INVESTIGACIÓN

Diseño de una aplicación SDN de seguridad y su impacto en el desempeño de la red

Design of SDN security application and its impact on the network performance

MSc. Ing. Yanko Antonio Marín Muro [1], DrSc. Félix Álvarez Paliza [2],
MSc. Ing. Abel Alfonso López Carbonell[3], MSc. Ing. Carlos Lester Dueñas Santos[4]

Recibido: 11/2017 | Aceptado: 02/2018

PALABRAS CLAVE

Redes Definidas por Software
OpenFlow
Mininet
Redes programables

RESUMEN

El nacimiento de las aplicaciones en tiempo real, la transmisión de video, la masificación de las redes sociales, la introducción de la computación en la nube y muchos otros servicios han dado como resultado el crecimiento exponencial del tráfico que circula por las redes de telecomunicaciones. En consecuencia, se plantea la necesidad de cambiar la forma de comunicación en las redes actuales hacia nuevos métodos que proporcionen mayor inteligencia a la misma. Es así que nace el concepto de las Redes Definidas por Software (SDN), el cual está revolucionando el campo de las comunicaciones mediante el control de los dispositivos de la red desde un software exterior. Serán las aplicaciones SDN que se ejecuten sobre el controlador de la red quienes gobernarán el comportamiento de los dispositivos. La aplicación que se expone en este artículo es un ejemplo de ello. Esta aplicación permite controlar el acceso de los usuarios a los recursos de una red, lo cual dota a la red de una mayor seguridad. Además, se ha comprobado su impacto sobre el desempeño de la red en términos de latencia y tasa de transferencia de datos. Los experimentos se han llevado a cabo en el emulador Mininet. La plataforma de prueba desarrollada en el trabajo junto a la aplicación creada evidencia las potencialidades de las SDN. Finalmente, se compara el impacto de la aplicación de seguridad en la red cuando se utilizan los métodos activo y proactivo.

KEYWORDS

Software Defined Network,
OpenFlow,
Mininet,
programmable networks

ABSTRACT

The birth of real-time applications, video transmission, the massification of social networks, the introduction of cloud computing and many other services have resulted in the exponential growth of traffic on the telecommunications networks. Consequently, it is necessary to change the way of communication in the current networks to new methods that provide greater network intelligence. This is how the concept of Software Defined Networks (SDN) is born which is revolutionizing the field of communications by controlling the device of the network from external

1 Empresa de Telecomunicaciones de Cuba S.A. La Habana, Cuba. yanko.marin@etecsa.cu
2 Universidad Central de Las Villas ¨Marta Abreu¨. Las Villas, Cuba. fpaliza@etecsa.cu
3 Empresa de Telecomunicaciones de Cuba S.A. La Habana, Cuba. abel.lopez@etecsa.cu
4 Empresa de Telecomunicaciones de Cuba S.A. La Habana, Cuba. carlos.duenas@etecsa.cu

software with the help of Openflow protocol specially created for this purpose. It will be the SDN applications running on the network controller that will control the behavior of the device. The application that is exposed in this paper is an example of this. This application allows to control the access of the users to the network resources, and improving network security. In addition it has been verified its impact on the performance of the network in terms of latency and throughput. The experiments have been carried out in the Mininet emulator and POX controller. The test platform developed in the work together with the application created evidences the potential of the SDN. Finally, the impact of the security application in the network when using active and proactive methods is compared.

Introducción

Durante mucho tiempo, las tecnologías de redes han evolucionado a un ritmo más lento en comparación con otras tecnologías de comunicaciones, lo cual ha incrementado la osificación de la red (Jarraya et al., 2014; Li y Chen,2015). La Internet basada en el Protocolo de Internet (IP) ha tenido gran éxito, centrándose principalmente en los conceptos tradicionales de conmutación y enrutamiento. Sin embargo, la complejidad de las redes actuales se enfrenta a problemas importantes entre los que figuran: la calidad del servicio, la seguridad, la gestión de la movilidad y escalabilidad (Kreutz, Ramos, Esteves Verissimo, Esteve Rothenberg, Azodolmolky y Uhlig, 2015).

Equipos de red como los conmutadores y enrutadores han sido tradicionalmente desarrollados por fabricantes que diseñan su propia lógica de control para operar sus dispositivos de una manera exclusiva y cerrada. Esto frena el avance de las innovaciones en las tecnologías de redes y provoca un aumento de los costos de gestión y operación, cada vez que se despliegan nuevos servicios, tecnologías y equipos. Este es un problema universal, que ha hecho que cada día las arquitecturas de redes sean cada vez más complejas y tengan una baja capacidad de escalabilidad. Se han propuesto durante todos estos años muchas soluciones para reemplazar la tecnología de red actual, pero nunca se han aplicado por ser extremadamente difíciles de probar (Nunes, Mendonca, Nguyen, Obraczka y Turletti, 2014).

