Propuesta de un Mecanismo de Seguridad para el Intercambio de Datos de Usuario en Redes de Próxima Generación

Universidad del Cauca

Jaime Andrés Oliva Ortega
Fabio Joaquín Fuertes Montenegro

ANEXO C
PRUEBAS DE SEGURIDAD

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Popayán, Octubre de 2009

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

1. Introducción

En este anexo se detallan las pruebas de seguridad llevadas a cabo según las actividades planteadas en el capítulo 5 de la monografía, específicamente en 5.2.3. Estas pruebas han sido adaptadas del manual OSSTMM al presente proyecto con el fin de realizar una evaluación exhaustiva de varios aspectos que tienen que ver con la correcta prestación de los servicios de seguridad en una red IP, como es el caso de IMS.

Para cada prueba, se presenta un objetivo que resume la finalidad de la misma y un procedimiento donde se describen los pasos realizados para alcanzar dicho objetivo. Este procedimiento se aplica sobre uno o más hosts del prototipo de pruebas descrito en 5.2.5.

2. Prueba de Actualización

Objetivo: Verificar si las versiones del software instaladas en los equipos corresponden a las versiones actualizadas o estables de los paquetes y del sistema operativo, con el fin de descartar posibles agujeros de seguridad presentes en versiones inestables o antiguas.

Cabe aclarar que estas pruebas se realizaron el día 15 de agosto de 2009, por lo tanto a la fecha de revisión de este documento es probable que existan algunas versiones de paquetes más recientes.

Procedimiento

En cada equipo, el auditor con privilegios de *root* realiza una revisión de las versiones tanto del sistema operativo como de los paquetes de la red IMS base e IMSeg.

En la Tabla 1. se hace una relación entre el software que se va revisar, con las versiones más actualizadas a la fecha, disponibles en los repositorios oficiales de la distribución estable de Debian (Lenny).

Tabla 1. Versiones estables del software a revisar en la prueba de Actualización

Software	Versión
Debian	5.0.2 Lenny
Open IMS Core (*)	Ultima actualización: octubre de 2008
Java	1.6.12
MySQL	5.0.5
Sequoia (*)	2.10.10
Bea Weblogic SIP Server (*)	3.1

Software	Versión
strongSwan	4.2.4
Iptables	1.4.2.6
Kernel de Linux	2.6.26-2
OpenSSH-server	5.1

Nota: (*) Significa que este software no existe en los repositorios de Debian, por lo tanto para su instalación es necesario descargarlo de los sitios Web oficiales de las organizaciones desarrolladoras del mismo y seguir el procedimiento de instalación descrito en la documentación del programa.

A continuación se muestran ejemplos de los comandos utilizados para realizar estas pruebas de verificación para cada uno de los elementos mostrados en la Tabla 1. .

- Sistema Operativo

La Figura 1. muestra la verificación de la versión del sistema operativo Debian en *dragoon*. Como resultado se muestra la versión 5.0.2; versión estable actualizada de Debian.

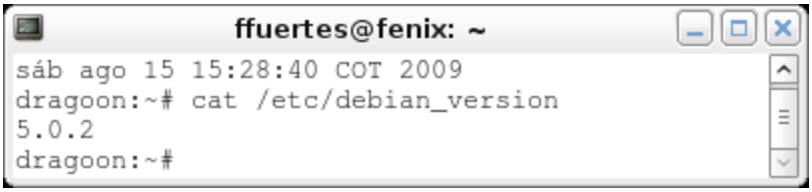

Figura 1. Verificación de la versión del S.O. Debian

- Open IMS Core

Esta implementación del Core de IMS utiliza la herramienta *svn* como sistema de control de versiones. La última actualización de la rama *trunk* o estable tanto del CSCF (ser_ims) como del HSS (FHoSS) corresponde a octubre del 2008.
La Figura 2. muestra la fecha de instalación de los directorios de Open IMS Core en *fenix*, los cuales corresponden a febrero de 2009, por tanto ya que la instalación se hizo a través de *svn,* se tiene la versión actualizada de esta implementación.

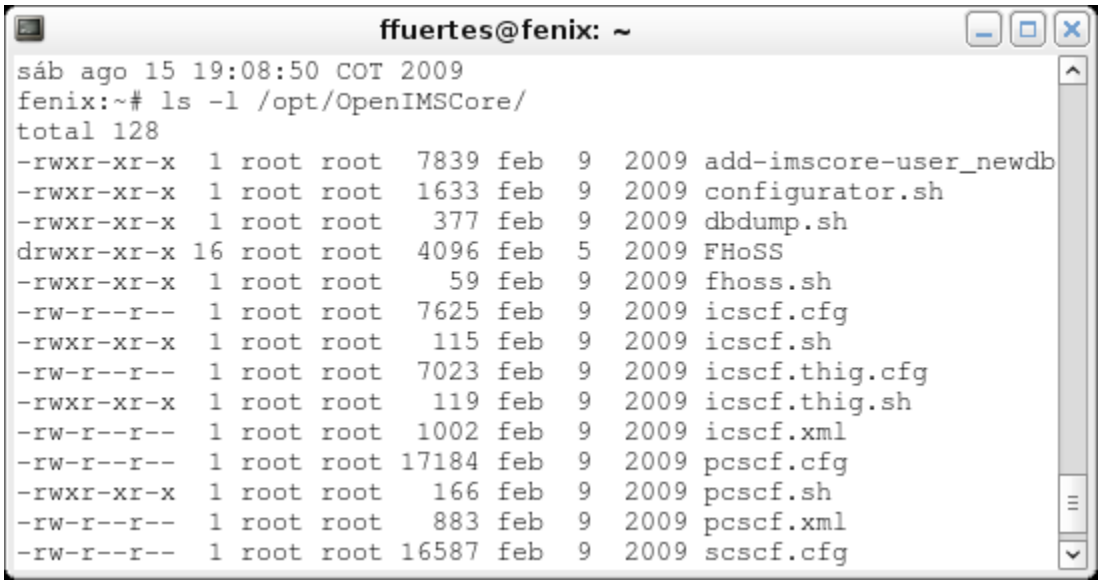

Figura 2. Fecha de Instalación de Open IMS Core en *fenix*

Para el resto del software mostrado en la Tabla 1. , las siguientes figuras corroboran que los equipos tienen instaladas las versiones actualizadas.

- Java

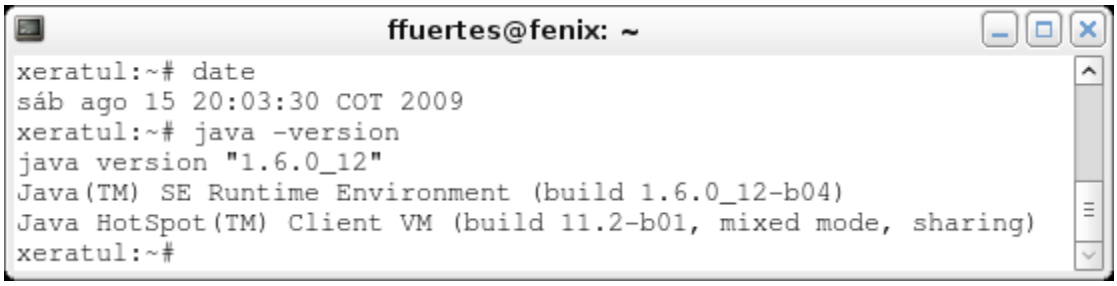

Figura 3. Verificación de la versión actualizada de Java en *xeratul*

- MySQL

Figura 4. Verificación de la versión actualizada de MySQL en *fenix*

- Sequoia

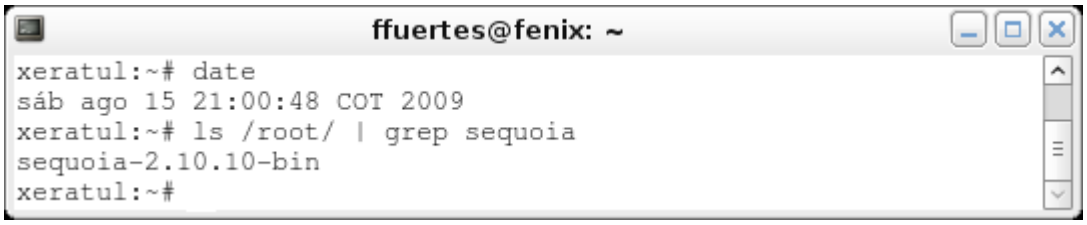

Figura 5. Verificación de la versión actualizada de Sequoia en *xeratul*

- Bea Weblogic Server

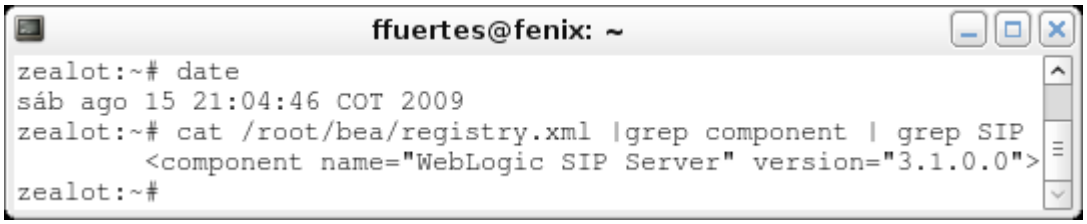

Figura 6. Verificación de la versión actualizada de Bea Weblogic SIP Server en *zealot*

- strongSwan

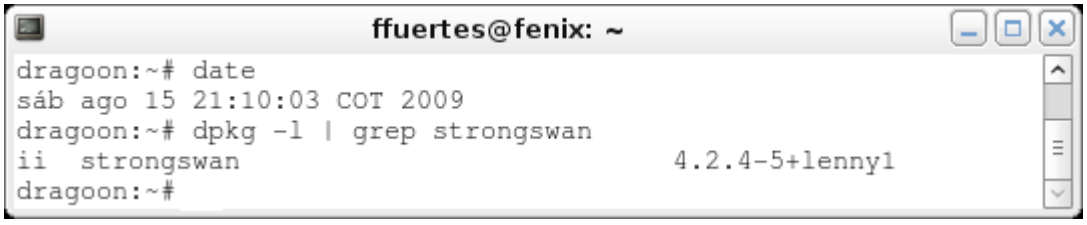

Figura 7. Verificación de la versión actualizada de strongSwan en *dragoon*

- Iptables

Figura 8. Verificación de la versión actualizada de Iptables en *zealot*

- OpenSSH-Server

Figura 9. Verificación de la versión actualizada de OpenSSH-Server en *dragoon*

- Kernel de Linux

Figura 10. Verificación de la versión actualizada del Kernel de Linux en *zealot*

En la Tabla 1. se muestra que la versión del Kernel actualmente estable para Debian es 2.6.26-2, sin embargo en la Figura 10. se observa que la versión instalada en *zealot* (igualmente en los demás hosts) es 2.6.27.10grsec1.0-grsec, este al contrario de presentar inconvenientes ofrece ventajas de seguridad que el Kernel estándar no tiene por defecto, ya que se trata de un *kernel endurecido* como se mostró en 4.3.1.

3. Prueba de Visibilidad

Objetivo: Revisar la *visibilidad* de los PC objetivo desde el exterior hacia la red prototipo, es decir verificar si cada uno de los cuatro hosts pueden ser alcanzados por un equipo externo que no tiene privilegios para llegar a ellos, ya sea mediante ICMP (ping) o conexión a puertos TCP/UDP. Este test pone a prueba el buen funcionamiento de los firewalls locales de los equipos.

Procedimiento

Para llevar a cabo estas pruebas en primer lugar se utilizará el comando *ping,* el cual generalmente se usa para obtener una respuesta a una petición de eco en un equipo remoto y así saber si este es alcanzable. Posteriormente se utilizará la herramienta Nmap para realizar una exploración de puertos TCP/UDP más profunda.

- Ping

Para esta prueba, en primer lugar se configuró el firewall local de *fenix* para tener acceso SSH hacia el equipo auditor (nombre del equipo: *koopa*, Dirección IP: 10.210.1.167), añadiendo la siguiente línea al script del firewall:

```
iptables -t filter -A OUTPUT -d 10.210.1.167 -p tcp --dport 22 -m state
--state NEW -j ACCEPT
```

Esto con el fin de tener en un mismo equipo, la consola de *fenix* que es un equipo con privilegios y la consola remota del equipo auditor. Una vez hecho esto se accede a *koopa* y desde allí se realizan las pruebas mostradas en las Figuras 11 y 12.

