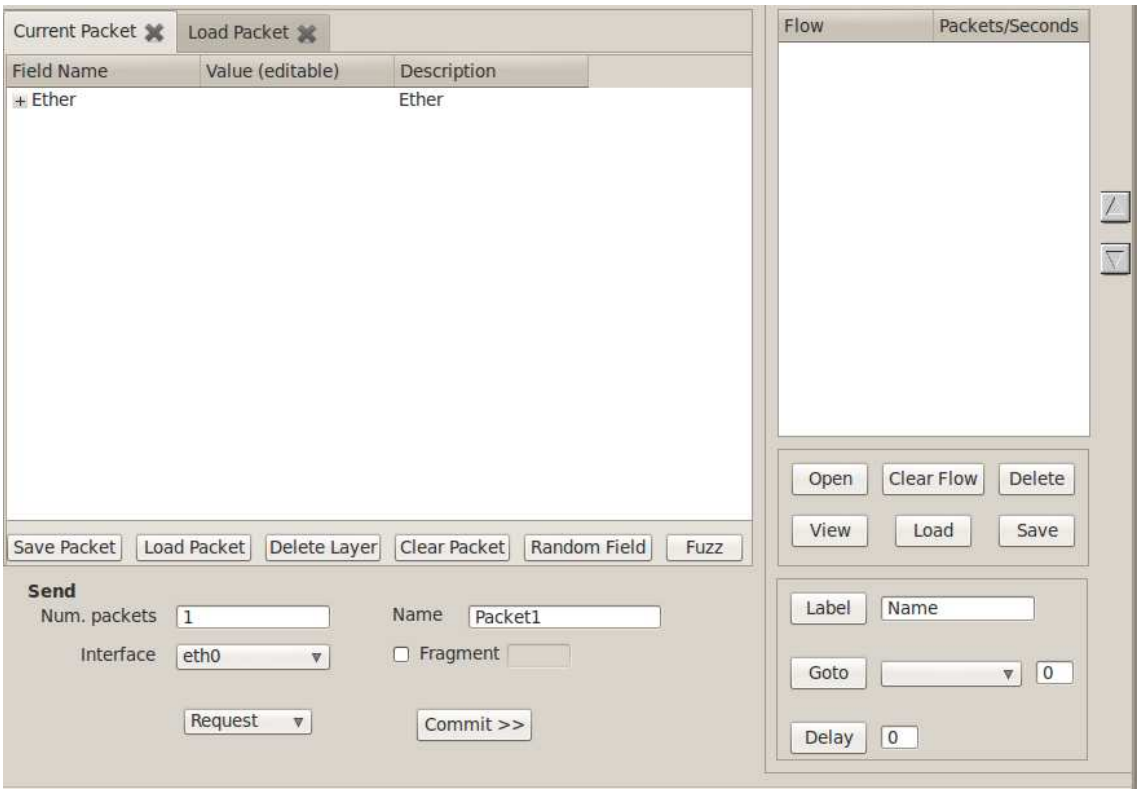

Figura 4.15. Modo Servidor

Como se puede apreciar, dos partes de la interfaz han cambiado: las opciones de envío aparecen con distintas opciones y la parte derecha donde aparece un nuevo panel que servirá para manejar el flujo de paquetes.

4.2.1 Creación del flujo de paquetes

Una vez visto cómo se forman paquetes IPv6, la creación de un flujo de paquetes con Scapyst es muy sencilla. Una vez construido el paquete que se quiere añadir al flujo, basta con seleccionar las opciones de envío (de envío al flujo) y pulsar el botón *Commit*. Con esto ya tendremos nuestro paquete en el flujo. En la Figura 4.16 se puede ver un ejemplo de cómo se añadiría un flujo de mensajes ICMPv6 *Echo*:

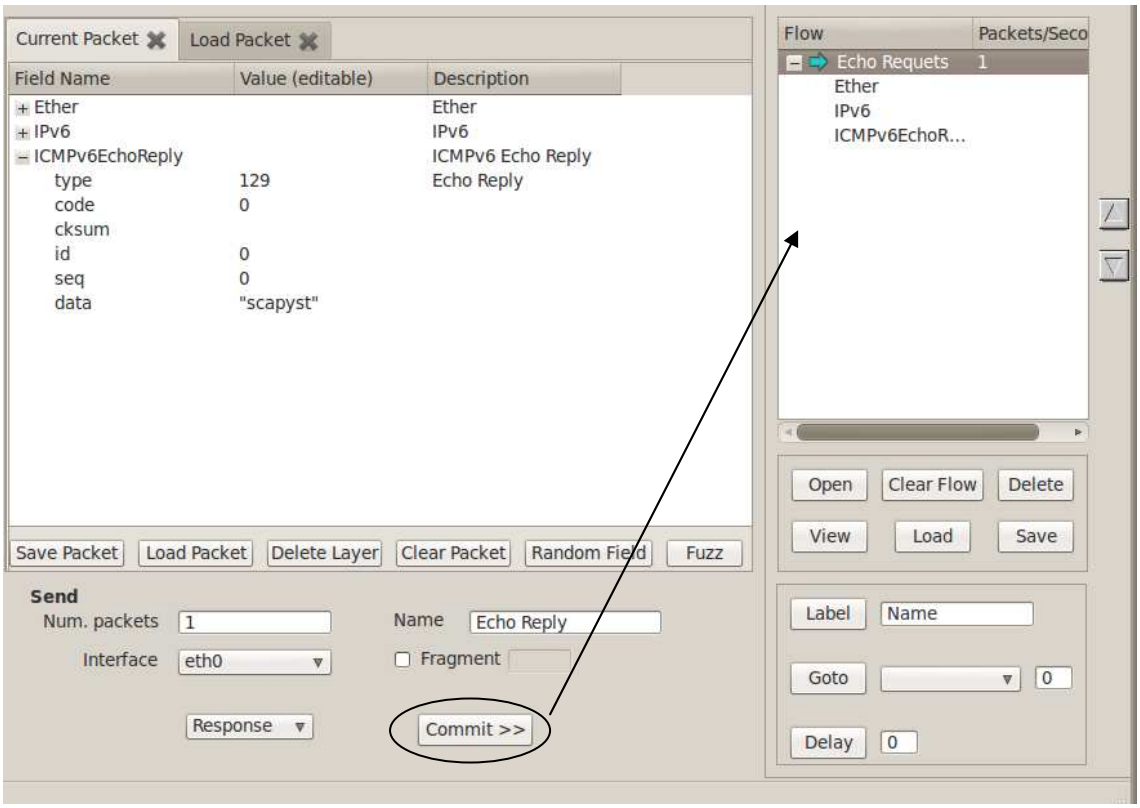

Figura 4.16. Creación de flujo ICMPv6

Las opciones de envío disponibles en el modo servidor son:

- Número de paquetes: indica cuántas veces se envía el paquete.
- Nombre: el nombre que identifica al paquete dentro del flujo.
- Fragmentación: indica si se desea utilizar fragmentación en el paquete.
- Interfaz de red por la que se ejecuta el flujo.
- Envío/Recepción: indica si el paquete es para enviar o si por el contrario es un paquete que se espera recibir. Los paquetes entrantes se identificarán con una flecha roja apuntando hacia la izquierda.

En la parte inferior derecha de la Figura 54 aparece un cuadro donde se pueden añadir otros elementos al flujo:

- Etiquetas: se añaden al flujo identificando un punto del mismo al cual podremos volver utilizando una clausula "*goto*". No tiene ningún efecto en la ejecución, su función es únicamente identificativa.

- *Goto*: se utiliza para romper el flujo ejecución saltando al punto identificado por la etiqueta seleccionada. Se puede indicar el número de veces que esta acción se ejecuta. A medida que se añaden etiquetas al flujo, estas también están disponibles para su selección en el elemento *Goto.*
- *Delay:* estable una pausa en la ejecución del flujo del tiempo que se indique.

Para ver cómo quedaría un flujo en el que estuvieran todos los elementos, se presenta la Figura 4.17:

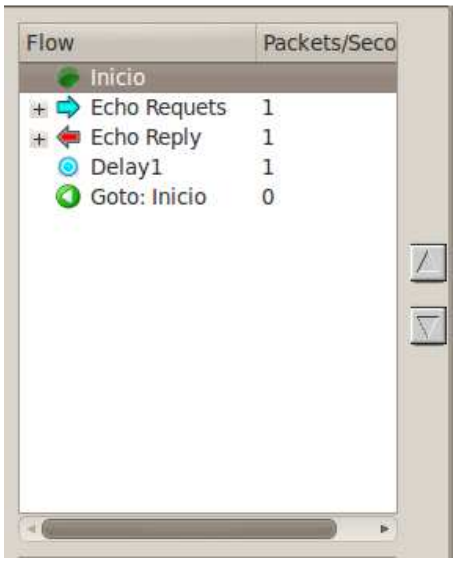

Figura 4.17. Flujo Completo

En este caso, primero se enviaría un mensaje ICMPv6 *Echo Requets*, se esperaría a recibir un mensaje ICMPv6 *Echo Reply*, a continuación se haría una pausa de 1 segundo para posteriormente volver a repetir el proceso volviendo al punto marcado por la etiqueta "Inicio" (un valor de 0 en el elemento *Goto* indica que no hay un límite en el número de veces que se ejecuta el salto).

Adicionalmente se han implementado una serie de funcionalidades que facilitan el trabajo: borrado de paquetes del flujo, visualización de paquetes, movimiento de la posición de los elementos dentro del flujo, guardado de sesiones y carga de sesiones previamente guardadas.

4.2.2 Variables y filtros

Como se comentó en el apartado anterior, uno de los modos de funcionamiento del modo servidor es interactuando con los paquetes que envían otras máquinas. De esta forma, resulta interesante no sólo esperar a recibir estos paquetes, sino poder utilizar los valores de estos campos en paquetes salientes posteriores. Por todo esto se han implementado dos nuevas funcionalidades (para seleccionar estas funcionalidades se debe pulsar el botón secundario sobre un paquete entrante y seleccionar la opción que se desee):

Establecimiento de filtros para focalizar el paquete que se desea recibir.

