Universidad Zaragoza
1542

Trabajo Fin de Grado

Configuración dinámica de túneles seguros LISP-SM

Dynamic LISP-SM secure tunnels configuration

Autor

Alberto Parra Romero

Director

Julián Fernández Navajas

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Alberto Parra Romero,

con nº de DNI 16634059E en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Configuración dinámica de túneles seguros LISP-SM,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 28 de Enero de 2019

Fdo: Alberto Parra

Agradecimientos:

A Julián y Rubén, por su ayuda y consejo.

A mis padres, mi hermano y mis amigos por su apoyo continuo.

CONFIGURACIÓN DINÁMICA DE TÚNELES SEGUROS LISP-SM

RESUMEN

Este TFG (Trabajo Fin de Grado) se ha fundamentado en el proyecto TISFIBE en el que trabajó el grupo CeNITEQ del i3a para añadir las nuevas funcionalidades de seguridad requeridas al proyecto ya desarrollado en GitHub. Para ello, se ha trabajado en un entorno virtualizado con un escenario de red montado, sobre el cuál se han realizado pruebas para testear el funcionamiento correcto del código añadido.

El proyecto tiene como base los protocolos LISP y Simplemux, que surgieron como posibles soluciones a problemas actuales como la escalabilidad y la movilidad en la arquitectura IP, en el caso de LISP; y a la búsqueda de una mejora de eficiencia en situaciones en las cuáles el *overhead* de las cabeceras tenga mucho impacto sobre la comunicación establecida, en el caso de Simplemux. Con la sinergia (LISP-SM) de ambos protocolos se pretende mejorar la Calidad de Servicio (QoS) en entornos difíciles.

Debido a que la seguridad es un aspecto muy importante en las comunicaciones y una preocupación de los usuarios, se ha añadido la posibilidad de securización de los datos enviados a través del túnel LISP-SM en el entorno virtual. Debido al *overhead* que introduce la seguridad, se da la posibilidad al usuario de elegir entre tres modos diferentes descritos en la memoria.

A lo largo de la memoria se describen las herramientas utilizadas, se realiza una descripción del código añadido y se presentan las pruebas a las que ha sido sometido el escenario. Se analizan los test realizados, explicando los encapsulados realizados para facilitar el entendimiento de la multiplexión. Se comprobará el funcionamiento dinámico de la comunicación, cambiando valores en su transcurso y analizando las implicaciones del uso de cada modo con diferentes valores de multiplexión.

Finalmente se recogen todas las conclusiones y posibles continuaciones de este TFG, así como se realiza la subida del código al repositorio de GitHub del proyecto inicial.

ÍNDICE

1. Introducción 1

1.1 Introducción y Motivación 1

1.2 Contexto 1

1.3 Objetivos 2

1.4 Organización de la memoria 2

2. Estado del arte 3

2.1 Problemática 3

2.2 LISP 4

2.3 Simplemux 5

2.4 IPsec 6

3. Herramientas 8

3.1 LISP-SM 8

3.2 VMware vsphere 8

3.3 IPsec-tools 9

3.4 D-ITG 9

3.5 Tcpdump 10

3.6 Wireshark 11

3.7 Conexión remota: MobaXterm y WinSCP 11

4. Escenario de red 13

5. Análisis del código 15

5.1 Inclusión de IPsec 15

5.2 Pruebas a realizar 18

6. Resultados 19

6.1 Pruebas modo=0 19

6.2 Pruebas modo=1 27

6.3 Pruebas modo=2 32

7. Subida a GitHub 38

8. Conclusiones y líneas futuras.....39

8.1. Conclusiones 39

8.2 Líneas Futuras 39

9. Referencias y Bibliografía 40

9.1 Referencias 40

9.2 Bibliografía 40

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Representación de un datagrama de tamaño medio 3

Ilustración 2: Representación de un posible datagrama de un servicio en tiempo real 3

Ilustración 3: Ejemplo de un sistema con LISP implantado. Imagen recuperada de: https://pdfs.semanticscholar.org/0219/24b26bf977843adf2bddc6d91dd5a15aff54.pdf 5

Ilustración 4: Representación de un paquete multiplexado con Simplemux 6

Ilustración 5: Representación de la encapsulación de un datagrama usando ESP en modo túnel y transporte. Recuperado de: http://blackspiral.org/docs/pfc/itis/node7.html 7

Ilustración 6: Representación de la encapsulación de un datagrama usando AH en modo túnel y transporte. Recuperado de: http://ingenieriadelaseguridad2013.blogspot.com/p/ip-sec.html 7

Ilustración 7: Modo de ejecutar ITGSend. Recuperado de http://www.grid.unina.it/software/ITG/manual/ 10

Ilustración 8: Modo de ejecutar ITGRecv. Recuperado de http://www.grid.unina.it/software/ITG/manual/ 10

Ilustración 9: Captura de MobaXterm 11

Ilustración 10: Captura de WinSCP 12

Ilustración 11: Representación esquemática del escenario con direcciones y nombres 13

Ilustración 12: Representación del intercambio de tramas en LISP para la resolución de nombres 13

Ilustración 13: Representación del escenario final 14

Ilustración 14: Captura de estadísticas de ITGDec en PC1 y PC2 para Modo=0 y Mux=2 19

Ilustración 15: Captura del envío de datos desde PC1 20

Ilustración 16: Captura del envío de datos en el túnel LISP-SM 20

Ilustración 17: Estadísticas de Wireshark sobre la captura 20

Ilustración 18: Paquete en el túnel extraído de la captura con modo 0 y multiplexión = 2 21

Ilustración 19: Representación de la primera cabecera de Simplemux para paquetes mayores de 65 bytes y menores de 8192, rescatada de https://tools.ietf.org/html/draft-saldana-tsvwg-simplemux-09#page-5 22

Ilustración 20: Cabecera Simplemux para paquetes mayores de 128 bytes y menores de 2^{14} bytes, rescatada de su Internet-Draft, https://tools.ietf.org/html/draft-saldana-tsvwg-simplemux-10 22

Ilustración 21: Captura de estadísticas de ITGDec en PC1 y PC2 para Modo=0 y Mux = 4 23

Ilustración 22: Captura de datos enviados por PC1 23

Ilustración 23: Captura del envío de datos en el túnel LISP-SM 24
Ilustración 24: Estadísticas de Wireshark sobre la captura 24
Ilustración 25: Captura de paquete en modo=0 con Mux=4 25
Ilustración 26: Captura del tráfico seguro generado desde PC1 26
Ilustración 27: Captura del tráfico en el túnel 26
Ilustración 28: Captura de las estadísticas realizadas por ITGDec en PC1 y PC2 27
Ilustración 29: Tráfico generado en PC1 27
Ilustración 30: Tráfico securizado por IPsec en el túnel 28
Ilustración 31: Captura de un paquete en Modo=1 y Mux=2 28
Ilustración 32: Ancho de banda utilizado en modo 0 y Mux = 4. 29
Ilustración 33: Ancho de banda utilizado en modo 1 y Mux = 4. 30
Ilustración 34: Gráfica generada por la comunicación con los cambios dinámicos 31
Ilustración 35: Uso de puertos en la prueba de cambios dinámicos 31
Ilustración 36: Captura que muestra los dos flujos de tráfico generados desde PC1 32
Ilustración 37: Captura del tráfico dentro del túnel LISP-SM con modo 2 y ratio = 0 33
Ilustración 38: Estadísticas de uso de puertos obtenidas de Wireshark 33
Ilustración 39: Captura del debug de xTR1 para el modo 2 y ratio = 0 34
Ilustración 40: Captura del debug de xTR1 para el modo 2 y ratio = 0 34
Ilustración 41: Tráfico generado desde PC1 35
Ilustración 42: Tráfico con modo 2 y ratio = 1 en el túnel 35
Ilustración 43: Estadísticas de puertos de Wireshark 36
Ilustración 44: Captura del debug de xTR1 para el modo 2 y ratio = 1 36
Ilustración 45: Debug de xTR1 para el modo 2 y ratio = 1 37
Ilustración 46: Código subido al repositorio de GitHub 38

1. INTRODUCCIÓN

1.1 INTRODUCCIÓN Y MOTIVACIÓN

En la sociedad tecnológica que vivimos, los usuarios demandan mayor velocidad y calidad en las comunicaciones, ya sea entre terminales móviles u ordenadores, sin detenerse a pensar en las dificultades tecnológicas subyacentes. Esto afecta al diseño correcto de las infraestructuras de red correspondientes, para solucionar problemas tales como la escalabilidad, movilidad, etc.

En los últimos años, hemos sido testigos de la evolución constante de las redes de comunicaciones móviles. Desde el 2013, en España hemos visto como el 3G se quedaba obsoleto, y se sustituyó por el 4G, que parece que está viviendo sus últimos días. Por otro lado, si ponemos el foco sobre las redes de acceso cableado a Internet, podemos darnos cuenta de que la fibra óptica ha llegado ya a la misma casa del cliente con las ventajas que ello trae consigo. Estos avances en los accesos hacen creer al usuario que las comunicaciones han mejorado a su vez proporcionalmente a ellos, y esto no es cierto en todas las situaciones.

Por otra parte, el crecimiento producido en los últimos años en las redes de comunicación nos lleva a otra preocupación de los usuarios, esta no es otra que la seguridad. Según el *Informe de la Sociedad de la Información en España de 2017*, el 85.6% de los usuarios dejaría de utilizar un servicio que no cumple los requisitos de seguridad, ya que temen una pérdida de privacidad en sus datos. En este caso, la seguridad está referida a datos y sistemas de almacenamiento de los mismos. A su vez, el aumento del número de redes *wireless*, o inalámbricas, que se fundamentan en el reparto de un canal común para todos los usuarios, pone en relieve otro aspecto o vertiente de la seguridad muy importante, la seguridad en las comunicaciones. Tal es la inquietud existente entre los usuarios sobre este tema, que, por ejemplo, la aplicación de mensajería instantánea más usada, WhatsApp, implantó un cifrado extremo a extremo entre terminales, para tratar de eliminar esta intranquilidad.

Teniendo en cuenta estos dos aspectos, la velocidad y la seguridad, parece interesante encontrar una solución nueva que satisfaga estas demandas de parte de los usuarios y que a la vez resuelva problemáticas actuales como la movilidad o la escalabilidad. Pero, ¿se puede resolver de forma sencilla? Con este Trabajo Fin de Grado, se tratará de aportar una solución para mejorar los aspectos de escalabilidad, movilidad y seguridad en las comunicaciones.

1.2 CONTEXTO

Teniendo presente lo expuesto en la introducción, el Plan Estatal de Investigación Científica y Técnica y de Innovación 2013-2016, aprobó el proyecto TISFIBE (Túneles inteligentes y seguros para flujos IP con baja eficiencia), en el cual está fundamentado este TFG, y en el que trabajaron los investigadores del grupo CeNITEQ del I3A. Su objetivo principal era la optimización y securización de flujos IP en entornos de red que presentan cuellos de botella.

A raíz de este proyecto, se creó el repositorio de GitHub "Lispmob-with-simplemux". Dicho proyecto está escrito en lenguaje de programación C y permite la creación de un túnel LISP, Locator/ID Separation Protocol, con las funcionalidades de Simplemux añadidas (a partir de ahora, se referirá a este túnel como LISP-SM). LISP se trata tanto de una nueva arquitectura de enrutamiento como de un protocolo desarrollado por Cisco Systems para la mejora de la escalabilidad y movilidad. Simplemux, se trata de un protocolo capaz de multiplexar paquetes de distintos protocolos en uno nuevo. La utilización conjunta de ambos protocolos da respuesta a los problemas de escalabilidad y movilidad.

LISP-SM puede ser configurado con diferentes parámetros, siendo posible elegir el número de paquetes a multiplexar, las direcciones origen y destino para la creación del túnel de multiplexión, el tiempo de espera máximo, *timeout*, para que lleguen paquetes y formar el paquete multiplexado, etc. Además, está desarrollado para poder cambiar estos parámetros mientras una comunicación está en curso. Partiendo de esta base e intentando seguir el estilo de programación, se añadirán las nuevas funcionalidades planteadas para dar respuesta a los problemas de seguridad.

1.3 OBJETIVOS

El objetivo de este Trabajo Fin de Grado es la creación de túneles seguros LISP-SM de forma dinámica de tal forma que puedan mejorar la eficiencia en infraestructuras de red que presentan cuellos de botella. Se probará el funcionamiento correcto en distintas casuísticas y a partir de ellas, se valorará la mejora aportada por los protocolos en el escenario de red. Todo ello se probará en un entorno virtualizado que facilite la repetitividad de las pruebas.

1.4 ORGANIZACIÓN DE LA MEMORIA

La memoria se compone de los siguientes apartados:

1. En este primer bloque, se pondrá en situación al lector sobre el marco del trabajo y sus objetivos.
2. En el segundo capítulo, se presenta la problemática que ha dado origen al desarrollo de los protocolos usados y una breve descripción de estos.
3. En esta sección se procede a describir las herramientas y programas usados para la realización de este TFG.
4. En este cuarto capítulo, se describe el escenario de red sobre el cuál se ha trabajado y el proceso de instalación del *software* usado.
5. Durante este apartado, se explicarán las funcionalidades añadidas al código base, así como las dificultades encontradas. Además, se explican las pruebas realizadas sobre dicho escenario para validar el código.
6. En este capítulo, se presentan los resultados obtenidos y un análisis de las capturas realizadas.
7. Subida del código a GitHub.
8. En este apartado, se recogen las conclusiones obtenidas y se presentan ideas para la continuación de este Trabajo.
9. Referencias y bibliografía

2. ESTADO DEL ARTE

2.1 PROBLEMÁTICA

En la introducción se ha mencionado la exigencia por parte de los usuarios de mayores velocidades en las comunicaciones. Si concretamos un poco más, actualmente los servicios en tiempo real forman parte del día a día de la mayoría de personas; por ejemplo, la telefonía IP (VoIP), videoconferencias (Skype, Whatsapp, Facebook) o los videojuegos (E-Sports y su crecimiento actual). Este tipo de aplicaciones necesitan un tratamiento de sus datos parecido, en tanto en cuanto, ciertos parámetros de QoS (Calidad de servicio) son vitales para mantener una comunicación de garantías. Características en las comunicaciones como el retardo (*delay*), o el *jitter* (diferencias en el retardo de los paquetes) no deben superar ciertos valores si el objetivo final es satisfacer las necesidades del usuario.

A pesar de sus evidentes diferencias, presentan muchas similitudes en las características del tráfico de datos que generan. Se puede hablar de que son generadoras de paquetes de datos pequeños.

¿Qué implicaciones tiene un tráfico de paquetes de pequeño tamaño? La eficiencia empeora, como veremos a continuación.

Ilustración 1: Representación de un datagrama de tamaño medio

Llamamos eficiencia al ratio entre datos útiles y datos totales:

$$n = \frac{Datos\ Aplicación}{(H1 + H2 + H) + Datos\ Aplicación}$$

Ecuación 1: Ecuación de la eficiencia en la transmisión de datos

Como se ha mencionado, en servicios en tiempo real se usan paquetes pequeños ya que mejoran las condiciones de retardo en la red (menos tiempo en la creación del paquete, antes se produce el envío y por tanto la llegada). No solo este es el motivo, una posible pérdida (este tipo de servicios usan UDP, no orientado a conexión) de un paquete pequeño tendrá en consecuencia poca importancia.

Ilustración 2: Representación de un posible datagrama de un servicio en tiempo real

A simple vista, con el dibujo, ya se observa la pérdida de eficiencia. En este último caso se podría decir que estamos enviando la misma cantidad de datos útiles que de información de control, lo que supone una eficiencia del 50%. Este es el punto negativo de usar paquetes pequeños, el desperdicio de ancho de banda en el canal de comunicaciones. Si consiguiéramos un mejor aprovechamiento de dicho recurso, repercutiría en una mejora en la velocidad de las comunicaciones.

Por otra parte, en la actualidad se está trabajando para solucionar los grandes problemas de escalabilidad que está sufriendo el sistema de direccionamiento y enrutamiento de Internet. A raíz de esta situación, el IAB (Internet Architecture Board), organización responsable de la asignación de nuevas direcciones IP, celebró en Amsterdam en octubre de 2006 una conferencia llamada "Routing and Addressing Workshop" para buscar soluciones.

El continuo incremento de la población que conlleva una mayor cantidad de dispositivos IP, el *multi-homing*[1], la ingeniería de tráfico o las actuales políticas de enrutamiento están provocando un crecimiento insostenible de la tabla de enrutamiento de la DBZ[2] (Default Free Zone). Estos problemas se catalogaron como problemas de atención inmediata ya que, hasta entonces, no se habían encontrado soluciones efectivas, y se pensó que con el desarrollo y el despliegue de IPv6, el problema sería más difícil de manejar.

2.2 LISP

Se trata de un protocolo de encaminamiento y tunelización desarrollado por el IETF (*Internet Engineering Task Force*) que proporciona una forma nueva de enrutamiento y es compatible tanto con IPv4 como con IPv6.

LISP surgió como solución para mejorar el problema de la escalabilidad en el enrutamiento IP mencionado anteriormente. En dicho enrutamiento IP, una dirección relaciona quién con desde dónde se envía la información. Sin embargo, LISP trabaja con dos espacios de numeración nuevos y diferentes al direccionamiento IP, EIDs (*Endpoint IDentifiers*) asignados independientemente de la red a los *host* y RLOCs (*Routing LOCators*), asignados a los *router*, encargados del redireccionamiento de paquetes.

"Ambos, EIDs y RLOCs, son sintácticamente idénticos al direccionamiento IP, es la semántica de cómo se usa la que difiere". (Farinacci, Fuller, Meller y Lewis, 2013, p.4). LISP proporciona los mecanismos para el mapeo entre ambos espacios numéricos.

[1] Multi-homing: conectar un *host* a varias redes informáticas

[2] Default Free Zone: "recopilación de todos los sistemas autónomos de Internet (AS) que no requieren una ruta predeterminada para enrutar un paquete a cualquier destino" (Wikipedia, Zona de libre incumplimiento).

