A prototype for a honeynet based on SDN

Andrés Manzano
Escuela Politécnica Nacional
Ladrón de Guevara E11-253
Quito – Ecuador
(593) 2 2976-300
andres_manzano017@hotmail.com

José Naranjo
Escuela Politécnica Nacional
Ladrón de Guevara E11-253
Quito – Ecuador
(593) 2 2976-300
joseluisnaranjo@hotmail.es

Iván Bernal
Escuela Politécnica Nacional
Ladrón de Guevara E11-253
Quito – Ecuador
(593) 2 2976-300
ivan.bernal@epn.edu.ec

David Mejía
Escuela Politécnica Nacional
Ladrón de Guevara E11-253
Quito – Ecuador
(593) 2 2976-300
david.mejia@epn.edu.ec

ABSTRACT
This paper presents a prototype for a honeynet based on SDN. The prototype is structured by a Pyretic controller, an application, a switch with OpenFlow support and a network. The network is composed of two segments; one of them called the production network, and the other, the honeynet. The application helps detecting and deviating attacks to the honeynet. Six different types of attacks are considered, three of them are Denial of Service (DoS) attacks and the other two are spoofing attacks. The DoS selected attacks are SMURF, THC SSL DoS and TCP SYN Flood; and the spoofing attacks are based on DNS, IP and ARP.

The application contains a main module that is responsible for calling six additional modules, each designed for detecting a particular attack; the main module also loads a secondary module that injects the analyzed packets back into one of the two segments of the network. The application can be customized using a configuration file that flags whether the application will run a specific module or the six modules simultaneously; this allows protecting the network from either a particular attack or from all the six chosen attacks. A description of the developed modules using Pyretic is included. In addition, results of several tests carried out using the prototype for testing attacks are presented.

Categories and Subject Descriptors
C.2.0 [**Computer-Communication Networks**] - General-Security and Protection **(*honeynet*)**

General Terms
Security.

Keywords
Honeynet, Networks Security, Pyretic, Python, SDN.

EATIS'2016, April 27–29, 2016, Cartagena, Colombia.
978-1-5090-2435-3/16/$31.00 ©2016 IEEE.

1. INTRODUCCIÓN
Las redes de datos se han convertido en una necesidad fundamental en las empresas, las cuales demandan continuamente nuevos servicios y requerimientos; debido a que las redes tradicionales no fueron diseñadas para satisfacer muchas de estas necesidades, se deben incorporar periódicamente modificaciones que permitan proporcionar soluciones ante tales demandas.Por este motivo, se han buscado soluciones que permitan mejorar las redes tradicionales; una alternativa que conceptualiza las redes como programables, ágiles y flexibles y, por supuesto, que permite introducir fácilmente innovaciones, se denomina Redes Definidas por Software (SDN) [1].

Adicionalmente, el crecimiento incontrolable de las redes de datos y el aumento de aplicaciones y nuevas tecnologías, han provocado que la seguridad de la información se convierta en un requerimiento fundamental en cualquier sistema; en particular se presentan situaciones críticas que deben manejarse adecuadamente, sobre todo aquellas que involucran transacciones comerciales e información personal.

En el presente trabajo, se presenta una solución alternativa a ciertos problemas referentes a seguridad informática haciendo uso de un prototipo de honeynet basado en una SDN. El prototipo permite detectar seis diferentes tipos de ataques informáticos y desviarlos hacia una honeynet en la que se puede monitorear la forma de operar de los atacantes o inclusive descubrir nuevas herramientas utilizadas por los mismos. El prototipo está conformado por un controlador, una aplicación, un switch que soporta el protocolo OpenFlow, una red conformada por dos segmentos, uno con equipos que se consideran parte de la red de producción, y otro con equipos que forman parte de la honeynet. La aplicación es capaz de procesar y detectar ataques provenientes de equipos propios de la red o externos a la misma, y en caso de detectar un ataque desviará ese tráfico al segmento de la honeynet.

Este documento está organizado de la siguiente manera: presenta una breve revisión sobre SDN, honeynets, el controlador Pyretic y varios ataques informáticos; luego se describen los módulos desarrollados para el controlador, un par de ejemplos de pruebas ejecutadas, y, finalmente, las conclusiones y trabajo futuro.

2. REDES DEFINIDAS POR SOFTWARE
Las SDN son una nueva arquitectura de red en la que se separa el plano de control del plano de datos, de los dispositivos de red [2]; el plano de control se lo deriva a un computador denominado

controlador. Eta separación reduce significativamente la inteligencia de los dispositivos tradicionales trasladándola al controlador. Los dispositivos son tontos pero muy rápidos en las tareas de reenvío (*forwarding*) de los paquetes.

La ONF (*Open Networking Foundation*) define a las SDN como una arquitectura emergente, dinámica, manejable, rentable, y adaptable, lo que es ideal para manejar el gran ancho de banda demandando por las aplicaciones de hoy y su naturaleza dinámica [13]. Esta arquitectura desacopla el control de la red y las funciones de reenvío, permitiendo así que el control de la red sea directamente programable y que la infraestructura subyacente sea abstracta para las aplicaciones y servicios de red.

En la Fig. 1 se presenta la arquitectura de una SDN. Cada capa de esta arquitectura cumple una función específica: a) en la capa aplicación se ejecutan las diferentes aplicaciones que deben especificar los recursos y comportamiento que requieren de la red; b) la capa de control se encarga de establecer las políticas de reenvío de paquetes, pudiendo llevar a cabo tareas de enrutamiento, conmutación, aislamiento, ingeniería de tráfico, entre otras; además, debe tomar decisiones en base a la topología de la red y debe configurar cada dispositivo de red para que cumpla las funciones requeridas; c) la capa de infraestructura está conformada por todos los dispositivos de la red, los cuales no disponen de mecanismo alguno de toma de decisión respecto a que hacer con los datos, realizará únicamente procesamiento y entrega de paquetes en función de las políticas que el plano de control le indique mediante reglas que han sido instaladas en los dispositivos de red. Para poder gestionar la instalación de las reglas de renvío que se aplicarán a los flujos de datos en los dispositivos de red, se emplea el protocolo OpenFlow.

3. HONEYNETS

Una honeynet es un segmento de red con vulnerabilidades previamente configuradas, dispuesta como un blanco de ataque. La honeynet debe diseñarse para recolectar información de posibles amenazas; sin embargo, su objetivo no está completamente definido puesto que depende del uso que se le dé, por ejemplo, puede ser para la recolección de información cuando se la usa con propósitos de investigación, o se la puede usar para detectar e inclusive engañar a atacantes, cuando el propósito es evitar un ataque a un posible servidor real, u otros objetivos que podrían ser militares, políticos, gubernamentales, entre otros [3].