Por defecto para determinar si un paquete recibido es el que esperamos, únicamente se comprueba si coincide con la estructura (las capas) del paquete que hemos añadido al flujo. Mediante la adición de filtros, se puede concretar en valores de los campos incluyen por ejemplo que el paquete recibido tenga una determinada dirección IPv6 origen. En la Figura 4.18 se muestra un ejemplo de cómo establecer un filtro en un paquete de entrada:

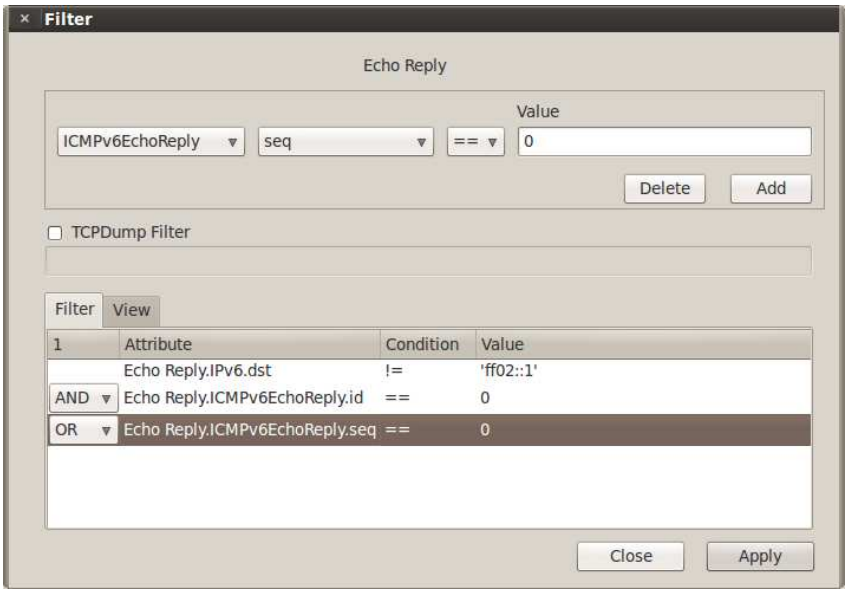

Figura 4.18. Establecimiento de filtro

Como vemos el filtro se crea mediante la definición de condiciones, eligiendo la capa, el campo, la condición lógica y el valor. Se pueden añadir varia condiciones y elegir cómo interactúan entre ellas mediante las opciones "*AND*" y

"*OR*". Además, es posible crear filtros mediante la sintaxis utilizada en la herramienta tcpdump. En el ejemplo anterior, al establecer el filtro se fuerza a que el paquete recibido no sólo tenga la estructura de un mensaje ICMPv6 *Echo Reply* sino que además tenga dirección IPv6 destino distinta de 'ff02::1' y id igual a 0 ó número de secuencia igual a 0.

Crear variables a partir de los valores de los campos de los paquetes recibidos.

En muchos casos es necesario utilizar valores de los paquetes recibidos en los paquetes que se envían. Un claro ejemplo es el de un establecimiento de sesión TCP, y es el que utilizaremos para mostrar cómo establecer variables en el flujo.

A modo de recordatorio, para establecer una sesión TCP se utiliza el mecanismo denominado *Three Way Handshake*. La máquina origen envía un paquete TCP con el flag SYN activo y con un número de secuencia "seq1". La máquina destino, cuando recibe el paquete, responde con un paquete TCP con los flags SYN y ACK activos, con un número de secuencia "seq2" y con el campo ack de valor "ack1=seq+1". La máquina origen recibe el paquete y para completar el establecimiento de la sesión envía un paquete TCP con el flag ACK activo, número de secuencia "ack1" y valor de ack "seq2+1". Este comportamiento lo podemos emular en la herramienta mediante la definición de variables, realizando los pasos que se muestran a continuación:

- Creación del flujo de paquetes:

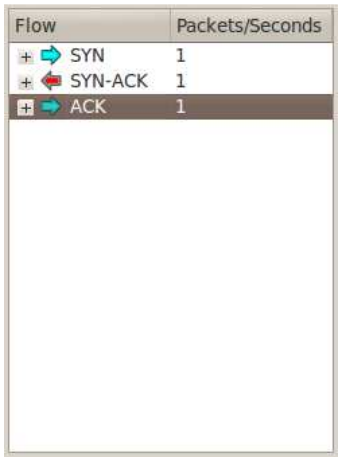

Figura 4.19. Flujo sesión TCP

- Definición de filtros (como vimos anteriormente) y variables:

Figura 4.20. Definición de variable

Una vez definidas las variables, las podemos consultar en el apartado "Info" que en el modo servidor queda como muestra la Figura 59:

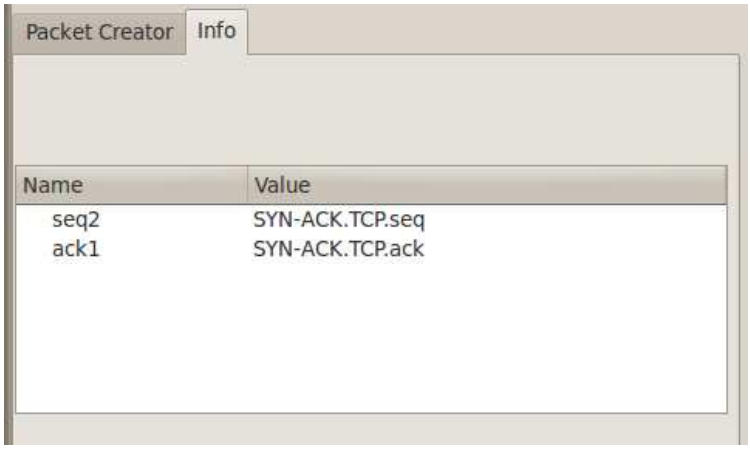

Figura 4.21. Visualización de variables

Ahora que ya están definidas las variables, éstas se pueden utilizar en el paquete "ACK" definido posteriormente. La sintaxis necesaria para utilizar la variable como valor del campo es: *var("nombre_variable")*. En la Figura 4.22 se muestra un ejemplo de utilización de las variables:

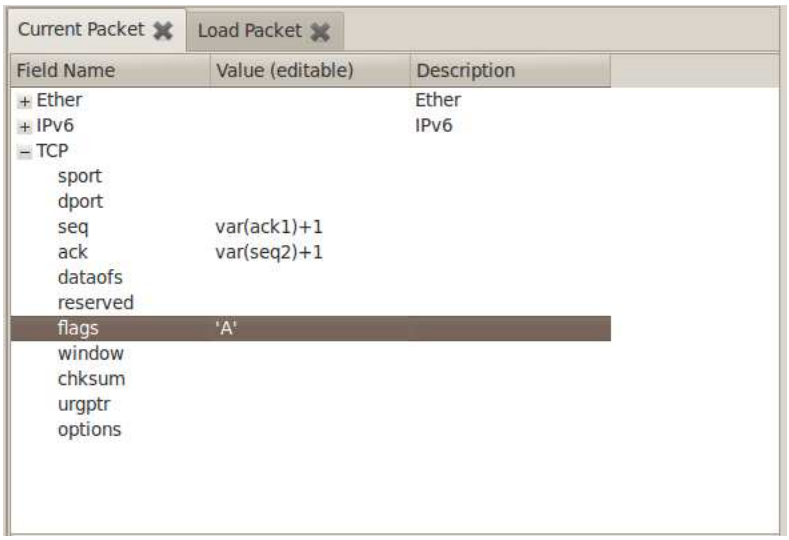

Figura 4.22. Ejemplo de utilización de variables

De este modo, el valor del campo "seq" en el paquete "ACK" tendrá el valor del campo "ack" en el paquete "SYN-ACK" más 1 y el campo "ack" del paquete "ACK" tendrá el valor del campo "seq" del paquete "SYN-ACK" más 1. Por último ejecutaríamos el flujo (ver apartado 4.2.3) y se establecería la sesión TCP.

4.2.3 Ejecución del flujo

Una vez se tiene completamente definido el flujo de paquetes con todos los elementos, el siguiente paso es su ejecución. Para ello se pulsa el botón "*Open"*, abriéndose la siguiente ventana:

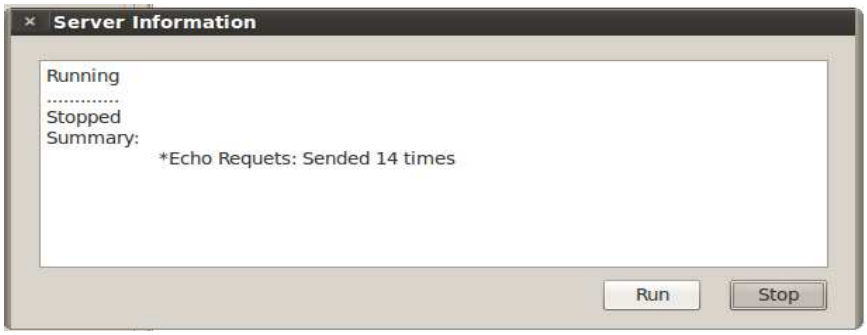

Figura 4.23. Ejecución del flujo en modo servidor

Para comenzar la ejecución del flujo se pulsa el botón "*Run"*. En este momento comienza la ejecución, y aparece un mensaje de información que anuncia que el proceso está en marcha. Para parar la ejecución se pulsa el botón *"Stop"* y a continuación se muestra un breve resumen de la actividad.

4.3 ATAQUES

Otro de los de los apartados que contiene la barra de menú es el de ataques. En él se han implementado una serie de ataques en redes IPv6 conocidos, y cada uno consta de su propia interfaz de configuración. En los siguientes apartados se muestran dichas interfaces y se describe el modo de configuración de cada ataque. Para información sobre la descripción teórica se remite al Capítulo 3.

4.3.1 *Flood Route*

Este ataque consiste en enviar un *flooding* de mensajes *Router Advertisement* anunciando prefijos aleatorios, de forma que el equipo comience a configurar direcciones IPv6 para estos prefijos. Esto conlleva una carga de procesamiento que puede llegar a saturar el equipo atacado, creando por lo tanto una denegación de servicio. De la descripción del ataque se concluye que principalmente es un ataque dirigido a hosts. La interfaz que nos permite ejecutar este ataque es muy sencilla, puede verse en la Figura 4.24. Los parámetros del ataque que se pueden configurar son:

- *Delay:* tiempo entre envío de paquetes.
- Número de paquetes que se enviarán en el ataque (si el campo se deja en blanco, se envía en modo *loop*).
- Dirección IPv6 del equipo (o equipos) sobre el que se ejecuta el ataque.
- Interfaz de red por la que se envían los paquetes.

Figura 4.24. Ataque Flood Route

4.3.2 *NDP Table Exhaustion*

Al igual que el ataque anterior estaba dirigido a hosts, este ataque está dirigido a routers o equipos de encaminamiento. El ataque se basa en el hecho de que los routers guardan en una tabla (*CAM Table*) relaciones entre direcciones IPv6 y direcciones MAC. Por ejemplo, cuando un *switch* tiene que encaminar un paquete IPv6 hacia una dirección IPv6, guarda en su tabla una entrada que contiene la dirección IPv6 origen y la dirección MAC destino, o en caso de no conocerla pone un valor temporal que indica que se trata de una entrada incompleta. Debido al gran número de host que pueden existir en una subred (recordemos que se fijan prefijos de 64 bits), podría realizarse *flooding* de paquetes a direcciones IPv6 aleatorios dentro de la subred para intentar sobrecargar la tabla CAM del equipo de encaminamiento. La potencia de este ataque radica en que puede realizarse tanto de forma local como de forma remota y en que afectaría a toda una subred.

Mediante la herramienta Scapyst se puede configurar y lanzar el ataque, mediante la interfaz que se muestra en la Figura 4.25. En ella se establecen los siguientes parámetros:

- *Delay:* tiempo entre envío de paquetes.
- Número de paquetes que se enviarán en el ataque (si el campo se deja en blanco, se envía en modo *loop*).
- Selección de ataque en modo local o remoto.
- Prefijo IPv6 (modo remoto) de la subred a atacar y dirección MAC de la puerta de enlace
- Interfaz de red por la que se envían los paquetes.