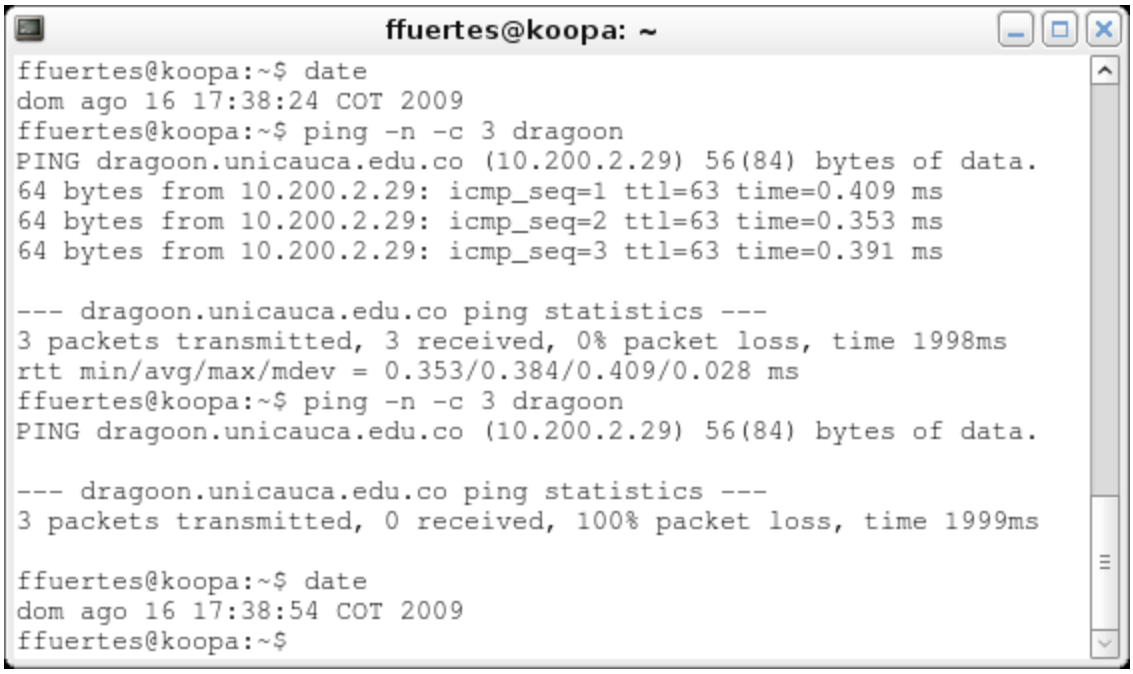

Figura 11. Ping desde *koopa* hacia *dragoon* antes y después de activar el Firewall

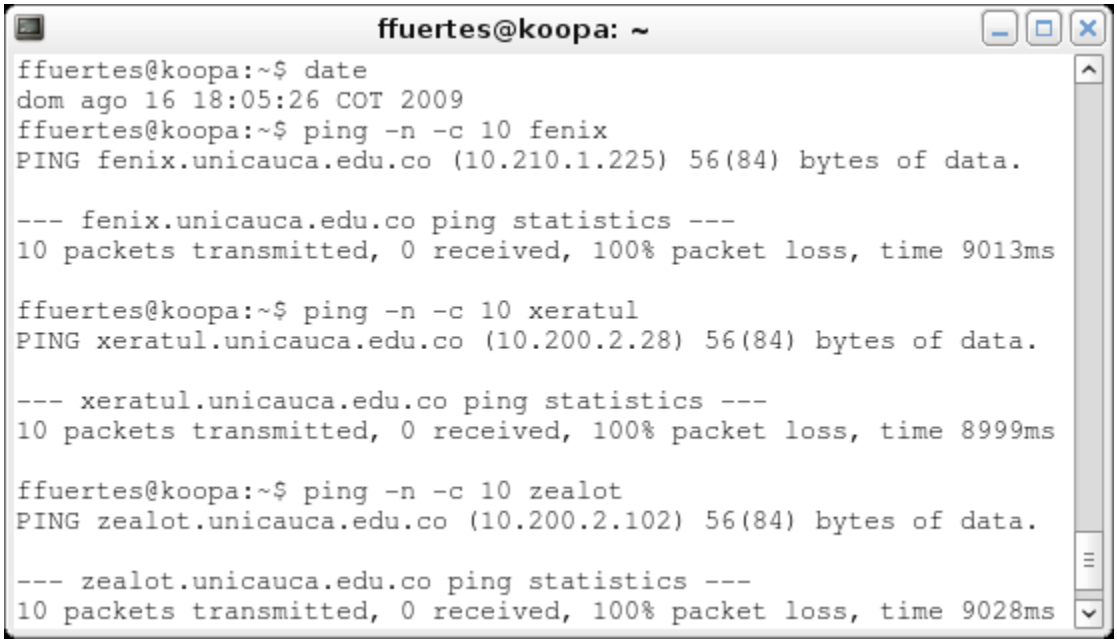

Figura 12. Ping desde koopa hacia los equipos de la red prototipo

En la Figura 11. se puede observar que inicialmente *dragoon* responde al ping desde *koopa*, esto gracias a que se tiene en desactivado el firewall local. Después del primer ping se activa el firewall y el segundo comando ping desde *koopa* no obtiene ninguna respuesta.

Del mismo modo la Figura 12. muestra los resultados de la prueba hacia los demás equipos de la red con el firewall activado, enviando 10 solicitudes sin obtener éxito en ninguna de ellas. Con esto se prueba que los equipos no son *visibles* al comando ping, el cual es el más utilizado incluso por herramientas como Nmap para descubrir hosts activos en una red.

- Nmap

Para esta prueba se realiza una exploración o escaneo de puertos desde el equipo auditor con le fin de determinar si existen puertos abiertos en el PC objetivo. En el primer ejemplo, mostrado en la Figura 13. , el auditor se encuentra logueado en *zealot,* desde allí realiza una conexión SSH hacia koopa, posteriormente realiza un escaneo de todos los puertos hacia en el objetivo (*zealot*), el cual tiene desactivado el firewall local, como se puede ver en la grafica se encuentran tres puertos TCP abiertos: 2022, 5060, 7001, correspondientes a los servicios SSH, SIP y HTTP, los dos últimos abiertos por el servidor Weblogic.

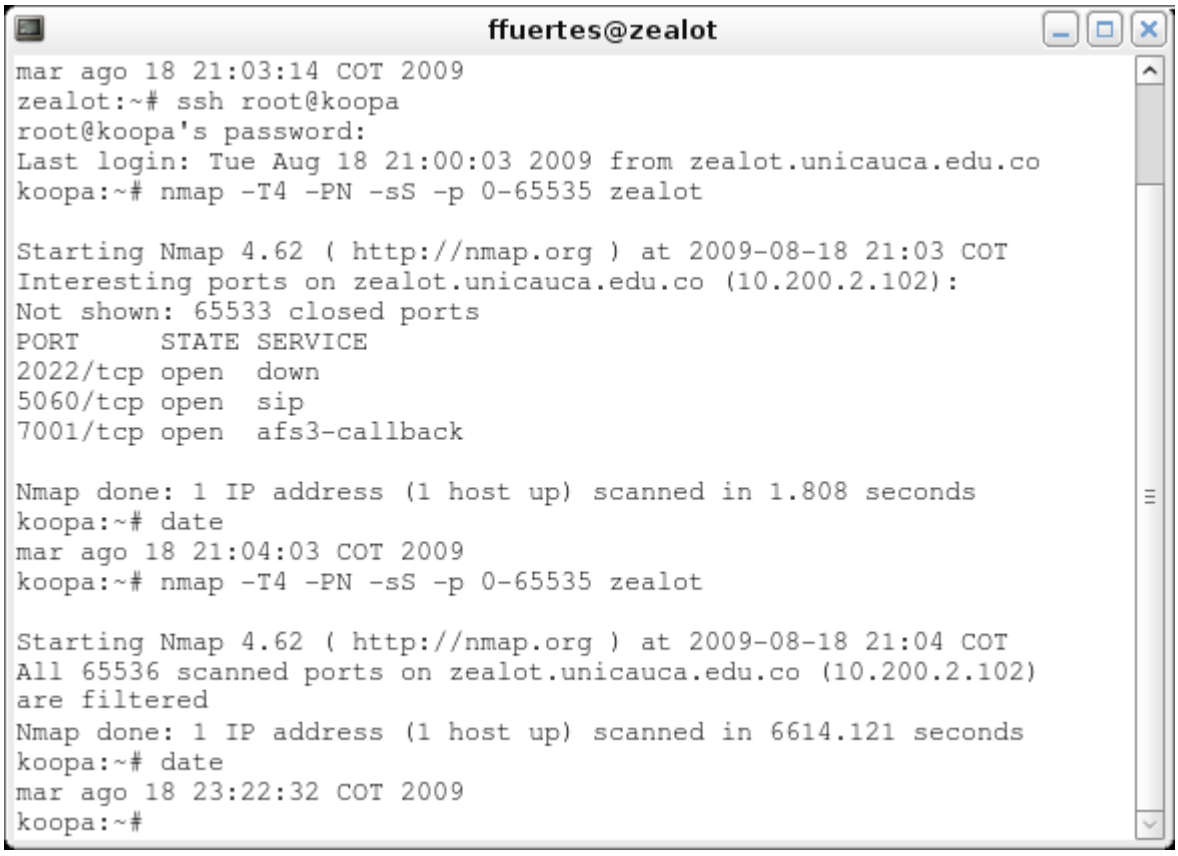

Figura 13. Exploración de puertos con Nmap antes y después de activar el firewall

Además de lo anterior, En la grafica se observa que después de la primera exploración de puertos se intenta repetir ese procedimiento con las mismas opciones y hacia el mismo objetivo, pero esta vez el firewall en *zealot* ha sido activado previamente. Entre las reglas de iptables dadas a las máquinas se encuentra una, que le indica al equipo que acepte todas las conexiones ya establecidas y solo bloquee las nuevas conexiones, por esta razón no se pierde el acceso hecho previamente mediante SSH desde *zealot* hacia *koopa* aunque el firewall este arriba, sin embargo, como lo corrobora la gráfica, la segunda exploración de puertos toma 6614.121 segundos (1 hora con 50 minutos aproximadamente), dando como resultado que los 65535 puertos escaneados se encuentran filtrados. Para Nmap el estado "filtrado" de un puerto es el que menos información ofrece, pues indica que no obtuvo ninguna respuesta a las peticiones de conexión al puerto ni siquiera con varias retransmisiones, por lo tanto no puede determinar si los puertos están siendo bloqueados por la maquina objetivo, por un firewall que se encuentra en esa ruta, por las reglas de algún enrutador o si el equipo se encuentra apagado (lo cual se descarta por el hecho de que la conexión SSH realizada de *zealot* a *koopa* se encuentra activa).

En la Figura 14. se presenta el mismo escaneo anterior, pero esta vez solo hacia los puertos que se sabe de antemano están abiertos en el objetivo (2022, 5060, 7001), mostrando el mismo resultado (*filtered*).

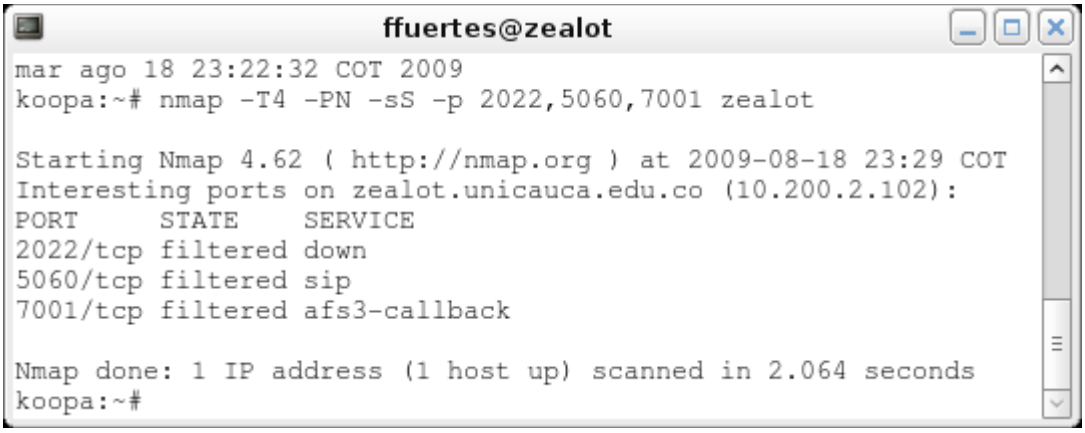

Figura 14. Exploración de los puertos abiertos en el host *zealot*

La Figura 15. muestra un escaneo con opciones más avanzadas de Nmap. El sondeo Xmas (-sX), al igual que otros sondeos como NULL, FIN y Maimon, fija ciertos bits del paquete enviado hacia el objetivo con el fin de que este le conteste con un mensaje RST si el puerto se encuentra cerrado mientras que no envía ninguna respuesta si el puerto se encuentra abierto, por consiguiente el estado *open|filtered* mostrado en la gráfica puede significar que el puerto está abierto o que el equipo no responde por alguna razón (podría incluso estar apagado o no existir). Para tener claridad en esto se ha incluido en la lista de puertos a sondear un puerto arbitrario que se encuentra cerrado en el equipo (58438), mostrando el mismo resultado.

Así mismo las opciones –sV y –O intentan determinar versiones de servicios y sistema operativo respectivamente, sin tener ningún resultado satisfactorio. Todo lo anterior confirma que Nmap no puede determinar si el equipo se encuentra activo y mucho menos

revelar información sobre servicios o sistema operativo corriendo en el Objetivo.