La arquitectura de las redes actuales se compone de tres planos lógicos: plano de control, plano de datos y el plano de gestión. Hasta ahora, los equipos de redes se han desarrollado con los planos de control y de datos estrechamente acoplados. Esto aumenta significativamente la complejidad, el costo de administración y la gestión de la red. Debido a estas limitaciones, las comunidades científicas de redes y líderes industriales del mercado han colaborado con el fin de replantear el diseño de las redes tradicionales. De esta manera, surgió un nuevo paradigma de redes llamado "Redes Programables" con el objetivo de facilitar la evolución de la red. Su principio fundamental es permitir que la red sea más flexible, adaptable y dinámica (Jarraya, et al., 2014). Para materializar este concepto, surgieron por los años 90, dos escuelas separadas de pensamientos: OpenSig, de la comunidad de las telecomunicaciones, y Redes Activas, de la comunidad de las redes IP. Entre las principales razones por las que estos grupos no tuvieron éxito figuran los problemas de rendimiento, complejidad y seguridad introducidos en sus propuestas (Nunes, Mendonca, Nguyen, Obraczka y Turletti, 2014).

Después de las redes programables, surgieron otros proyectos con un nuevo enfoque relacionado con la separación de los planos de control y de datos, apoyado principalmente en los esfuerzos de crear interfaces abiertas y estándares entre estos dos planos; destacándose la arquitectura para la separación de los elementos de transmisión y control (ForCES). Otros proyectos que promueven la separación de planos de control y de datos son la plataforma de control de enrutamiento (RCP) y la arquitectura segura para las redes empresariales (SANE) y Ethane (Nunes, Mendonca, Nguyen, Obraczka y Turletti, 2014).

En los últimos años, también ha surgido el paradigma de las Redes Definidas por Software (SDN) como una propuesta para superar las limitaciones anteriormente mencionadas (Jarraya, et al., 2014); (Kreutz, Ramos, Esteves Verissimo, Esteves Rothenberg,

Azodolmolky, y Uhlig, 2015). Recientemente, las Redes Definidas por Software han ganado popularidad en la academia y la industria. Las SDN es una reformulación de investigaciones anteriores con los empeños de separar el plano de control y el plano de datos, así como centralizar la inteligencia de la red, abriendo un camino para la innovación mediante la programación de aplicaciones que pueden controlar directamente el plano de datos de la red. Al ser un nuevo concepto, no se ha alcanzado aún un consenso sobre su definición exacta. De hecho, una gran variedad de diferentes definiciones (Hu, Hao y Bao, 2014) han surgido en los últimos años, cada una tiene sus propios méritos. Según la *Open Networking Foundation* (ONF), consorcio sin fines de lucro dedicado al desarrollo, estandarización y comercialización de las SDN, la definición más aceptada es:

"Las Redes Definidas por Software se definen como una arquitectura de red dinámica, gestionable, adaptable, de costo eficiente. Lo cual la hace ideal para las altas demandas de ancho de banda y la naturaleza dinámica de las aplicaciones actuales. Esta arquitectura desacopla el control de la red y la funcionalidad de reenvío de información permitiendo que el control de la red pueda ser completamente programable logrando que las aplicaciones y servicios de red se abstraigan de la infraestructura de red subyacente".

Los beneficios del despliegue de las SDN son variados y entre ellos se destacan: visión unificada de la estructura de la red, enrutamiento mejorado, monitorización y alertas centralizadas, Ingeniería de Tráfico (TE) centralizada, manejo más rápido de fallos, reducción de la complejidad a través de la automatización, control centralizado de entornos de múltiples proveedores, mayor tasa de innovación y un aumento de la fiabilidad y la seguridad de la red (Hu, Hao y Bao, 2014).

En la arquitectura SDN, el plano de control es responsable de la configuración de los dispositivos de la capa de infraestructura. Un controlador OpenFlow instala las entradas de flujo en las tablas de reenvío de los conmutadores.

Los conmutadores tradicionales utilizan STP, SPB o TRILL para determinar cómo se reenvían los paquetes. OpenFlow traslada esta decisión de reenvío de los conmutadores a los controladores. Una aplicación de gestión se ejecutará en las interfaces del controlador que unen todos los conmutadores en la red, facilitando la configuración de caminos de reenvío que utilizarán todo el ancho de banda disponible. La especificación define el protocolo entre el controlador y los conmutadores y un conjunto de operaciones que se pueden realizar entre ellos. Las instrucciones de reenvío se basan en flujos y son aplicadas a todos los paquetes que comparten una serie de características comunes.