Figura 4.25. NDP Table Exhaustion

4.3.3 *HopByHop DoS*

Como se vio, este ataque consiste en hacer *flooding* de paquetes IPv6 con cabecera de extensión *Hop-By-Hop* que contenga un número elevado de opciones, con el objetivo de aumentar la carga de procesamiento de la maquina atacada y crear una denegación de servicio. La ventana en la cual se puede configurar este ataque se muestra a continuación:

Figura 4.26. Ataque Hop-By-Hop DoS

Los parámetros que se permiten configurar son:

- *Delay:* tiempo de espera en el envío de un paquete y el siguiente.
- Número de paquete*:* número de paquetes que se enviarán en el ataque (si el campo se deja en blanco, se envía en modo *loop*).
- Número de opciones: indica el número de opciones que se incluirán en la cabecera Hop-By-Hop del paquete malicioso.

- Dirección IPv6 objetivo: la dirección IPv6 del equipo que queremos atacar.
- Interfaz de red por la que se ejecuta el ataque.

4.3.4 *Destination Options DoS*

El ataque de denegación de servicio mediante *flooding* de cabeceras de extensión *Destination Options* con múltiples opciones es similar al visto anteriormente, solo que cambia el tipo de cabecera de extensión utilizada. Dado que en este caso tratamos con la cabecera *Destination Options*, este ataque sólo afecta (en principio) a los que equipos finales, ya que esta cabecera únicamente debe procesarse en el equipo final. La forma de configurar el ataque es idéntica al caso anterior.

Figura 4.27. Ataque Destination Options

4.3.5 *Cache Poisoning*

El ataque *Cache Poisoning* o como se suele conocer "*Man in the Middle*" consiste en enviar paquetes maliciosos de forma que la máquina objetivo (M1) establezca que nuestra dirección IPv6 se corresponde con la dirección MAC de otra máquina (M1). De esta forma todo el tráfico que se envíe desde la máquina M1 a la máquina M2 será enrutado hacia nuestro equipo. Si este mecanismo se aplica en ambos sentidos, se habrá interceptado la comunicación entre ambas máquinas. En IPv6 este proceso se realiza enviando mensajes *Neighbor Advertisement* modificados. Scapyst contiene una opción para realizar este ataque de forma sencilla, introduciendo unos pocos parámetros como se muestra en la Figura 4.28:

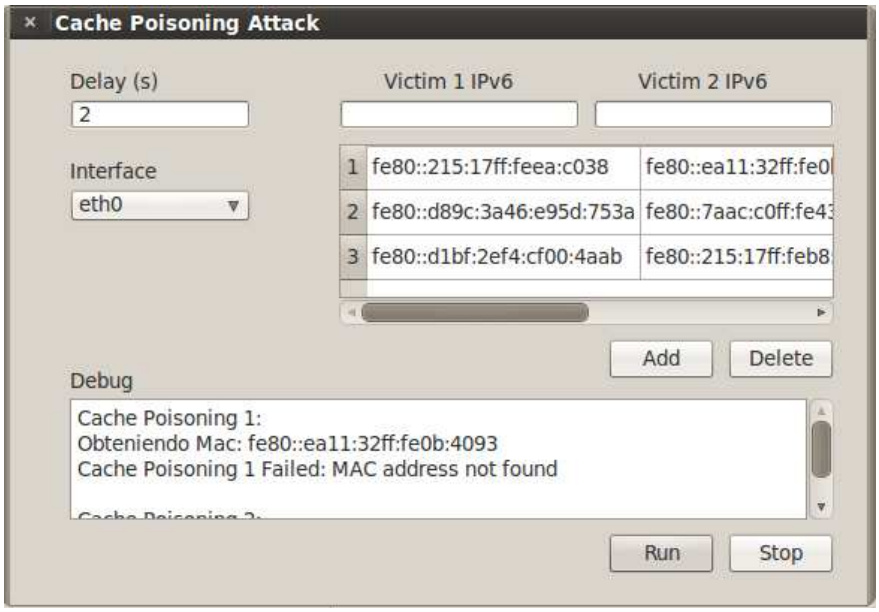

Figura 4.28. Ataque Cache Poisoning

Los parámetros que se pueden configurar en este caso son: *delay,* interfaz de red y las direcciones IPv6 de las víctimas entre las cuales queremos interceptar la comunicación. Además se muestra un pequeño cuadro con información sobre el proceso.

4.3.6 *Flood* DHCPv6 *Solicit*

Mediante este ataque podemos realizar un *flooding* de mensajes DHCPv6 *Solicit* sobre servidores DHCPv6 para intentar agotar los recursos (direcciones IPv6, prefijos…) de los que dispone y de esta forma inhabilitar a los demás usuarios para utilizar este servicio. La interfaz se muestra en la Figura 4.29 y como se observa, se pueden configurar los siguientes parámetros: *delay*, número de paquets, interfaz de red y dirección IPv6 objetivo.

Figura 4.29. Ataque Flood DHCPv6 Solicit

4.4 *FUZZING*

En el Capítulo 3 se describió en qué consistían las pruebas de *fuzzing*. Son técnicas englobadas dentro del *packet crafting* que permiten la detección de errores de implementación en el software mediante en el envío de datos pseudo-aleatorios. En la herramienta Scapyst se han implementado algunas pruebas de *fuzzing* que nos permitirán testear distintas implementaciones de la pila IPv6. La ejecución de estas pruebas se realiza mediante la opción *Fuzzing* contenida en la barra de menú. Cuando pulsamos esta opción, aparece una ventana donde se ofrecen distintas opciones para realizar las pruebas:

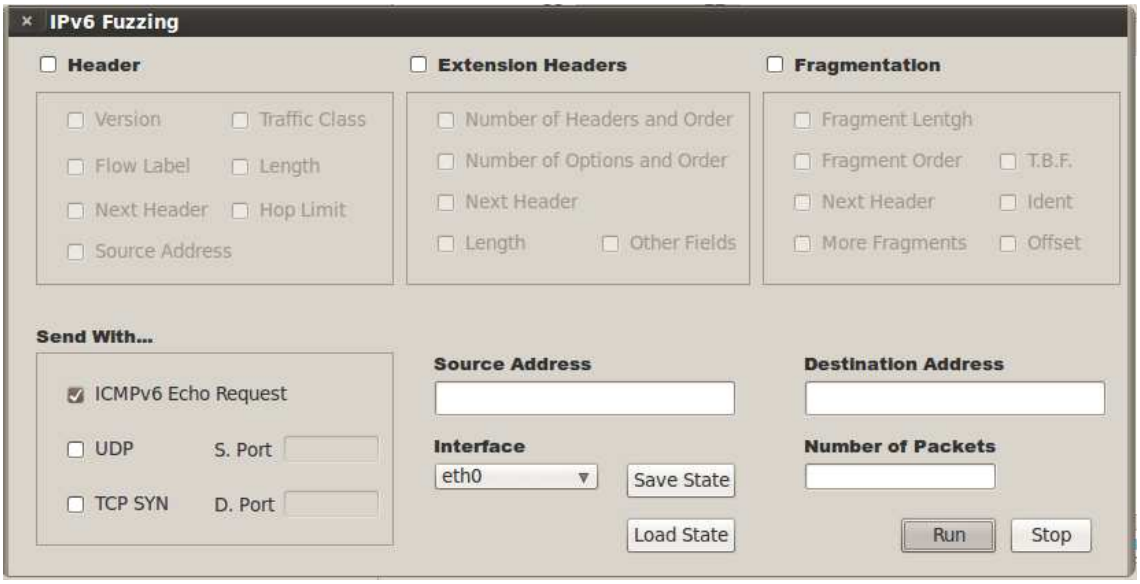

Figura 4.30. Pantalla general fuzzing

Como se puede observar, se pueden realizar pruebas de *fuzzing,* sobre la cabecera IPv6, cabeceras de extensión y sobre la fragmentación (por separado o de forma conjunta). En los siguientes sub-apartados se detallan cada una de estas pruebas por separado.

4.4.1 IPv6 *Header*

Las pruebas de *fuzzing* sobre la cabecera IPv6 son las más sencillas ya que únicamente consisten en asignar un valor aleatorio a los campos. Se realizan sobre todos los campos excepto la dirección IPv6 destino, como se puede ver en la Figura 4.31. En la Figura 4.30 se ha visto que en la pantalla general de *fuzzing* se incluye un campo para la dirección origen, y en este apartado se ha comentado que podemos hacer *fuzzing* sobre dicho campo. Obviamente ambas opciones no pueden ejecutarse de forma conjunta por lo que al seleccionar la opción de *fuzzing* sobre la dirección origen, el cuadro de la pantalla general se desactivará.

Figura 4.31. Fuzzing cabecera IPv6

4.4.2 *Extension Headers*

En la Figura 4.32 pueden verse las pruebas de *fuzzing* disponibles para las cabeceras de extensión. Se puede variar el número y orden de las cabeceras de extensión que se incluirán en el paquete, el número y orden de opciones de las opciones que aparecen en cada cabecera, el valor del campo "*next header"* y el de longitud y por último el valor de otros campos que aparecen en las opciones y en la cabecera de algunas extensiones. Hay que resaltar que la velocidad de generación de estos paquetes no es muy elevada debido a que conlleva la generación de

número aleatorios y una alta carga de procesamiento. Sin embargo, esto no es un factor muy importante en las pruebas de *fuzzing* ya que el objetivo no es realizar un ataque por denegación de servicio, sino probar una gran cantidad de combinaciones de valores de los campos del paquete.

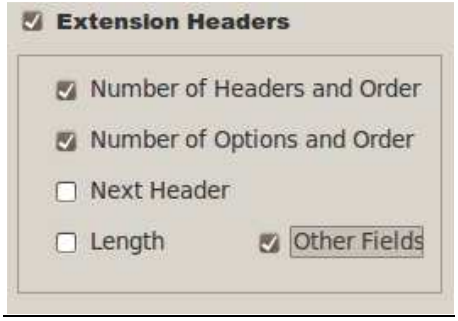

Figura 4.32. Fuzzing cabeceras de extensión

4.4.3 *Fragmentation*

Los casos de prueba para la fragmentación se muestran en la Figura 4.33. En este caso se puede aleatorizar el la longitud de los fragmentos, el orden de envío, el tiempo entre envío de fragmentos (T.B.F), y los valores de los campos "*Next Header*", "*More Fragments*", "*Ident*" y "*Offset*" pertenecientes a las cabecera de extensión de fragmentación. De nuevo esto es un proceso lento ya que el número potencial de fragmentos puede ser muy elevado y la carga de procesamiento a la hora de generar los paquetes es elevada.

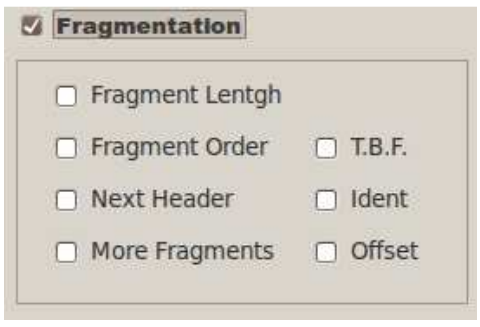

Figura 4.33. Fuzzing de fragmentación

4.4.4 Otras Opciones

Para terminar con el apartado de *fuzzing,* en la Figura 4.34 se muestran el resto de opciones que se pueden utilizar para aplicar las técnicas de *fuzzing.* El primer apartado que se observa es el recuadro "*Send With"* que ofrece la posibilidad de enviar el paquete IPv6 *fuzzeado* mediante un mensaje ICMPv6 *Echo Requets*, o mediante los protcolos UDP o TCP (SYN) en los puertos que se especifiquen. A la derecha se pueden especificar las direcciones IPv6 origen y destino. Como se comentó anteriormente, si se realiza *fuzzing* sobre el campo dirección origen de la cabecera IPv6, la opción de incluir esta dirección manualmente se desactivará. Además se puede seleccionar la interfaz de red por la que se realizarán las pruebas y establecer el número de paquetes a enviar (si no se pone ningún, se envirará en modo *loop*).