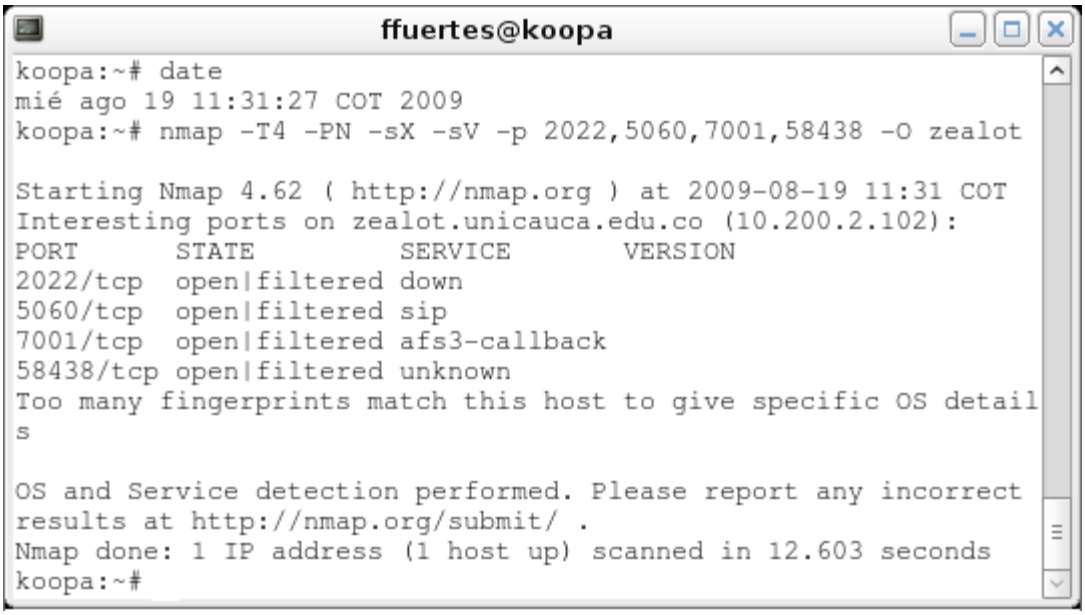

Figura 15. Exploración avanzada de puertos abiertos en el host *zealot*

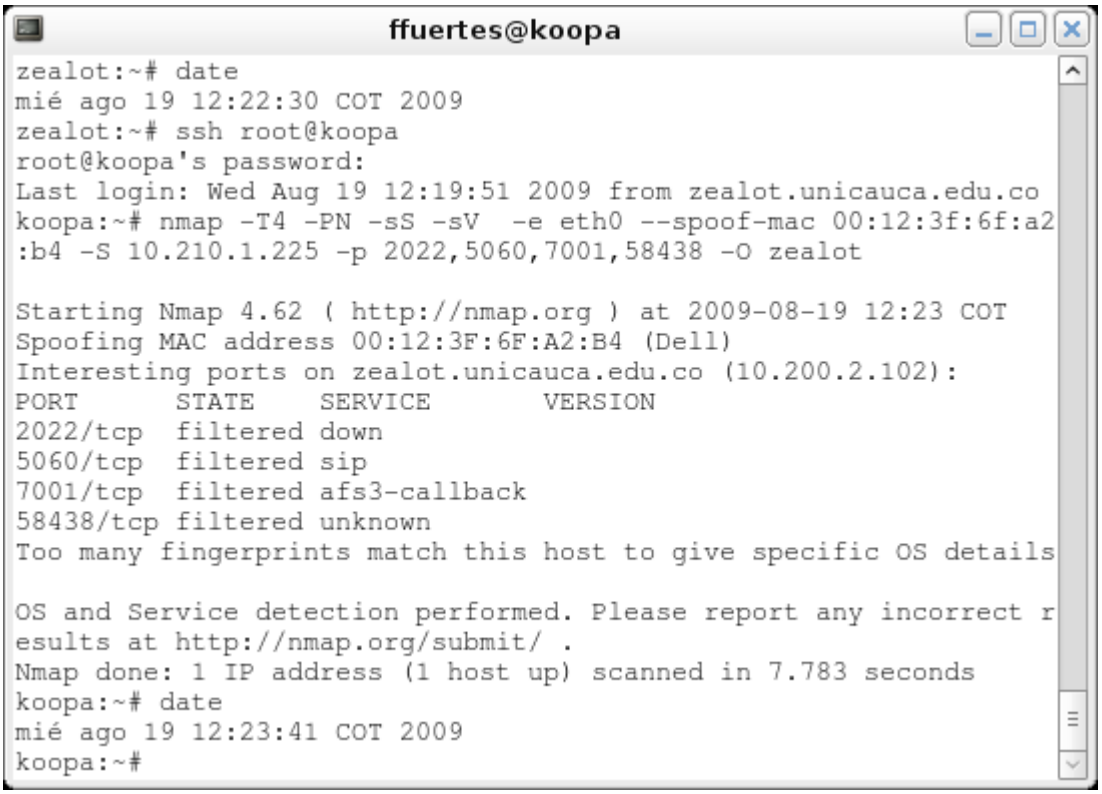

Figura 16. Exploración avanzada de puertos con intento de *spoof* en el host *zealot*

El siguiente sondeo mostrado en la Figura 16. , incluye opciones para falsear o realizar *spoof* tanto de lP (opción -S) como de dirección MAC (opción --spoof-mac) del equipo

fenix (IP: 10.210.1.225 MAC: 00:12:3f:6f:a2:b4), pues este tiene privilegios de acceso hacia los servicios SSH y HTTP en *zealot*. El escaneo como se puede observar no tiene éxito, ya que entre *fenix* y *zealot* existe un túnel IPsec el cual exige que todo el tráfico entre estos dos host pase a través de él, impidiendo este tipo de ataques.

Finalmente las Figuras 17, 18 y 19 muestran los escaneos hacia los host *fenix, xeratul* y *dragoon* respectivamente. Se aprecia que se intenta verificar el estado de los puertos que normalmente se encuentran abiertos en estos equipos después de habilitar el firewall local, dando como resultado que se encuentran en estado "filtrado", lo cual demuestra que el objetivo es inalcanzable, ya que si la respuesta fuera "cerrado" o "abierto", se confirmaría que se puede llegar al puerto aunque este no tenga un servicio en escucha, sin embargo el estado filtrado no logra determinar esta característica. De la misma manera, el intento de establecer las versiones de servicios y sistema operativo es infructuoso.

Figura 17. Exploración avanzada de puertos abiertos en el host *fenix*

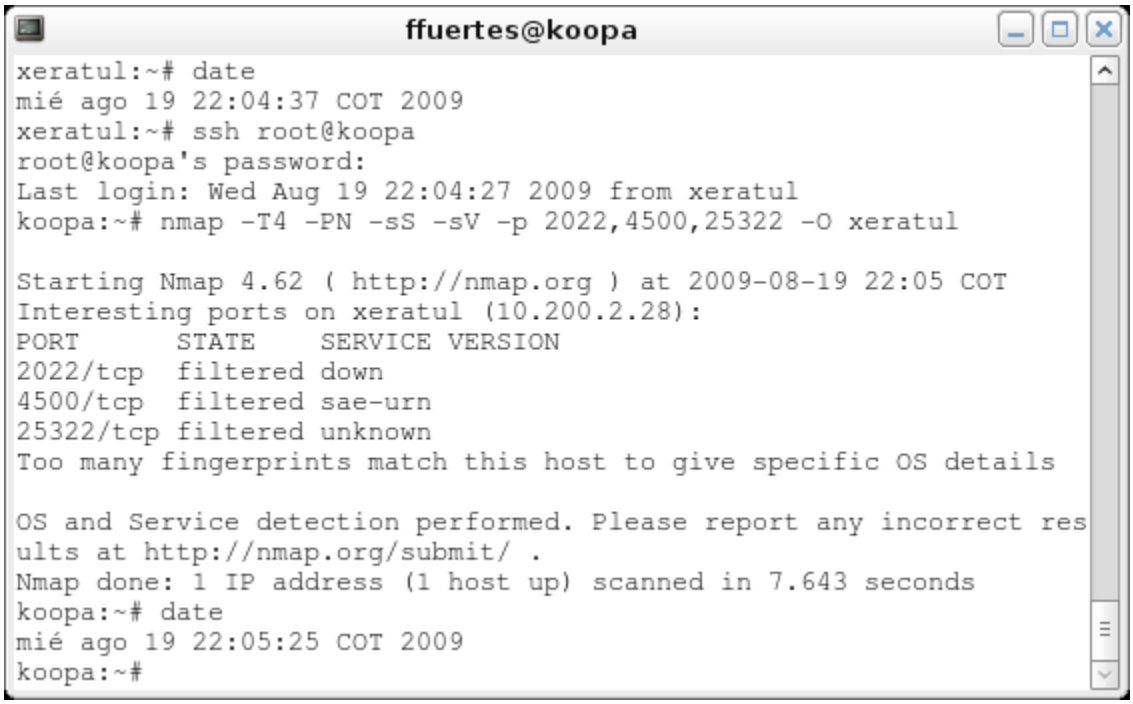

Figura 18. Exploración avanzada de puertos abiertos en el host *xeratul*

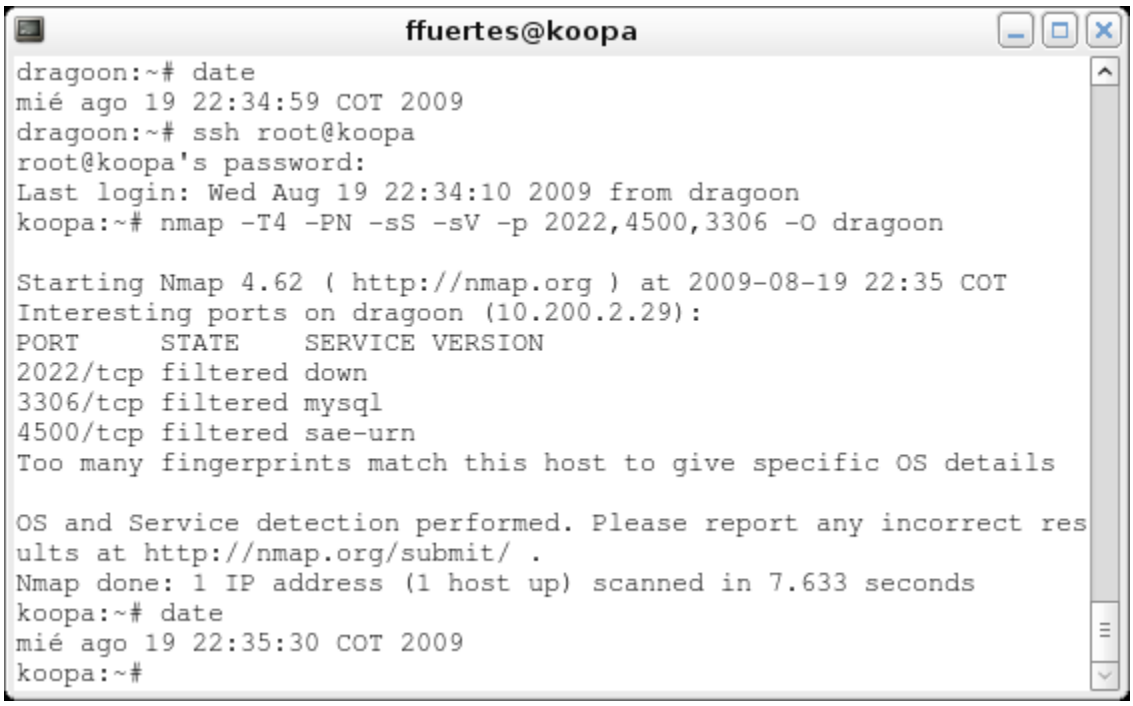

Figura 19. Exploración avanzada de puertos abiertos en el host *dragoon*

4. Prueba de Acceso

Objetivo: Revisar los métodos de autenticación en los equipos, tanto de la administración del sistema como de servicios particulares, entre estos controles se encuentra la verificación de cuentas por defecto y escalada de privilegios.

Procedimiento:

- Verificar usuarios del sistema

La primera prueba consiste en determinar cuáles son los usuarios "privilegiados" en cada equipo, es decir aquellos que poseen una consola para ejecutar comandos de sistema, de los cuales por seguridad debe haber la menor cantidad posible. En sistemas operativos basados en Unix, cada usuario tiene asignada una *Shell* o intérprete de línea de comandos, representada por el archivo ejecutable de la misma, en el caso de Debian es /bin/bash y la descripción de cada usuario se encuentra en el archivo /etc/passwd.

En las Figuras 20 a 23 se pueden observar los usuarios con Shell tipo /bin/bash en los host del prototipo, mostrando que en cada uno de ellos aparte del *root,* solo existe un usuario privilegiado, el cual es el único con acceso al equipo mediante SSH desde ubicaciones diferentes a *localhost,* como se verá en la próxima prueba.

Figura 20. Usuarios con consola privilegiada en *fenix*

Figura 21. Usuarios con consola privilegiada en *dragoon*

Figura 22. Usuarios con consola privilegiada en *zealot*

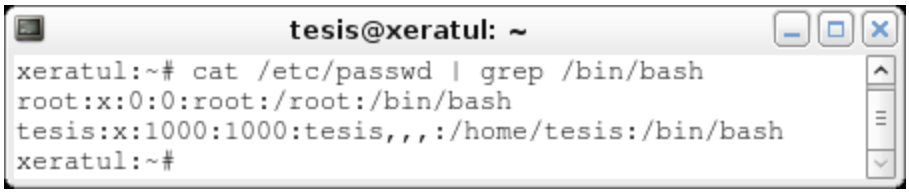

Figura 23. Usuarios con consola privilegiada en *xeratul*

- Verificar autenticación por defecto

En esta prueba se examina que la autenticación de los usuarios del sistema en cada equipo tenga opciones distintas a la instalación del servicio SSH estándar, ya que pueden convertirse en focos de posibles ataques. Como se describió en 4.3.1 el acceso por SSH a los equipos se protege cambiando las opciones por defecto del archivo de configuración del servidor SSH y autorizando solamente a una dirección IP para tal acceso (10.210.1.225), mediante reglas con TCP Wrappers e Iptables.