Una honeynet puede contener una serie de dispositivos de red, conocidos como honeypots, que pueden ser de diferentes fabricantes. Un ejemplo de una red que incluye una honeynet se presenta en la Fig. 2. Los sistemas que conforman la honeynet pueden ser equipos estándar, con aplicaciones reales instaladas y con configuraciones por defecto, intentando, no revelar que se trata de un sistema señuelo.

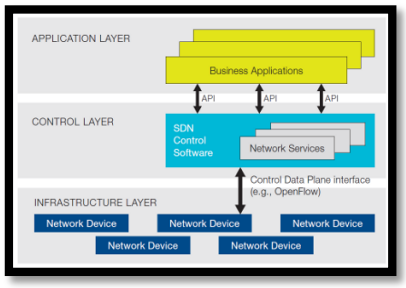

Figura 1. Arquitectura de las SDN [13]

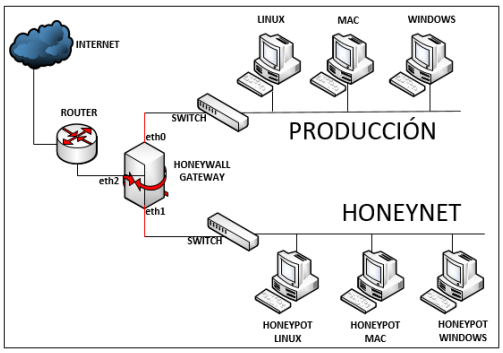

Figura 2. Ejemplo de una Honeynet

En la literatura se han encontrado proyectos de honeynets que hacen uso de SDN, como por ejemplo HoneyMix [6], una honeynet inteligente basada en SDN que mantiene un mapa de todos los servicios disponibles en la red; y, FRESCO [14], un framework para desarrollo de aplicaciones diseñado para facilitar el diseño rápido y la composición de módulos para detectar y mitigar ataques informáticos.

4. EL CONTROLADOR PYRETIC

El "Proyecto Frenetic" diseñó un sistema de tiempo de ejecución simple, reusable y de un alto nivel de abstracción para programar las SDN, que genera e instala automáticamente las reglas de bajo nivel en cada switch. Permite administrar una red mediante: políticas de envío de paquetes, monitoreo de las condiciones de la red, y políticas de actualización automática para responder a diferentes eventos generados en la red. Este proyecto desarrolló el controlador Pyretic, una plataforma de programación escrita en Python que habilita la creación de software modular, permitiendo crear sofisticadas aplicaciones para SDN [5].

Pyretic ofrece tres modos de operación [4]: a) Interpretado, en el cual el controlador instala solo una regla en el switch que establece que todo paquete debe enviarse al controlador; b) Reactivo, en el cual el controlador instala reglas conforme procesa paquetes enviados por el switch al controlador; y, c) Proactiva, en el cual el controlador instala reglas previamente en el switch. Para ejecutar el controlador Pyretic, se requiere especificar el archivo que inicia el controlador (`pyretic.py`), seguido del modo de operación (`-m`) y, finalmente, el archivo que contiene la aplicación que ejecutará el controlador. A continuación se indica un ejemplo del comando que permite iniciar al controlador Pyretic en el modo interpretado y con la aplicación `controlador.py` que debe ejecutar, la cual debe ubicarse en el directorio `pyrectic/sdnhoneynet/`.

$ pyretic.py -m i pyretic.sdnhoneynet.controlador

El Proyecto Frenetic, proporciona una máquina virtual preinstalada para arquitecturas de 32 y 64 bits, con el controlador Pyretic, disponible para descargar en [7].

5. ATAQUES INFORMÁTICOS

ARP Spoofing [11] consiste en el envío de paquetes ARP falsos en la red, con el objetivo de asociar la dirección MAC del atacante con la dirección IP de otro equipo, de tal manera que todo el tráfico dirigido a la dirección IP de ese nodo, sea erróneamente enviado al atacante, en lugar de a su destino real. IP Spoofing [8] consiste en suplantar la dirección IP de la víctima, para lo cual el atacante generará solicitudes usando las direcciones IP de otros

equipos de la red como dirección IP destino, y como dirección origen, la dirección IP de la víctima. De esta forma, el paquete llegará al equipo destino el cual analizará el paquete y generará una respuesta que se enviará al equipo de la víctima. DNS Spoofing [10] consiste en que un equipo malicioso realiza la traducción de nombres de dominio a direcciones IP en lugar del servidor DNS, suplantando la identidad del servidor DNS.

El ataque TCP SYN FLOOD [12] consiste en enviar paquetes SYN para que la víctima responda con paquetes SYN/ACK a cada solicitud, pero el atacante simplemente descarta dichas respuestas y continúa enviando paquetes SYN. El ataque TCH SSL DoS [15] consiste en enviar mensajes ClientHello y desechar los mensajes ServerHello empleados en el handshake de SSL. Para el servidor resulta en mayor procesamiento que para una conexión normal ya que después de establecer la conexión TCP, responde a cada uno de las solicitudes del atacante e intercambia credenciales indefinidamente. Smurf [9] utiliza el protocolo ICMP, el atacante envía un mensaje Echo Request a la dirección destino de *broadcast*, y con dirección origen de la víctima. Este mensaje llega a todos los equipos de la red y estos responden a la víctima, pudiendo saturarle dependiendo del número de equipos en la red y el número de respuestas generadas.

6. APLICACIÓN

La aplicación desarrollada consta de varios módulos: un módulo principal, denominado controlador, que será ejecutado por Pyretic y que procesará cada paquete que el switch reciba desde o hacia la LAN, haciendo uso de los diferentes módulos de procesamiento, los cuales deciden si el paquete debe enviarse a la LAN o a la honeynet, y procederá a llamar al módulo que envía los paquetes de acuerdo a lo establecido por los módulos de procesamiento. El módulo denominado ARP analiza el paquete para determinar si es un posible ataque ARP Spoofing; el módulo denominado IP analiza el paquete para determinar si es un posible ataque IP Spoofing; el módulo llamado TCP permite determinar si el paquete es parte de un ataque TCP SYN Flood; el módulo UDP permite decidir si es un ataque DNS Spoofing; el módulo ICMP permite conocer si se trata de un ataque Smurf; el módulo HTTPS sirve para conocer si es un ataque TCH SSL DoS; y el módulo denominado Enviar se encarga de insertar los paquetes en la red. La Fig. 3 y la Fig. 4 presentan los diagramas de flujo de la aplicación.

