UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE CIENCIAS

TESIS

SnorUNI: Aplicación móvil para Sistemas de Detección y Prevención de Intrusiones basados en Snort

PARA OBTENER EL TÍTULO PROFESIONAL DE:

LICENCIADO EN CIENCIA DE LA COMPUTACIÓN

Elaborado por:

Eduardo Yauri Lozano

Asesor:

Mg. Jose Manuel Castillo Cara

LIMA-PERÚ

2017

Agradecimientos

A mis Abuelos, Padres y Hermanos por apoyarme siempre y generar todas las oportunidades de las que disfruto.

Al M.Sc. Manuel Castillo Cara, por el asesoramiento en la elaboración de la Tesis con bastante esmero, paciencia y sabiduría.

Al M.Sc. Alonso Tenorio Trigoso, Director del Centro de Tecnologías de Información y Comunicaciones de la Universidad Nacional de Ingeniería, por darme la oportunidad de desarrollar este Proyecto de Investigación en los laboratorios de la institución.

Resumen

Actualmente, con el constante desarrollo de la Ciencia de la Computación, la seguridad de redes es uno de los problemas más importantes a resolver; el crecimiento de redes y equipos utilizados en casi todas las áreas de trabajo, hace que sean más sensibles a ser vulnerados. Dentro del ámbito de la seguridad, un tema importante es la organización y el fácil acceso a la información sobre posibles alertas por parte de los administradores, lo cual es vital para tomar rápidamente las medidas necesarias. Por otro lado, con el considerable aumento de dispositivos móviles y otras tecnologías, es necesario crear una conexión entre estos dispositivos y los sistemas de seguridad.

La presente tesis, describe la implementación de un aplicativo móvil que permitirá a los administradores de seguridad, obtener información de las alertas de forma fiable y de fácil acceso en sus celulares, tabletas o cualquier dispositivo móvil de bajo costo.

El trabajo consiste en desarrollar la aplicación móvil del administrador incluyendo las vistas básicas y *plugins*, la implementación del Sistema de Detección de Intrusiones (*IDS* por sus siglas en inglés *Intrusion Detection System*) utilizando Snort, la implementación del servicio web para transmisión de datos y, finalmente, el sistema de notificaciones.

Índice general

Índice de figuras

Índice de cuadros

Índice de Acrónimos

IDS	Intrusion Detection System
IPS	Intrusion Prevention System
IDEVAL	Darpa Intrusion Detection Evaluation
IDMEF	Intrusion Detection Message Exchange Format
ID	Integrated Development Enviroment
PHP	PHP: Hypertext Preprocessor
FCM	Firebase Cloud Messaging
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
XMPP	Extensible Messaging and Presence Protocol
API	Application Programming Interface
IP	Internet Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
ICMP	Internet Control Message Protocol
GPS	Global Positioning System

Capítulo 1

Introducción

En este capítulo, se presentan las motivaciones para el desarrollo del presente trabajo y se expone el objetivo principal y los objetivos especificos. Finalmente, se describe de forma resumida la estructura de la tesis.

1.1. Motivación

Hoy en día la seguridad de redes es uno de los problemas más importantes a resolver, el crecimiento de las redes y equipos usados en casi todas las áreas de trabajo, hace que sean más sensibles y propensos a ser vulnerados. La *Ciberdelincuencia* han ido en constante aumento en casi todas sus áreas, para noviembre del 2016 son aproximadamente el 80 % del total de ataques informáticos existentes [1]. Uno de los principales y más comunes es el sabotaje informático, cuyo objetivo es ingresar a los ordenadores y/o servidores con el fin de modificar o destruir información valiosa, perjudicando a instituciones, organizaciones o empresas; otro delito importante es el robo de identidad mediante el acceso a las cuentas de usuario, siendo la información financiera una de las más perjudicadas y comúnmente vulnerada.

Para realizar los ataques informáticos existe múltiples métodos, particularmente en el año 2016, los métodos más comunes fueron [2]:

- Ataques a las aplicaciones web (*SQL inyeccion, Cross-site Scripting*).

- *Malwares* (*virus, troyanos*, etc).
- Ataques hacia aplicaciones específicas (captura de paquetes de información utilizados por la aplicación a través de una vulnerabilidad específica).
- Ataques *DoS* y *DDoS* (sobrecarga a un sistema utilizando uno o varios servidores denominados *bots*)
- *Reconnaissance* (obtener información de la víctima con interacción directa o indirecta de su sistema).

La mayoría de estos ataques pueden ser mitigados o por lo menos minimizados implementando adecuadamente un *IDS*, actualmente existen muchos en el mercado, algunos privativos y otros *open source*. En esta última categoría destaca Snort, ampliamente utilizado por su robustez y capacidad de adicionar *plugins*.

Otro problema importante a resolver es la necesidad de una adecuada organización y fácil acceso a la información sobre posibles alertas por parte de los administradores, lo cual es vital para tomar rápidamente las medidas necesarias para una respuesta. Dado el aumento de dispositivos móviles y el uso masivo de aplicaciones para estos dispositivos, es necesario crear una conexión entre estos dispositivos y los sistemas de seguridad para mejorar la rapidez de acceso a la información.

La situación actual de la seguridad de redes y el problema de acceso a la información, han motivado el desarrollo de una aplicación móvil para la administración de *IDS* basados en Snort. Este aplicativo permitirá la conexión con los servidores para la extracción remota de información de forma rápida y eficiente, permitiendo además informar al administrador sobre nuevas alertas mediante un sistema de notificaciones. De esa forma los usuarios podrán contar con información clave para la toma de decisiones en cualquier lugar y momento.

1.2. Objetivos

El objetivo principal de este trabajo, es el desarrollo de una aplicación móvil en Android para la administración de un sistema de Detección y Prevención de Intrusiones implementado con Snort, el cual será liberado como un *plugin*.

Adicionalmente, este trabajo tiene los siguientes objetivos específicos:

- Instalación y configuración de un *IDS* utilizando Snort.
- Instalar y configurar *plugins* desarrollados por la comunidad de desarrolladores acorde a una institución académica.
- Mejorar la aplicación Android utilizando Material Design Library.
- Implementar el servicio web para la transmisión de información.
- Realizar una conexión web segura y encriptada para el servicio web.

Y los objetivos con respecto a las competencias académicas desplegadas en el trabajo son:

- Uso de *software libre* para la implementación integral de todo el sistema y aplicación móvil.
- Empleo de una metodología para el desarrollo de *software*.
- Administración de base de datos.
- Uso de ingeniería inversa.

1.3. Estructura de la Tesis

Para brindrar al lector una idea global del contenido de este trabajo, a continuación se hace una breve descripción del propósito de cada capítulo en la presente tesis.

- **Capítulo 1: Introducción**

  En este punto, se comenta la motivación para el desarrollo del trabajo y se definen el objetivo general y los objetivos específicos en el ámbito académico y el desarrollo del sistema. Adicionalmente, se describe de forma resumida la estructura de la tesis.

- **Capítulo 2: Estado del arte**

  En este capítulo, se realizará un análisis de trabajos relacionados a Snort, dispositivos móviles, *plugins* y otras herramientas, los cuales nos servirán como punto de partida y contribuirán al desarrollo de este trabajo.

- **Capítulo 3: Sistemas de Detección y Prevención de Intrusiones**

  En esta sección, se estudian los conceptos y definiciones básicas en seguridad. También se analizarán los conceptos relacionados a Sistemas de Detección de Intrusiones y Sistemas de Prevención de Intrusiones (*IPS* por sus siglas en ingles *Intrusion Prevention System*).

- **Capítulo 4: Metodología y herramientas**

  En este punto, se describe la metodología utilizada y las principales herramientas utilizadas para el desarrollo de esta tesis.

- **Capítulo 5: Diseño y estructura de SnorUNI**

  En esta sección, se describen las configuraciones y programas desarrollados para la conexión entre el servidor Snort y la aplicación móvil.

- **Capítulo 6: Casos de estudio**

  En este capítulo, se describirán detalladamente todos los casos de estudio de la aplicación SnorUNI, incluyendo sus principales funcionalidades.

- **Capítulo 7: Conclusiones y trabajo a futuro**

  En esta sección, se presentan las conclusiones obtenidas de este trabajo, además en base a ellas se proponen nuevos posibles trabajos a futuro.

- **Apéndice A: Modos de ejecución y componentes importantes de Snort**

  En este apéndice se estudia los modos de ejecución de Snort y se describirán en detalle sus principales *plugins*.

- **Apéndice B: Sprint Backlogs**

  En el apéndice B, se incluyen los Sprint Backlogs empleados para el desarrollo del sistema y aplicación móvil.

- **Apéndice C: Manual de instalación y configuración**

  En esta sección se describe detalladamente los pasos de instalación y configuración del sistema SnorUNI.

Capítulo 2

Estado del arte

En esta sección se realizará un estudio de los trabajos desarrollados previos al presente proyecto con el fin de mejorar y contrastar las ideas y objetivos planteados inicialmente. Además, este estudio nos permitirá definir funcionalidades y/o características importantes a incluir en el presente trabajo.

2.1. Swinedroid: Aplicacion móvil para Snort

Es una aplicación Android para monitoreo de eventos registrados por Snort, fue desarrollado por *William Budington* en el año 2011 y desde entonces ha tenido actualizaciones periódicas hasta el año 2015, fue uno de los primeros intentos de desarrollo de herramientas para Snort en el área de dispositivos móviles. Tiene las siguientes funcionalidades:[3]

- Vista de las últimas alertas del sistema.
- Barra de progreso indicando el número de alertas.
- Buscador de eventos.
- Estadísticas generales de ataques.

En la figura 2.1 se observa la vista principal de Swinedroid. En esta vista se observan los nombres de las alertas, incluyendo la fecha de registro y las *IPs* de origen y destino.

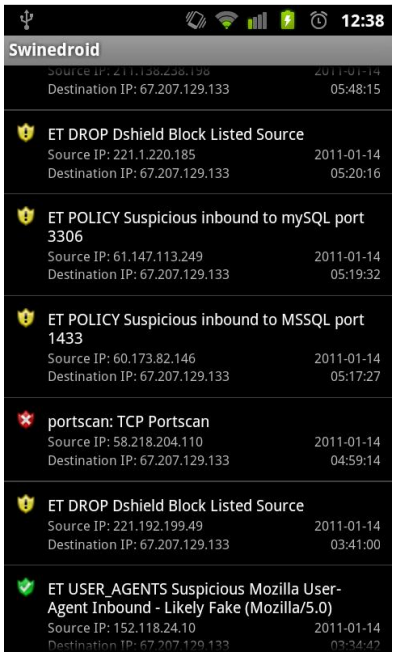

FIGURA 2.1: Swinedroid. Fuente: Androidir [4]

2.2. Sistemas de Detección de Intrusiones para Smartphones basadas en arquitectura Cloud

Actualmente el uso *Smartphones* se ha incrementado notablemente, dada su gran capacidad para realizar múltiples tareas similar a una computadora de escritorio; los usuarios pueden guardar información, navegar por internet, enviar mensajes mediante distintas aplicaciones, comprar o hacer pagos online, etc. Sin embargo, realizar estas operaciones también involucra introducir contraseñas y/o usar información de las tarjetas de crédito, esto hace que sean más comunes los ataques a estos dispositivos con el fin de extraer dichos datos. Además, recientes artículos informan sobre el aumento de *virus* y *malwares* para los sistemas operativos de los *Smartphones* (Android y iOS), estos pueden introducirse al navegar, enviar mensajes y principalmente descargar archivos no seguros de internet.[5]

Una solución a los ataques en *Smartphones* es la implementación de un *IDS*, sin embargo, si bien el dispositivo móvil tiene varias funcionalidades, no cuenta

con suficiente poder de procesamiento ni tampoco con una adecuada capacidad de almacenamiento. Es por ello que las nuevas investigaciones se han centrado en la creación de una arquitectura en la nube para realizar el procesamiento del *IDS* y conectarlo a estos dispositivos para intercambiar información.

Entre las principales investigaciones destaca la arquitectura desarrollada por los investigadores *Manish Kumar* y *M. Hanumanthappa* [5], que proponen construir un sistema compuesto por un agente instalado en el dispositivo que registra sus características y en base a ellas es emulado en una máquina virtual en la nube. Dada la gran capacidad de estos servicios es posible emular varios dispositivos a la vez, se instalará Snort como motor principal del *IDS* implementado con las reglas para monitoreo de dispositivos móviles y, finalmente, se incluye un *proxy* para replicar todo el tráfico entrante y saliente del dispositivo físico y dirigirlo a la plataforma en la nube para su análisis.[5]

Otra investigación un poco más elaborada en este campo es **Secloud**, desarrollado por *Saman Zonouz*, *Amir Houmansadr*, *Robin Berthier*, *Nikita Borisov* y *William Sanders* en la Universidad de Illinois [6]. De forma similar al trabajo anterior, Secloud tiene un agente para recolección de datos del móvil físico, el cual emula en la nube y allí analiza el tráfico en busca de eventos, una de sus principales características es que incluye un conjunto de mecanismos para notificar al dispositivo la detección de un ataque:

- **Informe directo al agente:** la forma más rápida de notificar un ataque es a través del agente en el dispositivo, el cual puede notificar al usuario mediante un *pop-up*, e inclusive puede realizar algunas acciones para mitigar el ataque.

- **Notificación por email:** utilizar otro medio externo por ejemplo Gmail, es un forma segura de enviar la alerta desde el emulador al dispositivo infectado, sin que el mensaje sea comprometido en el ataque.

- **Respuesta y prevención:** con la finalidad de mitigar lo más pronto posible un ataque, el agente tiene que realizar una de las siguientes acciones: remover archivos o aplicaciones maliciosas, eliminar un proceso

correspondiente a una aplicación maliciosa, generación de *backups* periódicamente con el fin de salvaguardar la información del dispositivo.

En la figura 2.2, se puede observar a los componentes que integran el sistema Secloud. El tráfico de red dirigido al dispositivo móvil, es direccionado a través de un *proxy* hacia la máquina virtual. Toda la información es analizada por el *IDS* instalado en el mismo servidor. Si se detecta una anemanza el sistema enviará una alerta a dispositivo móvil original, permitiendole al usuario tomar las medidas necesarias.

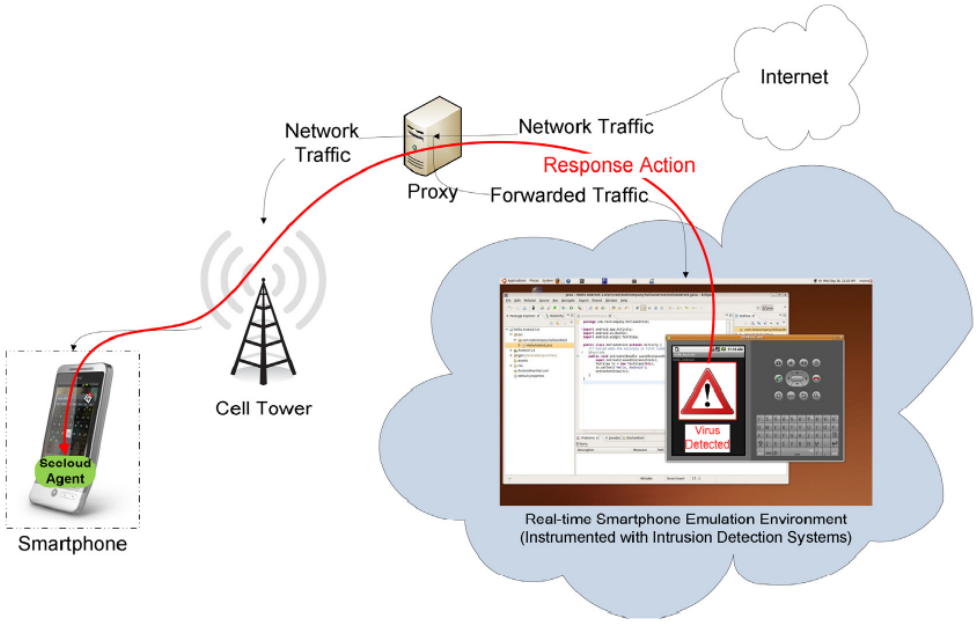

FIGURA 2.2: Arquitectura Secloud. Fuente: Computer and Security 37. [6]

2.3. Análisis de rendimiento de un IDS basado en Snort

En el trabajo realizado por *Akash Garp* y *Prachi Maheshwari*, se hace un estudio de la relación entre las detecciones reales y las falsas alarmas emitidas por Snort al utilizar la base de datos de ataque para sistemas de seguridad *Darpa Intrusion Detection Evaluation* (*IDEVAL*); además hace una

importante comparativa entre los *IDS* basados en firmas y los *IDS* basados en Anomalías.[7]

Al realizar las pruebas de detección de Snort utilizando *IDEVAL*, se observó que el número de falsas alarmas comparado al número de detecciones reales es relativamente bajo, de esa forma se prueba la alta capacidad de Snort para detectar ataques. Una recomendación para prevenir los errores ante nuevos ataques es mantener las reglas actualizadas constantemente.

En la figura 2.3, se puede observar que de 5000 ataques detectados, aproximadamente se generan entre 40 y 45 falsas alarmas. La proporción es bastante baja, esto prueba el alto rendimiento y eficacia de Snort.

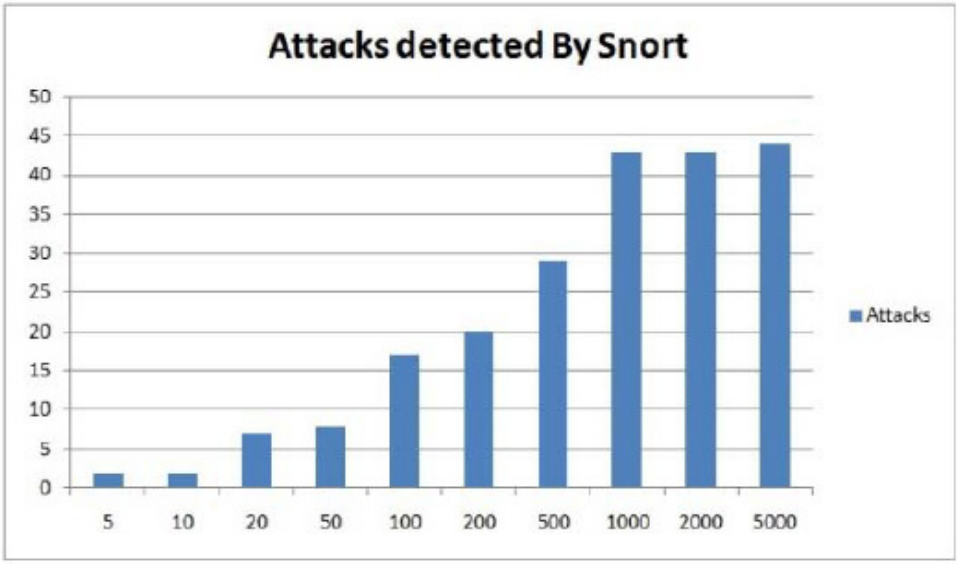

FIGURA 2.3: Gráfico de falsas alarmas vs ataques detectadas por Snort en el dataset IDEVAL. Fuente: ICACCS 2016 [7]

Existen dos tipos principales de *IDS*, los basados en firmas y los basados en anomalías, los *IDS* del primer tipo utilizan firmas (patrones de ataques) almacenadas previamente en su base de datos, es por ello que este tipo de *IDS* es altamente confiable en la detección de ataques conocidos, sin embargo produce un alto índice de falsas alarmas en el caso de nuevos ataques (no registrados en su base de datos). También se tiene a los *IDS* del segundo tipo basados en anomalías, los cuales analizan comportamientos extraños en

el tráfico y rendimiento de la red (ancho de banda usado, protocolos, puertos y dispositivos que generalmente se interconectan) para detectar posibles ataques.

En el cuadro 2.1 puede verse las principales ventajas y desventajas de la ejecución de ambos tipos de *IDS*, dependiendo del escenario en el cual se utilizan.

Tipo de IDS	**Ventajas**	**Desventajas**
IDS basados en firmas	Alta eficacia en detección de ataques conocidos. Genera baja tasa de falsos positivos en ataques conocidos. Capacidad de personalizar reglas dependiendo del entorno.	Solo puede detectar ataques previamente registrados en sus firmas. La información del ataque depende del sistema operativo y programa en uso.
IDS basados en anomalías	Eficaz para detectar ataques nuevos. Pueden ser utilizados para obtener información para las firmas del otro tipo de *IDS*.	Tendencia a generar alto ratio de falsas alarmas. No define la naturaleza del ataque. Baja cantidad de detecciones en situaciones normales.

CUADRO 2.1: Comparación entre IDS basados en firmas y anomalías. Fuente: Rendimiento de un IDS basado en Snort. [7]

2.4. Implementación general de Plugins para Snort

Desde que se creó Snort, se han implementado nuevos proyectos para el desarrollo de software complementario con el fin de mejorar sus funcionalidades, estos nuevos programas son los denominados *plugins* de Snort. Actualmente existen múltiples tipos dependiendo de las mejoras que brindarán a Snort en su funcionamiento.

Entre los principales plugins desarrollados destacan: **Barnyard2** [8], interprete que permite leer los archivos binarios de salida y almacenarlos en una base de datos Mysql, SqlServer u Oracle; **BASE** [9], interfaz web desarrollada en *PHP* que permite acceder y gestionar de una forma fácil la información generada por los *IDS* almacenada en base de datos; **SnoGE** [10], aplicación para graficar las localizaciones geográficas de las *IPs* atacadas y vulneradas, procesa de salida de Snort con formato *unified*, para luego representarlos en Google Earth; **Sguil** [11], interfaz gráfica similar a BASE incluyendo además herramientas para analítica.

En el apéndice A se describirán más a detalle estos *plugins*, sin embargo, son mencionados primero en el estado del arte ya que serán tomados como guía para desarrollar la aplicación móvil que al final será liberado como *plugin* para Snort.

En la figura 2.4, se observa el diagrama general de la interacción entre los *plugins* y Snort. La información recolectada por Snort es guardada en *logs* binarios, los cuales pueden ser utilizados por SnoGE o procesados por Barnyard2 para guardar la información en base de datos. Finalmente, BASE consume la información de la base de datos y la muestra al usuario.

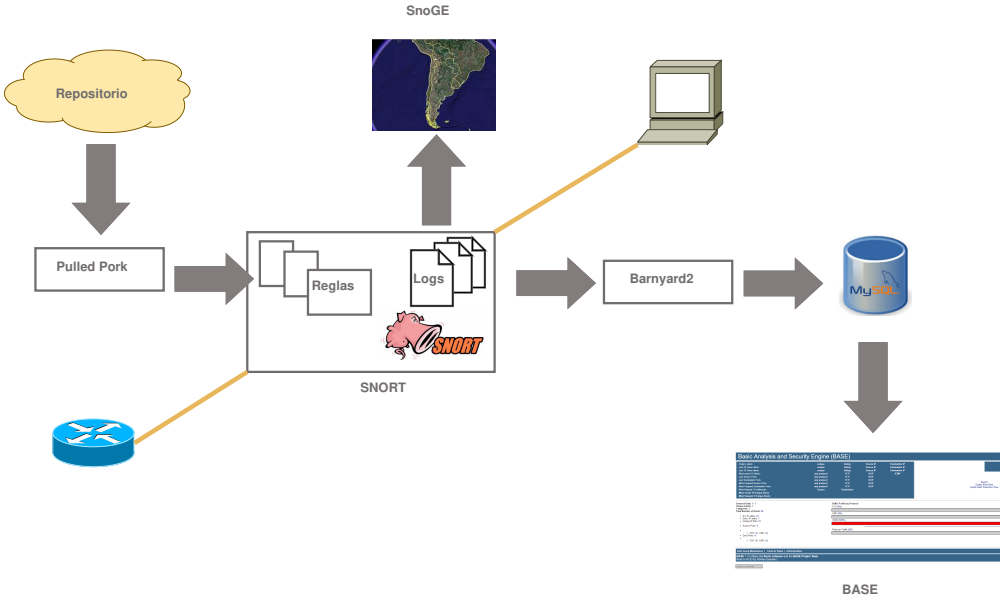

FIGURA 2.4: Snort y plugins en funcionamiento. Fuente: Elaboración propia.