Los *host* no necesitan de cambio alguno, son transparentes al uso de LISP. Los LISP *router*, se encargarán de redireccionar los paquetes con dirección a un EID y crearán un túnel con dirección un RLOC. Cuando un *router* acepta paquetes que llevan en el campo dirección destino uno de sus RLOC, se denomina ETR (*Egress Tunnel Router*), mientras que si encapsula dicho paquete con una dirección RLOC, se habla de ITR (*Ingress Tunnel Router*). Existe un tercer término que será el utilizado de aquí en adelante en el Trabajo, xTR. Con este término, nos referimos al LISP *router* que actúa como extremo del túnel, ya que en el contexto en el que se habla no es importante el sentido de los datos.

Para conocer la ruta correcta es necesario introducir un nuevo elemento llamado MS/MR, que son las siglas de *Map Server/Map resolver*. El funcionamiento del protocolo es parecido al de DNS, ya que el MS/MR hace de base datos del mapeo entre EIDs y RLOCs. Lo primero que hace un *router* LISP es registrarse en el MS/MR mediante un mensaje *Map-Register*, donde se almacenan los EID de los *host* a los que da servicio. LISP añade bytes en la cabecera de los mensajes enviados, aumentando el *overhead* del paquete.

LISP utiliza el puerto 4341 para el envío de datos mientras que usa el 4342 para la información de control. A continuación se presenta un esquema de un sistema con LISP:

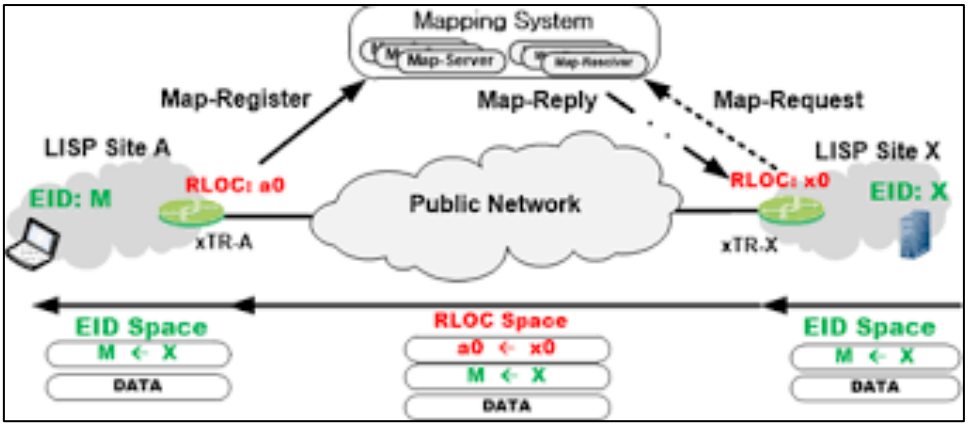


Ilustración 3: Ejemplo de un sistema con LISP implantado. Imagen recuperada de: https://pdfs.semanticscholar.org/0219/24b26bf977843adf2bddc6d91dd5a15aff54.pdf

2.3 SIMPLEMUX

Para solucionar el problema de la eficiencia, existen distintas técnicas de multiplexión de paquetes. Básicamente hay dos posibilidades: realizar una multiplexión con otros usuarios o bien hacerlo con el tráfico que genera un mismo usuario. Esta segunda opción, parece la más adecuada para paquetes con tiempos de generación muy pequeños, por ejemplo de voz, que transportan muy pocos bytes. Estos paquetes, pueden esperar a la llegada de otros (sin sobrepasar valores de retraso máximos en la comunicación), para ser enviados en paquete de mayor tamaño. Pero no siempre es posible realizarla y por tanto en una situación general querremos multiplexar información de varios usuarios.

Simplemux se trata por tanto de un protocolo de multiplexión de paquetes de varios usuarios. Se creó con el objetivo de mejorar el problema de la eficiencia mencionado

anteriormente multiplexando paquetes pequeños para formar otros más grandes, reduciendo de esa forma el número de paquetes que transcurre por la red y mejorando la eficiencia. Además, esta reducción del número de paquetes provoca, a su vez, un ahorro energético en los *router* intermedios que atraviese la comunicación, que tienen que procesar menos información.

Simplemux crea un túnel de paquetes que comparten la misma dirección destino y que comparten o no el mismo protocolo.

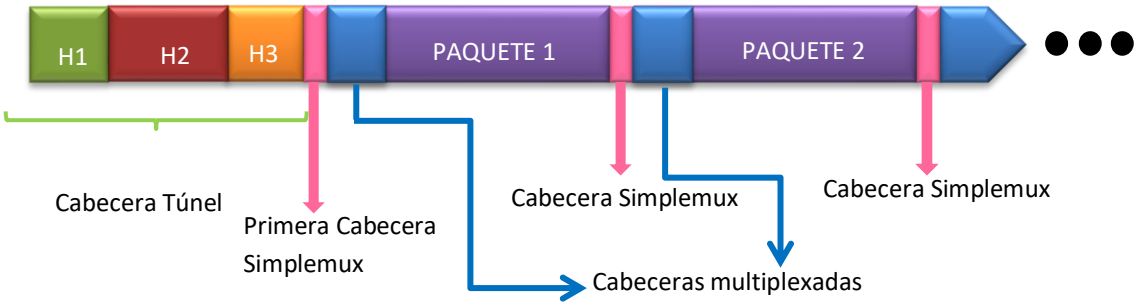


Ilustración 4: Representación de un paquete multiplexado con Simplemux

Como se aprecia en la ilustración anterior, Simplemux tiene dos tipos de cabeceras diferentes, dependiendo de su situación en el paquete. Se tratan de la *First Simplemux Header* y de la *Non-first Simplemux Headers*, es decir, la Cabecera Primera de Simplemux y las Cabeceras No Iniciales respectivamente. Estas cabeceras, varían en su formato ligeramente dependiendo del tamaño del paquete multiplexado. Por lo tanto, debido a esta inclusión de *bytes* extra, se debe tener en cuenta que Simplemux también genera ineficiencias. Además, se debe tener en cuenta, que su uso introduce un retraso por la espera de los paquetes para realizar la multiplexión y el procesado añadido. Es por esto, que se debe alcanzar un compromiso con los valores de multiplexión, que suelen ser bajos para obtener los mejores resultados.

En la implementación de GitHub, Lispmob-with-simplemux, Simplemux funciona en el puerto 4343 y no en el 4341, que se mantiene para los datos no multiplexados.

2.4 IPSEC

IPsec es un conjunto de protocolos y modos de funcionamiento creados con el objetivo de ofrecer seguridad en las comunicaciones cifrando los datos de nivel IP. Se puede utilizar en diferentes flujos de comunicación entre dos *host*, dos *router* (llamados *security gateways*) o entre *host* y *router*.

Protocolos:

- ESP: son las siglas de *Encapsulating security payload*. Protocolo número 50. Proporciona autenticación, integridad y confidencialidad en la transmisión de datos.

- AH: siglas de *Autenticaction Header*. Protocolo número 51. Ofrece integridad y autenticación aplicando un hash a la cabecera y datos del datagrama.

Modos:

- Transporte: solo se cifra la carga útil del paquete por lo que, el enrutamiento del paquete se mantiene intacto.

- Túnel: se cifra todo el paquete, incluida la cabecera. Posteriormente, se añade la cabecera ESP o AH y una nueva cabecera IP, para que el redireccionamiento sea posible.

Otro concepto importante para entender IPsec son las SA (*Security Association*) y las SP (*Security Policy*), almacenadas en bases de datos: SAD (*Security Association Database*) y SPD (*Security Policy Database*). Las SA son conexiones lógicas, con origen y destino, SPI (*Security Parameter Index*) que hace de identificador, y finalmente una orden sobre el tipo de cifrado y la clave a usar. Describe cómo cifrar el tráfico que necesite seguridad con ese mismo origen y destino. Sin embargo, las SP marcan qué hacer con el tráfico, tanto entrante como saliente. En ellas se decide qué direcciones IP, qué puertos y qué protocolos requieren seguridad, o cuáles no, qué cifrado se utiliza y el modo de funcionamiento.

De la misma forma que LISP y Simplemux, IPsec introduce bytes extra, por la inclusión de una cabecera extra, y en el caso de usar ESP, también un campo en la cola del paquete. Lo que provoca ineficiencias añadidas que podrán ser compensadas por la multiplexión.

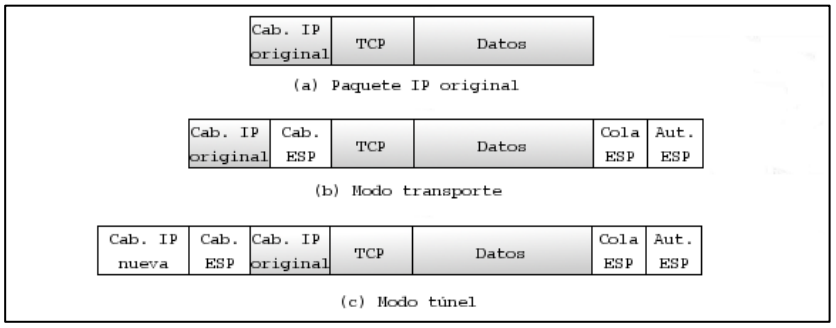


Ilustración 5: Representación de la encapsulación de un datagrama usando ESP en modo túnel y transporte. Recuperado de: http://blackspiral.org/docs/pfc/itis/node7.html

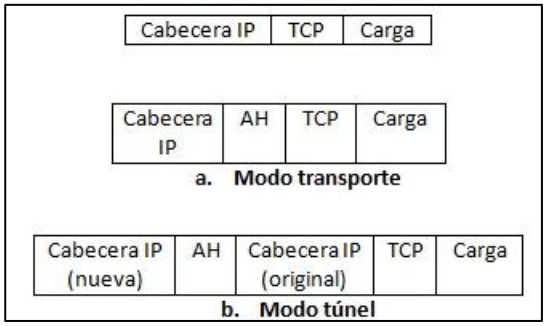


Ilustración 6: Representación de la encapsulación de un datagrama usando AH en modo túnel y transporte. Recuperado de: http://ingenieriadelaseguridad2013.blogspot.com/p/ip-sec.html

3. HERRAMIENTAS

En este capítulo se procede a describir las herramientas usadas para la realización de este Trabajo y la finalidad con que se han utilizado.

3.1 LISP-SM

LISP-SM (*Lispmob with Simplemux)* es la implementación del proyecto LISPmob con las funcionalidades de Simplemux añadidas y usada en los *router* del escenario para la creación de túneles. "El propósito consiste en enviar juntos un número pequeño de paquetes, los cuáles se encuentran en el *buffer* de un ITR, teniendo el mismo ETR de destino, en un mismo paquete. De esta forma compartirán la misma cabecera LISP y en consecuencia se producirá un ahorro en el ancho de banda así como una reducción del número de paquetes enviados por la red". Esta descripción se puede encontrar en el repositorio del proyecto: https://github.com/Simplemux/lispmob-with-simplemux.

Existen varios modos de funcionamiento para el dispositivo en el que se instale la herramienta, a elegir en el archivo de configuración de LISP-SM, *lispd.conf*. Actuar como xTR, MN (*Mobile node),* RTR (*Reencapsulating Tunnel Router)*, o las dos siguientes usadas en el escenario de red de este TFG: MS/MR, usado en el router que se encarga del mapeo de direcciones, y xTRSM en los LISP *router*, son diferentes posibilidades ofrecidas. A su vez es necesario configurar el Map Server y el Map Resolver, la dirección propia del *router*, la dirección de red de los *host* a los que se va a dar servicio…

En el mismo fichero también se encuentran los parámetros configurables para la creación del túnel de Simplemux: direcciones IP y de red de origen y de destino de los paquetes, direcciones LISP fuente y destino, número de paquetes a multiplexar, *threshold* o límite de *bytes* para la creación del paquete de multiplexión, *timeout* o tiempo máximo de espera…

El proceso de descarga e instalación de este repositorio se encuentra en el Anexo A.

3.2 VMWARE VSPHERE

VMware vSphere se trata de un *software* de virtualización y ha sido usado para la creación del entorno y escenario de pruebas. Permite la virtualización tanto de sistemas operativos como de *hardware*. Lanzado en 2009, su interfaz posibilita la creación y la gestión de máquinas virtuales.

El montaje realizado para este TFG, se encuentra explicado en el Anexo A. Para la descarga o consulta sobre sus muchos otros productos, la página de la empresa VMWare es: https://www.vmware.com/products/vsphere.html.

3.3 IPSEC-TOOLS

Para la inclusión de la seguridad en el escenario, se ha elegido Ipsec-tools, un paquete de Linux que permite el uso de políticas y la configuración, a través de un archivo de texto, de IPsec y sus funcionalidades. Contiene la herramienta Setkey, encargada de manipular y volcar las bases de datos mencionadas anteriormente, la SPD y la SAD.

Su descarga es posible a través de: http://ipsec-tools.sourceforge.net/. Su proceso de instalación se describe en el Anexo A.

3.4 D-ITG

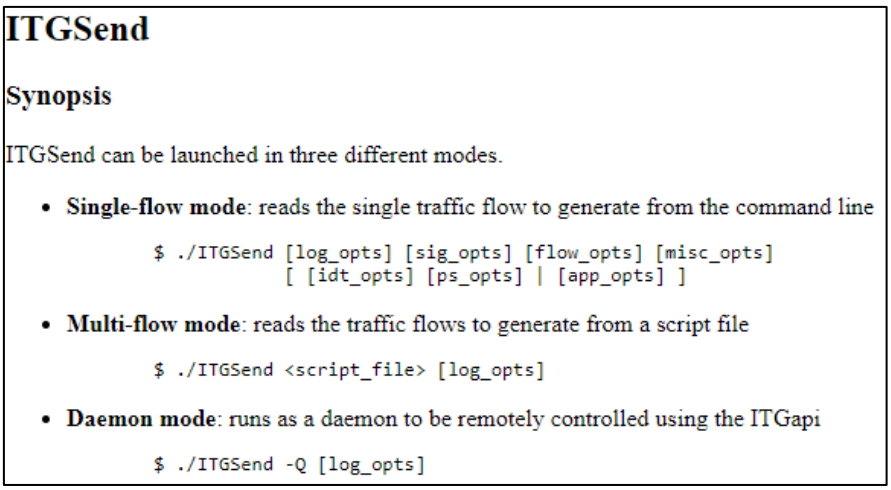
Para la generación del tráfico para las pruebas, se ha usado D-ITG, que son las siglas de "Distributed Internet Traffic Generator". Se trata de un *software* de código abierto y por tanto gratuito desarrollado por la Universidad de Nápoles Federico II. Permite generar tráfico a nivel de red, transporte y aplicación replicando características establecidas por el usuario. A su vez, también es capaz de medir estadísticas de red típicas como el jitter, retardo, etc.

La arquitectura de D-ITG está formado por tres componentes: ITGSend, ITGRecv e ITGDec. ITGSend es el *software* encargado de la generación de tráfico, mientras que ITGDec es el responsable de su recepción. Se puede generar un log voluntario para leer con ITGDec, en el que se recogen las estadísticas de la comunicación establecida.

Se ha optado por el uso de este *software* frente a otros, como pueda ser el caso de *iperf*, debido a su potencial en la caracterización del tráfico y a la posibilidad de generar más de un flujo distinto de datos. Además, otro punto a su favor, es que el envío de datos se realiza por otro puerto que la señalización (con *iperf* comparten el mismo canal) lo cual facilita el posterior análisis de las capturas.

Para generar un solo flujo de datos, es necesario lanzar primero por consola ITGRec en el receptor y luego ITGSend en el emisor con las características deseadas. Se puede establecer la duración de la comunicación, número de paquetes, retraso, tamaño de paquetes, protocolo…y por supuesto dirección y puerto destino. Por el contrario, para el envío de diferentes flujos de datos, es necesario generar un script, en el cual, cada flujo representará una línea del archivo.

La documentación sobre el uso y configuración de D-ITG se encuentra en: http://www.grid.unina.it/software/ITG/manual/ . En el Anexo A se describe el proceso de descarga e instalación de este *software*.



ITGSend

Synopsis

ITGSend can be launched in three different modes.

- **Single-flow mode**: reads the single traffic flow to generate from the command line

```
$ ./ITGSend [log_opts] [sig_opts] [flow_opts] [misc_opts]
            [ [idt_opts] [ps_opts] | [app_opts] ]
```

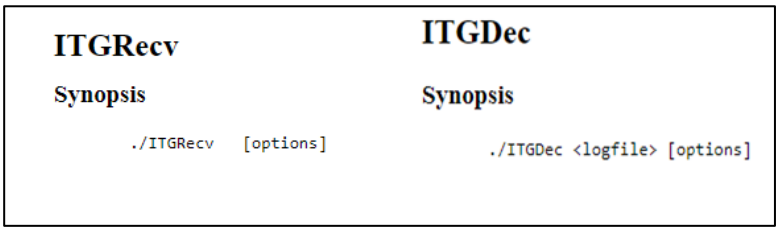
- **Multi-flow mode**: reads the traffic flows to generate from a script file

```
$ ./ITGSend <script_file> [log_opts]
```

- **Daemon mode**: runs as a daemon to be remotely controlled using the ITGapi

```
$ ./ITGSend -Q [log_opts]
```

Ilustración 7: Modo de ejecutar ITGSend. Recuperado de http://www.grid.unina.it/software/ITG/manual/



ITGRecv

Synopsis

```
./ITGRecv [options]
```

ITGDec

Synopsis

```
./ITGDec <logfile> [options]
```

Ilustración 8: Modo de ejecutar ITGRecv. Recuperado de http://www.grid.unina.it/software/ITG/manual/

3.5 TCPDUMP

Tcpdump fue inicialmente desarrollado en 1987 y en la actualidad es uno de los *sniffers* [3]más usados habitualmente. Se trata de una herramienta por consola compatible con prácticamente todos los sistemas operativos UNIX y que sirve para imprimir por pantalla el contenido de los paquetes entrantes o salientes de una interfaz de red en concreto. Permite también la creación y lectura de un archivo con los datos capturados con las opciones *–w* y *–r* respectivamente. Gracias a esta posibilidad, se han podido guardar las capturas de los tráficos generados para su posterior análisis. Tcpdump reconoce el establecimiento de filtros para la captura. Direcciones IP, MAC, puertos, (tanto en origen como destino) y protocolos son los más típicos. Su uso está muy extendido entre los administradores de red para la depuración del tráfico existente pero también entre atacantes informáticos, también llamados *hackers*, para el robo de contraseñas por ejemplo.