6.1 Módulo Controlador

El módulo controlador recibe y procesa cada uno de los paquetes que llegan del switch. Permite establecer qué módulo se cargará para el análisis de los paquetes. En caso de usar varios módulos, la aplicación los ejecuta secuencialmente dependiendo del tipo de paquete que se haya recibido, o solamente se procesa el paquete en un módulo si así se lo estableció. Este módulo lee la configuración desde un archivo, y en función de la variable `PROCESO` definida en tal archivo, inicia o bien todos los módulos o solamente uno de ellos. Una vez analizado el paquete (en uno o varios módulos), se devuelve al módulo controlador un indicador que establece a donde se enviará el paquete analizado.

6.2 Módulo arp.py

La Fig. 5 presenta el diagrama de flujo de este módulo. En este módulo se verifica si el paquete es ARP y se discrimina entre los distintos tipos de paquetes ARP. Si no es el caso, se comprueba si el paquete se dirige al atacante, y se lo envía únicamente por el puerto del switch en donde está conectado el atacante. Si el paquete se origina en un atacante, se lo envía a la honeynet.

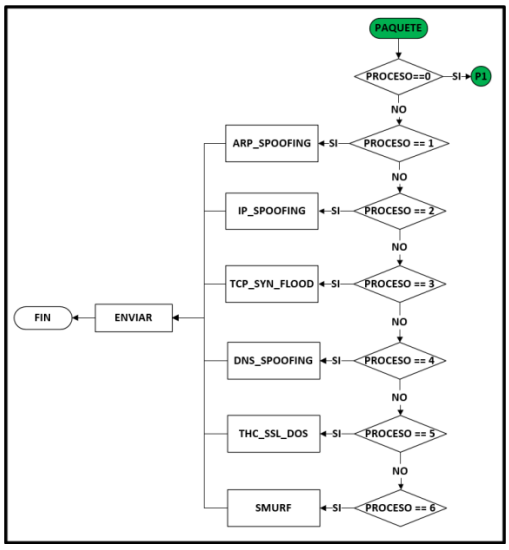

Figura 3. Diagrama de flujo de la aplicación (parte I)

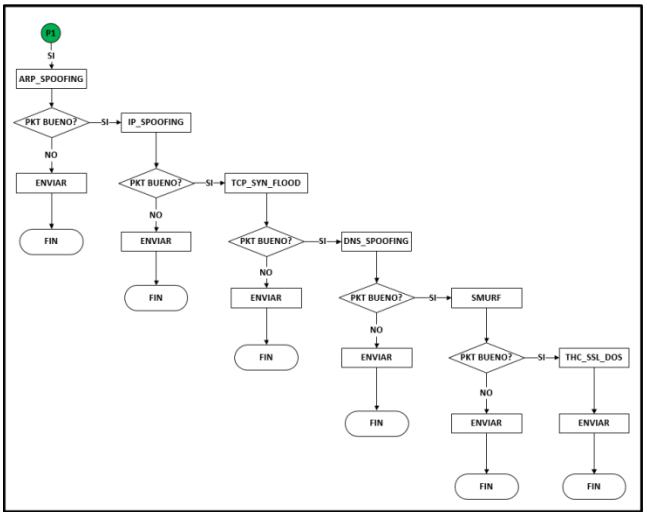

Figura 4. Diagrama de flujo de la aplicación (parte II)

Si se trata de un paquete ARP_REQUEST, se agrega la dirección MAC origen al diccionario `dicMacIp`, debido a que este ataque consiste únicamente en enviar paquetes del tipo ARP_REPLY. Si se recibe un paquete del tipo ARP_REPLY, se verifica si la dirección MAC origen se encuentra en el diccionario `dicMacIp`, si se encuentra se comprueba si la direcciona IP asociada a esta dirección MAC es la misma que la dirección IP origen del paquete, de ser así, se considera al paquete como legítimo y se lo envía a la LAN, además se comprueba si la direcciona MAC origen estaba en la lista de atacantes para proceder a removerla. En caso de que la dirección IP del diccionario y la del paquete recibido no sean iguales se procede a eliminar la entrada del diccionario, generando así una oportunidad antes de declarar un ataque, puesto que podría deberse a que un servidor DHCP de la LAN le asignó una nueva dirección IP, sin embargo, el paquete será enviado a la honeynet para evitar cualquier riesgo en la LAN. Si la dirección MAC del paquete ARP_REPLY no se encuentra en el diccionario `dicMacIP`, se procede a verificar si desde la dirección IP origen existen solicitudes ARP_REQUEST. Si no existen solicitudes, se lo clasifica inmediatamente como atacante (es imposible recibir respuestas si no existen solicitudes) agregando la dirección MAC origen a la lista de atacantes y

además se almacena el puerto del switch por el que se recibió el paquete en el diccionario `dicMacPuerto`, para que cuando lleguen paquetes destinados al atacante se los envíe a él únicamente y no a toda la LAN o a la honeynet, finalmente el paquete se envía a la honeynet.

En caso de que si existieron paquetes del tipo ARP_REQUEST, se verifica si se han recibido más respuestas que solicitudes, esto también indicará que se trata de un ataque por lo que se agrega la dirección MAC origen a la lista de atacantes y el número de puerto al diccionario `dicMacPuerto`.

6.3 Módulo ip.py

La Fig. 6 presenta el diagrama de flujo de este módulo. Este módulo verifica si la dirección MAC origen se encuentra en la lista de atacantes, de ser así, se desvía el paquete a la honeynet. Si la dirección MAC origen del paquete recibido no está en la lista de atacantes, se determinará el tipo de paquete. Si se trata de un paquete ARP, entonces se agrega un nuevo registro en el diccionario `dicIpMac` y se envía el paquete tanto a la LAN como a la honeynet, para poblar las tablas ARP de las dos redes, de tal forma que en el caso de descubrir un atacante y desviarlo a la honeynet, no se pierda la conectividad y el atacante no perciba el desvío realizado.