Un apartado de gran importancia a la hora de realizar las pruebas se implementa mediante las opciones de guardar y cargar el estado. Mediante esta funcionalidad podemos guardar el estado inicial del generador de números aleatorios, comenzar a ejecutar las pruebas y cuando terminen volver a repetir las mismas pruebas, lanzando los mismos paquetes. Esto es importante porque el *fuzzing* conlleva un alto grado de aleatoriedad, por lo que si no guardáramos el estado sería imposible repetir de forma exacta una prueba.

Por último, tenemos los botones para lanzar las pruebas y para detenerlas.

Figura 4.34. Otras opciones sobre pruebas fuzzing

4.5 *TOOLS*

Para terminar con la herramienta Scapyst, se describen una serie de funcionalidades añadidas tratan de facilitar algunos aspectos de trabajo del usuario. Estas son:

- Posibilidad de detectar sesiones PPP.
- Integración de una consola de Python en la herramienta Scpyst.
- Posibilidad de enviar archivos "pcap" al servicio web Pacpr.net[1].

4.5.1 PPP *Session*

Mediante esta opción pueden detectarse sesiones PPP activas y añadir automáticamente el número de sesión a la capa PPP de nuestro paquete. Esto evita que al querer comunicarnos mediante una sesión PPP (establecida por nosotros o por un tercero) con la herramienta, tengamos que buscar mediante un analizador de tráfico los paquetes que corresponden a la sesión.

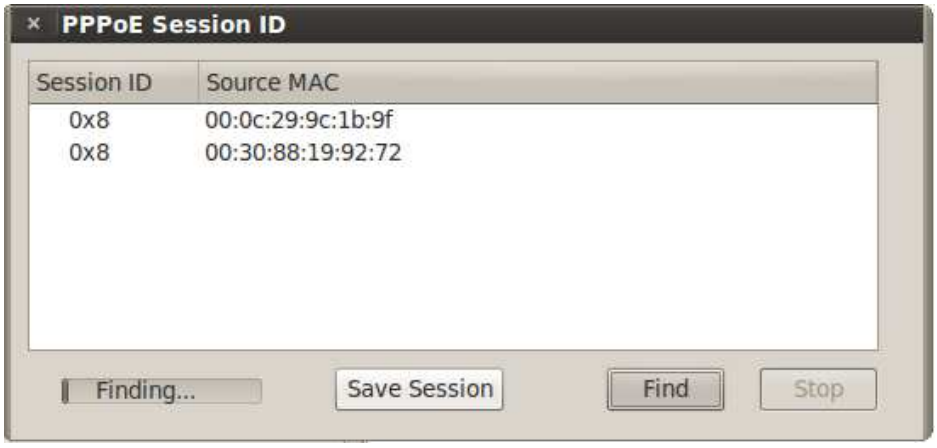

Figura 4.35. PPP Session

4.5.2 Consola de Python

También se proporciona la posibilidad de integrar una consola de Python en la herramienta Scapyst. Esto puede ser de utilidad en momentos que se desea realizar alguna comprobación rápida (recordemos que Scapy funciona bajo Python) de forma cómoda. Por defecto se importa la librería Scapy en la consola,

[1] www.pcapr.net

por lo que también nos ofrece la posibilidad de utilizar Scapy en modo comando dentro de la herramienta. La consola se integra a partir de un entorno de desarrollo creado para Python llamado Spyder[1].

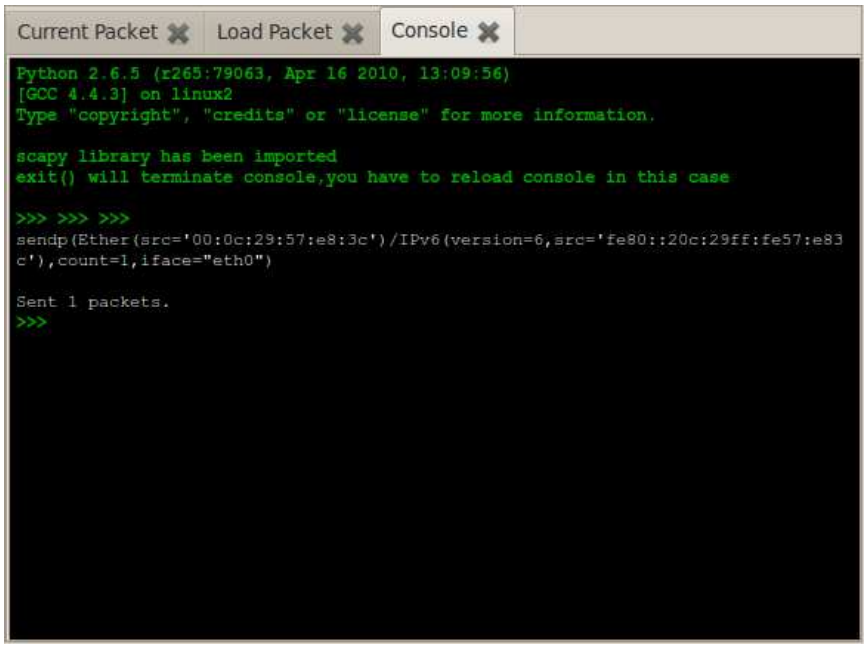

Figura 4.36. Consola de Comandos Python

4.5.3 Send To Pcapr.net

Por último, se ofrece la opción de subir archivos "pcap" al servicio web Pcapr.net, una red social de *networking* creada por la empresa Mu Dynamics.

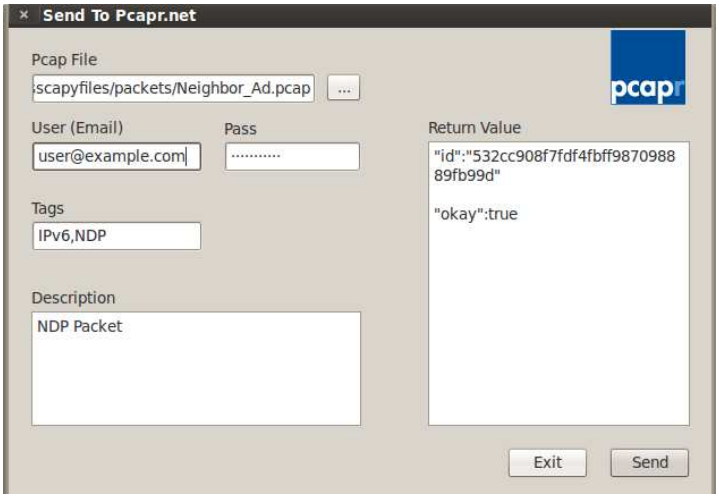

Figura 4.37. Send To Pcapr.net

[1] http://code.google.com/p/spyderlib/

CAPITULO 5

PRUEBAS DE ESTABILIDAD

PRUEBAS DE ESTABILIDAD

La implantación del protocolo IPv6 en sistemas de uso general es un hecho. Así, una gran cantidad de sistemas operativos presentes tanto en computadoras y routers como en otros dispositivos conectados a internet (como teléfonos móviles) incorporan el protocolo IPv6. De hecho, en los principales sistemas operativos utilizados en los ordenadores personales, el protocolo IPv6 es el que se utiliza por defecto, dando preferencia a éste frente a IPv4. El motivo de que el protocolo IPv4 siga siendo mayoritariamente utilizado es que la actualización del protocolo en las redes de las operadoras de telecomunicaciones es un proceso lento y costoso. Como se puede ver en el gráfico de la Figura 5.1, la utilización del protocolo IPv6 en la actualidad es muy baja. Es por ello que este es un punto ideal para realizar pruebas sobre las implementaciones de la pila IPv6 en los distintos dispositivos que la incorporan.

Figura 5.1. Estadística de utilización del protocolo IPv6

En el Capítulo 2 se describió el protocolo IPv6 y sus principales mecanismos de operación. Después se analizaron los principales riesgos de seguridad que la implantación del protocolo podría conllevar y una serie de ataques que han aparecido desde su implantación, además se definieron una serie de pruebas de *fuzzing.* En el Capítulo 4 se presentó la herramienta Scapyst, desarrollada con el

objetivo de facilitar a cualquier profesional de la seguridad informática la realización de pruebas sobre la pila IPv6. El presente capítulo se centra en la realización de pruebas sobre implementaciones del protocolo IPv6 en sistemas operativos de uso común. Además estas pruebas servirán para comprobar el correcto funcionamiento de la herramienta Scapyst.

5.1 DESCRIPCION DEL ENTORNO DE PRUEBAS

En la Figura 5.2 se puede ver el esquema del entorno utilizado para realizar las pruebas de estabilidad de distintas implementaciones de la pila del protocolo IPv6. Se trata de una maqueta que se compone de una red con direccionamiento IPv6 y equipos de red y terminales capaces de manejar el tráfico IPv6. Esta maqueta está formada por:

- Router *rt-seg-ipv6:* Ubuntu 10.04 LTS 32 bits Desktop.
- Equipo *ht-seg-ipv6-01*: Debian 6 server, sobre núcleo Linux 2.6.32-5-686.
- Equipo *ht-seg-ipv6-02,03*: Ubuntu 10.04 LTS 32 bits.
- Equipo *ht-seg-ipv6-04*: Windows 7.
- Equipo *ht-seg-ipv6-05*: Windows Server 2003.

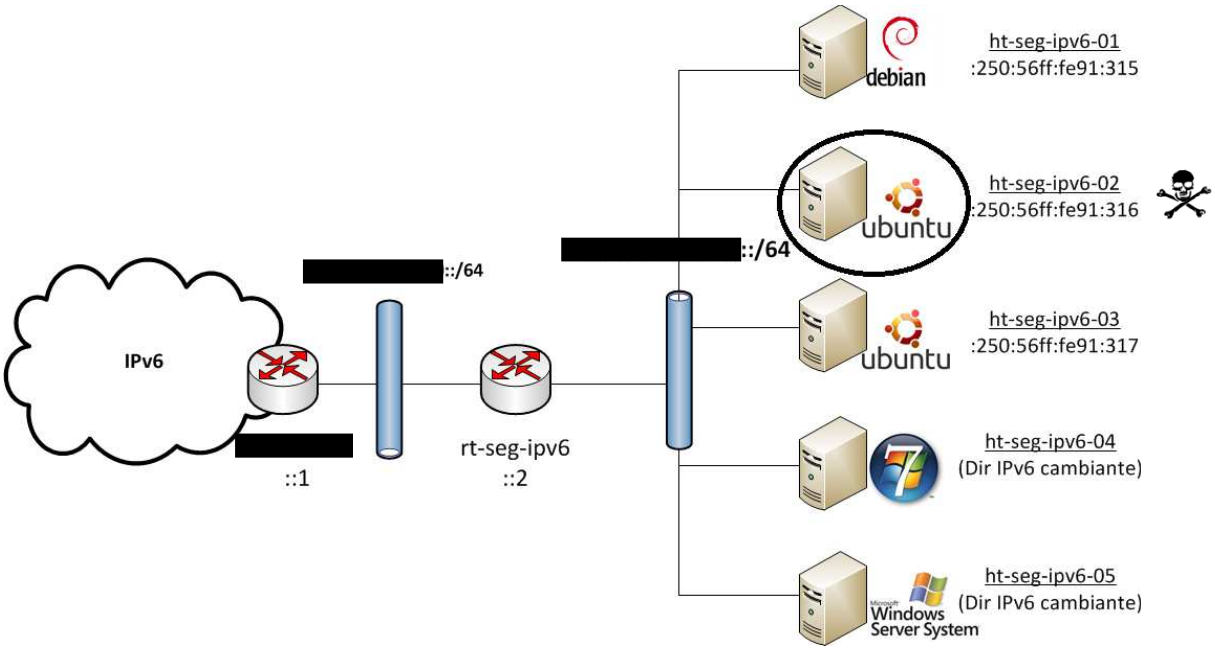

Figura 5.2. Entorno de pruebas IPv6

Dentro de este entorno, se han elegido las máquinas *ht-seg-ipv6-01 y ht-seg-ipv6-04* para realizar las pruebas. De este modo los test tienen una clara orientación al usuario, analizando dos implementaciones del protocolo IPv6 mayoritarias en internet.