Un flujo en un conmutador puede estar definido por muchos criterios entre los que se encuentran: puerto donde se recibe el paquete, etiqueta VLAN, MAC origen o destino, protocolo, etc. Si un paquete no encuentra ninguna coincidencia en las tablas debe ser enviado al controlador SDN. Este controlador definirá un nuevo flujo para el paquete y enviará al conmutador una o más entradas para las tablas de flujo existentes. Para la próxima ocasión los paquetes que coincidan con este nuevo flujo no tendrán que esperar la respuesta del controlador, sino que serán encaminados a velocidad de línea por el conmutador OpenFlow. Existen dos enfoques para la gestión de las tablas de flujos de los dispositivos de la capa de infraestructura: el control de flujo reactivo y proactivo.

El controlador puede instalar entradas de flujo de forma permanente o temporal, en particular, antes de que realmente se necesiten. Este enfoque se conoce como gestión de flujo proactiva. Este método requiere de grandes capacidades de memoria del tipo TCAM. Estas memorias son costosas, por lo que un enfoque para optimizar la cantidad de entradas de flujos y la capacidad de memoria utilizada en los conmutadores es la instalación de estas reglas en demanda y de forma reactiva.

El enfoque reactivo solo contiene las entradas de flujo utilizadas recientemente en la tabla. Por un lado, esta perspectiva permite lidiar con pequeñas tablas de reenvío. Por otro lado, se requiere la interacción con el controlador si los paquetes de un flujo llegan a un conmutador que no tiene una entrada apropiada en la tabla de flujo. Después de un tiempo, la entrada correcta se instalará eventualmente en el conmutador para que los paquetes puedan reenviarse. El retraso resultante depende del canal de control y la carga actual del controlador. Por lo tanto, la gestión de flujos de forma reactiva reduce la necesidad de grandes tablas en los conmutadores, pero aumenta el retardo en la red y los requerimientos de confiabilidad del canal de control, así como del software del controlador.

La seguridad de la red es una parte notable de la seguridad cibernética y está ganando atención constantemente.

ISSN: 1813-5056 Vol. 14, No. 1 , enero-julio, pp. 5 - 14, 2018 tono

Las prácticas tradicionales relacionadas con la seguridad de red se implementan desplegando cortafuegos, SBC, servidores proxy para proteger una red física.

Debido a la heterogeneidad de las aplicaciones en las redes, garantizar accesos exclusivos para las aplicaciones a la red, implica la aplicación de una política en toda la red y una tediosa configuración de cortafuegos, servidores proxy y otros dispositivos (Banse y Schuette, 2017). En este aspecto, las SDN ofrecen una plataforma conveniente para centralizar, combinar y controlar las políticas y configuraciones para asegurarse de que la implementación cumple con la protección requerida. De esta manera, de una forma proactiva se evitan brechas de seguridad. Por otra parte, las SDN proporcionan mejores métodos para detectar y defenderse de ataques de forma reactiva. Debido a que las SDN tienen la capacidad de recopilar el estado de la red, se pueden analizar los patrones de tráfico en busca de amenazas de seguridad potenciales. Los ataques, como "ataques de ráfaga de baja velocidad" y denegación de servicio distribuido (DDoS), se pueden detectar simplemente mediante el análisis de patrones de tráfico.

Las redes SDN pueden ser desplegadas en cualquier entorno de red tradicional, desde las redes domésticas y empresariales hasta en los centros de datos y puntos de acceso a Internet. Tal variedad de entornos ha dado lugar a una amplia gama de aplicaciones de red de seguridad.

Una de las conclusiones que expone el informe "Ciberseguridad: un desafio mundial", que recoge los temas que se trataron en el foro Future Trends Forum, think-tank de la Fundación de la Innovación Bankinter, que tuvo lugar en Madrid en diciembre de 2015, es la siguiente:

El cibercrimen, es decir, los ataques contra empresas y gobiernos, se realiza cada vez con mayor frecuencia y son ataques más sofisticados y organizados, con beneficios millonarios para sus autores. De hecho, solo en el año 2015 las pérdidas mundiales por ciberataques fueron de $350 000 millones de dólares. Según los datos encontrados en empresas norteamericanas, británicas, alemanas y australianas, solo 35% de estas identifica un ataque informático en cuestión de minutos, 22% requiere al menos de un día para notarlo y 5% indicó que necesita por lo menos una semana para poder detectar amenazas.

El estudio que a continuación se presenta se ha centrado en el desarrollo de una aplicación de seguridad que se ejecuta sobre un controlador SDN basado en OpenFlow con el objetivo de demostrar que la mayoría de las funcionalidades de un cortafuego pueden ser implementadas por software sin la necesidad de un hardware dedicado.

A través de la capa de control, las aplicaciones SDN, que son consideradas el "cerebro de la red" pueden tener en tiempo real una visión global de la misma a través de la interfaz hacia el norte de los controladores. Utilizando esta información, las aplicaciones SDN pueden implementar mediante la programación estrategias para manipular las redes físicas subyacentes utilizando un lenguaje de alto nivel proporcionado por la capa de control.