En el Anexo A se describen los pasos seguidos para su instalación. Su descarga es posible a través de: https://packages.debian.org/search?keywords=tcpdump mientras que, si se desea realizar una consulta a su manual para conocer todas las posibilidades que este *software* ofrece: https://www.tcpdump.org/manpages/tcpdump.1.html.

[3] Sniffer: herramienta analizadora de paquetes en una red

3.6 WIRESHARK

Wireshark se trata de otro analizador de protocolos. Es la continuación de un proyecto comenzado por Gerald Combs en 1998 y actualmente, ya han colaborado más de seiscientos voluntarios. Se ha usado para leer los archivos generados por Tcpdump, ya que incluye una interfaz gráfica con muchas opciones de filtrado, estadísticas de protocolos y puertos, etc.

La descarga de Wireshark es gratuita y posible a través del siguiente enlace: https://www.wireshark.org/#download.

3.7 CONEXIÓN REMOTA: MOBAXTERM Y WINSCP

Debido al uso de un escenario virtualizado, el acceso remoto a cada máquina virtual se ha realizado mediante MobaXterm. Se trata de una aplicación gratuita para Windows que aporta la posibilidad de uso de varios protocolos de acceso remoto, en este caso se ha usado SSH. Se ha optado por el uso de MobaXterm, por encima de Putty por ejemplo, debido a su interfaz de ventanas más amigable e intuitiva (incluye un sistema de ventanas X incorporado) y por consiguiente, la mejora en tiempo en la navegación por carpetas y ficheros. Incluye un editor propio de texto, que se ha usado para realizar cambios sencillos en el código, permite la ejecución simultánea de comandos en distintos terminales, etc.

Por otro lado, se ha complementado el uso de MobaXterm, con WinSCP, cliente SFTP, para la descarga y subida de archivos.

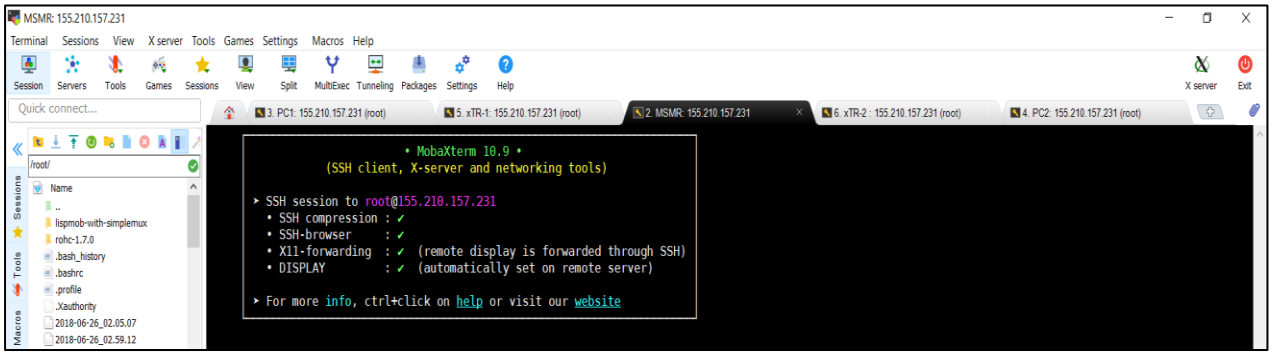


Ilustración 9: Captura de MobaXterm

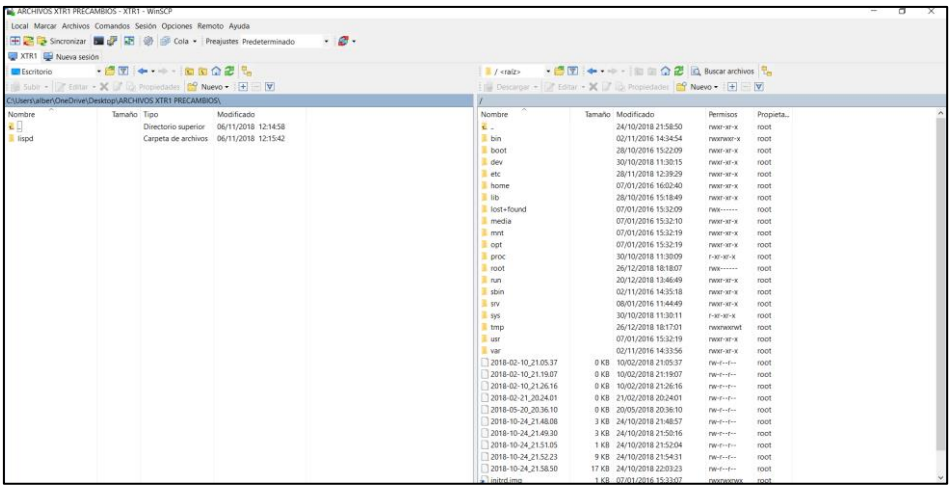


Ilustración 10: Captura de WinSCP

MobaXterm se encuentra disponible en: https://mobaxterm.mobatek.net/download.html mientras que WinSCP en: https://winscp.net/eng/download.php.

4. ESCENARIO DE RED

A continuación, se procede a describir el escenario de red virtualizado utilizado en este Trabajo con explicaciones complementarias para comprender el funcionamiento del mismo.

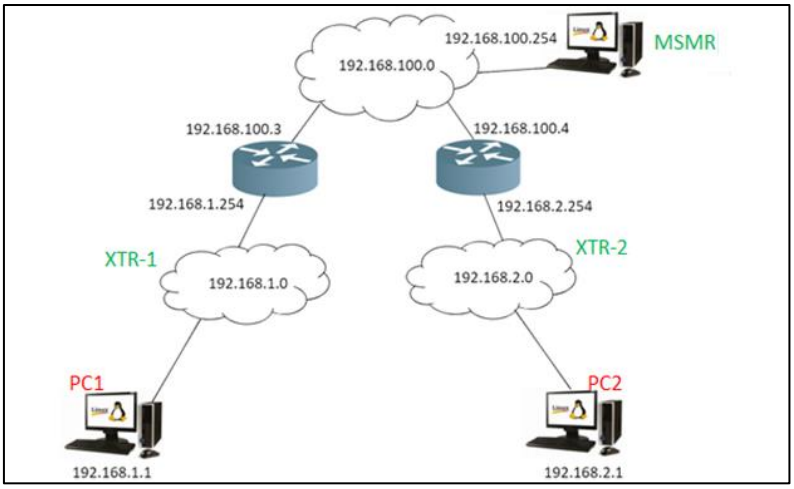


Ilustración 11: Representación esquemática del escenario con direcciones y nombres

Como se puede observar, consta de dos *host* en redes LAN distintas que harán de EIDs, entre los cuales se probarán distintos tráficos de datos. Para interconectar las redes, tenemos dos RLOCs, que harán el papel de xTR, ya que proporcionarán tráfico en ambos sentidos de la comunicación (serán tanto ITR como ETR). Para terminar, necesitaremos una tercera red que conecte ambos RLOC. En dicha red, tendremos un MS/MR, para realizar el control en la comunicación. Con este escenario de partida, el proceso de comunicación para el establecimiento del túnel LISP es el siguiente:

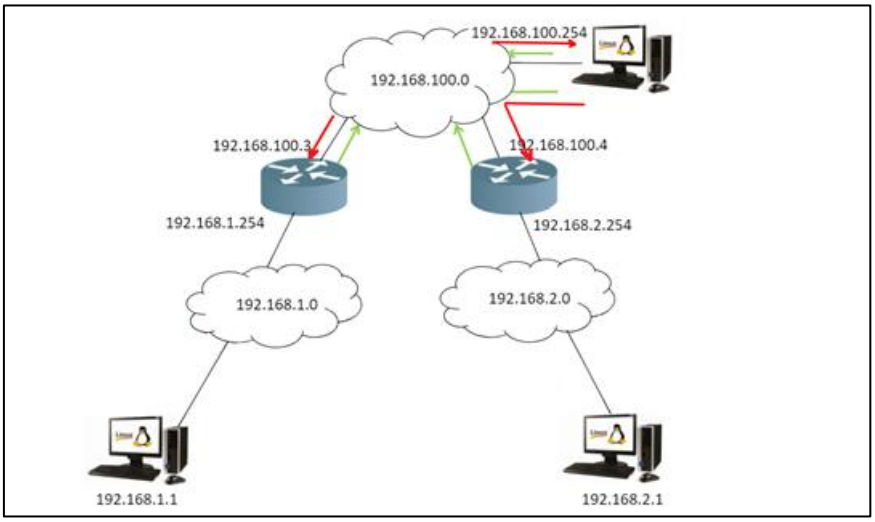


Ilustración 12: Representación del intercambio de tramas en LISP para la resolución de nombres

Ambos RLOC, tras el lanzamiento del archivo de LISP, proceden a comunicarse con su MSMR predefinido, para proporcionarle sus identificadores y direcciones que son capaces de alcanzar. De esta manera, el MS/MR tiene almacenado los RLOC de los LISP *router* a los que va a dar servicio y los EID conectados a cada *router*.

Cuando llega tráfico de PC1 a xTR1, este último no sabe realmente a quién se lo tiene que mandar. Por lo tanto, pregunta al MS/MR por el EID en cuestión (sentido en verde de la Ilustración 4), quién consulta las direcciones que tiene almacenadas, para proporcionarle el RLOC destino que corresponda (sentido en rojo en la Ilustración 4), en este caso xTR2. De esta forma, xTR1 es capaz de mandar tráfico con destino PC2. Tenemos que tener en cuenta, que para la respuesta a este tráfico, necesitamos el mismo proceso con xTR2 Y MS/MR, ya que xTR2 no sabrá como encaminar tráfico hacia PC1. Este proceso de consulta, se realiza únicamente cuando hay tráfico que enrutar, lo que provoca una pérdida de un paquete en cada sentido de la comunicación, por la expiración de su *timeout*. Cuando este proceso acaba, lo que tenemos es algo parecido a lo siguiente:

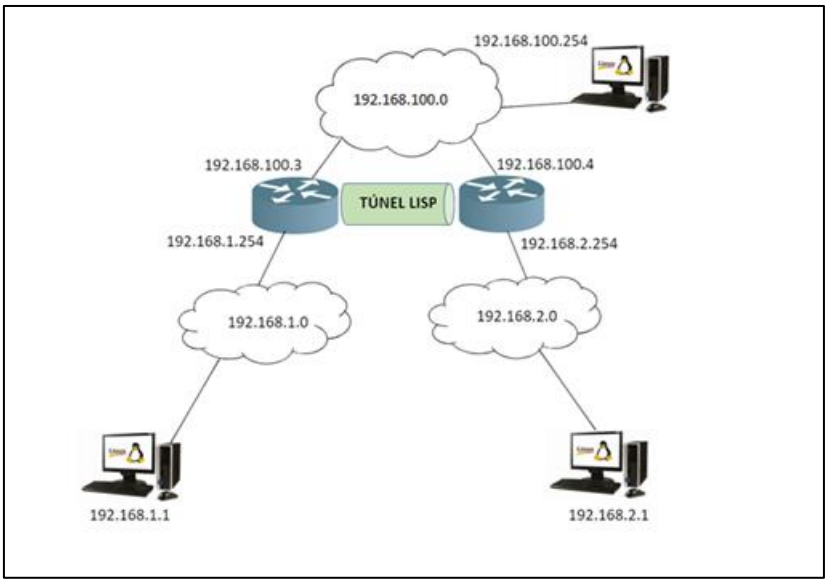


Ilustración 13: Representación del escenario final

De esta forma comprobamos, que para los EIDs, la comunicación mediante LISP, es completamente transparente ya que no necesitan ningún cambio.

5. ANÁLISIS DEL CÓDIGO

Con el objetivo de introducir seguridad en el escenario, se propone establecer un nuevo puerto, 4344, que será usado por un túnel IPsec para enviar los datos seleccionados de LISP-SM. Con la inclusión de la seguridad se busca ocultar la clase/tipo de información que se está transmitiendo y las direcciones privadas de los *host*, ya que no debería ser información accesible para ningún tercer actor en la comunicación. Por otro lado, debido al *overhead* que introduce IPsec, se presentó la idea de dejar a la elección del usuario el uso o no de esta posibilidad de securizar su comunicación dependiendo de sus intereses. En el archivo de configuración de la multiplexión se ha incluido un campo para elegir el modo de seguridad. Si no es necesaria la seguridad, se elegirá el modo 0, mientras que si interesa securizar todo el tráfico se establecerá como modo el 1.

Inicialmente se hizo un estudio detallado del código del repositorio de GitHub. Dicho proyecto cuenta con multitud de ficheros y librerías. Se buscaron las funciones claves en las cuales se realizaban la multiplexión, el envío de datos, el encapsulado, la lectura de datos del fichero de configuración, la comprobación de cambios en el fichero…para comprenderlas lo mejor posible. Al mismo tiempo que se realizaba el estudio de las funciones, se hizo el estudio de variables, punteros y estructuras de datos que almacenan las opciones de multiplexión.

5.1 INCLUSIÓN DE IPSEC

En primer lugar, se hicieron las modificaciones correspondientes en el código para incluir los modos de seguridad. Se incluyó la lectura de un campo nuevo del fichero de configuración de LISP-SM. Para el modo 0 el funcionamiento es idéntico al original, ya que la información será enviada por el puerto de multiplexión 4343. Sin embargo para la creación del modo seguro, el modo 1, se añadió en el código el puerto 4344 y se adaptó la función encargada de mandar los datos multiplexados a encapsular (cabeceras IP y UDP), cambiando los puertos origen y destino. Además, esta función previa al encapsulado de los datos fue cambiada para diferenciar el modo de seguridad y elegir el encapsulado correspondiente.

Para establecer una comunicación segura entre ambos xTR, se ha hecho uso de *ipsec-tools* para las direcciones IP de los extremos del túnel LISP y el puerto 4344, tal como se ha mencionado anteriormente. En su fichero de configuración, se encuentran las SPD y SA que crean una comunicación en modo transporte entre los XTR. Por otra parte, el intercambio de claves se ha realizado de forma manual, sin la utilización de ningún demonio de intercambio de claves automático, como *racoon*. Estas dos elecciones en la configuración se deben a la presuposición de confianza en el escenario. Es decir, se parte de que PC1 confía en XTR2 y que a su vez, PC2 confía en XTR1.

Sin embargo, esto no es suficiente puesto que: “Si en un sistema final la interfaz entre las aplicaciones y los protocolos TCP/IP está basada en ‘sockets’, la SPD se consulta sólo en el momento en que es creado un nuevo ‘socket’ y el procesado IPsec correspondiente se aplica a todo el tráfico que fluye en adelante a través del mismo”. (España, 2003, p. 289). En consecuencia, se deduce en el envío de datos mediante LISP-SM, se produce un conflicto con IPsec. Para realizar dicho envío, el código lo hace a través de *raw sockets*. Este es un tipo de *socket* que da acceso total a la capa de red, con el objetivo de, en pruebas posteriores, cambiar valores de la cabecera IP como son el TTL (*time to life*) y/o el ToS (*Type of Service*),ya que con sockets UDP/TCP no son posibles de cambiar.

Por tanto ha sido necesaria una adaptación en el código para que, dependiendo de la necesidad de seguridad en el tráfico de datos, se creara o no un nuevo socket UDP asociado al puerto 4344, por el cual mandar los datos. De esta forma, IPsec sí identifica correctamente el flujo en dicho puerto y el envío de datos se efectúa de la manera deseada. Por el contrario, se apreció que la demultiplexión en el xTR receptor era errónea. Tras analizar capturas de tráfico, se comprobó que la longitud del paquete multiplexado era mayor a la esperada en recepción. La cabecera UDP necesaria en el uso de *raw sockets*, era la causante del fallo. El propio *socket* UDP introduce dicha cabecera por lo que se introdujeron modificaciones en la función ocupada del encapsulado para evitar mandar paquetes con ambas cabeceras.

Al modo 0 y modo 1, se les ha complementado con el modo 2, que introduce seguridad dependiendo si el tráfico que llega al LISP *router* ya viene seguro o no. Con este funcionamiento, no se añade redundancia en la comunicación. Por otro lado, podría ocurrir que no todo el tráfico fuera homogéneo y tuviéramos diferentes flujos, unos seguros y otros no.

Para perfeccionar este modo, también en el fichero de configuración de LISP-SM, se ha añadido un campo en el que el usuario debe introducir un valor que considera como el mínimo ratio de paquetes seguros con respecto del total de la multiplexión en el paquete final necesario para no introducir seguridad redundante. Por ejemplo, teniendo una multiplexión configurada a 4 paquetes y un ratio de 0.5, significa que, el LISP *router* mandará los datos por el puerto 4344 (seguridad) si el paquete está formado por 3 o 4 no seguros. En cualquier otro caso, no se hará uso de IPsec. En este modo, hay dos situaciones que merecen una explicación particular:

- Ratio mínimo = 1: implica que el usuario requiere que todos los paquetes que formen la multiplexión lleguen seguros para no usar IPsec. Con otras palabras, con que llegue solo un paquete no seguro para formar el paquete final, se aplicarán las políticas de seguridad configuradas.

- Ratio mínimo = 0: en este caso, la seguridad no es requisito importante para el gestor del escenario ya que solamente se usará IPsec en el caso que todos los paquetes que lleguen estén sin cifrar. Es decir, si el paquete de multiplexión está formado por uno cifrado, se mandará el total sin cifrar.

Para la realización de estos modos, en total se han modificado 10 ficheros y el fichero de ejemplo de configuración, explicando cada campo añadido.