2.5. Conclusiones

Se describió a Swinedroid, uno de los primeros intentos para desarrollar aplicaciones móviles de administración para Snort, el cual cuenta con funcionalidades básicas para mostrar eventos. Se observa que además de Swinedroid, no se han desarrollado más aplicaciones conocidas de este tipo.

Se estudiaron algunos trabajos enfocados a dispositivos móviles y Snort, se puede observar que actualmente han aumentado las investigaciones en desarrollo de *IDS* para *Smartphones*; dado que estos dispositivos tienen limitado poder de procesamiento y capacidad de almacenamiento, se ha pensado emplear la nube para implementar el *IDS*, emular los dispositivos e intercambiar información para el análisis, esto permitirá alertar al usuario ante una posible intrusión directamente al dispositivo; en esta sección se estudiaron algunas arquitecturas de este tipo. Un trabajo más enfocado en Snort, fue el análisis de rendimiento para detección utilizado el data set *IDEVA*L, los resultados muestran la gran efectividad de Snort para detectar intrusiones si es que tiene su base de firmas correctamente actualizada.

Tomando como referencia estos trabajos y los diversos *plugins* ya desarrollados, el proyecto actual busca implementar una aplicación móvil que reciba notificaciones de alertas, muestre información detallada de eventos e implemente las principales funcionalidades de estos *plugins* en versión móvil. La mayoría de los *plugins* han sido creados como aplicaciones de escritorio, por tanto el acceso a esa información puede ser limitado y lento, sin embargo, en versión móvil será más fácil para el usuario ver los reportes y tomar decisiones rápidas. Las funcionalidades que se incluirán en la aplicación serán:

- Vista de eventos.
- Vista de información de protocolos.
- Cuadro de Indicadores.
- Grafica de *IP*s involucradas en los eventos.
- Herramientas para el manejo de eventos (buscador y ordenador).
- Servicio de notificaciones de alertas.
- Manejo de múltiples sesiones.

Capítulo 3

Sistemas de Detección de Intrusiones

En esta sección se estudiarán los conceptos más importantes relacionados a la seguridad de redes, Sistemas de Detección y Prevención de Intrusiones. Se describirán los tipos de IDS existentes y sus caracteristicas. Finalmente, se realizará un estudio de los componentes principales de Snort.

3.1. Conceptos y definiciones de Seguridad en Redes de Computadoras

3.1.1. Qué son los Sistemas de Detección de Intrusiones

Se entiende como *IDS* aquel sistema que se encuentra constantemente monitoreando recursos computacionales; los cuales pueden abarcar desde recolectar y analizar información procedente desde simples computadoras, hasta sistemas de redes muy complejas. El objetivo principal de los *IDS* es detectar o descubrir cualquier tipo de actividad sospechosa que conlleva al intento o realización de una intrusión al sistema de algún individuo no autorizado.[12]

Los *IDS* por medio del monitoreo identifican evidencia o manifestación de las intrusiones, para realizar esta labor necesitan información completa y sobre

todo confiable del sistema al cual están analizando. El desarrollo de los *IDS* tienen como aspiraciones principales:

- Presentar el análisis del sistema de una manera simple y fácil de entender por el usuario de éste.
- Detectar a las intrusiones de la manera más rápida posible. Lo ideal es que la respuesta sea en tiempo real de manera que se pueda responder inmediatamente a la intrusión, aunque para ello se debe de superar algunos problemas de retraso, por ejemplo a la hora de analizar comandos. Por ello, actualmente es permisible que la respuesta sea en pocos minutos o en todo caso pocas horas desde que se realizó el ataque.
- Mejorar la precisión de los resultados, evitando así dar información falsa como resultado del monitoreo, tanto positiva como negativa.
- Detectar la mayor cantidad de intrusiones, las cuales pueden ser tanto conocidas o desconocidas, ya que mediante algún proceso de aprendizaje se puede detectar nuevos tipos de intrusión o cambios que se puedan dar en los ya existentes.

Gracias a su naturaleza intrínseca de monitoreo de recursos, puede ser aplicado tanto a redes de computadoras (*IDS* basados en redes) o a computadoras personales (*IDS* basados en máquina). En el caso de computadoras, se realiza un análisis a nivel de sistema operativo obteniendo información sobre control de usuarios, condición actual de archivos importantes del sistema y uso de recursos hardware. Si el *IDS* abarca toda la red, existe mayor cantidad de recursos con la posibilidad de ser analizados, por ejemplo acceso y salida a la red desde direcciones peligrosas, ancho de banda, etc. [13]

En la figura 3.1, se puede observar la instalación del *IDS* en red. Dependiendo de la posición en donde se instale (delante o detrás del *firewall*), la información recolectada y el procesamiento serán diferentes.

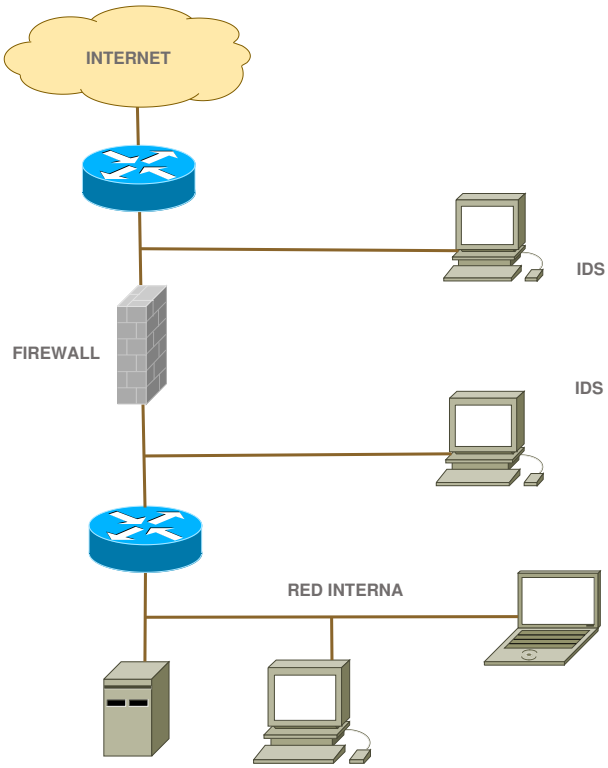

FIGURA 3.1: Ubicación de los IDS en la red. Fuente: Mural. [14]

3.1.2. Qué es un Sistema de Prevención de Intrusiones

Actualmente existe una gran variedad de *IDS*, cuyas características son muy variadas gracias a su gran desarrollo durante los últimos años (inicios en los años 70). Sin embargo se sabe que estos sistemas tienen algunas limitaciones, entre ellas, la principal es que estos sistemas no paran los ataques de los intrusos. Su funcionamiento consta en detectar posibles intrusiones y enviar las alertas respectivas al administrador del sistema. Es por ello que se pensó en el desarrollo de otra herramienta, que además de detectar intrusiones, pueda hacerlas frente previniéndolas, desarrollando los Sistemas de Prevención de Intrusiones (*IPS*).

Se define *IPS* como aquel sistema cuya función es detener los ataques antes de que estos ingresen a los sistemas de redes causando daños a las aplicaciones o infraestructura, además de ello los ataques son detectados y bloqueados sin tener algún tipo de impacto en el rendimiento de las redes.[15]

A continuación se definen algunas de las características principales de los

IPS mencionadas en [15], algunas de las cuales hacen la diferencia con los *IDS*:

- Los *IPS* no deben influir en el rendimiento de la red a la cual han sido conectados, de esta manera no demandará ningún gasto adicional, ni tiempo extra de funcionamiento.
- La arquitectura es 100 % *IPS*, no tiene como referente a otras arquitecturas como las de los *IDS*.
- Deben contar con varias técnicas y tecnologías de detección como:
  - Filtros de vulnerabilidad
  - Capacidad de detectar y bloquear ataques *DoS*.
  - La cantidad de amenazas no debe afectar en el rendimiento.

Al igual que los *IDS* se podían clasificar en sistemas de detección de intrusiones basados en host (*HIDS*) y sistemas de detección de intrusiones basados en red (*NIDS*), ello en base a la fuente de datos para su funcionamiento. Los *IPS* también pueden ser clasificados de manera similar en base a su ubicación en la red, estos dos tipos son *NIPS* y *HIPS*, a continuación una breve descripción de ellos:

- **NPIS.-** Estos dispositivos se encuentran en posiciones estratégicas de la red monitorizando toda la información que pasa a través de ella, cuenta con un sistema independiente de la red la cual monitorea ya que tiene sus propias unidades de almacenamiento y procesamiento, entre otros. Actualmente es el más usado en los sistemas de redes modernas.
- **HIPS.-** Estos son instalados en cada uno de los componentes de la red ya sean computadoras, servidores u otros. Su funcionamiento es un poco más dificultoso ya que se tiene que instalar en cada *host* y luego ser controlado mediante una consola única, generalmente esta junto con el antivirus de la máquina. Este tipo de *IPS* no son muy comunes en las redes modernas.

El *IPS*, utiliza técnicas y métodos específicos implementados en su motor central para poder detectar y responder a los intrusos de red, las principales son:[16]

- **Basados en firmas:** los fabricantes del *IPS* desarrollan firmas incluidas en el producto, en base a estas firmas el motor realiza comparaciones para poder identificar posibles daños o modificaciones de protocolos o aplicaciones.
- **Detección de anormalidades:** se crea una norma base de funcionamiento bajo la cual trabaja la red. El motor se centra en verificar que esta norma no sea alterada por los componentes del sistema, lo cual supondría una alerta.
- **Inspección de paquetes y filtros de tráficos:** El motor analiza los protocolos de red, para verificar que no existan anormalidades, su forma de funcionamiento es similar al *firewall*.

En la figura 3.2, se observa la instalación del *IPS* antes del *firewall*. Al detectar posibles amenazas, el *IPS* tomará las acciones correspondientes para repeler el ataque (por ejemplo, bloquear el tráfico entrante). De esta forma el sistema tiene doble seguridad, incluyendo el firewall como último recurso para repeler intrusiones.

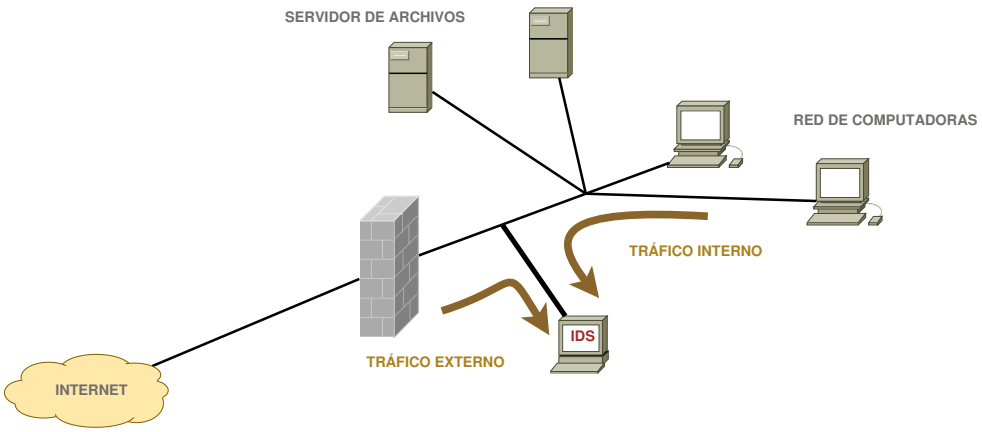

FIGURA 3.2: Ubicación y trabajo de monitoreo de un IPS. Fuente: Virusinformatico. [17]

3.2. SNORT

Snort es una de las herramientas más completas utilizadas en la creación Sistemas de Detección y Prevención de Intrusiones. Con esta herramienta, se pueden crear desde pequeños monitores de paquetes hasta grandes y complejos *IDS*. Snort tiene una gran capacidad de captura de paquetes *TCP/IP*, a partir de los cuales se puede detectar tráfico malicioso y generar alertas en tiempo real. Además, puede ser usado como parte de otros Software o sistemas de seguridad más complejos.[18]

Tal y como lo menciona el autor de esta herramienta, Snort funciona como una aspiradora de datagramas *IP* (de ahí proviene su nombre) y ofrece una gran cantidad de posibilidades para su tratamiento. Por su forma de funcionamiento como *sniffer* o monitor de paquetes en red, Snort es definido como un *NIDS* ligero, basado en reconocimiento de patrones. Además, desde el punto de vista del motor de detección, Snort es calificado como un detector de intrusiones basado en usos indebidos, debido a que contrasta el tráfico capturado con las reglas de detección previamente definidas.[18]

El uso de Snort se ha incrementado considerablemente, ello debido a que, además de ser de código abierto con licencia GLP Linux puede ser instalado en una gran variedad de sistemas operativos libres (GNU/Linux, Fedora, NetBSD) y privativos (Windows). Por otra parte, actualmente existen gran cantidad de accesorios para Snort, soporte a base de datos y nuevos *pre-procesadores*.[19]

3.2.1. Elementos que componen la arquitectura de Snort

Snort está formado por varios componentes que en conjunto, definen la arquitectura software, en su mayoría estos componentes son *plugins* que permiten configurar y personalizar a Snort de la mejor forma. A continuación se describen los componentes principales:

- **Captura de datos:** el paso inicial básico para que se inicie todo el proceso de detección y prevención de intrusiones, es la captura de datos (paquetes que están circulando por la red) de manera similar a los *sniffing* de red.

Para ello, utiliza librerías específicas, entre ellas una de las más usadas es *libcap*, debido a su facilidad de ejecución en todos los sistemas operativos y hardware.

- **Decodificador:** el motor de decodificación es utilizado para descifrar cada uno de los datos relacionados a los protocolos que vienen en cada uno de los paquetes. Se sigue un proceso de descifrado de cada protocolo desde el nivel más bajo, siendo el paquetes almacenado finalmente en una estructura de datos en base al cual trabajarán los demás componentes de Snort.

- **Módulo del preprocesador y preprocesadores:** el preprocesador de Snort se encarga de realizar una ordenación y clasificación de todos los paquetes que son capturados para posteriormente ser analizados por el motor de detección [19]. Cuando los paquetes viajan a través de la red de un destino a otro, generalmente están desordenados y es el destino final en donde se realiza el ordenamiento para poder interpretar la información que contienen. Este procedimiento también es realizado por Snort al momento de analizar los paquetes, para ello utiliza el preprocesador. Cada preprocesador es un *plugin*, puede ser instalado dependiendo de los requerimientos del sistema o a criterio de su administrador. A continuación se muestran los más importantes:

  - *SMTP*.
  - *Stream5*.
  - *Flow*.

- **Reglas y formatos:** las reglas, o también denominadas firmas, son los patrones que el motor de detecciones busca en cada uno de los paquetes analizados, si alguna de estas firmas es detectada, se emite una alerta sobre una posible amenaza [18]. Existen 4 categorías de reglas:

  - Reglas de Protocolo: variables dependiendo el protocolo a ser analizado.

- Reglas de contenido genéricas: utilizadas para buscar patrones binarios o *ASCII* dentro del campo *contenido* de los paquetes.
- Reglas de patrones mal formados: utilizados para búsqueda de patrones en las cabeceras de los paquetes.
- Reglas *IP*: utilizadas para el análisis de datagramas *IP*.

- **El motor de detección:** el motor es la parte central y la más importante de Snort y se encarga de procesar cada uno de los paquetes que fueron capturados de la red con el objetivo de encontrar actividad sospechosa en la misma. El procesamiento consiste en comparar los patrones de cadenas de información establecidos en las reglas con la información de los paquetes. Si existe coincidencia, se emitirá una alerta, de lo contrario el paquete es descartado.[18]

- **Extensiones de salida:** los módulos de salida de Snort aplican diferentes operaciones según la manera que se desea almacenar la información de salida. Dependiendo del formato en que se deseen obtener los datos se pueden utilizar una gran variedad de módulos, por ejemplo, *SYSLOG*, *Database* y uno de los más usados para transferencia de datos *Unified*.[18]

En la figura 3.3 se observa la arquitectura básica de Snort. Los componentes de la arquitectura están interconectados y procesan la información secuencialmente. El principal componente es el motor de detección, dado que utiliza las reglas para buscar posibles intrusiones dentro de la información recolectada.

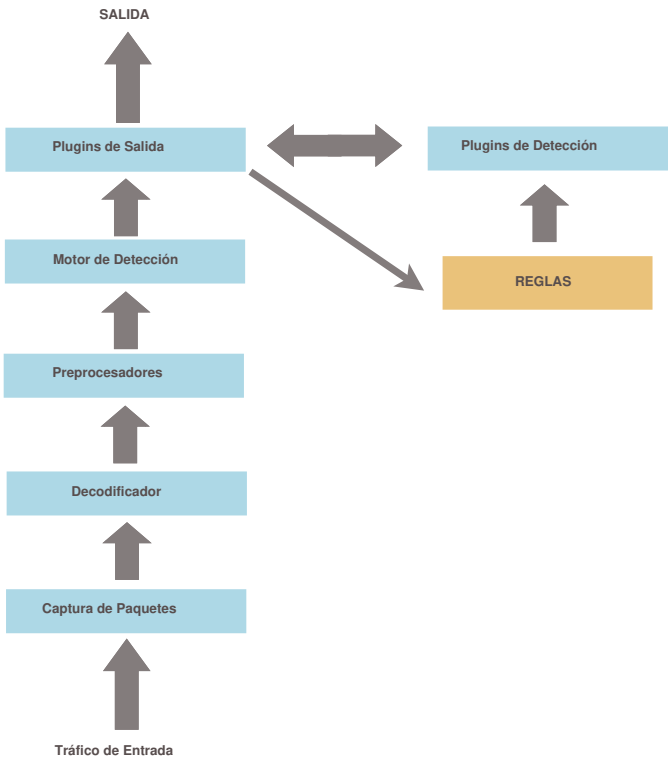

FIGURA 3.3: Componentes de la Arquitectura de Snort. Fuente: Snortudenar. [19]

3.2.2. Posibilidades de instalación de Snort

Antes de la instalación de Snort, es muy importante determinar adecuadamente en qué posición dentro de la red será colocado, esto dependiendo de las condiciones del sistema de red, o de los criterios del administrador. El *IDS* debe ser colocado de forma que se posibilite al máximo el intercambio de información con los otros componentes del sistema como *firewalls*, *routers* o *switches* [20]. Las posiciones más comunes para la ubicación del *IDS* son:

- **Delante del Firewall:** en esta ubicación el *IDS* captura y analiza todos los paquetes de información que llegan a la red. Se detecta una mayor cantidad de posibles ataques, aunque muchos de estos sean falsas alarmas. Al procesar gran cantidad de datos se genera bastante información en los *log*, lo cual en muchos casos es innecesario e ineficiente.

- **Detrás del Firewall:** esta ubicación es la más común en la instalación de Snort. De esta forma el *IDS* analiza todo el tráfico que entra a la red y que ha sido admitido por el *firewall*. Se realiza un continuo monitoreo del funcionamiento del *firewall* y permite solucionar cualquier error de filtrado de paquetes. La cantidad de información almacenada en los *logs* es menor debido a que sólo analiza información procesada y filtrada por el *firewall*. Existen 3 esquemas en este modelo que pueden ser implementados:

  - **IDS colocado detrás de un Hub:** en la figura 3.4 se observa el *hub* ubicado antes que Snort. El recorrido de los paquetes que llegan para el análisis no es modificado.

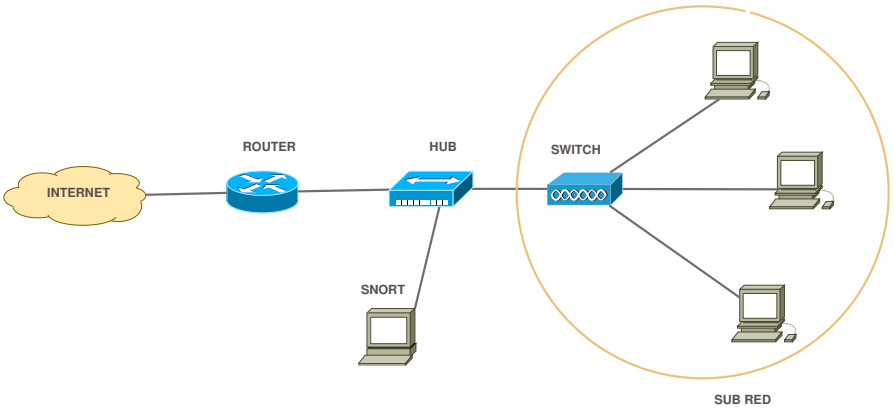

FIGURA 3.4: Implementación de Snort detrás del Hub. Fuente: Uso de Sistemas de Detección de intrusos para la seguridad perimetral. [21]

  - **IDS colocado tras un Switch:** en la figura 3.5 se observa que el *switch* recibe la trama, la almacena y luego recién se la envía a Snort.

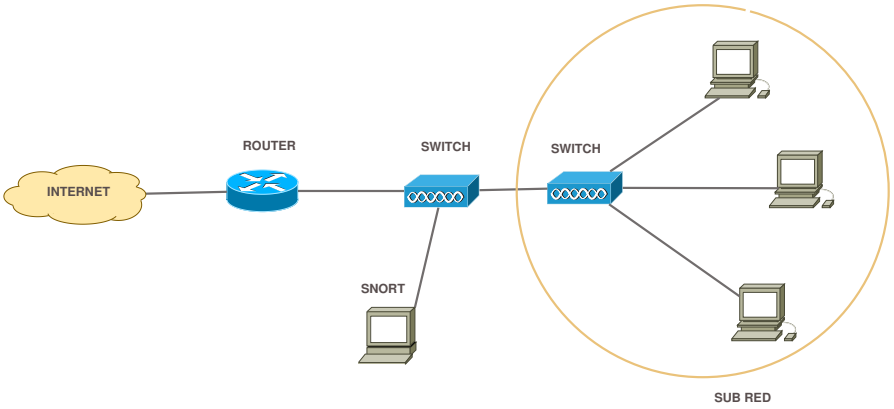

FIGURA 3.5: Implementación de Snort detrás del switch. Fuente: Uso de Sistemas de Detección de intrusos para la seguridad perimetral. [21]

- **IDS colocado en modo bridge:** como se observa en la figura 3.6, Snort está ubicado entre los dos adaptadores de red . En este esquema puede procesar toda la información que se intercambien entre ellos.