De lo anterior se infiere la necesidad de crear una aplicación de seguridad para la investigación, enseñanza y evaluación del desempeño de las redes SDN basadas en el protocolo OpenFlow, lo cual conduce al problema científico de esta investigación:

¿Cómo implementar una aplicación SDN que permita detectar intrusos y controlar el acceso a los recursos de la red?

¿Cuánto será el impacto en la latencia y la tasa de trasferencia de datos en la red cuando se aplican políticas de seguridad?

La aplicación que se describirá permite el control de acceso a la red en tiempo real, detectando aquellos usuarios que traten de acceder a los recursos sin autorización. Se mostrarán los detalles de implementación de la aplicación, así como el resultado de la experimentación. Además, se analiza el impacto en la latencia y la tasa de trasferencia de datos en la red cuando se aplican políticas de seguridad en los modos reactivos y proactivos.

Materiales y métodos

Para la ejecución de la investigación se realizó un estudio descriptivo del estado del arte de la problemática en cuestión. Se desarrolló una aplicación de seguridad que se ejecuta sobre un controlador SDN basado en OpenFlow con el objetivo de demostrar que la mayoría de las funcionalidades de un cortafuego pueden ser implementadas por software sin la necesidad de un hardware dedicado. Se eligió el controlador POX entre muchos controladores por ser de código abierto, po-

seer buena documentación y estar escrito en Python que es un lenguaje de programación muy popular. El entorno de desarrollo utilizado fue el PyCharm Community Edition 4.0.4. Para crear la topología de red SDN, utilizó como hipervisor Oracle Virtual Box versión 5.0.14r105127 y Mininet 1.0 los cuales se ejecutan sobre una computadora con procesador Intel (R) Core ™ i5-2540M CPU @ 2.60 GHz y 4 GB de memoria instalada.

Resultados y discusión

Aplicación para el control de acceso a la red

Con el objetivo de desarrollar la aplicación de seguridad fueron manejados dos enfoques. El primero consistía en preestablecer las reglas en las tablas de flujo de forma proactiva; mientras que el segundo sería manejar los paquetes directamente de forma reactiva a medida que entren al conmutador. A pesar de que el manejo proactivo permite bloquear paquetes innecesarios desde la capa de infraestructura utilizando las ventajas del método de instalación de reglas de forma predeterminada, se decidió, debido a la flexibilidad que ofrece para la gestión de la red y la detección de eventos en tiempo real, manejar los paquetes entrantes en modo reactivo. Este enfoque requiere enviar todo el tráfico ARP al controlador para su análisis y la aplicación que se está ejecutando en él determina el comportamiento de los elementos en la capa de infraestructura.

La aplicación parte de una base de datos de usuarios autorizados. Además, debe ser capaz de recopilar aspectos importantes de la red tales como la topología, configuración de los puertos en los conmutadores y direcciones IP/MAC de los usuarios autorizados. En la figura 1 se muestra el escenario de prueba sobre el cual se ejecuta la aplicación. El escenario es creado con Mininet y está constituido por tres conmutadores con topología lineal, los que a su vez tienen conectado un host.

En dicha red se desea que solamente los usuarios autorizados (host h1 y h2) puedan acceder a la misma desde sus conmutadores, dirección MAC de la interfaz de red e IP especificados en una lista de control de acceso, la cual se muestra en la figura 2. Sin un usuario que no esté en esta lista, como es el caso del host "h3", trata de acceder a la red; se le debe denegar la comunicación

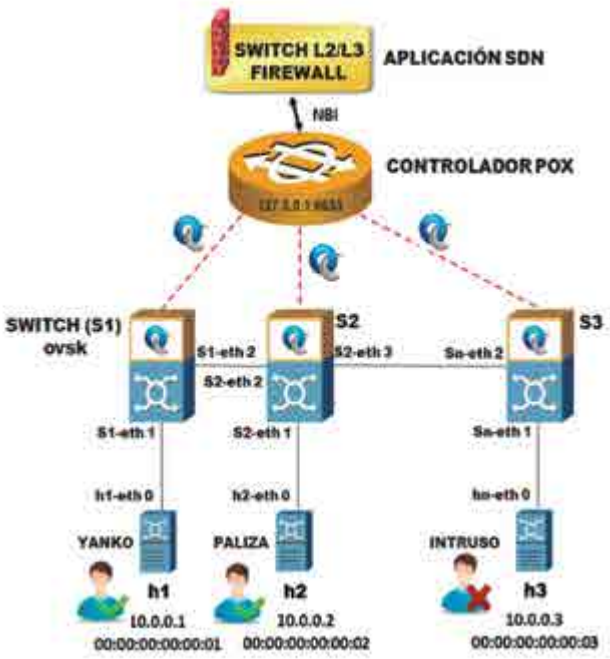

Figura 1. Escenario de prueba para la implementación de la aplicación SDN.

y generar una alarma que permita detectar intrusos en la red. También, si los usuarios autorizados tratan de acceder desde un conmutador, puerto, MAC, IP, no previamente autorizadas en la tabla se les deberá denegar el acceso a la red.