Si se recibe un paquete IP, se verifica si proviene de la honeynet, y luego se verifica la existencia de un registro en el diccionario `dicMacPuerto`. La ausencia del registro indica que el paquete está destinado a la LAN y, por lo tanto, se descarta, pues no debe existir conectividad entra la LAN y la honeynet. Si existe el registro, se procede a enviar el paquete únicamente al atacante. En el caso que el paquete recibido no provenga de la honeynet, se verifica si la dirección IP origen se encuentra en el diccionario `dicIpMac`, si no hay un registro se desvía hacia la honeynet, pues si un equipo no ha enviado ni recibido ningún paquete ARP, no puede enviar paquetes IP; si hay el registro y la dirección MAC asociada a la dirección IP del registro corresponde a la dirección origen del paquete, entonces se lo considera legítimo y se establece que debe pasar a la LAN, en cualquier otro caso, se agrega la dirección MAC origen del paquete a la lista de atacantes y se guarda el número de puerto por el que se recibió el paquete en el diccionario `dicMacPuerto`, y finalmente se establece que el paquete debe enviarse a la honeynet.

6.4 Módulo tcp.py

La Fig. 7 y la Fig. 8 presentan el diagrama de flujo de este módulo En este módulo se verifica si es un paquete TCP. Si se trata de otro tipo se revisa si proviene de un atacante, y dependiendo de eso se lo deja pasar a la LAN o se lo desvía a la honeynet.

Si es un paquete TCP, se verifica si la dirección IP origen es la del servidor, y se lo deja pasar a la LAN. Si el paquete TCP recibido proviene de otro origen, se diferencia entre los distintos paquetes que se envían en el establecimiento de conexión de tres vías. Si el paquete recibido no es del tipo SYN, se procede a comprobar si se trata de un paquete del tipo ACK, y en caso de que tampoco lo fuera, se verifica si la dirección IP origen está en la lista de atacantes para enviar el paquete a la honeynet o en caso contrario a la LAN. Si el paquete es ACK, esto indica la confirmación de la conexión y, por lo tanto, se garantiza que no es un ataque; si la dirección IP origen se encuentra en el diccionario de solicitudes se la elimina y se la agrega al diccionario `dicClientes`, y se envía el paquete a la LAN. Si la dirección IP origen está en la lista de atacantes, se elimina de la lista de solicitudes y se la agrega al diccionario `dicClientes`. Si ya estaba en el diccionario `dicClientes` se envía el paquete a la LAN.

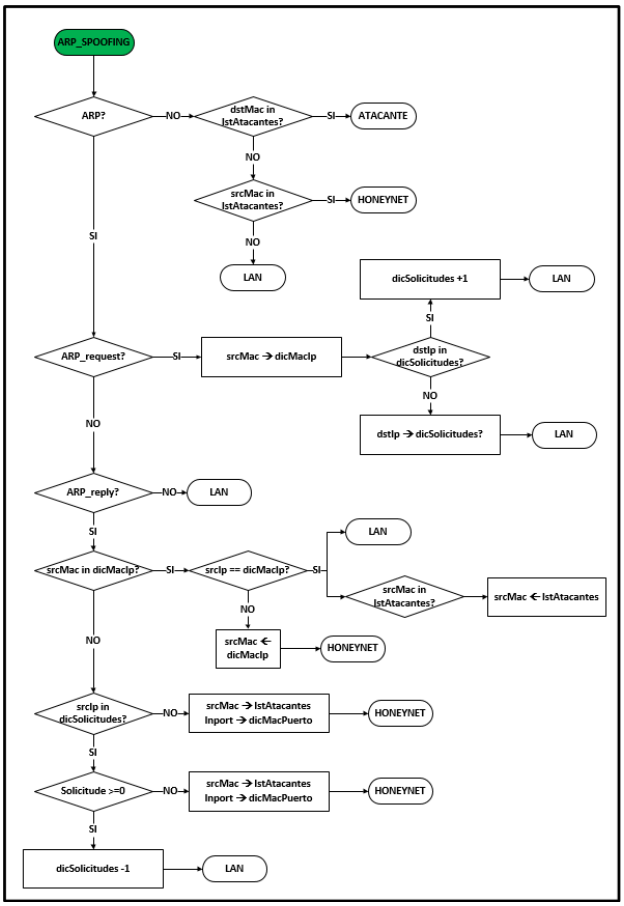

Figura 5. Diagrama de flujo del módulo ARP

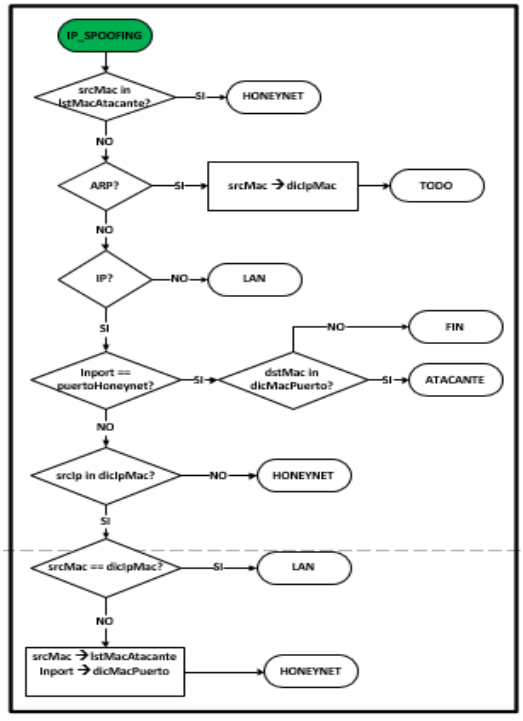

Figura 6. Diagrama de flujo del módulo IP

En el caso de que el paquete recibido corresponda al tipo SYN, se debe tener cuidado porque podría tratarse de un ataque. En este

caso, el procedimiento de verificación consiste en determinar si la dirección IP origen del paquete está en la lista de atacantes, en cuyo caso el paquete se desvía a la honeynet. Si no se encuentra en la lista de atacantes, se verifica si se encuentra en el diccionario `dicSolicitudes`. Si se encuentra en este diccionario, se debe comprobar que el número de solicitudes recibidas sea menor al máximo establecido en el archivo de configuración, si es así, se debe incrementar el contador de solicitudes de ese cliente y se deja pasar el paquete a la LAN; pero si el número de solicitudes ha alcanzado el máximo permitido, entonces se elimina el registro de `dicSolicitudes`, y se agrega la dirección IP origen del paquete a la lista de atacantes y se envía el paquete a la honeynet. En caso de que la dirección IP origen no esté en el diccionario `dicSolicitudes`, se comprueba si está en el diccionario `dicClientes`, si no está en este diccionario, se la agrega al diccionario `dicSolicitudes` indicando que es la primera conexión y se envía el paquete a la LAN. En caso de que si exista la dirección IP origen del paquete en el diccionario `dicClientes`, se debe verificar que el número de conexiones activas sea menor al máximo de conexiones permitidas por cada cliente. Si el número de conexiones es menor, entonces se incrementa el contador y se lo deja pasar a la LAN, pero si ha alcanzado el valor máximo, se elimina el registro de este diccionario y se agrega la dirección IP origen a la lista de atacantes. De esta manera cuando un ataque de este tipo provenga de una determinada dirección IP, se desvía hacia la honeynet todo el tráfico que genere mientras se comporte como atacante, sin embargo, si al acabo de un intervalo de tiempo nuevamente actúa como un cliente legítimo, se elimina de la lista de atacantes y se permite su tráfico a la LAN.