5.2 METODOLOGIA EMPLEADA PARA LAS PRUEBAS

Como se ha comentado, las máquinas sobre las que se realizarán las pruebas son *ht-seg-ipv6-01 y ht-seg-ipv6-04*. Además para llevarlas a cabo, se utilizará el equipo *ht-seg-ipv6-02* como atacante y el equipo *ht-seg-ipv6-03* para comprobar el estado de la conexión de las máquinas bajo prueba. De forma general, se pueden definir una serie de pasos que se siguen para realizar las pruebas:

1. Comprobación y puesta a punto del entorno.
2. Ejecución del ataque.
3. Comprobación del estado de la conexión del equipo atacado.
4. Monitorización de los recursos (utilización de CPU, estado de la memoria…) del equipo atacado y su evolución con el tiempo.
5. Estudio de los mensajes de respuesta enviados por el equipo bajo pruebas.
6. Fin del ataque.

Los ataques se ejecutarán utilizando la herramienta Scapyst descrita en el Capítulo 4. Dado que las pruebas se realizan sobre equipos finales, no todos los se ejecutarán todos los ataques implementados en Scapyst. Los ataques empleados en las pruebas son: ataques de denegación de servicio mediante *flooding* de cabeceras de extensión *Hop-By-Hop* y *Destination Options* con número de opciones variables, ataque mediante *flooding* de paquetes *Neighbor Advertisement* y ataque *Cache Poisoning* (*Man in the middle*).

Posteriormente se realizarán las pruebas de *fuzzing* de la cabecera IPv6, cabeceras de extensión y fragmentación. La monitorización de estas pruebas se realiza de forma similar a la monitorización de los ataques, con la diferencia de que el estudio del comportamiento en cuanto a mensajes de respuesta del equipo bajo

pruebas se basará en las pruebas definidas por el IPv6 *Forum* [38]. De esta forma, tomamos este documento como referencia, algo importante ya que no todas las implementaciones manejan todos los paquetes IPv6 de la misma forma. Estas pruebas se realizan en primer lugar de forma individual, realizando *fuzzing* sobre un solo parámetro para después realizar las pruebas de forma global. Los parámetros modificados se enviarán en un mensaje ICMPv6 *Echo Request*, de forma que podamos observar si el equipo responde a este mensaje o lo descarta. A continuación se resume el comportamiento esperado de las máquinas atacadas, basándonos en la referencia anterior:

- Cabecera IPv6: el comportamiento esperado al realizar pruebas de *fuzzing* sobre cada uno de los campos que componen la cabecera es:
    - *Version*: el equipo bajo pruebas no debe generar mensaje de respuesta ante mensajes donde el valor del campo es distinto de 6.
    - *Traffic Class*: se debe responder mediante un mensaje ICMPv6 *Echo Reply*. El valor del campo *Traffic Class* del mensaje de respuesta debe ser 0 si el equipo no soporta el uso de esta característica.
    - *Flow Label*: se debe responder mediante un mensaje ICMPv6 *Echo Reply*. El valor del campo *Flow Label* del mensaje de respuesta debe ser 0 si el equipo no soporta el uso de esta característica.
    - *Length:* ante un valor erróneo en este campo, el equipo debe descartar el paquete sin generar ningún tipo de respuesta.
    - *Next Header:* la máquina debe responder con un mensaje ICMPv6 *Parameter Problem* (*code* 1, *offest* 0x06) cuando el valor del campo es erróneo.
    - *Hop Limit*: el equipo debe responder mediante un mensaje ICMPv6 *Echo Reply* con el valor del campo *Hop Limit* mayor que 0.
- Cabeceras de extensión: las pruebas de *fuzzing* realizadas en las cabeceras de extensión consisten en modificar de forma aleatoria varios parámetros, esperando el siguiente comportamiento:
    - Número y orden de las cabeceras: ante un orden incorrecto de las cabeceras o un número inadecuado de éstas, el equipo debe descartar el paquete sin generar respuesta.

- Número y orden de las opciones: el equipo debe actuar en función del valor que tomen los dos primeros bits del campo *Option Type* de la opción correspondiente.
- *Next Header*: la máquina debe responder con un mensaje ICMPv6 *Parameter Problem* (*code* 1,*offest* 0x28) cuando el valor del campo es erróneo.
- *Length:* ante un valor erróneo en este campo, el equipo debe descartar el paquete sin generar ningún tipo de respuesta.

- Fragmentación: en este caso se comprobará el correcto funcionamiento del mecanismo de reensamblado cuando se envían fragmentos con valores erróneos en sus campos, o se varía el orden de envío de los fragmentos.

Además del comportamiento, se monitorizará el estado de la conexión del equipo bajo pruebas, así como otros parámetros que puedan indicar un tratamiento erróneo de los paquetes malformados.

5.3 RESULTADO DE LAS PRUEBAS

En este apartado se describe el resultado obtenido en cada una de las pruebas aplicadas sobre las dos pilas IPv6 escogidas.

5.3.1 Pruebas sobre Debian Server

Como se ha comentado, la metodología seguida para realizar las pruebas consiste en atacar al equipo objetivo desde la máquina *ht-seg-ipv6-02*, monitorizando el estado de la conexión desde la máquina *ht-seg-ipv6-03* (ver Figura 5.2). Teniendo esto en cuenta, a continuación se describen los resultados obtenidos al realizar cada uno de los ataques.

Ataque *Hop-By-Hop DoS*

Al realizar *flooding* de paquetes IPv6 con cabecera de extensión *Hop-By-Hop* con un número variable de opciones sobre la máquina *ht-seg-ipv6-01*, no se observa disminución del rendimiento del equipo ni degradación de la calidad de la conexión. De esta forma, el equipo presenta un buen comportamiento ante este ataque, descartando los paquetes de forma correcta.

Ataque *Destination Options DoS*

Al realizar *flooding* de paquetes IPv6 con cabecera de extensión *Destination Options* con un número variable de opciones sobre la máquina *ht-seg-ipv6-01*, no se observa disminución del rendimiento del equipo ni degradación de la calidad de la conexión. De esta forma, el equipo presenta un buen comportamiento ante este ataque, descartando los paquetes de forma correcta.

Ataque *Flood Route*

Se detecta que mediante un ataque de *flooding* de paquetes *Router Advertisement* con determinados parámetros, se produce una denegación de servicio en la máquina atacada. Al lanzar el ataque no se observa ninguna degradación importante ni en el rendimiento de la máquina ni en el estado de la conexión. Sin embargo, la máquina se configura una serie de direcciones IPv6 y su tabla de vecinos crece, como se puede ver en las Figuras 5.3 y 5.4.

Durante la ejecución del ataque, la máquina no pierde conectividad y el aumento en uso de CPU y memoria no es significativo. Es al detener el ataque cuando el equipo pierde conectividad, teniendo que reiniciar el equipo para que se restablezca la conexión con la red local.

```
inet6 addr: c8ef:781:143f:1264:250:56ff:fe91:315/64 Scope:Global
inet6 addr: 1cf:177c:4914:a3d5:250:56ff:fe91:315/64 Scope:Global
inet6 addr: f4a2:6f1d:2c7d:10fa:250:56ff:fe91:315/64 Scope:Global
inet6 addr: fe47:cee0:6965:56e1:250:56ff:fe91:315/64 Scope:Global
inet6 addr: 512c:a4e7:acd:4d91:250:56ff:fe91:315/64 Scope:Global
inet6 addr: 4ad8:2adf:aae5:9d58:250:56ff:fe91:315/64 Scope:Global
inet6 addr: b76a:e676:8c95:6681:250:56ff:fe91:315/64 Scope:Global
inet6 addr: 3763:ef9b:b613:b4e3:250:56ff:fe91:315/64 Scope:Global
inet6 addr: 819c:47ee:d95d:fa7c:250:56ff:fe91:315/64 Scope:Global
inet6 addr: 3959:ad00:e7ac:7ce2:250:56ff:fe91:315/64 Scope:Global
inet6 addr: 1ef9:5899:a38c:5663:250:56ff:fe91:315/64 Scope:Global
inet6 addr: f887:65a0:a2f0:3194:250:56ff:fe91:315/64 Scope:Global
inet6 addr: 87d7:f81d:9b80:8941:250:56ff:fe91:315/64 Scope:Global
inet6 addr: 9b55:9058:94ed:c944:250:56ff:fe91:315/64 Scope:Global
```