La aplicación para poder ejecutar estas políticas en la red primero debe ser capaz de realizar las funciones de un conmutador tradicional de capa dos en las que es necesario implementar un mecanismo en el que constantemente los conmutadores estén aprendiendo las direcciones MAC de las interfaces, así como el puerto por donde se observa el tráfico. En este proceso es vital la implementación del protocolo ARP que es la fuente principal para el aprendizaje de las direcciones MAC y el puerto del conmutador.

La creación de la base de datos de usuarios autorizados en la red fue implementada utilizando variables del tipo diccionario de Python. Los usuarios autorizados son previamente establecidos y no es objeto de esta investigación la gestión dinámica de los mismos. La figura 2 muestra el código fuente la creación de la base de datos anteriormente mencionada, así como la habilitación de las funciones de seguridad. Dado que POX está basado en Python, cualquiera de las estructuras de datos de este lenguaje puede usarse para almacenar la información necesaria.

ISSN: 1813-5056 Vol. 14, No. 1 , enero-julio, pp. 5 - 14, 2018 tono

```
# DECLARANDO LOS USUARIOS AUTORIZADOS
opcion_seguridad_arp=True;
opcion_seguridad_ip=True;
opcion_seguridad_registro=False;
usuario = {{'nombre':'YANNO','ip':'10.0.0.1',
            'switch':1, 'puerto':1,
            'mac':EthAddr("00:00:00:00:00:01")},
           {'nombre':'PALIZA','ip':'10.0.0.2',
           'switch':2, 'puerto':1,
           'mac':EthAddr("00:00:00:00:00:02")}};
```

Figura 2. Base de datos que contiene la lista de control de acceso.

En el diseño de la aplicación está previsto que cuando inicia la aplicación, se invoca automáticamente la función launch que es donde la aplicación registra todos los eventos y crea los objetos de cualquier clase de aplicación. En la línea 254 es creado el argumento de entrada "reactivo" para definir por línea de comandos el comportamiento de la aplicación en los modos reactivo y proactivo. (Figura 3)

```
def launch (reactivo = True):
  if reactivo:
    core.openflow.addListenerByName("PacketIn",
                                    _handle_PacketIn)
    log.info("Switch en modo Reactivo")
  else:
    core.openflow.addListenerByName("ConnectionUp",
                                    _handle_ConnectionUp)
    log.info("Switch en modo proactivo.")
```

Figura 3. La función launch es invocada en el inicio de la aplicación.

En el modo reactivo se invoca la función _ handle_PacketIn que es la encargada de recibir el tráfico proveniente de los conmutadores, identificar los protocolos; así como de establecer en los dispositivos de la capa de infraestructura las reglas necesarias para permitir o no la comunicación a partir del análisis de los puertos de entrada, conmutadores, direcciones IP y MAC de los dispositivos.

La función _ handle_PacketIn se activa cada vez que llega un mensaje OFPT_PACKET_IN al controlador. Lo primero que debe realizar la función es identificar el conmutador que envió la trama, obteniendo el dpid del evento (Figura 4). El puerto del conmutador por donde entra el tráfico se obtiene en la línea 97. El controlador debe posibilitar primero las funcionalidades de conmutación L2 y L3; para ello es necesario crear las tablas ARP e IP de todos los conmutadores. En la medida que el tráfico va circulando por la red el controlador va aprendiendo las direcciones MAC, IP y el conmutador de cada usuario al mismo tiempo que va comparando con la base de datos que posee la lista de control de acceso. El controlador va creando una visión de toda la red que permite aplicar políticas de seguridad en tiempo real a uno o múltiples nodos de la red.

```
def _handle_PacketIn (event):
  # OBTENIENDO EL ID DEL SWITCH
    dpid = event.connection.dpid
  # ASIGNANDO EL EVENTO A LA VARIABLE PACKET
    packet = event.parsed
  #PUERTO DEL SWITCH POR DONDE ENTRA EL PAQUETE
    inport = event.port
  # DETECTANDO SI ES UN NUEVO SWITCH
    if dpid not in TablaArp:
  # NUEVO SWITCH -- CREAR TABLA VACIA MAC/PUERTO
        TablaArp[dpid] = {}
  # NUEVO SWITCH -- CREAR TABLA VACIA IP/PUERTO
        TablaIP [dpid] = {}
  #DETECTANDO EL PROTOCOLO ARP 0x0806 (2054)
    if packet.type == 0x0806:
      procesamiento_arp(event)
    #DETECTANDO EL PROTOCOLO IP 0x0800 (2048)
  if packet.type == 0x0800:
      procesamiento_ip(event)
```

Figura 4. Función que maneja los paquetes entrantes.

