

Nicolas Ricardo Enciso (nicardoe@unal.edu.co),
Ingeniería de sistemas y computación, Facultad de
ingeniería, Universidad Nacional de Colombia, 2017

Software Detector de Ataques ARP Poisoning y Sniffers en Redes WLAN Implementando Machine Learning Supervisado

I. ABSTRACT

En la actualidad, el creciente número de usuarios de dispositivos móviles como smartphones y tabletas ha incrementado el uso de redes inalámbricas wifi, a la vez que ha crecido la cantidad de ataques a estas redes, aprovechando vulnerabilidades propias de los protocolos como los son el ARP y el mismo 802.11n como lo muestra un estudio de Verizon sobre ataques a redes locales. Se propone entonces una herramienta de software capaz de detectar ataques de man in the middle (MITM) como lo son el ARP Poisoning/Spoofing y los sniffers de red que usan NICs en modo monitor. Se genera un algoritmo de machine learning al que se le entrena con datos de redes que están siendo atacadas y neutrales para posteriormente ser capaz de clasificar datos de red entrantes y catalogarlos como alerta de ataque o no.

Palabras Clave—Machine learning supervisado, ataques MITM, tabla ARP, NIC modo monitor, sniffers de paquetes.

II. INTRODUCCIÓN

El creciente uso de redes wifi ha representado de igual manera el aumento de ataques provenientes del uso de redes WLAN, siendo la principal forma de ataques informáticos den el año 2017 [1], por lo que se necesita entonces, una alternativa que ayude a la disminución de esos ataques, enfocados en los de tipo inalámbricos. Adicionalmente, los actuales firewalls y antivirus son incapaces de mantenerse al mismo ritmo de avance respecto a los nacientes nuevos tipos de ataques, añadiendo a la necesidad, la capacidad de adaptación a nuevos tipos de ataques en la herramienta que se propone.

A. Trabajos relacionados

Se han estudiado en varios campos el uso de machine learning en la seguridad. Uno de ellos se enfoca en la detección de intrusiones a partir del análisis de tráfico de red para alertar principalmente sobre malware en dispositivos Android [2], implementando ML, de manera efectiva, con tasas de más del 98% de alertas verdaderas, así como en el análisis de big data, usando framework, en los que se analiza el tráfico con el uso de ML [9]. Los sistemas de detección de intrusiones han ido implementando el uso de ML para dar la posibilidad de adaptación que se necesita, así como también la implementación de análisis de flujos de red, considerando casos en los cuales se pueden detectar hasta 3 tipos de ataques por medio del análisis de machine learning [10]. El análisis de patrones de tráfico en red ha sido de un campo de gran avance en la implementación de sistemas de intrusiones, con las metodologías de Bayes y K vecinos cercanos como formas de clasificación de los datos en atacantes o neutrales [3], también en el campo de Internet Of Things (IoT) en donde sus comunicaciones por redes wifi conceden una contante vulnerabilidad, la cual se ha ido atacando a través del estudio de los datos generados por estos dispositivos [11], así como métodos en WLAN que usan capas de direcciones MAC [13], como en wifi también, se ha implementado el data mining y ML en la búsqueda de detección en vulnerabilidades propias del protocolo de red WLAN más usado, el 802.11 [14]. Con uso de Deep learning, también se ha logrado mejorar los sistemas de detección de intrusiones, usando redes neuronales recurrentes [12], teniendo resultados importantes.

Se muestra entonces que el análisis de tráfico de red es una alternativa posible a la detección de ciertos tipos de intrusiones. En este artículo se enfoca en los ataques que usan las vulnerabilidades propias del protocolo ARP [5-6] y de 802.11n. Con laboratorios de estudio acerca de estos ataques [7], muestra que es posible implementar de manera sencilla detectores de ataques MITM de tipo ARP Poisoning en redes ethernet, además de poder crear redes totalmente virtuales, con el ánimo de tener costos bajos, lo cual representa también la muestra de que es económico tener un ambiente simulado de ataques [8]. En cuanto a la detección de sniffers, se presenta un panorama diferente, debido a la naturaleza de estos ataques, los cuales son silenciosos y en la gran mayoría de casos, indetectables. Un método [4], propone el uso del envío de mensajes de red a todos los posibles host conectados a la red, a la espera de mensajes de respuesta, lo cual confirmaría que una NIC está en modo monitor, clara muestra de la existencia de un sniffer en la red, ya que, el modo monitor es poner en total recepción de paquete a la NIC, sin importar que los paquetes no sean para él, se aprovecha pues, esta metodología de funcionamiento para que la NIC responda algún mensaje sin que sea para él.