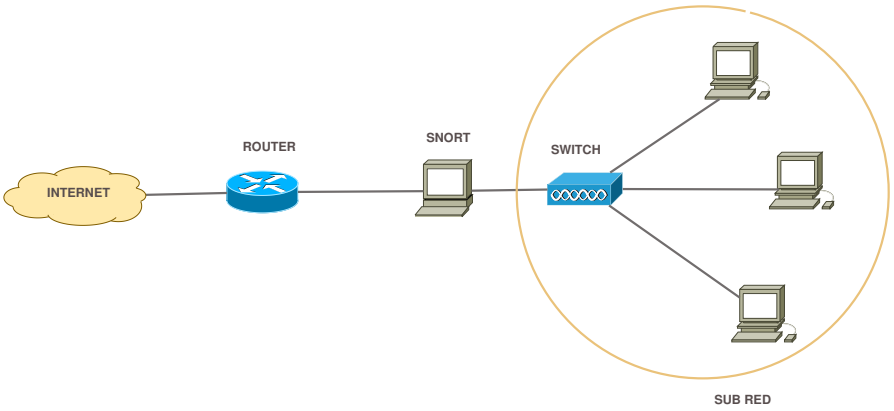

FIGURA 3.6: Implementación de Snort en modo bridge. Fuente: Uso de Sistemas de Detección de intrusos para la seguridad perimetral. [21]

- **Combinación de ambos casos:** implementando esta opción en red, el control que se tiene es mayor que en cualquiera de los casos anteriores. Permite hacer una comparación entre los ataques detectados en un lado del *firewall* y el otro, de esta forma se obtiene un diagnóstico más acertado.
- **En cada Host o Equipo:** esta opción consiste en instalar Snort en cada uno

de los equipos pertenecientes a la red. Ello adiciona una protección extra a los equipos del sistema, en caso de que el *firewall* o el *IDS* principal no detecten la intrusión. De esta forma se tiene un control y monitoreo más detallado en comparación a usar un solo monitor para toda la red.

Capítulo 4

Metodología y herramientas

4.1. Metodología

Durante el desarrollo de la aplicación móvil se utilizó la metodología **SCRUM**. En esta sección se estudiarán sus principales características y fases de implementación.

4.1.1. Metodologías Ágiles

La constante evolución de las nuevas tecnologías, la aparición de nuevos lenguajes de programación y el cambio de estilo en los programadores modernos ha causado que las metodologías tradicionales no puedan acoplarse adecuadamente a ellos. Bajo este contexto, desde principios de este siglo se han ido desarrollando las metodologías ágiles, cuya principal característica es trabajar con menos documentación, de forma contraria a las metodologías tradicionales. [22]

Para comprender en qué consiste las metodologías ágiles se describe el contenido del **manifiesto ágil**; documento que se resume la filosofía de este enfoque: [22]

- **El individuo y las iteraciones del equipo:** este primer valor menciona que primero se debe formar el equipo de trabajo adecuado y posteriormente

crear el entorno y no al contrario, lo que en muchos casos nunca se logra al 100 %.

- **Software funcional en lugar de demasiada documentación:** el valor menciona que a diferencia de las metodologías tradicionales, la documentación debe ser corta y breve, la adaptación de nuevos miembros será mediante la interacción con el equipo de trabajo y lectura del código existente.

- **Colaboración con el cliente:** este valor menciona que debe existir una continua interacción entre el cliente y el equipo de desarrolladores, de modo que el cliente observe el sistema y analice nuevas funcionalidades y objetivos. De esta forma el cliente al final quedará totalmente satisfecho.

- **Posibilidad de cambios de planes a medio proyecto:** este valor menciona que durante el desarrollo del software, si el cliente tiene la idea de incrementar sus objetivos, especificaciones o requerimientos, lo puede hacer sin ningún problema. Dado que, básicamente el sistema debe ser flexible para todo lo que pueda surgir en el proceso.

4.1.2. SCRUM

Es una de las metodologías de desarrollo ágil más populares actualmente, dado que es una combinación de los modelos incremental e iterativo para la administración del proceso de desarrollo de software. [34]

Esta metodología tiene las siguientes características: [22]

- **Desarrollo incremental:** utilizando SCRUM, el desarrollo se irá incrementando poco a poco, olvidando de cierta forma la planeación y ejecución de las líneas, pero sin desviarse de lo pre establecido, los resultados se muestran en los entregables.

- **Calidad de las personas:** la calidad final del producto no dependerá de los procesos, al contrario, dependerá de la organización y el conocimiento del trabajo en equipo.

- **Adiós a lo secuencial y cascada:** sin importar el proceso en el que se encuentre el equipo actualmente, si existe un requerimiento es posible volver a otro proceso y trabajarlo, lo cual no es posible en las metodologías cascada y secuencial, esta característica se llama *solapamiento*.

- **La comunicación es fundamental:** esta metodología permite una constante comunicación entre los miembros y equipos del proyecto, de esa forma toda la información es conocida y manejada por la gran mayoría.

Roles de la metodología Ágil SCRUM

- *Product Owner:* es el que representa y actúa como punto de contacto de los clientes, además es quien prioriza la lista de *Product Backlogs*, el cual debe ser cumplido por los miembros del equipo de trabajo.

- SCRUM *master:* actúa como facilitador del equipo de desarrollo *SCRUM*, su principal función es aclarar consultas acerca del uso de *SCRUM* y organizar al equipo previniendo posibles distracciones.

- Equipo SCRUM: Son los encargados en desarrollar todo el proyecto, están divididos en equipos de 5 a 7 personas con conocimiento en diferentes áreas; por ejemplo, el equipo de desarrolladores, se encarga de la implementación del producto, ellos pueden medir el esfuerzo estimado necesario para completar un entregable o ítem del Product *Backlog*.

- *Stakeholders:* personas en general interesadas en el proyecto, pueden ser un cliente o un *sponsor*. [22]

Artefactos de la metodología SCRUM

- Historia de Usuario: el historial de usuario, es un simple y descriptivo requerimiento del cliente, mediante su uso, los *stackeholders* mencionan las características del producto que necesitan y desean lograr.

- *Product Backlog:* es un repositorio donde se encuentran la lista de trabajos

pendientes, administrados y mantenidos por el *product owner*, quién puede darles una determinada prioridad.

- *Sprint Backlog:* es un conjunto de historias de usuario, los cuales son realizados por el equipo de trabajo durante un *Sprint* determinado. [24]

Flujo del proceso SCRUM

- Se define cada iteración a realizar siguiendo SCRUM conocido como *Sprint*.Durante el planeamiento de cada *Sprint* los ítem del *Product Backlog* son seleccionados por el *Product Owner* y colocados dentro del *Sprint Backlog*, según prioridad.

- El equipo de trabajo comienza a desarrollar el *Sprint* actual. Puede medir el esfuerzo necesario para completar dicho *Sprint*.

- El equipo hace revisiones periódicas (generalmente días) del trabajo. Si la revisión es diaria, cada miembro reporta lo que realizó el día anterior, que va a realizar el mismo día y que obstáculos presenta en su desarrollo. Generalmente las reuniones duran aproximadamente 15 minutos.

- Al final de periodo de ejecución del *Sprint*, se evalúan las funcionalidades del producto, y en base a ello el *Product Owner* decide si el *Sprint* ha sido completado. satisfactoriamente. [35]

En la figura 4.1 se observa el diagrama del proceso de la metodología SCRUM. En cada reunión de planificación, se seleccionan nuevos *Product Backlog* para ser ejecutados durante los *Sprints*. Un determindo *Sprint* es finalizado cuando el producto resultante es aprobado.

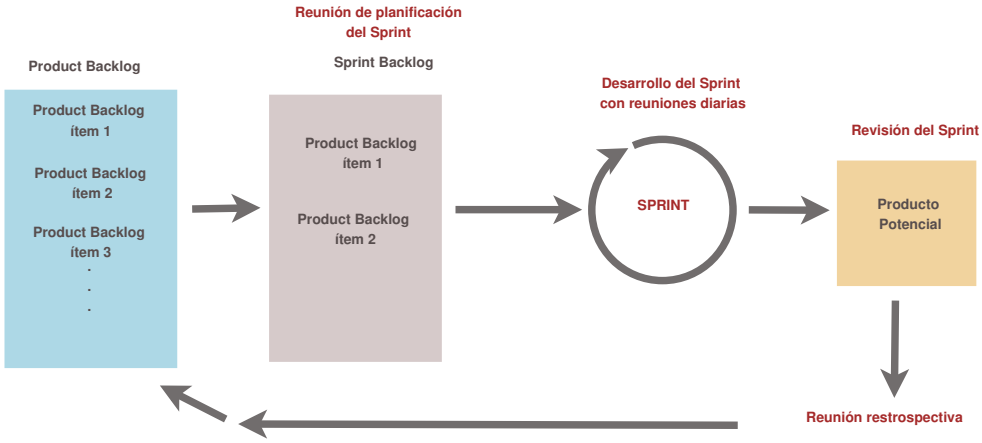

FIGURA 4.1: Proceso de la metodología SCRUM. Fuente: Software testing material. [23]

4.1.3. Uso de SCRUM en el proyecto actual

A continuación, se definen los artefactos y procesos SCRUM que serán utilizados en nuestro proyecto. En el cuadro 4.1, se especifican cada uno de los artefactos de SCRUM definidos en base a las funcionalidades que se requieren para la aplicación móvil.

Artefacto	Definición
Historia de Uso	El requerimiento principal es la implementación de una aplicación móvil para la administración de Snort de forma remota.Los requerimientos funcionales iniciales fueron: • Autenticación de usuarios. • Muestra de alertas generales registradas por Snort. • Muestra de información de los protocolos. • Implementación parcial o total de algunos *plugins* en versión móvil. • Sistema de notificaciones.

Product Backlog	En base a los requerimientos iniciales, se elaboró una lista de los trabajos a realizar: • Implementación de la funcionalidad de inicio de Sesión. • Diseño de las vistas de información general. • Diseño de las vistas de información de protocolos. • Implementación del tablero de indicadores del plugin BASE. • Desarrollo del plugin SnoGE. • Creación del sistema para el servicio de notificaciones.
Sprint Backlog	Cada uno de los puntos definidos en el *Product Backlog*, serán desarrollados independientemente en cada *Sprint*. En cada iteración, se irán mejorando las funcionalidades en base a las planteadas inicialmente. La lista detallada de *Sprint Backlogs* será especificada en el apéndice B.

CUADRO 4.1: Definición de los artefactos SCRUM para el proyecto.

En el cuadro 4.2, se describen las actividades y acciones a tomar en cada proceso de SCRUM en función a los artefactos previamente definidos.

proceso SCRUM	**Definición**
Definición de los *Sprint*	En la definición de cada *Sprint*, se toma como objetivo implementar alguna de las funcionalidades descritas en el *Product Backlog*.

Inicio del desarrollo de los *Sprint*	Antes de iniciar la ejecución, se definen el número de horas de trabajo (4 horas por día aproximadamente). Dependiendo de la dificultad de la programación o de las configuraciones del servidor, se aumentarán los días de trabajo (aproximadamente se calcula 15 días por cada uno).
Revisiones periódicas	Las revisiones de los Sprints serán cada 15 días aproximadamente y con una duración de 20 minutos cada una. En cada reunión se definirán mejoras funcionales de la aplicación móvil que se irán adicionando a los *Sprint Backlogs*. Se puede trabajar en paralelo uno o más *Sprints* o pausar temporalmente la ejecución de alguno de ellos.
Final de la ejecución del *Sprint*	Se verificará que la funcionalidad de la aplicación requerida para un determinado *Sprint*, haya sido implementada satisfactoriamente. Se incluyen todas las mejoras definidas en las revisiones.

CUADRO 4.2: Definición de los procesos SCRUM para el proyecto.

4.2. Técnicas, Herramientas y Tecnologías

4.2.1. Android

Android es un sistema operativo basado en GNU/Linux, orientado a dispositivos móviles con pantalla táctil como teléfonos inteligentes, tablets, televisores, etc. Actualmente utilizando las contribuciones de la comunidad de código abierto Linux y más de 300 colaboradores en *hardware* y *software*, Android se ha convertido el sistema operativo móvil de mayor y más rápido crecimiento en el mundo.

Una de sus principales características es el *framework* de desarrollo que

brinda a la comunidad. Android ofrece a los desarrolladores todas las herramientas necesarias para crear aplicaciones que aprovechen al máximo el *hardware* disponible en el dispositivo y a su vez adapten su interfaz de usuario de la mejor forma a cada dispositivo. [25]

Android también ofrece Google Play, una plataforma internacional abierta que permite la venta y distribución de aplicaciones al instante, en esta plataforma se puede publicar lo que el desarrollador desea, para los clientes que desea y con la frecuencia que desea. [25]

Dado que Android poseen una gran variedad de características, beneficios y es uno de los más utilizados en comparación a otros sistemas para dispositivos móviles, la aplicación será desarrollada en este sistema.

4.2.2. Android Studio

Android Studio es actualmente el *IDE* (*Integrated Development Enviroment*) oficial para desarrollo de aplicaciones Android, está basado en IntelliJ IDEA, otro *IDE* para el desarrollo de programas informaticos. Android Studio ofrece una gama extra de funciones adicionales a las ya existentes en IntelliJ. A continuación se mencionan algunas de ellas:[26]

- Sistema de compilación flexible basado en *Gradle*.
- Emulador con varias funciones.
- Entorno que permite desarrollar aplicaciones para todas las versiones de Android existentes.
- *Instant RUN*, para modificar la aplicación sin la necesidad de generar otro *APK*.
- Gran variedad de herramientas *frameworks* de prueba
- Herramientas para detectar problemas de compatibilidad de versión rendimiento, etc.

Android Studio será utilizado para implementar la aplicación móvil del presente trabajo, dado las ventajas que ofrece, en comparación a otros *IDE* como Eclipse.

4.2.3. PHP

PHP: *Hipertext Preprocessor* por su acrónimo recursivo en inglés, es un lenguaje de propósito general para desarrollo de *scripts* con licencia *open source*, muy utilizado en la actualidad, especialmente para desarrollo web.[27]

Entre las características más importantes de *PHP* destacan, su capacidad para soportar un amplio rango de bases de datos, soporte para enlazarse con otros servicios mediante el uso de múltiples protocolos *(HTTP, LDAP, IMAP*, etc.). Además de ser un lenguaje que puede ser ejecutado en la mayoría de sistemas operativos de servidores y computadoras personales (Windows, GNU/Linux, Mac OS X, etc.). [28]

Existen tres principales formas de ejecución para *PHP*, siendo la principal en un servidor web, el código es ejecutado en el servidor generando contenido en diferentes formatos (*HTML* el más conocido, *JSON*, *XML*, entre otros) el cual es enviado al cliente, la información puede ser vista mediante un navegador web o consumido por otro servicio (*php-mysql*, *php-fcm*,etc.).

En el presente trabajo, se hará uso de este lenguaje para implementar los *scripts* de extracción de datos, la actualización de la base de datos Snort y además envío de notificaciones a nuestra aplicación, por medio de las diferentes librerías que nos facilita para estas tareas.

4.2.4. Apache 2.0

Es un servidor web que implementa los últimos protocolos de transferencia de Hypertexto, siendo el principal *HTTP/1.1*; Apache es de código libre y puede ejecutarse en múltiples sistemas operativos como Windows y GNU/Linux. Sus principales características son: [29]

- Soporte para conexión con base de datos relacional y no relacional.

- Autenticación de usuarios mediante *LDAP*.
- Opción de configuración y personalización para sus diferentes módulos, por ejemplo respuesta a errores.
- Creación de múltiples espacios de trabajo (directorios) para ejecución independiente de múltiples archivos. Además uso de múltiples alias o *URL* mediante la configuración de virtual *hosts*.

La implementación del servidor será íntegramente *open source* empleando el sistema operativo *Ubuntu Server*, por ello el servidor web más apropiado es Apache, dado las características que ofrece en comparación a otros servidores *open source*.

4.2.5. Firebase Cloud Messaging

Firebase Cloud Messaging (*FCM*) es una solución multiplataforma para el envío de forma segura y gratuita de mensajes y notificaciones a dispositivos móviles con sistema operativo Android o iOS.

La implementación de *FCM*, incluye un servidor de *app* en un entorno local que interactúa con el servidor central de *FCM* utilizando el protocolo *HTTP* o *XMPP*, y una *app* cliente (dispositivo móvil de usuario). Las funcionalidades clave que ofrece *FCM* son: [30]

- Perfilamiento versátil de mensajes, permite distribuir mensajes a los clientes de tres formas: dispositivos individuales, grupos de dispositivos o dispositivos subscritos a temas.
- Soporte de notificaciones y mensajes de datos, envía mensajes con carga de datos (tamaño máximo 2*KB*) y notificaciones (tamaño máximo 4*KB*)
- Mensajería ascendente a las *app* cliente, envió de mensajes desde la aplicación hacia el servidor con poco consumo de energía.

El sistema de notificaciones para la aplicación del proyecto, utilizará *FCM* en lugar del clásico *GCM* (Google Cloud Messaging) actualmente en desuso

gradual, *FCM* ofrece un nuevo *SDK* con más funcionalidades y mayor facilidad para la implementación de la *app* cliente.

En la figura 4.2 se observa el esquema de fucionamiento de Firebase Cloud Messaging. Las notificaciones son enviadas al servidor central desde la consola de Firebase o desde un servidor externo. Finalmente, el servidor central envía dichas notificaciones hacia los dispositivos móviles previamente registrados.

FIGURA 4.2: Esquema de funcionamiento FCM. Fuente: Página de inicio Firebase. [30]

4.2.6. Google Maps API para Android

La *API* de Google Maps, brinda la opción de insertar mapas en aplicaciones Android, todo el proceso de transmisión de información, acceso a servidores, visualización de información y respuesta a interacciones con el usuario es administrador por esta *API*.[31]

Las principales características que Google Maps brinda a los desarrolladores son: [32]

- Múltiples Herramientas para agregar mapas a la aplicación Android, la versión 2 permite utilizar *fragments* para insertar mapas vectoriales que

mejoran la eficiencia de animación y transición, las vistas pueden ser 3D, satelital, de terreno o hibrido.

- Personalización de mapas mediante la adición de *markets* para definir puntos específicos y líneas poligonales para definir áreas de terreno con múltiples opciones gráficas (colores, leyendas, estilos, etc.).

- Control de vista de usuario, los mapas tiene las opciones de *zoom*, rotación, etc.

- *Street view*, de forma similar a los mapas de Google más versión web, es posible realizar una vista panorámica real en 360 grados de un punto específico.

Se utilizará la *API* de google Maps para la implementación del *plugin* snoGE, para ello se creará un mapa en donde se graficarán todas las posiciones de las *IPs* involucradas en los eventos (atacantes y victimas) empleando marcadores. Adicionalmente se trazará una ruta atacante victima empleando las líneas poligonales.

En la figura 4.3, se observa un mapa de Google Hibrido caracterizado por estar implementado utilizando imágenes satelitales. En este tipo de mapas se puede ver más claramente las condiciones del terreno.

FIGURA 4.3: Google Maps Android Tipo Hybrid. Fuente: Androidsrc. [33]

4.2.7. Mysql

Mysql, es un sistema de administración de base de datos *SQL open source*, desarrollado y distribuido por Oracle [34]. Sus principales características son:

- Las base de datos Mysql son relacionales, la información es almacenada y separada en tablas.
- Mysql tiene un servidor caracterizado principalmente por la velocidad, escalabilidad y fácil uso, puede ser instalado en varios tipos de computadores desde personales hasta *clústeres*.
- El sistema se ejecuta en múltiples hilos que soportan varios tipos de clientes, herramientas de administración y *APIS* para programación.

El almacenamiento de alertas en base de datos es realizado por el *plugin* Barnyard2 el cual utiliza Mysql como gestor *open source* para GNU/Linux, por ello utilizaremos Mysql como base de datos principal para el proyecto, además todos los *scripts PHP* para administración y extracción de información serán implementados con soporte a Mysql.

4.2.8. SQlite

SQlite es un motor de base de datos *SQL* embebida, a diferencia de otros motores *SQL*, SQLite lee y escribe directamente en los archivos de disco. Toda la estructura compuesta por tablas, índices, vistas, triggers es almacenada en un simple archivo dentro del disco. [35]

SQLite será el motor de base de datos interna utilizado para el desarrollo de nuestra aplicación, debido a sus características y eficiencia de funcionamiento, las más resaltantes son:

- Mínimo espacio de 4*KiB* en disco para ejecutarse.
- Tamaño de librería en disco mínimo entre 300 *KiB* y 500 *KiB* y tamaño de pila de solo 100 *KiB*,

Todas las características mencionadas anteriormente hacen que SQLite sea una bases de datos bastante popular para uso en dispositivos móviles (*Smartphones*, reproductores *MP3* y *MP4*).

4.2.9. Material Design

Es una guía para el diseño y creción de nuevos componentes visuales, movimientos e interacciones en distintas plataformas. Está compuesto por un conjunto de librerias que ofrecen a los desarrolladores nuevos temas, nuevos *widgets* y nuevas *APIs* para animaciones. Estas caracteristicas estan disponibles a partir de Android 5.0 en adelante. [36]

Entre los principales recursos que ofrecen estas librerias destacan: [37]

- Listas y Tarjetas: los nuevos componentes *recyclerView* y *cardview* tiene un mayor rendimiento, mejores estilos y más opciones para mostrar la información.
- Botones: distintos tipos de animaciones para botones, incluyendo el conocido *float button*.

- Animaciones: existen varios tipos de animaciones en los que destacan las transiciones entre vistas o *fragments*, también los despliegues de vistas en pestañas, menús y barras.
- Efectos: uno de los más conocidos son los efectos de sombras para mayor realismo en la interacción con los usuarios.
- Colores: permite utilizar una amplia gama de colores en paletas combinadas adecuadamente para cada situación.

En el diseño de SnorUNI, se utilizará la librería *Material Design* para mejor la apariencia visual de las vistas de la aplicación (por ejemplo, mostrar las alertas y usuarios en *cards* y emplear *float buttons*), hacer más entretenidas las interacciones con el usuario (animación de movimiento de vistas y *fragments*) y mejorar el rendimiento de la aplicación reemplazando componentes obsoletos.

4.2.10. Ingeniería Inversa

Es el proceso que consiste en analizar el funcionamiento de un objeto con el fin de duplicarlo, mejorarlo u obtener un esquema general de su estructura interna. Este proceso es bastante usado en diferentes campos, siendo uno de los más frecuentes el desarrollo de *software* dentro del cual consiste en obtener el código fuente de un programa, cabe resaltar que estos procesos son sin fines negativos. [38]

En el caso específico de base de datos, ingeniería inversa consiste en reconstruir el modelo físico y/o el modelo *ER* de un determinado esquema, generalmente es utilizado para crear documentación inexistente, proveer a los desarrolladores una mejor visión del esquema dentro del sistema y realizar mejoras de rendimiento al sistema. [38]

Se va a utilizar ingeniería inversa para implementar los diagramas *ER* y físico de la base de datos Snort, con el fin de analizar las características de sus tablas y sus relaciones, de esa forma se crearán tablas auxiliares y se implementarán las consultas de extracción de información.

Capítulo 5

Estructura y diseño del servidor Snort

En este punto, se analizará a detalle la estructura y diseño del servidor de Snort incluyendo sus principales funcionalidades. Además, se muestra el estudio de ingeniería inversa realizada a la base de datos con la finalidad de crear los diagramas físicos y *ER*.

5.1. Servidor Snort

El servidor de Snort es el encargado de procesar todo el tráfico de red en busca de posibles eventos, encargándose también de almacenar la información y enviarla al dispositivo móvil incluyendo las notificaciones. En esta sección se describen los componentes del servidor donde se encuentra la base de datos, la *web service* (extracción de información) y el servidor de notificaciones; se analizarán sus funcionalidades y características.