Como se vio en la prueba de visibilidad, ningún equipo que no sea permitido por los mecanismos de seguridad puede alcanzar a los hosts de la red prototipo, por lo tanto se prueba intentando acceder desde *fenix (*único equipo autorizado para iniciar estas conexiones) hacia *xeratul*, como se puede apreciar en la Figura 24. .

Figura 24. Acceso SSH desde *fenix* hacia *xeratul*

La gráfica anterior muestra varios elementos de seguridad; en primer lugar se observa que en el intento de acceder al puerto por defecto de SSH (22) se niega la conexión, ya que se han modificado las opciones del demonio para que escuche en el puerto 2022. Después, al intentar acceder como *root* en ese puerto, el resultado es un "permiso denegado", puesto que en la configuración del servicio niega el acceso a este usuario desde otra ubicación que no sea *localhost.* Finalmente se realiza la conexión con el

usuario *tesis* y desde allí es posible subir de privilegios de manera normal mediante el comando `su`.

- Verificar fortaleza de las contraseñas de usuario

En esta prueba se intenta descifrar las contraseñas de usuario en las maquinas objetivo, para esto se utiliza la herramienta John the Ripper en el modo incremental, el cual realiza un ataque de fuerza bruta combinando letras, números y caracteres especiales para probarlos como posibles contraseñas en un archivo que contenga estos cifrados.

Como ejemplo, la Figura 25. muestra que se han obtenido los archivos passwd y shadow, (en los cuales se guardan usuarios y contraseñas en sistemas Unix) del host *zealot* estos son combinados mediante el comando `unshadow` en el archivo "zealot.users", a este se le realiza el ataque de fuerza bruta sin obtener resultado, ya que fue detenido después de casi 17 horas sin descifrar ninguna contraseña.

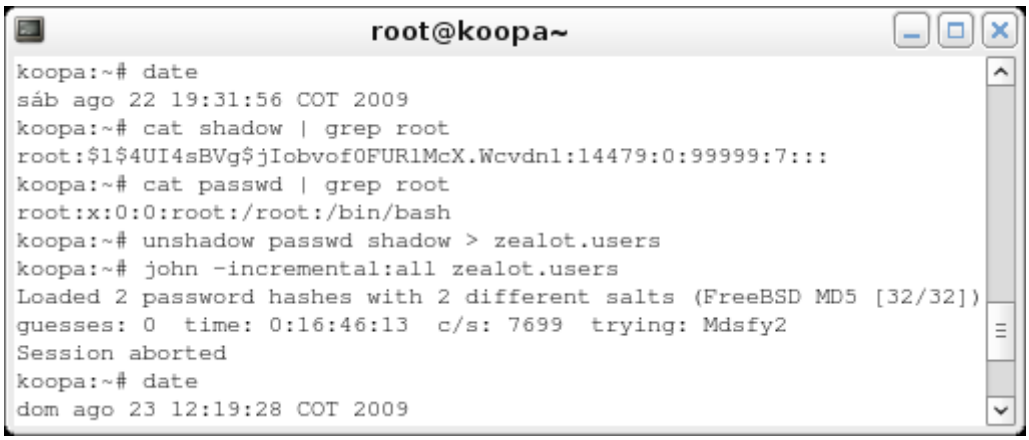

Figura 25. Ataque de fuerza bruta con John the Ripper

Lo anterior se debe a que tanto para los usuarios del sistema como para las aplicaciones como MySQL o Sequoia se han utilizado contraseñas fuertes de no menos de 12 caracteres que combinan letras, números y caracteres especiales, las cuales tienen muy baja probabilidad de ser encontradas bien sea por el método anterior como por el método diccionario el cual compara la contraseña cifrada con una base de datos de contraseñas pre establecidas.

De la misma manera archivos donde se guardan contraseñas o claves de aplicaciones como /etc/shadow, /etc/ipsec.secrets se han asegurado mediante permisos restrictivos para que solamente el usuario *root* pueda leerlos.

- Verificar seguridad del kernel

En la Figura 26. se muestra el resultado de correr la herramienta Paxtest para detectar vulnerabilidades en un kernel estándar. Durante la prueba, el host *fenix* cuenta con el kernel instalado con la distribución Debian 5 (2.6.26-1). Es claro que este es vulnerable a varios tipos de ataques que implican ejecutar código malicioso en la máquina, los cuales podrían conceder privilegios al intruso o bloquear el equipo.

La Figura 27. muestra el resultado de correr Paxtest en el host *zealot*, en el cual previamente se había compilado e instalado un nuevo (2.6.27.10grsec-1.0) habilitando diversas opciones de seguridad. Como puede verse, el mensaje *Killed* significa que el sistema está protegido contra un intento de explotar una vulnerabilidad determinada.

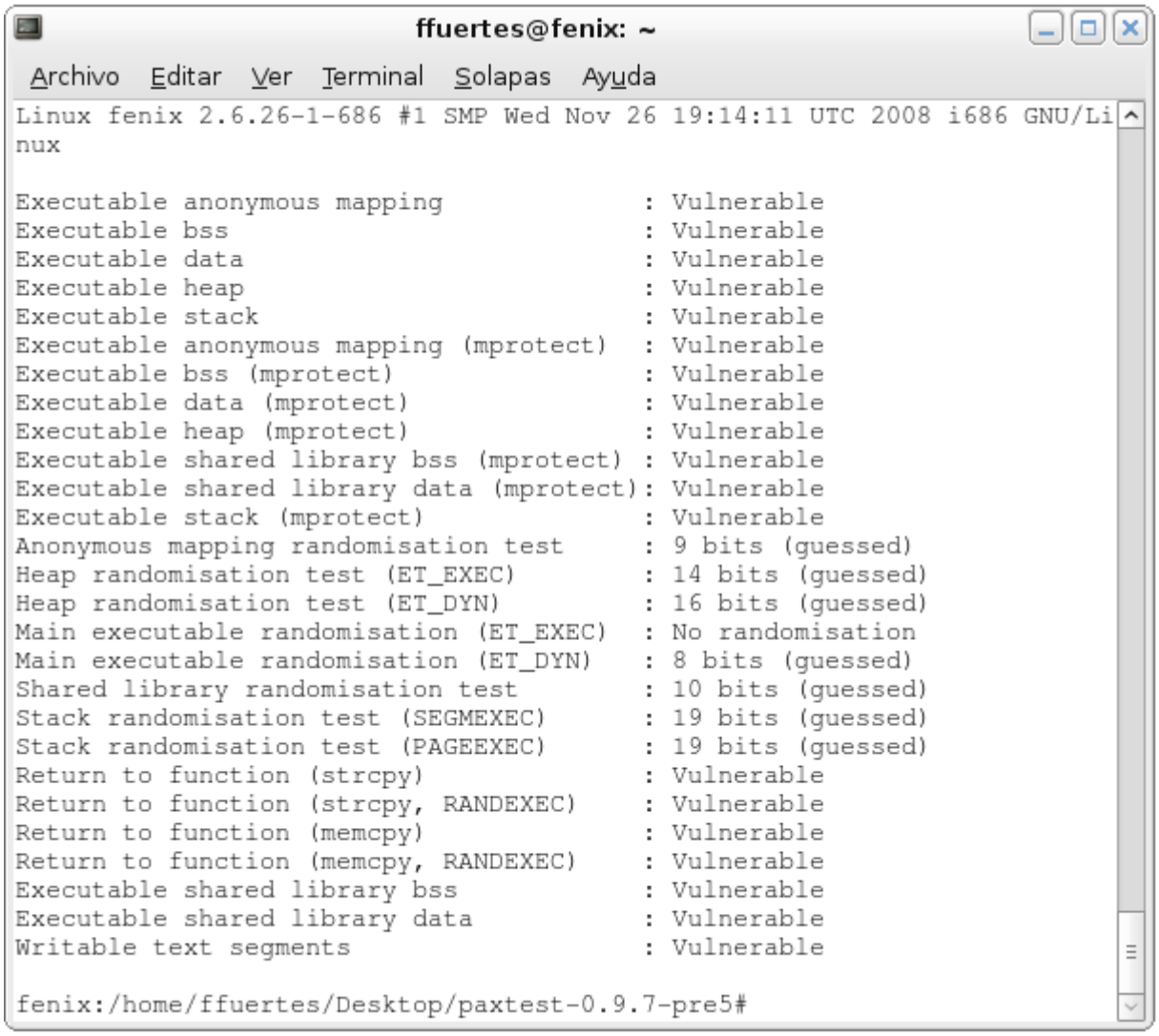

Figura 26. Vulnerabilidades del 2.6.26 Estándar

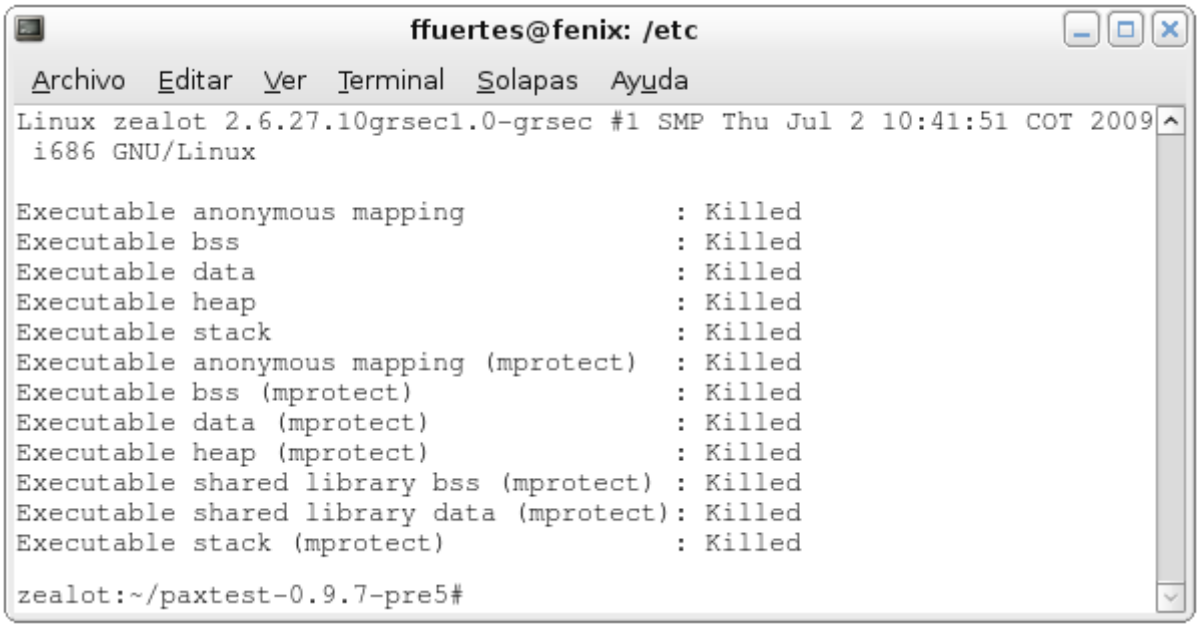

Figura 27. Kernel 2.6.27 compilado para optimizar la seguridad

5. Prueba de Confianza

Objetivo: Probar la confianza entre equipos dentro de la red, referente al acceso a recursos del sistema, servicios o información sin necesidad de identificación o autenticación. Para este tipo de pruebas se verifica si es posible realizar *spoofing* o falseo tanto de direcciones IP como MAC, desde un equipo sin privilegios en la red.

Procedimiento

Envenenamiento ARP

Esta prueba consiste en verificar si es posible realizar ataques tipo *hombre en el medio* valiéndose del método de envenenamiento ARP, en el cual el atacante envía mensajes *ARP replay* falsos hacia dos equipos que necesiten establecer conexión en un mismo segmento de red, con el fin de que estos se comuniquen sin saberlo a través del equipo atacante, así este puede espiar y/o modificar la comunicación.

En primer lugar se probará si se tiene éxito con este ataque sin activar el mecanismo de seguridad entre los equipos *dragoon* y *xeratul.*

En las Figuras 28 y 29 se muestran las entradas normales en la tabla caché ARP de estos equipos. Después de realizar el ataque el valor de la dirección MAC de estas entradas debe ser reemplazado por la MAC del atacante con el fin de desviar el tráfico entre las víctimas.

Figura 28. Entrada para el host *xeratul* en la tabla ARP de *dragoon*

Figura 29. Entrada para el host *dragoon* en la tabla ARP de *xeratul*

Ahora, se lanza el ataque desde el equipo auditor utilizando la herramienta Ettercap como se muestra en la Figura 30. , allí puede verse que la dirección MAC de este host es 00:12:3F:6F:9E:60, la cual se envía repetidamente en mensajes *ARP replay* a las víctimas, anunciando que tanto la dirección IP de *dragoon* como la de *xeratul* se asocian a esta dirección MAC.