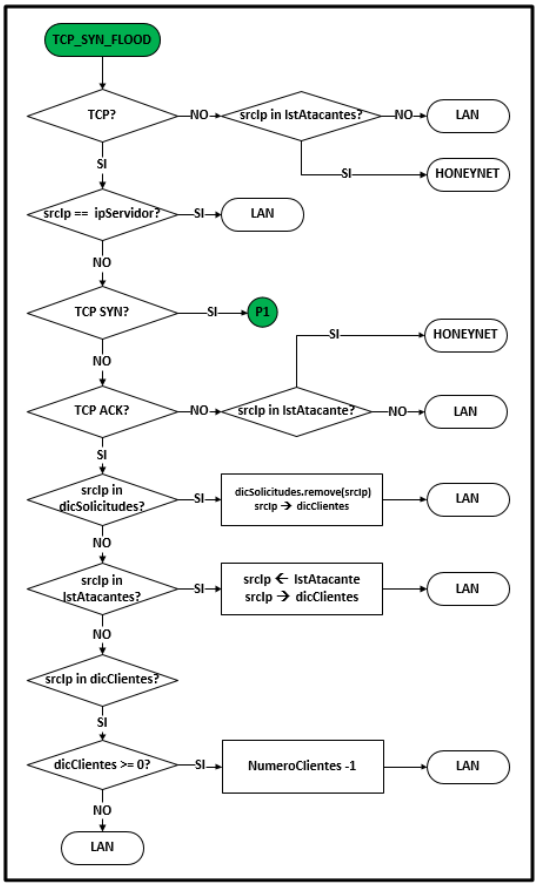

Figura 7. Diagrama de flujo del módulo TCP (parte I)

6.5 Módulo icmp.py

En la Fig. 9 se presenta el diagrama de flujo de este módulo. Este módulo determina el tipo de paquete y debido a que este ataque trabaja sobre el protocolo ICMP, cualquier paquete encapsulado sobre otro protocolo se lo deja pasar a la LAN.

En caso de que el paquete recibido sea del tipo ICMP, se procede a comprobar si la dirección IP de destino no es la dirección IP de *broadcast*; en caso de serlo, se desvía el paquete a la honeynet. Pero si el paquete recibido tiene una dirección IP destino diferente a la dirección IP de *broadcast*, entonces se deja pasar el paquete hacia la LAN.

6.6 Módulo https.py

En la Fig. 10 y la Fig. 11 se presenta el diagrama de flujo de este módulo. Este módulo verifica si se trata de un paquete TCP, debido a que HTTPS se encapsula sobre este, y para iniciar la negociación SSL debe establecerse una conexión TCP. Si se trata de otro protocolo y se origina en una dirección IP que esté en la lista de atacantes se lo desvía a la honeynet, sino se lo deja pasar a la LAN.

Si es un paquete TCP y no se origina en el servidor se lo deja pasar a la LAN. Si el paquete proviene de un origen diferente al servidor, se verifica si se trata de un paquete de inicio de sesión SSL (ClientHello), o de un paquete SSL_DATA, si no se trata de ninguno de estos, se verifica que no provenga de una dirección IP de la lista de atacantes, de no ser así se lo deja pasar a la LAN pues puede corresponder a paquetes de otras aplicaciones o inclusive a los paquetes de establecimiento de la conexión TCP previa a la negociación SSL. Si el paquete recibido es del tipo SSL_DATA, se trata de un usuario legítimo, debido a que el ataque THC SSL DoS se realiza en la negociación SS, luego se verifica si la dirección IP origen está en el diccionario `dicSolicitudes`, en cuyo caso se lo elimina y se agrega una nueva entrada en `dicClientes` indicando que se trata de la primera conexión y se envía el paquete a la LAN. Si la dirección IP origen no se encuentra en el diccionario `dicSolicitudes`, se verifica si está en la lista de atacantes, y en caso afirmativo se elimina este registro, se lo agrega a `dicClientes` pues se trata de la primera conexión y se envía el paquete a la LAN. Finalmente, si no se encuentra en la lista de atacantes, y está en `dicClientes`, se lo envía a la LAN.

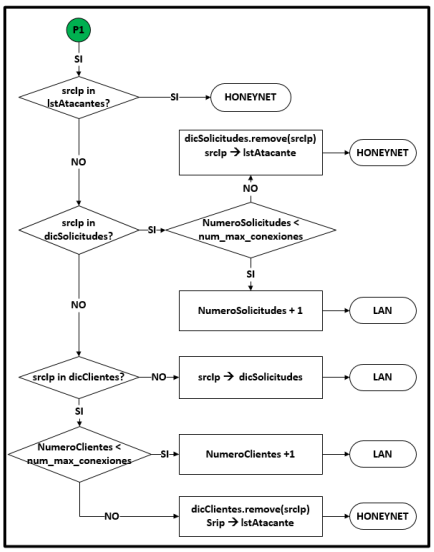

Figura 8. Diagrama de flujo del módulo TCP (parte II)