5.1.1. Estructura y manejo de la base de datos Snort

Como se mencionó en el estado del arte, en las nuevas versiones de Snort es necesarios utilizar el *plugin* Barnyard2 para implementar el almacenamiento de la información en una base de datos. Para su funcionamiento, en el proceso

de instalación se crea una base de datos denominada "*Snort*", la cual varía dependiendo del motor (Mysql, Oracle, *SQL* server), esa base de datos será utilizada para guardar toda la información de Snort.

Nuestro servidor de prueba tiene como motor principal de base de datos Mysql ejecutándose sobre el sistema operativo *Ubuntu 14.04 LTS*. Una vez instalada la base de datos, es necesario crear tablas auxiliares que permitan administrar y extraer información de forma eficiente a través de la *web service*; la base de datos es continuamente actualizada por el *plugin* guardando nueva información. Por tanto, las tablas adicionales también deben ser actualizadas constantemente.

a) **Estructura inicial de la base de datos**

Una vez creada la base de datos Snort, se hizo uso de ingeniería inversa para crear sus diagramas Físico y *ER*. Estos diagramas nos brindan una visión completa de las tablas y sus relaciones, permitiendo realizar diferentes modificaciones a la base de datos de nuestro sistema. Las fases del proceso de ingeniería inversa fueron:

- Descripción de las funciones y componentes de cada tabla y la información que almacenan.
- Implementación del diagrama físico de la base de datos utilizando el programa *open source* PhpMyadmin. Este programa ofrece la funcionalidad de realizar ingeniería inversa en una base de datos para obtener su diagrama físico.
- Con la información recabada en los puntos anteriores, se implementó el diagrama *ER* para la base de datos.

En la figura 5.1 se muestra el diagrama físico de la base de datos Snort creado por PhpMyadmin. A partir de este diagrama se definen sus principales características:

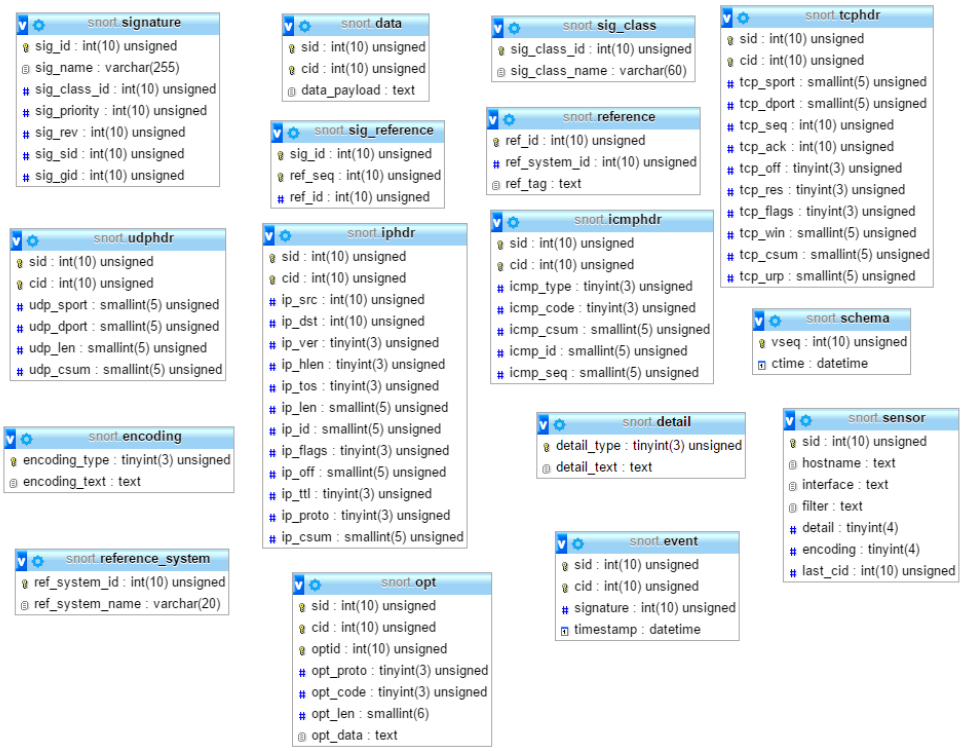

FIGURA 5.1: Diagrama Físico de la base de datos Snort. Fuente: Elaboración propia.

- La base de datos está formada por 16 tablas para el almacenamiento de la información proveniente de los *logs* de Snort.
- La información referente a los protocolos es almacenada en una tabla propia para cada protocolo (por ejemplo, *icmphdr* almacena toda la información referente al protocolo *ICMP*).
- La tabla *signature* contiene información predefinida de todos las firmas utilizadas por Snort, las cuales hacen referencia a un evento o alerta determinado.
- La tabla *sig_class* contiene información predefinida de los tipos de firmas existentes, haciendo referencia a los tipos de ataque registrados.
- La tabla más importante es *event*, dicha tabla guarda la información de los eventos registrados. Cada una de sus entradas tiene un identificador único *cid* y el identificador del sensor el cual lo registró

sid. Estos dos campos son constantemente utilizados por las otras tablas como índices para guardar información.

- Los *payload* de los paquetes capturados son guardados en la tabla *data*, esta información es muy importante para posibles nuevos estudios o un análisis más profundo de esa información.
- La tabla *schema* guarda información de la versión del esquema utilizado para la creación de base de datos. Cabe resaltar que dependiendo de esta versión los campos y/o nombres de las tablas pueden variar. Dicha información será muy importante para la implementación del *script* de actualización.

En la figura 5.2 se observa el diagrama *ER (Min-Max/ISO)* presentando las relaciones entre las tablas, se puede constatar que la tabla *events*, es la principal, seguida de *sensor*, *signature* y las tablas de información de protocolos.

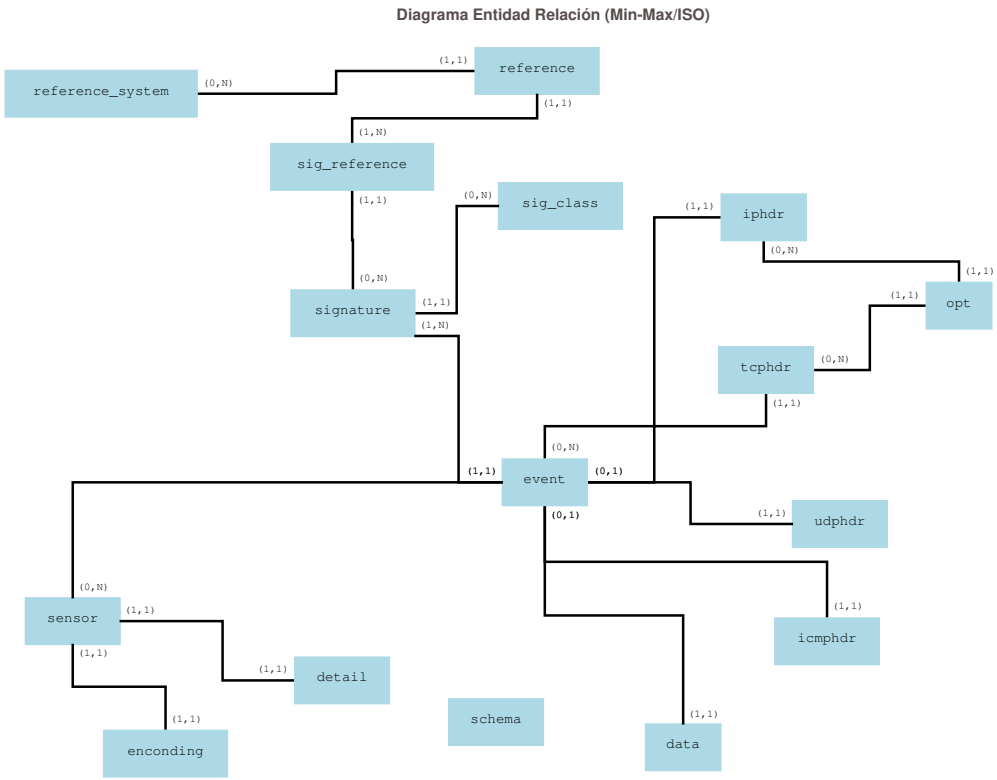

FIGURA 5.2: Diagrama ER de la base de datos Snort. Fuente: Elaboración propia.

En el cuadro 5.1, se describe la información almacenada por las tablas de la base de datos Snort y sus posibles usos.

Tablas Mysql	**Descripción**
schema	Brinda información de la base de datos, los datos guardados incluyen el *ID* del esquema de la base de datos y la fecha de creación de la misma.
sensor	Muestra detalles de los sensores que recogen información para Snort.
event	Almacena información de los eventos, los datos guardados incluyen al sensor que los detecto, la regla activada de Snort y la fecha en la que fue detectado.
signature	Almacena información de las firmas detectadas, incluye el nombre, prioridad y clasificación.
sig_reference	Muestra información de las referencias de una determinada firma.
reference	Contiene información de las etiquetas de las referencias, está relacionado a las referencias del sistema.
reference_system	Contiene una lista de todas las referencias del sistema.
sig_class	Esta tabla contiene información acerca de la clasificación de las firmas incluyendo su *ID* y nombre.
data	Muestra una fracción (la parte importante o *payload*) del paquete capturado y codificado por el sensor.
iphdr	Muestra información del protocolo *IP* relacionada a los eventos registrados por los sensores.
tcphdr	Muestra información del protocolo *TCP* relacionada a los eventos registrados por los sensores.
udphdr	Muestra información del protocolo *UDP* relacionada a los eventos registrados por los sensores.
icmphdr	Muestra información del protocolo *ICMP* relacionada a los eventos registrados por los sensores.

opt	Muestra información de las opciones *TCP* ó *UDP*.
detail	Guarda información del nivel de detalle utilizado por el sensor para almacenar información.
encoding	Contiene información sobre el tipo de codificación utilizado en los *payload* en los sensores.

CUADRO 5.1: Descripción de la información almacenada por las tablas de la base de datos Snort.

b) **Modificación de la base de datos**

Antes de iniciar el desarrollo de la aplicación móvil, se hizo un análisis del funcionamiento y estructura de los *plugins* BASE y SnorGE, los cuales también serán plasmados en versión móvil.

Se observó que el plugin BASE al ser instalado, modifica la estructura inicial de la base de datos Snort agregando tablas auxiliares para un mejor manejo de la información almacenada y facilitar las consultas, en la siguiente tabla se describe de forma resumida sus características.

En el cuadro 5.2, se describen las tablas *base_roles, base_users* y *acid_event* utilizadas por BASE. Las dos primeras tablas guardan información de los usuarios y roles para realizar la autenticación. La tercera tabla *acid_ event* es la más importante de las tres e inclusive más importante que la tabla original *event*, dado que guarda información concisa y resumida que es muy importante para la mayoría de consultas hechas por BASE.

Tablas Auxiliares	**función**
base_roles	Contiene información de los roles de administración de los usuarios dentro de la base de datos.
base_users	Contiene información de los usuarios de la base de datos.

acid_event	Contiene información resumida de todos los eventos registrados, tiene los mismos campos definidos en *event* y adiciona información del protocolo *IP*, la clasificación del evento, prioridad, etc.
acid_ag	Muestra información sobre grupos de alerta generados, los principales campos son *ID*, nombre, descripción y fechas de creación y modificación.
acid_ag_alert	Muestra los eventos de cada uno de los grupos creados.
acid_ip_cache	Muestra información *DNS* y de identificación de las *IPs* registradas de los eventos.

CUADRO 5.2: Descripción de las tablas auxiliares de BASE.

Para el sistema de SnorUNI, se crearon tres tablas con la misma estructura que las tablas auxiliares de BASE con la finalidad de cumplir los siguientes objetivos:

- Guardar la información de los campos principales dentro de una sola tabla de forma resumida y concisa, de forma que las consultas principales sean mucho más simples y eficientes.
- Utilizar consultas *SQL* similares a las implementadas en BASE para la extracción de información.
- Guardar información de usuarios del sistema para poder autenticar la conexión con la *web service* y la aplicación móvil.

En la figura 5.3 se muestra el diagrama físico de las tablas auxiliares creadas exclusivamente para interactuar con la aplicación móvil. La tabla *snoruni_event* será constantemente actualizada, dado que es la tabla principal para la extracción de información.

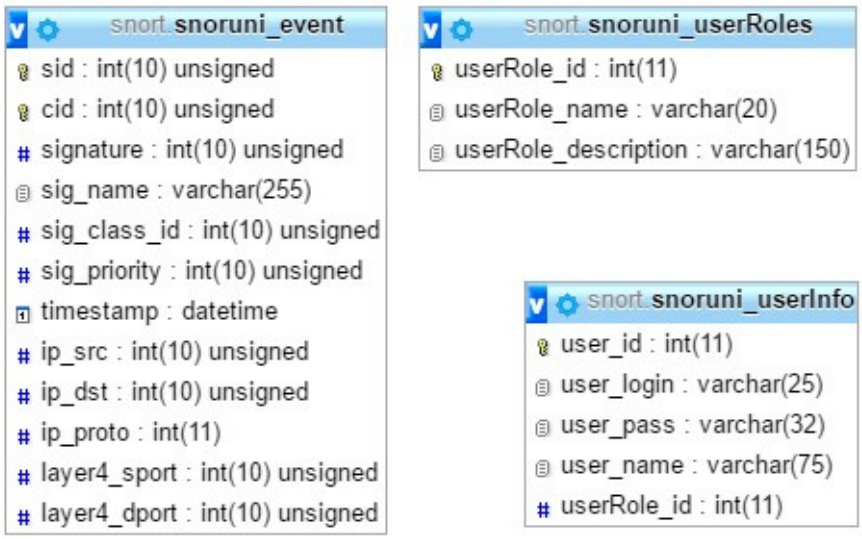

FIGURA 5.3: Diagrama Físico de las tablas auxiliares snoruni_event, snoruni_userRoles y snoruni_userInfo. Fuente: Elaboración propia.

La primera tabla es *snoruni_event* y cumplirá la misma función que *acid_event* almacenando datos conbinados de varias tablas. La segunda tabla es denominada *snoruni_userRoles*, en esta tabla se almacenarán los posibles roles de los administradores. Finalmente, la tercera tabla es *snoruni_userInfo* y guardará información de las credenciales de los usuarios.

En la figura 5.4 se observa el diagrama *ER* final de la base de datos Snort. La tabla auxiliar *snoruni_event* interactúa con la mayoría de las otras tablas al igual que *acid_event*. Es por ello que sus campos contienen información centralizada, permitiendo extraer información de forma fácil y rápida.

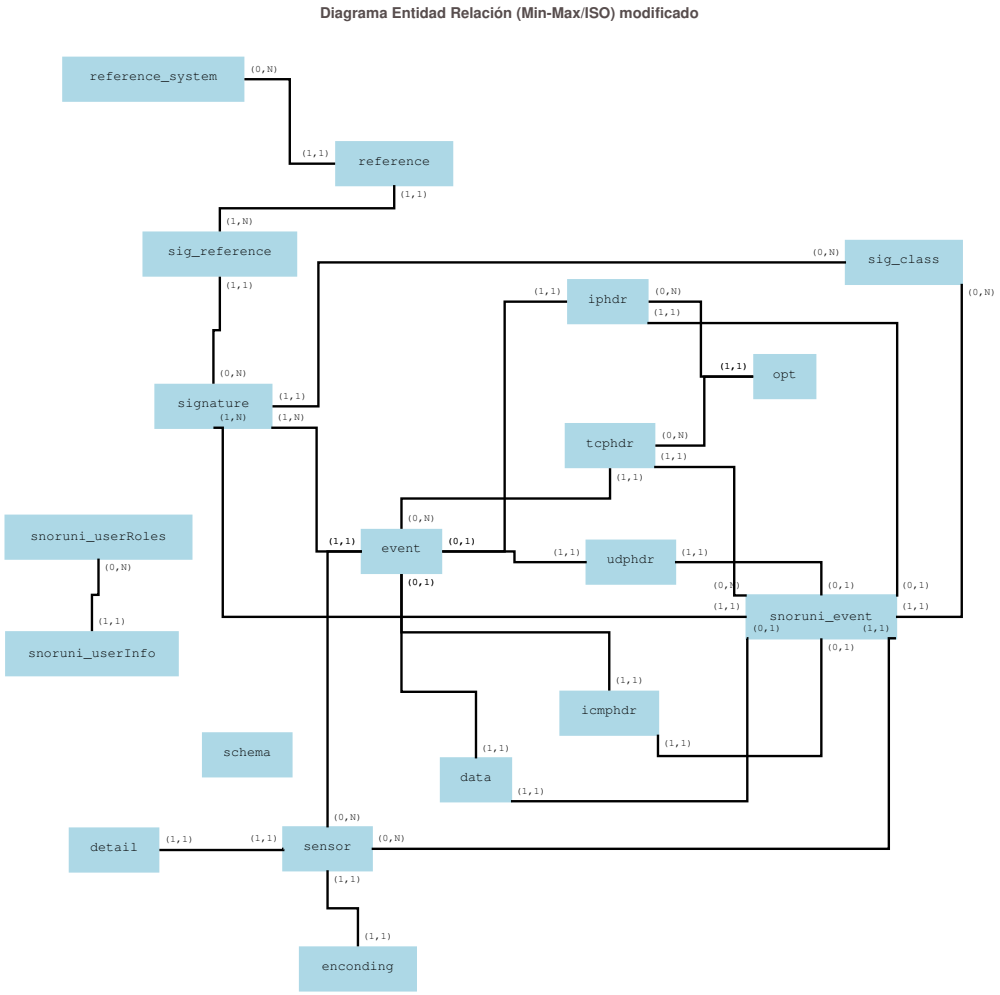

FIGURA 5.4: Diagrama ER final para el sistema SnorUNI. Fuente: Elaboración propia.

c) **Creación y actualización automática de la base de datos**

Es necesaria la actualización periódica de las tablas auxiliares creadas para la aplicación, con ese propósito se implementó dos *scripts* en PHP para cumplir esta función. A continuación se describen los dos procesos principales:

- **Creación de tablas**

  La creación de las tablas auxiliares, se realiza mediante un *script* denominado "*create_dbtables.php*", a continuación se detallan sus características y funcionalidades:

- Define las variables para las credenciales de acceso a la base de datos.
- Define las sentencias *SQL* con las estructuras de las tablas auxiliares (*snoruni_event, snoruni_userRoles, snoruni_userInfo*), al ejecutarse crean las tablas dentro de la base de datos.
- Define un conjunto de consultas, las cuales varian dependiendo de la versión de base de datos creada por Barnyard2.
- Para llenar la tabla *snoruni_event*, las consultas *SQL* extraen datos de las tablas *icmpdr, tcpdr, udpdr, ipdr* y *event*.
- Por defecto se crearon dos usuarios y un rol en las tablas de usuarios y roles respectivamente.

- **Actualización de tablas**

  La actualización de la información, consiste en insertar los últimos registros ingresados a la base de datos dentro de la tabla *snoruni_event*. Esta funcionalidad la realiza el *script* denominado "*update_dbtables.php*", a continuación se describen su forma de funcionamiento y características:

  - El programa definen las variables y credenciales para la conexión a la base de datos.
  - Utiliza el procedimiento *insertRowSnorUNI* para insertar un determinado evento dentro de la tabla auxiliar, utiliza los parámetros *cid* (identificador del evento), *sid* (identificador del sensor) y *conexion* (conección con la base de datos).
  - Para calcular el número de eventos (filas) a insertar utiliza las tablas *sensor* y *snoruni_event*; de la primera tabla se extrae el número de sensores y por cada sensor el máximo identificador de registro (*cid*); de la segunda tabla se extrae el máximo *cid* guardado, la diferencia de estos será el número de registro nuevos que necesitan ser ingresados.
  - Utilizando un *bucle* se ingresa cada registro nuevo empleando el

procedimiento *insertRowSnorUNI*.

- **Función inserRowSnorUNI**

  A continuacón se describe el funcionamiento de la función auxiliar para insertar cada registro a la tabla auxiliar:

  - Una vez definido un *sid* y *cid* y extraídos todos los campos de su respectiva fila desde la tabla *event*, se analiza el campo *ip_proto* (tipo protocolo), dependiendo de su valor se puede establecer que la información guardada en esa fila corresponde a un evento *TCP*, *UDP* o ninguno. Si es alguno de los dos primeros, se realiza una consulta adicional para extraer información de sus puertos desde las tablas *udpdr* o *tcpdr* empleando los *sid* y *cid* iniciales.
  - Otro elemento importante es el nombre de la firma *sig_name* el cual es un campo obligatorio en la tabla auxiliar, por tanto se realiza una consulta para extraer dicha información de la tabla *signature*. Dependiendo de la versión de base de datos se extraen campos adicionales como el identificador de la clase de firma o la prioridad.
  - De las consultas anteriores se obtienen los valores de los campos de la tabla *snoruni_event* para los *cid* y *sid* definidos al inicio, si algún campo no tiene valor se le define como nulo (*null*).
  - Finalmente, se insertan los valores dentro de la tabla *snoruni_event* para los *cid* y *sid* definidos en una columna determinada, con ello termina la ejecución de la función *insertRowSnorUNI*.

En la figura 5.5, se observa el diagrama de flujo de la función utilizada por *update_dbtables.php* para insertar eventos en la tabla *snoruni_event*. La función, utilizando los identificadores *sid* y *cid* extrae información de las tablas en base al tipo de protocolo de la información y el modelo del esquema de la base de datos.

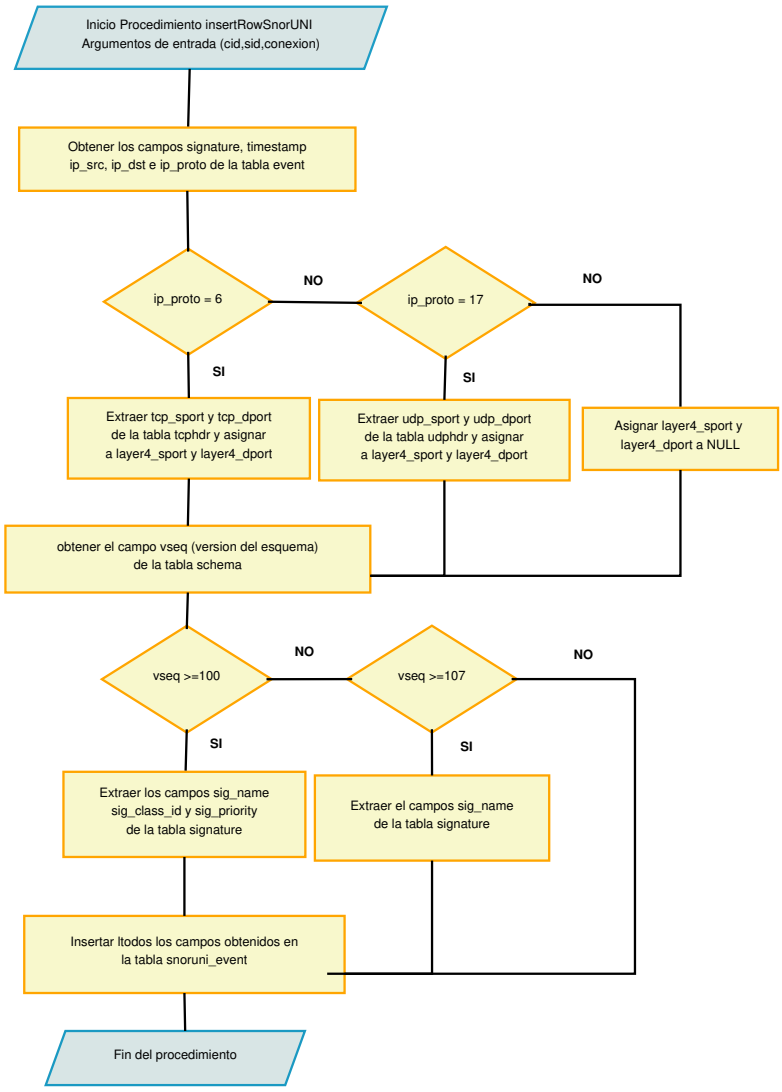

FIGURA 5.5: Diagrama de flujo de la función insertRowSnorUNI.