Para el análisis de los paquetes en la red el controlador POX proporciona varias primitivas que permiten procesar paquetes bien conocidos.

La función _ handle_PacketIn además identifica (Líneas 105 y 108) en el tráfico entrante los protocolos ARP e IP para invocar las funciones de procesamiento de estos tipos de tráficos.

La figura 5 muestra el código fuente de la función que procesa el protocolo ARP.

Esta función lo primero que realiza es comprobar si están habilitadas las políticas de seguridad en la red y que el evento a analizar sea el protocolo ARP. Posteriormente, se obtiene el ID del switch, SRCMAC, DSTMAC, SRCIP y DSTIP para llenar las tablas ARP e IP de cada uno de los nodos de la red.

La aplicación detecta los mensajes de difusión ARP para darle la orden a los conmutadores que repliquen el mensaje por todos los puertos con excepción del puerto por donde se recibió el tráfico. Cuando se recibe una trama con una MAC de destino que no esté en la tabla ARP de la aplicación, se debe también aprender esta nueva dirección y difundir la trama por todos los puertos con excepción de por donde se recibió el tráfico. El caso en el que la trama recibida esté dirigida a una DSTMAC ya descubierta en la tabla

Figura 5. Función que procesa el protocolo ARP.

ARP se debe buscar el puerto y el conmutador donde se encuentra o alcanza ese usuario y enviar el tráfico a ese puerto en específico.

En las figuras 5 y 6 se puede observar que la función de procesamiento de tráfico ARP invoca otra función llamada "seguridad_arp", si está habilitada la opción de seguridad en la red. Esta función es el núcleo fundamental que garantiza la seguridad dentro de la red según los requerimientos de esta investigación y puede ser tan compleja como se desee. y pudiera invocar otras funciones de red tales como inspección profunda de paquetes (DPI), detección de ataques de denegación de servicio distribuidos (DDoS), entre otras.

Otro aspecto importante a tener en cuenta en el desarrollo de aplicaciones de este tipo es que es vital conocer la topología de la red para identificar los puertos que sirven como enlaces entre elementos de red. Supongamos que se permite que un usuario solamente pueda generar y recibir tráfico desde un conmutador/puerto/MAC/IP. Teniendo en cuenta el ejemplo de la figura 1, el usuario h1 solamente puede acceder desde el conmutador S1-eth1. Si el host h1 genera un paquete de difusión ARP, este debe llegar a todos los puertos involucrados en la red. Para que esto ocurra debe ese tráfico viajar por los enlaces troncales entre los conmutadores S1, S2 y S3. Cuando el paquete ARP llega al conmutador S2, este último lo envía al controlador para realizar un análisis de seguridad. El controlador entonces bloquea el tráfico porque la MAC de origen 00:00:00:00:00:01 no está permitida pasar por el puerto S2-eth2.

El caso ideal en este tipo de estudio es implementar aplicaciones SDN encargadas de descubrir la topología de la red y utilizar esta información para nutrir la aplicación de seguridad, de forma tal que se pueda solucionar el problema anteriormente planteado. Debido a que descubrir la topología de la red no es objeto del presente estudio, fue necesario crear una base de datos con los tipos de puertos de red (ver figura 7) donde se identifican los puertos como troncal o acceso.

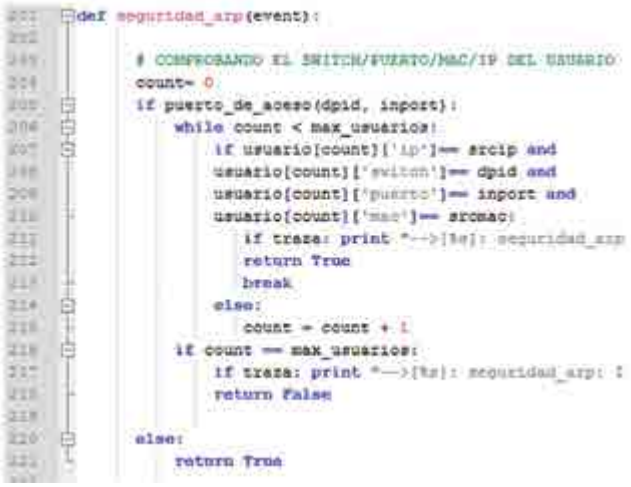

Figura 6. Fragmento de la función de seguridad.