¹Nicolas Ricardo Enciso: nicardoe@unal.edu.co, estudiante de Ingeniería - Ingeniería de sistemas y computación, Universidad Nacional de Colombia.

III. MATERIALES

El software se desarrolla en el lenguaje de programación Python 3, debido a su sencillez de uso, compatibilidad con librerías y compatibilidad con múltiples sistemas operativos. Adicionalmente se usa software externo, como lo es NMAP, el cual es compatible con sistemas basados en Linux.

El data set se genera en local, a partir de múltiples simulaciones de red, las cuales son posteriormente etiquetadas, organizadas y almacenadas en archivos de texto plano. El data set se compone entonces de los resultados del análisis hecho en el software, teniendo como resultado las variables estudiadas, así como la etiqueta de la clase a la que pertenece: ataque o neutral.

El software y las pruebas se construyen en el sistema operativo Linux, distribución Kali, debido a que esta se desarrolló con el fin de proveer de una herramienta especializada en la seguridad informática, incluyendo variadas herramientas para auditoría de redes y demás software de seguridad. Por tal motivo el software resultante del presente artículo en su prototipo funciona únicamente en sistemas operativos Kali Linux.

En la infraestructura de red, se usa una simple WLAN, sólo necesario un Access Point (AP) router, y dos hosts: el analizador con software de detección de ataques, y el host atacante. La magnitud de la red afecta de manera directa los tiempos de recopilación de información de red, debido a que se hace para todos los host posibles conectados o no a la red WLAN que se está analizando. El router debe usar el protocolo de comunicación inalámbrica IEEE 802.11n, y usar las bandas de espectro de 2.4ghz.

IV. METODOLOGÍA

El software sigue un desarrollo en dos momentos, ambos embebidos en un único software de extensión py en Python 3, por lo que se tiene una arquitectura monolítica, debido a que no se es muy extenso el software, cada momento se explica a continuación:

A. *Recopilación y análisis datos de red*

La recopilación de información de red comprende el uso de la terminal del sistema, para el requerimiento de información del dispositivo. Primero, se hace selección de la interfaz de red que se está usando, de manera que se obtenga la dirección IP y la máscara de red del dispositivo. Con esta información, se hace llamado al software de pentesting en redes NMAP, usando el script de detección de sniffers [4], el cual hace uso de la IP y máscara de red del dispositivo que está conectado a la red inalámbrica como argumentos. El comando se ejecuta enviando 7 mensajes de red a cada host posible de la red, y espera la respuesta de ellos, si se da, marca un 1, la salida de para cada host será una cadena de longitud 7 caracteres, 1 si se

recibió retorno, o “_” si no se respondió nada. Estas respuestas son tomadas de un archivo de texto el cual guarda los resultados del comando ejecutado.

Posteriormente, se ejecuta la parte de revisión continua de la tabla ARP del dispositivo en busca de cambios, que indiquen spoofing de las direcciones MAC con la IP [7]. Se evalúa múltiples veces con el fin de determinar una medida de casos positivos, para su posterior evaluación y análisis.

Los resultados generales de la evaluación son almacenados en un archivo de salida en texto plano, en la misma ubicación en la que se encuentra el archivo Python.

B. *Análisis de resultados*

El segundo módulo corresponde al de análisis de la recopilación de datos previa. La base está en el algoritmo de machine learning, el cual previamente fue entrenado, es decir, primero se crearon muchas recopilaciones de información, con la diferencia de saber cuáles correspondían a un ataque, y cuales, no, con la idea de desarrollar dos grupos de puntos ubicados en gráfica, los cuales, se agrupan de forma natural en un área especial, mostrando así la tendencia a la clasificación. El entrenamiento es pues, la clasificación de esas áreas que distinguen entre grupos diferentes, en este caso, el grupo de los datos que corresponden a los de un ataque, y el grupo de puntos que corresponde a los de una red normal, neutral. Luego, se hace una clasificación por dos métodos: k vecinos y subdivisión del área. El primero consiste en que, dado un punto, se mide los 3 o 5 puntos más cercanos a este, para hacer una votación de esos puntos más cercanos, sobre cuál es la mayoría representante de grupo de clasificación, el punto entonces es catalogado como de una clase, si la mayoría de sus vecinos pertenecen a esa clase. El segundo usa de modo similar el método anterior, pero, en vez de evaluar un punto, va trazando una línea a medida que va calculando distancias entre los grupos de distintas clases, se hace con una gran cantidad de iteraciones, hasta que se llega a una situación estacionaria, donde la línea deja de moverse por la gráfica, y divide de forma completa a los dos grupos, ayudando así con ambos métodos a clasificar puntos entrantes y finalizar la clasificación.