Fuente: Elaboración propia.

5.1.2. Servidor de Notificaciones

El servidor de notificaciones se encarga de enviar una alerta al servidor central de **Firebase Cloud Messaging** con información de las nuevas filas insertadas en la tabla *snoruni_event*, a su vez estas alertas son enviadas a las aplicaciones móviles registradas a un tópico determinado, siendo el usuario quien al final recibe una notificación.

Se implementó el servidor en *PHP* en el mismo *script update_dbtable.php*, a continuación se describe sus principales características:

- Hace uso de la librería **phpFCM**, la cual contiene los módulos **phpFCM/Client**, **phpFCM/Message**, **phpFCM/Recipient** y **phpFCM/Notification** para manejar lo referente a creación del cliente, creación del mensaje, definición del tópico y el envío de la notificación respectivamente.

- El usuario define el tópico único para el envió de notificaciones desde su servidor, el patrón de cadena del nombre es "*channelsnorUNI_<ip servidor>*", así con la *IP* pública o *URL* de su servidor pueden crear un tópico propio y único para el envío de notificaciones hacia un conjunto de dispositivos determinados.

- El envío de información es definido en los campos de la sección *data*, y no en la sección *notification*; este aspecto permitirá ingresar automáticamente a la aplicación sección *events* para ver los nuevos eventos ingresados.

- Cada vez que se ingresan nuevos registros a la tabla *snoruni_event* para un determinado sensor, se envía una notificación con los nuevos identificadores (*cid*) ingresadas.

5.1.3. Automatización de ejecución

Una vez que el servidor entre en funcionamiento, es necesario que la base de datos se actualice constantemente y que las notificaciones se envíen en el momento oportuno, es por ello que el programa *update_dbtables.php* debe ejecutarse periódicamente.

Para la automatización de la ejecución se utilizó el programa *Crontab*, el cual viene incluido por defecto en casi todas las versiones de Linux. Sus principales beneficios, son la ejecución en modo súper usuario (no hay necesidad de ingresar contraseñas) y la robustez de su funcionamiento.

Los archivos de la *web service* y los programas de automatización y creación se encuentran en el directorio *snoruni* dentro del */var/www/html*, por ello se configura Crontab para ejecutar *PHP* sobre este directorio, agregando un

archivo *log* para mostrar el resultado de los eventos programados.

5.1.4. Configuración HTTPS

Actualmente para transferencia de datos de hipertexto, los dos protocolos más utilizados son *Hypertext Transfer Protocol (HTTP)* y *Hypertext Transfer Protocol Secure (HTTPS)*, siendo el segundo la versión segura dado que encripta la información transferida entre el servidor y la aplicación que consume datos (generalmente un *Browser*); de esta forma se previene que personas ajenas tengan acceso a esta información interceptando la comunicación.

HTTPS utiliza a *SSL* o *TLS* como protocolo de encriptación. Estos dos protocolos utilizan el cifrado asimétrico, el cual está compuesto por una llave publica y una privada; la llave pública es necesaria para desencriptar información previamente encriptada con la llave privada y viceversa. En el caso del protocolo *HTTPS*, la llave privada es almacenada en el servidor y la pública es enviada al navegador o a la aplicación en específico dentro del certificado *SSL*, estableciendo finalmente una conexión segura para el intercambio de información. [50]

En la figura 5.6, se observan las diferencias entre el uso de los protocolos *HTTP* y *HTTPS*. Gracias a que la comunicación es cifrada en el segundo caso, si el flujo de datos es interceptado, será muy complicado para el atacante descrifrar la información.

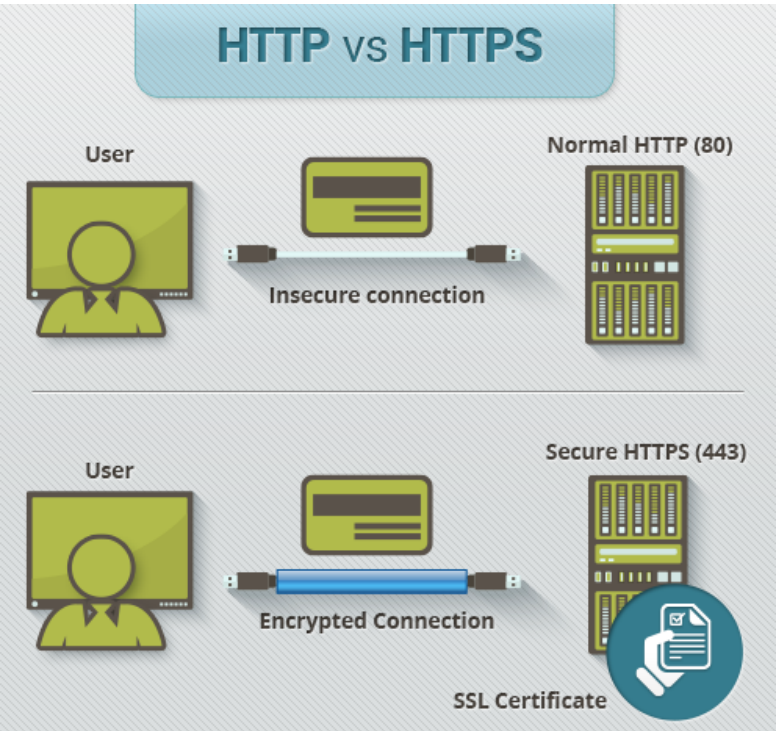

FIGURA 5.6: Diferencia entre el protocolo HTTP y HTTPS. Fuente: Instantssl.com. [39]

En el caso de nuestro sistema, para el intercambio de información entre la aplicación y el servidor utilizaremos el protocolo *HTTPS*, para ello se tiene que configurar y crear un certificado *SSL* con firma propia (*self-signed SSL certificate*) para su uso en Apache. A continuación se describen los pasos más importantes: [40]

- Activar el modulo *SSL*.
- Crear el certificado *SSL* con firma propia.
- Configurar el archivo *ssl-default.conf* para crear el *virtual host* con *SSL*.
- Activar el *virtual host SSL*.

5.2. Web Service

La implementación de la *web service*, consta de un conjunto de programas implementados en PHP para acceder a la base de datos Snort, extraer

información diversa y enviarla a otras aplicaciones mediante los protocolos *HTTP* o *HTTPS*.

El proceso empieza cuando la aplicación móvil inicia sesión, entonces ejecuta el *script infousers.php* para verificar la autenticación del usuario, posteriormente si se ha realizado satisfactoriamente la autenticación y una se ha ingresado a la aplicación, el usuario puede seleccionar múltiples opciones de vistas. Para mostrar la información al usuario, aplicación móvil envía un pedido al servidor especificando el *script* a ejecutar dependiendo de la consulta. Durante la ejecución en el servidor los *scripts* ingresan a la base de datos y extraen la información deseada; finalmente el servidor devuelve una respuesta en formato *JSON*, la cual es recibida y procesada por la aplicación móvil para luego mostrársela al usuario.

En el cuadro 5.3, se describe de forma detallada las funcionalidades de cada uno de los *scripts* que conforman la *web service*. También describen los principales campos de la información devuelta por la consulta a la base de datos.

Programa	**función**
user_authenticate.php	Realiza la consulta de verificación para el ingreso de usuarios, empleando usuario y contraseña devuelve el rol del usuario como respuesta para ingresar al sistema.
events.php	Extrae información de la base de datos referente a las alertas guardadas en la tabla *snoruni_event*, los campos principales incluyen nombre de la firma (ataque), fecha del evento, clasificación de la firma y *host* donde se detectó el evento.
icmpinfo.php	Extrae información de los eventos *ICMP* registrados por Snort, utiliza las tablas *snoruni_event* e *icmpdhr*.
infousers.php	Extrae información de los usuarios registrados en las tablas *snoruni_userRole* y *snoruni_userInfo*.

ipinfo.php	Extrae información combinada de las tablas *snoruni_event* e *iphdr*, referente a los eventos tipo *IP* registrados, entre sus principales campos se tiene las *IPs* de destino y origen involucradas, la versión de la *IP* y el protocolo utilizado.
getlistips.php	Extrae todas las *IPs* destino para cada una de las *IPs* origen, la información es extraída de la tabla *snoruni_event*, esta información es empleada por el *plugin* snorGE para graficar sus coordenadas *GPS* en Google Maps.
resume.php	Extrae información del número de eventos registrados de una determinada firma o ataque, incluyendo su clasificación, para ello emplea la tabla *snoruni_event*.
tcpinfo.php	Extrae información de los eventos *TCP* registrados por Snort, los campos son extraídos de las tablas *tcphdr* y *snoruni_event*.
udpinfo.php	Extrae información de los eventos *UDP* registrados por Snort, los campos extraídos son de las tablas *udphdr* y *snoruni_event*.
extractión_base.php	Es empleado por el plugin BASE para extraer diferentes tipos de información dependiendo de la consulta seleccionada en el panel de datos.

CUADRO 5.3: Descripción de los programas PHP que conforman la Web Service.

Capítulo 6

Casos de estudio

En este capítulo se describen los casos de estudio de la aplicación móvil SnorUNI, en ellos se muestra su funcionamiento, las funcionalidades que brinda al usuario y la información de respuesta obtenida.

6.1. Inicio de sesión

En esta sección, se describe las principales vistas que constituyen a las funcionalidades de iniciar, guardar y eliminar sesiones.

Vista de inicio de sesión de la aplicación

En la figura 6.1 se observa la vista de inicio de sesión de la aplicación móvil SnorUNI, sus principales funcionalidades son:

- Mostrar las sesiones guardadas en la aplicación.
- Agregar nuevas sesiones a la lista (clic en el icono *add* en el *Action bar*).
- Eliminar una sesión determinada (clic en el icono *delete* en el *Action bar*), la opción se activa solo si hay por lo menos una sesión guardada.
- Consultar por el manual de ayuda o información de la aplicación. (clic en el icono del menú *overflow* en el *Action bar*).

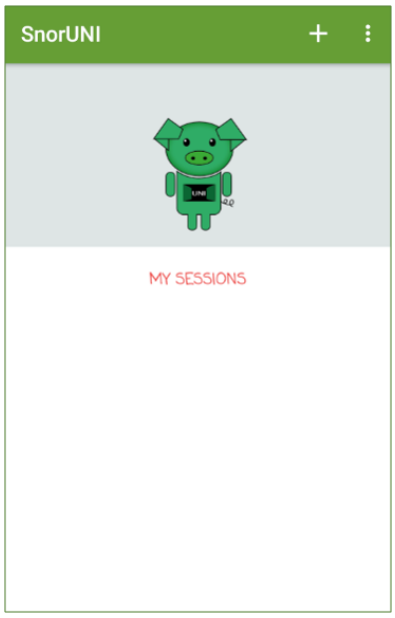

FIGURA 6.1: Vista principal de la aplicación SnorUNI. Fuente: Elaboración propia.

Agregar sesión

Esta funcionalidad permite al usuario iniciar nuevas sesiones en la aplicación y guardarlas para un fácil acceso en posteriores ocasiones. Para agregar e iniciar sesión, se tiene que hacer clic en el icono *add* del *Action bar*, se activará un formulario como el mostrado en la figura 6.2, donde se tiene que escribir la *IP* o *URL* del servidor a conectar, el usuario y la contraseña. Si los datos son correctos se inicia sesión normalmente accediendo a la aplicación; la próxima vez que el usuario cierre sesión podrá observar en la vista de inicio de sesión la información guardada como se ve en la figura 6.4.

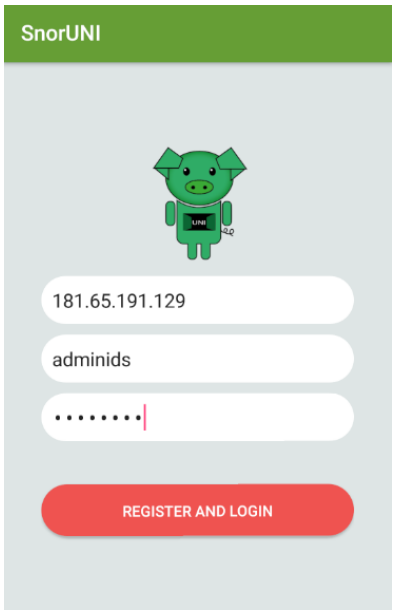

FIGURA 6.2: Formulario para agregar nueva sesión. Fuente: Elaboración propia.

Eliminar sesión permanentemente

Esta opción permite al usuario eliminar una sesión guardada en la aplicación. Para utilizar esta funcionalidad, el usuario tiene que hacer clic en el icono *delete* del *Action bar*, se activará el botón *clear* en cada uno de los *cards* de la lista, con ello el usuario puede seleccionar la sesión a eliminar.

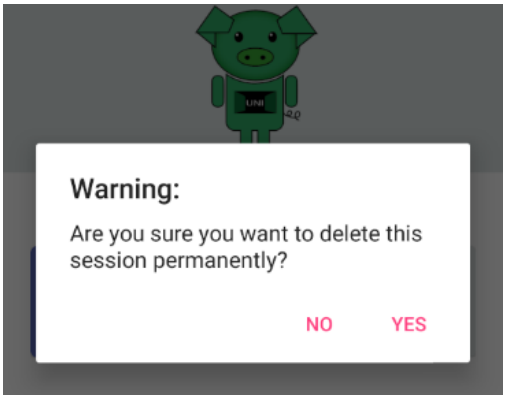

FIGURA 6.3: Mensaje de advertencia antes de eliminar una sesión. Fuente: Elaboración propia.

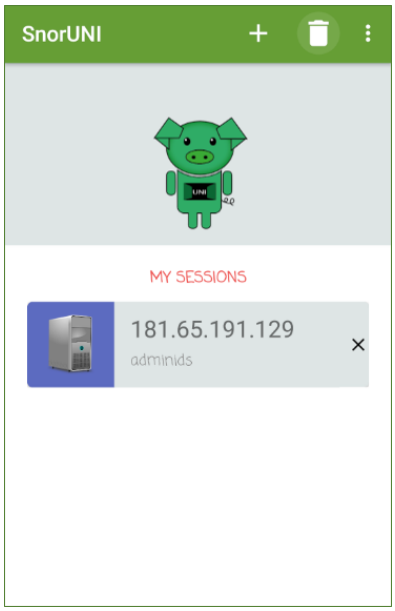

FIGURA 6.4: Vista de sesiones guardadas en la aplicación, con opción de borrado habilitada. Fuente: Elaboración propia.

Si se confirma el mensaje de advertencia mostrado en la figura 6.3, la sesión seleccionada será eliminada de la base de datos de la aplicación, así mismo, se eliminará automáticamente la suscripción a las notificaciones de ese servidor.

6.2. Información de ataques

En este punto se describen las vistas generales de la aplicación. Se muestra la forma de acceso, las herramientas disponibles y la información obtenida.

Ver usuarios del sistema

Esta vista permite al usuario observar la información de todos los usuarios registrados en la base de datos Snort y que tienen acceso la información, para acceder a esta vista hacer clic en la opción *System Users* del menú principal (*Navigation View*). Como se observa en la figura 6.5, los campos del resultado son el nombre de usuario, nombre de inicio de sesión (*user login*) y rol.

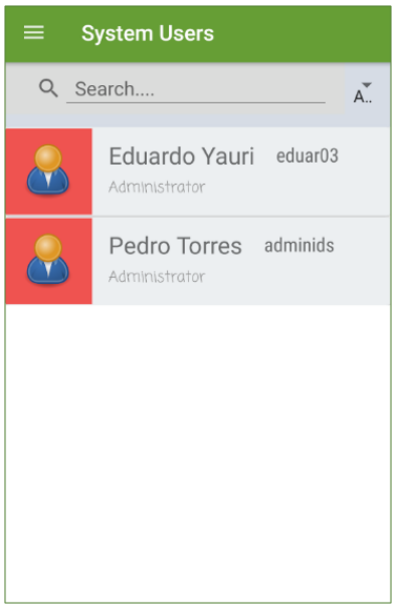

FIGURA 6.5: Usuarios del sistema en el servidor conectado.
Fuente: Elaboración propia.

Vista principal de alertas

Esta es una de las principales funcionalidades del sistema, dado que muestra las alertas registradas por Snort de forma concisa y resumida. Para acceder a esta vista, el usuario debe seleccionar la opción *Alert Log* en el menú principal. En la figura 6.6, se observa la información obtenida distribuida en columnas, las principales columnas son *attack* (nombre del ataque), *timestamp* (fecha) y *classification* (clasificación de ataque).

Esta y todas las vistas de información brindan al usuario las funcionalidades extras:

- Búsqueda de información por columnas, para ello seleccionar el nombre de la columna en el *spinner* superior derecho.
- Ordenamiento de filas por columnas, al hacer clic en las cabeceras de tabla con letras de color amarillo, se realizará un orden ascedente de la información en base a esa columna.

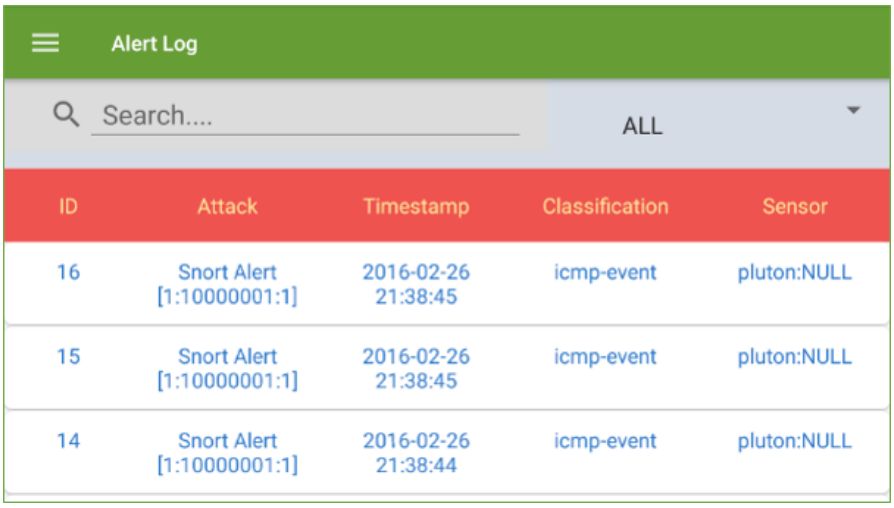

FIGURA 6.6: Alertas registradas por Snort. Vista Horizontal del dispositivo. Fuente: Elaboración propia.

Ver estadísticas del Sistema

Brinda al usuario un resumen completo de la cantidad de ataques por tipos, estos datos serán de utilidad en caso de hacer un reporte estadístico. Para acceder a esta vista el usuario tiene que seleccionar la opción *Statistyc* del menú principal. Las opciones de búsqueda y ordenamiento por columna también están disponibles como se puede observar en la figura 6.7.

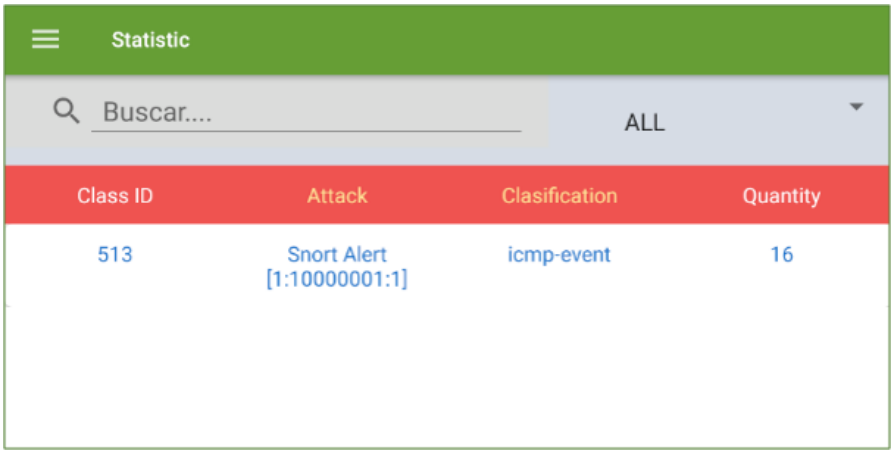

FIGURA 6.7: Número de ataques por tipo. Vista Horizontal del dispositivo. Fuente: Elaboración propia.

Ver información de alertas ICMP

Esta vista ofrece al usuario información relacionada al protocolo *ICMP* de las alertas de ese tipo. Para acceder a esta vista, el usuario tiene que selecionar la opción *ICMP protocol information* del menú principal. Como se ve en la figura 6.8 los campos devueltos son *attack* (nombre de ataque), *type* (tipo) y *cheksum*. Se puede hacer un ordenamiento por *ID* o por ataque.

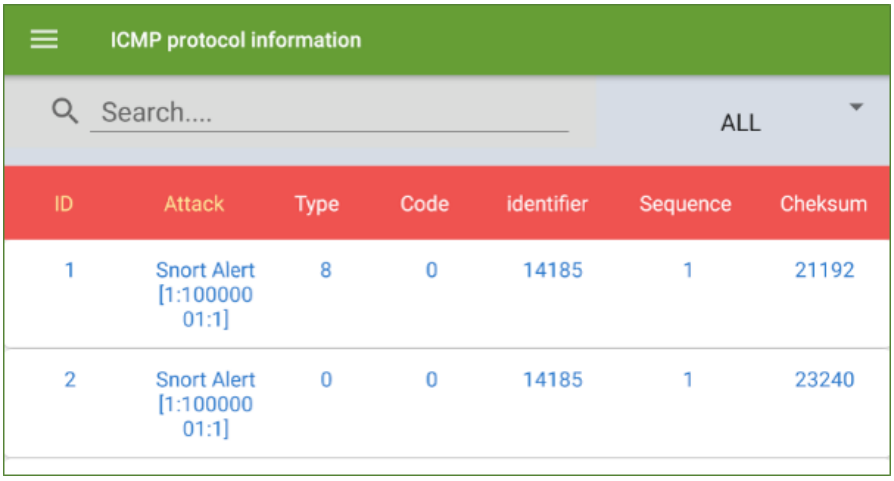

FIGURA 6.8: Información de ataques ICMP. Vista Horizontal del dispositivo. Fuente: Elaboración propia.

Ver información de alertas IP

Brinda al usuario información relacionada al protocolo *IP* de las alertas de ese tipo. Para acceder a esta vista el usuario tiene que seleccionar la opción *IP protocol information* del menú principal. En la figura 6.9 se observa que los campos principales campos devueltos son *attack* (nombre de ataque), *source_ip* (*IP* de origen), *destination_ip* (*IP* de destino) e *IP versión* (Versión de la IP).