- *Lispd_config_confuse.c:* fichero que se encarga de leer los campos del archivo de configuración. Se añade el reconocimiento de los dos parámetros nuevos.
- *Simplemux.h*: inclusión de las variables necesarias en la estructura de datos que almacena los campos del fichero de configuración para Simplemux. Nueva variable en la estructura de datos usada en la multiplexión usada para marcar el uso o no de la seguridad. Se añade un campo a la función de multiplexión.
- *Simplemux.c*: en la función que recoge los datos bytes a enviar por el túnel, *mux_tun_output,* se añade la lectura del protocolo de los paquetes para que en la llamada a la función encargada de la multiplexión, *mux_packets,* esta cuente cada tipo de paquete hasta el momento que tiene listo el paquete final. Cambios en la función que inicia, en la que resetea y en la que actualiza dinámicamente los parámetros para incluir los nuevos campos. Se ha añadido la función que crea el socket UDP, *socket_for_ipsec*. Se ha modificado la función que envía los datos a encapsular, *mux_output_unicast,* para que diferencie según el modo de seguridad, qué puerto es el necesario, así como el uso o no del socket UDP para el envío. Cambio en la función de expiración del *timeout*, *muxed_timer_process_all,* para que la llamada a la multiplexión sea correcta.
- *Liblisp.h*: inclusión de un parámetro nuevo en la función de encapsulado.
- *Liblisp*.c: cambios en la función de encapsulado, *lisd_data_encap,* para que introduzca una cabecera UDP o no.
- *Liblisp.h:* inclusión del nuevo puerto 4344, *IPSEC_MUX_DATA_PORT* y modificación de la función *lisp_data_encap.*
- *Tun_input.c* y *tun_output.c:* modificación en llamadas a funciones como *lisp_data_encap* y reconocimiento del nuevo puerto añadido.
- *Sockets.c:* al igual que se hace con los otros puertos, 4341,4342 y 4343, creación de un *dummy socket* para que no se envíen paquetes *ICMP unreachable* sobre el puerto nuevo.
- *Sockets-utils.c*: modificación de un parámetro en la creación del socket UDP, *MSG-DONTROUTE,* para que no se envíen los paquetes al *router* por defecto.

5.2 PRUEBAS A REALIZAR

Para comprobar el funcionamiento correcto de las nuevas funcionalidades, se han realizado diferentes pruebas en el escenario. Se ha usado PC1 como emisor, por consiguiente, PC2 será el receptor. Las pruebas consisten en generar un tráfico determinado y, con diferentes configuraciones de multiplexión y de seguridad en xTR1 y xTR2 (iguales en ambos), estudiar los datos capturados. Se han realizado capturas entre PC1 y xTR1, para ver la generación del tráfico original, y entre xTR1 Y xTR2, para comprobar el funcionamiento de la multiplexión y de la seguridad.

Para los modos 0 y 1 se ha generado el mismo tráfico desde PC1. Se ha optado por el protocolo UDP, puesto que es el usado habitualmente como protocolo en la capa de transporte para servicios como la transmisión de vídeo y voz, por ejemplo, y se ha elegido un tamaño de paquete pequeño, 100 bytes. Se ha configurado para que el tiempo total del envío sean 10 segundos, suficientes para evaluar el funcionamiento correcto o no del escenario. El comando usado es el siguiente:

./ITGSend -a 192.168.2.1 -c 100 -T UDP -t 10000 –l mod_mux_sender.log

- -a: dirección destino (PC2)
- -c: tamaño de los paquetes
- -T: protocolo de los paquetes
- -t: milisegundos de comunicación
- –l : generar un archivo de log en el remitente

En cuanto a la multiplexión y a modo de ejemplo, se crearán túneles de multiplexión de dos y cuatro paquetes. Para completar las pruebas de cada modo, se ha introducido seguridad con *ipsec-tools* a este tráfico generado desde PC1, para comprobar su correcto funcionamiento ante la llegada de tráfico cifrado. Para finalizar con estos dos modos, se compararán los anchos de banda utilizados por cada uno, para comprobar el *overhead* introducido por la seguridad. Además, se probará el cambio dinámico entre estos modos y como varía el ancho de banda para cada uno.

Para verificar el modo 2 de funcionamiento, se lanzarán dos tráficos diferentes desde PC1, uno de ellos seguro, con el objetivo de probar la capacidad del router para identificar cada flujo e introducir seguridad extra o no dependiendo del ratio configurado.

Para descartar tráfico no útil para este Trabajo, se realiza la captura configurando filtros con Tcpdump. En la interfaz privada de xTR1, se filtra con la dirección destino de PC2, es decir 192.168.2.1. Mientras que, en la interfaz del túnel de xTR2, se filtra con las direcciones de los *host* de ambos LISP *router*, 192.168.100.3 y 192.168.100.4.

6. RESULTADOS

A continuación, se realiza un estudio sobre los resultados obtenidos en las pruebas efectuadas para cada modo de seguridad y número de paquetes multiplexados. En el Anexo B, se incluyen los ficheros (LISP-SM e *ipsec-tools.conf)* de configuración correspondientes a cada test.

6.1 PRUEBAS MODO=0

Para el modo 0, multiplexión de 2 paquetes y el tráfico previo generado, obtenemos los siguientes *logs* de D-ITG:

```
root@pc1:/usr/bin# ./ITGDec mode0_mux2_sender.log
ITGDec version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
------------------------------------------------------------
Flow number: 1
From 192.168.1.1:38509
To     192.168.2.1:8999
------------------------------------------------------------
Total time                =      9.999522 s
Total packets             =          9149
Minimum delay             =      0.000000 s
Maximum delay             =      0.000000 s
Average delay             =      0.000000 s
Average jitter            =      0.000000 s
Delay standard deviation  =      0.000000 s
Bytes received            =        914900
Average bitrate           =    731.954987 Kbit/s
Average packet rate       =    914.943734 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =      0.000000 pkt
------------------------------------------------------------


****************  TOTAL RESULTS    ****************

------------------------------------------------------------
Number of flows           =             1
Total time                =      9.999522 s
Total packets             =          9149
Minimum delay             =      0.000000 s
Maximum delay             =      0.000000 s
Average delay             =      0.000000 s
Average jitter            =      0.000000 s
Delay standard deviation  =      0.000000 s
Bytes received            =        914900
Average bitrate           =    731.954987 Kbit/s
Average packet rate       =    914.943734 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =             0 pkt
Error lines               =             0
------------------------------------------------------------
```

```
root@pc2:/usr/bin#  ./ITGDec mode0_mx2_recv.log
ITGDec version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
------------------------------------------------------------
Flow number: 1
From 192.168.1.1:38509
To     192.168.2.1:8999
------------------------------------------------------------
Total time                =      9.997945 s
Total packets             =          9149
Minimum delay             =     -0.035740 s
Maximum delay             =     -0.030497 s
Average delay             =     -0.034920 s
Average jitter            =      0.000904 s
Delay standard deviation  =      0.000500 s
Bytes received            =        914900
Average bitrate           =    732.070440 Kbit/s
Average packet rate       =    915.088051 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =      0.000000 pkt
------------------------------------------------------------


****************  TOTAL RESULTS    ****************

------------------------------------------------------------
Number of flows           =             1
Total time                =      9.997945 s
Total packets             =          9149
Minimum delay             =     -0.035740 s
Maximum delay             =     -0.030497 s
Average delay             =     -0.034920 s
Average jitter            =      0.000904 s
Delay standard deviation  =      0.000500 s
Bytes received            =        914900
Average bitrate           =    732.070440 Kbit/s
Average packet rate       =    915.088051 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =             0 pkt
Error lines               =             0
------------------------------------------------------------
```

Ilustración 144: Captura de estadísticas de ITGDec en PC1 y PC2 para Modo=0 y Mux=2

Las imágenes son capturas de los resultados que ofrece ITGDec, ya que no son leíbles de ninguna otra forma. En ellas podemos observar que no ha habido ninguna pérdida de paquetes, por lo tanto la comunicación se ha realizado correctamente. También vemos que los puertos que han usado ITGSend e ITGRec son: el 38509 en el envío y el 8999 en recepción (éste será el mismo en las siguientes pruebas, ya que es el que tiene por defecto y no se ha cambiado). Para ver realmente, como se han mandado los datos UDP generados, se deben estudiar las capturas realizadas antes y en del túnel LISP-SM. Se adjuntan a continuación, con estadísticas de uso de puertos:

1	0.000000	192.168.1.1	192.168.2.1	TCP	74 56897 → 9000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460
2	0.998810	192.168.1.1	192.168.2.1	TCP	74 [TCP Retransmission] 56897 → 9000 [SYN] Seq=0 Win=
3	2.449069	192.168.2.1	192.168.1.1	TCP	74 9000 → 56897 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=
4	2.449181	192.168.1.1	192.168.2.1	TCP	66 56897 → 9000 [ACK] Seq=1 Ack=1 Win=29248 Len=0 TSv
5	2.449246	192.168.1.1	192.168.2.1	TCP	67 56897 → 9000 [PSH, ACK] Seq=1 Ack=1 Win=29248 Len=
6	2.454111	192.168.2.1	192.168.1.1	TCP	74 [TCP Out-Of-Order] 9000 → 56897 [SYN, ACK] Seq=0 A
7	2.454224	192.168.1.1	192.168.2.1	TCP	66 [TCP Dup ACK 4#1] 56897 → 9000 [ACK] Seq=2 Ack=1 W
8	2.454518	192.168.2.1	192.168.1.1	TCP	66 9000 → 56897 [ACK] Seq=1 Ack=2 Win=28992 Len=0 TSv
9	3.455363	192.168.2.1	192.168.1.1	TCP	67 9000 → 56897 [PSH, ACK] Seq=1 Ack=2 Win=28992 Len=
10	3.455456	192.168.1.1	192.168.2.1	TCP	66 56897 → 9000 [ACK] Seq=2 Ack=2 Win=29248 Len=0 TSv
11	3.455590	192.168.1.1	192.168.2.1	TCP	93 56897 → 9000 [PSH, ACK] Seq=2 Ack=2 Win=29248 Len=
12	4.457120	192.168.2.1	192.168.1.1	TCP	73 9000 → 56897 [PSH, ACK] Seq=2 Ack=29 Win=28992 Len
13	4.460060	192.168.1.1	192.168.2.1	UDP	142 38509 → 8999 Len=100
14	4.461115	192.168.1.1	192.168.2.1	UDP	142 38509 → 8999 Len=100
15	4.462209	192.168.1.1	192.168.2.1	UDP	142 38509 → 8999 Len=100
16	4.463278	192.168.1.1	192.168.2.1	UDP	142 38509 → 8999 Len=100
17	4.464330	192.168.1.1	192.168.2.1	UDP	142 38509 → 8999 Len=100
18	4.465441	192.168.1.1	192.168.2.1	UDP	142 38509 → 8999 Len=100
19	4.466492	192.168.1.1	192.168.2.1	UDP	142 38509 → 8999 Len=100
20	4.467585	192.168.1.1	192.168.2.1	UDP	142 38509 → 8999 Len=100
21	4.468652	192.168.1.1	192.168.2.1	UDP	142 38509 → 8999 Len=100
22	4.469762	192.168.1.1	192.168.2.1	UDP	142 38509 → 8999 Len=100
23	4.470789	192.168.1.1	192.168.2.1	UDP	142 38509 → 8999 Len=100

Ilustración 155: Captura del envío de datos desde PC1

1	0.000000	192.168.100.4	192.168.100.3	LISP	82 Map-Reply for 192.168.2.0/24
2	1.452255	192.168.100.3	192.168.100.4	UDP	112 4343 → 4343 Len=70
3	2.446153	192.168.100.4	192.168.100.3	UDP	112 4343 → 4343 Len=70
4	2.448899	192.168.100.3	192.168.100.4	UDP	158 4343 → 4343 Len=116
5	2.451325	192.168.100.4	192.168.100.3	UDP	165 4343 → 4343 Len=123
6	3.450132	192.168.100.3	192.168.100.4	UDP	104 4343 → 4343 Len=62
7	3.452716	192.168.100.4	192.168.100.3	UDP	105 4343 → 4343 Len=63
8	3.454796	192.168.100.3	192.168.100.4	UDP	184 4343 → 4343 Len=142
9	4.454199	192.168.100.4	192.168.100.3	UDP	111 4343 → 4343 Len=69
10	4.459827	192.168.100.3	192.168.100.4	UDP	311 4343 → 4343 Len=269
11	4.462039	192.168.100.3	192.168.100.4	UDP	311 4343 → 4343 Len=269
12	4.464205	192.168.100.3	192.168.100.4	UDP	311 4343 → 4343 Len=269
13	4.466771	192.168.100.3	192.168.100.4	UDP	311 4343 → 4343 Len=269
14	4.468930	192.168.100.3	192.168.100.4	UDP	311 4343 → 4343 Len=269
15	4.470199	192.168.100.3	192.168.100.4	UDP	311 4343 → 4343 Len=269
16	4.472346	192.168.100.3	192.168.100.4	UDP	311 4343 → 4343 Len=269
17	4.474696	192.168.100.3	192.168.100.4	UDP	311 4343 → 4343 Len=269
18	4.476750	192.168.100.3	192.168.100.4	UDP	311 4343 → 4343 Len=269
19	4.478858	192.168.100.3	192.168.100.4	UDP	311 4343 → 4343 Len=269
20	4.481038	192.168.100.3	192.168.100.4	UDP	311 4343 → 4343 Len=269

Ilustración 166: Captura del envío de datos en el túnel LISP-SM

Wireshark · Destinations and Ports · xtr2_mode0_mx2.log

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Destinations and Ports	4591				0.2486	100%	0.4800	14.066
192.168.100.4	4583				0.2482	99.83%	0.4800	14.066
UDP	4583				0.2482	100.00%	0.4800	14.066
4343	4583				0.2482	100.00%	0.4800	14.066
192.168.100.3	8				0.0004	0.17%	0.0200	2.446
UDP	8				0.0004	100.00%	0.0200	2.446
4343	7				0.0004	87.50%	0.0200	2.446
4342	1				0.0001	12.50%	0.0100	0.000

Ilustración 177: Estadísticas de Wireshark sobre la captura

En la primera imagen se aprecia cómo se realiza la señalización para el establecimiento de la comunicación y cómo comienzan a mandarse los paquetes UDP. Los puertos son los

mismos que se reflejan en las estadísticas de ITGDec. Cada paquete transporta 100 bytes de datos y un total de 142 (100+8 (Cabecera UDP)+ 20 (Cabecera IP)+14(Cabecera Ethernet)). Si se estudia la segunda captura, vemos que las direcciones origen y destino corresponden, como es debido, a las direcciones de xTR1 y xTR2. El puerto usado es el 4343, el puerto que implementa Simplemux.

Si observamos las estadísticas de puertos usados, comprobamos que se ha usado siempre el puerto 4343, sin diferenciar señalización de datos. Aparece una vez el puerto 4342 por el envío de un mensaje *Map-Reply*, de PC2 a PC1. Este mensaje es el primero en la Ilustración 18.

0000	00 0c 29 e8 91 32 00 0c 29 25 61 ba 08 00 45 00	..)è.2..)%aº..E.
0010	01 29 24 0c 40 00 80 **11** 8c 5f **c0 a8 64 03 c0 a8**	.)$.@...._À¨d.À¨
0020	**64 04** **10 f7 10 f7** 01 15 6e 05 00 00 00 00 00 00	d..÷.÷..n.....
0030	00 00 **c1 00 04** 45 00 00 80 f1 6b 40 00 3f 11 c5	..Á..E...ñk@.?.Å
0040	ae **c0 a8 01 01 c0 a8 02 01** **96 6d 23 27** 00 6c 0b	®À¨..À¨...m#'.l.
0050	a3 00 00 00 01 00 00 00 11 00 00 b9 09 00 0e 02	£.....¹....
0060	f0 6c 4f bb 7a 14 fa 4c 1e 08 ba 04 13 a3 c8 60	ðlO»z.úL..º..£È`
0070	8a 2d 43 ca c9 42 eb da 63 8b d3 7f 8b 4b fa 2b	.-CÊÉBëÚc.Ó..Kú+
0080	b7 4a 66 b2 dd 62 7e fc 6a 39 80 fc 5d 4a 5d 67	·Jf²Ýb~üj9.ü]J]g
0090	77 20 32 c1 62 9e 9c 45 a9 70 c4 35 bb 3f df 74	w 2Áb..E©pÄ5»?ßt
00a0	09 46 a6 e7 a8 25 64 92 5f e4 0f 3c 2f 6d a3 27	.F¦ç¨%d._ä.</m£'
00b0	8d 56 e8 70 74 **81 00** 45 00 00 80 f1 6c 40 00 3f	.Vèpt..E...ñl@.?
00c0	11 c5 ad **c0 a8 01 01 c0 a8 02 01** **96 6d 23 27** 00	.Å.À¨..À¨...m#'.
00d0	6c 07 65 00 00 00 01 00 00 00 12 00 00 b9 09 00	l.e.....¹..
00e0	0e 07 2d 6c 4f bb 7a 14 fa 4c 1e 08 ba 04 13 a3	..-lO»z.úL..º..£
00f0	c8 60 8a 2d 43 ca c9 42 eb da 63 8b d3 7f 8b 4b	È`.-CÊÉBëÚc.Ó..K
0100	fa 2b b7 4a 66 b2 dd 62 7e fc 6a 39 80 fc 5d 4a	ú+·Jf²Ýb~üj9.ü]J
0110	5d 67 77 20 32 c1 62 9e 9c 45 a9 70 c4 35 bb 3f	]gw 2Áb..E©pÄ5»?
0120	df 74 09 46 a6 e7 a8 25 64 92 5f e4 0f 3c 2f 6d	ßt.F¦ç¨%d._ä.</m
0130	a3 27 8d 56 e8 70 74	£'.Vèpt

Ilustración 18: Paquete en el túnel extraído de la captura con modo 0 y multiplexión = 2

En la anterior tabla se ha representado un paquete del túnel:

- En verde la cabecera Ethernet.
- En amarillo la cabecera IP. En ella se han resaltado los bytes del protocolo: 0x11 -> 17 (UDP), y los de direcciones origen y destino: 0xc0a86403 y 0xc0a86404 equivalen a 192.168.100.3 y 192.168.100.4 respectivamente.
- La cabecera UDP, en rojo. Con ella, y las dos anteriores obtenemos la cabecera del túnel. En la cabecera UDP, destaca el puerto origen y destino: 0x10f7 que equivale al puerto 4343.
- En negro bytes de LISP. Son *flags* a cero creados con el propósito ser usados en entornos con VPN y etiquetas. El resto del paquete son datos, pero en ellos se pueden diferenciar los dos paquetes multiplexados.