Cuando se finaliza entonces una ejecución del programa, se recopilan los datos, luego se pasan al algoritmo de machine learning previamente entrenado, para luego dar como salida si se tiene una alerta de ataque o no. Con más datos de prueba a modo de entrenamiento, se tendrá un modo más preciso de evaluar la situación de la red, dando también paso a posteriores mejoramientos del software, como podrían ser la implementación de más variables de análisis, o nuevos métodos de machine learning.

C. *Algoritmo de clasificación de machine learning*

El algoritmo debe responder a la capacidad de poder evaluar por algún criterio de aproximación a datos previamente dados y evaluados en el espacio que se está trabajando. Para ellos se usa un algoritmo sencillo de ML supervisado, como lo es el de

los K vecinos. Este algoritmo necesita de dos variables, dos valores ya que, el algoritmo hace el estudio del criterio de aproximación por lo cercano que esté de un punto en el espacio de dos dimensiones, a través del cálculo de la distancia euclidiana, la cuál es la raíz cuadrada de la sumatoria del cuadrado de la resta de cada punto de la misma variable [15]:

$$d(\vec{x}, \vec{y}) = \|\vec{x} - \vec{y}\| = \sqrt{(x_1 - y_1)^2 + \dots + (x_n - y_n)^2}$$

Fig 1. Ecuación distancia euclidiana en un espacio vectorial
Autoría [15].

El algoritmo en su fase de entrenamiento consiste en la ubicación espacial de los mismos, conociendo la clase a la que pertenece cada punto, por lo cual se puede decir que es supervisado, previamente el algoritmo conoce el tipo al que pertenece cada clase. En el área dimensional, en el caso de la aplicación del software se usan dos variables correspondientes a las dos salidas que dieron los análisis de los datos recopilados previamente, donde cada variable corresponde a un tipo de ataque que se está evaluando. Los datos entonces se traducen en puntos que se ubican en el espacio de dos dimensiones. El concepto de K vecinos usa la idea de que, los puntos que se comportan de una manera similar ubican un área específica del espacio, es decir, los puntos que pertenecen a una misma clase se comportan de manera similar por lo que se van a agrupar en un espacio de la gráfica de dos dimensiones. Con esa idea, entonces se tendrán dos agrupaciones aproximadas en la gráfica, que corresponden a los dos tipos de casos que se pueden tener: neutral o bajo ataque. Una vez entonces los puntos de los casos ya estudiados puestos en la gráfica, se puede decir que el algoritmo ya está entrenado, porque ya tiene bien diferenciados las dos clases que se quieren clasificar.

Posteriormente, cuando se está en ejecución el software, y recopila los datos de la primera fase, la de revisión de tabla ARP y envío de mensajes en busca de sniffers, los resultados son pasados a valores numéricos para que el punto sea puesto en la gráfica de dos dimensiones. Posteriormente se ejecuta K vecinos, donde el punto nuevo mide su distancia euclidiana con todos los puntos que tenga cercanos, tomando el cálculo de la distancia un número impar de vecinos, para posteriormente hacer una votación, en donde se revisa a que clase pertenecen los vecinos que calculó como los más cercanos, y, la clase que tenga la mayor cantidad de representantes, es la que determina la clase del punto que entró como nuevo, por lo que, el punto va a ser clasificado en una clase basado en la proximidad que tenga respecto a sus vecinos, haciendo uso de la idea de que la aproximación del punto con otros puntos corresponde a que los puntos que pertenecen a una misma clase, se comportan de manera similar, por lo que, el punto que corresponde a la actual ejecución del software, va a dar un resultado basado en su

clasificación de si el equipo que hace la ejecución está bajo ataque, o está neutral, es decir, no tiene riesgo. El algoritmo depende en gran medida de los datos de entrada, los de entrenamiento, debido a que basado en ellos, es que hace la clasificación, en ese orden de ideas, deben ser múltiples los datos o casos de entrenamiento, entre más datos de entrenamiento se den, mejor va a ser la clasificación de los puntos entrantes correspondientes a nuevos análisis del software, variando de forma acorde a los previos resultados el valor de los vecinos que van a ser evaluados.