De este modo, las entradas de la tabla ARP en cada equipo mostradas en las Figuras 28 y 29 son reemplazadas como se puede apreciar en las Figuras 31 y 32. Para probar el buen funcionamiento del ataque se envía un ping desde *dragoon* hacia *xeratul,* el cual llega sin problemas y puede ser monitoreado por el auditor como se muestra en la Figura 33. , en donde se utiliza la herramienta Tcpdump para capturar todo el tráfico entre las víctimas.

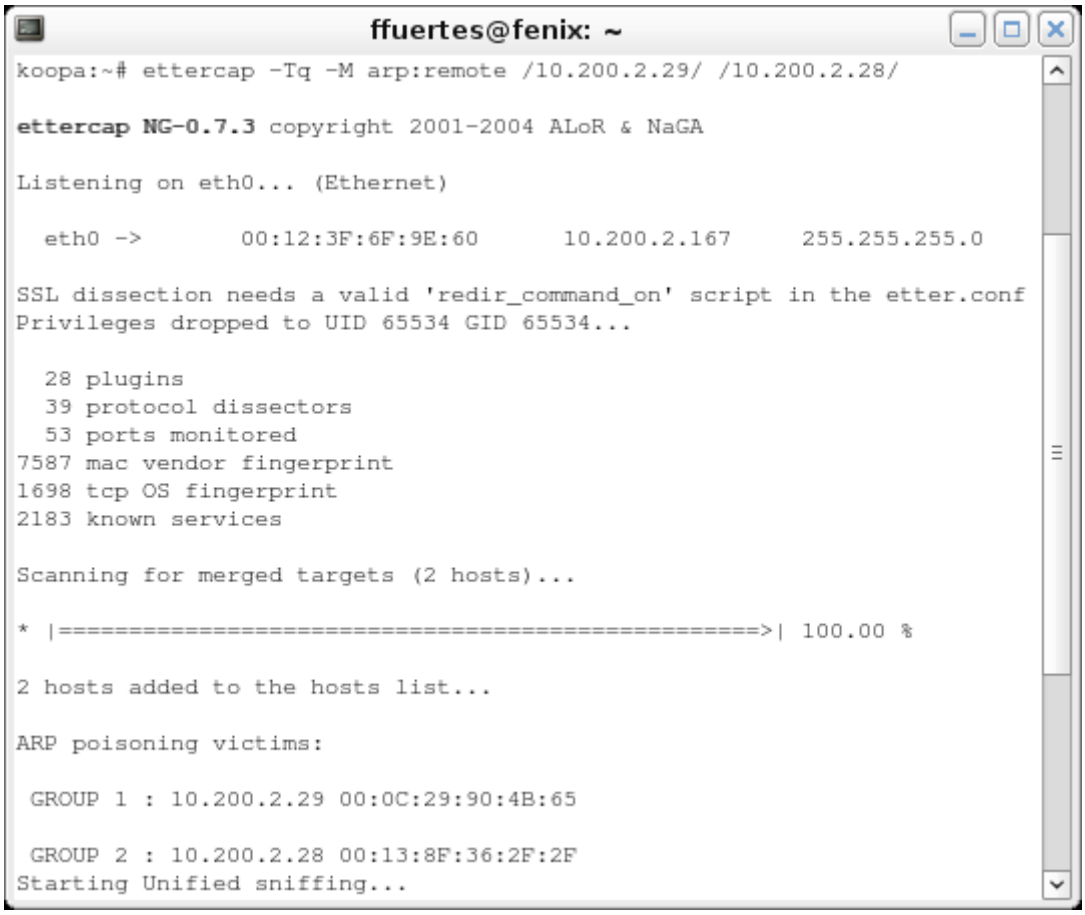

Figura 30. Lanzamiento del ataque de envenenamiento ARP utilizando Ettercap

Figura 31. Entrada para el host *dragoon* en la tabla ARP de *xeratul* después del ataque

Figura 32. Entrada para el host *xeratul* en la tabla ARP de *dragoon* después del ataque

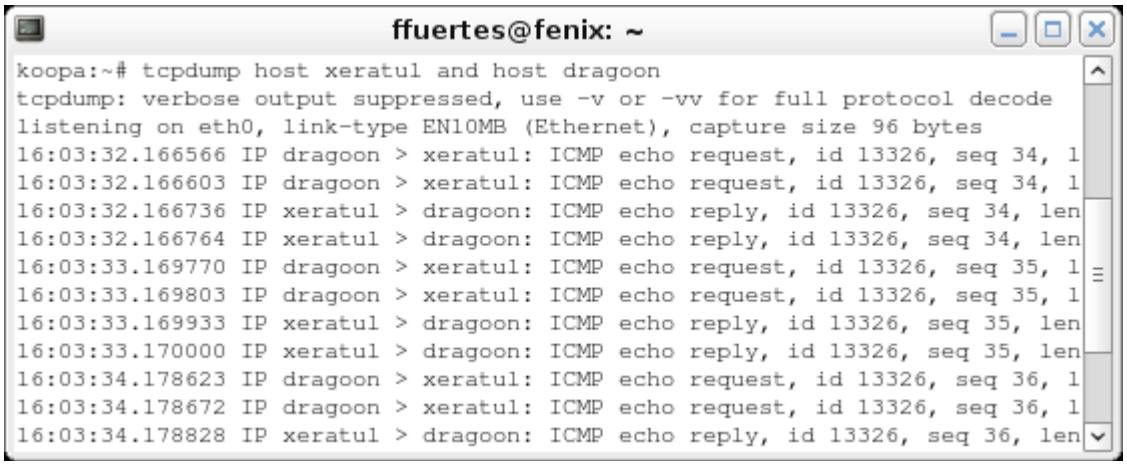

Figura 33. Captura de los mensajes ping (ICMP echo) entre *xeratul* y *dragoon*

Ahora, se activa el firewall en los dos equipos a prueba (ver Figuras 34 y 35) y se intenta realizar el ataque de la misma manera que se mostró en la Figura 30.

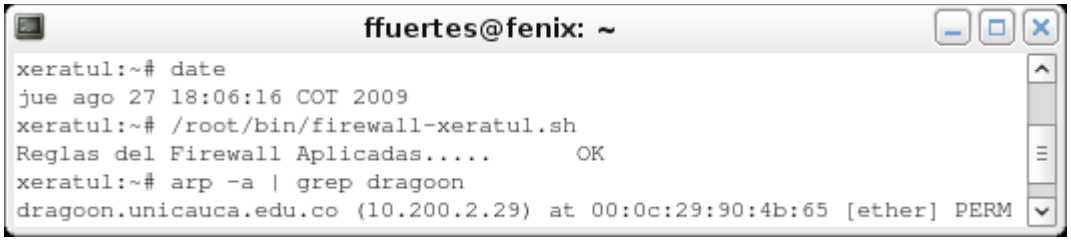

Figura 34. Activación del firewall local en *xeratul* y comprobación de la entrada ARP para *dragoon*

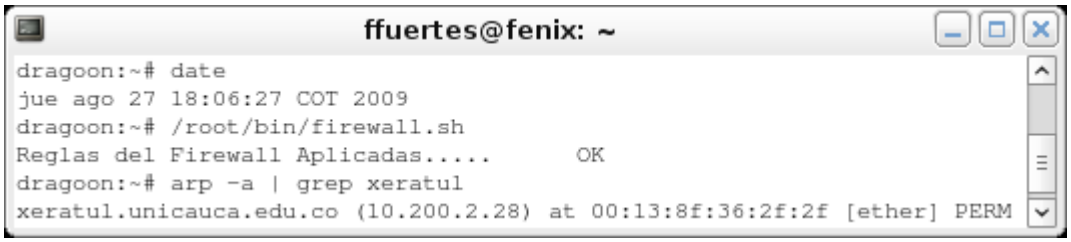

Figura 35. Activación del firewall local en *dragoon* y comprobación de la entrada ARP para *xeratul*

Como se observa en las dos graficas anteriores y como se describió en 4.3.2, los firewalls locales incluyen entradas ARP estáticas para los equipos de confianza de la red, por lo tanto estas no pueden ser cambiadas mediante mensajes *ARP replay*. Con esto, el ataque por envenenamiento deja de funcionar, como se puede corroborar en las Figuras 36 y 37.

Por ejemplo, En la Figura 36. se aprecia que al utilizar la herramienta tcpdump en *xeratul* para monitorear tráfico proveniente *dragoon,* están llegando mensajes *ARP replay* anunciando que los dos hosts tienen la misma dirección MAC (dirección perteneciente al equipo atacante) pero al comprobar si esto "envenena" el caché, mediante el comando `arp -a`, se puede ver que la dirección estática configurada por el firewall no se cambia, por lo tanto el ataque no surte ningún efecto. El mismo resultado se muestra en la Figura

37. para el host *dragoon*.

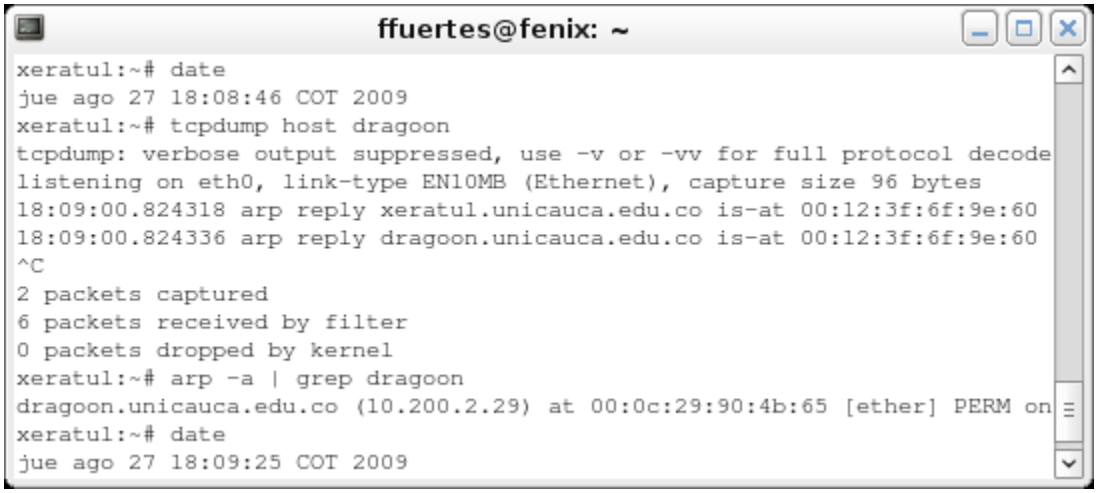

Figura 36. Captura de los mensajes *ARP replay* que envía el atacante y verificación de la entrada estática en la tabla cache ARP en *xeratul*

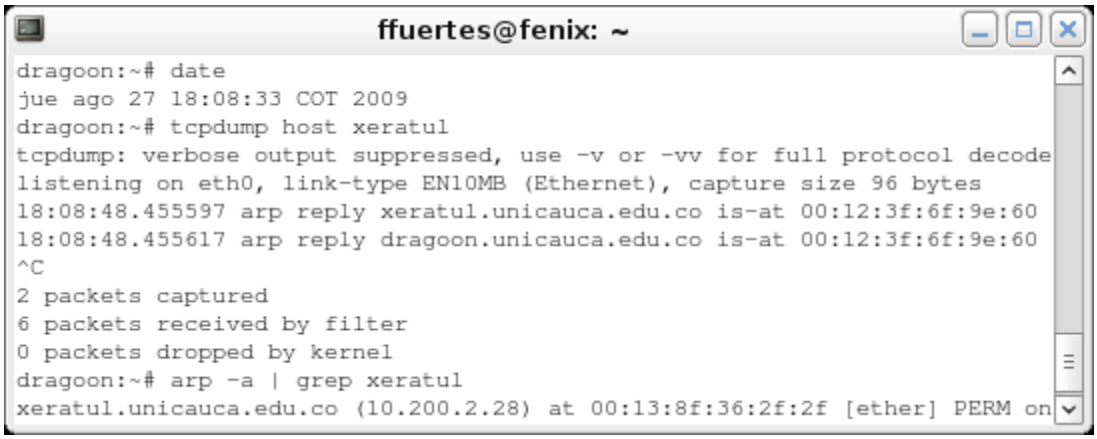

Figura 37. Captura de los mensajes *ARP replay* que envía el atacante y verificación de la entrada estática en la tabla cache ARP en *dragoon*

Todo lo anterior se complementa con las pruebas de visibilidad realizadas en la sección 3 de este documento, en donde se utilizó la herramienta Nmap para sondear puertos sobre el host *zealot,* mediante opciones de *spoof* tanto de IP como de MAC sin tener ningún éxito, gracias a que se encontraba activo el mecanismo de seguridad el cual impide este tipo de ataques (ver Figura 16.).

6. Prueba de No Repudio

Objetivo: Probar si los elementos de seguridad del prototipo propuesto guardan registros con la identificación apropiada de los sistemas o personas que han interactuado con cada equipo.

Procedimiento

En estas pruebas se revisará si la aplicación IPsec y el firewall configurados en cada host están guardando *logs* donde se registren la ubicación y el momento de acceso al equipo.