- En azul oscuro se aprecia la cabecera IP del paquete, con direcciones origen y destino 192.168.1.1 y 192.168.2.1 (representadas de forma hexadecimal como 0xc0a80101/02. A continuación, su cabecera UDP, con los puertos origen y destino de D-ITG. En morado, los datos mandados.
- Se puede observar, como al inicio del primer paquete y a su final aparecen bytes extra marcado en rosa. 0xc100c4 corresponde a la primera cabecera de Simplemux, la cual es diferente, mientras que 0x8100, son los bytes de la cabecera de separación que introduce Simplemux.

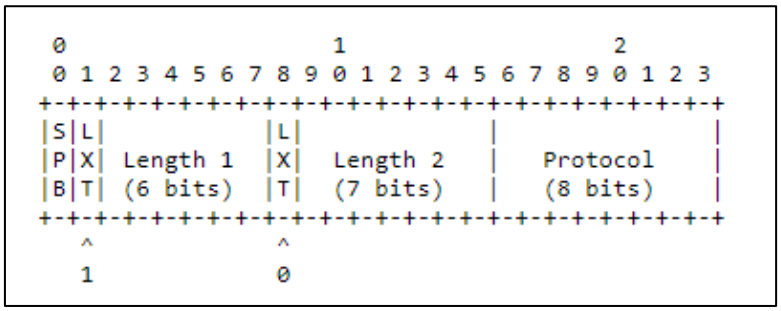


Ilustración 19: Representación de la primera cabecera de Simplemux para paquetes mayores de 65 bytes y menores de 8192, rescatada de https://tools.ietf.org/html/draft-saldana-tsvwg-simplemux-09#page-5

En la primera cabecera, el primer bit es del campo Single Protocol Bit (SPB). Si todos los paquetes multiplexados pertenecen al mismo protocolo, se establece a 1. Si el primer bit LTX vale 1, significa que hay un byte extra para representar la longitud del paquete.

0xc1 00 04 -> **1**100 0001 **0**000 0000 **0000 0100**. Si concatenamos los bits de la primera y de la segunda longitud, tenemos la longitud del paquete multiplexado => 2^7 = 128 bytes. En el campo Protocolo aparece 4, ya que lo que viene a continuación en el paquete es un paquete IP encapsulado.

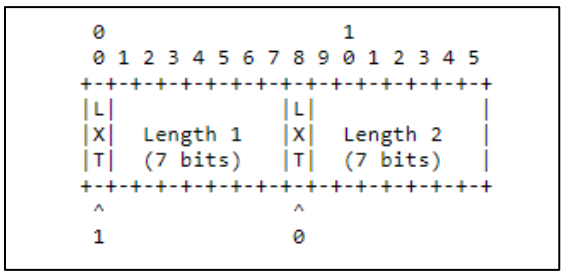


Ilustración 20: Cabecera Simplemux para paquetes mayores de 128 bytes y menores de 2^{14} bytes, rescatada de su Internet-Draft, https://tools.ietf.org/html/draft-saldana-tsvwg-simplemux-10

0x81 00 -> **1**000 0001 **0**000 0000. De la misma forma que en la primera cabecera, la longitud del paquete es la concatenación de ambos campos de longitud. Obtenemos 2^7 =128 bytes (100 bytes de datos + 20 bytes cabecera IP + 8 cabecera UDP).

A continuación, se cambia el valor de paquetes multiplexados a 4.

```
root@pc1:/usr/bin# ./ITGDec mode0_mux4_sender.log
ITGDec version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
------------------------------------------------------------
Flow number: 1
From 192.168.1.1:42218
To     192.168.2.1:8999
------------------------------------------------------------
Total time                =      9.999449 s
Total packets             =          9166
Minimum delay             =      0.000000 s
Maximum delay             =      0.000000 s
Average delay             =      0.000000 s
Average jitter            =      0.000000 s
Delay standard deviation  =      0.000000 s
Bytes received            =        916600
Average bitrate           =    733.320406 Kbit/s
Average packet rate       =    916.650507 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =      0.000000 pkt
------------------------------------------------------------


****************  TOTAL RESULTS   ****************

Number of flows           =              1
Total time                =      9.999449 s
Total packets             =          9166
Minimum delay             =      0.000000 s
Maximum delay             =      0.000000 s
Average delay             =      0.000000 s
Average jitter            =      0.000000 s
Delay standard deviation  =      0.000000 s
Bytes received            =        916600
Average bitrate           =    733.320406 Kbit/s
Average packet rate       =    916.650507 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =             0 pkt
Error lines               =             0
------------------------------------------------------------
```

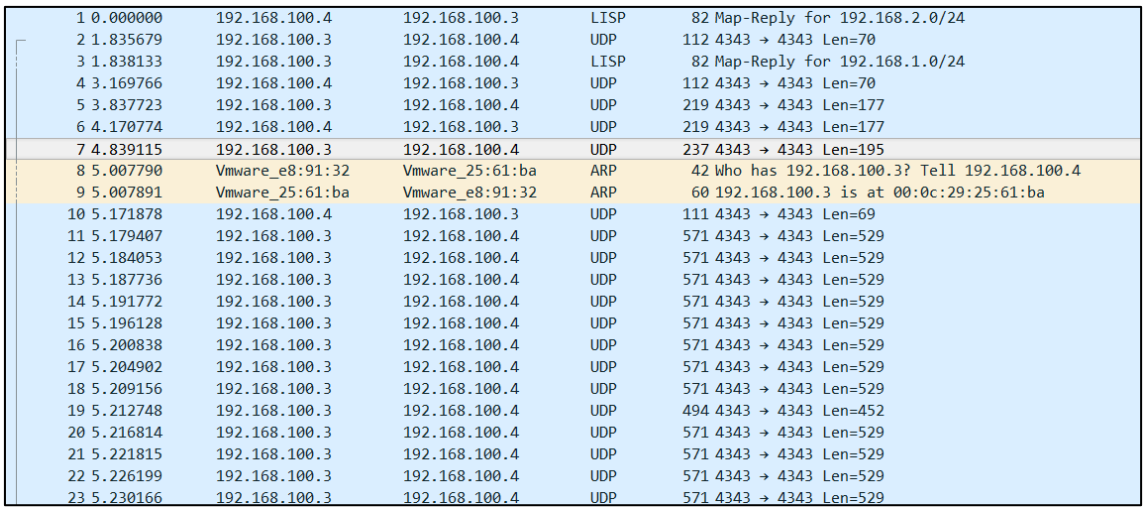
```
root@pc2:/usr/bin#  ./ITGDec mode0_mx4_recv.log
ITGDec version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
------------------------------------------------------------
Flow number: 1
From 192.168.1.1:42218
To     192.168.2.1:8999
------------------------------------------------------------
Total time                =      9.997055 s
Total packets             =          9166
Minimum delay             =     -0.035641 s
Maximum delay             =     -0.027052 s
Average delay             =     -0.033646 s
Average jitter            =      0.001356 s
Delay standard deviation  =      0.001069 s
Bytes received            =        916600
Average bitrate           =    733.496015 Kbit/s
Average packet rate       =    916.870018 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =      0.000000 pkt
------------------------------------------------------------


****************  TOTAL RESULTS   ****************

Number of flows           =              1
Total time                =      9.997055 s
Total packets             =          9166
Minimum delay             =     -0.035641 s
Maximum delay             =     -0.027052 s
Average delay             =     -0.033646 s
Average jitter            =      0.001356 s
Delay standard deviation  =      0.001069 s
Bytes received            =        916600
Average bitrate           =    733.496015 Kbit/s
Average packet rate       =    916.870018 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =             0 pkt
Error lines               =             0
------------------------------------------------------------
```

Ilustración 181: Captura de estadísticas de ITGDec en PC1 y PC2 para Modo=0 y Mux = 4

En esta prueba se observa cómo el puerto de origen ha cambiado con respecto a la primera prueba y que la comunicación se ha realizado de forma correcta, porque no se ha perdido ningún paquete. El tráfico generado es el mismo que en el anterior caso, por lo que la captura es prácticamente idéntica a la de la anterior prueba.

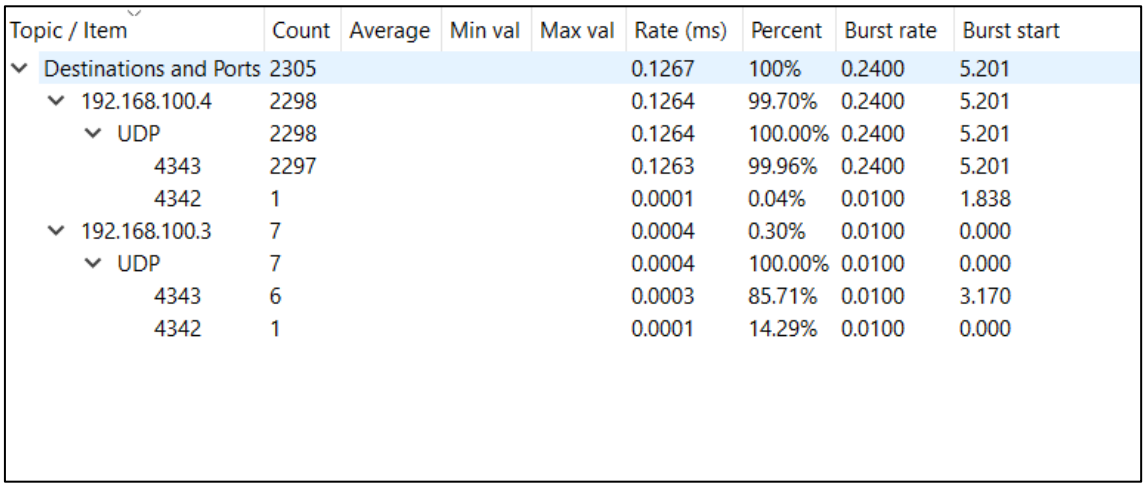
1	0.000000	192.168.1.1	192.168.2.1	TCP	74 56900 → 9000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460
2	0.999007	192.168.1.1	192.168.2.1	TCP	74 [TCP Retransmission] 56900 → 9000 [SYN] Seq=0 Win
3	3.002960	192.168.1.1	192.168.2.1	TCP	74 [TCP Retransmission] 56900 → 9000 [SYN] Seq=0 Win
4	3.172725	192.168.2.1	192.168.1.1	TCP	74 9000 → 56900 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len
5	3.172853	192.168.1.1	192.168.2.1	TCP	66 56900 → 9000 [ACK] Seq=1 Ack=1 Win=29248 Len=0 TS
6	3.172909	192.168.1.1	192.168.2.1	TCP	67 56900 → 9000 [PSH, ACK] Seq=1 Ack=1 Win=29248 Len
7	4.173773	192.168.2.1	192.168.1.1	TCP	74 [TCP Retransmission] 9000 → 56900 [SYN, ACK] Seq=
8	4.173874	192.168.1.1	192.168.2.1	TCP	66 [TCP Dup ACK 5#1] 56900 → 9000 [ACK] Seq=2 Ack=1
9	4.174232	192.168.2.1	192.168.1.1	TCP	66 9000 → 56900 [ACK] Seq=1 Ack=2 Win=28992 Len=0 TS
10	4.174416	192.168.2.1	192.168.1.1	TCP	67 9000 → 56900 [PSH, ACK] Seq=1 Ack=2 Win=28992 Len
11	4.174474	192.168.1.1	192.168.2.1	TCP	66 56900 → 9000 [ACK] Seq=2 Ack=2 Win=29248 Len=0 TS
12	4.174603	192.168.1.1	192.168.2.1	TCP	93 56900 → 9000 [PSH, ACK] Seq=2 Ack=2 Win=29248 Len
13	5.174828	192.168.2.1	192.168.1.1	TCP	73 9000 → 56900 [PSH, ACK] Seq=2 Ack=29 Win=28992 Le
14	5.177644	192.168.1.1	192.168.2.1	UDP	142 42218 → 8999 Len=100
15	5.178704	192.168.1.1	192.168.2.1	UDP	142 42218 → 8999 Len=100
16	5.179802	192.168.1.1	192.168.2.1	UDP	142 42218 → 8999 Len=100
17	5.180833	192.168.1.1	192.168.2.1	UDP	142 42218 → 8999 Len=100
18	5.181905	192.168.1.1	192.168.2.1	UDP	142 42218 → 8999 Len=100
19	5.183043	192.168.1.1	192.168.2.1	UDP	142 42218 → 8999 Len=100
20	5.184095	192.168.1.1	192.168.2.1	UDP	142 42218 → 8999 Len=100
21	5.185183	192.168.1.1	192.168.2.1	UDP	142 42218 → 8999 Len=100
22	5.186257	192.168.1.1	192.168.2.1	UDP	142 42218 → 8999 Len=100
23	5.187404	192.168.1.1	192.168.2.1	UDP	142 42218 → 8999 Len=100

Ilustración 22: Captura de datos enviados por PC1



1	0.000000	192.168.100.4	192.168.100.3	LISP	82	Map-Reply for 192.168.2.0/24
2	1.835679	192.168.100.3	192.168.100.4	UDP	112	4343 → 4343 Len=70
3	1.838133	192.168.100.3	192.168.100.4	LISP	82	Map-Reply for 192.168.1.0/24
4	3.169766	192.168.100.4	192.168.100.3	UDP	112	4343 → 4343 Len=70
5	3.837723	192.168.100.3	192.168.100.4	UDP	219	4343 → 4343 Len=177
6	4.170774	192.168.100.4	192.168.100.3	UDP	219	4343 → 4343 Len=177
7	4.839115	192.168.100.3	192.168.100.4	UDP	237	4343 → 4343 Len=195
8	5.007790	Vmware_e8:91:32	Vmware_25:61:ba	ARP	42	Who has 192.168.100.3? Tell 192.168.100.4
9	5.007891	Vmware_25:61:ba	Vmware_e8:91:32	ARP	60	192.168.100.3 is at 00:0c:29:25:61:ba
10	5.171878	192.168.100.4	192.168.100.3	UDP	111	4343 → 4343 Len=69
11	5.179407	192.168.100.3	192.168.100.4	UDP	571	4343 → 4343 Len=529
12	5.184053	192.168.100.3	192.168.100.4	UDP	571	4343 → 4343 Len=529
13	5.187736	192.168.100.3	192.168.100.4	UDP	571	4343 → 4343 Len=529
14	5.191772	192.168.100.3	192.168.100.4	UDP	571	4343 → 4343 Len=529
15	5.196128	192.168.100.3	192.168.100.4	UDP	571	4343 → 4343 Len=529
16	5.200838	192.168.100.3	192.168.100.4	UDP	571	4343 → 4343 Len=529
17	5.204902	192.168.100.3	192.168.100.4	UDP	571	4343 → 4343 Len=529
18	5.209156	192.168.100.3	192.168.100.4	UDP	571	4343 → 4343 Len=529
19	5.212748	192.168.100.3	192.168.100.4	UDP	494	4343 → 4343 Len=452
20	5.216814	192.168.100.3	192.168.100.4	UDP	571	4343 → 4343 Len=529
21	5.221815	192.168.100.3	192.168.100.4	UDP	571	4343 → 4343 Len=529
22	5.226199	192.168.100.3	192.168.100.4	UDP	571	4343 → 4343 Len=529
23	5.230166	192.168.100.3	192.168.100.4	UDP	571	4343 → 4343 Len=529

Ilustración 23: Captura del envío de datos en el túnel LISP-SM



Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Destinations and Ports	2305				0.1267	100%	0.2400	5.201
192.168.100.4	2298				0.1264	99.70%	0.2400	5.201
UDP	2298				0.1264	100.00%	0.2400	5.201
4343	2297				0.1263	99.96%	0.2400	5.201
4342	1				0.0001	0.04%	0.0100	1.838
192.168.100.3	7				0.0004	0.30%	0.0100	0.000
UDP	7				0.0004	100.00%	0.0100	0.000
4343	6				0.0003	85.71%	0.0100	3.170
4342	1				0.0001	14.29%	0.0100	0.000

Ilustración 24: Estadísticas de Wireshark sobre la captura

Tanto en las imágenes 23 como 24 se ve el uso del puerto de multiplexión. La longitud del paquete de multiplexión es de 529 bytes.