V. DISEÑO

El software se compone de dos módulos fundamentales de recopilación de datos y análisis. El diseño se forma de la entrada de datos, los cuales son el estado de la red, la tabla ARP del dispositivo, y el envío y posible recepción de mensajes a toda la red WLAN en busca de posibles NIC en modo monitor. Luego el análisis compuesto del algoritmo de machine learning que clasifica las variables de los datos dados, para dar como salida una alerta o no al dispositivo.

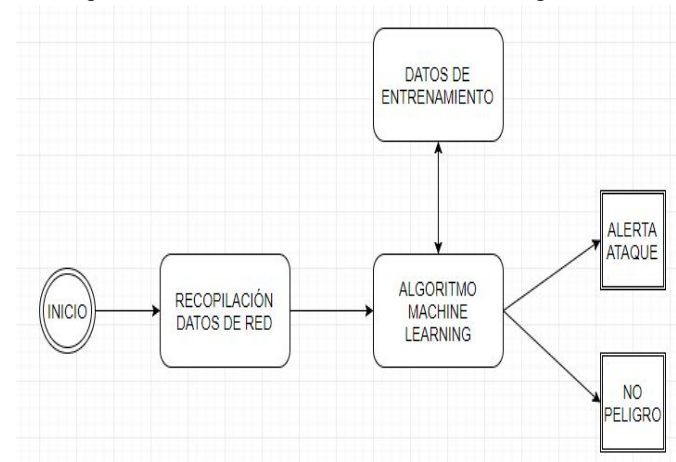


Fig 2. Esquema modelo general funcionamiento de software de detección de ataques. Autoría propia.

Como se puede ver en la Fig1, el diseño del modelo general se basa en recopilar datos y analizarlos con el algoritmo de machine learning para saber la respuesta.

La recopilación de los datos se compone de dos partes, una por cada tipo de ataque: ARP Poisoning/Spoofing y detección de sniffers. La parte ARP, consiste en múltiples revisiones, un número impar de revisiones, con las cuales se determina de forma continua en un rango de tiempo la tabla ARP del dispositivo, comparando de forma constante cada iteración la anterior tabla que se tenía con la actual, en busca de modificaciones que indican posibles cambios por ataque de Poisoning a la tabla ARP, haciéndose pasar un host atacante por el router, interceptando así las comunicaciones. Se hace un número impar de veces para evitar situaciones de empate.

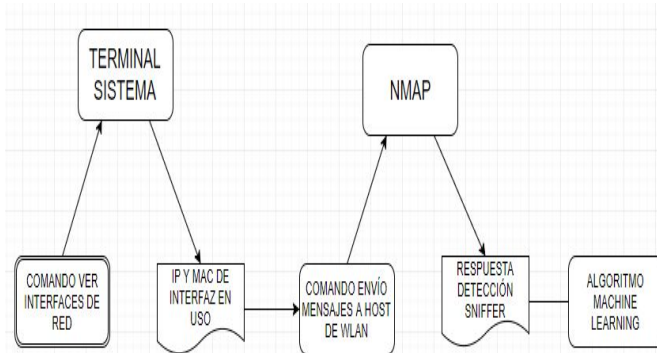


Fig 3. Modelo funcionamiento detector de sniffers en red WLAN. Autoría propia.

El modelo de la Fig2 ejemplifica la recolección de las respuestas de los mensajes enviados a todos los hosts de la red, buscando respuestas por error enviados por las NIC que estén en modo monitor, confirmando posible sniffer en la red. Se inicia con la selección de la interfaz de red con la que se esté conectado a la red que se quiere revisar, dando la posibilidad al usuario de elegirlo manualmente o, que se haga de forma automática, para luego obtener así la IP del dispositivo con su máscara de red, necesarios como argumentos para el comando que envía los mensajes, usando NMAP.