- Logs de strongSwan

Para esta prueba se verifica que la herramienta strongSwan registre los establecimientos de Asociaciones de Seguridad con otros equipos. Para esto, se toma como ejemplo del establecimiento y la terminación un túnel IPsec entre los host *fenix* y *zealot*, en donde *fenix* es quien inicia y finaliza la conexión.

En primer lugar, en *fenix* se realiza la petición para el establecimiento del túnel con *zealot* mediante el comando: `ipsec up zealot`, como lo muestra La Figura 38. .

Figura 38. Petición de establecimiento de un túnel IPsec desde *fenix* hacia *zealot*

En ese momento, el demonio *charon* encargado de establecer las Asociaciones de Seguridad IPsec, también registra este inicio de sesión en el archivo de log /var/log/daemon.log, tanto en el equipo que envía la petición como en el que la recibe, describiendo detalladamente las acciones realizadas con fecha, hora, y direcciones IP implicadas en el proceso, como lo muestran las Figuras 39 y 40.

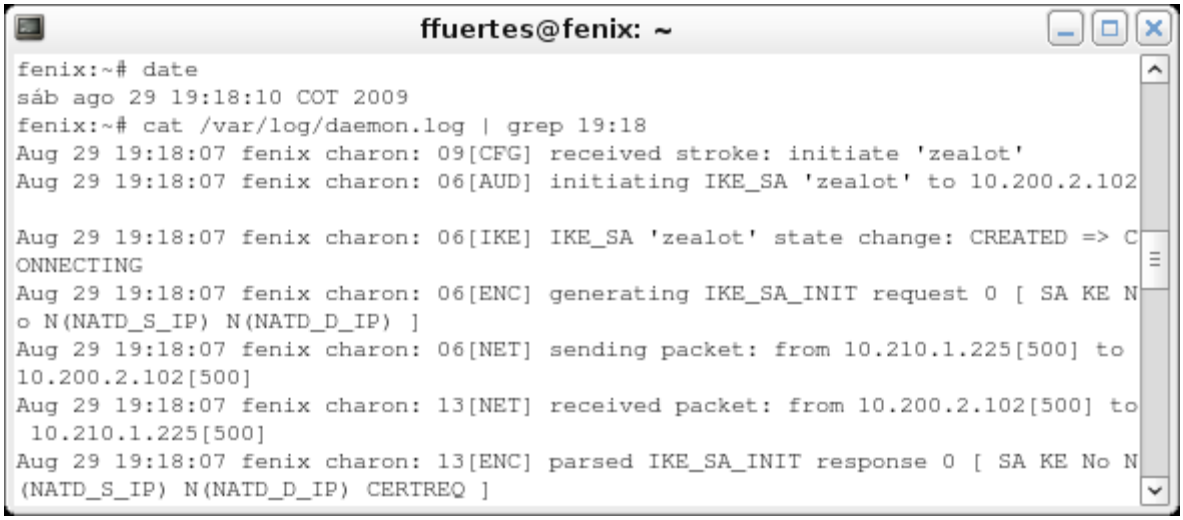

Figura 39. Revisión del log daemon.log en *fenix*, donde se encuentra el registro del establecimiento de la conexión IPsec con *zealot*

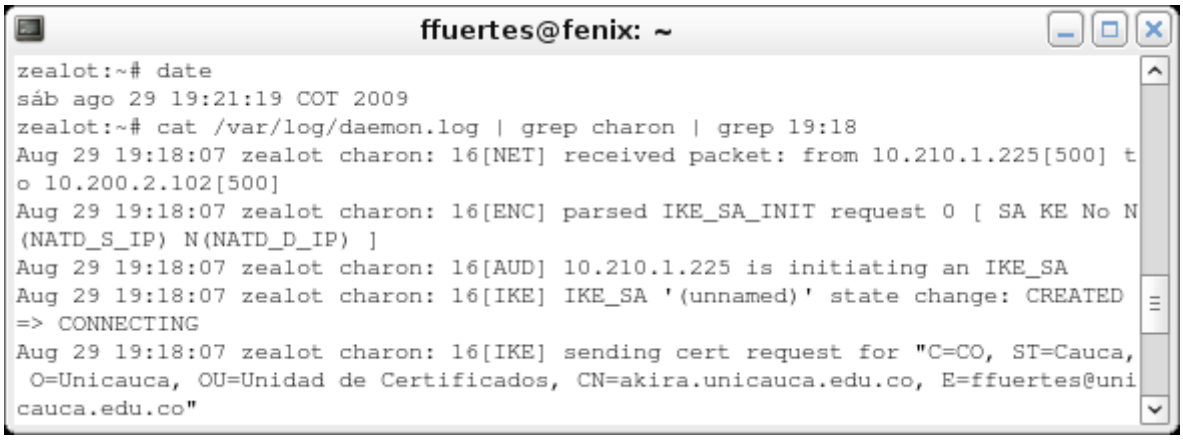

Figura 40. Revisión del log daemon.log en *zealot*, donde se encuentra el registro del establecimiento de la conexión IPsec hecho desde *fenix*

Ahora, desde *fenix* se envía una petición para terminar la sesión IPsec con el comando: `ipsec down zealot` (ver Figura 41.).

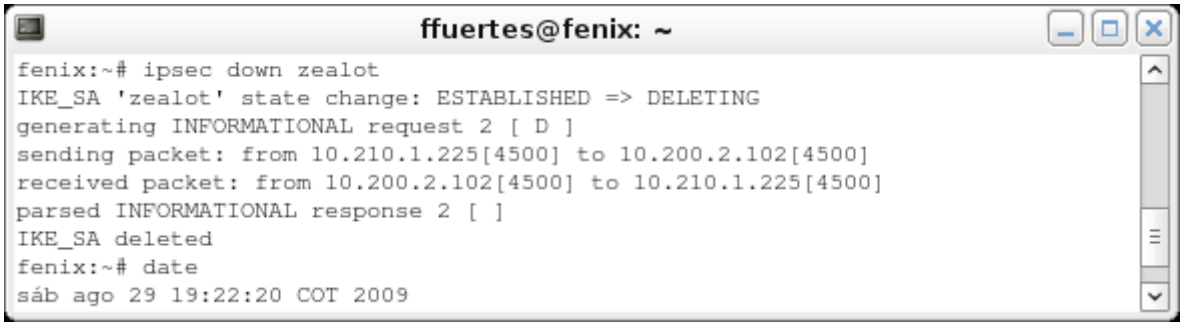

Figura 41. Finalización del túnel IPsec en *fenix*

La acción anterior también es registrada en los logs de ambos equipos, como lo muestran las Figuras 42 y 43.

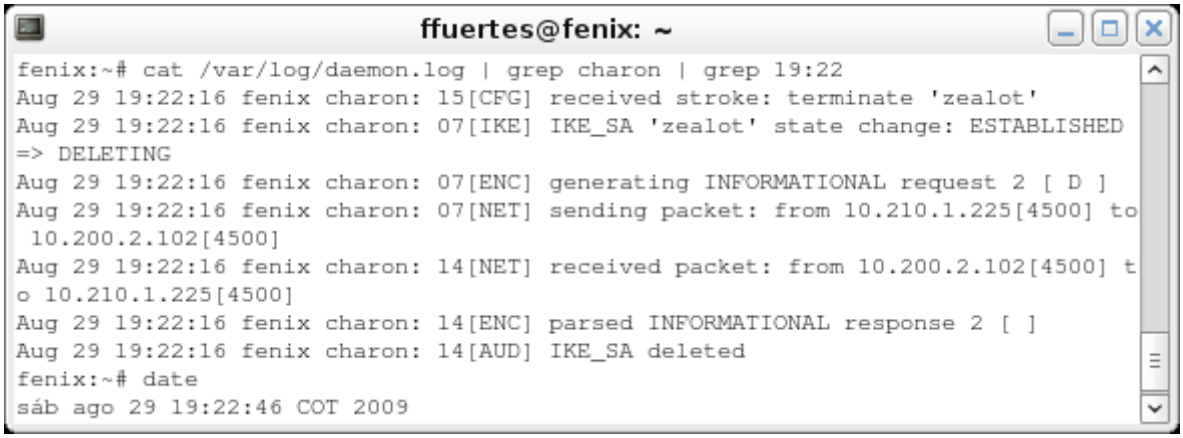

Figura 42. Revisión del log daemon.log en *fenix*, donde se encuentra el registro de la terminación de la conexión IPsec con *zealot*

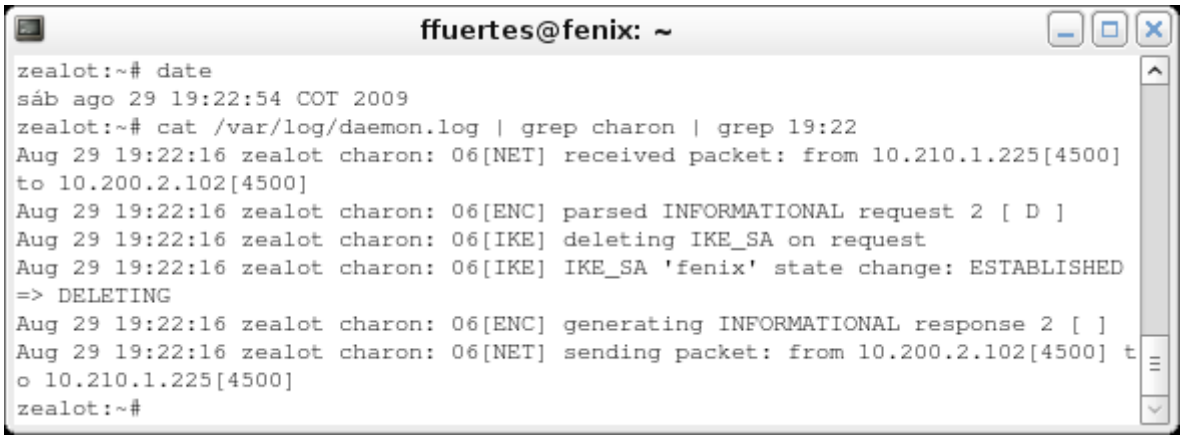

Figura 43. Revisión del log daemon.log en *zealot*, donde se encuentra el registro de la terminación de la conexión IPsec hecho desde *fenix*

- Logs de Iptables

Como se describió en 4.3.2, la configuración del firewall en los equipos incluye la característica de guardar logs, tanto de las conexiones establecidas (aceptadas por el firewall), como de los intentos de conexión no permitidos (rechazados por el firewall).

Para comprobar lo anterior, se toman como ejemplo dos acciones rutinarias: iniciar una conexión SSH y realizar un ping; nuevamente se prueba con los host *fenix* y *zealot*, siendo el primero quien inicia cada interacción. Luego se observa si estas conexiones son registradas en los logs de ambos equipos.

Como primer ejemplo, desde *fenix* se envían 3 mensajes ping (mensajes ICMP de petición de echo) hacia *zealot* (Ver Figura 44.).

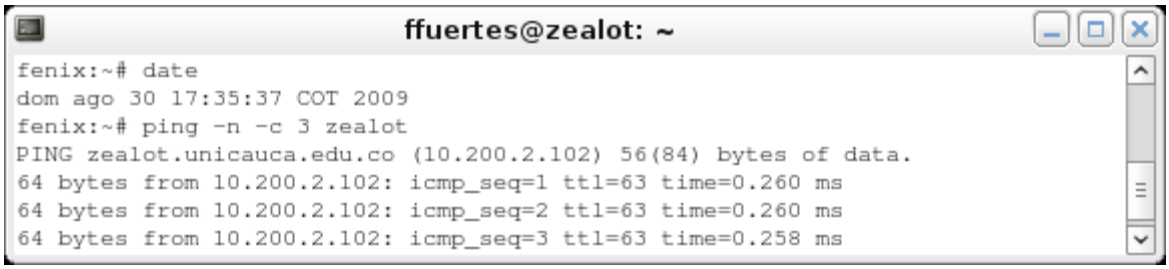

Figura 44. Envío de mensajes ping desde *fenix* hacia *zealot*

Ahora, revisando los logs de Iptables en *zealot* (ver Figura 45.), se encuentra el registro de los mensajes ICMP tipo 8 (petición de echo) provenientes de la dirección IP de *fenix* (10.210.1.225), los cuales aparecen con fecha, hora de envío, direcciones MAC, entre otros datos y con el rotulo de “IMSeg_ACEPTADO”, lo cual confirma que el recibió y dejo pasar este mensaje.

Figura 45. Revisión del log de Iptables en *zealot* donde se registra el ping proveniente de *fenix*

Para el segundo ejemplo, como se muestra en la Figura 46. , se inicia una sesión SSH en *zealot*. Posteriormente, se revisa el log de Iptables encontrando este acceso como "aceptado" (ver Figura 47.).