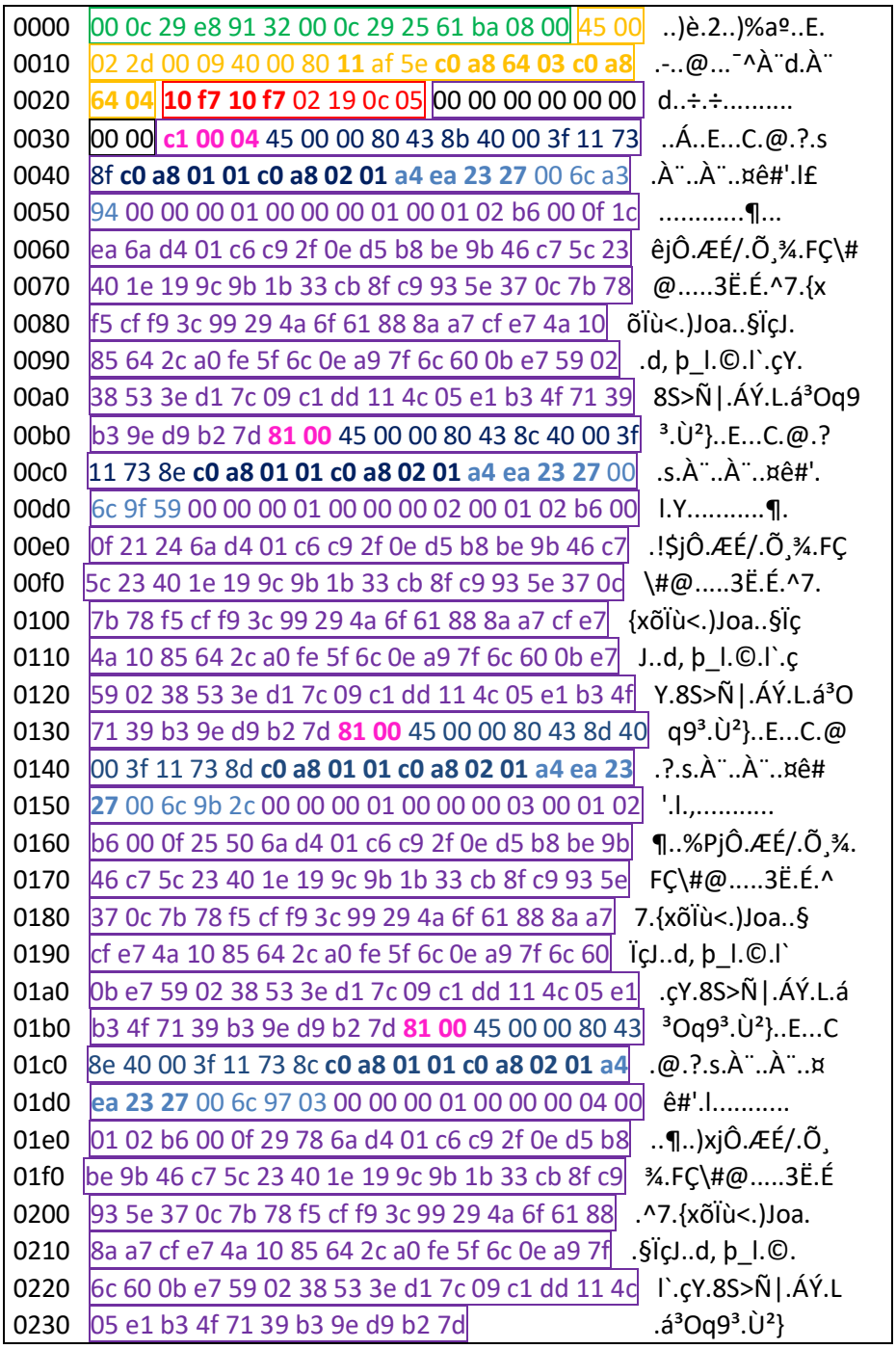


Ilustración 25: Captura de paquete en modo=0 con Mux=4

El análisis del paquete es idéntico al realizado en la anterior prueba, por lo que se obvia para evitar la repetición.

Para acabar con las pruebas del modo 0, se securizó tráfico desde PC1 hacia PC2, usando el paquete *ipsec-tools*, en modo transporte y ESP. La multiplexión se ha mantenido en 4 paquetes.

15 2.332293	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
16 2.333328	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
17 2.334386	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
18 2.335481	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
19 2.336542	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
20 2.337685	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
21 2.338763	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
22 2.339791	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
23 2.340848	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
24 2.342039	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
25 2.343081	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
26 2.344142	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
27 2.345245	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
28 2.346327	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
29 2.347442	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
30 2.348517	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
31 2.349619	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
32 2.350704	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
33 2.351774	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
34 2.352856	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
35 2.353965	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)
36 2.355046	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)

Ilustración 196: Captura del tráfico seguro generado desde PC1

14 4.694342	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
15 4.698585	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
16 4.703060	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
17 4.706220	192.168.100.3	192.168.100.4	UDP	554 4343 → 4343 Len=512
18 4.710480	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
19 4.714946	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
20 4.719228	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
21 4.723799	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
22 4.728206	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
23 4.732495	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
24 4.736901	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
25 4.741264	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
26 4.745592	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
27 4.749926	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
28 4.754319	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
29 4.758620	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
30 4.762983	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
31 4.767386	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
32 4.771830	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
33 4.776215	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
34 4.780703	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
35 4.784986	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
36 4.789325	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609

Ilustración 27: Captura del tráfico en el túnel

Como se puede observar en las capturas previas, en el túnel aparece tráfico UDP, corroborando que el modo 0 no introduce seguridad bajo ningún concepto.

6.2 PRUEBAS MODO=1

Para comprobar el funcionamiento de este modo, se cambia el modo de seguridad en el archivo de ambos *router*. Se configuran y se ejecutan las políticas de seguridad en cada *router*. Igual que para el modo 0, se comienza con una multiplexión de 2 paquetes. Se mantiene la forma de obtener las capturas y de generar tráfico.

```
root@pc1:/usr/bin# ./ITGDec model_mux2_sender.log
ITGDec version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
------------------------------------------------------------
Flow number: 1
From 192.168.1.1:58373
To     192.168.2.1:8999
------------------------------------------------------------
Total time                =      9.999652 s
Total packets             =          9146
Minimum delay             =      0.000000 s
Maximum delay             =      0.000000 s
Average delay             =      0.000000 s
Average jitter            =      0.000000 s
Delay standard deviation  =      0.000000 s
Bytes received            =        914600
Average bitrate           =    731.705463 Kbit/s
Average packet rate       =    914.631829 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =      0.000000 pkt
------------------------------------------------------------


****************  TOTAL RESULTS   ****************

Number of flows           =              1
Total time                =      9.999652 s
Total packets             =          9146
Minimum delay             =      0.000000 s
Maximum delay             =      0.000000 s
Average delay             =      0.000000 s
Average jitter            =      0.000000 s
Delay standard deviation  =      0.000000 s
Bytes received            =        914600
Average bitrate           =    731.705463 Kbit/s
Average packet rate       =    914.631829 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =             0 pkt
Error lines               =             0
------------------------------------------------------------
```

```
root@pc2:/usr/bin#  ./ITGDec model_mx2_recv.log
ITGDec version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
------------------------------------------------------------
Flow number: 1
From 192.168.1.1:58373
To     192.168.2.1:8999
------------------------------------------------------------
Total time                =      9.998149 s
Total packets             =          9146
Minimum delay             =     -0.035843 s
Maximum delay             =     -0.009345 s
Average delay             =     -0.034654 s
Average jitter            =      0.000914 s
Delay standard deviation  =      0.001155 s
Bytes received            =        914600
Average bitrate           =    731.815459 Kbit/s
Average packet rate       =    914.769324 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =      0.000000 pkt
------------------------------------------------------------


****************  TOTAL RESULTS   ****************

Number of flows           =              1
Total time                =      9.998149 s
Total packets             =          9146
Minimum delay             =     -0.035843 s
Maximum delay             =     -0.009345 s
Average delay             =     -0.034654 s
Average jitter            =      0.000914 s
Delay standard deviation  =      0.001155 s
Bytes received            =        914600
Average bitrate           =    731.815459 Kbit/s
Average packet rate       =    914.769324 pkt/s
Packets dropped           =             0 (0.00 %)
Average loss-burst size   =             0 pkt
Error lines               =             0
------------------------------------------------------------
```

Ilustración 2820: Captura de las estadísticas realizadas por ITGDec en PC1 y PC2

Igualmente que con el modo 0, la comunicación entre ambos *host* ha sido correcta, como marca ese 0% de pérdidas en paquetes.

```
    14 6.661311    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    15 6.662386    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    16 6.663444    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    17 6.664542    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    18 6.665593    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    19 6.666707    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    20 6.667783    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    21 6.668854    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    22 6.669952    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    23 6.671006    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    24 6.672163    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    25 6.673421    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    26 6.674490    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    27 6.675625    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    28 6.676678    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    29 6.677784    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    30 6.678867    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    31 6.679989    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    32 6.681055    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    33 6.682144    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    34 6.683225    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    35 6.684320    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
    36 6.685430    192.168.1.1    192.168.2.1    UDP    142 58373 → 8999 Len=100
```

Ilustración 2921: Tráfico generado en PC1

37	6.710257	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
38	6.712515	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
39	6.714869	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
40	6.716989	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
41	6.719025	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
42	6.721190	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
43	6.723358	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
44	6.725500	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
45	6.727871	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
46	6.729964	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
47	6.732287	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
48	6.734280	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
49	6.736629	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
50	6.738736	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
51	6.740986	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
52	6.743259	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
53	6.745580	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
54	6.747440	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
55	6.749724	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
56	6.752096	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
57	6.754018	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
58	6.756147	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)
59	6.758437	192.168.100.3	192.168.100.4	ESP	330 ESP (SPI=0x00000202)

Ilustración 30: Tráfico securizado por IPsec en el túnel

En la captura de la Ilustración 30, se puede observar la seguridad introducida en la comunicación entre xTR1 y xTR2. Con el modo 1 activo, todo el tráfico que xTR1 tenga que redireccionar a xTR2 saldrá por el puerto 4344. El protocolo de datos, puerto origen y destino no aparecen en la captura ya que se encuentran bajo la encriptación de ESP y, por lo tanto, no es información descifrable por ningún *sniffer*. Esto se debe al uso del modo de transporte en el archivo de configuración de IPsec, que recordemos, cifra todo el *payload* del paquete. Como añadido, tampoco son descifrables las IP privadas de origen y destino que aparecen en las cabeceras multiplexadas. En la siguiente imagen se demuestra:

```
0000  00 0c 29 e8 91 32 00 0c 29 25 61 ba 08 00 45 00   ..)è.2..)%aº..E.
0010  01 3c 70 ef 40 00 40 32 7f 48 c0 a8 64 03 c0 a8   .<pï@.@2.HÀ¨d.À¨
0020  64 04 00 00 02 02 00 00 86 3c 33 d5 e6 7b 98 1b   d........<3Õæ{..
0030  d2 f4 9e 6c 6a a7 23 a0 c6 70 f5 ed 6f ab 3d bd   Òô.lj§# Æpõío«=½
0040  51 ed 47 0b e9 e2 57 3c 35 4b bb eb 35 73 f0 f6   QíG.éâW<5K»ë5sðö
0050  30 6b f3 d6 40 49 08 5c cf 8d 97 75 97 59 57 d9   0kóÖ@I.\Ï..u.YWÙ
0060  4a 2d 59 55 21 63 ce b4 47 ab 4f cf 14 7a e7 8e   J-YU!cÎ´G«OÏ.zç.
0070  91 91 48 f9 60 0e 37 bd 5b 60 54 fb 18 ca 43 bc   ..Hù`.7½[`Tû.ÊC¼
0080  d9 c2 90 88 b1 50 a1 16 4d 8d a1 1e ae 16 68 8a   ÙÂ..±P¡.M.¡.®.h.
0090  44 f8 d0 cf 20 b8 09 38 81 8e cd ea 0c 91 62 1c   DøÐÏ ¸.8..Íê..b.
00a0  1e 76 76 2b 84 a2 f1 79 c0 8e b4 a4 ef 8e 09 29   .vv+.¢ñyÀ.´¤ï..)
00b0  39 3e fc 6f 43 cd 3a de 72 dd 1b 5f 73 d7 99 29   9>üoCÍ:ÞrÝ._s×.)
00c0  27 ea d3 68 e1 0f 84 b6 53 0e d2 a2 bc dd 02 cd   'êÓhá..¶S.Ò¢¼Ý.Í
00d0  89 5e bb 18 49 77 8c ff 65 7a 22 3d 6a fa ee a2   .^».Iw.ÿez"=júî¢
00e0  4f 2c 7a c0 b4 a5 d6 c1 49 50 f7 33 63 e9 5e 08   O,zÀ´¥ÖÁIP÷3cé^.
00f0  fa 44 14 92 e0 26 a5 2c aa 09 65 d3 44 26 fe ef   úD..à&¥,ª.eÓD&þï
0100  84 c1 57 5e c3 c8 74 0d 4e a9 cd 76 fa 2f 20 d9   .ÁW^ÃÈt.N©Ívú/ Ù
0110  b2 7a 2e 07 40 9c fd aa bc 8a 97 af cc 6b ee c9   ²z..@.ýª¼..¯ÌkîÉ
0120  56 8b 64 79 37 8a cf 34 e2 e2 40 7b c7 f6 cc 1c   V.dy7.Ï4ââ@{ÇöÌ.
0130  c7 18 1b 9c 2d f1 5a ce 4d de 27 6b 49 c9 5f 39   Ç...-ñZÎMÞ'kIÉ_9
0140  01 49 03 47 66 99 1a c1 17 46                     .I.Gf..Á.F
```

Ilustración 31: Captura de un paquete en Modo=1 y Mux=2

En esta imagen se aprecia perfectamente la encriptación. Las cabeceras multiplexadas, al igual que los bytes que introduce Simplemux, no son reconocibles en el paquete. Los *bytes* remarcados son, el protocolo y las direcciones origen y destino que corresponden a los *router*. No se van a incluir las capturas para una multiplexión de cuatro paquetes ni las pruebas con tráfico seguro desde PC1 por redundancia.

Para ver cuánto *overhead* introduce la seguridad en la comunicación, se va a proceder a comparar el ancho de banda utilizado en las comunicaciones previas. Para ello, se ha hecho uso de la posibilidad que ofrece Wireshark de generar una *I/O Graph*, que utiliza los paquetes de la captura para generar las variaciones de tráfico en el tiempo medido.

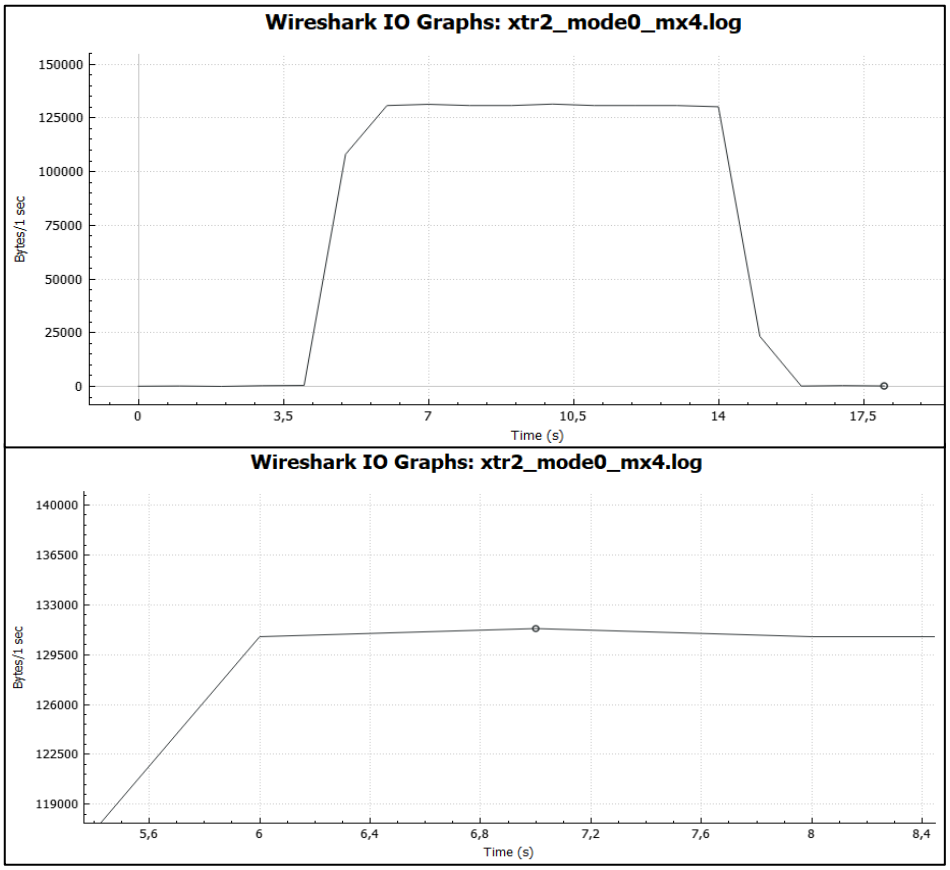


Ilustración 32: Ancho de banda utilizado en modo 0 y Mux = 4.

La imagen previa está compuesta por la gráfica mencionada y un zoom hecho para poder observar mejor el eje Y. A continuación, se adjuntan las mismas imágenes pero de la prueba hecha en modo 1.

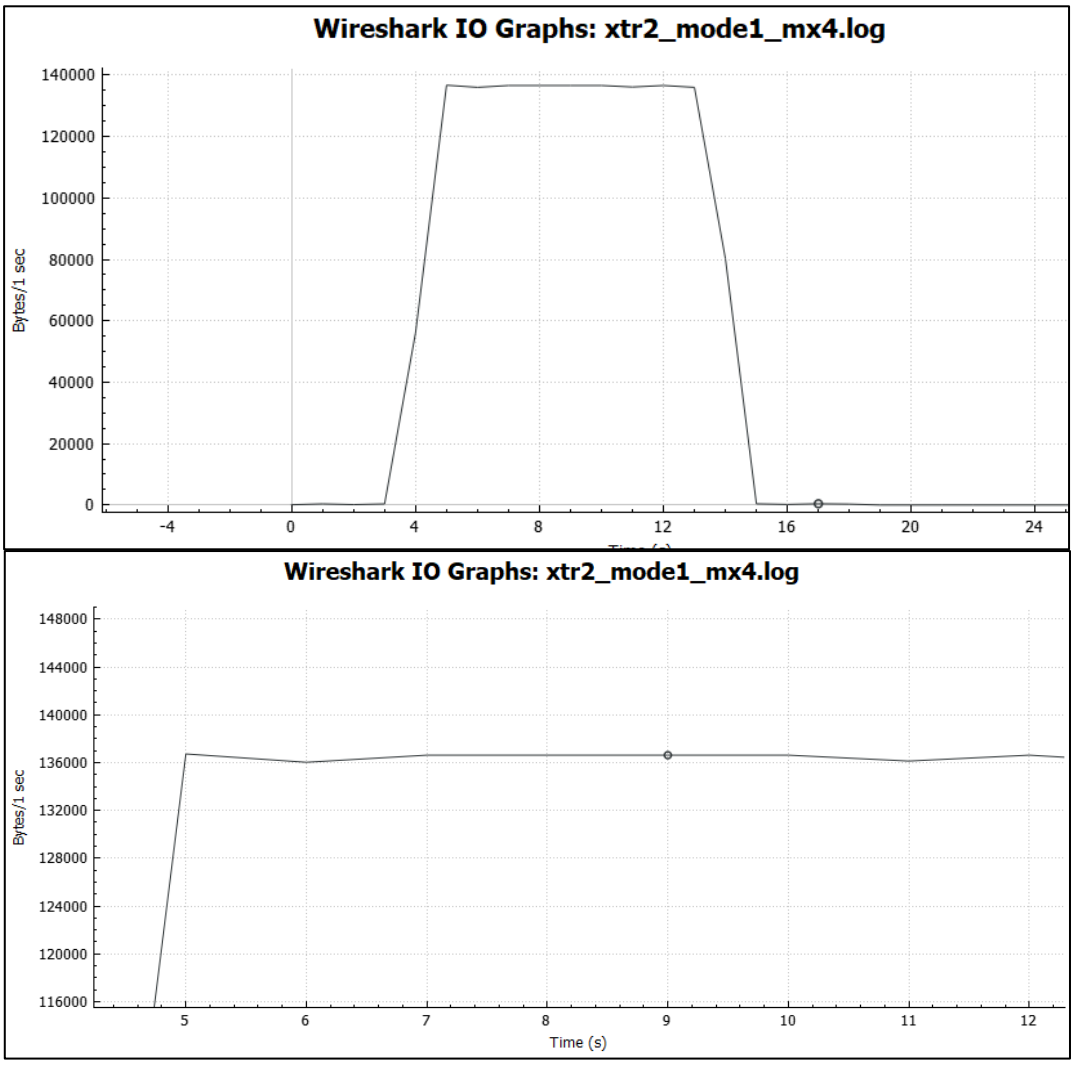


Ilustración 33: Ancho de banda utilizado en modo 1 y Mux = 4.