Luego, en cuanto a la revisión de la tabla ARP y los cambios, sólo se hace un llamado a la terminal, la cual muestra la tabla actual ARP, ésta es guardada en un archivo de texto, para posteriormente hacer otra revisión, compararla con la guardada en el texto y luego volver a guardarla, en cada iteración entonces se compara la actual y la anterior.

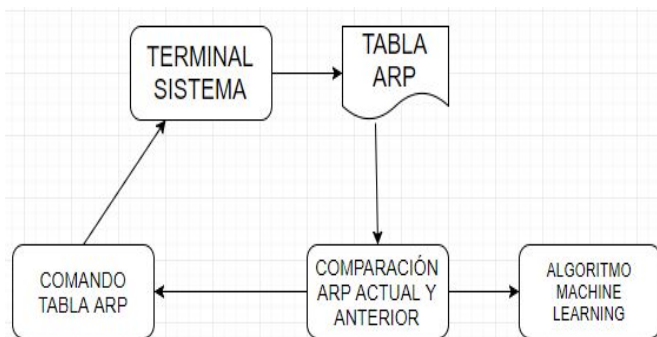


Fig 4. Modelo funcionamiento lectura tabla ARP y comparación constante en busca de cambios. Autoría propia.

Las comparaciones se hacen buscando si para la IP dada anteriormente se mantiene la misma MAC, en cada uno de los dispositivos listados, ya que, un típico ataque ARP, consiste en hacerse pasar por una MAC, la del router por la del dispositivo del atacante.

Finalmente, ambas revisiones se finalizan en un archivo de texto que luego es clasificado por el algoritmo de machine

learning.

VI. IMPLEMENTACIÓN

El software se piensa para funcionamiento en local, alertando los ataques existentes en el dispositivo en el que se ejecuta el software, por lo que, es de margen únicamente local. El dispositivo debe correr Kali Linux como sistema operativo en su versión 4, y contar con Python 3.

El software está pensado para redes WLAN de tamaño medio a pequeño en cantidad de host conectados, debido a que, en la detección de sniffer debido a la necesidad de enviar 7 mensajes a cada host, y esperar su retorno, hace que se tengan tiempos de ejecución muy grandes para grandes cantidades de host, sin la posibilidad de acelerarlo debido a que la interfaz de red sólo puede enviar en un solo nodo los mensajes, no es posible la paralelización. El tiempo normal de ejecución del software no excede los 10 minutos, para una única revisión. Es ideal la constante ejecución en segundo plano, para mantener una revisión de seguridad en el dispositivo.

Futuras mejoras pueden agilizar las revisiones a la vez de tener la posibilidad de concretar y concentrar mejor los resultados de los análisis. La ejecución del software se hace únicamente por consola, por la terminal del sistema, a través del comando de ejecución de archivos en Python3, ya que el software en su prototipo no cuenta con una interfaz gráfica, pero si provee de una comunicación sencilla con el usuario a través de mensajes de estado en la terminal que ejecuta el software, para que el usuario pueda conocer la situación del software y en qué estado de la ejecución se encuentra: recolección de datos, tipo de recolección, analizando clasificación o resultado final.