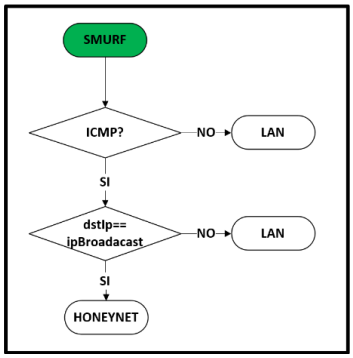

Figura 9. Diagrama de flujo del módulo ICMP

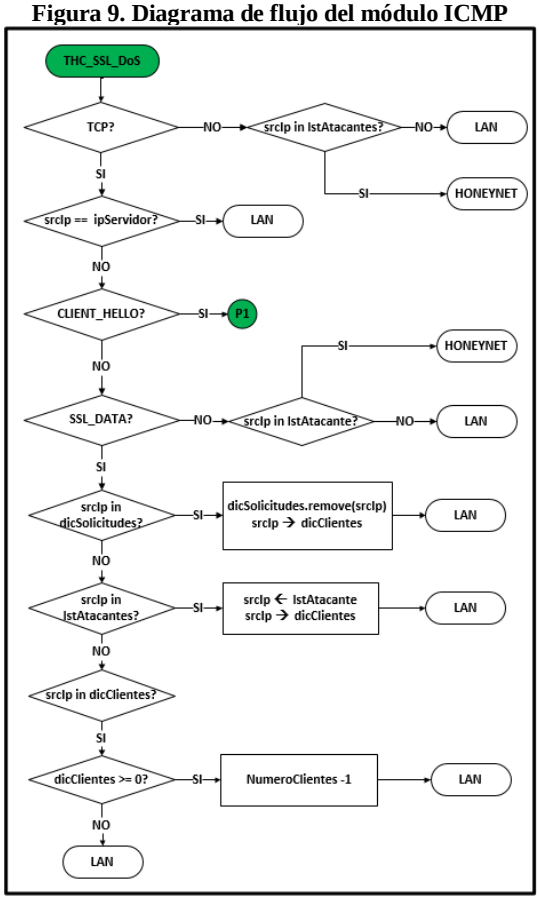

Figura 10. Diagrama de flujo del módulo HTTPS (parte I)

Si el paquete corresponde al tipo ClientHello, se verifica si la dirección IP origen está en la lista de atacantes, en cuyo caso se envía el paquete a la honeynet. Sin embargo, en caso de que no se encuentre en la lista de atacantes, se verifica si se encuentra en el diccionario `dicSolicitudes`. Si se encuentra, se verifica que el número de conexiones registradas en el diccionario de solicitudes no supere al número máximo permitido por cliente, si es menor se incrementa este contador y se deja pasar el paquete a la LAN. Pero en el caso de que haya alcanzado el número máximo permitido, se procede a eliminar la entrada del diccionario `dicSolicitudes` y se agrega la dirección IP a la lista de atacantes y a continuación se envía el paquete a la honeynet. Por otra parte, si la dirección IP origen no se encuentra en el diccionario `dicSolicitudes`, se procede a verificar si se encuentra en `dicClientes`, de ser así, se verifica que el número de conexiones que tenga el cliente no supere el máximo permitido. Si el número de conexiones registradas en el diccionario es menor a dicho número, se procede a incrementar el contador y se deja pasar el paquete a la LAN; en el caso de que se haya alcanzado el máximo, entonces se procede a eliminar la entrada de este diccionario y se agrega a la lista de atacantes. Finalmente, si la dirección IP origen tampoco está en el diccionario `dicClientes`, entonces se la agrega al diccionario `dicSolicitudes` y se inicializa en uno el número de solicitudes de este nuevo cliente.

6.7 Módulo udp.py

El diagrama de flujo de este módulo se presenta en la Fig. 12, en la cual se puede ver que al recibir el paquete, se revisa si la dirección MAC origen se encuentra en la lista de atacantes, en caso afirmativo, el paquete se desvía a la honeynet. Si no se encuentra en esta lista, se comprueba si se trata de un paquete UDP, en caso de no serlo se lo envía a la LAN, pero si se trata de un paquete UDP se analiza si es DNS_REQUEST, si no lo es, se lo deja pasar a la LAN. Pero si es un paquete DNS_REQUEST, se verifica si la dirección MAC destino corresponde a la dirección MAC del gateway, si es así se lo deja pasar a la LAN; caso contrario se agrega la dirección IP destino a la lista de atacantes y se descarta el paquete.

7. PRUEBAS DE FUNCIONAMIENTO

En esta sección se presentan los resultados de las pruebas de funcionamiento de dos ataques informáticos, uno del tipo spoofing y otro del tipo DoS.

7.1 Diagrama de red

La Fig. 16 y la Fig. 18 presentan un diagrama de la red empleada en las pruebas. En el Equipo 1 se ejecutará el controlador Pyretic y la aplicación desarrollada. El Equipo 2, dependiendo del caso, contendrá un equipo o los equipos que forman parte de la red real virtualizados. En el Equipo 3 se implementará la honeynet, o la red falsa, que tendrá una copia del equipo o los equipos que están en la red real. El Equipo 4 actuará como el atacante, en el cual se instalará la distribución Kali Linux.

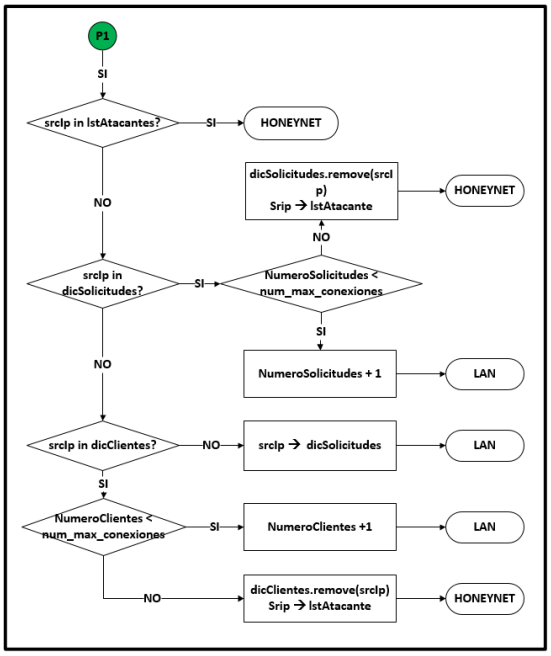

Figura 11. Diagrama de flujo del módulo HTTPS (parte II)