Mientras que con el modo 0 funcionando, se requería un ancho de banda de 131.25 kb/s, con el modo 1 incrementa hasta algo más de 136 kb/s. Al utilizar IPsec, estamos añadiendo bytes extra a los paquetes, tanto por la cabecera ESP como por los datos añadidos en la cola. Para completar este test y probar el funcionamiento dinámico de la seguridad, se ha lanzado tráfico UDP, pero durante un minuto. Mientras se desarrollaba la comunicación de PC1 a PC2, se fueron cambiando los parámetros de multiplexión y del modo de seguridad. En este caso, se ha reducido a 50 *bytes* el tamaño de los paquetes, para que la mejora por el uso de la multiplexión sea significativa visualmente en la gráfica.

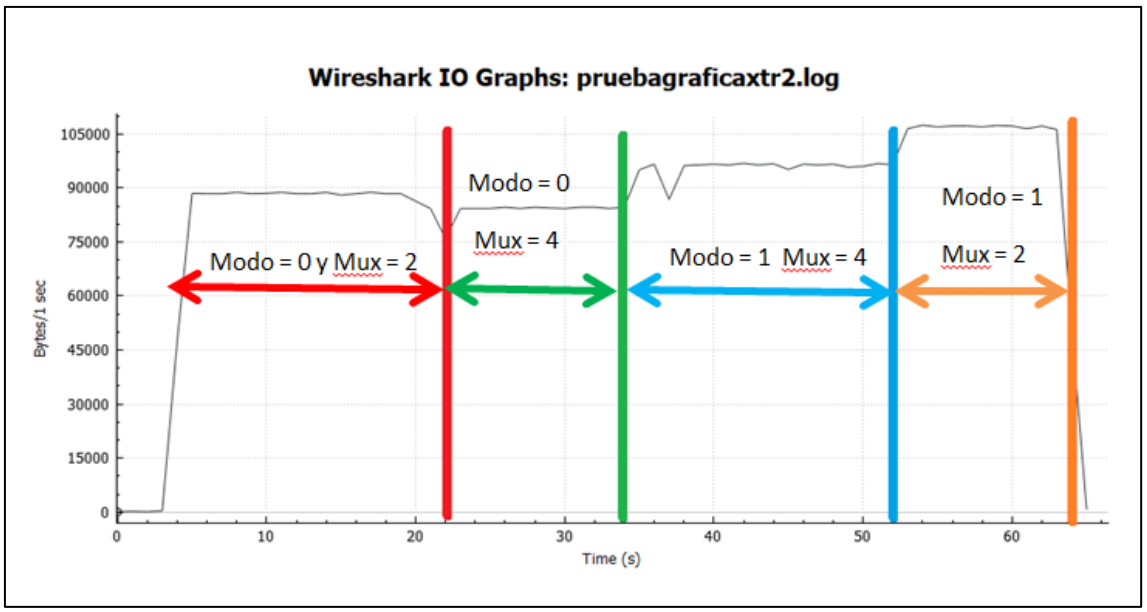


Ilustración 34: Gráfica generada por la comunicación con los cambios dinámicos

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Destinations and Ports	20390				0.3113	100%	0.5500	51.386
192.168.100.4	20372				0.3111	99.91%	0.5500	51.386
UDP	11043				0.1686	54.21%	0.5500	51.386
4343	11038				0.1685	99.95%	0.5500	51.386
4342	5				0.0001	0.05%	0.0100	1.460
NONE	9329				0.1425	45.79%	0.5100	54.356
192.168.100.3	18				0.0003	0.09%	0.0500	65.474

Ilustración 35: Uso de puertos en la prueba de cambios dinámicos

En la gráfica de la Ilustración 34 se observa que hay cuatro secciones muy diferenciadas. Las dos primeras suponen un uso de ancho de banda menor por la utilización del modo 0. Aun así, se aprecia claramente que hay una mejora multiplexando 4 paquetes con respecto a la multiplexión de 2 paquetes. En las dos últimas secciones se refleja la misma situación con el uso de la seguridad. La demanda del ancho de banda aumenta por lo comentado anteriormente sobre el *overhead* extra que introduce la seguridad y de la misma forma que pasa en el modo 0, una multiplexión de 4 paquetes supone una cierta mejora con respecto a una multiplexión de 2 paquetes. La Ilustración 35 demuestra el uso de ambos puertos, 4343 y 4344 (NONE) que han ido cambiando según el modo de seguridad elegido en cada momento.

6.3 PRUEBAS MODO=2

Para acabar este capítulo, se incluyen las pruebas realizadas con el modo 2. En primer lugar, se ha configurado la variable de ratio mínimo a 0. Como recordatorio, esto significa que solo se cifrarán los datos en el caso que el paquete de multiplexión esté formado totalmente por paquetes no seguros. En consecuencia, usando la posibilidad que ofrece D-ITG de generar flujos de datos diferentes, se ha creado un script con los siguientes tráficos:

- -a 192.168.2.1 -c 100 -T UDP -z 10 -C 1
- -a 192.168.2.1 -c 100 -T ICMP -z 1000 -C 100

Para ver correctamente el funcionamiento de este modo se ha establecido en PC1 y por tanto, también en PC2, las reglas en *ipsec-tools* que establecen un túnel en modo transporte solamente para el tráfico UDP. Es decir, PC1 cifrará el tráfico UDP saliente hacia PC2. Así, se producirá una mezcla de datos seguros con no seguros y se apreciará el funcionamiento de este modo 2.

Para esta prueba se ha cambiado un parámetro en la generación de ITGSend. En lugar de generar un tráfico que dure un determinado tiempo, se ha optado por generar un número concreto de paquetes de cada flujo y determinar la velocidad de generación de paquetes. Son los parámetros *–z* y *–C* respectivamente. En cuanto al tráfico UDP, se generará un paquete por segundo; mientras que se mandarán 100 paquetes por segundo de datos ICMP. El parámetro *–z,* número total de paquetes a generar, se ha elegido en consecuencia para que la simulación dure 10 segundos. Si se establece una multiplexión de 4 paquetes, encontraremos 10 paquetes de multiplexión formados por 1 paquete seguro y 3 no seguros, por lo tanto, no se introducirán al túnel LISP-SM securizados. Sin embargo, el resto de paquetes de multiplexión estarán formados en su totalidad por paquetes no seguros, por lo tanto, ya que el ratio está establecido a 0, deberán introducirse al túnel por el puerto 4344.

209	6.219128	192.168.2.1	192.168.1.1	ICMP	142	Echo (ping) reply id=0x0000, seq=0/0, ttl=62 (request in 203)
210	6.221554	192.168.1.1	192.168.2.1	ICMP	142	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 216)
211	6.231664	192.168.1.1	192.168.2.1	ICMP	142	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 217)
212	6.241744	192.168.1.1	192.168.2.1	ICMP	142	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 222)
213	6.251830	192.168.1.1	192.168.2.1	ICMP	142	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 223)
214	6.258040	192.168.2.1	192.168.1.1	ICMP	142	Echo (ping) reply id=0x0000, seq=0/0, ttl=62 (request in 204)
215	6.258495	192.168.2.1	192.168.1.1	ICMP	142	Echo (ping) reply id=0x0000, seq=0/0, ttl=62 (request in 205)
216	6.258891	192.168.2.1	192.168.1.1	ICMP	142	Echo (ping) reply id=0x0000, seq=0/0, ttl=62 (request in 210)
217	6.259263	192.168.2.1	192.168.1.1	ICMP	142	Echo (ping) reply id=0x0000, seq=0/0, ttl=62 (request in 211)
218	6.261917	192.168.1.1	192.168.2.1	ICMP	142	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 224)
219	6.272114	192.168.1.1	192.168.2.1	ICMP	142	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 225)
220	6.278224	192.168.1.1	192.168.2.1	ESP	162	ESP (SPI=0x00000101)
221	6.282177	192.168.1.1	192.168.2.1	ICMP	142	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 230)
222	6.288129	192.168.2.1	192.168.1.1	ICMP	142	Echo (ping) reply id=0x0000, seq=0/0, ttl=62 (request in 212)
223	6.288558	192.168.2.1	192.168.1.1	ICMP	142	Echo (ping) reply id=0x0000, seq=0/0, ttl=62 (request in 213)
224	6.288947	192.168.2.1	192.168.1.1	ICMP	142	Echo (ping) reply id=0x0000, seq=0/0, ttl=62 (request in 218)
225	6.289308	192.168.2.1	192.168.1.1	ICMP	142	Echo (ping) reply id=0x0000, seq=0/0, ttl=62 (request in 219)
226	6.292257	192.168.1.1	192.168.2.1	ICMP	142	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 231)
227	6.302325	192.168.1.1	192.168.2.1	ICMP	142	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 232)
228	6.312420	192.168.1.1	192.168.2.1	ICMP	142	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 233)
229	6.322506	192.168.1.1	192.168.2.1	ICMP	142	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 238)
230	6.329768	192.168.2.1	192.168.1.1	ICMP	142	Echo (ping) reply id=0x0000, seq=0/0, ttl=62 (request in 221)
231	6.330204	192.168.2.1	192.168.1.1	ICMP	142	Echo (ping) reply id=0x0000, seq=0/0, ttl=62 (request in 226)

Ilustración 36: Captura que muestra los dos flujos de tráfico generados desde PC1

51	6.052104	192.168.100.4	192.168.100.3	ESP	594 ESP (SPI=0x00000302)
52	6.088481	192.168.100.3	192.168.100.4	ESP	594 ESP (SPI=0x00000202)
53	6.092438	192.168.100.4	192.168.100.3	ESP	594 ESP (SPI=0x00000302)
54	6.128581	192.168.100.3	192.168.100.4	ESP	594 ESP (SPI=0x00000202)
55	6.132716	192.168.100.4	192.168.100.3	ESP	594 ESP (SPI=0x00000302)
56	6.168984	192.168.100.3	192.168.100.4	ESP	594 ESP (SPI=0x00000202)
57	6.172928	192.168.100.4	192.168.100.3	ESP	594 ESP (SPI=0x00000302)
58	6.209184	192.168.100.3	192.168.100.4	ESP	594 ESP (SPI=0x00000202)
59	6.213285	192.168.100.4	192.168.100.3	ESP	594 ESP (SPI=0x00000302)
60	6.249655	192.168.100.3	192.168.100.4	ESP	594 ESP (SPI=0x00000202)
61	6.253671	192.168.100.4	192.168.100.3	ESP	594 ESP (SPI=0x00000302)
62	6.279588	192.168.100.3	192.168.100.4	UDP	591 4343 → 4343 Len=549
63	6.283588	192.168.100.4	192.168.100.3	ESP	594 ESP (SPI=0x00000302)
64	6.320396	192.168.100.3	192.168.100.4	ESP	594 ESP (SPI=0x00000202)
65	6.325021	192.168.100.4	192.168.100.3	ESP	594 ESP (SPI=0x00000302)
66	6.360700	192.168.100.3	192.168.100.4	ESP	594 ESP (SPI=0x00000202)
67	6.365661	192.168.100.4	192.168.100.3	ESP	594 ESP (SPI=0x00000302)
68	6.401021	192.168.100.3	192.168.100.4	ESP	594 ESP (SPI=0x00000202)
69	6.406206	192.168.100.4	192.168.100.3	ESP	594 ESP (SPI=0x00000302)
70	6.441501	192.168.100.3	192.168.100.4	ESP	594 ESP (SPI=0x00000202)
71	6.446358	192.168.100.4	192.168.100.3	ESP	594 ESP (SPI=0x00000302)
72	6.481812	192.168.100.3	192.168.100.4	ESP	594 ESP (SPI=0x00000202)
73	6.486729	192.168.100.4	192.168.100.3	ESP	594 ESP (SPI=0x00000302)

Ilustración 37: Captura del tráfico dentro del túnel LISP-SM con modo 2 y ratio = 0

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Destinations and Ports	522				0.0234	100%	0.0600	5.234
192.168.100.4	262				0.0117	50.19%	0.0300	5.261
UDP	11				0.0005	4.20%	0.0100	1.910
4343	10				0.0004	90.91%	0.0100	5.290
4342	1				0.0000	9.09%	0.0100	1.910
NONE	251				0.0112	95.80%	0.0300	5.321
192.168.100.3	260				0.0116	49.81%	0.0300	5.234

Ilustración 38: Estadísticas de uso de puertos obtenidas de Wireshark

En la ilustración 38, que recoge las estadísticas de uso de los puertos de la captura, se corrobora la hipótesis antes mencionada. Se han enviado 10 paquetes por el puerto 4343 y el resto por el puerto 4344, que aparece como NONE ya que se encuentra cifrado en el paquete y Wireshark no puede descifrarlo. Para complementar, se adjuntan a continuación capturas del *debug* realizado por xTR1 durante la comunicación, en las cuales diferencia el tipo de tráfico que le llega.

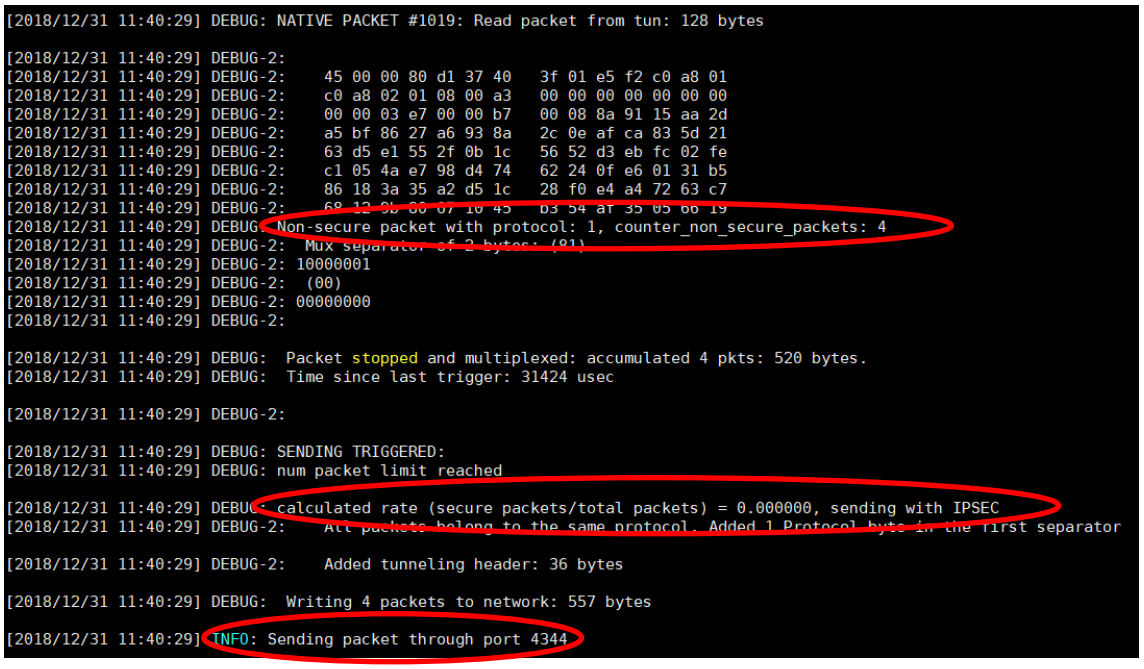


Ilustración 39: Captura del debug de xTR1 para el modo 2 y ratio = 0

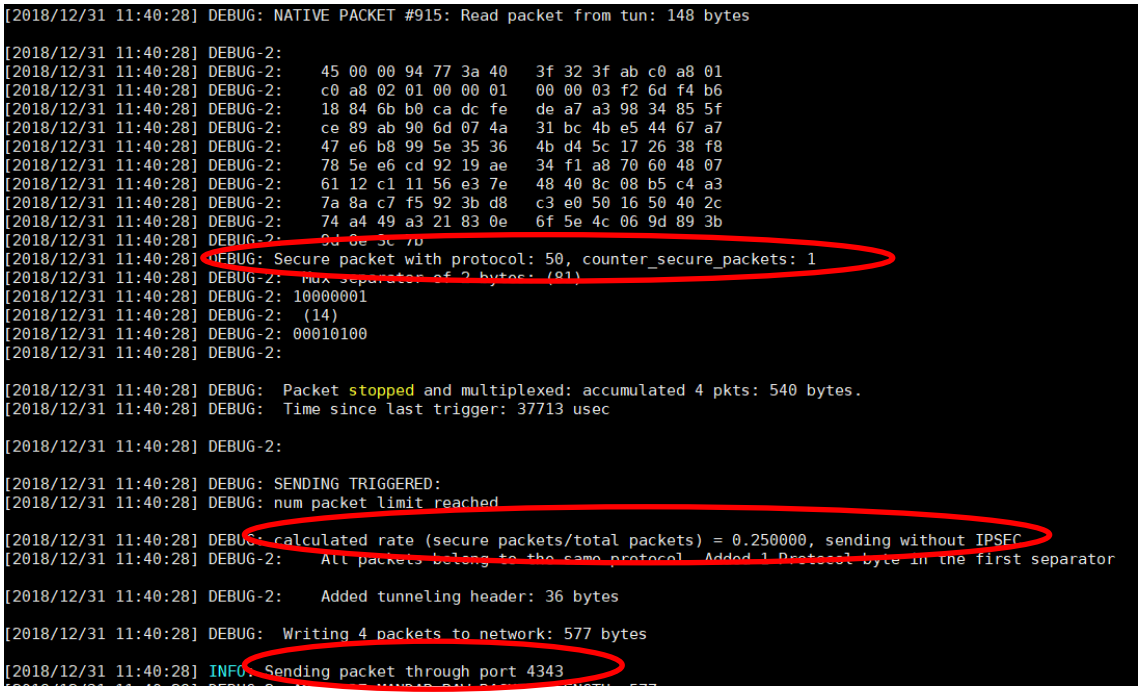


Ilustración 40: Captura del debug de xTR1 para el modo 2 y ratio = 0

Estas dos últimas imágenes, presentan los dos casos planteados. En la primera, llega un paquete ICMP, por lo tanto no seguro y como es el último que forma el paquete de multiplexión, se calcula el ratio. Al ser los cuatro paquetes no seguros, se envía por el puerto 4344. En la siguiente imagen, Ilustración 40, se presenta el caso contrario. Llega un paquete ESP, protocolo 50, el cual es el último para formar el paquete final. El ratio calculado es mayor que 0. En consecuencia, se elige el puerto 4343.