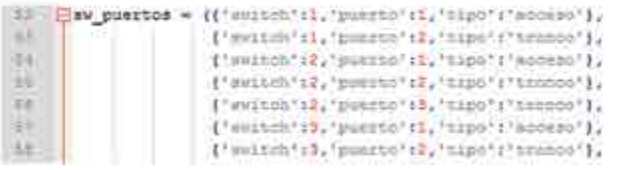

Figura 7. Variable del tipo diccionario que posee los tipos de puertos de red.

A partir de la base de datos de tipos de puertos de red, se creó la función que aparece en la figura 8 llamada "puerto_de_acceso" que se invoca desde las funciones "seguridad_arp" y "seguridad_IP" con el fin de aplicar las políticas de seguridad anteriormente mencionadas solamente en los puertos de acceso. En los puertos troncales se pudieran aplicar otras políticas de seguridad, lo cual puede ser abordado en futuras investigaciones.

Figura 8. Función que identifica si un puerto es de acceso.

ISSN: 1813-5056 Vol. 14, No. 1 , enero-julio, pp. 5 - 14, 2018 tono

Hasta este punto del trabajo se han mostrado las principales partes del código fuente de la aplicación creada, el código completo se puede encontrar en (Marín Muro, 2016).

Comprobación del funcionamiento de la aplicación

Para comprobar el funcionamiento de la aplicación desde Mininet se realiza un PING desde el host h3 al h1 en el que todos los paquetes son descartados, como se puede apreciar en la figura 9. Al mismo tiempo la aplicación SDN genera una alarma el día 11 de octubre de 2017 a las 20:37:12 indicando que se ha detectado un intruso en la red por el puerto 1 del conmutador S3 con dirección IP=10.0.0.3, MAC=00:00:00:00:00:03 tratando de acceder a la IP 10.0.0.1 que corresponde al host h1, ver figura 10.

Figura 9. PING desde el host h3 al h1 es descartado.

Figura 10. Alarma generada por la aplicación al detectar un evento de seguridad producido por el intento de acceso a la red de un usuario no autorizado.

Evaluación del desempeño de la red

Otro de los objetivos perseguidos con esta investigación es evaluar el impacto en la latencia y la tasa de trasferencia de datos en la red cuando se aplican políticas de seguridad en los modos reactivos y proactivos. Para ello, se creó una red SDN lineal con topología similar a la de la figura 1 pero de tamaño diferente hasta n usuarios y se configuraron las listas de acceso de la aplicación SDN de seguridad desarrollada para que el usuario autorizado se conecte desde los host (h2, h3...hn). Posteriormente, se va ampliando el tamaño de la red según el valor de n y luego se realiza una prueba de conectividad (Ping desde el usuario h1 a hn) en la que se computan la latencia mínima, máxima, promedio y la desviación estándar. También se computa la pérdida de paquetes, pero no se muestra en forma de gráfica porque en todos los casos fue cero. Fue utilizada la herramienta de código abierto IPERF, para medir la tasa de transferencia de datos entre los nodos h1 y hn con el objetivo de estudiar también el impacto en la tasa de transferencia de datos entre dos puntos de una red en crecimiento y con políticas de seguridad.

El experimento fue realizado con el siguiente procedimiento:

sudo./pox.pyforwarding.uclv_
L2_L3_SWITCH_ACL_TRAZA

sudo mn --topo linear,n --mac --switch ovsk --controller remote

h1 ping hn -c 3 & iperf

Analizando el resultado del experimento y que es mostrado en las figuras 11 y 12, se pudo comprobar que la aplicación de seguridad cuando es implementada en el modo reactivo genera una latencia en la red que depende del tamaño de la misma. Por su parte, la tasa de transferencia disminuye entre los nodos de la red en la medida que esta crece.

Posteriormente, se desarrolla el mismo experimento activando el módulo de seguridad de defensa proactiva y son analizadas las mismas métricas. Este módulo instala las políticas de seguridad en toda la red en el momento que el conmutador se conecta al controlador. De esta manera cuando ingresa una trama al conmutador, este último posee de forma previa las reglas y acciones para encaminar cualquier tipo de tráfico. Cuando una trama entrante no encuentra coincidencias, es enviada al controlador y analizada por el módulo de seguridad.

Analizando el resultado del experimento y que es mostrado en las figuras 13 y 14, se pudo comprobar que la aplicación de seguridad cuando es implementada en el modo proactivo reduce considerablemente la latencia en la red y aunque depende del tamaño de la misma, los valores están por debajo de 1 ms.

Por otra parte, la tasa de transferencia disminuye entre los nodos de la red en la medida que la red crece.