Figura 5.3. Direcciones IPv6 configuradas (Debian)

```
root@ht-seg-ipv6-01:~# ip -6 neigh show dev eth0
fe80::79bd:3aff:fe6a:9d2 lladdr 4c:bd:3a:6a:9d:02 router STALE
fe80::62cf:abff:feab:2396 lladdr d4:cf:ab:ab:23:96 router STALE
fe80::d2c3:9bff:fe15:e272 lladdr bc:c3:9b:15:e2:72 router STALE
fe80::6af0:31ff:fefd:ff1f lladdr d2:f0:31:fd:ff:1f router STALE
fe80::80b7:a7ff:feab:fed6 lladdr 4a:b7:a7:ab:fe:d6 router STALE
fe80::ce22:89ff:fe81:3d8a lladdr d4:22:89:81:3d:8a router STALE
fe80::78d:56ff:fe41:2745 lladdr 0e:8d:56:41:27:45 router STALE
fe80::291f:66ff:fe46:b48a lladdr 36:1f:66:46:b4:8a router STALE
fe80::b795:7eff:fec4:766 lladdr 76:95:7e:c4:76:06 router STALE
fe80::a07e:3aff:fe94:858 lladdr 92:7e:3a:94:85:08 router STALE
fe80::659a:b3ff:fe7:7f7b lladdr f6:9a:b3:07:7f:7b router STALE
fe80::8d8a:6ff:fe7f:6872 lladdr 4e:8a:06:7f:68:72 router STALE
fe80::d6cd:51ff:fe97:a5f8 lladdr de:cd:51:97:a5:f8 router STALE
fe80::9bbc:32ff:fe8a:e047 lladdr 34:bc:32:8a:e0:47 router STALE
fe80::be:52ff:fe44:690 lladdr bc:be:52:44:69:00 router STALE
fe80::cf5f:b2ff:febd:b79d lladdr 1e:5f:b2:bd:b7:9d router STALE
fe80::d23b:3ff:fe42:76b1 lladdr 8a:3b:03:42:76:b1 router STALE
fe80::bc1f:35ff:feb:9844 lladdr 94:1f:35:0b:98:44 router STALE
fe80::663d:c4ff:fe6d:6afd lladdr ac:3d:c4:6d:6a:fd router STALE
fe80::2988:beff:fe69:eb13 lladdr c0:88:be:69:eb:13 router STALE
fe80::d3d5:9fff:fec3:f721 lladdr f2:d5:9f:c3:f7:21 router STALE
fe80::13c:d0ff:fe8b:357 lladdr 34:3c:d0:8b:03:57 router STALE
fe80::7a64:68ff:fe78:1222 lladdr 4c:64:68:78:12:22 router STALE
fe80::88e9:32ff:fe3a:4cdd lladdr 54:e9:32:3a:4c:dd router STALE
fe80::a1d7:26ff:fef4:8145 lladdr 8a:d7:26:f4:81:45 router STALE
fe80::e73d:dff:fef4:1efe lladdr 9c:3d:0d:f4:1e:fe router STALE
fe80::8cd:daff:febe:527a lladdr 38:cd:da:be:52:7a router STALE
fe80::7e95:b9ff:fe97:57f0 lladdr 96:95:b9:97:57:f0 router STALE
fe80::fe10:2eff:fe8:3358 lladdr 34:10:2e:08:33:58 router STALE
fe80::b15:28ff:fe5f:a02e lladdr 60:15:28:5f:a0:2e router STALE
fe80::653b:34ff:fef7:dbb3 lladdr 8a:3b:34:f7:db:b3 router STALE
fe80::3bcb:44ff:fec3:32af lladdr 58:cb:44:c3:32:af router STALE
fe80::c9c3:31ff:fe6a:2598 lladdr bc:c3:31:6a:25:98 router STALE
fe80::c261:29ff:fef1:917b lladdr e8:61:29:f1:91:7b router STALE
fe80::f83b:7ff:fe51:de9e lladdr 44:3b:07:51:de:9e router STALE
fe80::cc2:e0ff:fec9:9990 lladdr e6:02:e0:c9:99:90 router STALE
fe80::c359:23ff:fe0:9b4a lladdr 9e:59:23:00:9b:4a router STALE
fe80::adcf:18ff:fe5b:d43 lladdr 36:cf:18:5b:d4:03 router STALE
```

Figura 5.4. Tabla de vecinos (Debian)

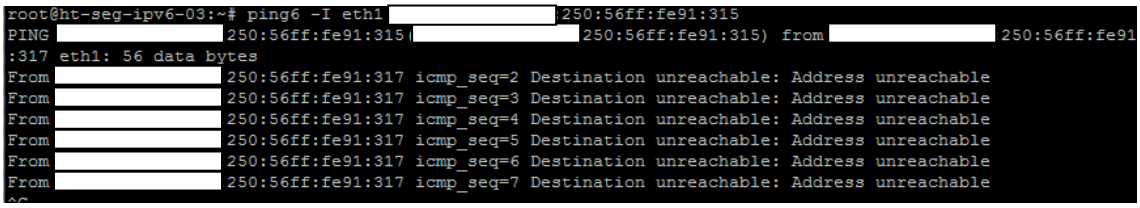

Figura 5.5. Pérdida de conectividad (Debian)

Ataque *Cache Poisoning*

El equipo resulta vulnerable a un ataque *Cache Poisoning* mediante mensajes *Neighbor Advertisement* modificados. Esta prueba se realizó atacando a la máquina *ht-seg-ipv6-01* y a la máquina *ht-seg-ipv6-03* de forma que se interceptaba la comunicación entre ellas, pasando todos los mensajes por la máquina *ht-seg-ipv6-02* (ver Figura 5.6).

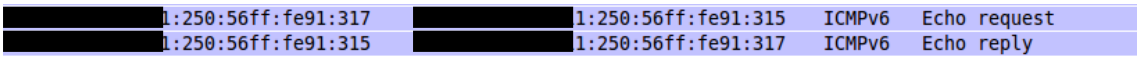

Figura 5.6. Comunicación interceptada (Debian)

Como se puede ver en la Figura 5.7, las máquinas atacadas se configuran en sus tablas de vecinos la dirección IPv6 de la otra máquina atacada con nuestra dirección MAC. Esto es lo que hace que cualquier paquete que intercambien entre ellas, pasará antes por nuestra máquina (es necesario habilitar el reenvío de paquetes en el fichero *sysctl.conf* para que nuestra máquina reenvíe el tráfico a la máquina destino y así no se produzca una denegación de servicio).

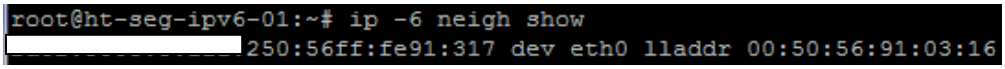

Figura 5.7. Tabla de vecinos de la máquina atacada (Debian)

Fuzzing IPv6

Ante un ataque de *fuzzing*, se ha podido observar que de forma general el equipo Debian 6 Server realiza un tratamiento correcto de los paquetes malformados.

El equipo maneja bien el *fuzzing* sobre la cabecera IPv6. Resuelve bien el trato con todo tipo de valores de los campos *Version, Traffic Class, Flow Label* (la máquina no soporta el uso de clases de tráfico ni etiquetas), *Length* y *Hop Limit*. En cuanto al campo *Next Header* el equipo responde, en un principio, correctamente con un mensaje ICMPv6 *Parameter Problem*. Sin embargo, sólo envía estos mensajes en respuesta a los primeros paquetes con *Next Header* erróneo, ya que después se limita el envío de estos mensajes y no responde a todos los paquetes con el campo erróneo. También maneja bien los paquetes cuando se realiza *fuzzing* conjunto de todos los campo de la cabecera. No se aprecia un aumento en el uso de recursos ni degradación en el estado de la conexión.

En cuanto a las pruebas de *fuzzing* realizadas sobre las cabeceras de extensión, se vuelve a observar un buen comportamiento en el tratamiento de los paquetes malformados. El único comportamiento no recomendado encontrado se produce en el manejo del número y orden de las cabeceras de extensión en un determinado caso. El equipo trata como correcto cualquier paquete que o bien no contenga la cabecera *Hop-By-Hop* o bien la contenga en primer lugar, sin importar el número, tipo y orden de las cabeceras de extensión que se sitúen después en el paquete, con lo que la implementación del protocolo no sigue las recomendaciones que se hacen en [6]. Este comportamiento puede ser utilizado maliciosamente para la creación de canales encubiertos de información.

Mediante las pruebas de *fuzzing* aplicadas al mecanismo de fragmentación se comprueba que la máquina gestiona correctamente los paquetes fragmentados, así como las situaciones inesperadas a las que se le somete mediante la herramienta Scapyst.

Por último, destacar que el equipo presenta un buen comportamiento ante pruebas de fuzzing completas, es decir, cuando se aplica el fuzzing sobre todos los campos y opciones de forma conjunta.

5.3.2 Pruebas sobre Windows 7

A continuación se describen los resultados obtenidos después del análisis de los ataques ejecutados sobre la máquina *ht-seg-ipv6-04* (Windows 7).

Ataque *Hop-By-Hop DoS*

Al realizar *flooding* de paquetes IPv6 con cabecera de extensión *Hop-By-Hop* con un número variable de opciones sobre la máquina *ht-seg-ipv6-04*, no se observa disminución del rendimiento del equipo ni degradación de la calidad de la conexión. De esta forma, el equipo presenta un buen comportamiento ante este ataque, descartando los paquetes de forma correcta.

Ataque *Destination Options DoS*

Al realizar *flooding* de paquetes IPv6 con cabecera de extensión *Destination Options* con un número variable de opciones sobre la máquina *ht-seg-ipv6-04*, no se observa disminución del rendimiento del equipo ni degradación de la calidad de la conexión. De esta forma, el equipo presenta un buen comportamiento ante este ataque, descartando los paquetes de forma correcta.

Ataque *Flood Route*

Mediante un ataque de *flooding* de paquetes *Router Advertisement* con determinados parámetros, se produce una denegación de servicio en la máquina atacada. Al instante de lanzar el ataque, se produce un incremento en el uso de CPU que llega al 100% como muestra la Figura 5.8:

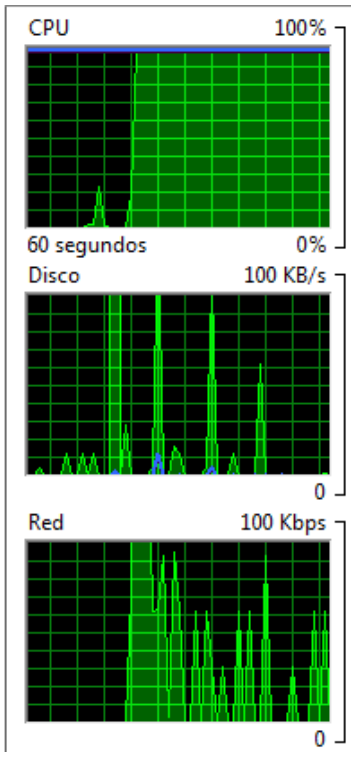

Figura 5.8. Consumo de recursos en Windows 7

Si el ataque se ejecuta de forma prolongada, la máquina pierde conectividad ya que, como se puede ver en la Figura 5.9, al realizar *ping* desde el equipo de monitorización, las respuestas llegan a tardar del orden de 15 segundos:

```
64 bytes from [redacted]:a129:23f0:41c5:f9e9: icmp_seq=93 ttl=255 time=2959 ms
64 bytes from [redacted]:a129:23f0:41c5:f9e9: icmp_seq=94 ttl=255 time=15104 ms
64 bytes from [redacted]:a129:23f0:41c5:f9e9: icmp_seq=95 ttl=255 time=14105 ms
64 bytes from [redacted]:a129:23f0:41c5:f9e9: icmp_seq=96 ttl=255 time=14253 ms
64 bytes from [redacted]:a129:23f0:41c5:f9e9: icmp_seq=103 ttl=255 time=7197 ms
```

Figura 5.9. Respuesta ping Windows 7

Este ataque provoca que la máquina se configure una dirección IPv6 para cada uno de los prefijos anunciados. Este proceso hace que la tabla de vecinos crezca de forma incontrolada, además de que el equipo realice una serie de operaciones que hace que el consumo de CPU crezca al 100%. Estos síntomas perduran aún después de parar el ataque, y si la duración del mismo es prolongada se bloquea la interacción con la interfaz de usuario, por lo que es necesario reiniciar el equipo.