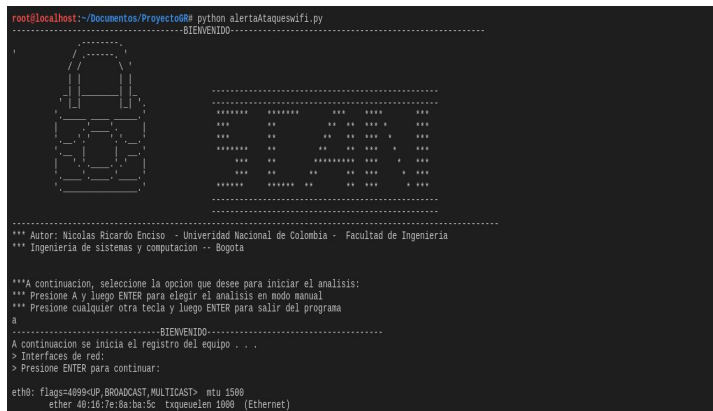
VII. EVALUACIÓN DE RESULTADOS

Luego de varias pruebas consistentes en los dos tipos de casos que se pueden tener: bajo ataque o neutral, se obtuvo para 20 pruebas hechas para cada tipo de caso, se tiene una efectividad de 60%, donde más de 14 casos, fueron analizados por el software de forma correcta, lo que demuestra la efectividad del uso de machine learning y del método básico de clasificación usado, como lo fue el de clasificación por K vecinos. Los casos errados se consolidaron como causales por casos extremos ya sea del caso de ataque o el neutral, es decir, adicional al margen de error del software y sus herramientas asociadas, los datos de entrenamiento juegan un papel fundamental, debido a que son ellos los que posibilitan la forma en la que se clasifican los casos que se dan y por consiguiente la clasificación de alerta que se puede tener cuando el usuario ejecuta el software. Se puede adicionalmente notar que la cantidad de hosts que se pueden tener por el rango de direcciones de la red es directamente proporcional al tiempo requerido por el software para hacer el análisis. Para redes con más de 100 hosts, direcciones posibles

asignables para la dirección de red dada y la máscara de red, los tiempos empiezan a crecer exponencialmente en la finalización del escaneo de la red y su posterior análisis, por lo que el software es capaz de poder resolver problemas de seguridad de intrusiones por medio de sniffer de paquetes y ataques por ARP poisoning de manera efectiva en redes con menos de 100 hosts en su rango de direcciones asignables en la WLAN.

VIII. DISCUSIÓN DE RESULTADOS

El software da sus salidas luego del análisis hecho a través de mensajes de en la terminal donde se ejecuta el software, y por donde se da el diálogo con el usuario. En una primera instancia se da la bienvenida al usuario y se le da una visualización de las interfaces de red que tiene en uso el dispositivo del usuario para que sea él quien introduzca la interfaz actual que tiene en uso



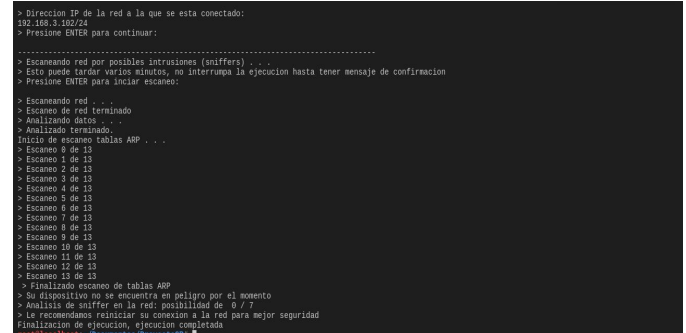
```
root@localhost:~/Documentos/Proyecto08# python alertaAtaqueWifi.py
-----BIENVENIDO-----
*** Autor: Nicolas Ricardo Enciso - Universidad Nacional de Colombia - Facultad de Ingenieria
*** Ingenieria de sistemas y computacion -- Bogota

***A continuacion, seleccione la opcion que desee para iniciar el analisis:
*** Presione A y luego ENTER para elegir el analisis en modo manual
*** Presione cualquier otra tecla y luego ENTER para salir del programa
a
-----BIENVENIDO-----
A continuacion se inicia el registro del equipo . . .
> Interfaces de red:
> Presione ENTER para continuar:

eth0: flags=4099<UP, BROADCAST, MULTICAST> mtu 1500
ether 48:1d:7e:8a:ba:5c: txqueuelen 1000 (Ethernet)
```

Fig 5. Visualización de terminal al ejecutar el software, bienvenida al usuario y menú inicial. Autoría propia

Se le provee al usuario de un menú donde introduce la interfaz, se le proporciona la visualización de la dirección IP y máscara de red que se usará en el análisis, obteniendo como resultados la alerta o no de estar en un posible ataque, se le provee la tasa de posibles intrusiones de sniffer en ambos casos: ataque y neutral. En el caso de ataque, si se tiene un ataque con ARP poisoning se muestra al usuario las MAC de los dispositivos que puede que hayan sido los que están atacando al usuario. Los archivos usados quedan consignados en la misma carpeta donde el usuario tiene guardado el software para permitir la opción de que el usuario pueda revisar la completa ejecución del software.



```
> Dirección IP de la red a la que se está conectado:
192.168.2.202/24
> Presione ENTER para continuar:

-----
> Escaneando red por posibles intrusiones (sniffers) . . .
> Esta puede tardar varios minutos, no interrumpa la ejecución hasta tener mensaje de confirmación
> Presione ENTER para iniciar escaneo:

> Escaneando red
> Escaneo de red terminado
> Analizando datos . . .
> Analizado terminado.

Inicio de escaneo de tablas ARP . . .
> Escaneo 0 de 13
> Escaneo 1 de 13
> Escaneo 2 de 13
> Escaneo 3 de 13
> Escaneo 4 de 13
> Escaneo 5 de 13
> Escaneo 6 de 13
> Escaneo 7 de 13
> Escaneo 8 de 13
> Escaneo 9 de 13
> Escaneo 10 de 13
> Escaneo 11 de 13
> Escaneo 12 de 13
> Escaneo 13 de 13

> Finalizando escaneo de tablas ARP
> No dispositivo no se encuentra en peligro por el momento
> Analisis de sniffer en la red: posibilidad de 0 / 7
> Le recomendamos reiniciar su conexión a la red para mejor seguridad
Finalización de ejecución, ejecución completada
```