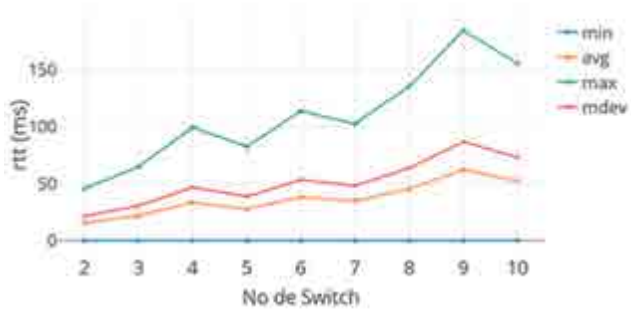

Figura 11. Impacto de la aplicación de seguridad en la latencia entre dos nodos de la red.

Figura 12. Impacto de la aplicación de seguridad en modo reactivo en el ancho de banda entre dos nodos de la red.

Figura 13. Impacto de la aplicación de seguridad en la latencia entre dos nodos de la red en modo proactivo

Figura 14. Impacto de la aplicación de seguridad en modo proactivo en el ancho de banda entre dos nodos de la red

Conclusiones

Las SDN mejoran la seguridad de la red mediante la visión global del estado de la red, donde una amenaza puede resolverse fácilmente desde el plano de control lógicamente centralizado.

La arquitectura SDN permite monitorear activamente el tráfico, así como la detección de amenazas en la red en tiempo real; permitiendo además alterar el comportamiento de la misma en cuestión de milisegundos.

La popularidad de las SDN en el mundo de las telecomunicaciones hoy se demuestra por los numerosos casos de uso que proliferan en esta área. Aunque existen cortafuegos excepcionales en el mercado, es bastante costoso para las empresas instalar numerosos cortafuegos en toda la red para garantizar una alta seguridad. Si se fuera a realizar esta misma implementación en una red tradicional sería necesario configurar cada conmutador por separado con la consiguiente posibilidad de generar problemas en toda la red por errores humanos. El mismo problema se presentaría si se quisieran eliminar o crear nuevos usuarios en las listas de control de acceso.

En las SDN con la separación del control de los elementos de la capa de infraestructura se logra una flexibilidad en la red sin precedentes que crea enormes potencialidades en el campo de la seguridad de las redes como se intentó demostrar en este trabajo.

Las SDN desde una perspectiva de seguridad, tienen una buena característica, que es la conectividad de ruta explícita. En las redes tradicionales, los dispositivos cooperan para establecer rutas entre todos los usuarios de la red. Si algunas de estas rutas representan interacciones no válidas o inseguras, es necesario un cortafuego para bloquearlas. Con las SDN es posible definir solo rutas válidas y otros paquetes simplemente se eliminarían. La aplicación realizada cumple satisfactoriamente con esta funcionalidad y adiciona una nueva función que es la supervisión de amenazas en tiempo real.

En los experimentos realizados se pudo comprobar que la tasa de transferencia disminuye entre los nodos de la red en la medida que la red crece sin importar si la aplicación de seguridad es implementada en los modos proactivo o reactivo.

La aplicación de seguridad, cuando es implementada en el modo proactivo reduce considera-

ISSN: 1813-5056 Vol. 14, No. 1 , enero-julio, pp. 5 - 14, 2018 tono

blemente la latencia en la red y aunque depende del tamaño de la misma, los valores están por debajo de 1 ms. Ocurre lo contrario cuando se emplea el modo reactivo, en el que el retardo puede alcanzar valores superiores a los 150 ms en una red con topología lineal de 10 conmutadores.

La implementación de aplicaciones SDN de seguridad deben instalar reglas de forma proactiva a aquellos servicios que sean sensibles a demoras en la red. Entonces manejar de forma reactiva las tramas que no coincidan con las reglas preestablecidas.

Referencia

Banse, C. y Schuette, J. (2017) A taxonomy based approach for security in software defined networking. IEEE International Conference on Communications (ICC), 1-6

Hu, F., Hao, Q. y Bao, K. (2014) A Survey on Software Defined Network and OpenFlow: From Concept to Implementation. IEEE Communications Surveys and Tutorials. 16(4), 2181-2206

Kreutz, D., Ramos, F. M. V., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S. y Uhlig, S. (2015). Software Defined Networking: A Comprehensive Survey. IEEE 103(1): 14-76.

Li, Y. y Chen, M. (2015) Software Defined Network Function Virtualization: A Survey. IEEE Access 3, 2542-2553.

Marín Muro, Yanko Antonio. (2016). Plataforma de pruebas para evaluar el desempeño de las redes definidas por software basadas en el protocolo OPENFLOW. Villa Clara, Universidad Central Marta Abreu de Las Villas

Nunes, B. A. A., Mendonca, M., Nguyen, X. N., Obraczka, K. y Turletti, T. (2014) A Survey of Software Defined Networking: Past, Present, and Future of Programmable Networks. IEEE Communications Surveys and Tutorials16, 1617-1634

Open Networking Foundation. Software Defined Networking (SDN) Definition