Para ilustrar esto, se ha realizado el ataque enviando sólo 15 paquetes. En las siguientes figuras se muestran las direcciones IPv6 configuradas, así como la tabla de vecinos del equipo:

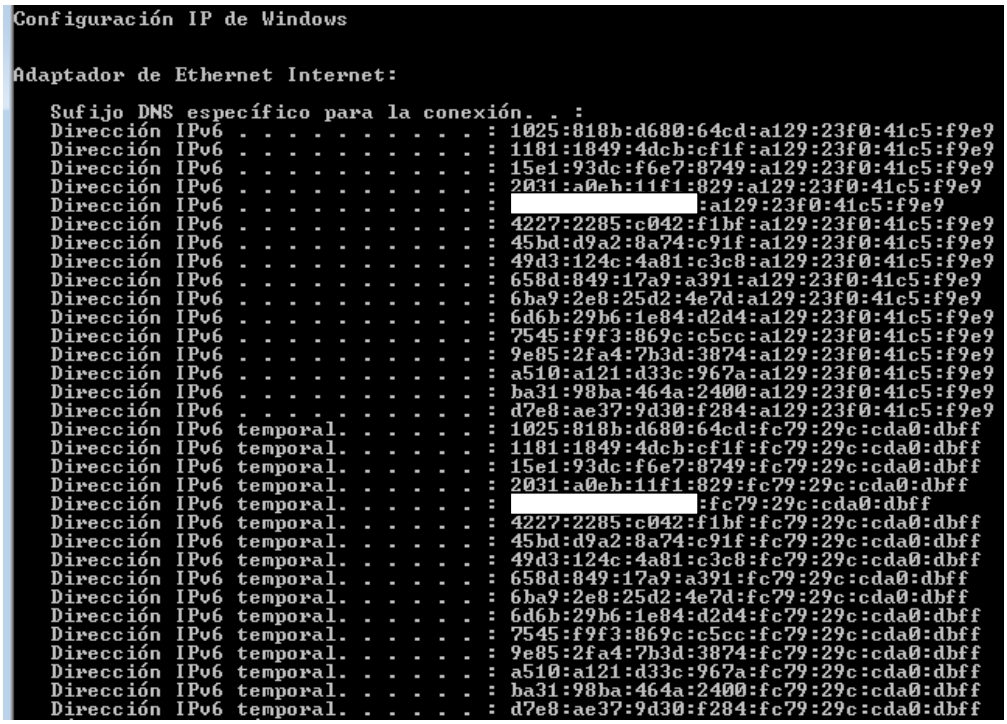

Figura 5.10. Direcciones IPv6 configuradas (Windows 7)

Ataque *Cache Poisoning*

El equipo resulta vulnerable a un ataque *Cache Poisoning* mediante mensajes *Neighbor Advertisement* modificados. Esta prueba se realizó atacando a la máquina *ht-seg-ipv6-04* y a la máquina *ht-seg-ipv6-03* de forma que se interceptaba la comunicación entre ellas, pasando todos los mensajes por la máquina *ht-seg-ipv6-02* (ver Figura 5.11).

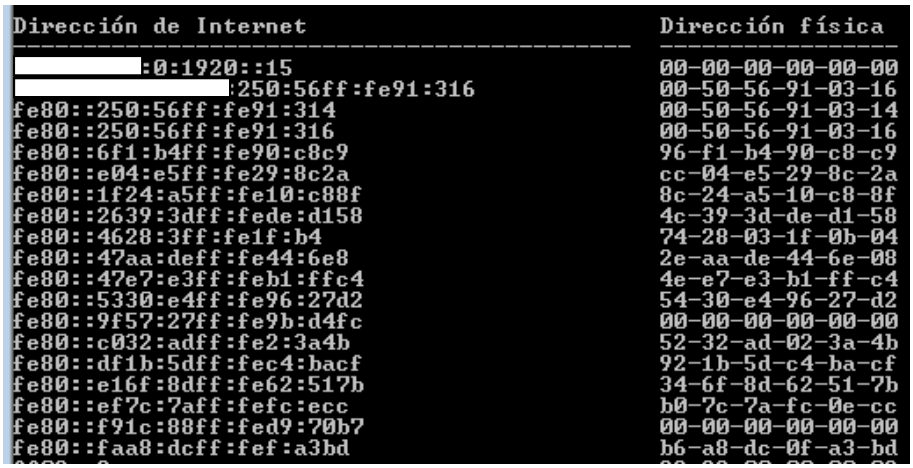

Figura 5.11. Tabla de vecinos (Windows 7)

Figura 5.12. Comunicación interceptada (Windows 7)

Como se puede ver en la Figura 5.13, las máquinas atacadas se configuran en sus tablas de vecinos la dirección IPv6 de la otra máquina atacada con nuestra dirección MAC. Esto hace que cualquier paquete que intercambien entre ellas, pasará antes por nuestra máquina (es necesario habilitar el reenvío de paquetes en el fichero *sysctl.conf* para que nuestra máquina reenvíe el tráfico a la máquina destino y así no se produzca una denegación de servicio).

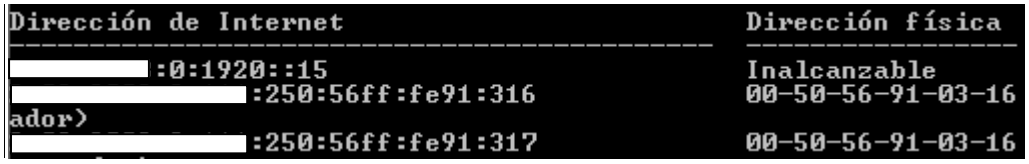

Figura 5.13. Tabla de vecinos de la máquina atacada

Fuzzing IPv6

Ante un ataque de *fuzzing*, se ha podido observar que de forma general el equipo Windows 7 realiza un tratamiento correcto de los paquetes malformados.

El equipo maneja bien el *fuzzing* sobre la cabecera IPv6. Resuelve bien el trato con todo tipo de valores de los campos *Version, Traffic Class, Flow Label* (la máquina no soporta el uso de clases de tráfico ni etiquetas), *Length* y *Hop Limit*. En

cuanto al campo *Next Header* el equipo no responde mediante mensajes ICMPv6 *Parameter Problem* como se recomienda en [38]. También maneja bien los paquetes cuando se realiza *fuzzing* conjunto de todos los campo de la cabecera. No se aprecia un aumento en el uso de recursos ni degradación en el estado de la conexión.

En cuanto a las pruebas de *fuzzing* realizadas sobre las cabeceras de extensión, se vuelve a observar un buen comportamiento en el tratamiento de los paquetes malformados. El único comportamiento no recomendado encontrado se produce en el manejo del número y orden de las cabeceras de extensión en un determinado caso. El equipo trata como correcto cualquier paquete que o bien no contenga la cabecera *Hop-By-Hop* o bien la contenga en primer lugar, sin importar el número, tipo y orden de las cabeceras de extensión que se sitúen después en el paquete, con lo que la implementación del protocolo no sigue las recomendaciones que se hacen en [6]. Este comportamiento puede ser utilizado maliciosamente para la creación de canales encubiertos de información.

Mediante las pruebas de *fuzzing* aplicadas al mecanismo de fragmentación se comprueba que la máquina gestiona correctamente los paquetes fragmentados, así como las situaciones inesperadas a las que se le somete mediante la herramienta Scapyst.

Por último, el equipo presenta un buen comportamiento ante pruebas de fuzzing completas, es decir, cuando se aplica el fuzzing sobre todos los campos y opciones de forma conjunta.

CAPITULO 6

CONCLUSIONES Y TRABAJO FUTURO

CONCLUSIONES Y TRABAJO FUTURO

Este capítulo se dedica a explicar por un lado las conclusiones que se han obtenido después de la realización de este Proyecto Fin de Carrera. Por otro lado, dado que el tema tratado en este documento es muy amplio, propondremos una serie de líneas de trabajo que sería conveniente realizar en el futuro.

6.1 CONCLUSIONES

En primer lugar, queda claro que la implantación del protocolo IPv6 es un reto tanto para la industria como para los usuarios de Internet. El cambio de protocolo trae de la mano nuevos riesgos de seguridad, además de mantener los ya existentes con IPv4. IPv6 no sólo cambia la longitud de las direcciones IP para permitir la introducción de nuevos dispositivos a la red, sino que presenta nuevos mecanismos de configuración en protocolos como ICMPv6 o DHCPv6 y una nueva forma de introducir las opciones mediante cabeceras de extensión.

El análisis de seguridad realizado indica que a corto plazo pueden surgir riesgos derivados del uso simultáneo del protocolo IPv4 y el protocolo IPv6. Por ello y dado que los principales sistemas operativos actuales tienen activado IPv6 por defecto, es importante concienciar de estos riesgos, siendo recomendable desactivar IPv6 cuando no se haga uso de él. A medio plazo pueden aparecer problemas en los mecanismos de control de acceso basados en direcciones IP y riesgos de privacidad derivados de la desaparición de NATs. En estos casos es recomendable establecer una buena política de privacidad y formar a los administradores de red en el protocolo IPv6 para la aplicación de estas. La ampliación del tamaño de las direcciones IP con IPv6 facilitará el crecimiento del número de dispositivos conectados a internet, lo que supone un riesgo a largo plazo debido a la probable aparición de nuevos ataques distribuidos que verán multiplicada su potencia.

Como se ha dicho, el protocolo IPv6 introduce nuevas características que mejoran algunas deficiencias del protocolo IPv4. Sin embargo, estas características han dado pie a la aparición de nuevos ataques exclusivos de IPv6 en un periodo corto de tiempo. Por ello, es muy importante el uso de los mecanismos de seguridad que ofrece IPv6, como IPSEC, y que en la actualidad no son prácticamente utilizados. De no ser así los equipos tanto de red como de usuario, estarían comprometidos en una red IPv6.

Estos problemas de seguridad presentes IPv6 son la lógica consecuencia de la implantación de un nuevo protocolo de comunicaciones, que por otro lado aporta significativas mejoras al protocolo actualmente utilizado. Es aquí donde radica la importancia de realizar análisis de seguridad sobre los equipos que implementen IPv6. Para realizar estos análisis es necesario el uso de herramientas que nos permitan conocer el nivel de seguridad de nuestras redes. Este es el objetivo que se ha querido alcanzar con el desarrollo de la herramienta Scapyst, facilitar el análisis de seguridad de los equipos que pertenecen a la red IPv6, no sólo a profesionales de la seguridad, sino a cualquier persona interesada en mantener su red o equipos seguros.

Las pruebas realizadas sobre las pilas IPv6 de uso mayoritario han desvelado que aunque los fabricantes realizan un gran esfuerzo en la securización de sus implementaciones del protocolo, aún existen vulnerabilidades que pueden ser explotadas por usuarios maliciosos. Por ello, es importante volver a enfatizar la importancia del uso de sistemas de seguridad disponibles como IPSEC. Además, debido a lo novedoso de este protocolo resulta muy importante que las personas que administran redes de comunicaciones estén bien formadas en el nuevo protocolo, siendo conscientes de los riesgos que conlleva su utilización y conociendo los mecanismos de seguridad que deben aplicar.