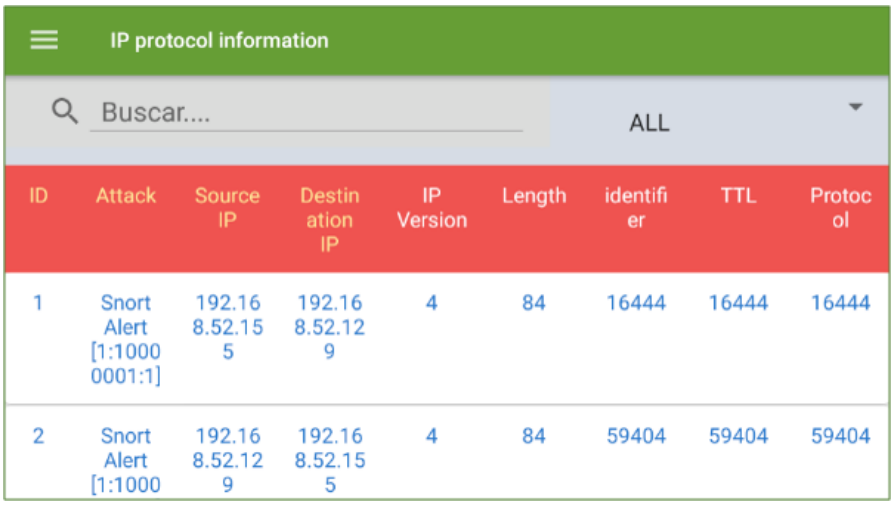

FIGURA 6.9: Información de ataques IP. Vista Horizontal del dispositivo. Fuente: Elaboración propia.

Ver información de alertas TCP

Esta funcionalidad permite al usuario ver información relacionada al protocolo *TCP* de las alertas de ese tipo. Para acceder a esta vista el usuario tiene que selecionar la opción *TCP protocol information* del menú principal. En la figura 6.10 se observa que los principales campos devueltos son *attack* (nombre de ataque), *source_port* (puerto *TCP* de origen) y *destination_port* (puerto *TCP* de destino).

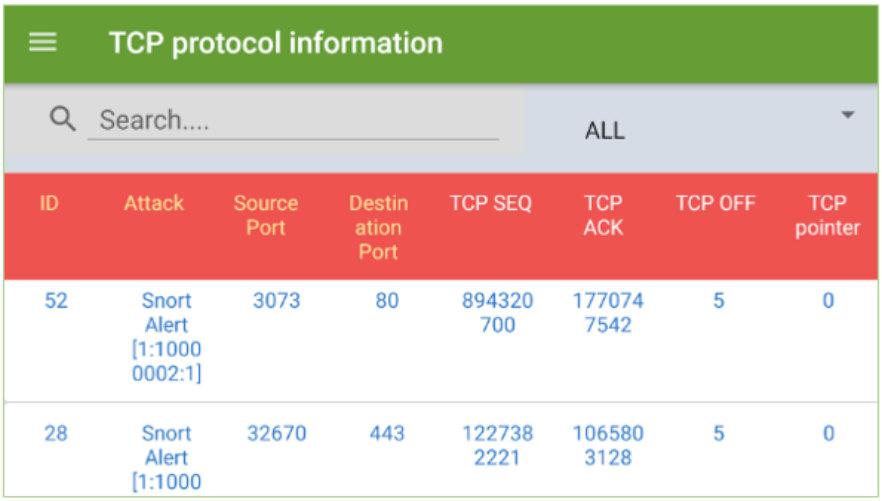

FIGURA 6.10: Información de ataques TCP. Vista Horizontal del dispositivo. Fuente: Elaboración propia.

Ver información de alertas UDP

Esta vista brinda al usuario información relacionada al protocolo *UDP* de las alertas de ese tipo. Para acceder a esta vista el usuario tiene sleecionar la opción *UDP protocol information* del menú principal. En la figura 6.11 se observa que los principales campos devueltos son *attack* (nombre de ataque), *source_port* (puerto *UDP* de origen) y *destination_port* (puerto *UDP* de destino).

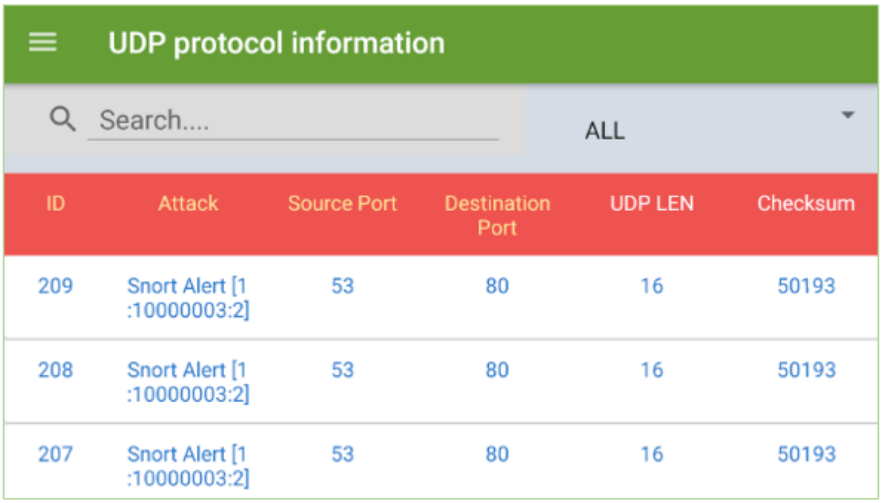

FIGURA 6.11: Información de ataques UDP. Vista Horizontal del dispositivo. Fuente: Elaboración propia.

6.3. Plugin BASE

En esta sección, se describen las vistas que conforman al plugin BASE. Dependiendo de la opción elegida en el panel principal, se tiene tres tipos básicos de vistas, estas se diferencian por las opciónes de acceso a la información que brindan.

Vista principal del plugin BASE

Permite acceder a la versión móvil del tablero de indicadores del *plugin* BASE. Para acceder a esta vista el usuario tiene que seleccionar la opción BASE del menú principal. Como se ve en la figura 6.12, esta vista brinda al usuario gran cantidad de opciones de vistas, las cuales puede ser:

- Alertas del día (*Today's Alerts*).
- 15 únicas alertas más recientes(*recent 15 unique alerts*).
- 15 puertos de origen más frecuentes (*most frequente source ports*).
- 15 puertos de destino más frecuentes (*most frequent destination ports*).
- 15 ips de origen más frecuentes (*most frequent 15 source ips*).
- 15 ips de destino más frecuentes (*most frequent 15 destination ips*).
- 15 últimos puertos de origen (*last source ports*).
- 15 últimos puertos de destino (*last destination ports*).
- 15 últimas alertas (*last 15 alerts*).
- Alertas de las últimas 24 horas (*last 24 hours alerts*).
- Alertas de las últimas 72 horas (*last 72 hours alerts*).
- 15 alertas únicas más frecuentes (*most frequent 15 unique alerts*).

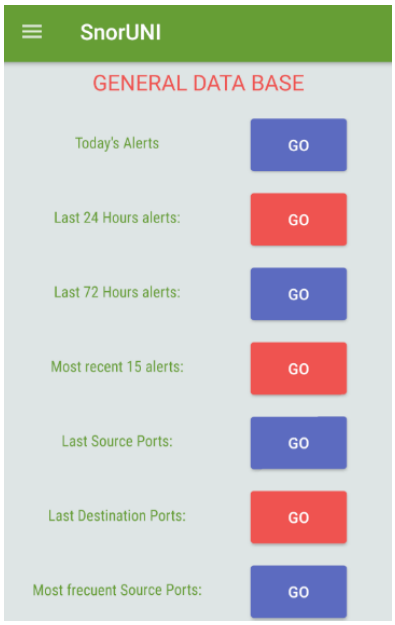

FIGURA 6.12: Tablero principal de BASE. Fuente: Elaboración propia.

Opciones de BASE tipo 1

En esta vista el usuario puede seleccionar una de las siguientes opciones:

- *Unique*: muestra las alertas únicas (sin repetición) de diferentes tipos.
- *Listening*: brinda información de Snort en escucha.
- *Source IP*: muestra información de todas las *IPs* de origen involucradas en ese caso.
- *Destination IP*: presenta información de todas las *IPs* de destino involucradas en ese caso.
- La última opción permite regresar a la vista anterior para seleccionar otro indicador del tablero principal de BASE.

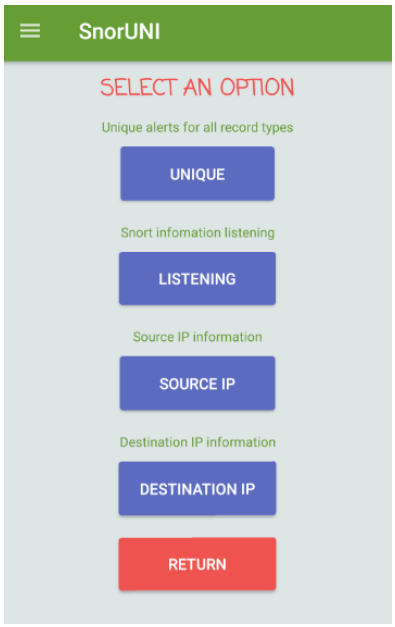

FIGURA 6.13: Opciones BASE tipo 1. Fuente: Elaboración propia.

Vista de BASE tipo 2

Esta funcionalidad brinda al usuario las siguientes opciones:

- *Any protocol*: muestra información de todos los protocolos sin excepción.
- *TCP information*: expone información del protocolo *TCP*.
- *UDP information*: brinda información del protocolo *UDP*.
- *ICMP information*: muestra información del protocolo *ICMP*.
- La última opción permite regresar a la vista anterior para seleccionar otro indicador del tablero principal de BASE.

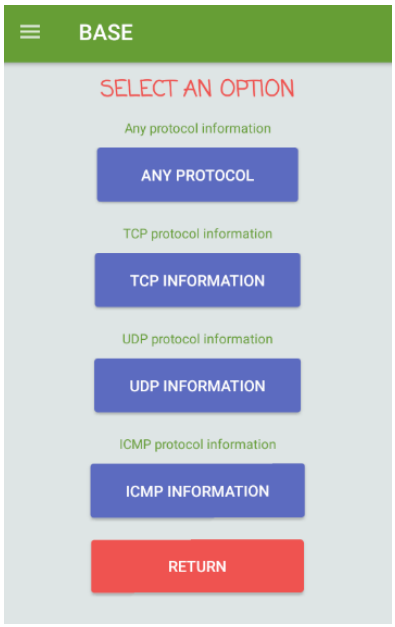

FIGURA 6.14: Opciones BASE tipo 2. Fuente: Elaboración propia.

Vista de BASE tipo 3

Esta vista, es muy similar a la del tipo 2, sin embargo aquí solamente permite seleccionar entre los protocolos *TCP* o *UDP*, también incluye el botón de regreso a la vista anterior para seleccionar un nuevo indicador del tablero principal.

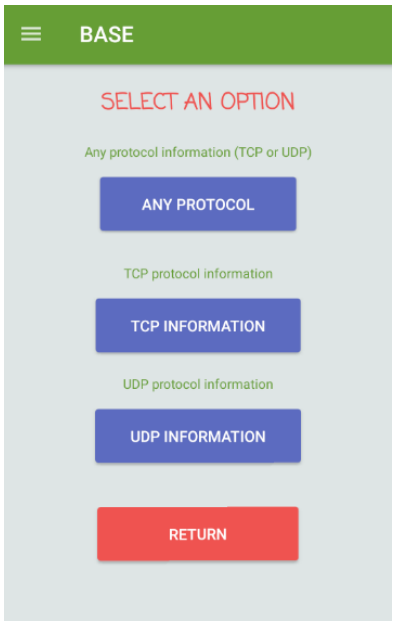

FIGURA 6.15: Opciones BASE tipo 3. Fuente: Elaboración propia.

6.4. Plugin SnoGE

La versión móvil del *plugin* SnoGE permite al usuario ver la localización geográfica de todas las *IPs* en un mapa de Google Earth. Para acceder el usuario debe seleccionar la opción *SnoGE* del menú principal.

En la figura 6.16 se observa las coordenadas de varias *IPs* (los *markers* rojos representan a los atacantes y los azules a los vulnerados) enlazadas por líneas que simbolizan el camino de los ataques entre *IPs*. Cabe resaltar que solamente se grafican las *IPs* públicas, dado que no es posible obtener las coordenadas *GPS* de las *IPs* privadas.

Adicionalmente al seleccionar cualquier *marker* automáticamente se abre otra vista con información de los ataques en los que está involucrada dicha *IP*, ver figura 6.17.

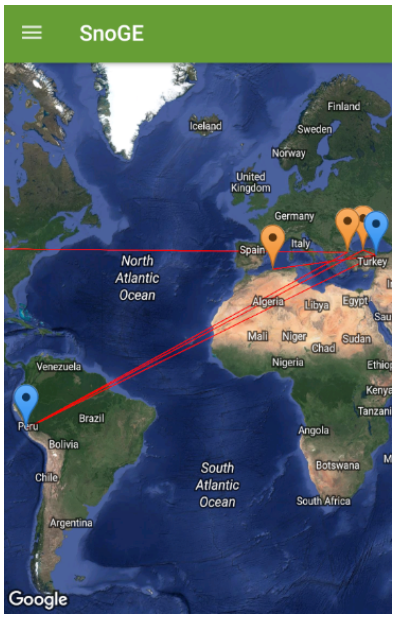

FIGURA 6.16: Mapa de SnoGE. Fuente: Elaboración propia.

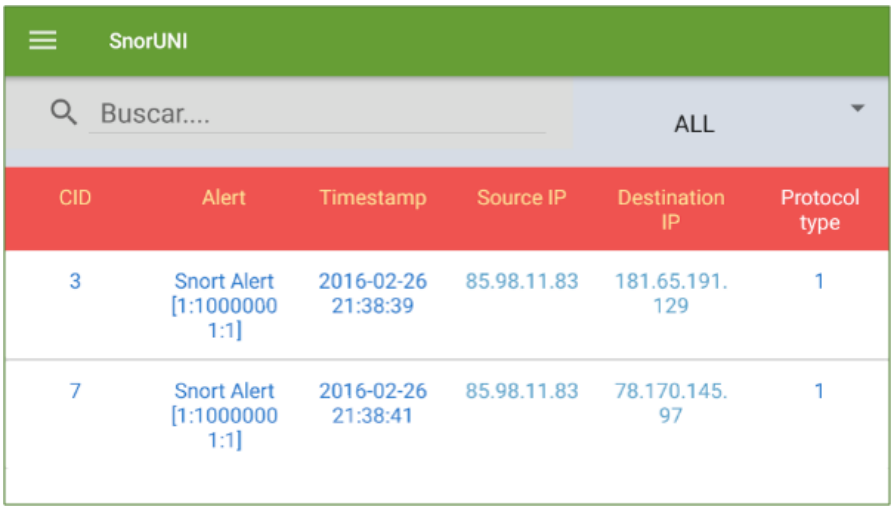

FIGURA 6.17: Información extra de una IP seleccionada del Mapa de SnoGE. Fuente: Elaboración propia.

6.5. Administración de Notificaciones

Esta funcionalidad permite al usuario, administrar la suscripción a las notificaciones de nuevas alertas enviadas desde los servidores locales a través de **Firebase Cloud Messaging**. Para acceder a esta vista el usuario debe seleccionar la opción *Notifications* del menú principal.

En la figura 6.18, se observa la vista de administración de notificaciones, cuando el usuario presione el botón *add* automáticamente se suscribirá al servicio de notificaciones del servidor actualmente conectado. De igual forma que la vista de inicio de sesión, todas las suscripciones se almacenan en la base de datos interna de la aplicación y se muestran en una lista. Para eliminar la suscripción presionar el *card* del ítem de la suscripción deseada, se mostrará un mensaje de alerta para confirmar la eliminación de la suscripción.

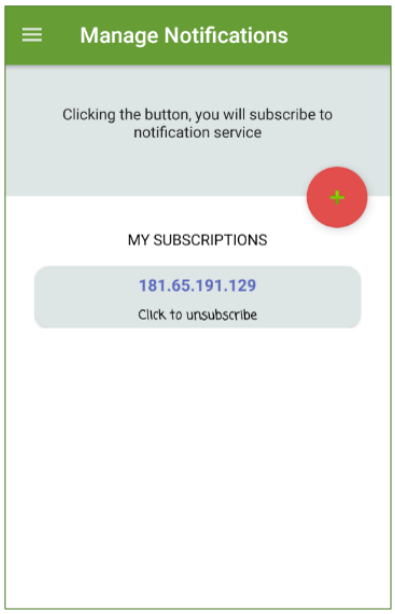

FIGURA 6.18: Administrador de Notificaciones de SnorUNI.
Fuente: Elaboración propia.

6.6. Información de ayuda

Esta vista muestra información de ayuda acerca de todas las funcionalidades de la aplicación, con el fin de facilitar el uso de los usuarios. Para acceder el usuario debe seleccionar la opción *Help* del menú principal.

En la figura 6.19 se muestra la vista de ayuda, está compuesto por varias pestañas en forma de *carrusel*, esto le permitirá al usuario desplazar los diferentes contenidos que brinda.

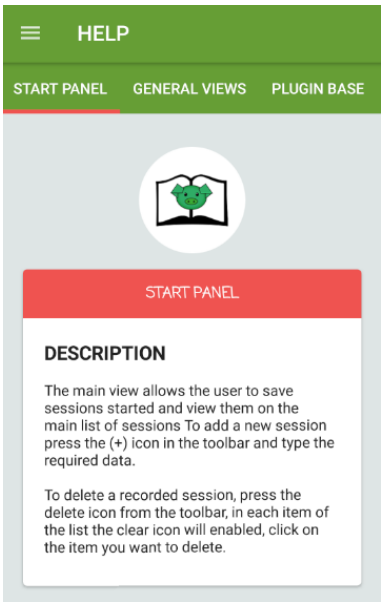

FIGURA 6.19: Menú Ayuda con pestañas deslizables. Fuente: Elaboración propia.

6.7. Información del dispositivo

Muestra información general sobre la aplicación SnorUNI, incluyendo la Licencia y la Autoría.

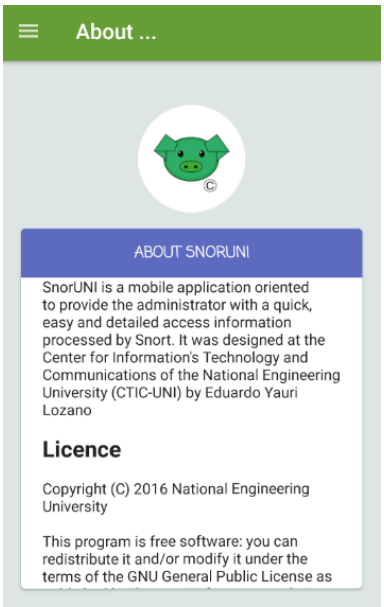

FIGURA 6.20: Información general de la Aplicación. Fuente: Elaboración propia.

Capítulo 7

Conclusiones y Trabajo a Futuro

Después de finalizar la implementación de la aplicación móvil SnorUNI, en el presente capítulo se mencionan las principales conclusiones obtenidas de este trabajo. Adicionalmente, se establecerán los posibles trabajos a futuro que se pueden desarrollar partiendo del sistema implementado.

7.1. Conclusiones

El desarrollo completo de SnorUNI tuvo 2 fases principales: (i) la primera fue la configuración y el desarrollo de componentes extras para el servidor Snort y; (ii) la segunda fué la implementación de la aplicación móvil en su totalidad.

Al finalizar la fase de configuración del servidor, se logró implementar y configurar las funcionalidades requeridas para realizar la conexión y el intercambio de información con la aplicación móvil. A continuación se describe de forma más detallada las funcionalidades desarrolladas:

- **Estudio y modificación de la base de datos:** se realizó el análisis de la base de datos utilizando ingeniería inversa con el proposito de conocer su estructura interna y su funcionamiento. A partir de ello, se modificó la base de datos creando tablas auxiliares de manera que la aplicación móvil pueda realizar las conexiones para la autenticación de usuarios y pueda extraer información de forma sencilla.

- **Actualización automática de la base de datos:** mediante los *scripts* implementados en *PHP* se crean las tablas auxiliares, se insertan los datos iniciales y utilizando el *daemon crontab* presente en casi todas las versiones de Sistema Operativo Linux, se actualiza periódicamente la información en la tabla auxiliar *snoruni_event*. De esa forma se brinda a los usuarios los datos más recientes.

- **Extracción de información:** se implementó el *web service*, mediante el cual la aplicación móvil extrae información de los eventos desde la base de datos. Este servicio permite acceder a una gran variedad de información entre ellas: alertas, información de protocolos, datos claves, etc. Previamente el usuario de la aplicación debe estar registrado en la base de datos dado que se utilizan estas credenciales para que la *web service* valide su ingreso al sistema.

- **Servicio de Notificaciones:** gracias al servicio de notificaciones, los usuarios reciben alertas de los últimos eventos registrados. Cada vez que el sistema actualiza la tabla *snoruni_event*, el servidor envía una notificación a Firebase Cloud Messaging con las *ID* de los últimos registros actualizados para que sean notificados todas las aplicaciones móviles registradas.

- **Cifrado de datos:** se configuró el servidor para que el *web service* intercambie información con la aplicación móvil utilizando el protocolo *HTTPS*, por lo cual la comunicación es cifrada empleando un certificado *SSL*. De esa forma se reduce el riesgo de interceptación y captura de información legible en el camino.

En la fase de desarrollo de SnorUNI, se lograron implementar las funcionalidades planteadas en el estado del arte y en los objetivos. De esa forma, la aplicación cumple con su objetivo principal de brindar información rápida y organizada a los usuarios en cualquier lugar y a cualquier hora, permitiendo potenciar la usabilidad de los *plugins* implementados.

A continuación se describe de forma más detallada las funcionalidades desarrolladas y los beneficios que brindarán a los usuarios:

- **Vistas de información general:** el usuario tiene acceso a los eventos registrados por el sistema, así como también a la información de los protocolos a los que pertenecen y un informe general de la cantidad de eventos de cada tipo. Con esta información el usuario puede elaborar informes de estado de la red, mejorar la seguridad en los protocolos de red y sobre todo tomar decisiones rápidas ante cualquier evento.

- **Plugin Base:** el usuario tiene acceso al tablero de indicadores similar al programa original en web, permitiendo al administrador establecer planes a futuro de prevención en base a los datos actuales obtenidos. Además se potencia la usabilidad de dicho *plugi*n al mejorar su accesibilidad.

- **Plugin SnoGE:** esta vista permite observar las localizaciones geográficas de las *IPs* de los atacantes y victimas en nuestra red. En base a ello es posible determinar información extra que pueda ayudar a su identificación.

- **Notificaciones:** esta es una de las principales funcionalidades de la aplicación dado que alertará al administrador de forma casi inmediata sobre una posible intrusión, pudiendo confirmar dicha alerta ingresando directamente a los registros de la aplicación. Esto permitirá tomar medidas de respuesta de forma rápida.

- **Manejo de sesiones:** esta funcionalidad facilitará el acceso de los usuarios a la aplicación previniendo en muchos casos inconvenientes por olvido o pérdida de credenciales, dado que todas las sesiones automáticamente quedan guardadas en la aplicación.

7.2. Trabajo a Futuro

En esta sección se plantearán los posibles trabajos a futuro que se puede desarrollar tomando como base el trabajo actual. Básicamente, las propuestas consisten en posibles mejoras o adición de nuevas funcionalidades que permitan potenciar la aplicación haciéndola más completa. Se plantean los siguientes:

- **Ampliación de las funcionalidades de BASE**
  Si bien se ha implementado el tablero de indicadores de BASE, la aplicación web brinda más funcionalidades las cuales sería factible tenerlas en su versión móvil. Entre ellas se puede considerar: gráfico de indicadores y barras contadoras de eventos.