Para la prueba con el ratio 1, si se forma un paquete de multiplexión con uno no seguro se aplicará seguridad, se ha utilizado el tráfico opuesto al usado con el ratio 0:

- -a 192.168.2.1 -c 100 -T UDP -z 1000 -C 100
- -a 192.168.2.1 -c 100 -T ICMP -z 10 -C 1

En esta situación, se han generado 1000 paquetes UDP que se encriptarán y 10 paquetes no seguros ICMP. Se generarán 100 paquetes seguros por cada uno no seguro durante diez segundos. En consecuencia, con la misma multiplexión de 4 paquetes de la anterior prueba, tendremos diez paquetes ESP en el túnel mientras que el resto serán UDP.

409 7.490002	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
410 7.500089	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
411 7.510222	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
412 7.520326	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
413 7.530426	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
414 7.540498	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
415 7.550627	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
416 7.560747	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
417 7.570851	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
418 7.580967	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
419 7.591107	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
420 7.601213	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
421 7.604450	192.168.2.1	192.168.1.1	ICMP	142 Echo (ping) reply	id=0x0000, seq=0/0, ttl=62 (request in 321)
422 7.608106	192.168.1.1	192.168.2.1	ICMP	142 Echo (ping) request	id=0x0000, seq=0/0, ttl=64 (reply in 522)
423 7.611322	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
424 7.621449	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
425 7.631568	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
426 7.641682	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
427 7.651791	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
428 7.661931	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
429 7.672009	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
430 7.682089	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	
431 7.692189	192.168.1.1	192.168.2.1	ESP	162 ESP (SPI=0x00000101)	

Ilustración 41: Tráfico generado desde PC1

24 3.947018	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
25 3.987154	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
26 4.027533	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
27 4.067936	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
28 4.108327	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
29 4.148856	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
30 4.189444	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
31 4.229552	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
32 4.269814	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
33 4.310313	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
34 4.326702	192.168.100.4	192.168.100.3	ESP	202 ESP (SPI=0x00000302)
35 4.340833	192.168.100.3	192.168.100.4	ESP	650 ESP (SPI=0x00000202)
36 4.380720	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
37 4.421236	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
38 4.461492	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
39 4.502111	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
40 4.542175	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
41 4.582532	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
42 4.622850	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
43 4.663519	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
44 4.703545	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
45 4.744140	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609
46 4.784248	192.168.100.3	192.168.100.4	UDP	651 4343 → 4343 Len=609

Ilustración 42: Tráfico con modo 2 y ratio = 1 en el túnel

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Destinations and Ports	280				0.0128	100%	0.0400	3.326
192.168.100.4	261				0.0119	93.21%	0.0300	3.351
UDP	242				0.0111	92.72%	0.0300	3.422
4343	241				0.0110	99.59%	0.0300	3.422
4342	1				0.0000	0.41%	0.0100	21.844
NONE	19				0.0009	7.28%	0.0200	3.351
0	19				0.0009	100.00%	0.0200	3.351
192.168.100.3	19				0.0009	6.79%	0.0100	0.322

Ilustración 43: Estadísticas de puertos de Wireshark

Se observa que el uso del puerto 4343 es casi del 100%, ya que casi todos los paquetes estaban formados por paquetes seguros UDP. En este caso vemos que se han mandado por el puerto 4344 19 paquetes hacia xTR2. Entre esos paquetes, hay 9 que están formados o bien por el control de TCP que realiza D-ITG para el establecimiento y la finalización de la comunicación, o bien por paquetes formados por algún ACK y paquetes UDP, por lo tanto al no estar compuesto completamente por paquetes seguros, se securizan.

```
[2018/12/31 11:12:15] DEBUG: NATIVE PACKET #922: Read packet from tun: 57 bytes

[2018/12/31 11:12:15] DEBUG-2:
[2018/12/31 11:12:15] DEBUG-2:    45 00 00 39 22 cc 40   3f 06 94 a0 c0 a8 01
[2018/12/31 11:12:15] DEBUG-2:    c0 a8 02 01 de 51 23   cd af 5b f8 5d 91 7c
[2018/12/31 11:12:15] DEBUG-2:    80 18 01 c9 46 86 00   01 01 08 0a 4f ed 85
[2018/12/31 11:12:15] DEBUG-2:    4f ed 79 43 04 00 00   02
[2018/12/31 11:12:15] DEBUG: Non-secure packet with protocol: 6, counter_non_secure_packets: 1
[2018/12/31 11:12:15] DEBUG-2:  MUX separator of 1 byte: (81)
[2018/12/31 11:12:15] DEBUG-2: 00111001
[2018/12/31 11:12:15] DEBUG-2:

[2018/12/31 11:12:15] DEBUG:  Packet stopped and multiplexed: accumulated 4 pkts: 508 bytes.
[2018/12/31 11:12:15] DEBUG:  Time since last trigger: 34996 usec

[2018/12/31 11:12:15] DEBUG-2:

[2018/12/31 11:12:15] DEBUG: SENDING TRIGGERED:
[2018/12/31 11:12:15] DEBUG: num packet limit reached

[2018/12/31 11:12:15] DEBUG: calculated rate (secure packets/total packets) = 0.750000, sending with IPSEC
[2018/12/31 11:12:15] DEBUG-2:    All packets belong to the same protocol. Added 1 Protocol byte in the first separator

[2018/12/31 11:12:15] DEBUG-2:    Added tunneling header: 36 bytes

[2018/12/31 11:12:15] DEBUG:  Writing 4 packets to network: 545 bytes

[2018/12/31 11:12:15] INFO: Sending packet through port 4344
```

Ilustración 44: Captura del debug de xTR1 para el modo 2 y ratio = 1

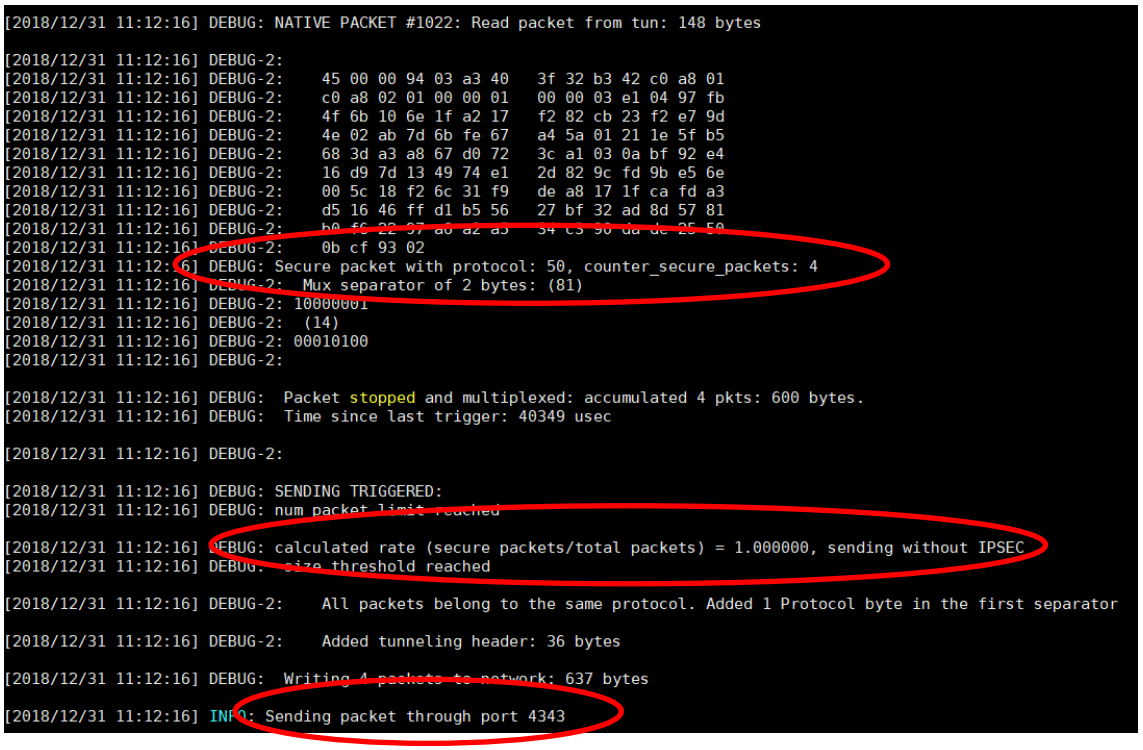


Ilustración 45: Debug de xTR1 para el modo 2 y ratio = 1

En el *debug*, aparece precisamente la situación contraria al caso con ratio 0. En la ilustración 44 llega un paquete no seguro para enviarse con tres seguros. Ya que el ratio no llega a 1, se envía por el puerto 4344. En la imagen 45, llega un paquete ESP. Como los otros tres paquetes también son seguros, el ratio calculado es 1, por lo tanto se envía sin securizar para no introducir seguridad redundante por el puerto 4343.

7. SUBIDA A GITHUB

El último paso de este TFG consiste en subir el código modificado al repositorio de Github inicial. Antes de ello, se ha limpiado el código realizado y se ha comentado. Para saber usar GitHub, se realizó un curso MOOC (“*Gestión de proyectos Software con Git y GitHub”)* gratuito ofrecido por la Universidad Politécnica de Madrid a través de la plataforma https://miriadax.net/cursos . En el Anexo C se adjunta el diploma que acredita la realización de dicho curso.

Ya que se realizaron cambios en algunas funciones para el envío de datos, el código se ha subido a una rama distinta de la *master*. Para ello se ha creado la rama *security*.

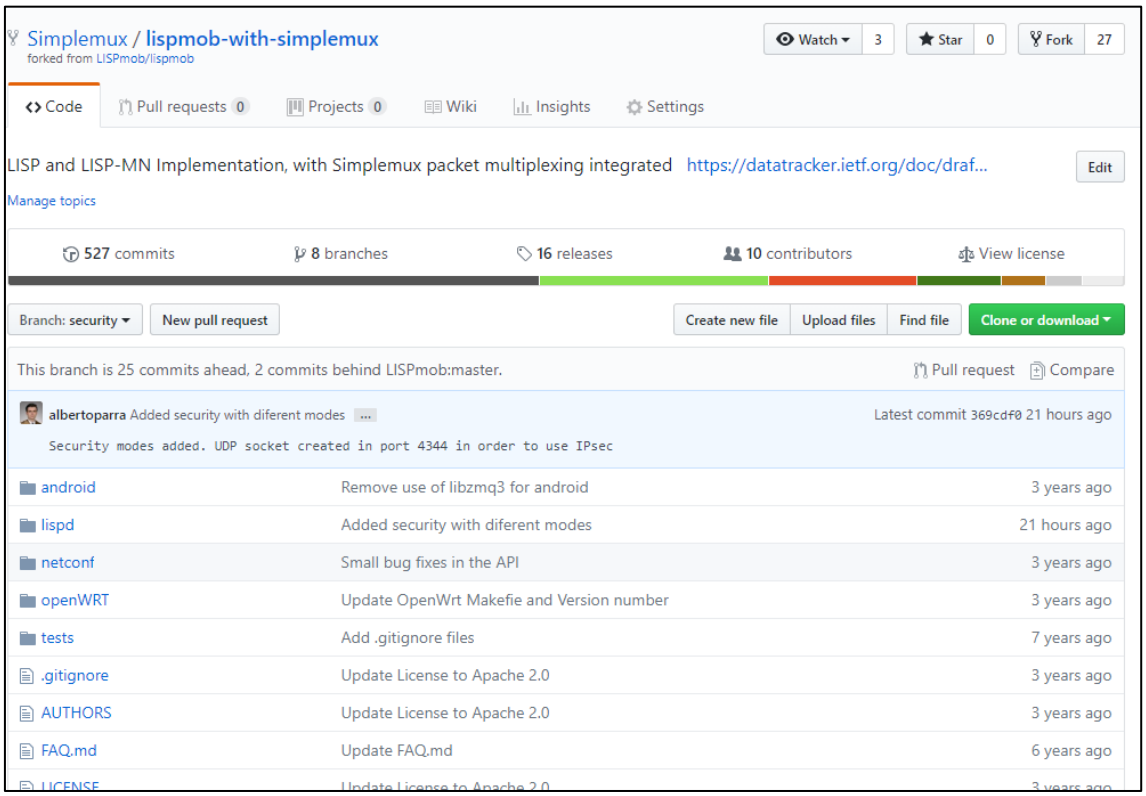


Ilustración 226: Código subido al repositorio de GitHub

8. CONCLUSIONES Y LÍNEAS FUTURAS

8.1. CONCLUSIONES

Para este TFG se planteó como objetivo la creación de túneles seguros LISP-SM y que fueran configurables dinámicamente. Para ello, tras la comprensión del código que crea los túneles LISP-SM, se han introducido los cambios necesarios en los archivos pertinentes para que dicho código admita el uso de un nuevo puerto para el envío de tráfico seguro y la posibilidad de elegir entre diferentes modos de seguridad.

Para la comprobación del funcionamiento correcto, se realizaron una serie de pruebas en el escenario de red durante las cuales, se han adquirido conocimientos sobre el uso de diferentes *software* y se encontró un fallo inesperado en el uso de IPsec con *raw sockets.* Tras el estudio de los resultados, se ha verificado el funcionamiento correcto de las funcionalidades añadidas, así como del funcionamiento dinámico de los modos de seguridad. Además, gracias a la inclusión del modo 2 de seguridad, se puede realizar una securización del tráfico más eficiente en situaciones donde el ancho de banda sea limitado.

Por otro lado, se ha reconfirmado el ahorro en el ancho de banda con el uso de la multiplexión y se ha podido observar el impacto extra del *overhead* introducido por la seguridad.

Por último, se realizó un curso para adquirir conocimientos sobre el uso de *git* y Github que resultó necesario para poder colaborar en el repositorio *Lispmob-with-Simplemux* y añadir los cambios desarrollados.

8.2 LÍNEAS FUTURAS

Como continuación de este TFG se presentan varias alternativas posibles:

- Investigar el porqué del fallo en el uso conjunto de IPsec + *raw sockets*.
- Realizar un análisis más exhaustivo de la mejora del uso del ancho de banda dependiendo del modo de seguridad y multiplexión elegida.
- Añadir nuevas funcionalidades al proyecto. Por ejemplo, nuevos modos u opciones de seguridad. Otra opción es crear nuevas opciones en la configuración de Simplemux, como puede ser, elegir formar el túnel de multiplexión dependiendo de los puertos o protocolos en lugar de direcciones IP o redes.
- Automatizar los cambios en los ficheros de configuración mediante un script, por ejemplo para Vi, que edite las líneas donde se encuentran los parámetros de número de paquetes a multiplexar y el modo de seguridad cada cierto tiempo.

9. REFERENCIAS Y BIBLIOGRAFÍA

9.1 REFERENCIAS

España, M.C. (2003). *Servicios avanzados de telecomunicación*. Madrid: Díaz de Santos

Farinacci, D., Fuller, V., Meller, D. y Lewis D. (2013). *The Locator/ID Separation Protcol (LISP) IETF-RFC 6830 Internet Engineering Task Force*. Recuperado de: https://tools.ietf.org/html/rfc6830

9.2 BIBLIOGRAFÍA

Botta A., Donato W., Dainotti A, Avallone S. y Pescapé A. (2013). *D-ITG 2.8.1 Manual.* Recuperado de: http://www.grid.unina.it/software/ITG/manual/

Francisconi, H.A. (2005). *IPsec en Ambientes IPv4 e IPv6*. Recuperado de: https://francisconi.org/sites/default/files/IPsec_en_Ambientes_IPv4_e_IPv6.pdf

Fundación Telefónica. Informe sobre la Sociedad Digital en España 2017. Recuperado de : https://www.fundaciontelefonica.com/arte_cultura/sociedad-de-la-informacion/sdie-2017/

GitHub – Lispmob-with-simplemux. Recuperado de: https://github.com/Simplemux/lispmob-with-simplemux

IETF – Internet Draft Internet Engineering Task Force. Recuperado de: https://tools.ietf.org/html/draft-saldana-tsvwg-simplemux-10

IETF-RFC 6830 Internet Engineering Task Force. Recuperado de: https://tools.ietf.org/html/rfc6830

IPsecHowTo. Recuperado de: https://help.ubuntu.com/community/IPSecHowTo

Kurose, J.F. Ros, K.W. (2013). *Computer networking*. Pearson

Manpage Setkey. Recuperado de: https://manpages.debian.org/testing/ipsec-tools/setkey.8.en.html

Network Working Group –RFC 4984

Programación de sockets en C de Unix/Linux. Recuperado de: http://www.chuidiang.org/clinux/sockets/sockets_simp.php#cliente