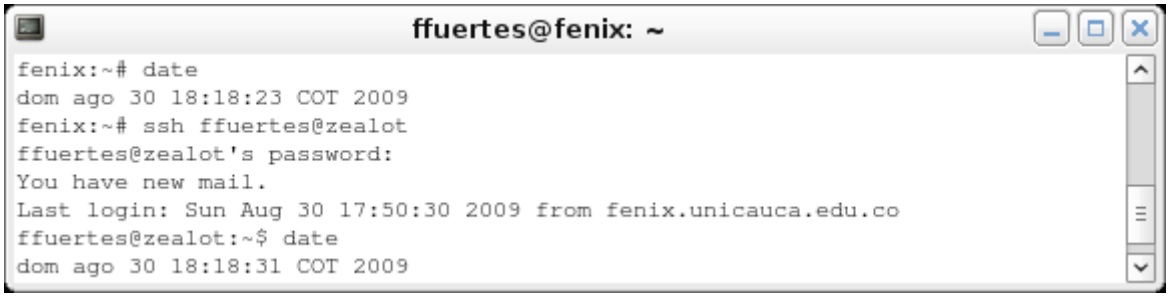

Figura 46. Inicio de sesión SSH desde *fenix* hacia *zealot*

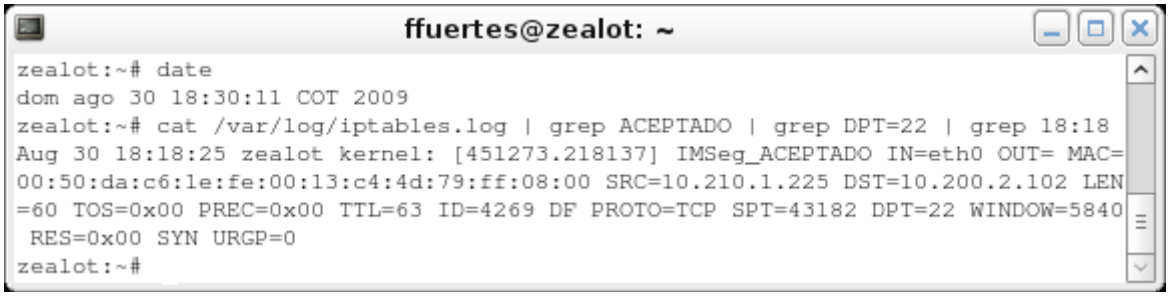

Figura 47. Revisión del log de Iptables en *zealot* donde se registra el acceso SSH proveniente de *fenix*

- Logs de TCP Wrappers

En la sección 4.3.1 se mostro la configuración de TCP Wrappers para añadir una capa más de protección a la administración remota por SSH en los host del prototipo, igualmente se configuró esta herramienta para que registre en el log `ssh.log`, todos los accesos permitidos. Tomando el ejemplo de la Figura 45, se puede apreciar que este inicio de sesión se guarda como "acceso SSH satisfactorio" en el log `ssh.log` (ver Figura 48.).

Figura 48. Revisión del log ssh.log en *zealot* donde se registra el acceso SSH proveniente de *fenix*

Todo lo anterior se complementa con el hecho de que aplicaciones como MySQL ó Sequoia guardan registro de los eventos ocurridos durante su funcionamiento normal en logs particulares, lo cual incrementa la característica de No Repudio en el prototipo de seguridad propuesto.

7. Prueba de Confidencialidad e Integridad:

Objetivo: Verificar el correcto funcionamiento de IPsec para proteger la confidencialidad e integridad de la información en un canal de comunicación entre dos hosts.

Procedimiento

En estas pruebas se intentará redirigir y espiar la comunicación entre dos equipos del prototipo, con el fin de vulnerar los servicios de confidencialidad e integridad, los cuales son los principales requerimientos de seguridad en IMS.

Como se ha visto en pruebas anteriores, (ver secciones 3 y 5), el mecanismo de seguridad impide espiar la comunicación entre dos host del prototipo, lo cual garantiza que no se redirijan los paquetes de datos, siendo el primer escudo para proteger la confidencialidad e integridad de la información, por lo tanto para realizar estas pruebas se deshabilita el firewall con lo cual es posible realizar un ataque de envenenamiento ARP desde el equipo auditor como se mostró en la sección 5 hacia los objetivos de la prueba, que en esta oportunidad son los host *xeratul* y *dragoon* (ver Figura 49.).

Figura 49. Envenenamiento ARP desde koopa hacia *dragoon* y *xeratul*

Como primera prueba, también se deshabilita el servicio IPsec, para corroborar que sin él, es posible vulnerar la confidencialidad en el intercambio de datos.

Ahora, ya que la interacción entre *xeratul* y *dragoon* siempre será para que el primero realice consultas MySQL sobre el segundo, en esta prueba se realiza un inicio de sesión por consola al servidor MySQL en *dragoon* y posteriormente se hace una consulta puntual tipo SELECT sobre la tabla *impi* de la base de datos *user_data*, al igual que lo haría cualquier aplicación, como lo muestra la Figura 50. .

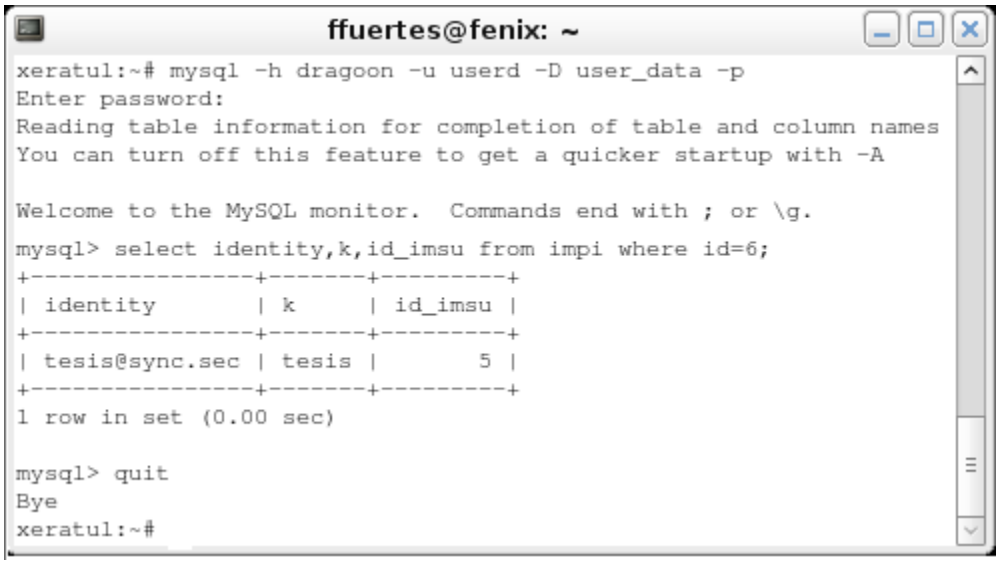

Figura 50. Inicio de sesión MySQL y consulta SQL SELECT desde *xeratul* a *dragoon*

Las anteriores transacciones, son capturadas por la herramienta Wireshark corriendo en el host auditor. Los datos del inicio de sesión se muestran en la Figura 51. resaltados en los recuadros amarillo y rojo, mientras que en la Figura 52. , se aprecia cómo, utilizando la opción *Follow TCP Stream* de Wireshark, se puede observar claramente la respuesta del servidor a la consulta SELECT hecha previamente.

Con esta prueba, queda en evidencia que los servicios de confidencialidad e integridad se ven seriamente comprometidos, ya que con este tipo de técnicas es posible capturar y modificar cualquier información que se transmita en texto plano, como una consulta MySQL o HTTP. Incluso, es posible utilizar herramientas de ataque tipo fuerza bruta para tratar de descifrar contraseñas como la mostrada en el recuadro rojo de la Figura 51. .

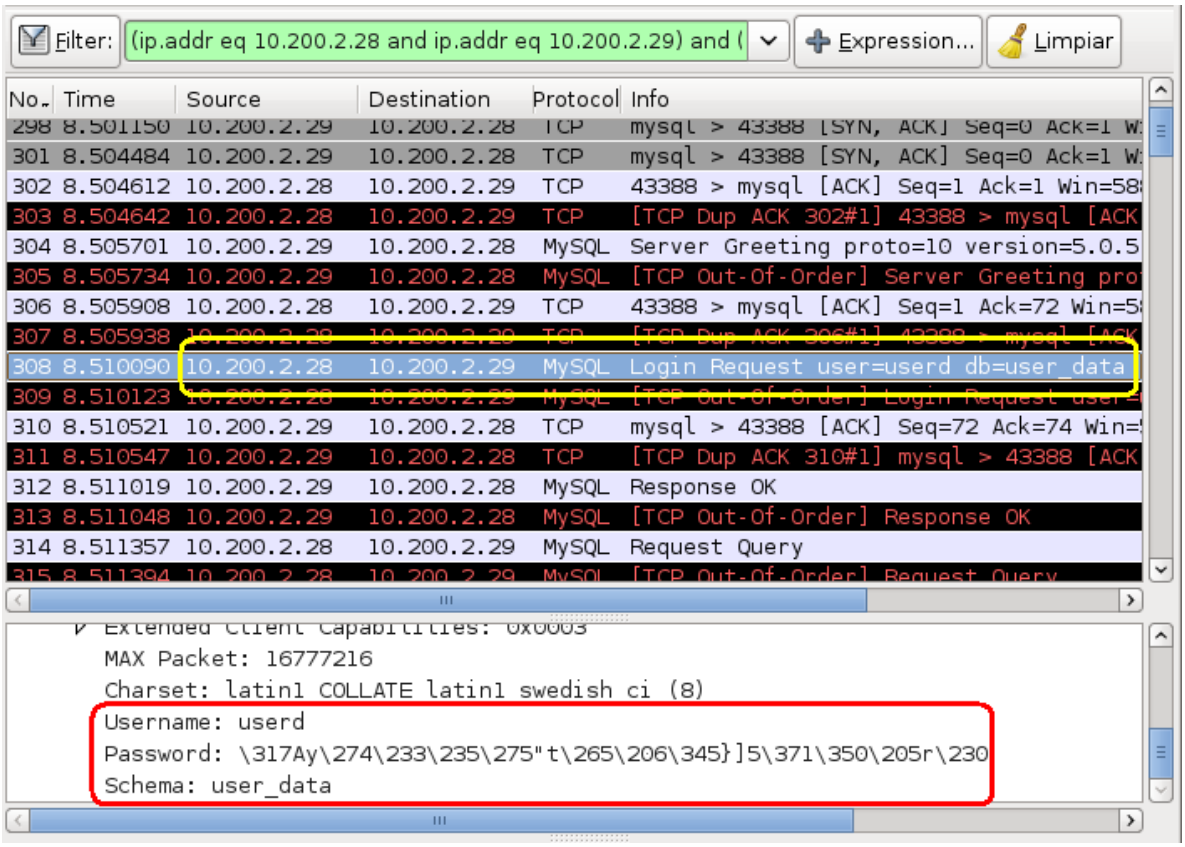

Figura 51. Captura del Inicio de sesión MySQL entre *xeratul* y *dragoon* usando Wireshark

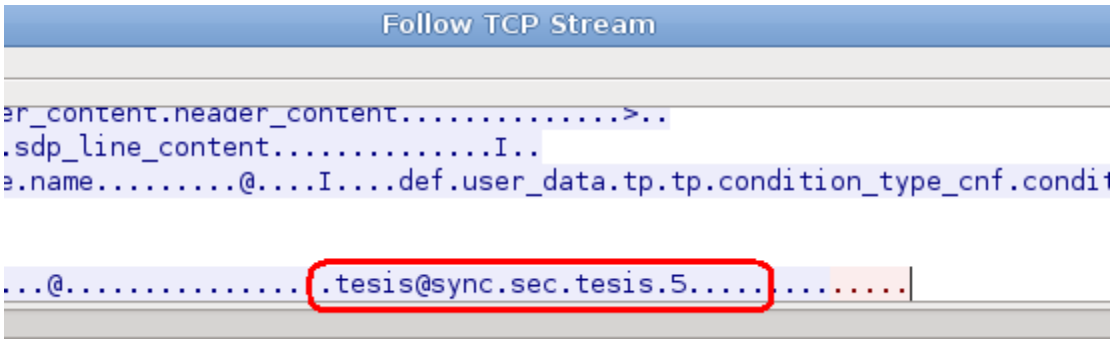

Figura 52. Captura del Flujo TCP donde se muestra la respuesta del servidor MySQL a la consulta hecha

Para la segunda prueba, se mantiene el ataque de envenenamiento ARP proveniente del equipo auditor para capturar el tráfico entre los host objetivo, pero se habilita el servicio IPsec y se activa el túnel entre ambos equipos. Posteriormente se realiza el mismo inicio de sesión y consulta MySQL mostrados en la Figura 50. .

La captura del trafico generado por estas acciones se muestra en la Figura 53. , allí se puede observar que solamente se despliegan mensajes cifrados con el protocolo ESP, perteneciente a la pila IPsec en los cuales se encapsulan los mensajes MySQL originales.