Fig 6. Visualización del estado de ejecución del software con un análisis del estado en el que se encuentra. Autoría propia.

El software muestra al usuario de forma adicional una gráfica en tres dimensiones donde se puede observar los datos usados para entrenamiento, siendo los puntos de color rojo los correspondientes a los casos de ataque, y en azul los casos correspondientes a los casos neutrales. El caso actual, es decir el que se está analizando, la situación actual del usuario se muestra en la gráfica en color verde, para que se pueda tener una mejor idea de la proximidad del caso a los que se tienen de entrenamiento y se tenga una mejor dimensionalidad de que tan alejado o cercano se encuentra a alguno de los casos que se tiene para clasificación.

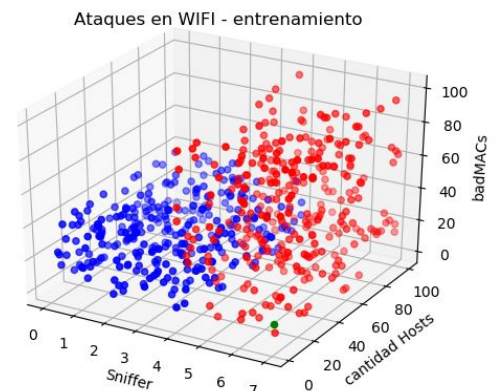


Fig 7. Visualización de los casos de entrenamiento y actual: rojo caso de ataque, azul caso neutral, verde caso actual. Autoría propia.

La clasificación se hace con 7 puntos cercanos con el ánimo de tener una precisión significativa respecto a los datos dados. Con los datos mostrados y el usuario obteniendo el resultado se procede a sugerir reiniciar la conexión al router wifi de manera que ésta es una medida que puede ayudar a que se puedan descartar casos donde el software no haya podido alertar de un ataque.

```

root@localhost:~/Documentos/Proyecto6# python alertaAtaqueswifi.py
> Su dispositivo esta bajo ataque, reinicie la conexion a su red
> Analisis de sniffer en la red: positivo para posible intrusion con grado de 7 / 7
Finalizacion de ejecucion, ejecucion completada

```

Fig 8. Visualización de resultado de uno de los casos probados, donde se alerta de un posible ataque. Autoría propia

IX. CONCLUSIONES

El uso de machine learning para la seguridad informática resulta en una potente herramienta con las capacidades de adaptación y precisión ideales, que pueden ayudar de forma significativa a combatir los ataques en redes inalámbricas wifi. Para los casos de redes muy grandes, se propone continuar con el estudio de la posibilidad de usar el software por módulos o secciones de una misma red, para precisar los análisis y no entrar en tiempos de ejecución muy largos.

Se presenta la posibilidad de aumentar la capacidad de otros sistemas operativos para que hagan uso del software, como posibilidad de mejoramiento del presente producto el artículo: el software.

Adicionalmente, se hace visible la posibilidad de poder adicionar más formas de análisis de red inalámbrica, de manera que ayude a mejorar la precisión de los análisis, y la cobertura de la gran cantidad de tipos de ataques diferentes que existen, además de poder adicionar más módulos que hagan uso de las diferentes alternativas de clasificación y estudio que hacen parte de machine learning y que pueden mejorar de forma importante la cantidad de alertas reales de ataque.