- **Mejoramiento de herramientas extras**
  Las herramientas de búsqueda y ordenamiento que brindan las vistas de eventos serán mejoradas aumentando las opciones que brindan actualmente. Se plantea mejorar los filtros de búsqueda (personalizar busqueda por fecha de inicio y fin, tipo de protocolo) y los criterios de ordenamiento (ascendente, descendente, por letra inicial, por valor).

- **Limitar el acceso a las funcionalidades en base al rol**
  Por ahora solo se tiene un rol de usuario el cual es administrador. Se plantea crear más roles para que en base a estos se definan los accesos a determinadas vistas de la aplicación. Ello prevendrá que un usuario no autorizado tenga acceso a información confidencial.

7.3. Competencias adquiridas

En esta sección, se exponen las competencias adquiridas después del desarrollo de la presente tesis. Los conocimientos adquiridos involucran distintas áreas de aprendizaje, las cuales fueron necesarias para el desarrollo

del trabajo. Las compentencias estan relacionadas a los objetivos inicialmente planteados.

- Se adquirió la capacidad para el desarrollo de aplicativos en plataformas móviles utilizando el lenguaje de programación Android. De esta forma, se podrá hacer uso de los principales recursos que brinda este lenguaje, incluyendo manejo de *web services* y el uso de *Material Design Library* para mejorar las vistas de la aplicación.

- Con la finalidad de analizar y modificar la base de datos Snort, se adquirió la capacidad de realizar el procedimiento de ingeniería inversar a bases de datos. Esto incluye el diseño de diagramás *ER* y físicos, los cuales permitirán un mejor entendimiento de la estructura de cualquier base de datos.

- Gracias el proceso de desarrollo y configuración de la aplicación, se adquirió la capacidad para implementar y seguir la metodología ágil *SCRUM*.

- Se adquirio la capacidad de una correcta configuración y manejo de paquetes y herramientas de *software libre* en el Sistema Operativo GNU/Linux.

Bibliografía

[1] HACKMAGEDDON Information Security Timelines and Statistics. November 2016 cyber attacks statistics. http://www.hackmageddon.com/2016/12/21/november-2016-cyber-attacks-statistics/. Último acceso: 2017-01-03.

[2] Calyptix Security. Top 5 Cyber Attack Types in 2016 So Far http://www.calyptix.com/top-threats/top-5-cyber-attack-types-in-2016-so-far/. Último acceso: 2017-01-03.

[3] Github Repository.Swinedroid. https://github.com/Hainish/Swinedroid. Último acceso: 2017-01-03.

[4] ANDRODIR Apps.Swinedroid. http://androdir.com/com.legind.swinedroid.html. Último acceso: 2017-01-03.

[5] Manish Kumar and M. Hanumanthappa. Cloud based intrusion detectionarchitecture for smartphones. In *International Conference on Innovations in Information, Embedded and Communication systems (ICIIECS)*, 2015.

[6] Saman Zonouz, Amir Houmansadr, Robin Berthier, Nikita Borisov, andWilliam Sanders. Secloud: A cloud-based comprehensive and lightweightsecurity solution for smartphones. In *Computers and Security 37*, paginas 215–227, 2013.

[7] Akash Garg and Prachi Maheshwari.Performance Analysis of Snort-based Intrusion Detection System. In *3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2016.

[8] Github repository of barnyard2. Barnyard2 description. https://github.com/firnsy/barnyard2. Último acceso: 2017-01-02.

[9] BASE source code. Readme BASE. https://sourceforge.net/projects/secureideas/. Último acceso: 2017-01-27.

[10] SnoGE. Snoge features. https://leonward.wordpress.com/snoge/. Último acceso: 2017-01-04.

[11] Sguil FAQ. What is sguil? http://nsmwiki.org/Sguil_FAQ#What_is_Sguil.3F. Último acceso: 2016-12-27.

[12] Almulhen A. Network security engineering. In *Network SecurityEngineering*, pagina 15, 2008.

[13] Gonzales D. Seguridad en redes IP. In *Tesis de Doctorado en Informática*. Bellaterra: Universidad autónoma de Barcelona, página 234, 2003.

[14] Mira Jose. Introducción a los IDS. http://mural.uv.es/emial/informatica/html/IDS.html. Último acceso: 2017-01-04.

[15] Rojas R. Sistemas de Prevención de Intrusos. In *Tipping Point*, página 39.

[16] Retana R. Funcionamiento y aplicación actual. In *Sistema de Prevención deIntrusos (IPS)*, página 4.

[17] Virus Informatico. Que son los IDS. http://virusinformatico.net/conceptos-de-seguridad/que-son-el-ids-y-el-ips-cuales-la-diferencia-entre-ellos. Último acceso: 2014-12-27.

[18] Garcia J. In *Atáques Detección de ataques en red con Snort. Barcelona: depatamento de la información y comunicaciones*.

[19] Snortudenar. Snort sistemas de detección y prevención de intrusiones. https://snortudenar.wordpress.com/snort-2/. Último acceso: 2016-12-30.

[20] Jimenez Galindo. Diseño y optimización de un sistema de detección de intrusos hibrido. In *Título de Ingeniero en Informática-Universidad de Almería*, página 169, 2009.

[21] Gimenez M. Utilización de sistemas de detección de intrusos como elementos de seguridad perimetal. In *Título de Ingeniero enInformática-Universidad de Almería*, página 307, 2008.

[22] OK hosting. Metodologías de desarrollo de software. `http://okhosting.com/blog/metodologias-del-desarrollo-de-software/`. Último acceso: 2016-12-10.

[23] Software Testing Material. Agile scrum methodology in softwaredevelopment. `http://www.softwaretestingmaterial.com/agile-scrum-methodology/`. Último acceso: 2016-12-10.

[24] Yodiz organized empowered and productive teams. Agile softwaredevelopmentmethodology. `http://www.yodiz.com/blog/agile-software-development-methodology-definition-principle-of-agile/`. Último acceso: 2016-12-11.

[25] Android Developers. Android, the world's most popular mobile platform. `https://developer.android.com/about/index.html`. Último acceso: 2016-12-16.

[26] Android Studio. Conoce Android Studio. `https://developer.android.com/studio/intro/index.html?hl=es-419`. Último acceso: 2016-12-16.

[27] PHP. What is php?. `http://php.net/manual/en/intro-whatis.php`. Último acceso: 2016-12-16.

[28] PHP.What can php do?. `http://php.net/manual/en/intro-whatcando.php`. Último acceso: 2016-12-16.

[29] Apache. What is apache?. `https://wiki.apache.org/httpd/FAQ#What_is_Apache.3F`. Último acceso: 2016-12-10.

[30] Firebase. Firebase cloud messaging. https://firebase.google.com/docs/cloud-messaging/?hl=es. Último acceso: 2016-12-17.

[31] API de google Maps. Introducción a la API de Google Maps. https://developers.google.com/maps/documentation/android-api/intro. Último acceso: 2016-12-14.

[32] Google Developers. Adding Maps. https://developer.android.com/training/maps/index.html. Último acceso: 2016-12-15.

[33] AndroidSRC. Google maps android api v2. http://androidsrc.net/google-maps-android-api-v2-tutorial/. Último acceso: 2016-12-19

[34] Mysql home. Información General. https://dev.mysql.com/doc/refman/8.0/en/introduction.html. Último acceso: 2017-01-22.

[35] SQLite. About sqlite. https://www.sqlite.org/about.html. Último acceso: 2016-12-15.

[36] Android Development. Material Design.https://developer.android.com/design/material/index.html. Último acceso: 2017-01-26.

[37] Github Repository. Awesome-MaterialDesign. https://github.com/lightSky/Awesome-MaterialDesign. Último acceso: 2017-01-27.

[38] Holowczak.com. Database reverse engineering. http://holowczak.com/database-reverse-engineering/. Último acceso: 2016-12-26.

[39] Instant SSL. What is HTTPS?. https://www.instantssl.com/ssl-certificate-products/https.html. Último acceso: 2017-01-28.

[40] DigialOcean. How To Create a SSL Certificate on Apache for Ubuntu 14.04. https://www.digitalocean.com/community/tutorials/how-to-create-a-ssl-certificate-on-apache-for-ubuntu-14-04. Último acceso: 2017-01-30.

[41] IBM developerworks. Making linux an IPS device using snort. https://www.ibm.com/developerworks/community/blogs/58e72888-6340-46ac-b488-d31aa4058e9c/entry/august_8_2012_12_01_pm6?lang=en. Último acceso: 2017-01-05.

[42] HIGHSEC. Como montar nuestro IDS utilizando barnyard2. http://highsec.es/2013/12/como-montar-nuestro-ids-intrusion-detection-system-parte-ii-instalando-barnyard/. Último acceso: 2016-12-28.

[43] About Sguil. Sguil: The analyst console for network security monitoring. http://bammv.github.io/sguil/index.html. Último acceso: 2017-01-03.

[44] Github Repository. Pulled Pork. https://github.com/shirkdog/pulledpork. Último acceso: 2017-01-16.

[45] Snort downloads. Dumpig. https://www.snort.org/downloads. Último acceso: 2017-01-01.

[46] Sourceforge. Snor-IDMEF. https://sourceforge.net/projects/snort-idmef. Último acceso: 2017-01-14.

[47] Snort downloads. OfficeCAT. https://www.snort.org/downloads. Último acceso: 2017-01-01.

[48] O'really. SnortSam. https://www.safaribooksonline.com/library/view/intrusion-detection-systems/0131407333/0131407333_ch07lev1sec1.html. Último acceso: 2017-01-14.

[49] PacketFece Home. Overview. http://packetfence.org/about/overview.html. Último acceso: 2017-01-18.

[50] Snort downloads. IBlock. https://www.snort.org/downloads. Último acceso: 2017-01-01.

[51] Github Repository. Snorby Wiki. https://github.com/Snorby/snorby/wiki. Último acceso: 2017-01-10.

[52] DigialOcean. Install LAMP in Ubuntu 14.04. https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-14-04. Último acceso: 2017-02-10.

[53] DigialOcean. Install LAMP in Ubuntu 16.04. https://www.digitalocean.com/community/tutorials/como-instalar-linux-apache-mysql-php-lamp-en-ubuntu-16-04-es. Último acceso: 2017-02-10.

[54] Smart City Perú. SnorUNI. http://www.smartcityperu.org/snoruni. Último Acceso: 2017-03-10.

[55] DigialOcean. How To Create a SSL Certificate on Apache for Ubuntu 16.04. https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-apache-in-ubuntu-16-04. Último acceso: 2017-03-30.

Anexo A

Modos de ejecución y componentes importantes de Snort

En esta sección se estudiarán los principales modos de ejecución de Snort y se describirán detallamente sus componentes principales (*plugins*). Para cada modo de ejecución se especifican los comandos a utilizar y los resultados de la ejecución.

Modos de ejecución de Snort

a) **Sniffer Mode**

Ejecutar Snort en modo *Sniffer*, permite escuchar el tráfico de información que circula por la red y mostrar la información capturada a través de la pantalla del *terminal*. Dependiendo de los parámetros de ejecución es posible observar el contenido del paquete. Para ejecutar Snort en modo *Sniffer*, se adiciona el parámetro *"-v"*, adicionalmente si se desea mostrar en terminal los datos obtenidos el parámetro utilizado es *"-dv"*. Se ejecuta de la siguiente forma:

```
>>$ sudo snort -dv
```

En la figura A.1, se muestra un ejemplo de *sniffing* sobre una sub red *192.168.56.0*, el equipo con sistema operativo Windows e *IP 192.168.56.104*

envía paquetes al *broadcast*.

```
WARNING: No preprocessors configured for policy 0.
06/02-11:20:41.912468 192.168.10.159:137 -> 192.168.10.255:137
UDP TTL:128 TOS:0x0 ID:867 IpLen:20 DgmLen:96
Len: 68
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
06/02-11:20:42.322748 192.168.10.159:137 -> 192.168.10.255:137
UDP TTL:128 TOS:0x0 ID:868 IpLen:20 DgmLen:78
Len: 50
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
06/02-11:20:43.041301 192.168.10.159:137 -> 192.168.10.255:137
UDP TTL:128 TOS:0x0 ID:869 IpLen:20 DgmLen:78
Len: 50
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
06/02-11:20:44.371220 192.168.10.83:49389 -> 239.255.255.250:1900
UDP TTL:4 TOS:0x0 ID:14352 IpLen:20 DgmLen:125
Len: 97
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

FIGURA A.1: Ejecución de Snort en modo Sniffing: Fuente: Elaboración propia

b) **Packet Logger Mode**

Ejecutar Snort en modo *Packet Logger*, permite almacenar la información recolectada dentro de un archivo *log* para poder analizar la información posteriormente y en más detalle. Generalmente el archivo utilizado para almacenar la información de Snort es */var/log/snort*. Para activar este modo, se utiliza la opción *"- l"* seguido de la ruta del archivo donde se desea almacenar información, la información guardada es muy similar al resultado mostrado en la figura A.1. La ejecución del comando es de la siguiente forma:

```
>>$ sudo snort –dv –l archivo
```

La información es almacenada en archivos *"snort.log.xxxxxxx"* que son del tipo binario, es por ello que para visualizarlos se utiliza el comando snort junto con la opción *"–r"* de la siguiente forma:

```
>>$ sudo snort –r /var/log/snort/snort.log.xxxxx
```

c) **Network Intrusion Detection System (NDIS)**

Este es el modo más completo de ejecución de Snort. En este modo se utilizan las reglas definidas por el administrador para la detección de intrusos o amenazas. Todos los *IDS* basados en Snort se ejecutan en este modo, generando gran cantidad de información en algunos casos, la cual puede ser almacenada en *logs*, bases de datos o simplemente mostrados por el terminal. Para ejecutar Snort en este modo, se utiliza la opción *"-c"* seguido de la ruta del directorio donde se encuentran almacenadas las reglas:

```
>>$ sudo snort -c /var/snort/rules
```

En la figura A.2, se pueden observar los eventos registrados por Snort en modo *NIDS* en respuesta a la regla habilitada para captar paquetes del comando *ping* ejecutados hacia el servidor de Snort.

```
eduardo@ubuntuS14:~$ sudo /usr/local/bin/snort -A console -q -u snort -g snort -
c /etc/snort/snort.conf -i eth0
[sudo] password for eduardo:
06/02-14:49:58.807502  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
92.168.10.229 -> 192.168.10.105
06/02-14:49:58.809277  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
92.168.10.105 -> 192.168.10.229
06/02-14:49:59.809261  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
92.168.10.229 -> 192.168.10.105
06/02-14:49:59.809472  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
92.168.10.105 -> 192.168.10.229
06/02-14:50:00.808964  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
92.168.10.229 -> 192.168.10.105
06/02-14:50:00.809309  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
92.168.10.105 -> 192.168.10.229
06/02-14:50:01.810969  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
92.168.10.229 -> 192.168.10.105
```

FIGURA A.2: Paquetes detectados por Snort en modo NDIS.

Fuente: Elaboración propia

d) **Inline Mode**

Al ejecutar Snort en modo *inline*, éste trabajará como un *IPS*. Por definición los *IPS* no solo detectan las intrusiones a una red, también previenen los ataques eliminando o bloqueando intentos de intrusión. Los *IPS* toman acciones inmediatas sobre los paquetes de información sospechosos según se haya definido en las reglas. Generalmente las reglas implementadas para estos casos son *drop*, *reject* o *sdrop*.[41]