Por último, destacar que la realización de pruebas de seguridad sobre cualquier nueva tecnología es muy importante para la protección de los usuarios de dicha tecnología. Esto queda patente observando los protocolos de seguridad que importantes empresas aplican a sus productos, dentro de los cuales encontramos algunas de las técnicas que se han descrito en este documento como el *packet crafting* y el *fuzzing.*

6.2 TRABAJO FUTURO

El mantenimiento y estudio de la seguridad en redes de comunicaciones es un proceso continuo en el que van apareciendo nuevas vulnerabilidades y riesgos de seguridad. Por ello es importante mantener una buena formación en los protocolos utilizados (entre ellos IPv6) del personal técnico que trabaja en el mantenimiento de estas redes. Por lo tanto, la primera línea futura de trabajo sería el detectar los nuevos riesgos que irán apareciendo a medida que se incremente la utilización del protocolo IPv6, analizándolos y estudiando medidas de protección frente a ellos.

Por otro lado, se pueden enumerar una serie de características que en un principio se habían pensado implementar en la herramienta Scapyst, pero que por falta de tiempo no ha sido posible llevar a cabo:

- Implementación de capas de distintos protocolos: aunque la herramienta Scapyst proporciona los protocolos básicos de IPv6, algunos tipos de mensajes aún no están implementados (como algunos mensajes DHCPv6, movilidad IPv6). Por lo tanto un primer paso sería completar estos protocolos e incluso implementar algunos nuevos.
- Incluir nuevos ataques: a medida que vayan apareciendo nuevos ataques que utilicen el protocolo IPv6, podrían incluirse en el *framework* ya existente.
- Implementación de nuevas pruebas de *fuzzing*: implementar las pruebas de *fuzzing* para otros protocolos relacionados con IPv6 como ICMPv6, DHCPv6.
- Creación de protocolos personalizados: la limitación en cuanto a la inclusión de nuevos protocolos en la herramienta viene limitada por el número de capas que están desarrolladas en Scapy. Sin embargo, Scapy nos permite desarrollar nuevos protocolos definiendo el número, longitud y tipo de los campos que lo forman, además de las acciones que se podrían realizar con ellos. Sería interesante incluir en la herramienta Scapyst un *framework* que permitiera crear estos protocolos de forma intuitiva para después poder utilizarlos en la propia herramienta.

Como última línea de trabajo, sería necesario ampliar las pruebas realizadas a otros equipos que puedan estar presentes en la red. Cuando se habla de otros equipos, no quiere decir únicamente otros sistemas operativos, sino otras máquinas que no sean de usuario como equipos presentes en el núcleo de red. La realización de estas pruebas es un punto muy importante ya que si el control de seguridad en equipos de usuario es importante, no lo es menos en equipos de red.

CAPITULO 7

PRESUPUESTO

PRESUPUESTO

Este capítulo final presenta el coste relativo exclusivamente al trabajo realizado y presentado en esta memoria, sin tener en cuenta el otro tipo de costes asociados como por ejemplo costes de equipos de red, acceso o instalación. Por lo tanto, se detalla el coste del estudio a nivel personal, teniendo en cuenta que este trabajo ha sido realizado por una única persona.

7.1 COSTE DEL MATERIAL DE TRABAJO

En este apartado se presentan los costes asociados al material utilizado para la realización de las actividades que se han llevado a cabo en el presente Proyecto Fin de Carrera: estudio teórico, documentación, desarrollo de la herramienta y realización de las pruebas.

En primer lugar, ha de tenerse en cuenta el coste procedente del entorno de trabajo en el que se ha realizado el proyecto. La gran mayoría de las actividades se han realizado utilizando las instalaciones de Telefónica Digital. El gasto mensual medio derivado de la utilización de estas instalaciones, teniendo en cuenta el coste del puesto de trabajo, material de oficina, climatización, agua, luz y otros aspectos de acondicionamiento es de 120 euros por mes. Para los 8 meses de duración del proyecto es de 8 meses (Noviembre 2011-Junio 2012), por lo que el gasto total en este concepto es de 960 euros.

Además, ha de incluirse el coste del equipo informático utilizado para la realización del trabajo. El equipo usado ha sido un MacBook Pro Intel Core 2 Duo, cuyo coste es de 1.490 euros. Teniendo en cuenta que la depreciación de estos equipos suele tomarse del 20% por año y que la duración del proyecto ha sido de 8 meses, el coste en concepto de equipos informático es de 199 euros dado que el valor residual del producto después de estos 8 meses es de 1.291 euros. Indicar que los costes asociados al software utilizado están incluidos en el precio del equipo ya que el software adicional utilizado es completamente gratuito.

7.2 COSTE DE HONORARIOS

Dado que la duración del proyecto ha sido de 8 meses con una dedicación de 40 horas semanales, se calcula un total de 1280 horas. El precio por hora de un ingeniero responde a acuerdos de libre mercado entre particulares y profesionales. Anteriormente, el Colegio de Oficial de Ingenieros de Telecomunicaciones publicaba de forma orientativa los honorarios por hora de un ingeniero. Debido a la nueva normativa europea, esta información ya no se encuentra disponible y es por ello que utilizaremos como referencia un valor anterior de 75 euros por hora de trabajo del ingeniero. Utilizando el número de horas calculado, obtenemos un coste de 96.000 euros.

Por último, se calculan los costes asociados los directores del proyecto. La estimación que se hace para estos costes es del 7% de los honorarios derivados del trabajo del ingeniero. Teniendo en cuenta que estos honorarios son los calculados anteriormente (96.000 euros) y que el proyecto ha tenido dos directores (uno por parte de la universidad y otro por parte de la empresa), el coste asociado a este concepto es de 13.440 euros.

7.3 COSTE TOTAL

En la siguiente tabla se detallan todos los costes, incluyendo el valor del coste total del proyecto.

Concepto	Coste	Cantidad	Total
Entorno de Trabajo	120 €/mes	8 meses	960 €
Equipo Informático	199 €	1	199 €
Ingeniero de Proyecto	75 €/hora	1280 horas	96.000 €
Director de Proyecto (1)	7%	96.000 €	6.720 €
Director de Proyecto (2)	7%	96.000 €	6.720 €
Total sin IVA			110.599 €
Total con IVA (18%)			**130.506,82 €**

BIBLIOGRAFIA

[1] A-M. Juuso. T. Rontti, J-M. Tirila. “Securing next generation networks by fuzzing protocol implementations”. Telecom World (ITU WT), 2011 Technical Symposium at ITU, 24-27 Oct. 2011, pp 7-11.

[2] J. Hoagland. “Windows vista network attack surface analysis”. http://www.symantec.com/avcenter/reference/Vista_Network_Attack_Surface_RTM.pdf

[3] http://www.microsoft.com/security/sdl/discover/verification.aspx. Accedido en Marzo 2012.

[4] J. Postel. “RFC 791: Internet Protocol”. Septiembre 1981.

[5] S. Deering, R. Hinden. “RFC 1883: Internet Protocol, Version 6 (IPv6)”. Diciembre 1995.

[6] S. Deering, R. Hinden. “RFC 1883: Internet Protocol, Version 6 (IPv6)”. Diciembre 1998.

[7] http://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xml. Accedido en Marzo 2012.

[8] C. Huitema, B. Carpenter. “RFC 3879: Deprecating Site Local Addresses”. Septiembre 2004.

[9] R. Hinden, B. Haberman. “RFC 4193: Unique Local IPv6 Unicast Addresses”. Octubre 2005.

[10] R. Hinden, S. Deering. “RFC 4291: IP Version 6 Addressing Architecturte”. Febrero 2006.

[11] Hagen, S: “IPv6 Essentials, 2nd edition” O’Reilly, Mayo 2006.

[12] http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xml. Accedido en Marzo 2012.

[13] J. Abley, P. Savola, G. Neville-Neil. “RFC 5095: Deprecation of Type 0 Routing Headers in IPv6”. Diciembre 2007.

[14] C. Perkins, D. Johnson. ”RFC 6275: Mobility support in IPv6”. Julio 2011.

[15] S. Kent, R. Atkinson. “RFC 2402: IP Authentication Header”. Noviembre 1998.

[16] S. Kent, R. Atkinson. “RFC 2406: IP Encapsulating Security Payload (ESP)”. Noviembre 1998.

[17] A. Conta, S. Deering, M.Gupta. “RFC 4443: Internet Control Message Protocol (ICMPv6) for the Internet Protocolo Version 6 (IPv6) Specification”. Marzo 2006.

[18] P. Loshin: “IPv6 Theory, Protocol and Practice”. 2nd Edition. Elsevier, 2004, Sebastopol (USA).

[19] T. Narten, E. Nordmark, W. Simpson, H. Soliman. “RFC 4861: Neighbor Discovery for IP version 6 (IPv6)”. Septiembre 2007.

[20] S. Thompson, T. Narten, T. Jinmei. “RFC 4862: IPv6 Stateless Address Autoconfiguration”. Septiembre 2007.

[21] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, M. Carney. “RFC 3315: Dynamic Host Configuration Protocol for IPv6 (DHCPv6)”. Julio 2003.

[22] H. Schulzrinne. “RFC 3319: Dynamic Host Configuration Protocol (DHCPv6) Options for Session Initation Protocol (SIP) Servers”. Julio 2003.

[23] O. Troam, R. Droms. “RFC 3633: IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6”. Diciembre 2003.

[24] R. Droms. “RFC 3646: DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6) “. Diciembre 2003.

[25] S. Thompson, C. Huitema. “RFC 1886: DNS Extensions to support IP version 6”. Diciembre 1995.

[26] M. Crawford, C. Huitema. “RFC 2874: DNS Extensions to Support IPv6 Address Aggregation and Renumbering”. Julio 2000.

[27] R. Bush, A. Durand, B. Fink, O. Gudmundsson, T. Hain. “RFC 3363: Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS)”. Agosto 2002.

[28] A. Takanen, J. DeMott, C. Miller: “Fuzzing for Software Security Testing and Quality Assurance”. Artech House, 2008, Norwood (USA).

[29] R. McNally, K. Yiu, D. Grove, D. Gerhardy: “Fuzzing: The State of the Art”. DSTO Defence Science and Technology Organisation, 2012, Edinburgh (Australia).

[30] http://www.secdev.org/projects/scapy/. Accedido en Abril 2012.

[31] S. Hogg, E. Vyncke: “IPv6 Security”. Cisco Press, 2008, Indianapolis (USA).

[32] D. Prieto, R. Sánchez. “New Network Connectivity & New Malware/Phishing Cases”. Counter-eCrime Operations Summit (CeCOS VI). 25-27 Abril 2012, Praga (República Checa)

[33] T. Chown. “RFC 5157: IPv6 Implications for Network Scanning”. Marzo 2008.

[34] https://media.defcon.org/dc-19/presentations/Bowne/DEFCON-19-Bowne-Three-Generations-of-DoS-Attacks.pdf

[35] J.S. Wheeler. “IPv6 NDP Table Exhaustion Attack”. http://inconcepts.biz/~jsw/IPv6_NDP_Exhaustion.pdf

[36] J. Arkko, J. Kempf, B. Zill, P. Nikander. “RFC 3971: SEcure Neighbor Discovery (SEND)”. Marzo 2005.

[37] A. Takanen, D. Rajnovic. “Robustness Testing to Proactively Remove Security Flaws with a Cisco Case Study.” Cisco Systems Boiler Maker Conference, 25 Oct. 2005. Milpitas, California(USA).

[38] “IPv6 READY. Phase-1/Phase-2 Test Specification Core Protocols.Technical Document. Revision 4.0.6”. University of New Hampshire InterOperability Laboratory, Yokogawa Electric Corporation and IPv6 Forum. 2009.