Se tiene como una idea clave la posibilidad de mejorar de manera muy importante la precisión de los datos, usando el mismo software producto del artículo, teniendo un dataset con muchos más casos, es decir, se determina que la cantidad de casos que se estudian, pueden mejorar la precisión del software son la necesidad de añadir módulos al software, debido a que los datos son los que determinan la clasificación hecha, con un dataset mucho más grande y con muchos casos estudiados usados como entrenamiento, precisan mejores resultados en la detección de ataques.

X. REFERENCIAS

- [1] Verizon 2017 Data Breach Investigations Report (DBIR, 2017),
Revisado en October 23, 2017 de
<http://www.verizonenterprise.com/verizon-insights-lab/dbir/2017/>
- [2] Sanjay Kumar, Ari Viinikainen, Timo Hamalainen, « *Machine learning classification model for network based intrusion detection system* », in 11th International conference for internet technology and secured transactions (ICITST), 2016.
- [3] Kriangkrai Limthing, Thidarat Tawsook, « *Network traffic anomaly detection using machine learning approaches* », Computer engineering department Bngkok University, 2015.
- [4] Marek Majkowski, « *Detection of promiscuous nodes using ARP packets* », NMAP documentation.
- [5] Libro: Bob Fleck, Bruce Potter. (2015). *802.11 Security Securing Wireless Networks*. Estados Unidos: O'Reily.
- [6] Libro: Stuart McClure, Joel Scambray, George Kurtz. (2016). *Hacking exposed 7: network security secrets & solutions*. Estados Unidos, New York: Mc Graw Hill.
- Nguyen Thanh Van, Tran Ngoc Thinh, Le Thanh Sach, « *An anomaly-based network intrusion detection system using deep learning* », International conference on system science an engineering (ICSSE), 2017.
- [7] Jinsheng Xu1, Xiaohong Yuan, Anna Yu, Jung Hee Kim, Taehee Kim, Jinghua Zhang, « *Developing and Evaluating a Hands-On Lab for Teaching Local Area Network Vulnerabilities* », Departamento de ciencias de la computación, Universidad del Estado de Carolina del Norte Greensboro A&T, Departamento de ciencias de la computación
- [8] Kazuki Fukuyama, Yoshiaki Taniguchi, Nobukazu Iguchi, « *A Study on Attacker Agent in Virtual Machine-based Network Security Learning System* », Universidad de Kink, Osaka Japón, IEEE 4ta Conferencia de electronica de consume, 2015.
- [9] Pedro Casas, Francesca Soro, Juan Vanerio, Giuseppe Settanni, Alessandro D' Alconzo, « *Network security and anomaly detection with Big-DAMA, a big data analytics framework* », Cloud Networking (CloudNet), 2017 IEEE 16va Conferencia internacional, República Checa, Praga, 2017.
- [10] Eduardo Massato Kakiata, Helton Molina Sapia, Ronaldo Toshiaki Oiakawa, Danillo Roberto Pereira, Joao Paulo Papa, Victor Hugo Costa de Albuquerque, Francisco Assis da Silva, « *Intrusion Detection System Based On Flows Using Machine Learning Algorithms* », IEEE Latin América Transactions, Volúmen 15, Octubre 2017 páginas 1988 – 1993, 2017.
- [11] Muhamad Erza Aminanto, Rakyong Choi, Harry Chandra Tanuwidjaja, Paul D. Yoo, Kwangjo Kim, « *Deep Abstraction and Weighted Feature Selection for Wi-Fi Impersonation Detection* », IEEE Transactions Seguridad y forense de información Volúmen PP, página 1, 2017.
- [12] Yin Chuan-long, Zhu Yue-fei, Fei Jin-long, He Xin-zheng, « *A Deep Learning Approach for Intrusion Detection using Recurrent Neural Networks* », IEEE Access Volúmen PP, página 1, Laboratorio de ingeniería matemática y computación avanzada, Zhengzhou China., 2017.
- [13] Bandar Alotaibi, Khaled Elleithy, « *A majority voting technique for Wireless Intrusion Detection Systems* », Conferencia de tecnología y aplicaciones en sistemas LISAT IEEE Long Island, 2016.
- [14] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, « *Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset* », IEEE Encuestas de comunicaciones y tutoriales volúmen 18, Universidad del Egeo, Samos, Grecia, 2015.
- [15] Facultad de Ciencias UNAM, « *Espacios métricos* », Universidad Nacional Autónoma de México, Facultad de ciencias, área de sistemas, 2017.