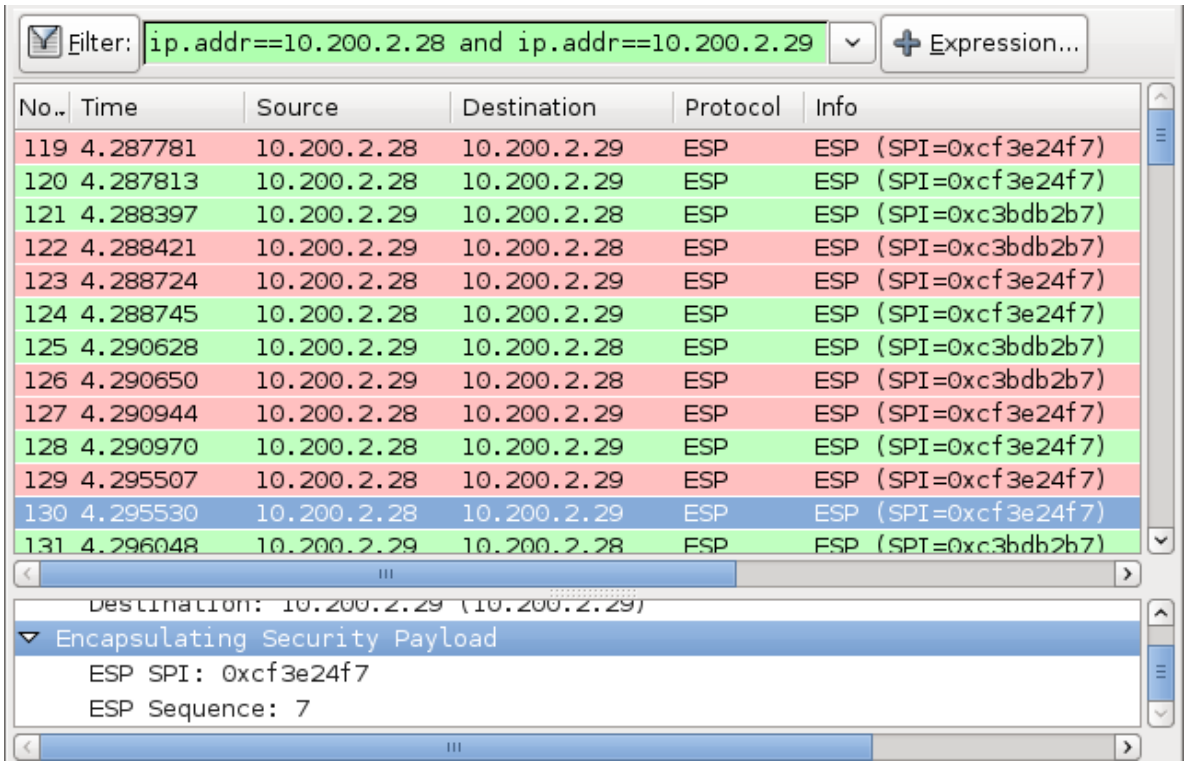

Figura 53. Captura de los mensajes cifrados con ESP, correspondientes al Inicio de sesión en MySQL y consulta SELECT entre *xeratul* y *dragoon*

En la actualidad no se han encontrado ataques efectivos contra este tipo de cifrado, pues tanto su fortaleza (AES_CBC-256), como el hecho de que los parámetros de autenticación cambian constantemente a través de Asociaciones de Seguridad IPsec, hacen muy difícil que un ataque de fuerza bruta surta efecto. Sin embargo, Wireshark ofrece una opción para descifrarlo, siempre y cuando se conozcan los algoritmos utilizados en IPsec, las direcciones entre los equipos que existe la AS y las llaves de cifrado y autenticación, lo cual para el caso del prototipo no aplica ya que no se usan llaves pre-compartidas sino certificados X.509.

Por lo tanto, así sea posible capturar este tipo de paquetes cifrados, el hecho de que no se pueda conocer el tipo de protocolo o aplicación que está encapsulada en ellos, hace que el ataque no vulnere ni la confidencialidad ni la integridad de la información que transportan, ya que no es posible tener acceso ni modificar su contenido. Con esto, se deduce que incluso sin la activación del firewall en el prototipo, los servicios de confidencialidad e integridad ofrecidos se ajustan a los requerimientos de IMS.

8. Prueba de Alertas y logs

Objetivo: Comprobar el correcto funcionamiento de sistemas de logs o alarmas que informen sobre situaciones sospechosas o actividades perjudiciales, además que su acceso se realice solo por usuarios privilegiados.

Procedimiento

Para estas verificaciones, se realizan intentos de conexiones no autorizadas en los equipos del prototipo y se revisa si los logs de aplicaciones como strongSwan, Iptables, TCP Wrappers y ARP Alert guardan estos registros.

- strongSwan

En esta prueba, se intenta establecer un túnel IPsec desde el host *dragoon* hacia *zealot* como se muestra en la Figura 54. , sin embargo según el diseño de la red para los laboratorios de pruebas (ver sección 5.2.5 de la monografía), estos equipos no tienen una conexión directa, por lo tanto se considera como un intento de acceso no autorizado.

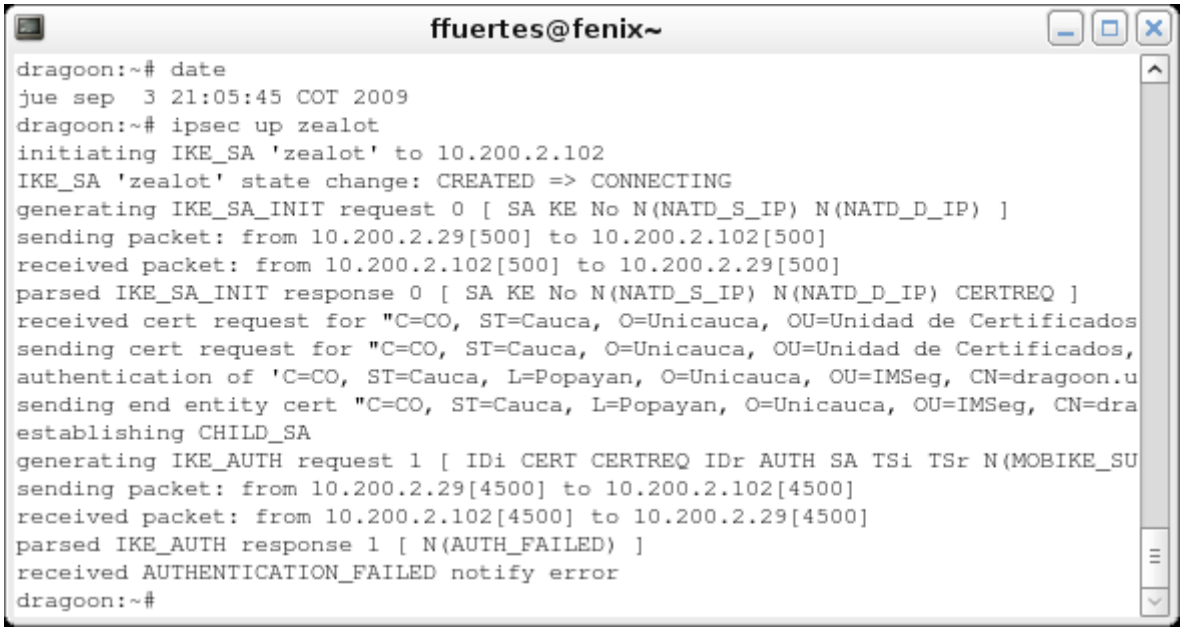

Figura 54. Intento de establecer túnel IPsec desde *dragoon* hacia *zealot*

Ahora, se revisa el log *daemon.log* donde se guardan entre otros, los eventos propios del demonio de IPsec: *charon.* En este, se encuentra el registro del intento de conexión con el estado AUTH_FAILED como se observa en la Figura 55. , lo cual comprueba el buen funcionamiento del log de IPsec.

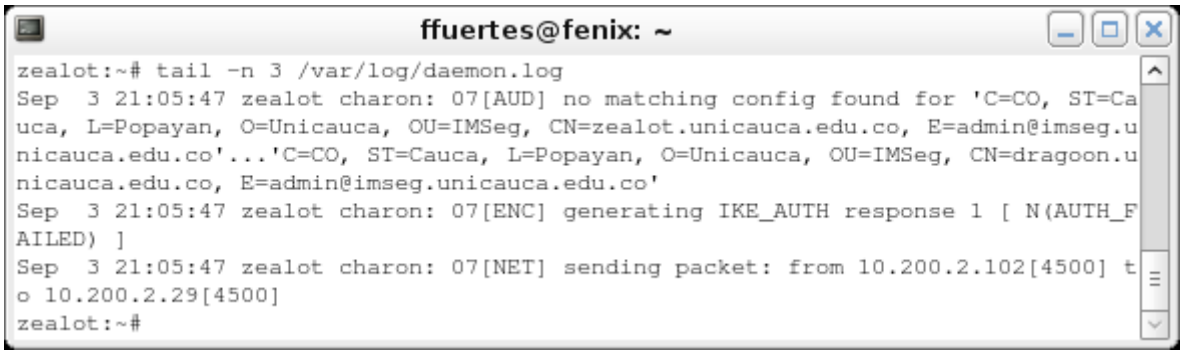

Figura 55. Revisión del log daemon.log en *zealot* donde se encuentra el intento de acceso fallido desde *dragoon*

- TCP Wrappers

En esta prueba se intenta realizar un acceso SSH no autorizado desde el host auditor hacia *xeratul* (ver Figura 56.) y posteriormente se revisa el log de TCP Wrappers para comprobar el registro de este evento como se observa en la Figura 57. .

Figura 56. Intento de acceso SSH no autorizado hacia *xeratul*

Figura 57. Revisión del log *ssh.log* en *xeratul* donde TCP Wrappers registra el intento de acceso fallido

- Iptables

Para esta prueba, se envía una petición de conexión hacia el servidor MySQL en *dragoon* desde el equipo auditor como se muestra en la Figura 58. .

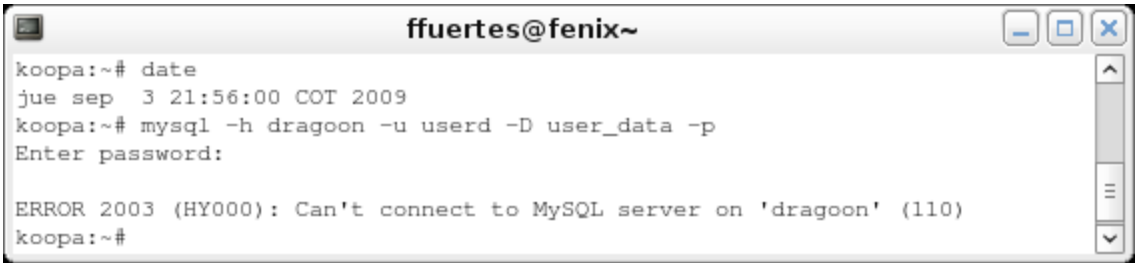

Figura 58. Intento de inicio de sesión MySQL no autorizado hacia *dragoon*

Posteriormente, se revisa el log *iptables.log* en donde se guarda este evento con la etiqueta de IMSeg_RECHAZADO (ver Figura 59.), lo cual identifica a este paquete como descartado por el Kernel.

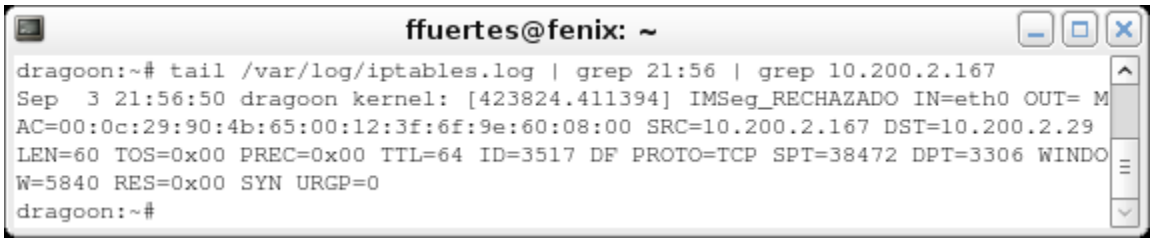

Figura 59. Revisión del log *Iptables.log* en *dragoon* donde se registra el intento de inicio de sesión MySQL fallido

- ARP Alert

Esta herramienta, guarda automáticamente registro de anormalidades presentadas en el tráfico ARP de la red local, lo cual alerta de posibles intentos de ataques como envenenamiento ARP. Para la prueba se realiza un ataque similar a los vistos en secciones anteriores utilizando la herramienta *Ettercap* como lo muestra la Figura 60. .

Figura 60. Lanzamiento del ataque de envenenamiento ARP hacia *zealot* y *dragoon*

Ahora, se busca en el log *arpalert.log* el registro de este evento como se muestra en la Figura 61. , donde se presentan las anormalidades en la tabla caché ARP del equipo *zealot* provenientes del ataque de envenenamiento anterior, en el cual se puede observar que la misma dirección MAC (00:12:3f:9e:60) pretende ser asociada a dos direcciones IP (10.200.2.29 y 10.200.2.167).

Figura 61. Revisión del log *arpalert.log* en *zealot* donde se registran las anormalidades en la tabla caché ARP

- Revisión de permisos de logs

Como se ha visto, los archivos de log son elementos importantes en cualquier sistema de seguridad, puesto que pueden brindar información valiosa sobre eventos o anomalías referentes al sistema que ayude a encontrar causas a problemas puntuales, por lo tanto,

deben ser totalmente confidenciales, siendo *root* el único usuario con capacidad para manipular estos archivos.

En la Figura 62. , se muestra que los permisos de los *logs* vistos anteriormente son de lectura y escritura únicamente para el dueño de los mismos que es el usuario *root.* Con esto se reserva el acceso a estos archivos únicamente al administrador del sistema.

Figura 62. Revisión de los permisos de los archivos de log en *dragoon*