```
>>$ drop icmp any any -> $HOME_NET any (msg:"ICMP test";
sid:10000001; rev:001;)
```

En el código anterior se observa una regla *IPS* para eliminar todos los paquetes enviados hacia el servidor generados por el comando *ping*, se puede observar la sentencia *drop* a cambio de *alert*, sentencia definida en el *NDIS*.

En la figura A.3, se observa las detecciones hechas por Snort en modo *Inline*. Cada paquete que llega al equipo es eliminado automáticamente, ello se puede comprobar al observar el resultado *"Destination port unreachable"* al ejecutar *ping* por segunda vez mientras se está ejecutando Snort.

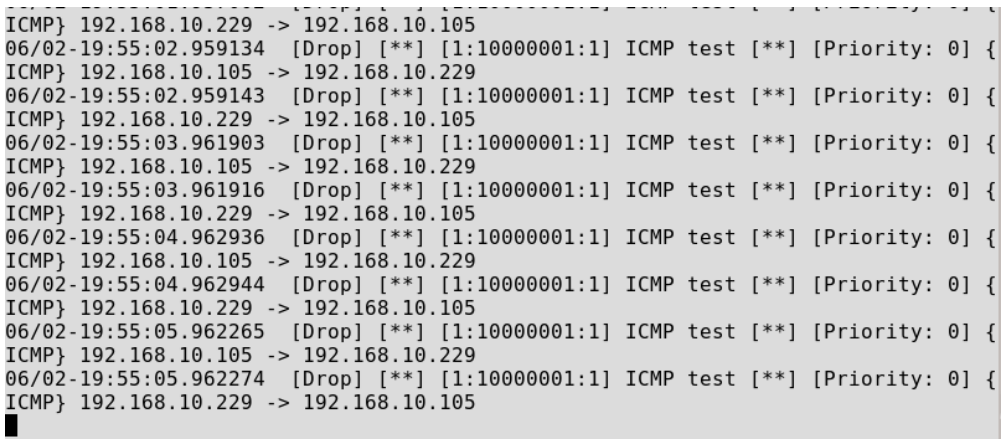

FIGURA A.3: Eventos detectados por Snort en modo inline.
Fuente: Elaboración propia.

Componentes importantes de Snort (Plugins)

Una de las características más resaltantes de Snort, es la posibilidad de agregarle componentes extras *(plugins)* ya sea para mejorar su rendimiento o para tener un mejor accesos a la información que nos brinda. Los *plugins*, cuyas funcionalidades están relacionadas a la aplicación móvil serán estudiados primero a profundidad:

a) **Barnyard2**

Es un intérprete que permite procesar la información almacenada en archivos binarios de salida. Estos archivos son generados por Snort en

formato *unified2*. Gracias a Barnyard2 es posible almacenar información en el disco de una manera eficiente y en varios formatos, por ejemplo *SQL* uno de los más conocidos. El proceso de almacenamiento se realiza de manera independiente, por tanto no afecta el funcionamiento de Snort. Barnyard2 tiene tres modos de funcionamiento: [8]

- ***Batch*:** se procesarán los archivos especificados previamente y luego se finaliza la ejecución. Se utiliza la opción *"– o"*.
- ***Continual mode*:** en este modo se procesarán archivos con nombres especificados por patrón definido. Cuando se termine de procesar los archivos, Barnyard2 quedará a la espera de más información. Se utiliza la opción *"– w"*.
- ***Continual mode / Bookmarking*:** en este modo de ejecución, Barnyard2 procesará archivos de forma continua utilizando un archivo guía denominado *"waldo file"*. En este archivo se especifica detalladamente la línea y el *log* del último registro procesado por Barnyard2, por tanto en el próximo inicio de ejecución procesará registros empezando desde ese punto específico. Para ejecutar esta opción se utiliza la opción *"– f"*.

En la figura A.4, se observa el proceso de funcionamiento de Barnyard2. Snort guarda los eventos detectados en archivos binarios y posteriormente Barnyard2 accede a estos archivos, los procesa y los almacena en una base de datos.

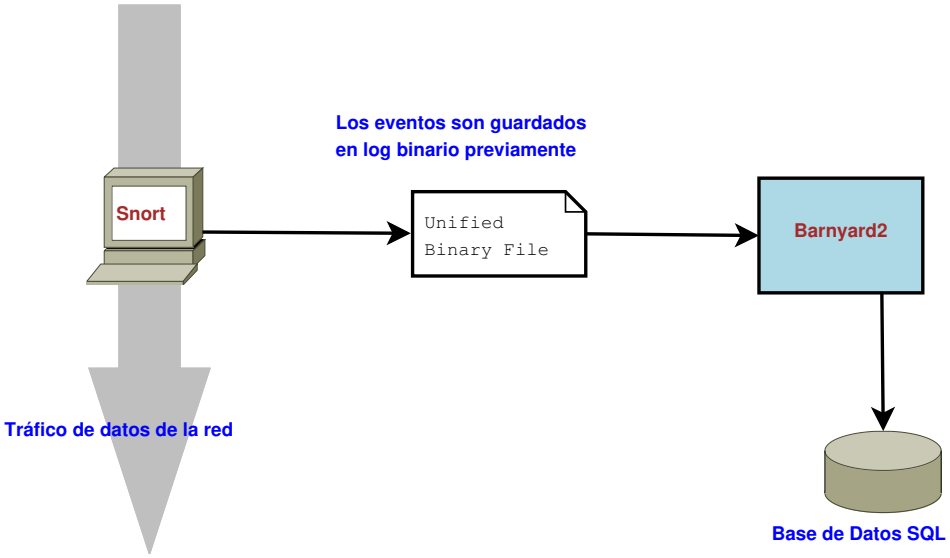

FIGURA A.4: Diagrama de funcionamiento Barnyard2. Fuente: HIGHSEC [42]

b) **BASE**

Es una interfaz web desarrollada en *PHP*, que permite acceder y gestionar la información generada por los *IDS*, cortafuegos, etc. BASE extrae información sobre alertas y eventos almacenados en las bases de datos. Puede soportar varios tipos y motores de bases de datos (*SQL* Server, Mysql, Orable, etc.).

BASE brinda al administrador las siguientes funcionalidades: categorizar las alertas en grupos, eliminar falsos positivos y exportar las alertas para utilizarlas como información de correos de notificaciones. Una de las características que hacen de BASE una de las principales interfaces gráficas, es su tabla de indicadores en la página principal de donde se puede extraer gran variedad de información y datos clave. La información de la tabla de indicadores puede ser utilizada en cualquier estudio o reporte posterior.[9]

En la figura A.5, se observa la vista del panel principal de BASE. En esta vista se observa información resumida de los eventos registrados, incluyendo su tabla de indicadores.

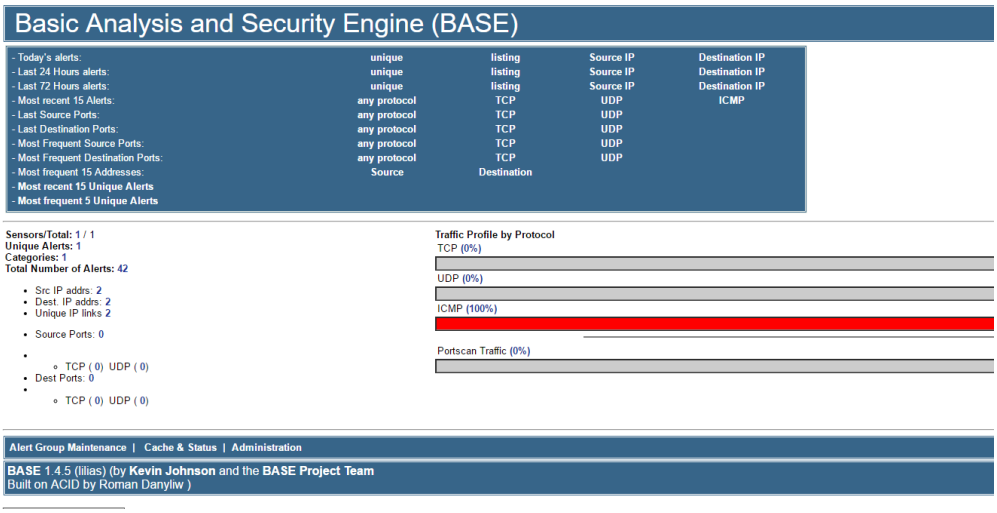

FIGURA A.5: Panel principal y Tablero de indicadores de BASE: Fuente: Elaboración propia.

c) **Sguil**

Es un sistema que brinda herramientas de monitorización en seguridad de redes. Fue desarrollado inicialmente por *Bamm Visscher*; actualmente *Steve Halligan* y *Bamm* son los dos principales desarrolladores. Al ser un software *open source*, toda la comunidad de desarrolladores contribuye continuamente a su mejora.[11]

El principal componente de Sguil, es su interfaz gráfica en la que se pueden observar los eventos en tiempo real e información sobre sesiones y paquetes capturados, es ideal para realizar un monitoreo completo de la red. Los datos recolectados pueden ser: eventos de Snort, eventos de Barnyard2, información de sesiones *TCP/IP* e información de programas de escaneo de red. Toda la información es recolectada por medio de sensores instalados en diferentes puntos de red.[43]

A diferencia de un sistema conjunto Snort y BASE o Snort y *ACID*, además de mostrar información de la base de datos a través del navegador, permite obtener información más detallada, por ejemplo: falsos positivos de eventos registrados, posibles ataques a futuro, etc.

En la figura A.6, se observa la arquitectura de Sguil conformada por los

sensores, el servidor y los clientes de administración. La información de la red, es captada por los sensores y es enviada al servidor central en donde son procesadas en busqueda de intrusiones. Finalmente, los clientes tienen acceso al servidor central para poder observar los resultados.

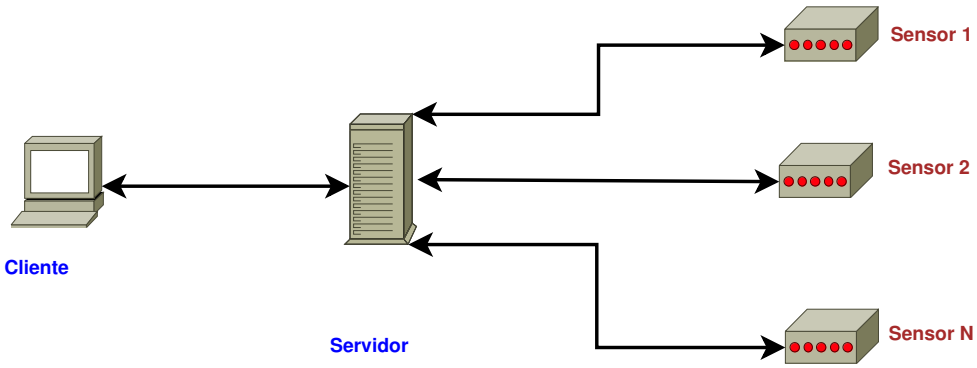

FIGURA A.6: Arquitectura Sguil. Fuente: Páguina de inicio Sguil FAQ. [11]

d) **SnoGE** Es una herramienta que procesa archivos de salida de Snort con formato *unified*, para luego representarlos en Google Earth. Es decir, permite graficar en un mapa de Google las *IPs* de los equipos vulnerados y atacantes. A partir de los eventos procesados se crea un archivo con extensión *kml* con toda la información lista para ser mostrada a través del navegador con la ayuda de Google Earth [10]. Las principales características de SnoGE son:

- Las últimas versiones soportan archivos binarios *unified* (*logs* de Snort).
- Actualización automática opcional para mostrar ataques registrados recientemente.
- Capacidad de uso multiusuario.
- Representación de puntos de localización por medio de barras (coloreado azul para ciudades y verde para países).

En la figura A.7, se observa el resultado de la ejecución de SnoGE después

de procesar un archivo *kml*. En el mapa de Google Earth se grafican y enlazan a través de conectores a todas las *IPs* involucradas en los eventos.

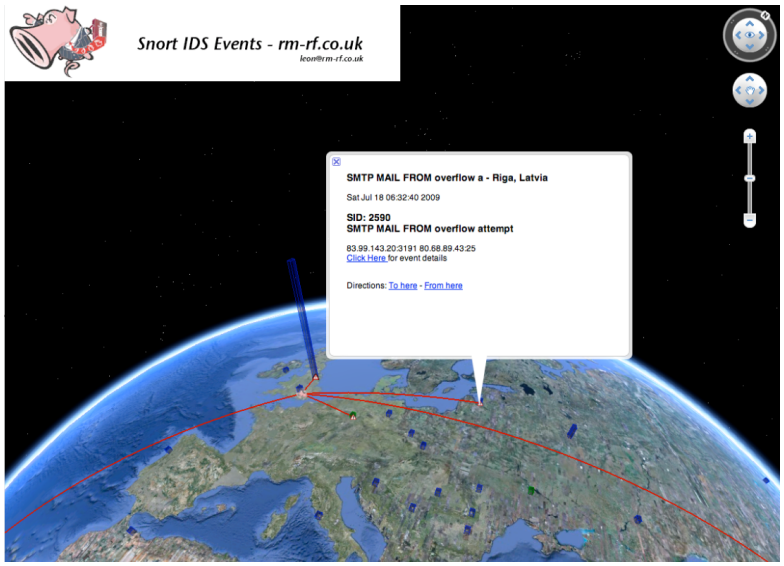

FIGURA A.7: Ploteo de coordenadas GPS de las IP registradas por Snort. Fuente: Página principal de SnorGE. [10]

Existen otros *plugins* que permiten agregar diversas funcionalidades a Snort aparte de los ya mencionados. Se realizará un estudio de los más desatacados de forma resumida:

- **PulledPork:** *script* desarrollado en Perl que permite actualizar las reglas de Snort de forma automática desde su repositorio central en la web. Este *plugin* también es de vital importancia dado que previene posibles inconvenientes por no actualizar las reglas oportunamente (filtración de ataques, identificación errónea de eventos, etc.). [44]

- **Dumpig:** es un detector de errores de gramática para reglas de Snort. Este *plugin* escanea cada una de las reglas de un archivo especificado y muestra un informe detallado de los errores que pueden existir. [45]

- **SnortIDMEF:** esta herramienta permite generar alertas de eventos en formato *IDMEF*. De esa forma es posible que Snort pueda intercambiar, compartir información e incluso interactuar con otras aplicaciones. [46]

- **OfficeCAT:** es una herramienta utilizada en línea de comandos, cuya función es procesar documentos de Office con el fin de detectar posibles *exploits* dentro del documento. Está disponible para los sistemas operativos GNU/Linux y Windows. [47]

- **SnorSam:** permite el bloqueo automático de *IP* en los *firewalls*. Esta herramienta tiene dos componentes: un componente instalado junto a Snort y un agente inteligente que se encuentra en el *firewall* o en un *host* cercano al *firewall* que se encarga de realizar las tareas de bloqueo. [48]

- **PacketFence:** es una herramienta *open source* que permite el control de acceso a la red, caracterizado por su continuo soporte y confiabilidad. Incluye un portal para el registro y administración de redes cableadas o inalámbricas. Además, es posible su integración con Snort y con la herramienta para escaneo de vulnerabilidades NESSUS. [49]

- **Iblock:** esta herramienta utiliza el archivo de alertas de Snort para obtener los *host* origen de un determinado ataque. En base a esa información, bloquea determinadas *IPs* utilizando *iptables* durante un tiempo determinado. Puede utilizar una lista blanca de *IPs* para prevenir el bloqueo un host confiable. [50]

- **Snorby:** es un *Front-End* para para *IDS* desarrollado recientemente. Las características principales de este *plugin* son la simplicidad, buena organización y poder de rendimiento. Puede ser implementado con Snort, Suricata o Sagan. [51]

Anexo B

Spring BackLogs

En esta sección se describe de forma resumida los *Sprint Backlogs* creados como resultado de seguir la metodología *SCRUM*. Cada uno representa una tarea a entregar como parte del desarrollo en conjunto de todo el sistema. Al final de cada *Sprint*, se muestran los siguientes resultados de su ejecución: tiempo de duración, resultado de la revisión y observaciones realizadas en caso de que se requiera nuevas modificaciones.

Primer Sprint

Actividad	Especificación	Días	Horas x día
Implementación de consultas a la base de datos para el inicio de sesión	Se instalará la base de datos Snort y el *plugin* BASE. Se diseña e implementan las consultas utilizadas por la *web service*. Para el inicio de sesión se utilizarán las credenciales guardadas en la tabla.	1	3

Implementación de Inicio de Sesión.	Se requiere que la aplicación cuente con un formulario de entrada y los métodos para el inicio de sesión utilizando la *web service* y credenciales de usuario para iniciar sesión.	2	4
Implementación del panel de sesiones.	Se requiere que la aplicación pueda guardar las sesiones iniciadas por el usuario. Dichas sesiones serán mostradas en una lista de forma similar al *App* Moodle.	3	3
Implementación de las funcionalidades complementarias en el inicio de sesión.	Se requiere que la aplicación pueda eliminar las sesiones guardadas. Se unirán todas las vistas implementadas para inicio de sesión.	3	3

CUADRO B.1: Cuadro de actividades del primer Sprint.

Resultados del *Sprint*:

- **Tiempo Estimado:** 9 días.
- **Estado:** Modificado.
- **Observaciones:** Cada uno de los puntos de este *Sprint* son mejoras funcionales de los puntos anteriores. Fueron planteados después de cada revisión y mención de propuestas de mejora.

Segundo Sprint

Actividad	Especificación	Días	Horas x día
Proceso de reingeniería de la base de datos Snort.	Se elaborarán dos diagramas para representar la estructura de la base de datos. En el primer diagrama *UML*, se plasma cada una de las tablas y campos de la base de datos. En el segundo diagrama *ER*, se representan las relaciones entre las tablas.	4	3
Implementación de las consultas de las vistas principales.	Se definen e implementan las consultas Mysql para obtener información de alertas, protocolos y otros.	2	4
Implementación de un menú de navegación.	La aplicación debe contar con un menú principal para el acceso a todas las vistas.	1	4
Implementación de las vistas generales.	La aplicación debe mostrar información general de alertas y protocolos. Las vistas deben estar constituidas por tablas y herramientas complementarias.	6	4

CUADRO B.2: Cuadro de actividades del segundo Sprint.

Resultados del *Sprint*:

- **Tiempo Estimado:** 13 días.

- **Estado:** Modificado.
- **Observaciones:** Las tablas deben tener las funcionalidades de buscar y ordenar datos. La adición de las nuevas funcionalidades se realizó progresivamente de acuerdo al desarrollo de la aplicación.

Tercer Sprint

Actividad	Especificación	Días	Horas x día
Implementación de consultas a la base de datos para BASE.	Se debe definir e implementar las consultas necesarias para obtener la misma información que la mostrada en el panel de BASE.	4	3
Implementación de la vista principal BASE y las vistas específicas.	La aplicación debe tener un panel principal para mostrar las opciones de BASE. Cada una de las opciones debe tener una vista específica.	6	4
Información de *IPs* utilizando servicios externos.	La aplicación debe contar con la opción de consultar información extra sobre las *IPs*.	2	4
Implementación de las herramientas buscador y ordenamiento.	La aplicación debe contar con un buscador y la opción de ordenar información por columnas.	4	4

CUADRO B.3: Cuadro de actividades del tercer Sprint.

Resultados del *Sprint*:

- **Tiempo Estimado:** 14 días.
- **Estado:** Modificado.
- **Observaciones:** El tercer y cuarto punto son mejoras realizadas después de cada revisión. La funcionalidad del cuarto punto, se generalizo para todas las vistas de información.

Cuarto Sprint

Actividad	Especificación	Días	Horas x día
Búsqueda de Servicios para obtener coordenadas *GPS* de una *IP*.	Se realizará una búsqueda de *web services* para obtener coordenadas *GPS* en base a una *IP* pública determinada.	2	1
Implementación del mapa SnoGE.	La aplicación debe tener un mapa donde se grafiquen las posiciones de todas las *IPs* registradas en la base de datos de Snort. (Solo *IPs* públicas).	2	4
Graficar conexión de ataques.	En el mapa de la aplicación se debe graficar los ataques realizados desde una *IP* atacante hacia una vulnerada.	3	4

Información extra.	Al presionar en los *markers* en el mapa, se debe mostrar información detallada de la *IP*.	2	4

CUADRO B.4: Cuadro de actividades del cuarto Sprint.

Resultados del *Sprint*:

- **Tiempo Estimado:** 9 días.
- **Estado:** Modificado.
- **Observaciones:** Los puntos 3 y 4 son mejoras realizadas al punto 2, planteadas después de cada revisión.

Quinto Sprint

Actividad	Especificación	Días	Horas x día
Configuración del servidor para uso de protocolo *HTTPS*.	El servidor debe utilizar el protocolo *HTTPS* para transferencia de información segura mediante cifrado de información.	1	3
Modificación de la aplicación para realizar conexiones seguras.	La aplicación debe conectarse al servidor mediante el protocolo seguro *HTTPS* y extraer información.	3	4

CUADRO B.5: Cuadro de actividades del quinto Sprint.

Resultados del *Sprint*:

- **Tiempo Estimado:** 4 días.
- **Estado:** Aceptado.

Sexto Sprint

Actividad	Especificación	Días	Horas x día
Configuración del servidor de Notificaciones Firebase.	Se creará una cuenta en Firebase Cloud message para configurar el servidor de notificaciones.	1	5
Implementación de clases y vistas para recibir notificaciones.	La aplicación móvil debe poder recibir notificaciones enviadas desde la consola Firebase.	4	4
Control de registro en función a temas.	La aplicación debe brindar al usuario la posibilidad de registro a un determinado servidor en función a un tema. El registro debe almacenarse en la memoria interna de la aplicación y mostrarse de forma similar al panel de inicio de sesión.	2	4
Implementación del servidor de envío.	Se desarrollará un *script* en *PHP*, para el envío de la notificación ante un evento en el servidor de Snort.	2	3

CUADRO B.6: Cuadro de actividades del sexto Sprint.

Resultados del *Sprint*:

- **Tiempo Estimado:** 9 días.
- **Estado:** Modificado.
- **Observaciones:** Cada uno de los puntos es una mejora de funcionalidades definidas después de cada reunión.

Septimo Sprint

Actividad	Especificación	Días	Horas x día
Creación de tablas auxiliares exclusivas para la aplicación.	En el servidor se deben crear tablas auxiliares para trabajar con la aplicación. La estructura de las nuevas tablas será similar a las tablas utilizadas por BASE.	2	4
Uso de *Material Design* en la aplicación	Se utilizarán los componentes: *sliders*, *cardview*, *float button* y desplazamientos.	6	4

CUADRO B.7: Cuadro de actividades del septimo Sprint.

Resultados del *Sprint*:

- **Tiempo Estimado:** 8 días.
- **Estado:** Modificado.
- **Observaciones:** Cada uno de los puntos es una mejora de funcionalidades del punto anterior, definido después de cada reunión.

Anexo C

Manual de Instalación y Configuración

En este apendice se describirán detalladamente el proceso de instalación y configuración de SnorUNI. Se siguen los siguientes pasos:

1. **Actualización del Sistema**
   Antes de instalar y descargar los paquetes principales, es necesario realizar la actualización del sistema para obtener la última versión de las librerías que soportarán el sistema. Ejecutar los siguientes comandos:

   ```
   >>$ sudo apt-get update
   >>$ sudo apt-get upgrade
   ```

2. **Instalación de LAMP**
   El servidor debe tener previamente instalado: Linux, Apache, MySQL y PHP *(LAMP)*. Si aún no se tienen instalados esos servicios, puede seguir el tutorial [52] para su instalación en *Ubuntu 14.04 LTS* o [53] para *Ubuntu 16.04 LTS*.

3. **Configuración de la base de datos**
   Crear el usuario snoruni en Mysql para la aplicación y asignarle permisos sobre la base de datos snort. Ejecutar los siguientes comandos:

   ```
   >>$ mysql -u root -p
   ```

```
mysql> CREATE USER 'snoruni'@'localhost'
IDENTIFIED BY '**********';
mysql> grant create, insert, select, delete,
update on snort.* to 'snoruni'@'localhost';
```

4. **Habilitar el protocolo HTTPS en Apache**

   Habilitar la transferencia segura de datos utilizando el protocolo *HTTPS*. Para ello se debe crear el certificado *SSL* en el servidor Apache siguiendo el tutorial [40] para *Ubuntu 14.04* o [55] para *Ubuntu 16.04*.

5. **Descargar la aplicación y los archivos del Servidor**

   Se debe descargar los archivos del Servidor y el *APK* de la aplicación. Para ello, ingresar al sitio web oficial del proyecto SnorUNI [54] e ir a la sección Descargas. Los archivos del servidor están comprimidos en un archivo *ZIP*, por ejemplo *snoruni_src.zip*.

6. **Creación de directorios y archivos**

   Copiar el archivo comprimido al directorio Home y luego descomprimir. Finalmente, se copian todos los archivos al directorio principal. Ejecutar los siguientes comandos:

```
>>$ cd <ruta_actual_del_zip>
>>$ cp snoruni_src.zip ~
>>$ cd ~
>>$ unzip snoruni_src.zip
>>$ sudo mkdir /var/www/html/snoruni
>>$ sudo cp –R ~/snoruni_src/*\\
 /var/www/html/snoruni
```

7. **Definir las credenciales**

   Editar el archivo *db_config.php* con las credenciales del usuario Mysql creadas en el tercer punto. Ejecutar los siguientes comandos:

```
>>$ cd /var/www/html/snoruni
>>$ sudo vim db_config.php
```

Ingresar el nombre de usuario, contraseña, nombre de la base de datos y nombre del servidor.

Adicionalente, agregar las credenciales al archivo de autenticación de usuarios *user_authenticate.php*.

```
>>$ sudo vim user_authenticate.php
```

Editar la linea 2 insertando el nombre de servidor, nombre de usuario, contraseña y nombre de la base de datos.

8. **Instalación de la librería Composer**

   Es necesario instalar la librería Composer para el envío de notificaciones. Las librerías descargadas deben estar en la misma carpeta que los scripts *update_dbtables.php* y *create_dbtables.php*. Por defecto están en el directorio snoruni. Ejecutar los siguientes comandos:

```
>>$ cd /var/www/html/snoruni
>>$ sudo apt-get update
>>$ sudo apt-get install curl php5-cli git
>>$ curl -sS https://getcomposer.org/installer |\\
sudo php --install-dir=/usr/local/bin\\
--filename=composer
```

9. **Instalar Firebase Cloud Messagge para PHP**

   Se tiene que instalar la librería paragraph1/php-fcm para enviar las notificaciones desde el *script*. Las librerías descargadas deben estar en la misma carpeta que los scripts *update_dbtables.php* y *create_dbtables.php*. Por defecto están en el directorio snoruni. Ejecutar los siguientes comandos:

```
>>$ cd /var/www/html/
>>$ sudo chmod -R 777 snoruni/
>>$ cd snoruni/
>>$ composer require paragraph1/php-fcm
>>$ sudo chmod -R 755 snoruni/
```

10. **Crear las tablas auxiliares**

    Para crear las tablas auxiliares en la base de datos, se debe ejecutar el *script create_dbtables.php* de la siguiente forma:

    ```
    >>$ cd /var/www/html/snoruni
    >>$ /usr/bin/php create_dbtables.php
    ```

    Si la ejecución fue exitosa, se mostrara en el terminal el mensaje *"Tables created and Data inserted"*.

11. **Crear los archivos logs para actualizaciones**

    Se creará un archivo *log* que guarde la información de los procesos de actualización de las tablas auxiliares. Se ejecutan los siguientes comandos:

    ```
    >>$ cd /var/www/html/snoruni
    >>$ sudo mkdir logs
    >>$ sudo chmod –R 755 logs
    >>$ sudo touch logs/updatelogs.txt
    >>$ sudo chmod 777 logs/updatelogs.txt
    ```

12. **Creación del Tópico de Firebase Cloud Message**

    El tópico de Firebase Cloud Message es uno de los elementos principales del sistema de notificaciones, dado que indica el canal de envío de información. Para la instalación actual es necesario crear un tópico distintivo al cual los dispositivos móviles se registren para recibir las notificaciones enviadas desde el servidor Snort.

    Para crear el tópico, el usuario tiene que ingresar a la vista de administración de notificaciones de la aplicación móvil y presionar el botón *add*. Inmediatamente se mostrará un mensaje indicando que la aplicación ha sido registrada al servicio de notificaciones como se muestra en la figura B.1. Firebase Cloud Messaging creará un nuevo tópico para el sistema en un periodo de tiempo de 5 minutos a 24 horas. Los tópicos creados son cadenas con el patrón: "*channelsnorUNI_<server IP>*".

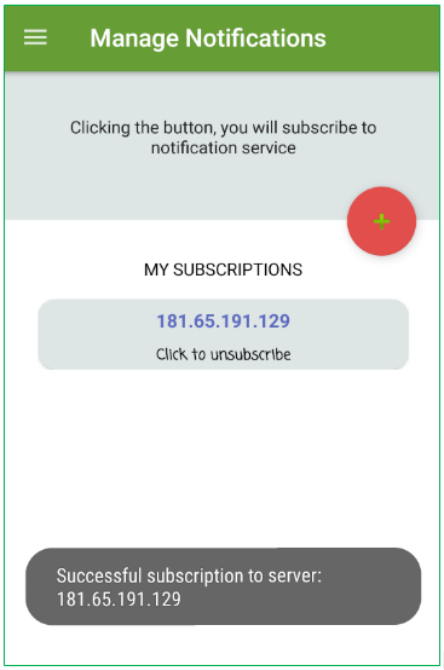

FIGURA C.1: Registro al servicio de notificaciones. Fuente: Elaboración propia.

Finalmente, editar las lineas 26 y 27 del script *update_dbtables.php* insertando el *IP* del servidor. Se ejecutan los siguientes comandos:

```
>>$ cd /var/www/html/snoruni
>>$ sudo vim update_dbtables.php
//-------------------------------------------------------------
25. $message = new Message();
26. $message->addRecipient(new \\
    Topic('channelsnorUNI_<IP>'));
27. $message->setData(array('iporigen' => '<IP>',
    'titulo'=>'SnorUNI Alert','mensaje'=>'New
    events registered with ID: '.$cids));
//-------------------------------------------------------------
```

13. **Creación del Certificado SSL para Apache**

    Para que la comunicación entre el servidor y la aplicación móvil sea de forma segura, se tiene que habilitar el uso del protocolo *HTTPS* en Apache

dentro del Servidor Snort.

Por tanto, se creará un certificado *SSL* con firma propia. Para realizar la configuración, seguir las instrucciones definidas en [40] para *Ubuntu 14.04 LTS* o [55] para *Ubuntu 16.04 LTS*.

14. **Automatización del servicio**

    Las tablas auxiliares deben ser constantemente actualizadas, por tanto el script *update_dbtables.php* debe ejecutarse periódicamente. La automatización de la ejecución se realiza utilizando el *daemon* crontab de GNU/Linux, definiendo un intervalo de 60 segundos entre cada actualización. Se ejecutan los siguientes comandos.

```
>>$ sudo crontab -e
//-------------------------------------------------------
 * * * * * /usr/bin/php /var/www/html/\\
 snoruni/update_dbtables.php >> /var/www/html/\\
 snoruni/logs/updatelogs.txt
//-------------------------------------------------------
```

    El intervalo de ejecución puede ser modificado dependiendo de las necesidades del sistema y administrador.