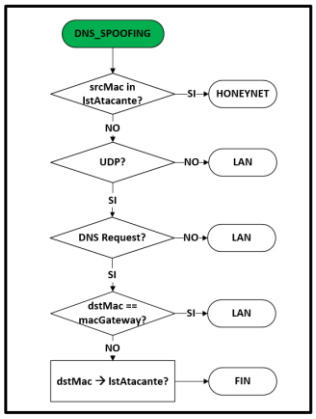

Figura 12. Diagrama de flujo del módulo UDP

7.2 Resultados obtenidos en ARP Spoofing

Para ARP Spoofing, la configuración empleada se presenta en la Fig. 16, donde: el Equipo 2 (víctima) tiene dirección IP 192.168.0.2 y dirección MAC 00:00:00:00:00:02; el Equipo 3 es una réplica del Equipo 2; el Equipo 4 (atacante) tiene dirección IP 192.168.0.3 y MAC 94:DE:80:7C:69:7E; y el Router (suplantado) tiene dirección IP 192.168.0.60 y MAC 00:1C:C0:EF:13:F9. La Fig. 13 presenta la captura de paquetes realizada en el Equipo 2, en la que se observa: un paquete del tipo ARP_REQUEST, proveniente del atacante (192.168.0.3) y solicitando la dirección MAC del equipo cuya dirección IP es 192.168.0.60 (equipo suplantado); un paquete del tipo ARP_REPLY proveniente del atacante, dirigido a la víctima, que indica que el equipo 192.168.0.60, posee la dirección MAC 94:DE:80:7C:69:7E, la cual corresponde al atacante; y los dos últimos paquetes de tipo ARP_REPLAY que se originan en el Router (192.168.0.60) e indican que la dirección MAC asociada es 00:1C:C0:EF:13:F9, lo cual es correcto. El controlador, al detectar el ataque procede a desviar todo el tráfico proveniente del Equipo 4 a la honeynet. La Fig. 14 muestra los paquetes ARP capturados en la honeynet.

Los paquetes capturados en la honeynet (Equipo 3) son parte del ataque, puesto que son respuestas ARP e indican que la dirección IP 192.168.0.60 tiene por dirección MAC 94:DE:80:7C:69:7E, y están destinados a la dirección de la víctima. La Fig. 16 muestra la tabla ARP del equipo réplica en la honeynet y se aprecia que la dirección IP 192.168.0.60 está asociada a la dirección MAC del atacante. Durante la realización del ataque se obtuvieron estadísticas en la honeynet, las mismas se pueden observar en la Fig. 17. Se capturaron 311 paquetes ARP (color rojo), que representan un 29% del tráfico total, de estos, 257 son ARP_REPLY (azul), y solamente 54 son ARP_REQUEST (verde), lo que implica que apenas un 17.36% fueron solicitudes ARP y el 82.64% corresponden a respuestas ARP.

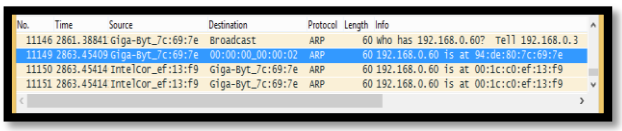

Figura 13. Captura de paquetes en el Equipo 2

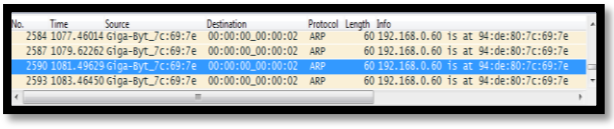

Figura 14. Tráfico capturado en la honeynet

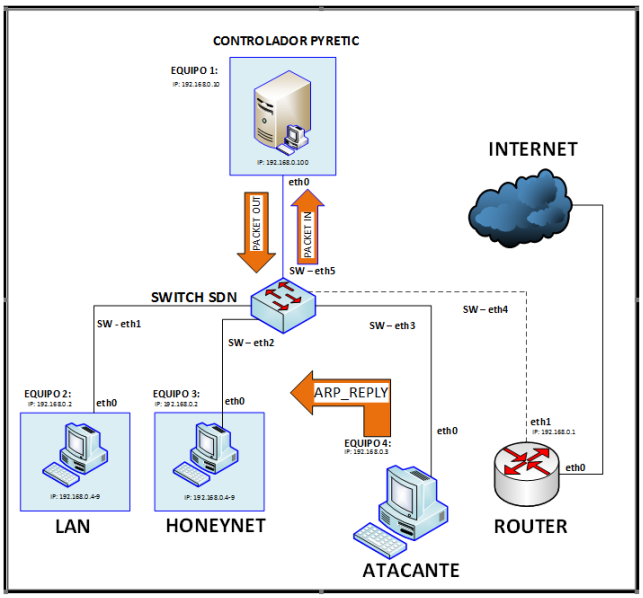

Figura 15. Ataque ARP_REPLY

```
C:\Windows\system32>arp -a

Interfaz: 192.168.0.2 --- 0xa
  Dirección de Internet          Dirección física      Tipo
  192.168.0.60          94-de-80-7c-69-7e     dinámico
  192.168.0.255         ff-ff-ff-ff-ff-ff     estático
  239.255.255.250       01-00-5e-7f-ff-fa     estático
```

Figura 16. Tabla ARP del equipo réplica en la honeynet

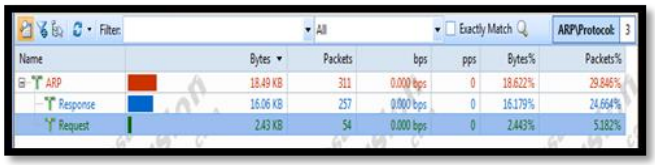

Figura 17. Estadísticas de paquetes ARP en la víctima

7.3 Resultados obtenidos en TCP SYN FLOOD

Para el ataque TCP_SYN_FLOOD, se empleó la siguiente configuración (ver Fig. 18): en el Equipo 2 se instaló un servidor web y se configuró la dirección IP 192.168.0.2; el Equipo 3 es réplica del Equipo 2; el Equipo 4 se configuró con la dirección IP 192.168.0.3 (atacante). En el servidor web no se detectan variaciones en el tráfico de datos, esto debido a que al detectarse el ataque, se desvía ese tráfico. En la honeynet se reciben datos en el equipo réplica. En la Fig. 19 se presenta la captura realizada en la honeynet durante el ataque, todos los paquetes que se reciben son del tipo SYN y provienen del equipo atacante (192.168.0.3) y se dirigen al equipo con dirección IP 192.168.0.2 (víctima).

La Fig. 20 muestra el tráfico capturado en la honeynet durante el ataque, en la cual se observa que la actividad de la red aumenta considerablemente, provocando la sobrecarga del equipo por el procesamiento de solicitudes falsas. La disminución que se observa en el intervalo de 460s a 480s se debe al colapso del equipo que intenta recuperarse y nuevamente procesar las peticiones que siguen llegando. Finalmente, la Fig. 21 presenta estadísticas obtenidas en la honeynet, de los paquetes recibidos durante el ataque. Se observa que de 8094 paquetes capturados, 7445 son 5de TCP, que equivale a un 91.98% del tráfico total.

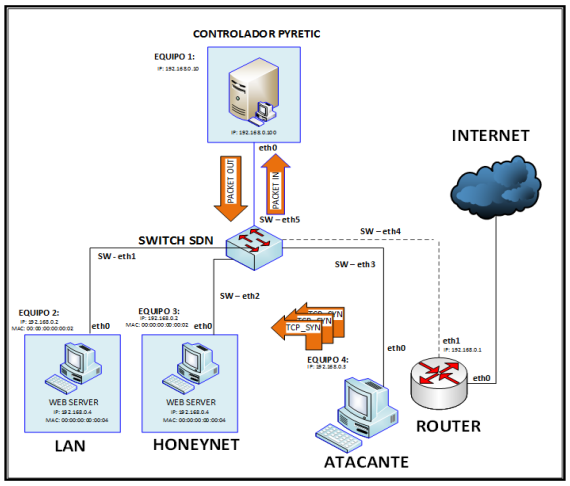

Figura 18. Ataque TCP_SYN_FLOOD

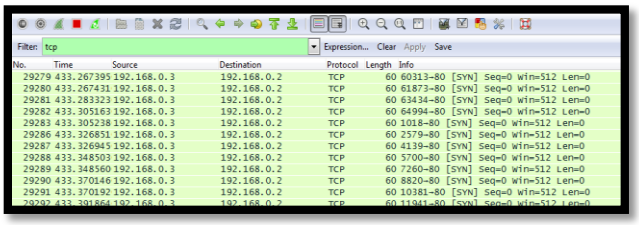

Figura 19. Captura de tráfico TCP durante el ataque

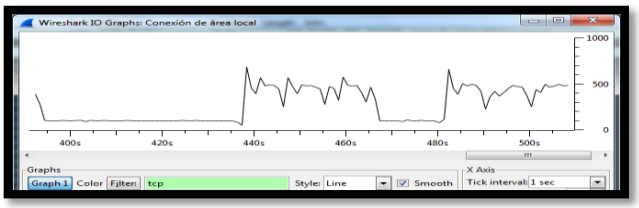

Figura 20. Variación del tráfico de red en la honeynet

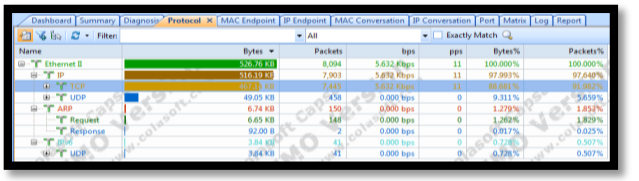

Figura 21. Estadísticas de paquetes capturados en la víctima

8. CONCLUSIONES

Con este prototipo, se pueden detectar seis tipos de ataques informáticos, entre los que se incluyen tres ataques de DoS y tres de Spoofing. Es preciso mencionar que al ejecutar todos los módulos de procesamiento, se incrementa considerablemente el tiempo requerido para procesar cada paquete por lo que es recomendable correr el controlador en un equipo robusto. La aplicación desarrollada permite detectar y analizar ataques informáticos, los mismos que pueden generarse en la red interna o provenir desde el exterior de la red y redirigirse a la honeynet. Este prototipo demuestra la flexibilidad con la que se cuenta en una SDN, en la cual un switch puede convertirse en un equipo capaz de enviar el tráfico a un segmento de la red se lo considera seguro, o de desviarlo a otro si se lo considera malicioso. Es posible insertar nuevas políticas mediante reglas que permitan tratar a los diferentes paquetes que se reciben en el controlador de manera estática o dinámica de acuerdo a las necesidades que se tenga en la red, por ejemplo, se pueden bloquear o desbloquear usuarios de acuerdo a las estadísticas recopiladas que indican si se trata o no de un atacante. Al utilizar una SDN el administrador de la red tiene control completo de la misma, los paquetes son tratados como se cree conveniente, lo que no sucede con las redes tradicionales. El trabajo futuro consiste en agregar y expandir la funcionalidad para manipular otros tipos de ataques, también se realizarán pruebas de carga para determinar la carga máxima que la aplicación podría soportar, y el uso de hilos para realizar el análisis de forma simultánea. El código de la aplicación y su documentación se encuentran disponibles en el repositorio:

https://github.com/joseluisnaranjo/sdnhoneynet.git

9. REFERENCIAS

[1] Cisco, [En línea]. Disponible en: https://www.cisco.com/web/ES/assets/pdf/networking_sdn_enhance_operator_monetization_wp.pdf

[2] Ericsson, [En línea]. Disponible en: http://www.ericsson.com/res/thecompany/docs/publications/ericsson_review/2013/er-software-defined-networking.pdf

[3] F. Leder, T. Werner: Honeynet Project, Know Your Enemy: Containing Conficker, USA, April 2009

[4] Frenetic, [En línea]. Disponible en: http://www.frenetic-lang.org/overview.php

[5] Frenetic, [En línea]. Disponible en: https://github.com/frenetic-lang/pyretic/wiki/running-pyretic

[6] H. Wonkyu, Z. Ziming, A. Doupé, G. Ahn, HoneyMix: Toward SDN-based Intelligent Honeynet. ACM International Workshop on Security in Software Defined Networks and Network Function Virtualization, USA, 2016

[7] J. Reich, *frenetic*, [En línea]. Disponibe en: https://bitbucket.org/reich/pyretic-vms/downloads/Pyretic-0.2.2-amd64.zip

[8] McMaster University, *IP Spoofing*. [En línea]. Disponible en: http://wiki.cas.mcmaster.ca/index.php/IP_Spoofing

[9] Norton Security, *Smurf*, [En línea]. Disponible en: http://es.norton.com/security_response/glossary/define.jsp?letter=s&word=smurf-dos-attack

[10] Olmedo Javier,Hackpuntes, Ataque DNS Spoofing con Cain&Abel, [En linea]. Disponible en: http://hackpuntes.com/ataque-dns-spoofing-con-cainabel/.

[11] R. Siles, Real World ARP Spoofing, [En línea]. Disponible en: http://pen-testing.sans.org/resources/papers/gcih/real-world-arp-spoofing-105411

[12] R. Margaret, *Syn-Flood*, [En línea]. Disponibe en: http://searchsecurity.techtarget.com/definition/SYN-flooding

[13] SDN, *Software Defined Network*. [En línea]. Disponible en: https://www.opennetworking.org/sdn-resources/sdn-definition

[14] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, M. Tyson, FRESCO: Modular Composable Security Services for Software-Defined Networks. ISOC Network and Distributed System Scurity Symposium, USA, 2013

[15] Thc.Org, *THC-SSL-DOS*, [En línea]. Disponible en: https://www.thc.org/thc-ssl-dos